VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

BAKALÁŘSKÁ PRÁCE

Brno, 2019 Radek Král

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV VÝKONOVÉ ELEKTROTECHNIKY A ELEKTRONIKY
DEPARTMENT OF POWER ELECTRICAL AND ELECTRONIC ENGINEERING

PROGRAMOVATELNÉ LOGICKÉ AUTOMATY PRO VÝUKU
PLC SYSTEMS IN EDUCATION PROCESS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **Radek Král**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. Dalibor Červinka, Ph.D.**
SUPERVISOR

BRNO 2019

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní obor **Silnoproudá elektrotechnika a elektroenergetika**
Ústav výkonové elektrotechniky a elektroniky

Student: Radek Král ***ID:*** 195670
Ročník: 3 ***Akademický rok:*** 2018/19

NÁZEV TÉMATU:

Programovatelné logické automaty pro výuku

POKYNY PRO VYPRACOVÁNÍ:

1. Vzájemně srovnejte parametry a příslušná programovací prostředí zvolených programovatelných logických automatů a zvolte nejvhodnější typ pro účely výuky.
2. Dle pokynů vedoucího navrhněte několik laboratorních úloh.
3. Vytvořte příslušné laboratorní návody

DOPORUČENÁ LITERATURA:

[1] PATOCKA, Miroslav. Magnetické jevy a obvody ve výkonové elektronice, mericí technice a silnoproudé elektrotechnice. 1. vyd. V Brne: VUTIUM, 2011, 564 s. ISBN 978-80-214-4003-6.

[2] Vorel P., Patocka M., Prumyslová elektronika, Vydání 1., skriptum FEKT VUT Brno, 2007

[3] SKALICKÝ, J. Navrhování elektrických pohonu. Navrhování elektrických pohonu. Brno: VUT FEKT, 2002.

Termín zadání: 4.2.2019 ***Termín odevzdání:*** 22.5.2019

Vedoucí práce: Ing. Dalibor Červinka, Ph.D.
Konzultant:

doc. Ing. Petr Toman, Ph.D.
předseda oborové rady

Abstrakt

Bakalářská práce se zabývá programovatelnými logickými automaty pro výuku. Práce je členěna do šesti kapitol. V první kapitole jsou popsány základní programovací jazyky, a také jejich rozlišení s grafickým náhledem. Druhá kapitola se zabývá programovacím prostředím GX Works2 od Mitsubishi Electric. Kapitoly třetí a čtvrtá popisují programovací prostředí SoMachine od výrobce Schneider Electric a Automation Studio od B&R. V páté kapitole této bakalářské práce jsou porovnána jednotlivá programovací prostředí z pohledu běžného uživatele. V závěrečné části je navrženo devět laboratorních úloh. Ke každé laboratorní úloze je vytvořen laboratorní návod a u většiny je příklad logického programu a názorné vizualizace.

Klíčová slova

PLC, Mitsubishi, GX Works, Schneider, SoMachine, B&R, Automation Studio

Abstract

The bachelor thesis deals with programmable logic controllers for teaching. The thesis is divided into six chapters. The first chapter describes the basic programming languages as well as their resolution with graphical preview. The second chapter deals with programming environment GX Works2 from Mitsubishi Electric. Chapters three and four describe the SoMachine programming environment from Schneider Electric and Automation Studio from B&R. The fifth chapter of this bachelor thesis compares individual programming environments from the perspective of a common user. Nine laboratory tasks are proposed in the final part. For each lab task, a lab tutorial is created, and for most, an example of a logical program and visualizations.

Keywords

PLC, Mitsubishi, GX Works, Schneider, SoMachine, B&R, Automation Studio

Bibliografická citace:

KRÁL, Radek. *Programovatelné logické automaty pro výuku* [online]. Brno, 2019 [cit. 2019-05-07]. Dostupné z: https://www.vutbr.cz/studenti/zav-prace/detail/117522. Bakalářská práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav výkonové elektrotechniky a elektroniky. Vedoucí práce Ing. Dalibor Červinka, Ph.D.

Prohlášení

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

V Brně dne: **20. května 2019**

……
podpis autora

Poděkování

Děkuji vedoucímu bakalářské práce Ing. Daliborovi Červinkovi, Ph.D. za účinnou metodickou, pedagogickou a odbornou pomoc a další cenné rady při zpracování mé bakalářské práce.

Obsah

Seznam obrázků 8
Úvod 9
1. Programovatelný logický automat 10
    1.1 Jazyk seznamu instrukcí (IL) 11
    1.2 Jazyk strukturovaného textu (ST) 11
    1.3 Jazyk funkčního blokového schématu (FBD) 12
    1.4 Jazyk příčkového diagramu (LD) 12
    1.5 Sekvenční funkční diagram (SFC) 13
2. Mitsubishi Electric – PLC řady MELSEC 14
    2.1 Programovací prostředí GX Works2 15
3. Schneider Electric – SoMachine pro programování PLC 17
    3.1 SoMachine – Logic Builder 19
    3.2 SoMachine – Vijeo-Designer 20
    3.3 SoMachine – SoMachine Basic 22
4. B&R – programovací prostředí Automation Studio 24
5. Srovnání jednotlivých typů programovacích prostředí 25
6. Návrh laboratorních úloh 26
    6.1 Laboratorní úloha 1: Spouštění motoru přepínáním Y/D 26
    6.2 Laboratorní úloha 2: Garážová vrata 27
    6.3 Laboratorní úloha 3: Lakovna 29
    6.4 Laboratorní úloha 4: Pásové dopravníky 31
    6.5 Laboratorní úloha 5: Automatizované vrtací rameno 34
    6.6 Laboratorní úloha 6: Regulace intenzity osvětlení 37
    6.7 Laboratorní úloha 7: Regulace teploty v nádržce 38
    6.8 Laboratorní úloha 8: Regulace otáček motoru a kontrola teploty vinutí 39
    6.9 Laboratorní úloha 9: Řízení krokového motoru 41
Závěr 44
Literatura 45
Seznam zkratek 47

Seznam obrázků

Obr. 1-1: Příklad jazyka IL [4] 11
Obr. 1-2: Příklad jazyka ST 12
Obr. 1-3: Příklad jazyka FBD [2] 12
Obr. 1-4: Příklad jazyka LD [2] 13
Obr. 1-5: Příklad jazyka SFC [2] 13
Obr. 2-1: FX3GE-24MT/DSS [7] 14
Obr. 2-2: Analogové vstupy a analogový výstup [8] 15
Obr. 2-3: GX Works2 16
Obr. 3-1: M238 TM238LDD24DT [11] 17
Obr. 3-2: Centrální obrazovka SoMachine 18
Obr. 3-3: Obrazovka po vytvoření nového projektu 18
Obr. 3-4: Logic Builder 19
Obr. 3-5: Vijeo-Designer 21
Obr. 3-6: SoMachine Basic – Konfigurace 22
Obr. 3-7: SoMachine Basic – Programování 23
Obr. 4-1: Automation Studio 24
Obr. 6-1: Ukázka programu v LD pro laboratorní úlohu 1 27
Obr. 6-2: Ukázka programu v LD pro laboratorní úlohu 2 29
Obr. 6-3: Ukázka programu v SFC a vizualizace pro laboratorní úlohu 3 31
Obr. 6-4: Ukázka vizualizace pro laboratorní úlohu 4 33
Obr. 6-5: Ukázka části programu v LD pro laboratorní úlohu 4 34
Obr. 6-6: Ukázka vizualizace pro laboratorní úlohu 5 36
Obr. 6-7: Ukázka programu v SFC a ST pro laboratorní úlohu 5 36
Obr. 6-8: Ukázka části programu v FBD pro laboratorní úlohu 8 40
Obr. 6-9: Ukázka vizualizace pro laboratorní úlohu 8 40
Obr. 6-10: Ukázka vizualizace pro laboratorní úlohu 9 42
Obr. 6-11: Ukázka části programu pro laboratorní úlohu 9 42
Obr. 6-12: Experimentální pracoviště pro laboratorní úlohu 9 43

ÚVOD

Elektronická zařízení a jejich uplatnění se v současné době šíří do všech oborů lidské činnosti. Programovatelné logické automaty jsou – mimo jiné také v průmyslových oblastech – již určitým standardem, jelikož snižují náklady a celkově optimalizují i výrobní proces. Účelem bakalářské práce je srovnání jednotlivých typů programovatelných logických automatů, neboť v současné době je plánováno jejich uplatnění ve výuce.

Hlavním cílem bakalářské práce jsou programovatelné logické automaty, které se dají využít ve výuce. V návaznosti na hlavní cíl jsou stanoveny následující cíle, a to nastudovat dokumentaci ke třem PLC jednotkám, dále pak vzájemné srovnání jejich schopností, parametrů a příslušných programovacích prostředí a zvolit nejvhodnější typ, a také navržení několika laboratorních úloh se zvoleným typem programovacího prostředí a napsání laboratorních návodů k těmto navrženým úlohám.

Bakalářská práce je uspořádána do šesti kapitol. První kapitola poskytuje náhled do teorie zabývající se danou problematikou a je zpracována na základě studia odborné literatury. Je zde vymezen jazyk seznamu instrukcí, jazyk strukturovaného textu, jazyk funkčního blokového schématu, dále pak jazyk příčkového diagramu i sekvenční funkční diagram. Programovací prostředí od jednotlivých výrobců jsou podrobněji popsána v kapitole druhé, třetí a čtvrté. Pátá kapitola se zabývá srovnáním jednotlivých typů programovacích prostředí. Jednotlivé laboratorní návody i s ukázkami řídícího programu k navrženým úlohám jsou v kapitole šesté.

1. PROGRAMOVATELNÝ LOGICKÝ AUTOMAT

Programovatelný logický automat neboli Programmable Logic Controller (dále jen PLC) nebo také PAC (Programmable Automation Controller) je programovatelný řídící systém, který slouží k řízení průmyslových, technologických systémů a procesů v reálném čase. Současná PLC neřeší pouze logické úlohy, ale lze s nimi řídit i spojité veličiny pomocí PID regulátorů. Použitím řídícího systému PLC lze dosáhnout optimalizace a synchronizace procesu i při měnících se podmínkách. Běžné PLC obsahuje procesorovou jednotku, paměť (systémovou i uživatelskou), interface pro komunikaci s programátorem a obsluhou nebo pro zapojení PLC do sítě, kde může komunikovat s dalšími PLC, periferie vstupní (vstupy) i výstupní (výstupy). Na vstupní periferie jsou přivedeny signály z řízeného procesu binární (zapnuto/vypnuto) nebo analogové např. teplota, tlak, hladina. Na periferie výstupní jsou také připojeny binární či spojité akční prvky řízeného procesu. Procesorová řídící jednotka realizuje soubor instrukcí, které jsou dány programovým algoritmem, jenž je uložen do paměti PLC a je cyklicky vykonáván. Podle stavů na vstupech ovlivňuje stavy výstupů. Paměťový prostor je rozdělen na dvě části. První část tvoří systémová paměť, která slouží pro uživatelské registry, čítače a časovače a dále pro komunikační, časové a jiné systémové proměnné. Druhá část je pro uložení uživatelského programu (programového algoritmu neboli PLC programu) a nazývá se uživatelská paměť [1]. Jakým způsobem se programuje, je dáno změnou způsobu, přizpůsobení a vývinem programovacích jazyků PLC. Dle [2] norma ČSN EN 61131-3 definuje pět programovacích jazyků:

- jazyk seznamu instrukcí (IL);
- jazyk strukturovaného textu (ST);
- jazyk funkčního blokového schématu (FBD);
- jazyk příčkového diagramu (LD);
- sekvenční funkční diagram (SFC).

Někteří výrobci nabízí i možnost programovat v jiných jazycích. Většinou v jednotlivých PLC není k dispozici těchto všech pět programovacích jazyků. Základní programovací pojmy jsou společné všem výrobcům, ale rozdíly v adresaci I/O

(Input/Output), instrukčních sadách a organizaci pamětí znamenají, že programy nejsou kompatibilní mezi různými výrobci [3].

1.1 Jazyk seznamu instrukcí (IL)

Textový jazyk IL (Instruction List), jazyk pokynů nebo také jazyk instrukcí připomíná assembler. Na každém řádku je instrukce, jak lze vidět na obrázku 1-1 [4], a může být připsán komentář. Negace, podmíněnost a nepodmíněnost instrukce skoků, návrat, volání a priorita jsou vyjadřovány pomocí modifikátorů [2].

```
      U(
      U(
      UN    "First_Scan"              M50.7
      SPBNB _00c
      L     100
      T     "MaxEngINT".Umid          DB5.DBW4
      SET
      SAVE
      CLR
_00c: U     BIE
      )
      SPBNB _00d
      L     "MaxEngINT".Umid          DB5.DBW4
      ITD
      T     "MaxEngDINT".Umid         DB6.DBD8
      SET
      SAVE
      CLR
```

Obr. 1-1: Příklad jazyka IL [4]

1.2 Jazyk strukturovaného textu (ST)

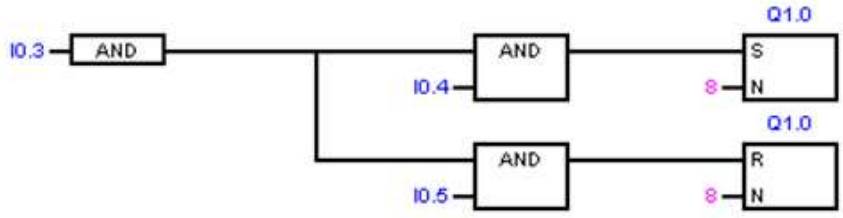
Textový jazyk ST (Structured Text) dle [2] má kořeny v programovacích jazycích Pascal a C a je považován za vyšší a výkonný jazyk. Je vhodným nástrojem pro definování funkčních bloků, které se poté mohou použít v libovolném programovacím jazyku. Je definováno několik příkazů (přiřazení, návrat, vyvolání funkce apod.), které jsou oddělené středníkem (ukončení aktuálního příkazu) a je možnost psát několik příkazů na jeden řádek. Příklad takto napsané části programu je na obrázku 1-2.

```
    PROG2 :=FALSE;
    PROG3 :=FALSE;
    PROG4 :=FALSE;
    START := FALSE;
    pov_prog1 := FALSE;
    pov_prog2 := FALSE;
    pov_prog3 := FALSE;
    pov_prog4 := FALSE;

ELSE
    Vyber:= TRUE;                   // bliká, že čeká na výběr programu
    stiskSTART := FALSE;

    IF (STOP = FALSE AND PROG1 = TRUE AND PROG2 = FALSE AND PROG3 = FALSE AND PROG4 = FALSE) THEN   // program 1
        stiskSTART := TRUE;         // bliká, že čeká na start
        Vyber := FALSE;

        IF START = TRUE THEN            // po stisku START běží program 1
                    stiskSTART := FALSE;
                    pov_prog1 := TRUE;
                    pov_prog2 := FALSE;
                    pov_prog3 := FALSE;
                    pov_prog4 := FALSE;
```

Obr. 1-2: Příklad jazyka ST

1.3 Jazyk funkčního blokového schématu (FBD)

Grafický jazyk FBD (Function Block Diagram) vyjadřuje chování funkčních bloků, funkcí a programů. V tomto jazyce jsou používány standartní funkční bloky (AND, OR, NAND, NOR apod.), časovače, čítače komunikační i speciální bloky podle potřeby, jak lze vidět na ukázce programu (obr. 1-3 [2]). Různí výrobci častokrát nabízejí poněkud odlišné sady funkčních bloků [2].



