ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

FAKULTA ELEKTROTECHNICKÁ

KATEDRA ELEKTROTECHNOLOGIE

Bakalářská práce

Školící robotické pracoviště

Ondřej Bělovský

vedoucí práce: Bc. Martin Hradecký

studijní program: Elektrotechnika, energetika a management
obor: Aplikovaná elektrotechnika

Praha, květen 2020

Čestné prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze dne _____ _____

podpis

Poděkování

Tímto bych chtěl poděkovat mému vedoucímu bakalářské práce Bc. Martinu Hradeckému, který mi velice pomohl a poradil při tvorbě této práce. Dále bych chtěl poděkovat firmě B:TECH, a.s., která mi poskytla dané robotické pracoviště, na kterém jsem mohl testovat program. Díky patří také Ing. Karlu Künzelovi, CSc. za poskytnuté konzultace a pomoc s formální stránkou bakalářské práce.

Abstrakt

Cílem bakalářské práce je vytvořit program pro ukázkovou aplikaci na robotickém pracovišti ve firmě B:TECH, a.s. V teoretické části se budu zabývat obecnou problematikou PLC, HMI panelů, robotů a jejich vzájemnou komunikací. V praktické části bude popsán postup při tvorbě programu a při uvádění robotického pracoviště do provozu. Dále bude následovat zhodnocení úlohy a rozdíl mezi simulací a reálným pracovištěm.

Abstract

The aim of the bachelor's thesis is to create a program for a sample application at a robotic workplace in the company B: TECH, a.s. In the theoretical part I will deal with general issues of PLC, HMI panels, robots and their mutual communication. The practical part will describe the procedure for creating a program and putting the robotic workplace into operation. This will be followed by an evaluation of the task and the difference between the simulation and the real workplace.

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Bělovský** Jméno: **Ondřej** Osobní číslo: **474389**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra elektrotechnologie**

Studijní program: **Elektrotechnika, energetika a management**

Studijní obor: **Aplikovaná elektrotechnika**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Školící robotické pracoviště

Název bakalářské práce anglicky:

Training robotic stand

Pokyny pro vypracování:

1. Navrhněte koncept robotického pracoviště pro školící účely ve společnosti B:Tech.
2. Nastudujte možnosti komunikace mezi PLC, HMI a robotem v rámci navrženého pracoviště.
3. Naprogramujte řídící PLC algoritmus včetně vizualizace pro komplexní ovládání robota v rámci vybrané školící aplikace
4. Sestavte školící pracoviště
5. Ověřte funkčnost pracoviště a zpracujte odpovídající dokumentaci

Seznam doporučené literatury:

[1] FX Series Programmable Controllers, Programming manual, Manual number JY992D48301, revision J, November 1999, Mitsubishi Electric Corporation
[2] Mitsubishi Programmable Logic Controller, Training Manual, Q-series Basic course, Mitsubishi Electric Corporation
[3] Programovatelné logické automaty Řada MELSEC FX, Příručka pro začátečníky, 2009, Mitsubishi Electric Corporation
[4] Mitsubishi Indurtrial Robot CR1... Instruction Manual, 2009, 2009, Mitsubishi Electric Corporation
[5] MELFA Industrial Robots, Instruction Manual, Mitsubishi Electric Corporation
[6] GT Designer2 Version2 Operating Manual, Mitsubishi Electric Corporation
[7] Screen design software SWDND-GTWD3-E, GT Designer3 Screen Design Manual, Mitsubishi Electric Corporation

Jméno a pracoviště vedoucí(ho) bakalářské práce:

Bc. Martin Hradecký, B:TECH, a.s.

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) bakalářské práce:

Datum zadání bakalářské práce: **31.01.2020** Termín odevzdání bakalářské práce: _____

Platnost zadání bakalářské práce: **30.09.2021**

Bc. Martin Hradecký podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Mgr. Petr Páta, Ph.D. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Student bere na vědomí, že je povinen vypracovat bakalářskou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v bakalářské práci.

. _____ Datum převzetí zadání

_____ Podpis studenta

Obsah

1 Úvod

Cílem této bakalářské práce je vytvoření funkčního programu pro vybranou školící aplikaci na robotickém pracovišti. Tato práce se zabývá problematikou programování programovatelných logických automatů, robotů, tvorbou vizualizace pro ovládání na HMI panelech a také možnostmi komunikace mezi těmito moduly.

PLC jsou dnes již velmi rozšířené především v průmyslové automatizaci, kde řídí jednotlivé výrobní procesy. Jedná se, dle mého názoru, o velmi perspektivní přístroje, protože s každým dalším vývojem dochází k rozšíření možnosti použití. K logickým automatům se přímo vážou HMI panely, které umožňují rozhraní obsluha – stroj. U těchto panelů došlo k velkému pokroku od prvních rozhraní. Nejstaršími předchůdci HMI panelů byly kombinace tlačítek, přepínačů a LED diod, které zobrazovaly stavy stroje. V dnešní době se vyrábějí různé druhy panelů. Ať už se liší odolností vůči vlivům prostředí, nebo možnostmi ovládání, velikostí, apod. Dnešní nejpokročilejší panely umožňují tvorbu SCADA systému, který je schopen komplexnějšího ovládání.

Další část bakalářské práce se bude zabývat problematikou robotů. Roboti jsou dalším velmi významným strojem používaným v průmyslové automatizaci. Tito roboti mohou vykonávat několik různých činností a umožňují komunikaci s nadřazeným řídícím systémem (PLC). V dnešní době existuje mnoho typů robotů s různými vlastnostmi.

Komunikace všech komponentů robotického pracoviště je také důležitým bodem, protože jinak by jednotlivé moduly nebyly schopné si vyměňovat data.

2 Teoretická příprava

2.1 PLC

2.1.1 Co je PLC?

Programovatelný logický automat, angl. *Programmable Logic Controller* (PLC) je průmyslový počítač používaný zejména pro automatizaci výrobních linek nebo třeba řízení strojů. Základní vlastností PLC je, že program se vykonává v cyklech, to znamená, že se vykoná hlavní funkce a když dojde na konec, tak se spustí znovu s přepsanými vstupními hodnotami. [1]

PLC můžeme rozdělit do dvou skupin – kompaktní a modulární. Kompaktní PLC je typické tím, že CPU obsahuje vstupy, komunikační bloky a někdy i zdroj. Modulární PLC nemá všechno v CPU, ale všechny komponenty jsou v jednotlivých modulech, mezi kterými je potřeba obstarat komunikaci. [1]

Obr. 1 - Ukázka modulárního PLC od firmy Mitsubishi Electric řady iQ-R s přídavnými moduly [2]

2.1.2 Historie PLC

První logické automaty vznikly v 60. letech 20. století jako náhrada reléových řídících systémů. Oproti reléovým systémům mohly PLC lépe reagovat na změny v systémech a mají také podstatně menší rozměry. V roce 1969 byl vyroben první logický automat, který byl vyroben společností Bedford Associates.

Dalším významným producentem logických automatů se staly společnosti Allen Bradley, ABB, Mitsubishi Electric a na evropském trhu se největšího rozšíření dostalo řadě S5

společnosti Siemens. Siemens patří do dnešní doby k největším výrobcům této technologie. [3]

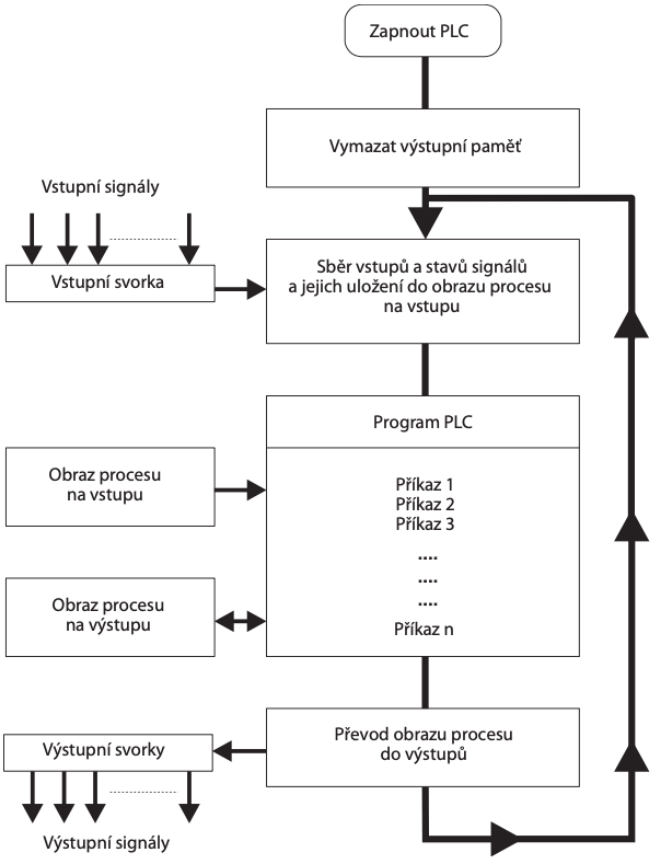
2.1.3 Zpracování programu PLC

PLC vykonává instrukce podle provedení programu, který je psán v nějakém vývojovém prostředí. Změny ve výrobním procesu, který PLC řídí se provádí jednoduše přepsáním programu. To je velká výhoda oproti reléovým systémům, kde byl potřeba zásah do hardwaru. Logické automaty zpracovávají instrukce sekvenčně. Jednotlivé instrukce se vykonávají jedna po druhé a když dojde program na konec, tak se celá sekvence opakuje, tzv. cyklické zpracování. Schéma jak funguje PLC je na obr. 2.

Oproti mikroprocesorovému řízení má jednu odlišnost. Stavy všech proměnných se přečtou před vykonáním cyklu, následuje vykonání jednoho cyklu programu a po dokončení jednoho cyklu se najednou přepíší.