Obr. 1-3: Příklad jazyka FBD [2]

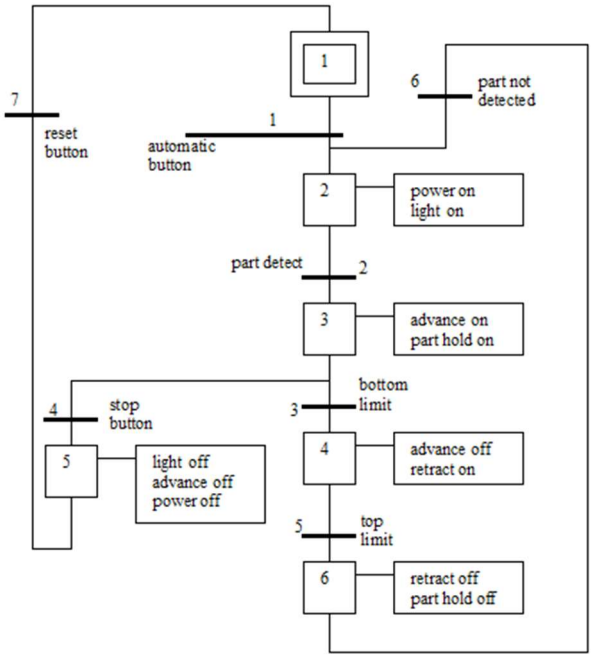
1.4 Jazyk příčkového diagramu (LD)

Grafický jazyk LD (Ladder Diagram) nebo také jazyk kontaktních schémat, je založen na znázornění reléové logiky. V jazyku LD je v programovém prostředí znázorněna síť, která je vlevo i vpravo ohraničena svislými napájecími sběrnicemi. Mezi ně se umísťují příčky s kontakty (spínací, rozpínací apod.), cívky, funkční bloky a funkce. Příčky mohou být rozvětveny, mohou být vodorovné, ale i svislé, jak je patrné na obrázku 1-4 [2].

Obr. 1-4: Příklad jazyka LD [2]

1.5 Sekvenční funkční diagram (SFC)

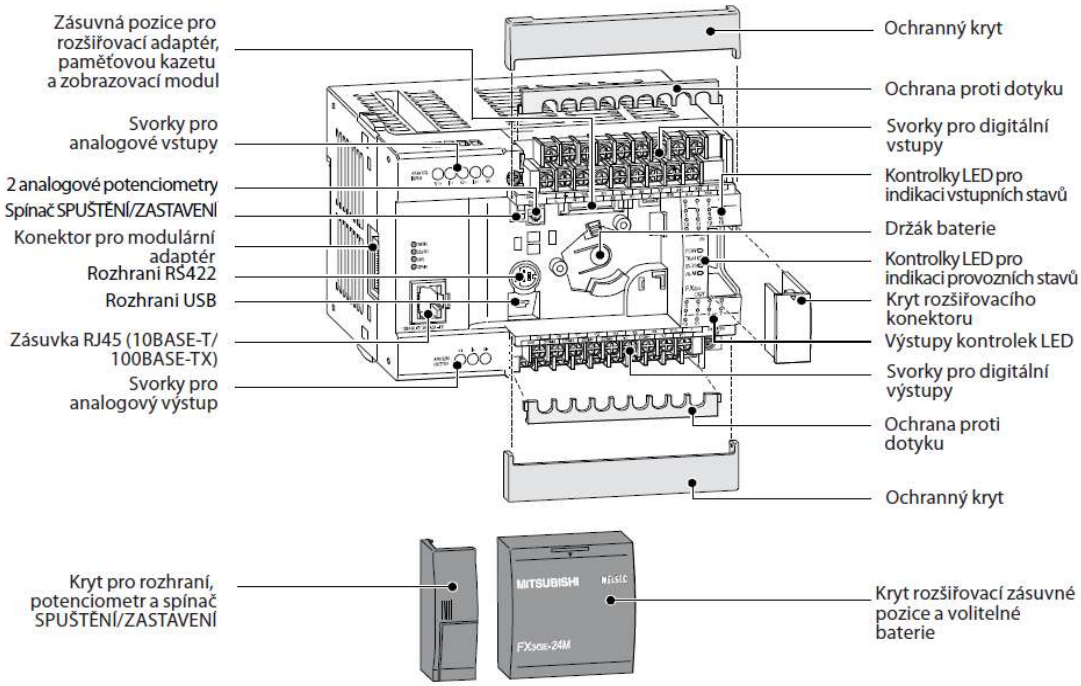
Jazyk SFC (Sequential Function Chart) znázorňuje sekvenční chování řídícího programu a je odvozen ze symboliky Petriho sítí. Zachovává přehled o chování celkového programu a rozkládá úlohu řízení na zvládnutelné části. SFC obsahuje kroky a přechody. Přechod je závislý na podmínce, která musí být splněna, aby byl ukončen předcházející krok a započat krok následující. Každý jednotlivý krok představuje stav, ve kterém se řízený systém aktuálně nachází. Tento jazyk umožňuje i větvení programu a souběh několika programových větví, které jsou následně synchronizovány [2]. Příklad programu v jazyce SFC je na obrázku 1-5 [2].



Obr. 1-5: Příklad jazyka SFC [2]

2. MITSUBISHI ELECTRIC – PLC ŘADY MELSEC

Mitsubishi Electric nabízí programovatelné kontroléry MELSEC několika sérií (MELSEC iQ-R, iQ-F, Q, L, F, QS/WS, A). Ve střední třídě v sérii MELSEC-F je kompaktní model typu FX3GE, který lze rozšířit pomocí modulárních a kompaktních rozšiřovacích jednotek až na maximální počet 256 vstupů a výstupů. Umožňuje zpracování rychlých vstupních pulzů pomocí integrovaných rychlých čítačů s frekvencemi 60 kHz a 10 kHz. Tyto rychlé vstupy jsou používány například na zpracování přerušení. Disponuje také integrovanými pulzními výstupy pro frekvence 2–100 000 Hz k ovládání krokových motorů. V této sérii je používána paměť RAM/EEPROM s kapacitou až 32 000 kroků PLC programu, zabudované hodiny reálného času a pomocí dvou sériových rozhraní je zajištěna komunikace s počítačem [5] [6].



Obr. 2-1: FX3GE-24MT/DSS [7]

Na obrázku 2-1 [7] je znázorněna základní jednotka FX3GE-24MT/DSS, která musí být napájena zdrojem 24 V DC. Disponuje 14 integrovanými digitálními vstupy, 10 integrovanými digitálními výstupy a dvěma potenciometry. Dále má k dispozici dva analogové vstupy a jeden analogový výstup [5]. Na analogové vstupy je možnost přivést

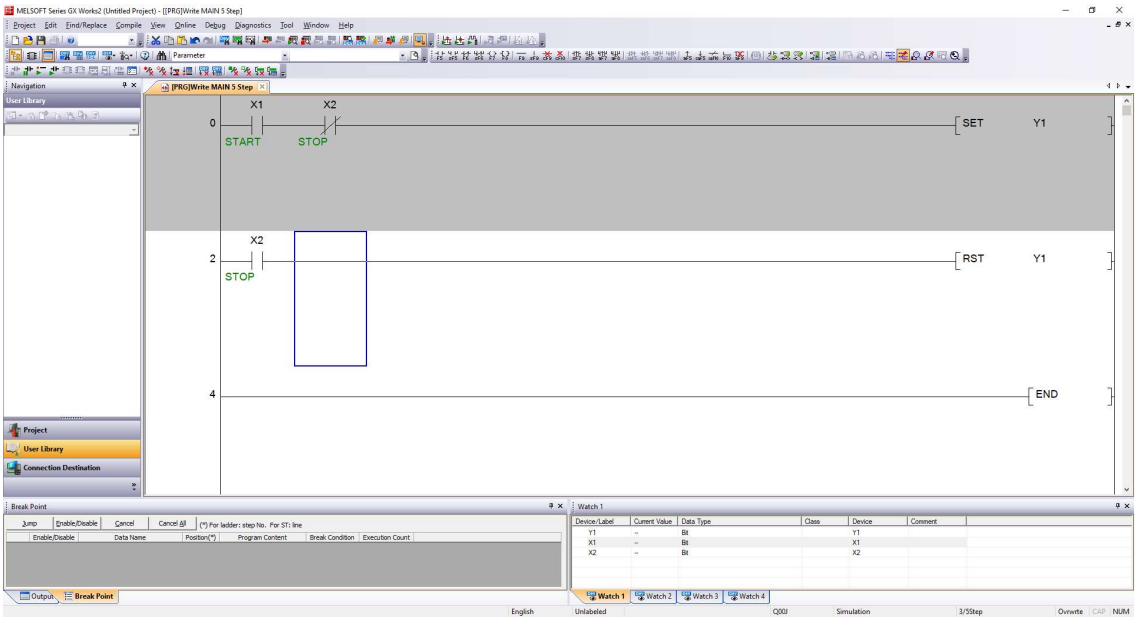
napěťový signál 0–10 V DC nebo proudový signál 4–20 mA. Analogový výstup může být opět napěťový 0–10 V DC nebo proudový 4–20 mA [8]. Analogové vstupy a analogový výstup jsou zobrazeny na obrázku 2-2 [8]. Tato základní jednotka také disponuje indikací vstupů a výstupů prostřednictvím LED kontrolek, připojovací svorkovnicí pro připojení vstupů, výstupů a napájení. Do zásuvné pozice lze zasunout paměťovou kazetu nebo základní jednotku rozšířit o zvláštní moduly. Pro komunikaci a programování má k dispozici dvě integrovaná sériová rozhraní (USB, RS422) a integrované rozhraní pro síť Ethernet, která může propojovat i vícero PLC [5].

Obr. 2-2: Analogové vstupy a analogový výstup [8]

2.1 Programovací prostředí GX Works2

Programovací prostředí GX Works2, vyvinuté společností Mitsubishi electric, je nástupce předchozího programovacího prostředí GX Developer a slouží k vytváření řídících programů pro kontroléry od téže společnosti. Po spuštění GX Works2 je nutné založit nový projekt. Při zakládání nového projektu je možnost založit jej ve dvou typech: *Simple Project* a *Structured Project*. Dále je třeba vybrat pro jakou sérii PLC a jaký konkrétní typ PLC je tento projekt vytvářen a vybrat programovací jazyk, jakým bude psán řídící program. Konkrétní typ PLC a programovací jazyk se dají v průběhu vytváření projektu editovat. U projektu typu *Simple Project* je na výběr ze dvou programovacích jazyků: LD a SFC. U druhého typu projektu *Structured Project* jsou na

výběr tři programovací jazyky: ST, IL a FBD. Po založení projektu je vhodné zkontrolovat, zda-li je spojení správně nakonfigurováno. To se provádí na levé straně v kartě *Navigation – Connection Destination*. Po výběru *Connection1* se zobrazí okno, ve kterém je možnost vybrat způsob komunikace s daným kontrolérem a odzkoušet ji tlačítkem *Connection Test* testovacím signálem. Společně s kartou *Connection Destinaton* lze na tomto místě zobrazit také karty *User Library* a *Project*. V kartě *Project* lze vidět celou strukturu projektu a z této karty je nejrychlejší způsob otevření potřebných oken a záložek. Pro psaní řídícího programu je nezbytné přepnout programovací prostředí do režimu *Write Mode* ikonou v pravé horní části, přes záložku *Edit – Ladder Edit Mode – Write Mode* nebo klávesovou zkratkou *F2*. V tomto prostředí existují další pracovní režimy – *Read Mode*, *Monitor Mode* a *Monitor* sloužící převážně k pozorování chování programu. V horní části je přehled základních logických prvků, které lze vložit do řídícího programu. Vkládání složitějších funkcí lze provést dvojím kliknutím na vybranou větev a korektním zápisem dané funkce nebo korektní zápis lze dohledat ikonou *Help*, následně vybrat požadovanou funkci a tento korektní zápis vytvoří samotný program. Pakliže je řídící program hotový, je třeba jej zkompilovat v záložce *Compile – Build/Rebuild All*, nezkompilované větve řídícího programu mají pro přehlednost šedě zbarvené pozadí dané větve, jak je patrné na obrázku 2-3. Pokud program nenahlásí žádné chyby, je možné tento řídící program otestovat simulací (*Debug – Start/Stop Simulation*) nebo rovnou nahrát do PLC (*Online – Write to PLC…*) [9] [10].



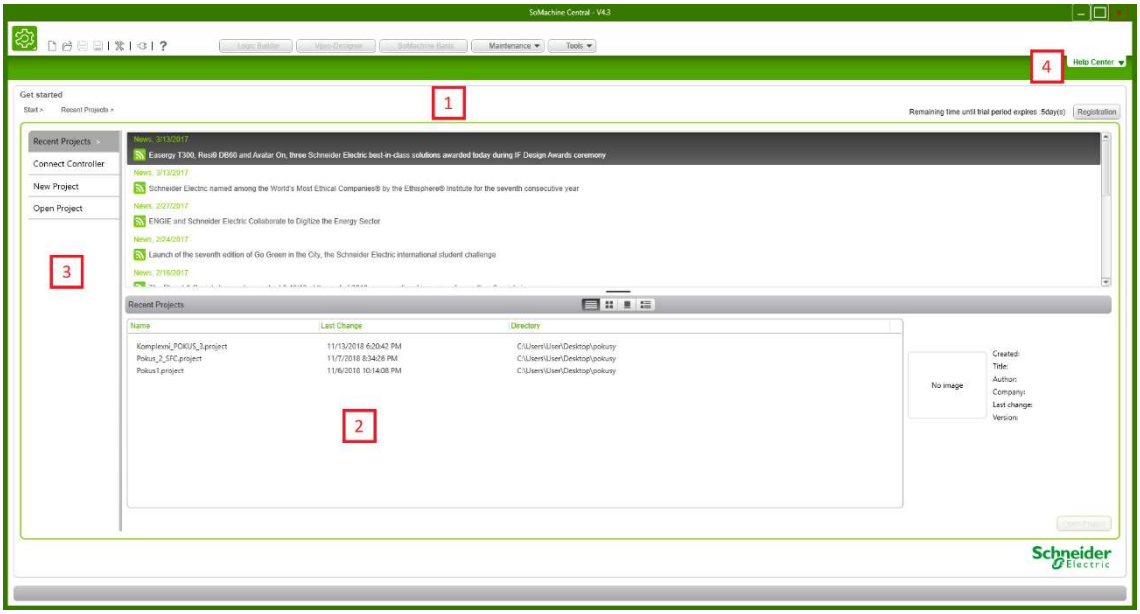
Obr. 2-3: GX Works2

3. SCHNEIDER ELECTRIC – SOMACHINE PRO PROGRAMOVÁNÍ PLC

Společnost Schneider Electric nabízí několik řad PLC kontrolérů. Pro tuto práci je zvolen k testování kompaktní model střední třídy M238 TM238LDD24DT, který lze vidět na obrázku 3-1 [11]. Tento logický kontrolér je ve svém základu (bez žádných přídavných modulů) napájen 24 V DC a má k dispozici 14 digitálních vstupů (osm rychlých) a 10 digitálních výstupů (čtyři rychlé). Tyto rychlé vstupy a výstupy dokáží pracovat až do frekvence 100 kHz. Na vstupy i na rychlé vstupy lze přivést buď signál napěťový, který je vnímám jako logická jedna od hodnoty 15 V do hodnoty 30 V, nebo signál proudový, který je vnímám jako logická jedna od hodnoty 2 mA. Výstupní signál má hodnotu 24 V DC a přijímaný i posílaný signál je indikován LED kontrolky na předním krytu PLC. Pro programování a komunikaci je k dispozici konektor mini USB a rychlejší konektor Ethernet [11].

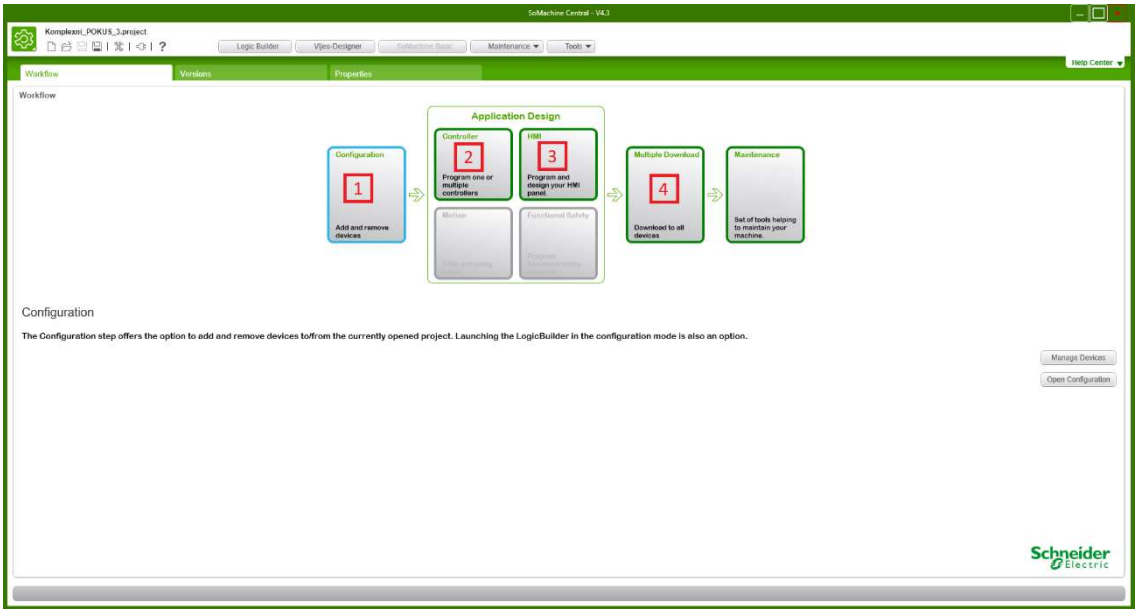
Obr. 3-1: M238 TM238LDD24DT [11]

Programové prostředí SoMachine, vyvinuté společností Schneider Electric, je složeno z několika dílčích softwarů, které společně tvoří celý systém kontroly a řízení. Při puštění SoMachine se objeví centrální obrazovka SoMachine, kterou lze vidět na obrázku 3-2. Vyznačená oblast 1 se nazývá SoMachine Central, nebo také domovská stránka, v oblasti 2 je seznam předchozích otevřených projektů, v oblasti 3 jsou dostupné operace a oblast 4 ukazuje přístup do centra nápovědy. V oblasti 3 je možnost vybrat operaci *Connect Controller*, kde lze zjistit, zda je již ve spojení s nějakým kontrolérem. Pokud je vybrána možnost *New Project*, je zde možnost otevřít nový prázdný projekt, do kterého je třeba dodatečně vložit patřičné komponenty, které mají být programovány a konfigurovány, anebo už při vytváření nového projektu ihned přiřadit patřičné komponenty [12].



Obr. 3-2: Centrální obrazovka SoMachine

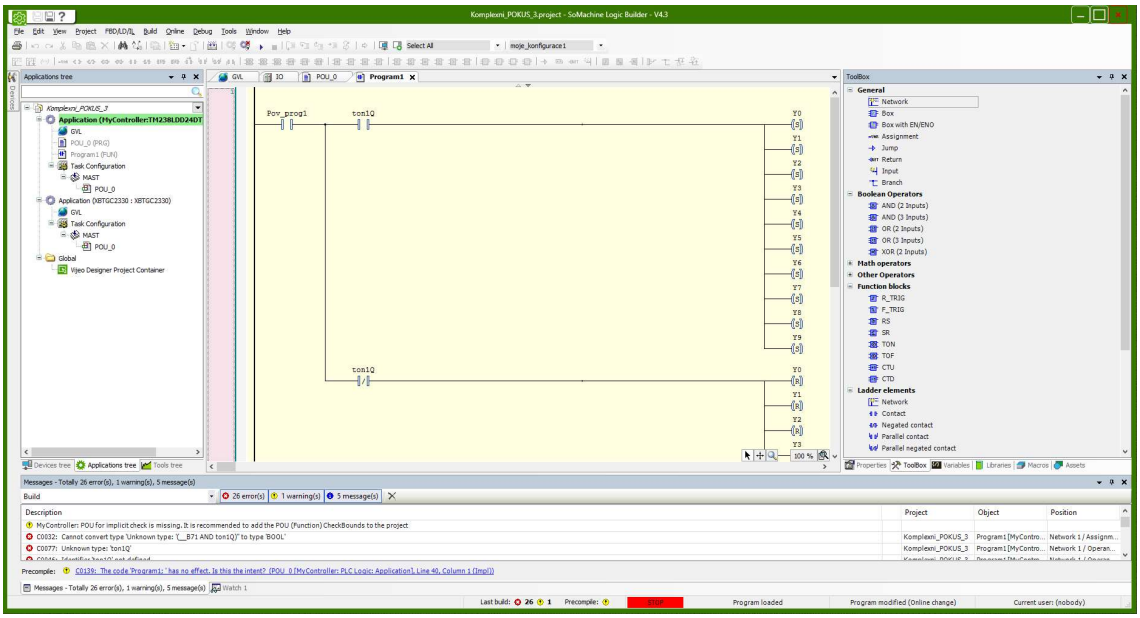
Jestliže je vybrán logický kontrolér i HMI kontrolér zobrazí se okno, které je vidět na obrázku 3-3. Na tomto obrázku jsou vyznačeny čtyři základní oblasti. V první oblasti je možnost editovat výběr komponentů. Kliknutím na oblast 2 se spustí program Logic Builder, ve kterém je psán vlastní program PLC. Kliknutím na oblast 3 se spustí program Vijeo-Designer, který slouží pro nakonfigurování chování a nastavení vzhledu vizualizace, jež se předpokládá pro nahrání do HMI kontroléru. Oblast 4 slouží k hromadnému nahrávání programů a konfigurace do všech zvolených komponentů v oblasti 1 [12].



Obr. 3-3: Obrazovka po vytvoření nového projektu

3.1 SoMachine – Logic Builder

Logic Builder je software pro psaní řídícího programu, který se následně posílá do logického kontroléru a jeho programovací prostředí je patrné na obrázku 3-4.



Obr. 3-4: Logic Builder

Na levé straně na obrázku 3-4 lze vidět karta *Devices tree*, ve které jsou vyobrazeny všechny nakonfigurované komponenty, které byly přiřazeny do projektu ihned po založení. Pokud je zařízení připojeno, je kolem jeho označení plný zelený rámeček. V této kartě lze přiřadit vstupy a výstupy, také se zde nastavuje způsob komunikace s daným logickým kontrolérem a konfigurace této komunikace. Ovšem, že lze na tomto místě zobrazit i jiné karty rychlým přístupem ve spodní části této karty. Pokud tam tento výběr není, dá se přidat v záložce *View – Navigators/Classic Navigators* a zde si vybrat potřebné karty, které se po vybrání otevřou a přiřadí do rychlého přístupu ve spodní části. V kartě *Applications tree* se vytvářejí programy, funkce a funkční bloky. Pokud je potřeba některý z těchto prvků vytvořit, je třeba kliknout pravým tlačítkem na vybraný logický kontrolér a vybrat položku *Add Object – POU*. Následně se zobrazí okno, ve kterém lze vybrat, jaký prvek je možné vytvořit. V poslední fázi je vybrán vhodný programovací jazyk. Hlavní program i funkční blok lze psát v programovacích jazycích CFC (Continuous Function Chart), FBD, IL, LD, SFC i ST a funkce je možné psát ve všech těchto jazycích kromě jazyka SFC. V kartě *Tools tree* je možné přidat nebo upravit knihovny funkcí, ze kterých je čerpáno. Také lze zobrazit i kartu *Devices* obsahující

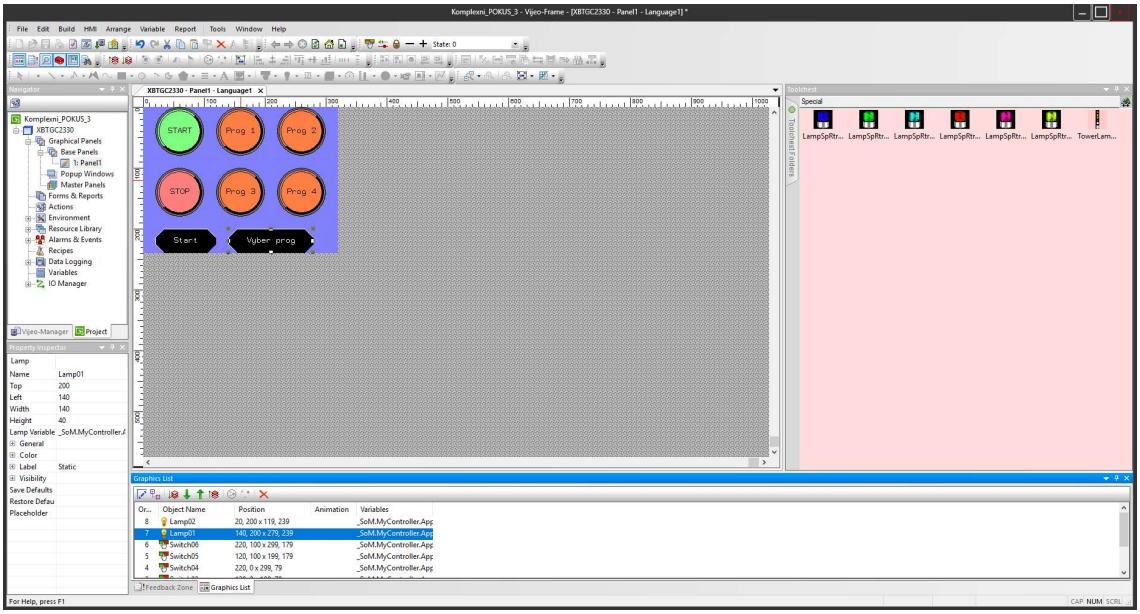
všechny tyto karty v jedné, která se dá přichytit na boční lištu, tudíž dokáže i šetřit místo na obrazovce, což je velice výhodné, pokud jsou vytvářeny komplexní programy [12].

Vpravo na obrázku 3-4 lze vidět otevřenou kartu *ToolBox*, ve které je nabídka základních logických prvků, které je možno přetažením přidat do aktuálně psaného programu. Jsou zde základní prvky booleovské algebry, matematické operátory, funkční bloky, časovače a další, ovšem do programu se dá vložit i prázdný funkční blok, do kterého následně lze nahrát složitější funkce z knihovny funkcí, třeba PID regulátor. V tomto místě se dají zobrazit i další karty (*Properties, Variables, Libraries* apod.) ve spodní části z rychlého přístupu nebo přes záložku *View – Hardware Catalog/Software Catalog* [12].

Na obrázku 3-4 uprostřed lze vidět, v jakém místě jsou defaultně zobrazovány aktuálně otevřené karty. Po zobrazení jiné karty, pokud není předchozí karta vypnuta, ale pouze zobrazena jiná karta, se automaticky tato předchozí karta uloží do rychlého výběru v horní části. Zde je možno v programové kartě zobrazit i lokální proměnné malou nenápadnou šipkou v horní části pod rychlým výběrem. Pokud je již vytvořen program, je nezbytné jej zkontrolovat v záložce *Build – Build*. Pokud je vyhodnocena nějaká chyba nebo něco není úplně v pořádku, je to napsáno v dolní části v kartě *Messages*. Stejně tak je nezbytné, pokud jsou v napsaném programu proměnné, které představují výstupní periferie či vstupní periferie, přiřadit je k určeným výstupním či vstupním periferiím. Toto lze provést tak, že při vytváření proměnné je k této proměnné přiřazena adresa požadované periferie. Nebo také je možno adrese přiřadit proměnou v levé části v kartě *Device tree* položkou *IO – IO Mapping – Imputs/Outputs* a ve sloupci *Channel* poté napsat k jednotlivým adresám názvy proměnných, které budou přiřazeny k danému výstupu. Program lze také simulovat v záložce *Online – Simulation*. Pokud je třeba za chodu programu sledovat jednotlivé stavy proměnných, pak je nezbytné přiřadit je v dolní části programového prostředí v kartě *Watch1*. Tato karta je velmi užitečná při odladění hrubých chyb programu [12].

3.2 SoMachine – Vijeo-Designer

Vijeo-Designer je další software z balíčku programů SoMachine. Slouží k vytváření a konfiguraci programové vizualizace. Programové vizualizační prostředí Vijeo-Dsigner lze vidět na obrázku 3-5.