Vykonávání programu můžeme rozdělit do 3 kroků (vstupní krok, krok zpracování a výstupní krok). Ve vstupním kroku se přečtou všechny signály vstupů, které přicházejí jako logické stavy, program je dále předá do kroku zpracování. Při zpracování se signály ze vstupního kroku kombinují a upravují pomocí logických a jiných funkcí, tak abychom dosáhli požadovaných výstupních hodnot. A v posledním, výstupním kroku jsou logické stavy převedeny na signály, které řídí proces (stykače, LED, …). Mezi hlavní periferie PLC patří zejména digitální vstupy (DI), výstupy (DO) a analogové vstupy (AI) a výstupy (AO). Analogové vstupy a výstupy slouží zejména pro zpracování spojitého signálu, oproti tomu digitální I/O jsou používány pouze pro signalizaci vypnuto/zapnuto, jsou to vlastně proměnné typu BOOL (pouze 0 a 1). [4]

Obr. 2 popisuje jak program pracuje se vstupy a výstupy. Program nepracuje přímo s fyzickými vstupy, ale vstupy se nejdříve uloží do přechodné paměti a tím vytvoří obraz procesu na vstupu. Podobně to platí také pro výstupy. Po vykonání programu se výstupy uloží do obrazu procesu na výstupu a teprve potom se všechny najednou přenesou na fyzické výstupy.



Obr. 2 - Postup zpracování programu logickým automatem [3]

2.2 Programování dle normy ČSN EN 61131-3

Při programování PLC je vhodné postupovat podle normy IEC EN 61131-3, která popisuje obecnou metodiku programování logických automatů. Norma nicméně upozorňuje, že i při dodržení postupů v ní popsaných není zaručeno, že takto napsaný program bude plně funkční. Funkčnost programu je podmíněna správným algoritmem, který píše a vymýšlí programátor podle funkce, kterou má PLC vykonávat. Tato norma pouze popisuje postupy jakými by se měl programátor řídit a sjednocuje programovací jazyky vhodné pro PLC programování.

Při vývoji programů můžeme využívat PLC od různých výrobců a tím i různá vývojová prostředí. Díky této normě je přechod mezi různými prostředími jednoduchý a zápis zůstává skoro stejný. [6]

2.2.1 Program Organization Unit (POU)

Základním pojmem v normě je tzv. Program Organization Unit (programová organizační jednotka, POU). Programové organizační jednotky tvoří buď programátor nebo jsou dodávány

výrobcem řídícího systému i s vývojovým prostředím (např. knihovny funkcí a funkčních bloků). Funkce, funkční blok a program jsou 3 základní typy POU. Celý program se skládá z jednotlivých POU, které si mezi sebou mohou předávat parametry. [3]

2.2.1.1 Funkce

Jedná se o nejjednodušší typ POU. „Zpracovává vstupní parametry a předává jediný výstupní parametr – ten je závislý jen na aktuální hodnotě vstupních parametrů, nezávisle na dřívějších aktivacích nebo na čase“. [6]

Kdybychom se na funkci dívali z klasického programátorského hlediska, tak funkce je vlastně podprogram hlavního programu (Main). Mezi standardní (knihovní) funkce patří například sčítání, odčítání, odmocnina atd. Jak je známo z matematiky, tak u těchto funkcí není třeba znát předchozí stav parametrů, což je základní vlastností funkce. V případě, že se použije již deklarovaná funkce, tak se nevytvoří její kopie v programu, ale pouze se jejím voláním program přesune do místa její deklarace, provede se funkce, zapíší se výstupní parametry a program pokračuje dál od místa volání. [5] [6]

2.2.1.2 Funkční blok

Tento typ POU je již o něco propracovanější než funkce. Rozdíl oproti funkcím je ten, že funkční blok může uchovávat ve svých vnitřních proměnných informace o stavu z předchozích volání a rozdíl je také při volání funkčního bloku. Abychom mohli funkční blok vůbec volat, tak musíme vytvořit v programu jeho deklaraci – to je stejné jako u funkce. Zatímco u funkce se program pouze přesune do místa deklarace a přepíše své výstupní parametry, tak u funkčního bloku se navíc ještě v místě každého volání provede tzv. *instance,* což je soubor proměnných s konkrétními názvy proměnných a parametrů pro dané volání funkčního bloku. Ve výsledku je funkční blok v paměti deklarován stejně jako funkce, při každém volání program vykoná soubor instrukcí ve funkčním bloku, ale výstupní parametry se zapíší pokaždé do jiné instance. Všechny již deklarované funkční bloky můžeme dále používat ve všech dalších blocích. Mezi funkční bloky patří např. časovače, čítače nebo regulační smyčka či PID regulátor. [5] [6]

2.2.1.3 Program

„Program je v normě definován jako „logický souhrn programovacích jazyků a konstrukcí nutných pro zamýšlené zpracování signálů, které je vyžadováno pro zamýšlené stroje nebo procesu systémem programovatelného automatu““. [6]

Jedná se o logické uspořádání funkcí a funkčních bloků. Program může být zapsán stejně jako funkce nebo funkční blok jazyky danými normou. Řídící jednotka PLC vykonává pouze program (Main), ve kterém voláme funkce a funkční bloky. [5] [6]

2.2.2 Datové typy

Typy dat jsou předdefinované a dělí se většinou podle délky nebo rozsahu hodnot. Při logickém programování nejčastěji využíváme dvouhodnotové veličiny, které mají velikost 1 bit a je pro ně určen formát BOOL (někdy se jim také říká bitové nebo booleovské). Další datové typy jsou odvozeny ze základní velikosti 1 bit. Jsou to například číselné datové typy *BYTE* (velikost 8 bit), *WORD, INTEGER* (16 bit), *REAL* (32 bit) nebo třeba *LREAL* (REAL s dvojitou přesností). Existují i další datové typy upřesňující předchozí typy jako třeba *SINT* (krátký integer, 8 bit), *UINT* (INT bez znaménka) a další.

Datový typ *STRING* je určen pro řetězce znaků nebo existují také časové datové typy proměnných. Mezi časové datové typy patří např. *TIME* (pro trvání času) nebo *DATE* (pro kalendářní datum). [5] [6]

2.2.3 *Programovací jazyky*

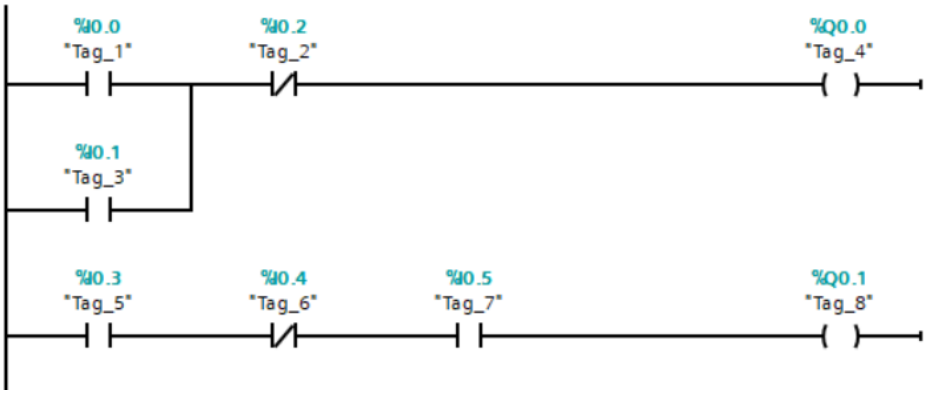
Program vykonává jednotlivé programové příkazy v pořadí, daném programátorem. Programové příkazy mohou být zapsány několika různými způsoby. Všechny způsoby programování logických automatů mají společné to, že pracují se stejnými datovými typy a stejnými funkcemi, které jsou v každém programovacím jazyce, pouze jsou zapsány různými způsoby. Všechny možné programovací jazyky, které jsou použitelné při programování PLC popisuje norma ČSN EN 61 131-3. [7]

2.2.3.1 Jazyk kontaktních schémat (Ladder Diagram, LD)

Jedná se o jakési zjednodušené schéma liniového schématu, používaného v elektrotechnice. LD je grafický programovací jazyk, který se využívá pro programování automatů již od konce 2. světové války. Původně obsahoval základní prvky, mezi které se řadí

spínací kontakty (NO – normally open), *rozpínací kontakty* (NC – normally closed), *výstupní relé, časovače* a *čítače*. Při rozvoji mikroprocesorové techniky byla možnost zařazení dalších prvků (např. kontakty reagující na náběžnou popř. sestupnou hranu).

Hlavními částmi jsou dvě svislé příčky, mezi které se píše samotný program. Jedna cívka reprezentuje hladinu nulového napětí a druhá většinou potenciál +24 V. Programové příkazy se píší zleva doprava a jsou vykonávány ze shora dolů. V levé části programových příkazů se nacházejí sériově paralelní kombinace kontaktů vyjadřující podmínky a v pravé části jsou výstupní cívky nebo bloky.



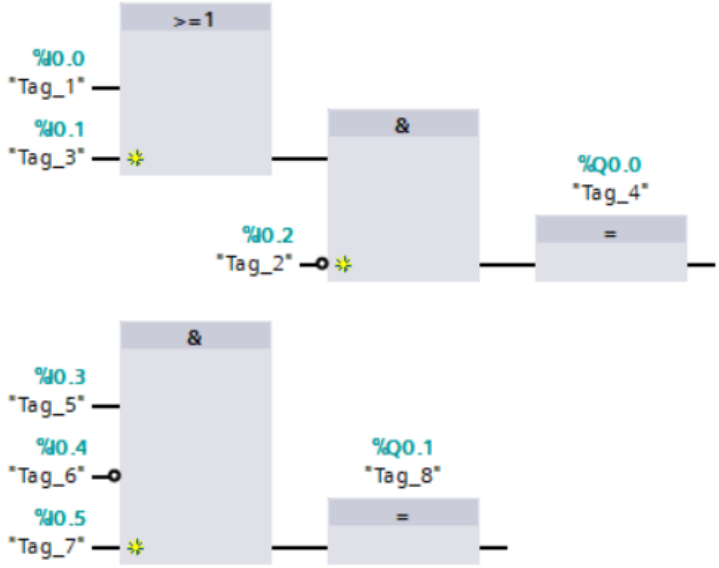
Obr. 3 - Ukázka programovacího jazyka LD

Na obr. 3 je vidět ukázka programu psaného pomocí kontaktních schémat. První programový příkaz obsahuje 2 NO, 1 NC kontakt a výstupní cívku s přímou funkcí. Když sepneme první spínací kontakt %I0.0 nebo %I0.1, tak se aktivuje výstup %Q0.0. Vypnout ho můžeme buď vypnutím jednoho ze dvou spínacích kontaktů nebo sepnutím rozpínacího kontaktu %I0.2. Druhý programový příkaz se skládá pouze ze 3 kontaktů (2 NO a 1 NC) v sérii a výstupní cívky s přímou funkcí. Aktivace výstupu %Q0.1 nastane pouze v případě, že jsou aktivované spínací kontakty s adresami %I0.3 a %I0.5 a rozpínací kontakt %I0.4 je rozepnutý. Změna stavu jakéhokoliv vstupu způsobí deaktivaci výstupu %Q0.1. [5] [6]

2.2.3.2 Jazyk funkčních bloků (Function Block Diagram, FBD)

I tento grafický jazyk popisuje již zmíněná norma. Skládá se z logických funkcí, které jsou vkládány do programu ve tvaru obdélníků s příslušným významem. Jednotlivé bloky jsou propojeny čárami (spojovacími vodiči).

Ukázka programu v jazyce funkčních bloků je na obr. 4 a je to stejný program jako na obr. 3. Je vidět, že se výstup jednotlivým bitům přiřadí obdélníkem se znaménkem = a adresou bitu. Popisky v uvozovkách jsou jména proměnných. [7] [8]



Obr. 4 - Ukázka programu v jazyku funkčních bloků

2.2.3.3 Instruction List (IL)

Program psaný v IL (Posloupnost příkazů) se skládá z posloupnosti základních operací, navíc dává programátorovi absolutní kontrolu nad programem. Hodí se pro programování jednoduchých, ale i důležitých funkcí. Většinou se pomocí IL píší krátké podprogramy, které jsou volané v hlavním programu. Z hlediska rychlosti a využití paměti je tento typ úspornější než předchozí 2 programovací jazyky. Mezi jeho nevýhody patří např. špatná přehlednost a orientace v programu, potřeba znalosti registrů a mnoha příkazů nebo, že tento typ programování vyžaduje mnoho psaní, oproti grafickým jazykům. [9]

Na obr. 5 je ukázka programu v Instruction List, který vyjadřuje stejný program, jako ukázky programů v grafických jazycích na obr. 3 a 4.

```
1       O     "Tag_1"                %I0.0
2       O     "Tag_3"                %I0.1
3       AN    "Tag_2"                %I0.2
4       =     "Tag_4"                %Q0.0
5
6       A     "Tag_5"                %I0.3
7       AN    "Tag_6"                %I0.4
8       A     "Tag_7"                %I0.5
9       =     "Tag_8"                %Q0.1
```

Obr. 5 - Ukázka programu v IL

2.3 Ovládací panely

Tyto panely představují rozhraní člověk – stroj, proto se také tyto panely většinou označují *HMI* (**H**uman **M**achine **I**nterface). Existuje také označení *GOT* (**G**raphic **O**peration **T**erminal) používané především firmou Mitsubishi Electric. HMI zprostředkovávají komunikaci mezi obsluhou a obsluhovaným strojem. Pomocí nich se nechá stroj ovládat nebo může zobrazovat informace o stroji. HMI se propojí nejčastěji ethernetovým kabelem s PLC a komunikace probíhá pomocí některého z komunikačních protokolů (PROFINET, CC-Link, …).

Obr. 6 – Ovládací panely od firmy Siemens [10]

2.4 Roboti

Dalším systémem, který je zahrnut ve školícím pracovišti je robotický manipulátor (viz obr. 7). Tito roboti se používají zejména k automatizaci opakované činnosti a mohou být řízeny logickým automatem. Každý robot má vlastní ovládací modul, který komunikuje s PLC. Komunikace probíhá stejně jako v případě HMI panelů. Do ovládacího modulu robota se ukládají programy pro robota a pomocí tohoto modulu jsou také ovládány jednotlivé osy robota. Z řídícího systému přicházejí signály na start programu, parametry pro program. Naopak do řídícího PLC se posílají např. signály, že robot dokončil program nebo signály o stavu robota. Tato komunikace je důležitá především kvůli řízení programu v závislosti na stavu robota. Každý robot má k dispozici tzv. pendant, který umožňuje ovládat robota manuálně (výběr programu, učení pozic, parametrizace apod.)

Obr. 7 - Robot na školícím pracovišti

2.5 Možnosti komunikace

Přenos informací a signálů mezi jednotlivými komponenty může probíhat několika způsoby. Způsoby komunikace se liší podle výrobců PLC příp. robotů.

2.5.1 PROFIBUS

PROFIBUS je komunikační sběrnice, která se začala používat v roce 1987 při automatizaci výrobních linek. V dnešní době je tato sběrnice standardizována 2 normami – IEC 61158 a IEC 61784. PROFIBUS DP a PROFIBUS PA patří mezi nejrozšířenější variantu tohoto komunikačního protokolu. Připojování k tomuto protokolu se realizuje stíněnými kabely s konektory RS 485

a. PROFIBUS DP

Tento komunikační protokol se může objevit v zařízeních průmyslové automatizace nebo CNC strojích. Kabel pro tuto komunikaci je tvořen dvěma vodiči, které mají různou rychlost přenosu dat a stíněním. Všechna zařízení mají zajištěné vlastní napájení. [12]

b. PROFIBUS PA

Tento typ PROFIBUSu se používá zejména tam, kde je zapotřebí sbírat a přenášet data na velké vzdálenosti, ale není potřeba tak velká rychlost čtení/zápisu dat. Rozdíl oproti PROFIBUS DP je ten, že data jsou přenášena pouze jednou rychlostí, po těchto vodičích také vede napájení zařízení připojených na PROFIBUS PA. [12]

2.5.2 PROFINET

PROFINET je nezávislý komunikační standard pro průmyslovou automatizaci. Je to nejpokročilejší řešení propojení celých automatizačních procesů. Tento standard vychází z Ethernetu, ale PROFINET je přímo určen pro průmyslové využití buď jako komunikace mezi stroji nebo jako komunikace mezi PLC a jeho periferiemi.

Pro nastavení zařízení a jeho připojení do sítě stačí obyčejné PC se síťovou kartou a softwarem pro tvorbu programu. PROFINET umožňuje vybudování rozsáhlejších sítí a oproti PROFIBUSu má také větší přenosovou rychlost, která dosahuje hodnoty 100 Mbit/s. Přenosová rychlost v síti PROFIBUS je omezena hodnotou 12 Mbit/s. Další velkou výhodou je, že po PROFINETU může běžet také klasická ethernetová komunikace. [11] [13]

2.5.3 CC-Link

Jedná se o vysokorychlostní průmyslovou síť, která dosahuje komunikační rychlosti až 10 Mb/s a umožňuje připojit až 64 stanic. Tato síť podstatně snižuje nároky na kabeláž a nabízí kompatibilitu mezi zařízeními od různých výrobců. CC-Link se hodí pro automatizaci výrobních procesů, které jsou rozprostřeny na velké ploše a je mezi řídicími jednotkami velká vzdálenost. Tato síť byla vyvinuta firmou Mitsubishi Electric v roce 1996. Od té doby vznikly další modifikace této sítě se speciálně upravenými vlastnostmi. Jsou to například sítě CC-Link IE, CC-Link IE Field nebo CC-Link IE TSN. [11] [14] [15]

2.5.4 CANbus

V tomto případě jde o sériový komunikační protokol, používající se zejména v automobilech nebo lékařské technice. Mezi jeho hlavní výhody patří jednoznačně spolehlivost a jednoduchá aplikace. [16] [17]

2.5.5 DeviceNet

DeviceNet je součástí většiny logických automatů jako přídavný komunikační modul a slouží především jako komunikace mezi jednotlivými PLC a ostatními zařízeními. Charakteristické pro tuto sběrnici je, že má přepínatelnou přenosovou rychlost. Výhoda tohoto typu komunikace spočívá v napájení jednotek přímo o sběrnici. Velikost sítě může být v rozmezí od 100 m až do 500 m. [18]

2.5.6 EtherCAT

Tato technologie je založena na komunikaci typu Master-Slave, mezi jejíž hlavní výhody nesporně patří přesná synchronizace a krátké časy cyklů (desítky µs). EtherCAT je komunikace mezi řídícími systémy založená na TCP/IP protokolu, tzn. na standardní Ethernetové síti. Používá se zejména pro komunikaci mezi řídící jednotkou a ostatních zařízení (I/O moduly, měniče, …). [19]

2.5.7 MODBUS

Používá se zejména pro komunikace mezi různými zařízeními. Je to otevřený protokol, který je schopen přenášet data po různých sítích. V této době je tento protokol schopen komunikovat na těchto sítích – Ethernet, RS – 485 a vysokorychlostní síť MODBUS+. [11] [20]

2.5.8 CANopen

Jedná se o typ komunikačního protokolu, který je vystaven na základě Controller Area Network (CAN) a je využíván v mnoha odvětvích automatizace. CANopen má tu výhodu, že poskytuje možnost vyhnout se problémům, které jsou typické pro CAN (např. synchronizace). [21] [22]