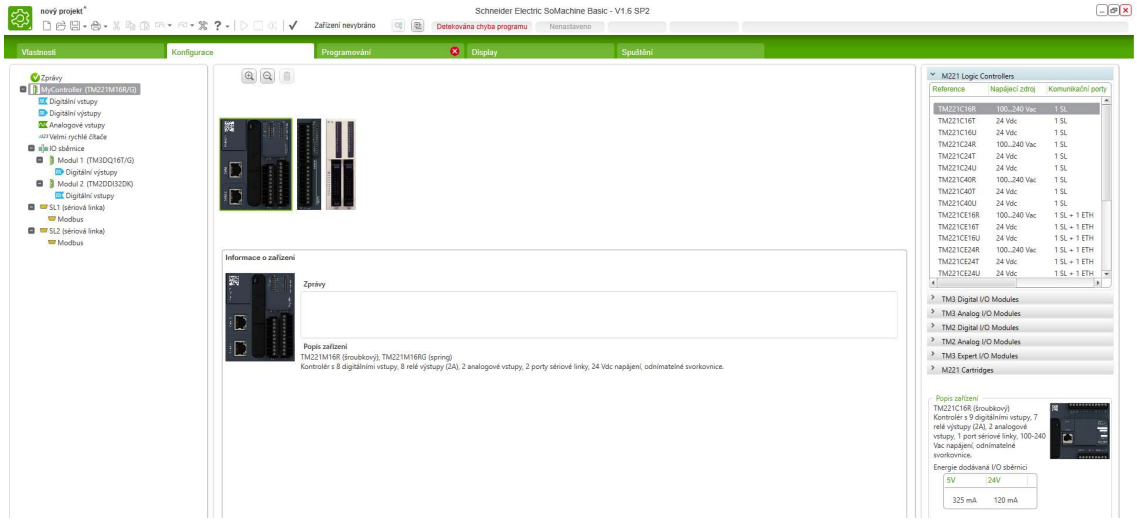


Obr. 3-5: Vijeo-Designer

Uprostřed programového prostředí Vijeo-Designer je pracovní plocha, do které jsou umísťovány programovatelné vizualizační prvky. Tyto prvky mohou mít charakter *Switch, Lamp, Numeric display, Message display, Meter* atd. a jejich ikony jsou umístěny v horní části programového prostředí. Těmto programovatelným prvkům lze v jejich vlastnostech nastavit barvu k jednotlivým stavům prvků, přiřadit proměnné, které mohou daný prvek řídit, ale současně daný prvek může řídit proměnnou, tedy ji setovat, resetovat, přepínat apod. Toto nastavení lze provést i v kartě v levé dolní části, kde se při kliknutí na určitý vizualizační prvek zobrazí jeho vlastnosti, ovšem ne všechny. V levé části se také nachází karta *Navigator*, která slouží k přístupu konfigurace. V pravé části lze zobrazit kartu *Toolchest* její ikonou v pravé horní části nebo klávesovou zkratkou *Shift+F9*. V této kartě jsou k dispozici všechny dostupné vizualizační prvky z knihovny. V dolní části je vidět karta *Graphics List* zobrazující všechny použité vizualizační prvky s jejich pozicí. Simulaci vizualizace lze spustit v záložce *Build – Simulation*. Pokud fyzicky nejsou k dispozici tlačítka ani spínače, které by bylo možno připojit na vstupní periferie PLC, je simulace velmi užitečná, a to tím, že tyto vstupní proměnné lze přiřadit k vizualizačním prvkům a je možno za chodu programu, nahraném v PLC, ovládat výstupní periferie v simulaci vizualizace v reálném čase [13].

3.3 SoMachine – SoMachine Basic

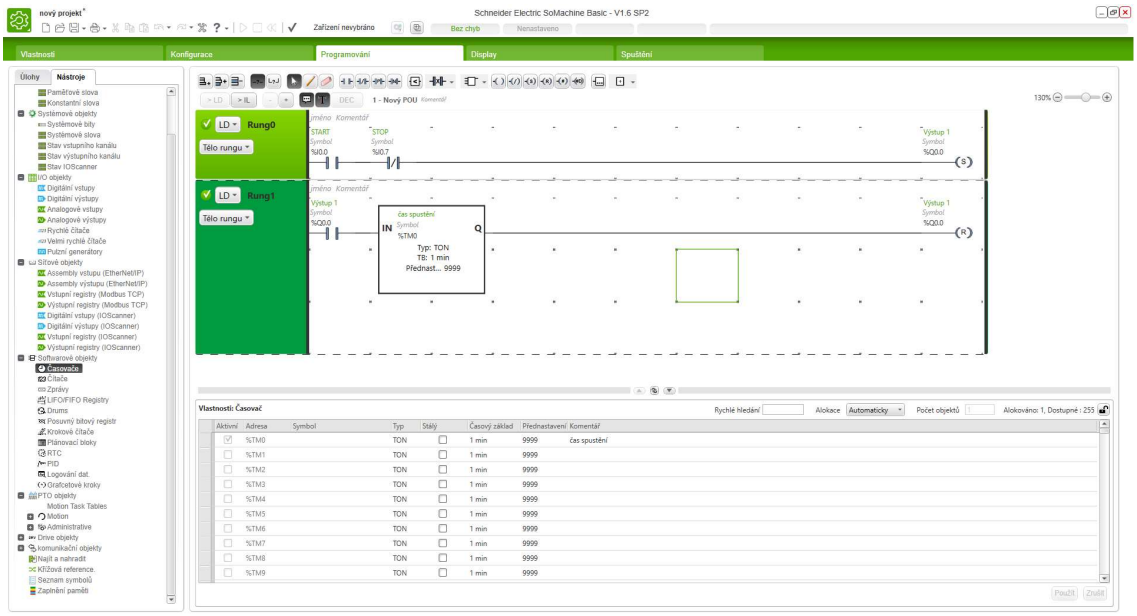
SoMachine Basic je součástí balíčku SoMachine a je to také programovací prostředí, stejně jako Logic Builder od společnosti Schneider Electric. Ovšem značný rozdíl je v možnosti použití. Logic Builder lze použít na programování všech řad modelů PLC od Schneider Electric, zatímco SoMachine Basic lze použít výhradně na základní řadu M221 [14].



Obr. 3-6: SoMachine Basic – Konfigurace

Po spuštění je potřeba založit nový projekt. Celý projekt je shrnut do pěti hlavních záložek (*Vlastnosti, Konfigurace, Programování, Display* a *Spuštění*) mezi kterými se přepíná v horní části v zelené liště, jak je patrné na obrázku 3-6. V záložce *Vlastnosti* je možnost vyplnit připravené kolonky informacemi o projektu a případně projekt zaheslovat. Editace zařízení a všech komponentů je v záložce *Konfigurace*. Pokud je třeba přidat nějaké další zařízení, je potřeba jej vybrat z nabídky na pravé straně a přetažením do prostoru uprostřed jej přidat. Automaticky se přidá i konfigurace tohoto nového zařízení do seznamu v levé části, kde je možné dohledat adresy a nastavit komunikaci všech přidaných zařízení a komponentů. V záložce *Programování* je vytvářen logický program. Struktura programu je tvořena jednotlivými větvemi, které mohou být psány v odlišných programovacích jazycích. Jsou zde nabídnuty dva programovací jazyky, a to LD a IL. Pro vkládání prvků na danou větev slouží rychlá nabídka základních prvků (*Kontakt, Negovaný kontakt, Kontakt náběžné hrany, Cívka, Resetovací cívka* atd.) nad programovacím prostorem, jak lze vidět na obrázku 3-7. Pro vložení složitějších funkcí slouží ikona *Funkční bloky* nebo ji také lze spustit klávesami *F8* či *ALT+F*. Poté vybrat

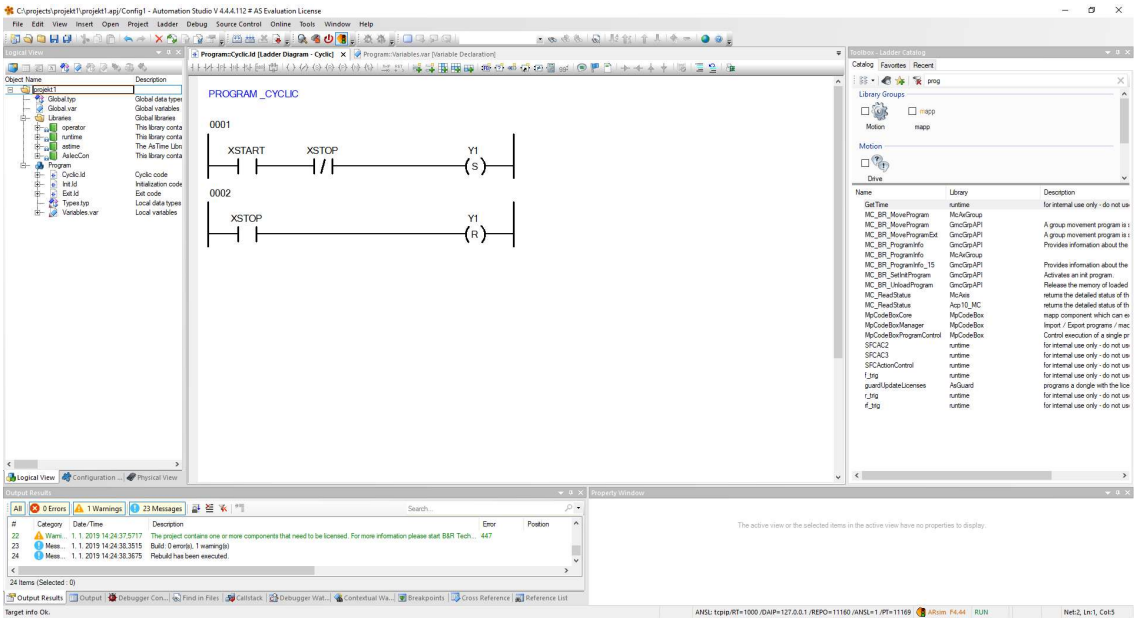
požadovaný blok a nastavit jeho chování. Program má již předem nachystané proměnné k jednotlivým složitějším funkcím, které lze dohledat v levé části v položce *Softwarové objekty*. Pokud je napsán komentář u proměnné, je jako funkční text zobrazen i v tomto přehledu. Program sám kontroluje, zda je naprogramovaná větev správně. Jestliže je program hotový, je třeba jej nechat zkompilovat ikonou *Kompilovat program*, a poté je zde možnost odzkoušet jeho chování pomocí simulace, a to ikonou *Spustit simulátor* v horní části programovacího prostředí [14].



Obr. 3-7: SoMachine Basic – Programování

4. B&R – PROGRAMOVACÍ PROSTŘEDÍ AUTOMATION STUDIO

Programovací prostředí Automation Studio od společnosti B&R Industrial Automation je integrované vývojové prostředí, které obsahuje nástroje pro všechny fáze projektu. Řídící jednotka, pohon, komunikace i vizualizace mohou být konfigurovány a programovány v jednom prostředí. Uživatelům jsou k dispozici všechny programovací jazyky dle normy IEC 61131-3 a ANSI C. Všechny tyto jazyky lze kombinovat dle potřeby. Po spuštění Automation Studio je nezbytné založit nový projekt, při jehož zakládání je nutné vybrat hlavní řídící jednotku. V založeném projektu je možnost editace komponentů, přesně dle potřeby. Ze záložky *Toolbox – Hardware Catalog* na pravé straně lze tyto komponenty přidat do prostoru tomu vyhrazenému. Tento prostor lze otevřít ze záložky *Physical View* na levé straně. Na straně levé jsou defaultně připnuty další dvě záložky *Configuration View* a *Logical View*. V *Configuraton View* je možnost nastavení veškeré komunikace mezi přiřazenými komponenty. V záložce *Logical View* se vytváří hlavní řídící program (viz obr. 4-1). Tento řídící program se přidává přetažením z pravé strany z *Toolbox – Object Catalog*, kde se nejdříve vybere soubor s požadovaným programovacím jazykem. Po přidání řídícího programu, je nutné definovat proměnné ve *Variables .var*. Program je možné vyzkoušet v simulaci, a také je možnost vytvořit vizualizaci pro tento program [15] [16].



Obr. 4-1: Automation Studio

5. SROVNÁNÍ JEDNOTLIVÝCH TYPŮ PROGRAMOVACÍCH PROSTŘEDÍ

GX Works2 je programovací prostředí od Mitsubishi, které je popsáno v kapitole 2.1. Mitsubishi nabízí řadu kontrolérů střední třídy, které již ve svém základu mají převážně analogové vstupy/výstupy, což je velice výhodné, zejména pro měření teploty, které je zcela běžné při průmyslových aplikacích, nebo pro regulaci pomocí PID regulátorů. Ovšem v GX Works2 je relativně zdlouhavé vyhledávání složitějších funkcí, pokud si uživatel nepamatuje jejich korektní zápis. Toto programovací prostředí v sobě nezahrnuje možnost tvorby vizualizace. Možnost instalace tohoto programovacího prostředí je značně omezena, poněvadž GX Works2 nepodporuje ovladače zařízení pro Windows 10. GX Works2 je možné stáhnout s 60denní bezplatnou verzí z internetových stránek výrobce.

Programovací prostředí SoMachine od výrobce Schneider Electric je popsáno v kapitole 3. Instalace tohoto prostředí je rozsáhlá, neboť se instaluje více vývojových prostředí. Kontroléry střední třídy nejsou běžně vybaveny analogovými vstupy/výstupy, ovšem je možnost připojení externího modulu s těmito periferiemi. Balíček SoMachine obsahuje i vývojové prostředí pro vizualizaci, které je jednoduché a pro běžného uživatele dostačující. Výrobce nabízí neomezené bezplatné vývojové prostředí SoMachine Basic, ovšem toto prostředí je omezeno pouze na jednu řadu kontrolérů. Celý balíček SoMachine je ke stažení na internetových stránkách výrobce s 21denní bezplatnou verzí.

Automation Studio, jenž je popsáno v kapitole 4, od výrobce B&R Industrial Automation je integrované prostředí pro vývoj řídícího programu i vizualizace. Toto programovací prostředí je v porovnání s GX Works2 a SoMachine složitější a méně přehledné pro nepříliš zkušeného uživatele. Automation Studio je ke stažení na internetových stránkách výrobce s 90denní plnou bezplatnou verzí a s možností pořízení studentské verze na 400 dní.

V porovnání těchto tří programovacích prostředí je pro výukové účely nejvýhodnější SoMachine, zejména kvůli možnosti tvorby jednoduché vizualizace, která je v současné době nedílnou součástí projektů, ale také i pro jednoduchost tvorby programu i se složitějšími funkcemi ve velmi intuitivním a jednoduchém programovacím prostředí.

6. NÁVRH LABORATORNÍCH ÚLOH

Součástí této bakalářské práce je také návrh několika laboratorních úloh, na kterých lze vyzkoušet řešení dané problematiky za pomoci řízení PLC a vytvoření příslušných laboratorních návodů. Na základě těchto laboratorních úloh se lze seznámit se základními logickými prvky, pomocí kterých je následně možné řešit složitější i komplexní úkoly v praxi.