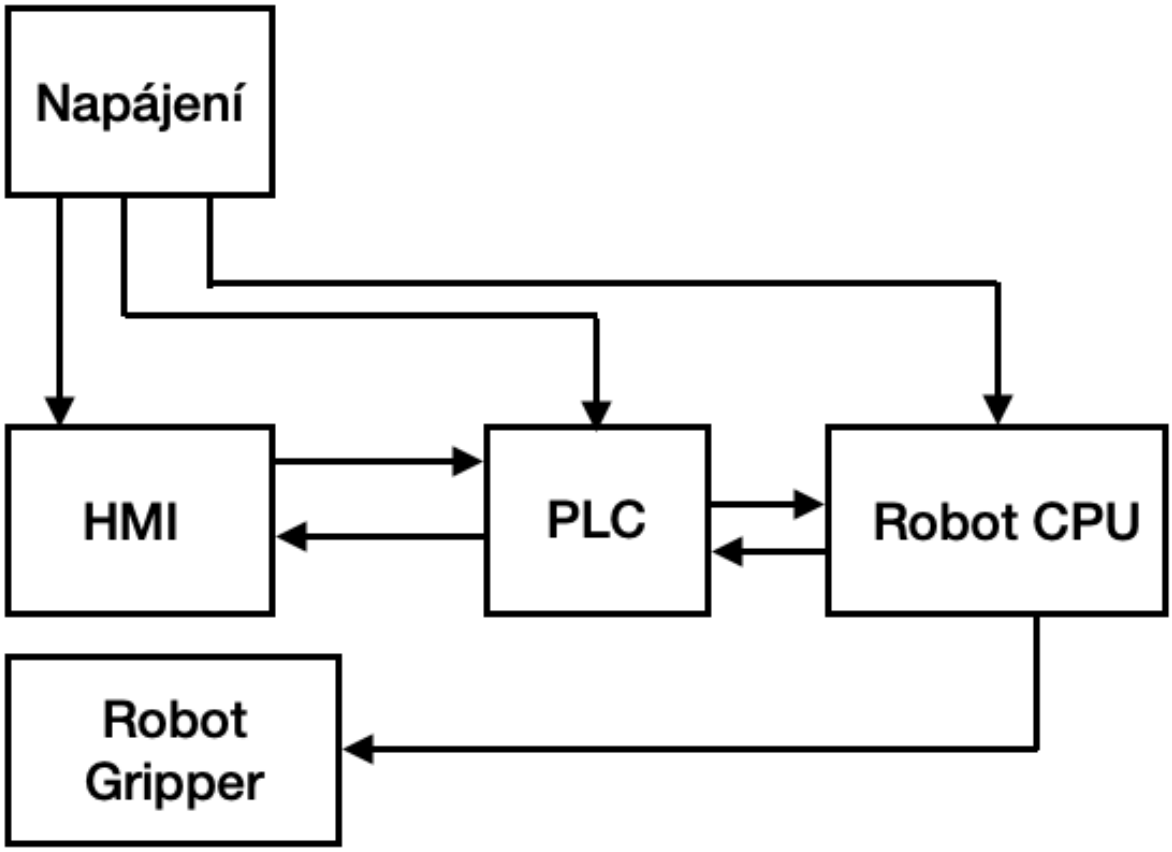
2.5.9 AS-interface

Fyzická vrstva této sběrnice je tvořena nestíněným dvouvodičovým kabelem, který slouží jak pro napájení, tak i pro přenos dat. Délka sítě dosahuje 100 m, s opakovači až 300 m. Přenosová rychlost je 167 kB/s. Doba cyklu této sběrnice je maximálně 10 ms. [7] [23]

3 Koncept robotického pracoviště

3.1 Návrh pracoviště

Na školícím pracovišti bude k dispozici PLC, HMI a robot. Celé pracoviště spolu musí komunikovat přes daný komunikační protokol. To znamená, že všechny moduly budou propojeny přes síťový switch. Obecné blokové schéma pracoviště znázorňuje obr. 8. Takto zapojené pracoviště bude řízeno pomocí PLC, ovládat ho bude možné pomocí HMI panelu a řídící jednotka robota umožní komunikaci mezi PLC a robotem, protože samo PLC nedokáže robota uvést do pohybu.



Obr. 8 - Návrh pracoviště

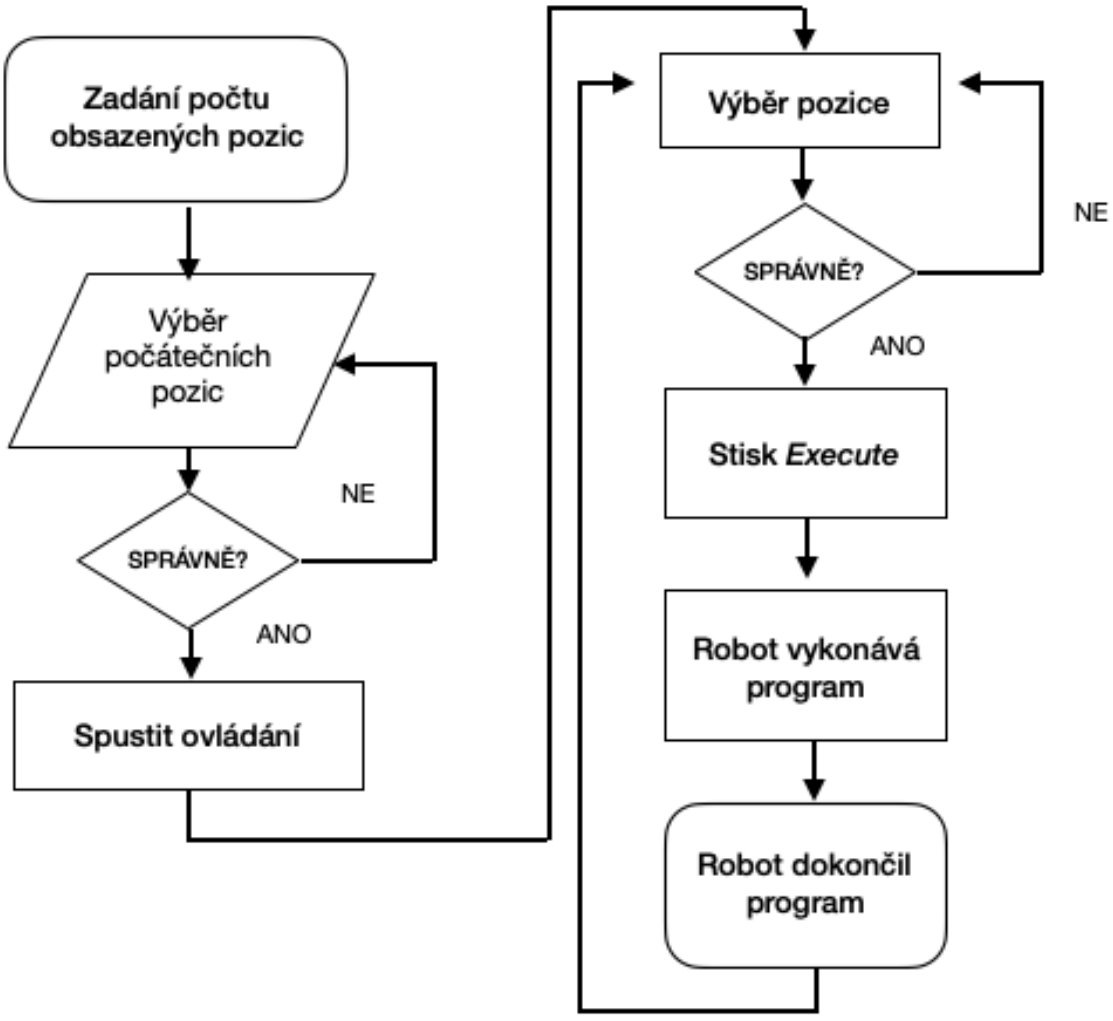
3.2 Popis vybrané školící aplikace

Jako ukázkovou úlohu jsem vybral přesouvání míčků na stojanu. Řídící algoritmus bude vycházet z PLC ovládaného pomocí HMI a řízen bude v této úloze robot tak, aby přesouval míčky na stojanu.

Uprostřed celé úlohy bude robot. Vedle robota bude stojan s míčky.

Na začátku celé úlohy se pracoviště zapne včetně potvrzení bezpečnosti. Dále by aplikace mohla obsahovat několik pozic, na které by se robot přesouval a vykonal na nich příslušnou činnost (např. uchopení předmětu).

Ovládání by mělo fungovat tak, že se vybere první pozice, ta se potvrdí tlačítkem, robot vykoná program a po vykonání programu se může opět zvolit další pozice. V ovládání bude vyřešena také možnost výběru nesprávné pozice zablokováním tlačítka pro vykonání programu. Logické zpracování úlohy je znázorněno vývojovým diagramem na obr. 9.



Obr. 9 - Vývojový diagram školící aplikace

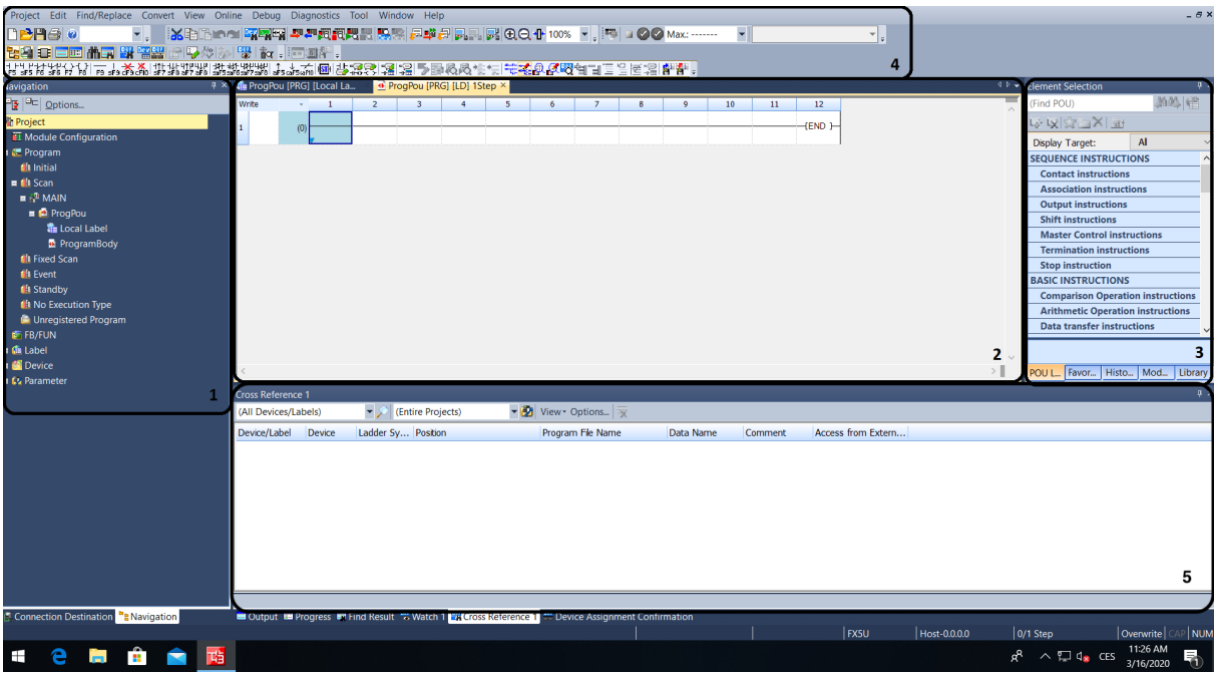
Tuto úlohy jsem zvolil z důvodu její komplexnosti pro celé pracoviště, nutnosti komunikace mezi robotem a PLC navíc je spousta možností jak tuto úlohu implementovat. Také je dobré, že mohu na této úloze vyzkoušet spoustu věcí z programování PLC, robota nebo HMI.

4 Praktická část

4.1 Vývojové prostředí pro programování PLC GX Works3

PLC od firmy Mitsubishi se programují v prostředí s názvem *GX Works3.* Jedná se o vývojové prostředí od stejné firmy, což značně zlepšuje komunikace mezi prostředím a řídící jednotkou logického automatu.

Při vytvoření nového projektu musíme zadat typ CPU a typ programovacího jazyka. Po potvrzení se objeví základní okno, které již umožňuje tvorbu softwaru a obsahuje několik částí (viz obr. 10). [24]



Obr. 10 - Okno po vytvoření nového projektu v GX Works3

V levé části se nachází část označená číslem 1 a nazývá se „*Navigation*“. Tato část obsahuje informace o projektu. Jsou to informace o hardwaru, všech POU, které jsou v rámci tohoto projektu používány nebo třeba informace o proměnných, jejich adresaci a popisu. V části *Program* jsou všechny programy, které se mají vykonávat. Vždy musí obsahovat *Main,* který se vykonává cyklicky. Dále tam v případě složitějších programů mohou být podprogramy, které se budou v hlavním programu volat. Také je na obrázku vidět, že každá POU obsahuje *Program Body* a *Local Label*. Program Body je část, která obsahuje samotný program a v části Local Label je možné definovat lokální proměnné, které budou moci být použité pouze v příslušném programu. Další důležitou částí této sekce je část nazvaná *FB/FUN*. Tato část obsahuje uživatelem vytvořené funkční bloky (FB) nebo funkce (FUN). *Label* je další důležitou

částí levého okna. V této části mohou být vytvořeny seznamy globálních nebo lokálních proměnných, ke kterým se nechají přiřadit adresy, typy proměnných, komentáře, ale hlavně také symbolické názvy proměnných, které značně zjednodušují orientaci v programu. [24]

Uprostřed je druhá část, která je vyznačená „2“ a v této části probíhá tvorba vlastních programů.

Další část („3“) se jmenuje *„Element Selection“* a obsahuje všechny funkce, které jsou definovány v používaných knihovnách.

V části „4“, v horní části prostředí GX Works3 jsou tlačítka pro vkládání často používaných prvků do programu, úprava okna „2“, spouštění, zastavení a ovládání simulace, komunikace s CPU aj.

Poslední část „5“ obsahuje různá okna, jak je vidět na obr. 10, konkrétně v jeho spodní liště. Mezi často používaná okna vyskytující se v této části patří *Output, Watch* nebo *Cross Reference*. *Output* obsahuje informace o chybách popřípadě o varováních v programu po jeho kompilaci. Pro sledování proměnných během simulace programu je možné využít okno *Watch*, do kterého je nutné nejdříve přidat dané proměnné, které je pak v tomto okně možné sledovat online. Dalším důležitým oknem při tvorbě programu je určitě *Cross Reference*, které umožňuje hledat proměnné podle názvu nebo adresy v celém projektu a jako výsledek hledání se objeví jak a kde jsou v programu použity. Usnadňuje to orientaci a hledání chyb v programu.

Po vytvoření programu je důležité mít možnost otestovat správnost programu. K tomu se nechá využít GX Simulator3, který je schopný simulovat reálné PLC podle hardwarové konfigurace v projektu. Během simulace je možné přepnout pracovní okna s programy do režimu monitorování a tím sledovat a ovládat proměnné podle potřeby programu. Simulace je velmi užitečná v případě tvorby programu „*offline*“, kdy není dostupné PLC a je potřeba kontrola funkčnosti daného programu. Značně to ulehčuje a zrychluje nahrání programu a ladění na reálném logickém automatu. Takto nasimulovaným PLC je možné ovládat a řídit také simulace HMI nebo robota. [24]

4.2 Vývojové prostředí pro vizualizaci GT Designer 3

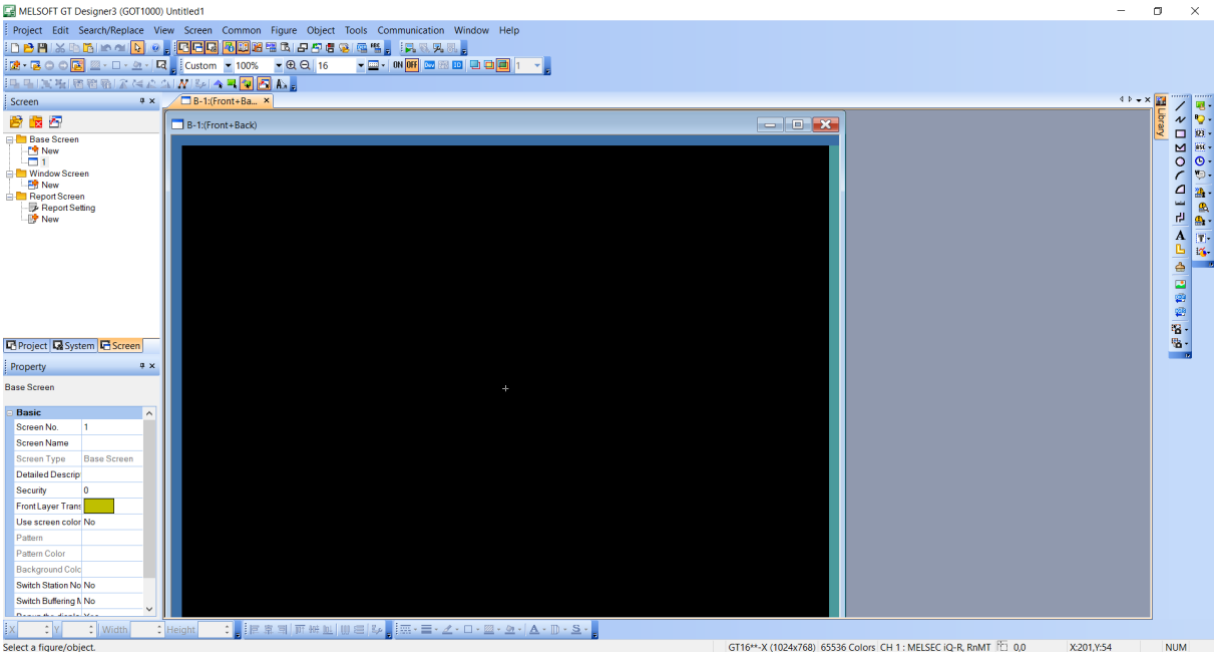
Vizualizace pro HMI panely značky Mitsubishi se tvoří ve vývojovém prostředí GT Designer3, které je od stejné firmy.

Při tvorbě nového projektu musíme nejdříve zvolit typ GOT. Zkratka GOT je obdoba více rozšířeného HMI. Pro nastavení projektu je potřeba otevřít okno *GOT Type Setting,* kde

se musí nejdříve vybrat typ GOT. Dále v okně *GOT Enviromental Setting* je možné nastavit jak se budou měnit jednotlivé obrazovky nebo se v tomto okně nechají nastavit různé typy obrazovek apod. [25]

Po vytvoření projektu a zadání typu panelu se objeví ve vývojovém prostředí prázdná obrazovka jako je na obr. 11.

Hlavní část s obrazovkou slouží k tvorbě vizualizací pro HMI panely. Tato část bude vysvětlena v části o tvorbě vizualizace.



Obr. 11 - Ukázka vývojového prostředí GT Designer3 od firmy Mitsubishi Electric

V levé části obrazovky jsou podobně jako v GX Works3 informace o projektu, systémovém nastavení, ale také seznam obrazovek, který je rozdělen do 3 částí. V části *Base Screen* jsou hlavní obrazovky, na kterých může probíhat ovládání úlohy. *Window Screen* jsou tzv. pop-up obrazovky. To znamená, že tyto obrazovky se mohou objevit na jakékoliv hlavní obrazovce když dojde k aktivaci např. nějakého bitu, který je s touto obrazovkou provázán nebo tlačítka, které danou obrazovku otevře. Tyto vyskakovací obrazovky se nechají použít jako informativní, potvrzující nebo chybové hlášení. Poslední sekcí je *Report Screen*. Na těchto obrazovkách se vyskytují hlášení o stavu PLC. Ve spodní levé části se nachází informace o objektu, který je právě zvolen. V případě obrázku 9 to jsou informace o hlavní obrazovce *1*. [25]

Pravá část prostředí obsahuje všechny možné možnosti úpravy obrazovek. Je zde například možnost kreslit obrazce, vkládat tlačítka, signalizační světla atd.

V části, která je nahoře jsou různé funkce pro nastavení HMI (*GOT Type (Enviromental) Setting, …*), tlačítka pro ovládání simulace nebo pro úpravu hlavní části obrazovky (vývojové prostředí umožňuje skrýt (zobrazit) určité vrstvy, prvky, apod.). [25]

Podobně jako GX Works3 je i v tomto prostředí možnost simulace na virtuálním HMI panelu. Virtuální panel umožňuje plně ovládat pracoviště řízené PLC, stejně jako panel reálný. Potíž je v tom, že při simulaci často bývá komunikace s PLC (např. přes CC-Link) značně pomalá.

4.3 Prostředí pro tvorbu programu pro robota

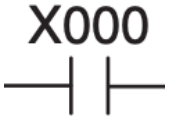
Roboti od firmy Mitsubishi se v praxi programují spíše na reálném robotovi a vývojová prostředí slouží spíše pro parametrizaci. Je to především kvůli nastavení správných pozic robota. Aby byl program vytvořený v *offline* módu, tak je nutný 3D výkres pracoviště, které robot bude obsluhovat.

Vývojové prostředí pro roboty Mitsubishi je *RT ToolBox3*. Prostředí umožňuje tvorbu a ladění programu lze také nastavit simulaci vstupů a výstupů při simulaci. Nechají se také nastavovat různé parametry robota. Obsahuje 3 různé módy – *offline, online, simulation*. Pro tvorbu programu a nastavení parametrů je možné být v režimu *offline*, při komunikaci s robotem a monitoringu musí být prostředí ve stavu *online* a konečně pro simulaci je poslední mód – *simulation*. [26]

4.4 Programování PLC

4.4.1 Struktura programového příkazu

Jednotlivé programové příkazy se skládají z instrukce a z příslušných operandů, které odpovídají proměnným.