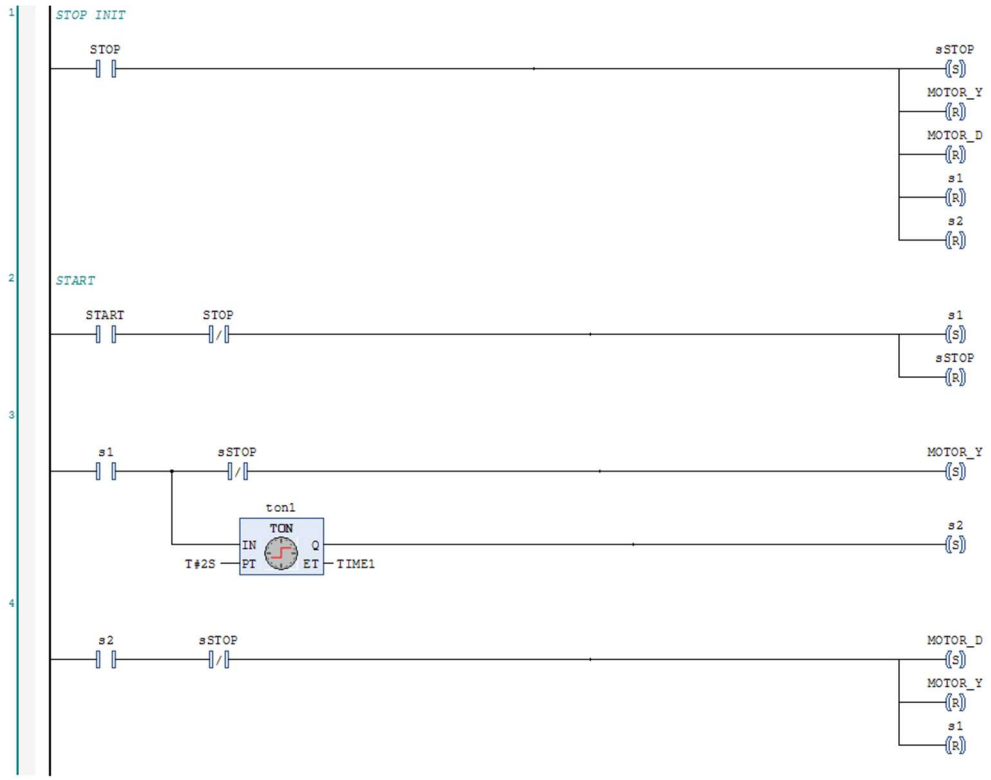
6.1 Laboratorní úloha 1: Spouštění motoru přepínáním Y/D

Zadání:

Seznamte se s programovacím prostředím SoMachine a s nápovědou k tomuto prostředí. Napište program pro automatické přepnutí stykačů Y/D po dvou sekundách chodu motoru. K ovládání motoru použijte pouze tlačítka START a STOP. Ukázka programu je na obrázku 6-1.

Příklad postupu řešení dané úlohy:

a) Spusťte SoMachine a založte nový projekt.
b) Zvolte vhodný programovací jazyk dle uvážení.
c) Definujte hlavní vstupní proměnné např. START a STOP.
d) Definujte hlavní výstupní proměnné např. MOTOR_Y a MOTOR_D.
e) Tlačítkem START spusťte motor do chodu Y a zajistěte paměť, aby nebylo nutné neustále držet tlačítko START.
f) Nastavte časovač (zpožděné zapnutí) tak, aby po dvou sekundách chodu motoru do Y přepnul na chod motoru do D a zároveň zajistěte vypnutí chodu motoru do Y.
g) Stykače Y a D musí být proti sobě blokovány, aby nemohlo dojít k současnému sepnutí obou stykačů.
h) Celý chod programu musí jít kdykoliv přerušit krátkým stisknutím tlačítka STOP.
i) Po krátkém stisku tlačítka STOP se musí program resetovat a po opětovném spuštění motoru tlačítkem START rozběhnout motor do chodu Y.



Obr. 6-1: Ukázka programu v LD pro laboratorní úlohu 1

6.2 Laboratorní úloha 2: Garážová vrata

Zadání:

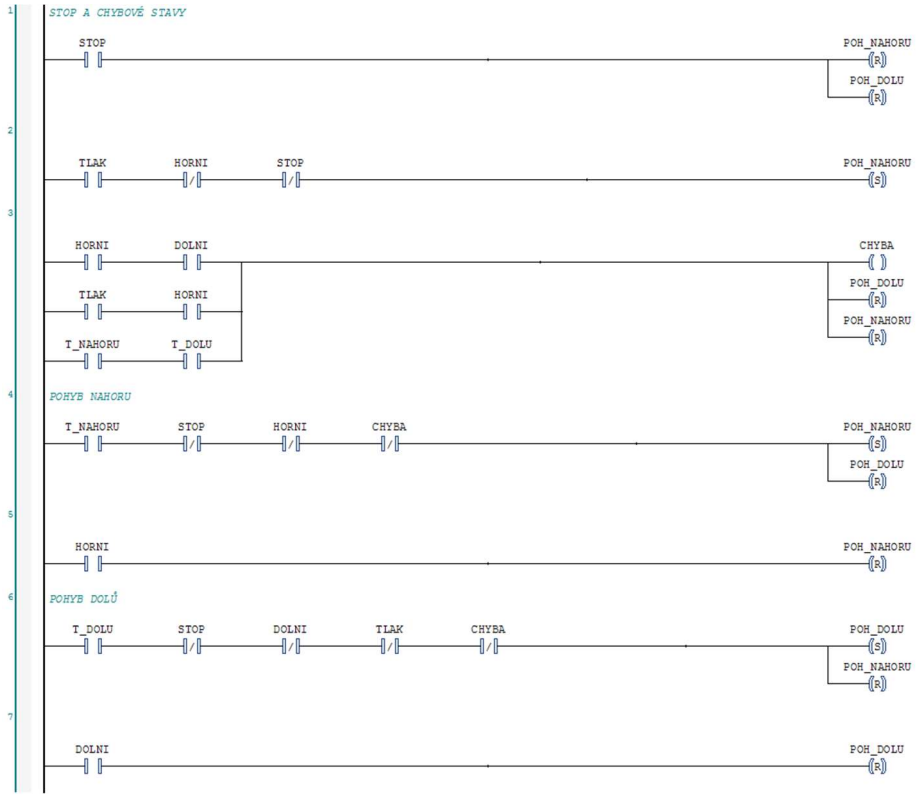
Napište program pro řízení garážových vrat. Tato garážová vrata mají pohon pro pohyb nahoru, pohon pro pohyb dolů, horní koncový spínač, dolní koncový spínač, bezpečnostní tlakový spínač a chybovou kontrolku. Pro ovládání těchto vrat jsou k dispozici tlačítka T_NAHORU, T_DOLU a STOP tlačítko, díky kterému musí garážová vrata kdykoliv a v jakékoliv poloze zastavit.

Při stisku tlačítka T_NAHORU musí sepnout pohon pro pohyb vrat nahoru a zůstat sepnutý, dokud nebude sepnut horní koncový spínač. Podobně se musí vrata chovat při stisknutí tlačítka T_DOLU, kdy musí sepnout pohon pro pohyb dolů, dokud nesepne dolní koncový spínač. Při pohybu vrat dolů se může stát, že sepne bezpečnostní tlakový spínač při nárazu na překážku. Tímto impulsem musí vypnout pohon pro pohyb vrat dolů a zapnout pohon pro pohyb vrat nahoru. Dále musí být ošetřeny chybové stavy

spínačů, při jejich případné poruše, rozsvícením chybové kontrolky a vypnutím pohonů pro pohyb vrat. Ukázka programu je na obrázku 6-2.

Příklad postupu řešení dané úlohy:

a) Spusťte SoMachine, založte nový projekt a zvolte vhodný programovací jazyk dle uvážení.
b) Definujte hlavní vstupní proměnné např. T_NAHORU, T_DOLU, STOP a spínače TLAK, HORNI a DOLNI.
c) Definujte hlavní výstupní proměnné např. POH_NAHORU, POH_DOLU a kontrolku CHYBA.
d) Tlačítkem NAHORU spusťte POH_NAHORU, aby vrata vyjela a zajistěte paměť, aby nebylo nutné držet neustále tlačítko T_NAHORU.
e) POH_NAHORU musí zůstat sepnutý, dokud nebude aktivní spínač horní polohy (HORNI). Poté musí POH_NAHORU vypnout.
f) Tlačítkem T_DOLU spusťte POH_DOLU a opět zajistěte paměť.
g) POH_DOLU musí opět vypnout při dosažení dolní polohy, která je signalizována spínačem DOLNI.
h) Pohony POH_NAHORU a POH_DOLU musí být proti sobě blokovány, aby se nemohlo stát, že by byly spuštěné zároveň (proti sobě).
i) Při aktivaci spínače TLAK (tlakový spínač) musí být aktivován POH_NAHORU (který samozřejmě nejde spustit, pokud garážová vrata dosáhla horní polohy a je sepnut spínač HORNI) a vypnut POH_DOLU.
j) Pokud jsou sepnuty oba polohové spínače HORNI a DOLNI nebo spínače TLAK a HORNI, musí tyto stavy být vyhodnoceny jako chybové. Při tomto stavu se musí zastavit pohon a rozsvítit kontrolka CHYBA.
k) Celý chod programu musí jít kdykoliv přerušit krátkým stisknutím tlačítka STOP.
l) Po přerušení tlačítkem STOP se stisknutím tlačítka T_NAHORU nebo T_DOLU musí chod programu obnovit.



Obr. 6-2: Ukázka programu v LD pro laboratorní úlohu 2

6.3 Laboratorní úloha 3: Lakovna

Zadání:

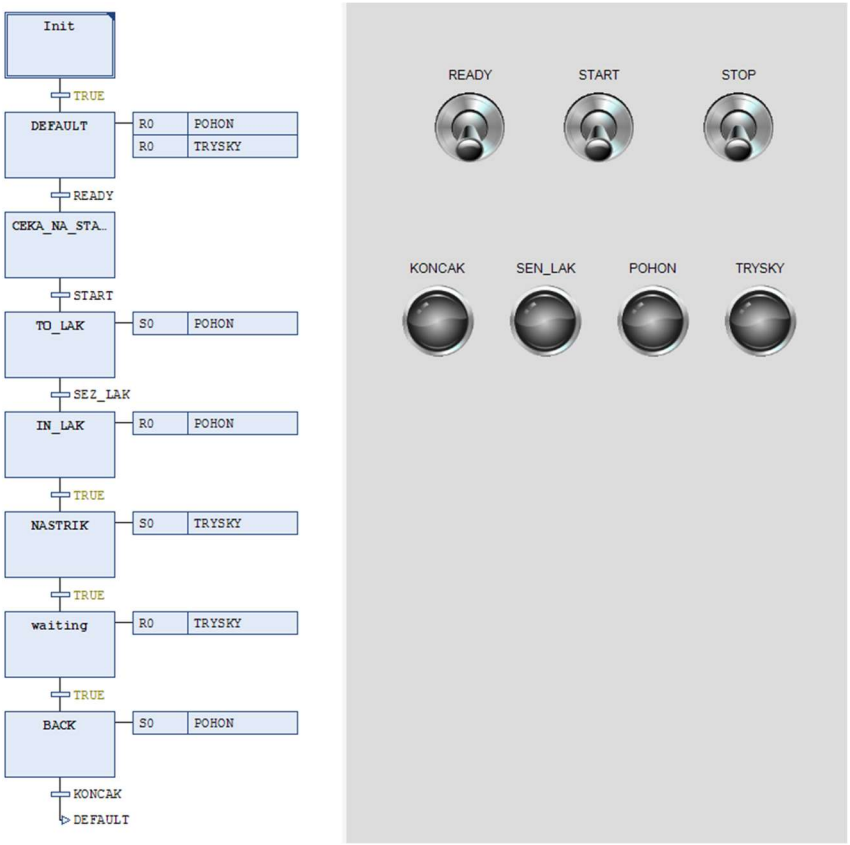
Vytvořte řídící program pro automatizovanou lakovnu výrobků. Tato lakovna má závěsný dopravník, díky kterému se výrobek, který má být lakován, dopraví do lakovací kabiny a následně je po dokončení lakování vyvezen zpět před lakovací kabinu. Zde je závěsný dopravník připraven na zavěšení dalšího výrobku pro lakování a opětovné spuštění celého lakovacího procesu.

Lakovna má na začátku závěsného dopravníku tlačítko READY, které musí obsluha zodpovídající za správné upnutí stisknout. Pakliže je toto zajištěno, pohon závěsného dopravníku se po stisknutí tlačítka START spustí a běží, dokud nezaznamená uvnitř lakovací kabiny senzor výrobek. Poté se tento pohon musí vypnout a v lakovací kabině se za 2 sekundy spustí trysky s lakem na 20 sekund. Sepnutí trysek s lakem musí být pro větší bezpečnost podmíněno zastavením závěsného dopravníku. Po dokončení

lakování se trysky s lakem vypnou a vyčká se dalších 20 sekund. Po tomto čase se závěsný dopravník opět spustí, aby vyvezl nalakovaný výrobek z lakovací kabiny. Jakmile sepne koncový spínač, závěsný dopravník se zastaví a čeká, až obsluha sundá nalakovaný výrobek. Po tomto kroku je lakovna připravená na opětovný lakovací proces. Stiskem tlačítka STOP musí jít kdykoliv přerušit lakovací proces. Po obnovení procesu se musí lakovací program obnovit v tom časovém okamžiku, kdy byl pozastaven. Doporučuje se použít vestavěnou funkci pro pozastavení celého programu. Ukázka programu a vizualizace je na obrázku 6-3.

Příklad postupu řešení dané úlohy:

a) Spusťte SoMachine, založte nový projekt a zvolte vhodný programovací jazyk.
b) Definujte hlavní vstupní proměnné START, STOP, READY, SENZ_LAK a KONCAK.
c) Definujte hlavní výstupní proměnné POHON a TRYSKY.
d) Tlačítkem START spusťte POHON (pod podmínkou, že je stisknuté tlačítko READY a zajistěte paměť, aby nebylo nutné stále držet tlačítko START).
e) POHON musí vypnout po aktivaci SEN_LAK a za 2 sekundy se musí aktivovat TRYSKY.
f) TRYSKY jsou aktivní 20 sekund, poté se vypnou a čeká se dalších 20 sekund.
g) Následně se aktivuje POHON a bude aktivní, dokud nesepne koncový spínač KONCAK.
h) Celý chod programu musí jít kdykoliv přerušit krátkým stisknutím tlačítka STOP.
i) Po přerušení se musí lakovací proces obnovit v tom okamžiku kdy byl pozastaven.
j) Vytvořte přehlednou a jednoduchou vizualizaci k ovládání a sledování aktuálních stavů senzorů, pohonu a trysek v lakovacím procesu.



Obr. 6-3: Ukázka programu v SFC a vizualizace pro laboratorní úlohu 3

6.4 Laboratorní úloha 4: Pásové dopravníky

Zadání:

Vytvořte řídící program pro ovládání dvou pásových dopravníků. Jakmile senzor na začátku prvního dopravníku zaznamená výrobek, spustí se pohon pro první dopravník. Až výrobek dojede k druhému dopravníku, tak se tento dopravník zapne a první dopravník se vypne. Před exportní bednou jsou pomocí senzoru počítány výrobky. Jakmile se exportní bedna naplní, musí se oba dopravníky zastavit a hlásit stav plné exportní bedny. Obsluha má k dispozici dvě tlačítka START, STOP a dvě kontrolky FULL a CHYBA.