Obr. 12 - Programový příkaz

Na obr. 12 je vidět ukázka programového příkazu, který se skládá, v tomto případě, ze spínacího kontaktu (instrukce) a operandu X000. Operand se dále skládá z názvu proměnné, který definuje o jakou proměnnou se jedná a adresy proměnné. Dále proměnná může být doplněna symbolickým názvem.

4.4.2 Typy proměnných

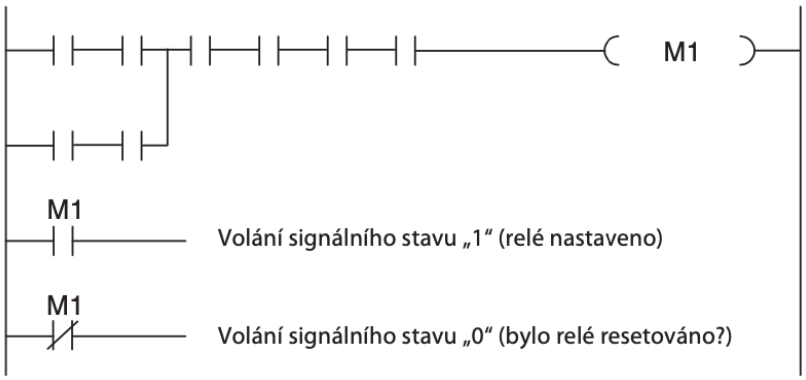
V každém vývojovém prostředí se používají různé značení pro typy proměnných, které jsou ale ve všech prostředích stejné.

4.4.2.1 Vstupy a výstupy

Vstupní proměnné jsou připojeny ke vstupním svorkám PLC a mohou to být například spínače. V případě volání vstupu se změří napětí na svorce a podle toho nabývá jedné ze dvou hodnot (logická 0 nebo 1). Výstupy fungují opačně. To znamená, že nejdříve se musí aktivovat výstupní proměnná v PLC a ta aktivuje fyzický výstup. Výstupy také nabývají pouze stavů ZAPNUTO (odpovídá logické 1) nebo VYPNUTO (logická 0). Vstupy se označují identifikátorem X a výstupy Y (např. X000 je vstup s adresou 000). [4]

4.4.2.2 Relé

Tento typ bitové proměnné se používá v případě, že je potřeba dočasně uložit výsledek logické operace. Pro relé se používá identifikátor M. Kromě klasických relé existují také tzv. diagnostická relé, která dávají informaci o stavu programu, PLC. Pro využití diagnostických relé je vyhrazen rozsah proměnných od M8000. [4]



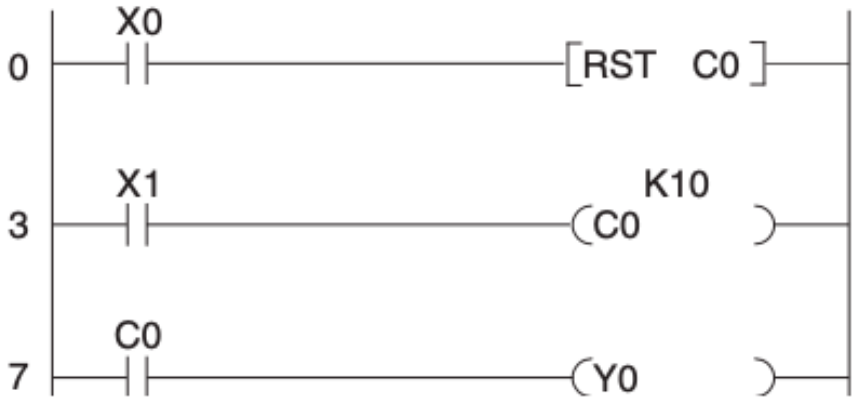
Obr. 13 - Možnost použití reléové proměnné [4]

4.4.2.3 Časovač

Časovače se používají v případě, že je potřeba zapnout/vypnout s časovým zpožděním. PLC poskytují možnost, jak se vyhnout napevno zapojeným časovým relé. Místo toho se v jejich aplikacích využívá interních časovačů, které jsou programovatelné. Fungují na principu čítačů, které počítají časové intervaly (např. 0,01 s) a po dosáhnutí naprogramované hranice se aktivují. V programu se označují písmenem T.[4]

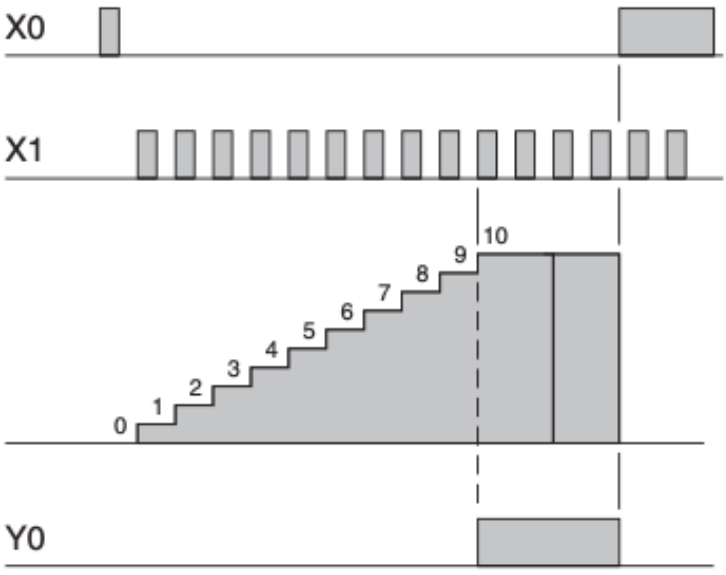
4.4.2.4 Čítač

Čítače se označují písmenem C a počítají signální pulzy. Jsou aktivovány/vypínány podle porovnávané hodnoty.



Obr. 14 - Možnost použití čítače [4]

Na obr. 14 je vidět příklad použití čítače. Aktivací proměnné X0 se resetuje čítač C0. Když se zapne vstup X1, tak je hodnota čítače zvýšena o 1 a když dosáhne hodnoty, která je uložena v proměnné K10 tak se aktivuje výstup Y0. Ten je sepnutý až do resetování čítače. Princip tohoto programu je zobrazen na obr. 15. [4]



Obr. 15 - Grafické znázornění funkce čítače

4.4.2.5 Datový registr

Datové registry (proměnné typu WORD) slouží jako paměť PLC, na rozdíl od relé, které ukládají pouze bitové proměnné, mohou tyto registry, označované písmenem D, ukládat 16

nebo 32-bitové proměnné (např. výsledky měření). Hodnoty uložené v klasických datových registrech se po vypnutí PLC smažou. Proto také existují registry zamčené, jejichž obsah zůstane zachován. [4]

4.4.3 Základní prvky používané při programování PLC

Následující prvky mohou být vyjádřeny různě v různých programovacích jazycích, ale jejich význam je totožný. Pro jednoduchost popíšu značení pouze v jazyku kontaktních schémat, protože jsem ho využíval ve svém programu.

První skupinou prvků jsou prvky vstupní, které jsou měněny buď ovládáním nebo mohou být programově spínány. Mezi vstupní prvky v jazyku kontaktních schémat patří např. *spínací kontakt (Normally open (NO) contact)*. Bit přiřazený tomuto kontaktu má v klidovém stavu (ve stavu bez napětí) hodnotu logické 0, když ho aktivujeme bit se nastaví do 1. Spínací i rozpínací kontakt může být reprezentován buď klasickým spínačem, tlačítkem nebo může tento bit znamenat výstup předcházející operace (např. výstup nějakého čidla nebo senzoru). Opakem spínacího kontaktu je kontakt *rozpínací (Normally closed (NC) contact).* Tyto prvky mohou být přes vstupní modul PLC propojeny s tlačítkem. [24]

Dále se v LD využívá pro vytváření podmínek různých funkcí, které jsou většinou předdefinované jako knihovní funkce, proto k nim stačí přiřadit pouze vstupní a výstupní hodnoty. Mezi nejčastěji používané funkce patří klasické matematické funkce nebo také třeba porovnávací bloky, které porovnávají vstupní hodnoty a když dojde ke splnění podmínky, která je tímto blokem definována, tak dojde k sepnutí výstupu porovnávacího bloku. [24]

Dalšími důležitými prvky jazyka kontaktních schémat jsou prvky výstupní. Tyto prvky mění hodnotu proměnných na základě programu. Nelze je ovládat napřímo jako vstupní prvky. Základní je *výstupní cívka s přímou funkcí (coil)*. Tato cívka ve stavu bez napětí má bit přiřazený dané cívce hodnotu logické 0. Po aktivaci tento bit bude mít hodnotu logické 1. Po poklesu napětí na nulu bude bit opět v 0. K cívce s přímou funkcí existuje také opačný prvek, který má opačné vlastnosti. Tomuto prvku se říká *výstupní cívka s negovanou funkcí (negated coil).* Poté existují další 2 výstupní funkce – *set, reset*. V případě funkce *set* dojde při aktivaci k zápisu 1 do daného bitu. Logická 1, ale zůstane zapsána v proměnné i při deaktivaci této funkce. V tom se liší od výstupních cívek, které nedokáží držet hodnotu při poklesu napětí. Je-li potřeba dostat do proměnné opět 0, je třeba použít funkce opačné – *reset*. Na tyto výstupní funkce mohou být navázané pouze bitové proměnné. V případě potřeby zápisu proměnné typu

WORD nebo jiných více bitových proměnných se používá výstupní funkce *move.* Tato funkce přesune konkrétní číslo nebo obsah registru do cílového registru. [24]

Jednotlivé programové příkazy jsou vlastně podmínky, při jejichž splnění dojde k sepnutí výstupu. Tyto podmínky se tvoří kombinacemi (definovány logickými funkcemi (and, or, xor, apod.)) vstupních parametrů a funkcí.

4.5 Tvorba vizualizace pro HMI panel

V této části bude vysvětlen obecný postup při tvorbě softwaru pro HMI panel, především prvky, které jsem aktivně využíval při tvorbě vizualizace. Zahrnuje to základní ovládání školícího pracoviště jako je např. zapnutí/vypnutí nebo resetování pracoviště.

4.5.1 Úprava obrazovek

Základní prvky využívané pro vytváření jednotlivých obrazovek jsou např. *Switch, Lamp, Numerical Display/Input* nebo různé tvary, obrázky, textová pole, atd., která se nechají nakreslit na obrazovku.

4.5.1.1 Switch (tlačítko)

Switch (česky přepínač) je obyčejné tlačítko, které může mít mnoho funkcí. Jednou ze základních funkcí je bezpochyby práce s bitovými proměnnými. V tomto vývojovém prostředí může *switch* resetovat, invertovat nebo setovat (a to i po dobu stisknutí nebo i po jeho puštění) jednotlivé bity. Jedno tlačítko může být provázáno s několika bity. Pomocí tlačítka také lze přepínat jednotlivé obrazovky, to bude popsáno v podkapitole přepínání obrazovek.

Mezi zajímavé vlastnosti tlačítka v prostředí GT Designer3 patří tzv. *Trigger.* Tato funkce umožňuje „zakázat“ stisknutí tlačítka v závislosti na jiném bitu. Tuto funkci jsem vhodně využil na obrazovce číslo 3, která je určená pro výběr začátečních pozic. Konkrétně je tato funkce na tlačítku „*Confirm Positions*“, které je červené a nelze jej stisknout pokud se počet pozic nerovná počtu skutečně vybraných pozic. Jakmile se počty rovnají, tak tlačítko zezelená a je možné potvrdit výběr a začít s ovládáním úlohy. [25]

Také lze měnit barvu tlačítka a text na tlačítku. Tato vlastnost se nastavuje také v nastavení tlačítka a lze to udělat dvěma způsoby. Buď se barva mění podle toho je-li tlačítko aktivované nebo nikoliv nebo v závislosti na aktivaci jiného bitu. Této vlastnosti jsem mj. využil

také u změny z červené na zelenou u tlačítka „*Confirm Positions*“. Text lze měnit pouze podle toho jestli je tlačítko stisknuté nebo ne.

4.5.1.2 Lamp

Jedná se o výstupní funkci panelu (zjednodušeně řečeno se jedná o indikátor), kterou uživatel nemůže přímo ovládat (pouze pomocí tlačítek). Jeho funkce je jednoduchá. Indikátor provážeme pomocí adresy s nějakou proměnnou a poté bude měnit barvu podle stavu dané proměnné. Využití této funkce je dobře vidět na obrazovce číslo 4, kterou se ovládá celá úloha a v dolní části je jedna dioda, která bliká když robot vykonává program. V pravé části zase diody ukazují aktuálně obsazené pozice. [25]

4.5.1.3 Numerical Display/Input

V tomto případě se jedná o vstupně výstupní okno. To znamená, že se nechají do tohoto okna zapisovat hodnoty obsluhou, ale také mohou být měněny programově a ukazovat aktuální hodnotu registru přiřazeného tomuto oknu. Jde opět o důležitou funkci ovládacích panelů. Využívá se mj. také k tomu, když je nutné kontrolovat počet výrobků při hromadné výrobě, ale využití této funkce je samozřejmě daleko rozsáhlejší. [25]

Stejně funguje také jiná funkce – *Text Display/Input*, avšak s tím rozdílem, že pracuje s textovými řetězci místo číselných hodnot.

4.5.2 Přepínání obrazovek

4.5.2.1 Pevné přepínání

Přecházení mezi obrazovkami je stejně důležité pro ovládání jako vytváření samotných obrazovek. Pevné přepínání znamená, že se tlačítku na obrazovce přiřadí funkce *Screen Switching* a v parametrech této funkce se nastaví číslo libovolné hlavní nebo dialogové obrazovky a po aktivaci tohoto tlačítka se na panelu zobrazí nastavená obrazovka.

Výhodou tohoto přecházení je, že je to velice jednoduché, rychlé a není potřeba žádného programového kroku v PLC. Avšak nevýhodou je, že na jednom tlačítku nemůže být možnost přepnutí na různé obrazovky podle aktuálních podmínek. Tuto možnost řeší následující možnost přepínání – pomocí registrů.

4.5.2.2 Přepínání pomocí registrů

Toto přepínání je o něco složitější a vyžaduje úpravu programu pro PLC. Nejdříve je třeba nastavit tzv. přepínací registry pro jednotlivé typy obrazovek. Základní registr pro hlavní obrazovky, který je přednastaven v GT Designer3 je GD100, pro další typy obrazovek to jsou registry GD101, GD102, atd. Poté je potřeba vymezit v PLC registr, do kterého se budou zapisovat čísla obrazovek podle aktuálních stavů proměnných v PLC. [25]

Potom už stačí tlačítku přiřadit funkci pro přepnutí obrazovky, ale tentokrát se vybere možnost **přepínání podle registrů** , kde zadáme registr z PLC, se kterým bude daný registr komunikovat.

Využití tohoto způsobu přepínání je např. při ovládání linky a když se objeví chybu tak je možné pomocí PLC vyhodnotit o jakou chybu se jedná a obsluze se na obrazovce objeví chybové hlášení, které říká proč se chyba objevila. Zrychluje to diagnostiku a obnovu provozuschopnosti strojů.

4.6 Programování robota

Program pro roboty od firmy Mitsubishi Electric se tvoří v prostředí RT ToolBox3 a píše se v jazyce Melfa Basic VI. Tento programovací jazyk je velmi podobný např. jazyku C.

Pro základní přehled zde uvedu některé z funkcí, které jsem aktivně používal v programu.

Základní pohybové operace jsou například *Mov, Mvs* nebo *Mvr*. Funkce *Mov* slouží pro pohyb robota do zadaného po nejkratší trase. Pro správné využití této funkce je třeba zadat cílový bod, dále se mohou přidat různé parametry pohybu jako jsou třeba rychlost, zrychlení apod. Jde o pohyb kloubovou interpolací a hodí se zejména pro pohyb po křivce. Funkce *Mvs* je velmi podobná předchozí funkci s tím rozdílem, že tentokrát jde o lineární interpolaci a využívá se především pro pohyb po přímce. *Mvr* je funkce kruhové interpolace a jde tedy o pohyb po kružnici. Dalšími pohybovými funkcemi jsou funkce pro vyhlazování dráhy robota, definování rychlostí, ovládání upínky, apod. [26]

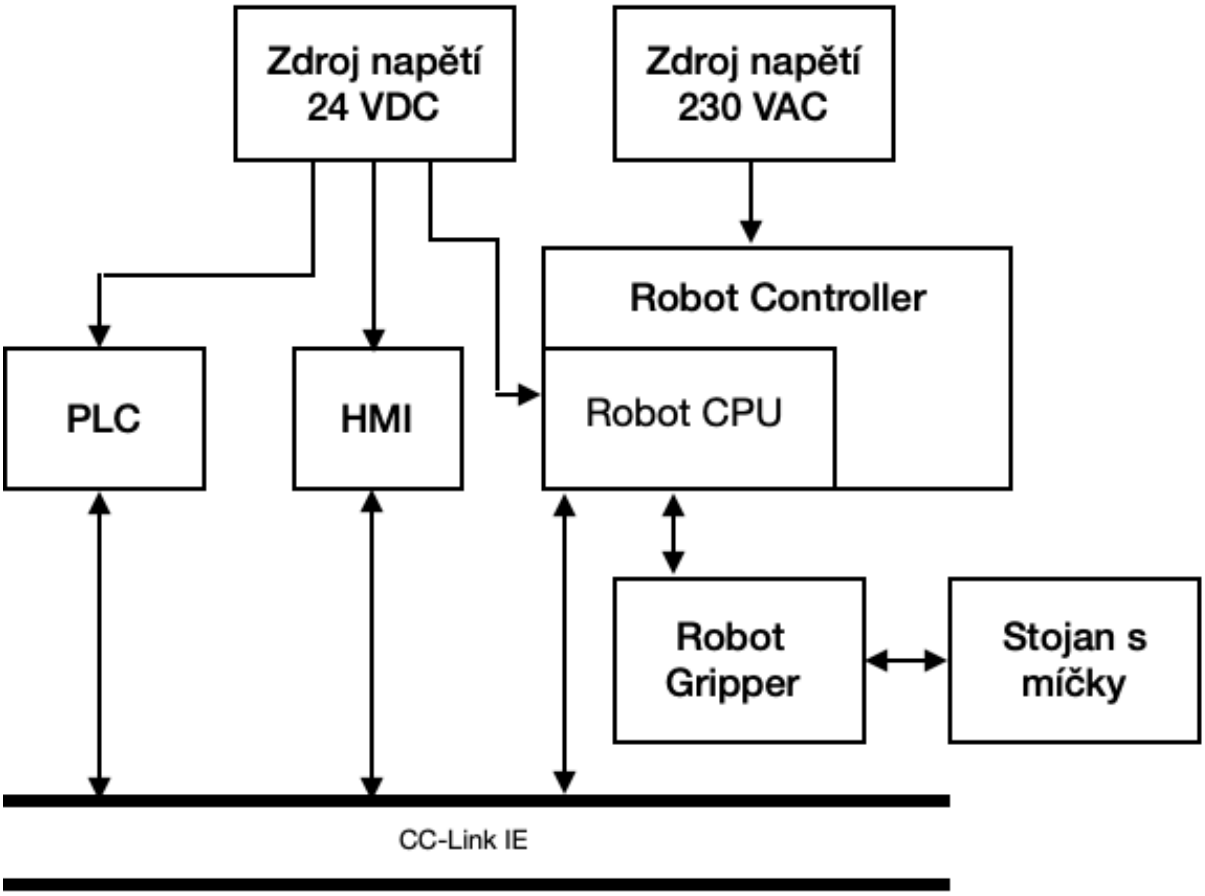
Další skupinou příkazů jsou příkazy pro kontrolu programu. Jde o příkazy, které jsou hojně používané ve vyšších programovacích jazycích a jde například o funkce *if, then, select (case), go to*. Do této skupiny se řadí také příkazy pro volání programu (*CallP*), nebo různé další příkazy, které kontrolují podmínky a na jejich základě umožňují skákat v programu. [26]

Užitečné příkazy jsou také *XLoad, XRun, XStp, XClr, GetM* nebo *RelM*. Pro tuto skupinu příkazů je důležité zmínit, že robot na školícím pracovišti má 8 slotů, do kterých budou nahrány programy a robot je může i současně vykonávat. Mohou se nastavit podmínky spuštění nebo jak má program probíhat (cyklicky nebo se vykoná pouze jednou a skončí). První příkaz – *Xload* nahrává do zadaného slotu robota libovolný program, příkazem *XClr* se program z daného slotu vymaže. *XRun* a *XStp* spouští a zastavují jednotlivé sloty robota. *GetM* a *RelM* povolují a zakazují práva, která umožňují pohyb. Tato práva má pouze první slot, pokud to pomocí těchto funkcí nepovolíme také v jiném programu. [26]

5 Postup při zpracování školící aplikace

5.1 Školící pracoviště

Všechny prvky na školícím pracovišti jsou od firmy Mitsubishi Electric. Přes síťový switch je na sběrnici připojeno hned několik - zdroj 24 VDC R61P, centrální řídící jednotku R04CPU, řídící jednotku robota R16RTCPU a HMI panel GT2705-VTBD na pracoviště je samozřejmě také robotický manipulátor MELFA RV-4FRL. Robot obsahuje elektrickou upínku, na které jsou namontovány úchytky na míčky vyrobené na 3D tiskárně. Součástí pracoviště je také stojan na míčky, který byl také zhotoven pomocí 3D tiskárny. Blokové schéma pracoviště je na obr. 16.