Na začátku prvního dopravníku je první senzor, který detekuje výrobek. Jestliže detekuje výrobek, aktivuje pohon prvního dopravníku. Až výrobek přijede na prvním pásovém dopravníku ke druhému senzoru, který se nachází na konci prvního dopravníku, dá pokyn pro puštění druhého dopravníku a zároveň za dvě sekundy vypne první

dopravník. Na konci druhého dopravníku je třetí senzor, který opět se zpožděním vypíná druhý dopravník, ale také počítá, kolik již prošlo kusů výrobku do exportní bedny. Jakmile projde pět kusů výrobků, dopravníky se zastaví a hlásí pomocí kontrolky FULL, že je třeba vyměnit exportní bednu s výrobky. Jakmile je bedna vyměněna, obsluha spustí proces stiskem tlačítka START. V ten moment se spustí oba dopravníky a jestliže jakýkoliv senzor nezaznamená žádný výrobek do pěti sekund, vypnou se. V tento moment jsou dopravníky opět připraveny na povel spuštění prvním senzorem. Pokud při aktivaci prvního senzoru nedojde k aktivaci druhého senzoru do osmi sekund nebo po aktivaci druhého senzoru nedojde do osmi sekund k aktivaci třetího senzoru, musí tento stav být vyhodnocen jako chybový aktivací chybové kontrolky CHYBA a vypnutím dopravníků. Tato ochrana by měla detekovat zaseknutí výrobku na dopravnících apod. Opětovná aktivace procesu je možná tlačítkem START. Celý proces dopravníků lze kdykoliv přerušit tlačítkem STOP. Po tomto přerušení se tlačítkem START spustí oba dopravníky, stejně jako v případě vyměnění exportní bedny nebo znovu aktivace po chybovém hlášení kontrolkou CHYBA. Ukázky části programu a vizualizace jsou na obrázcích 6-5 a 6-4.

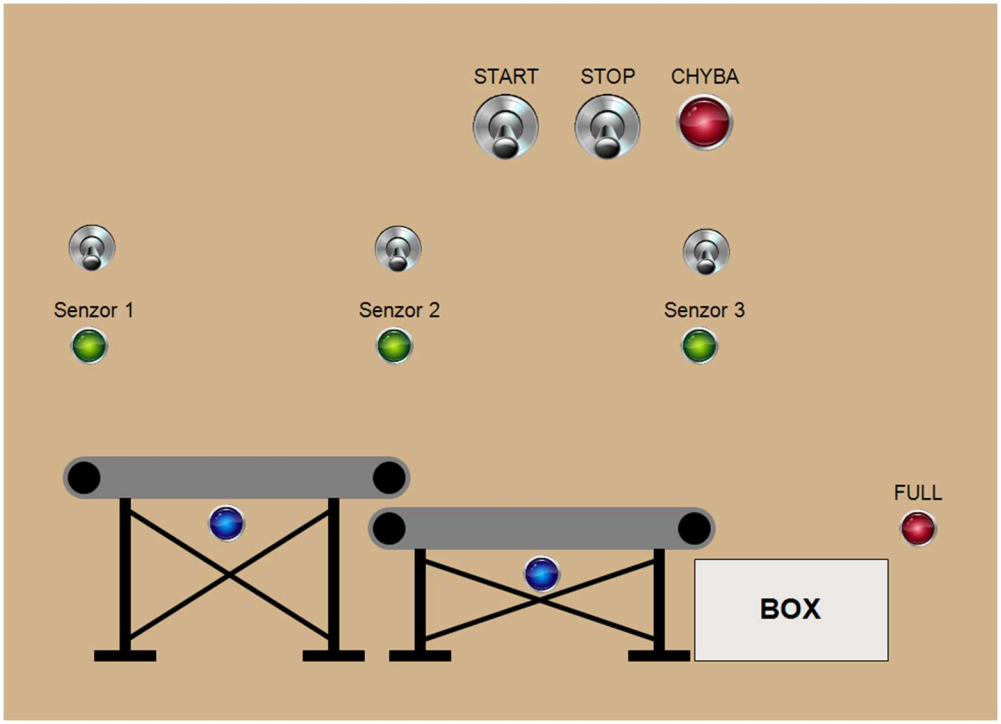
Příklad postupu řešení dané úlohy:

a) Spusťte SoMachine, založte nový projekt a zvolte vhodný programovací jazyk.
b) Definujte hlavní vstupní proměnné START, STOP, SENZOR1, SENZOR2 a SENZOR3.
c) Definujte hlavní výstupní proměnné FULL, CHYBA a pohony POH1 a POH2.
d) Při indikaci výrobku senzorem SENZOR1 se aktivuje POH1. Jakmile je aktivován SENZOR2 aktivuje se POH2 a za dvě sekundy je deaktivován POH1 (samozřejmě pokud mezitím opět nesepnul SENZOR1).
e) Při indikaci SENZOR3 je za dvě sekundy vypnut POH2 (opět pokud mezitím nesepnul SENZOR2) a také na tomto senzoru musí být čítač, který při načítání pěti indikací rozsvítí kontrolku FULL a ihned vypne oba pohony.
f) Po stisku START se spustí oba pohony. Pokud se neaktivuje žádný senzor do pěti sekund, tyto pohony se následně opět vypnou.
g) Při aktivaci SENZOR1 musí být aktivován SENZOR2 do osmi sekund, jinak tento stav musí být vyhodnocen jako chybový kontrolkou CHYBA.

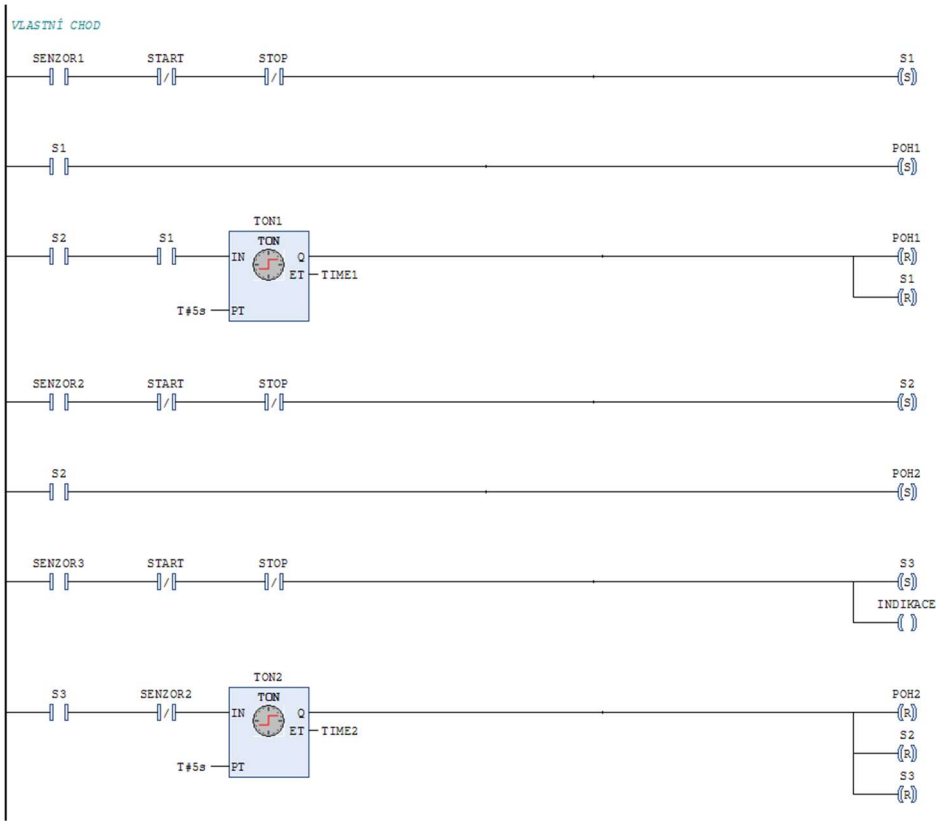
h) Při aktivaci SENZOR2 musí být aktivován SENZOR3 do osmi sekund, jinak tento stav musí být vyhodnocen jako chybový kontrolkou CHYBA.

i) Při stisknu STOP se oba pohony musí ihned zastavit. Opětovné spuštění pomocí tlačítka START.

j) Vytvořte přehlednou a jednoduchou vizualizaci k tomuto procesu.



Obr. 6-4: Ukázka vizualizace pro laboratorní úlohu 4



Obr. 6-5: Ukázka části programu v LD pro laboratorní úlohu 4

6.5 Laboratorní úloha 5: Automatizované vrtací rameno

Zadání:

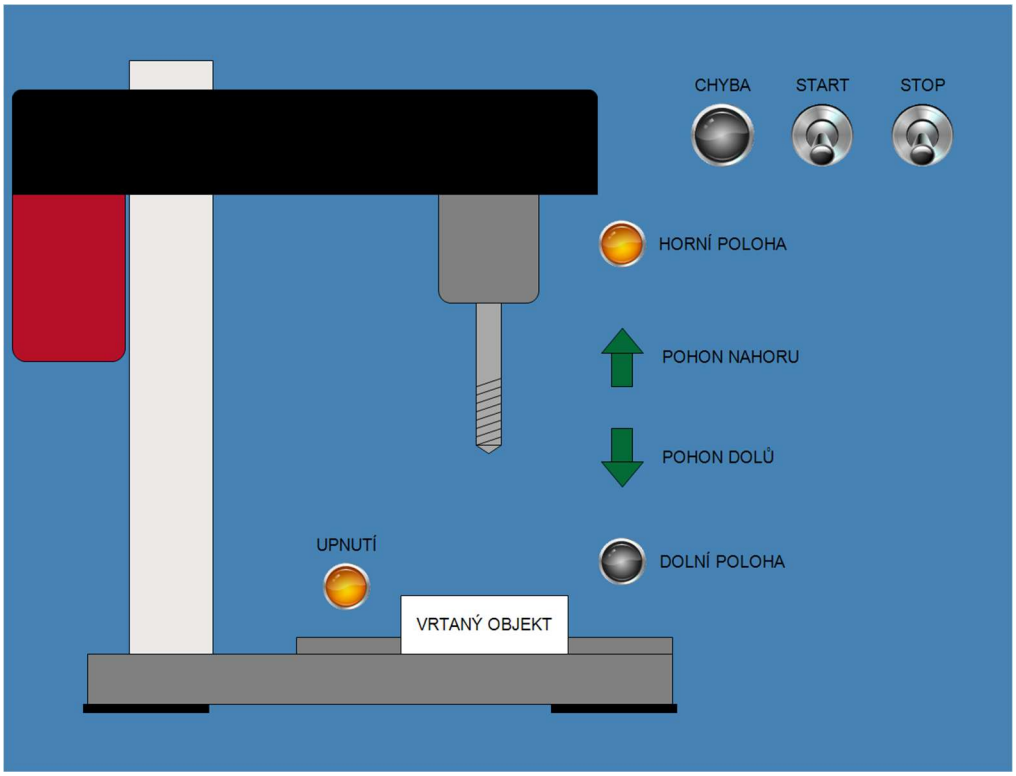
Napište řídící program pro chování automatizovaného vrtacího ramena. Toto rameno má přípravek na upevnění výrobku se senzorem, který detekuje správné upnutí. Jakmile je výrobek upnutý, stiskem tlačítka START sjede vrtací hlava s vrtajícím vrtákem dolů, kde setrvá jednu sekundu a následně vyjede zpět nahoru do výchozí polohy. K ovládání vrtacího ramene je zapotřebí pouze tlačítka START, STOP a chybové kontrolky.

Při aktivaci senzoru upnutí a stisku tlačítka START se zapne vrtání a pohon pro pohyb vrtacího ramene směrem dolů. Při dosažení dolní polohy (detekováno dolním koncovým spínačem) se pohon dolů musí vypnout. V této poloze vrtací rameno setrvá jednu sekundu s aktivním vrtáním. Následně se aktivuje pohon pro pohyb vrtacího ramene směrem vzhůru a zůstane aktivní, dokud nesepne horní koncový spínač.

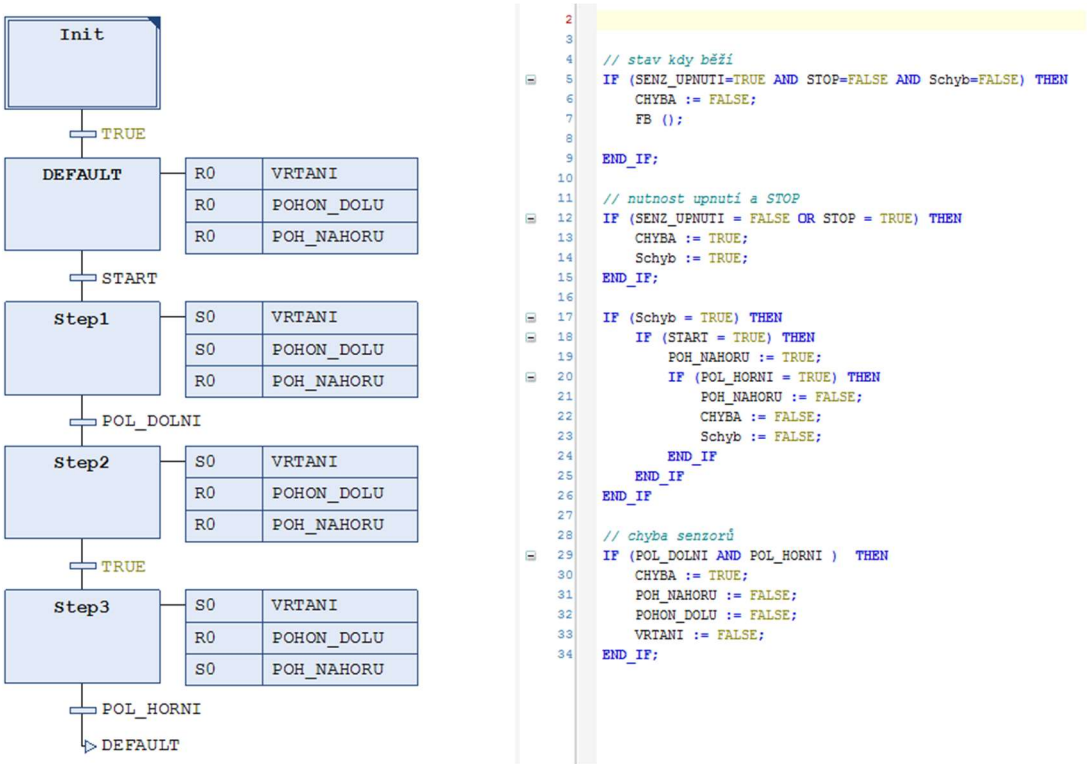
Po dosažení horní polohy se vypne vrtání. Pohony pro pohyb vrtacího ramene směrem nahoru a dolů musí být proti sobě blokovány, aby nemohlo dojít k současnému sepnutí. Při současném sepnutí horního i dolního koncového spínače se musí rozblikat chybová kontrolka a musí se zastavit pohony. Při stisknu tlačítka STOP se musí proces zastavit v jakékoliv fázi. Stejně tak, pokud senzor správného upnutí přestane být aktivní. Při následném stisku tlačítka START se vrtačka vrátí do horní výchozí polohy, kde čeká na správné upnutí a opětovným stiskem tlačítka START je zahájen nový proces vrtání. Ukázky programu a vizualizace jsou na obrázcích 6-7 a 6-6.

Příklad postupu řešení dané úlohy:

a) Spusťte SoMachine, založte nový projekt a zvolte vhodný programovací jazyk.
b) Definujte hlavní vstupní proměnné START, STOP, SENZ_UPNUTI, POL_HORNI a POL_DOLNI.
c) Definujte hlavní výstupní proměnné VRTANI, POH_NAHORU, POH_DOLU a CHYBA.
d) Jestliže je SENZ_UPNUTI sepnut, tlačítkem START spusťte VRTANI a POH_DOLU.
e) Jakmile vrtací rameno dosáhne dolní polohy (detekováno pomocí POL_DOLNI), vypne se POH_DOLU a za jednu sekundu se zapne POH_NAHORU.
f) Až vrtací rameno dosáhne horní polohy (detekováno pomocí POL_HORNI), vypne se POH_NAHORU a VRTANI.
g) Musí být provedena vzájemná blokace POH_NAHORU a POH_DOLU.
h) Musí být ošetřeny chyby senzorů. Pakliže jsou současně sepnuty POL_HORNI a POL_DOLNI, musí tento jev být vyhodnocen jako chyba a vrtací rameno nesmí jít zapnout. Tato chyba bude indikována blikáním chybové kontrolky CHYBA a vyhodnocena stejně jako stisk tlačítka STOP.
i) Program musí jít vždy přerušit tlačítkem STOP. Po následném krátkém stisku tlačítka START vyjede vrtačka do výchozí polohy a čeká na opětovný stisk tlačítka START, který započne další cyklus vrtání.
j) Pokud při vrtacím procesu dojde k uvolnění výrobku (signalizováno neaktivním senzorem upnutí) musí být proces přerušen, stejně jako při stisku tlačítka STOP.
k) Vytvořte přehlednou a jednoduchou vizualizaci k tomuto procesu.



Obr. 6-6: Ukázka vizualizace pro laboratorní úlohu 5




```
2
3
4   // stav kdy běží
5   IF (SENZ_UPNUTI=TRUE AND STOP=FALSE AND Schyb=FALSE) THEN
6       CHYBA := FALSE;
7       FB ();
8
9   END_IF;
10
11  // nutnost upnutí a STOP
12  IF (SENZ_UPNUTI = FALSE OR STOP = TRUE) THEN
13      CHYBA := TRUE;
14      Schyb := TRUE;
15  END_IF;
16
17  IF (Schyb = TRUE) THEN
18      IF (START = TRUE) THEN
19          POH_NAHORU := TRUE;
20          IF (POL_HORNI = TRUE) THEN
21              POH_NAHORU := FALSE;
22              CHYBA := FALSE;
23              Schyb := FALSE;
24          END_IF
25      END_IF
26  END_IF
27
28  // chyba senzorů
29  IF (POL_DOLNI AND POL_HORNI )  THEN
30      CHYBA := TRUE;
31      POH_NAHORU := FALSE;
32      POHON_DOLU := FALSE;
33      VRTANI := FALSE;
34  END_IF;
```

Obr. 6-7: Ukázka programu v SFC a ST pro laboratorní úlohu 5

6.6 Laboratorní úloha 6: Regulace intenzity osvětlení

Zadání:

Vytvořte řídící program pro regulaci intenzity osvětlení ovládané pomocí vizualizace. Podle nastavené požadované hodnoty intenzity osvětlení regulátor pomocí výstupního signálu PWM reguluje soustavu s žárovkou jakožto zdroj osvětlení na požadovanou hodnotu.

Ve vizualizaci se zapnutím přepínače připojí žárovka na napěťový signál. Přepnutím toho samého přepínače do původní polohy způsobí odpojení světelného zdroje od napěťového signálu. Následně nastavením požadované hodnoty intenzity osvětlení ve vizualizaci se zadá požadovaná hodnota pro regulátor. Intenzita osvětlení je měřena luxmetrem, který má na svém výstupu napěťový signál. Tento signál je třeba škálovat pro použití v regulační soustavě. Nastavení regulátoru se provede metodou Ziegler--Nicholson.

Příklad postupu řešení dané úlohy:

a) Spusťte SoMachine, založte nový projekt a zvolte vhodný programovací jazyk.
b) Definujte hlavní vstupní proměnné ON, OSVETLENI a OSVETLENI_AKT.
c) Definujte hlavní výstupní proměnnou VYSTUP.
d) Vložte PID regulátor a PWM generátor. Na jejich vstupy a výstupy přiřaďte správně škálované hodnoty (VYSTUP, OSVETLENI a OSVETLENI_AKT).
e) Zajistěte při deaktivaci vstupní proměnné ON, aby se zdroj osvětlení nerozsvítil.
f) Vytvořte vizualizaci, která bude obsahovat zapínací přepínač, okno, do kterého lze zapisovat požadovanou hodnotu a prvek, který bude zobrazovat aktuální hodnotu osvětlení.

6.7 Laboratorní úloha 7: Regulace teploty v nádržce

Zadání:

Vytvořte řídící program pro regulaci teploty v nádržce s vodou ovládanou pomocí vizualizace. Podle požadované hodnoty teploty vody regulátor pomocí výstupního signálu PWM, přiváděného na topné těleso reguluje soustavu s nádrží a s vodou na požadovanou teplotu.

Ve vizualizaci se zapnutím přepínače připojí topné těleso na napěťový signál. Přepnutím toho samého přepínače do původní polohy způsobí odpojení topného tělesa od napěťového signálu. Požadovaná teplota vody se nastaví ve vizualizaci a tato hodnota je také použita jako požadovaná hodnota teploty pro regulátor. Teplota vody je měřena teplotním čidlem, který má na svém výstupu napěťový signál. Pro zpracování tohoto signálu je třeba jej škálovat. Nastavení regulátoru se provede metodou Ziegler--Nicholson.

Příklad postupu řešení dané úlohy:

a) Spusťte SoMachine, založte nový projekt a zvolte vhodný programovací jazyk.
b) Definujte hlavní vstupní proměnné ON, TEPLOTA a TEPLOTA_AKT.
c) Definujte hlavní výstupní proměnnou VYSTUP.
d) Vložte PID regulátor a PWM generátor. Na jejich vstupy a výstupy přiřaďte správně škálované hodnoty (VYSTUP, TEPLOTA a TEPLOTA_AKT).
e) Zajistěte při deaktivaci vstupní proměnné ON, aby se zdroj tepla nezapnul.
f) Vytvořte vizualizaci, která bude obsahovat zapínací přepínač, okno, do kterého lze zapisovat požadovanou hodnotu a prvek, který bude zobrazovat aktuální hodnotu teploty vody v nádržce.

6.8 Laboratorní úloha 8: Regulace otáček motoru a kontrola teploty vinutí

Zadání:

Vytvořte řídící program pro regulaci otáček motoru a kontrolu teploty vinutí na jednotlivých fázích stroje za pomocí vizualizace. Podle požadované hodnoty otáček motoru regulátor pomocí výstupního signálu PWM reguluje soustavu s motorem. Při naměření teploty 90 °C na jednotlivých vinutích motoru se spustí externí ventilátor a při naměření teploty 150 °C musí být motor okamžitě vypnut.

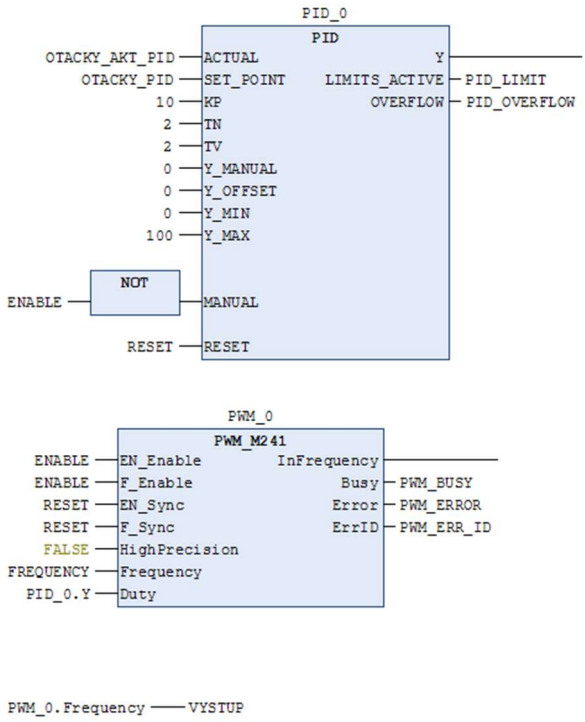
Ve vizualizaci zapnutím přepínače START do horní polohy se povolí spuštění motoru. Zapnutím přepínače STOP do horní polohy se motor okamžitě odpojí od zdroje energie. Požadovaná hodnota otáček se nastaví ve vizualizaci a tato hodnota se také použije jako požadovaná hodnota otáček pro regulátor. Hodnota otáček motoru je měřena otáčkoměrem a na svém výstupu má napěťový signál. Tento signál je pro zpracování potřeba škálovat. Nastavení regulátoru se provede metodou Ziegler-Nicholson. Pokud čidla teploty na jednotlivých vinutí naměří teplotu 90 °C, spustí se externí ventilátor. Pokud ovšem teplota přesáhne 150 °C, musí být motor okamžitě odpojen od zdroje energie, ale externí ventilátor musí být spuštěn, dokud teplota jednotlivých vinutí neklesne pod 90 °C. Ukázky části programu a vizualizace jsou na obrázcích 6-8 a 6-9.

Příklad postupu řešení dané úlohy:

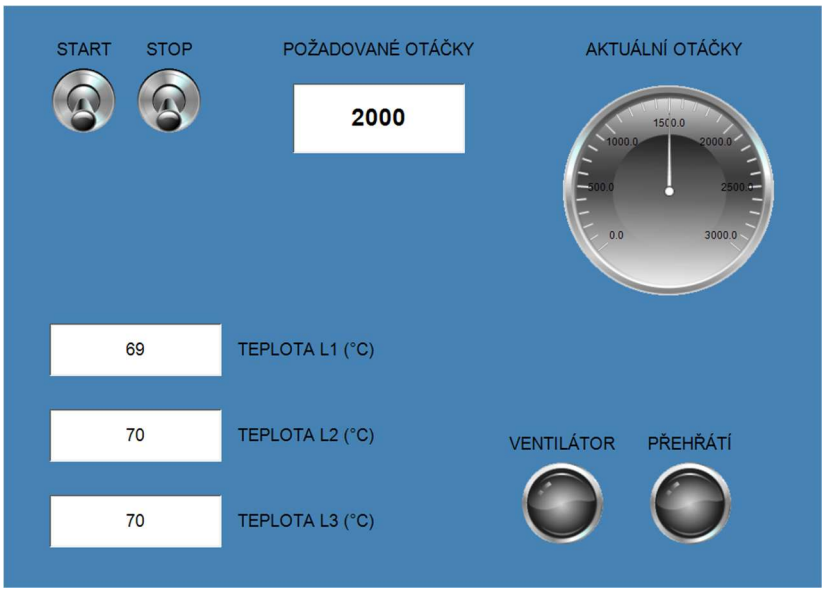
a) Spusťte SoMachine, založte nový projekt a zvolte vhodný programovací jazyk.
b) Definujte hlavní vstupní proměnné START, STOP, OTACKY, OTACKY_AKT, TEP_1, TEP_2 a TEP_3.
c) Definujte hlavní výstupní proměnné VYSTUP a VENTILATOR.
d) Vložte PID regulátor a PWM generátor. Na jejich vstupy a výstupy přiřaďte správně škálované hodnoty (VYSTUP, OTACKY a OTACKY_AKT).
e) Zajistěte při aktivaci vstupní proměnné STOP, aby se motor nespustil.
f) Vytvořte jednoduchou a přehlednou vizualizaci, ve které musí být numerický display, do kterého lze vpisovat hodnotu otáček, která se přes tento vizualizační prvek přiřadí k proměnné OTACKY.

g) Pokud TEP_1, TEP_2 a TEP_3 přesáhnou hodnotu teploty 90 °C zajistěte, aby se spustil VENTILATOR.

h) Definujte další podmínku pro spuštění motoru, že TEP_1, TEP_2 ani TEP_3 nesmí přesáhnout hodnotu teploty 150 °C.



Obr. 6-8: Ukázka části programu v FBD pro laboratorní úlohu 8



Obr. 6-9: Ukázka vizualizace pro laboratorní úlohu 8

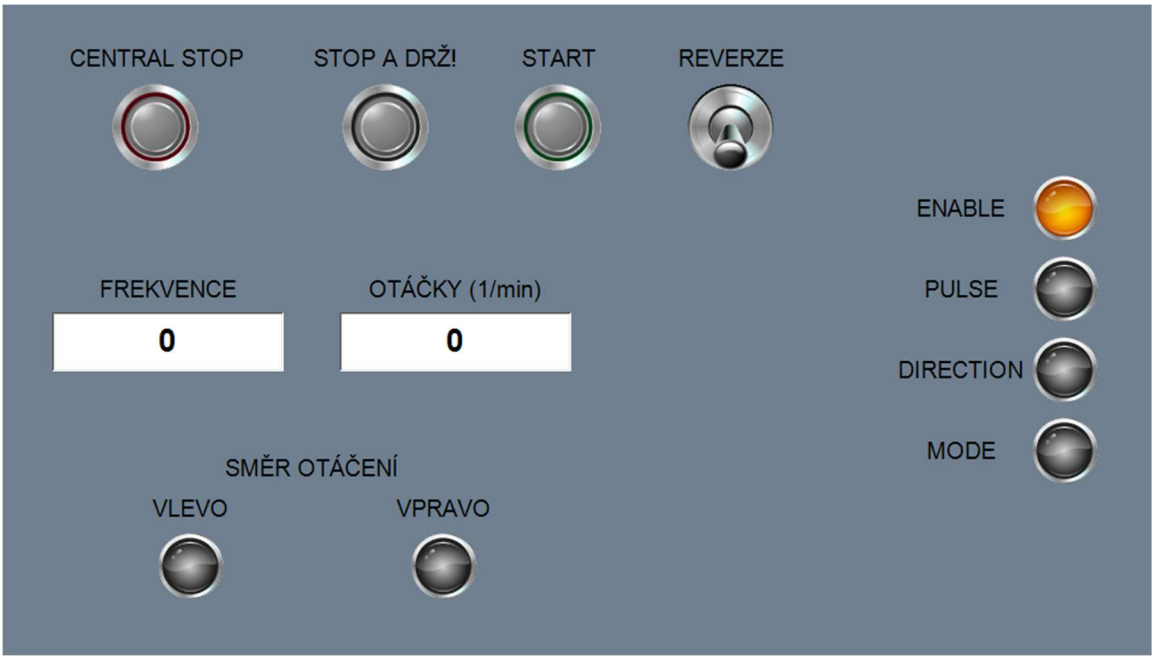
6.9 Laboratorní úloha 9: Řízení krokového motoru

Zadání:

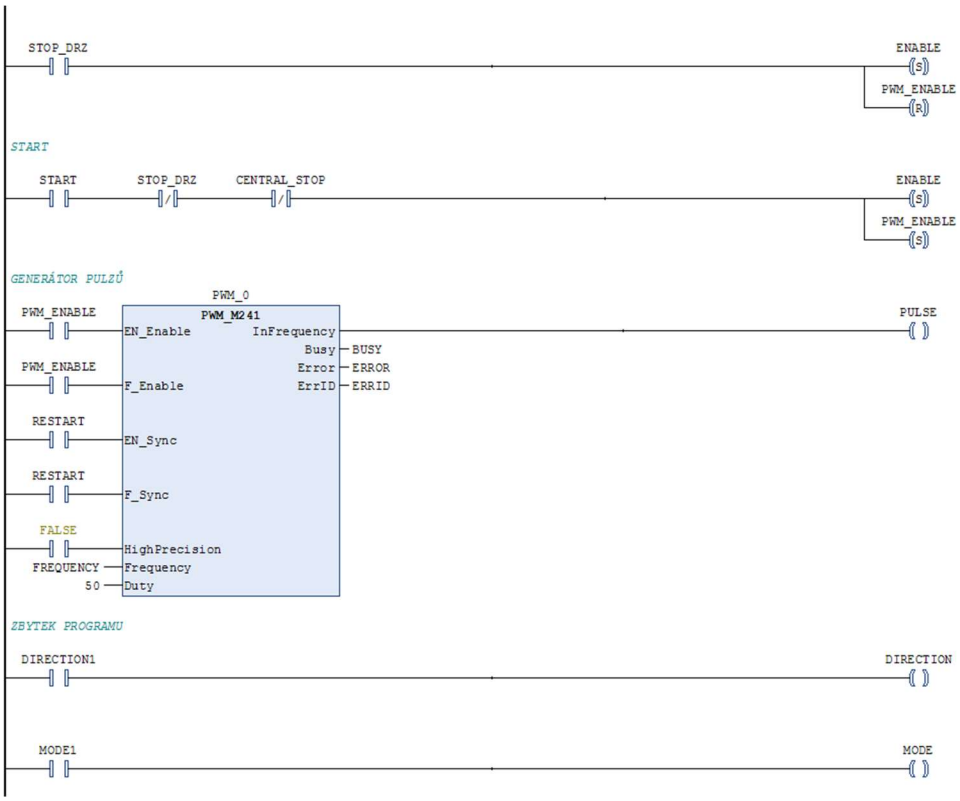
Vytvořte řídící program pro řízení krokového motoru. Pro řízení použijte standardní řídící jednotku. Při aktivním vstupu ENABLE na řídící jednotce je možné posílat pulsy PWM na vstup PULSE. Směr otáčení krokového motoru je dán aktivací nebo deaktivací vstupu DIRECTION. Vstup MODE přepíná mezi režimy fast a slow. Ovládání krokového motoru bude zajištěno pomocí vizualizace, do které lze vložit hodnotu frekvence, která bude přidělena jako vstupní proměnná do PWM generátoru. Touto frekvencí je také dána rychlost otáčení krokového motoru. Ukázka vizualizace, ze které vychází příklad postupu řešení dané úlohy, je na obrázku 6-10. Ukázky části programu a experimentálního pracoviště pro tuto úlohu jsou na obrázcích 6-11 a 6-12.

Příklad postupu řešení dané úlohy:

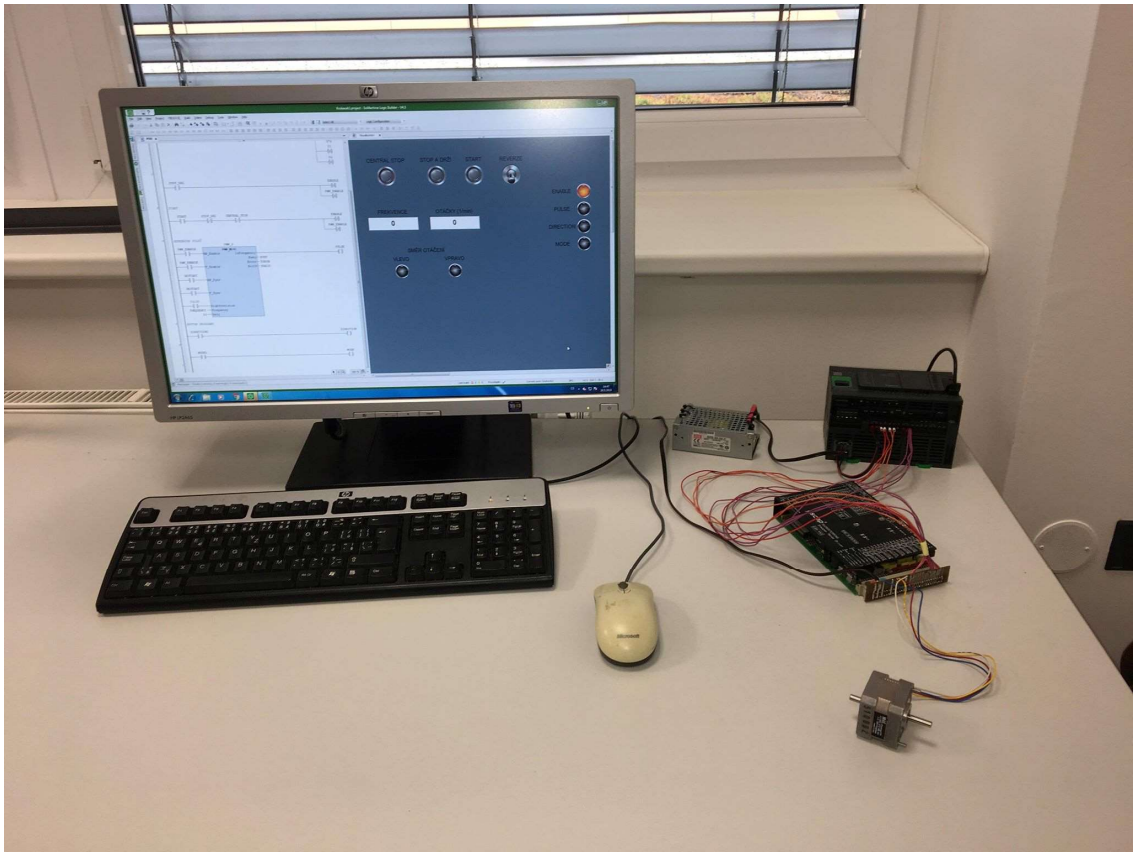
a) Spusťte SoMachine, založte nový projekt a zvolte vhodný programovací jazyk.
b) Definujte hlavní vstupní proměnné START a STOP.
c) Definujte hlavní výstupní proměnné ENABLE, PULSE, DIRECTION a MODE.
d) Vložte PWM generátor a na jeho vstupy přiřaďte povolovací proměnnou, frekvenci a nastavte střídu PWM. Výstup z PWM namapujte na rychlý výstup PLC (0 - 200kHz).
e) Vytvořte jednoduchou a přehlednou vizualizaci, ve které musí být numerický display (do kterého lze vpisovat hodnotu frekvence, která se přes tento vizualizační prvek přiřadí na vstup do PWM generátoru) a další potřebné prvky.
f) Při stisku tlačítka STOP A DRŽ! zajistěte, aby byl aktivní výstup ENABLE.
g) Při stisku tlačítka CENTRAL STOP zajistěte, aby se motor neotáčel a nebyl zabrzděn.
h) Při aktivaci přepínače REVERZE zajistěte, aby se aktivoval výstup DIRECTION a tím se změnil směr otáčení motoru.



Obr. 6-10: Ukázka vizualizace pro laboratorní úlohu 9



Obr. 6-11: Ukázka části programu pro laboratorní úlohu 9



Obr. 6-12: Experimentální pracoviště pro laboratorní úlohu 9

ZÁVĚR

Bakalářská práce se zabývala programovatelnými logickými automaty pro výuku. Cílem bylo uvést srovnání jednotlivých typů programovatelných logických automatů, navrhnout několik laboratorních úloh a vytvořit k nim příslušné laboratorní návody. V první části byla stručně rozebrána teorie této problematiky. Byly popsány základní programovací jazyky (IL, ST, FBD, LD a SFC), jejich rozlišení, a také popis s grafickým náhledem na tyto jazyky. Práce se dále věnovala programovacímu prostředí GX Works2 od Mitsubishi Electric a jeho podrobnému popisu, programovacímu prostředí SoMachine od výrobce Schneider Electric i s jeho vývojovým prostředím pro vizualizaci, a také programovacímu prostředí Automation Studio (B&R) a jeho popisem. Z vyjmenovaných programovacích prostředí bylo zvoleno programovací prostředí SoMachine zejména z důvodu jeho jednoduchosti, kompatibility, možnosti tvorby jednoduché vizualizace a vstřícné komunikace ze strany Schneider Electric.

V závěrečné části je navrženo devět laboratorních úloh. Laboratorní úlohy jsou řazeny dle obtížnosti. První úlohy mají za cíl seznámení s programovacím prostředím a s tvorbou jednoduchých programů. Poslední úlohy řeší složitější problematiku s pomocí PID regulátorů a PWM generátorů. Samozřejmostí je tvorba jednoduché, avšak přehledné vizualizace k některým laboratorním úlohám. Pro testování ovládání krokového motoru (laboratorní úloha 9) byla použita řídící jednotka ESCAP EDM-453 a bylo vytvořeno experimentální pracoviště. Na tomto pracovišti je možné nadále zlepšovat doposud nabyté zkušenosti s programovacím prostředím SoMachine a řízením krokového motoru. Navazující diplomová práce by se mohla věnovat problematice vytváření jednotlivých přípravků a pracovišť k laboratorním úlohám navržených v této práci.

LITERATURA

[1] ŠVARC, Ivan. *Základy automatizace* [online]. Brno, 2002 [cit. 2018-11-19]. Dostupné z: http://matlab.fei.tuke.sk/zar/subory/literatura/ZakladyAutomatizace_SVARC.pdf. Učební text. VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ Fakulta strojního inženýrství.

[2] KOCHANÍČEK, Ludvík. *Programovací jazyky pro PLC* [online]. [cit. 2018-11-19]. Dostupné z: https://coptkm.cz/portal/?doc=3905

[3] *Co by měl každý vědět o programovacích jazycích PLC* [online]. Elektroprůmysl, [cit. 2018-11-19]. Dostupné z: https://www.elektroprumysl.cz/software/co-by-mel-kazdy-vedet-o-programovacich-jazycich-plc

[4] MARČAN, Peter. *01 Programovanie PLC – Programovacie jazyky PLC* [online]. Slovensko: Daily Automation, s.r.o., [cit. 2018-11-19]. Dostupné z: http://dailyautomation.sk/01-programovacie-jazyky-plc/

[5] *Rodina FX: MELSEC PLC* [online]. Germany: Mitsubishi Electric Europe B.V., 2004 [cit. 2018-11-19]. Dostupné z: http://www.accs.cz/Files/FA/PLC/FX_Family_CZ.pdf

[6] *MITSUBISHNI ELECTRIC: Programmable Controllers MELSEC* [online]. Mitsubishi Electric Corporation, [cit. 2018-12-06]. Dostupné z: http://www.mitsubishielectric.com/fa/products/cnt/plc/index.html

[7] *Řada MELSEC FX: Programovatelné logické automaty příručka pro začátečníky* [online]. Germany: Mitsubishi Electric Europe B.V., 2009 [cit. 2018-11-19]. Dostupné z: http://www.accs.cz/Files/FA/PLC/FX_Prirucka_pro_zacatecniky.pdf

[8] *FX3GE Overview* [online]. Mitsubishi Electric Europe B.V., [cit. 2018-1119]. Dostupné z: http://www.accs.cz/Files/FA/PLC/FX3GE_Overwiew.pdf

[9] *GX Works2 Basics: PLC* [online]. Mitsubishi Electric Corporation, 2014 [cit. 2018-12-30]. Dostupné z: http://www.mitsubishielectric.com/fa/assist/e-learning/pdf/eng/1-GX_Wks2_Basics_na_eng-A.pdf

[10] *Q-series basic course (for GX Works2): Training manual* [online]. JAPAN: Mitsubishi Electric Corporation, 2012 [cit. 2018-12-30]. Dostupné z: http://dl.mitsubishielectric.com/dl/fa/document/manual/school_text/sh081123eng/sh081123enga.pdf

[11] *TM238LDD24DT: Product data sheet* [online]. Schneider Electric Inc.., 2012 [cit. 2018-11-24]. Dostupné z: https://docs-emea.rs-online.com/webdocs/116c/0900766b8116cdf4.pdf

[12] HOWLETT, Bruce. *Getting Started With SoMachine: Self Study Manual* [online]. Schneider Electric Inc.., 2014 [cit. 2018-11-20]. Dostupné z: https://www.schneider-electric.com/resources/sites/SCHNEIDER_ELECTRIC/content/live/FAQS/288000/FA288633/en_US/Training%20Manual%20EN.pdf

[13] *Human/Machine Interface: HMI configuration software* [online]. Schneider Electric Inc.., 2014 [cit. 2018-12-29]. Dostupné z: http://download.schneider-electric.com/files?p_enDocType=Catalog&p_File_Name=DIA5ED2130614EN.pdf&p_Doc_Ref=DIA5ED2130614EN

[14] IRONS, Richard. *SoMachine Basic: Training Manual* [online]. Schneider Electric Inc.., 2013 [cit. 2018-12-23]. Dostupné z: https://download.schneider-electric.com/files?p_enDocType=Training+material&p_File_Name=SoMachine+Basic_v1.1_Training_Manual.pdf&p_Doc_Ref=SoMachineBasicTutorial_Manual

[15] *B&R Automation Studio Quick Start: B&R AUTOMATION STUDIO* [online]. Bernecker + Rainer Industrie−Elektronik Ges.m.b.H., 2003 [cit. 2019-01-01]. Dostupné z: http://instrumentacionycontrol.net/wp-content/uploads/2017/11/IyCnet_BR_AutomationStudio_QuickStart_MASYS2ASQS-E_240.pdf

[16] *Br-automation: Automation Studio* [online]. BRNO: B&R Headquarters, [cit. 2019-01-01]. Dostupné z: https://www.br-automation.com/cs/produkty/software/automation-studio/

Seznam zkratek

CFC	…	Nepřetržitý funkční diagram (Continuous Function Chart)
DC	…	Stejnosměrné napětí (Direct Current)
EEPROM	…	Elektronicky vymazatelná paměť pouze pro čtení (Electrically Erasable Programmable Read-Only Memory)
FBD	…	Diagram funkčních bloků (Function Block Diagram)
HMI	…	Rozhraní mezi člověkem a strojem (Human-Machine Interface)
IL	…	Jazyk instrukcí (Instruction List)
I/O	…	Vstup/Výstup (Input/Output)
LD	…	Příčkový diagram (Ladder Diagram)
LED	…	Světlo emitující dioda (Light Emitting Diode)
PAC	…	Programovatelný automatizovaný kontrolér (Programmable Automation Controller)
PID	…	Proporcionální, integrační a derivační (Proportional, Integrative and Derivative)
PLC	…	Programovatelný logický kontrolér (Programmable Logic Controller)
PWM	…	Pulzně-šířková modulace (Pulse With Modulation)
RAM	…	Paměť s náhodným přístupem (Random-Access-Memory)
RS422	…	Sériová komunikace
SFC	…	Sekvenční funkční diagram (Sequential Function Chart)
ST	…	Strukturovaný text (Structured Text)
USB	…	Univerzální sériová sběrnice (Universal Serial Bus)