Obr. 16 - Blokové schéma pracoviště

Náhled celého školícího pracoviště je na obr. 17. Vlevo na obrázku je vidět stojan s míčky, který jsem používal při vybrané aplikaci.

Obr. 17 - Školící pracoviště ve firmě B:Tech

Na obr. 18 je vidět pendant, který slouží k ovládání robota v manuálním režimu a učení jednotlivých pozic robota, které jsou součástí vykonávaného pohybu.

Obr. 18 – Ovládací pendant robota

5.2 Programování úlohy

Jako první krok při tvoření programu pro PLC je nastavení hardwarové konfigurace. V praxi to znamená přidání jednotlivých modulů, které jsou reálně na pracovišti, do nově založeného projektu. Ve vývojovém prostředí GX Works3 se to dělá v okně *Module Configuration*. Vytvořená hardwarová konfigurace odpovídá reálnému stavu zařízení.

Dalším krokem bylo navrhnutí základního ovládání a vizualizace na HMI panel. V prostředí GT Designer3 jsem vytvořil ovládací tlačítka pro pracoviště – *On/Off, Reset, Safety*. Tlačítko *On/Off* slouží pro zapnutí nebo vypnutí celého pracoviště, *Reset* uvádí úlohy do původního stavu a tlačítko *Safety* je pro manuální potvrzení bezpečnosti robotického pracoviště. Následovalo vytvoření seznamů proměnných v PLC. Tyto proměnné jsou v tab. 1.

Label Name	Data Type	Assign	Comment
HMI_OnOff	BOOL	M0	Zapnutí/vypnutí pracoviště, ON=1, OFF=0
HMI_Reset	BOOL	M3	Reset, uvede pracoviště do původního stavu
HMI_Safety	BOOL	M4	Safety OK = 1, vizuální kontrola pracoviště
HMI_ConfirmSelection	BOOL	M6	Potvrzení vybraných pozic
HMI_StartPos0	BOOL	M10	
HMI_StartPos1	BOOL	M11	
HMI_StartPos2	BOOL	M12	
HMI_StartPos3	BOOL	M13	
HMI_StartPos4	BOOL	M14	
HMI_StartPos5	BOOL	M15	
HMI_StartPos6	BOOL	M16	
HMI_StartPos7	BOOL	M17	
HMI_ConfirmStartPos0	BOOL	M20	
HMI_ConfirmStartPos1	BOOL	M21	
HMI_ConfirmStartPos2	BOOL	M22	
HMI_ConfirmStartPos3	BOOL	M23	
HMI_ConfirmStartPos4	BOOL	M24	
HMI_ConfirmStartPos5	BOOL	M25	
HMI_ConfirmStartPos6	BOOL	M26	
HMI_ConfirmStartPos7	BOOL	M27	
HMI_CONTROL_Pos0	BOOL	M30	Ovládání úlohy - 0. pozice
HMI_CONTROL_Pos1	BOOL	M31	Ovládání úlohy - 1. pozice
HMI_CONTROL_Pos2	BOOL	M32	Ovládání úlohy - 2. pozice
HMI_CONTROL_Pos3	BOOL	M33	Ovládání úlohy - 3. pozice
HMI_CONTROL_Pos4	BOOL	M34	Ovládání úlohy - 4. pozice
HMI_CONTROL_Pos5	BOOL	M35	Ovládání úlohy - 5. pozice
HMI_CONTROL_Pos6	BOOL	M36	Ovládání úlohy - 6. pozice
HMI_CONTROL_Pos7	BOOL	M37	Ovládání úlohy - 7. pozice
HMI_Execute	BOOL	M100	Pokny pro spuštění programu robota (příkaz "Execute" na hmi)
HMI_NoOfStartPos	INT	D3	Počet počátečních pozic
HMI_WarningPositions	BOOL	M1	Pomocný bit pro kontrolu správného počtu pozic
NumberOfPositions	INT	D0	Počet pozic (musí být roven počtu počátečních pozic)
FirstPrgSel_Finished	BOOL	M101	Výběr plné pozice dokončen
SecondPrgSel_Finished	BOOL	M102	Výběr prázdné pozice dokončen
NoOfPrg	INT	D2	Číslo programu, který má být vykonán
HMI_WarningControl	BOOL	M2	Kontrola výběru správné pozice

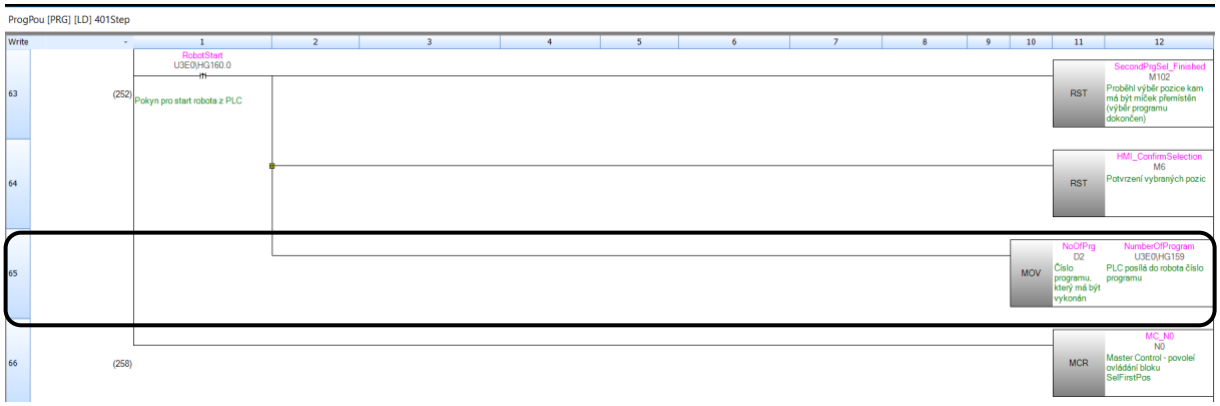
Tab. 1 - Seznam proměnných pro základní ovládání a komunikaci s HMI

Dále už jsem postupoval systematicky podle vývojového diagramu na obrázku 9. To znamená, že nejdříve jsem musel vymyslet výběr startovních pozic. Tento výběr jsem vyřešil

tak, že se na HMI objeví obrazovka s výběrem až po zapnutí celého pracoviště tlačítkem *On/Off* a výběr nebude fungovat pokud nepotvrdím bezpečnost pracoviště tlačítkem *Safety*. Na výběr je 8 pozic, které jsou simulovány tlačítky, potom je na obrazovce I/O display, do kterého se bude zadávat počet míčků, které budou na začátku k dispozici. Na obrazovce je také malé tlačítko s *i*, které dává informaci, že tlačítko pro potvrzení výběru nejde zmáčknout, je to pro to, že nesouhlasí počet vybraných pozic se číslem pro počet startovních pozic.

Programově je to řešené pomocí pomocného registru, do kterého se s náběžnou hranou tlačítek pro pozice přičítá 1 a s klesající hranou se odečítá 1. Tímto způsobem se programově kontroluje počet aktuálně vybraných pozic, který se pomocí porovnávacího bloku kontroluje se zadaným počtem pozic. Pokud jsou tato dvě čísla rozdílná, pak sepne bit *Warning Positions*, který znemožní potvrzení pozic.

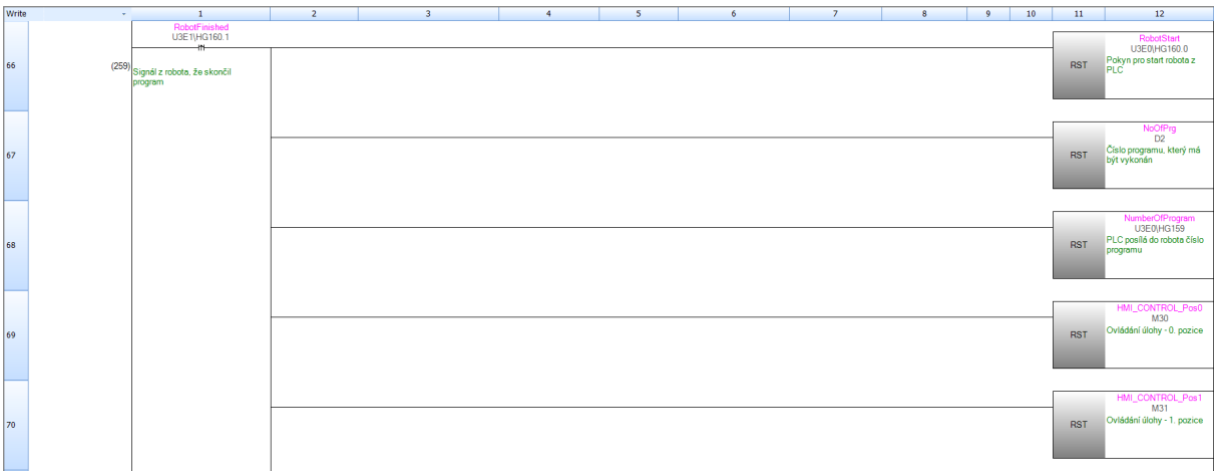
Po potvrzení výběru se hodnoty propíší do proměnných potvrzujících aktuální obsazené pozice, startovní pozice se resetují a na HMI panelu se objeví obrazovka s ovládáním pracoviště. Ovládání funguje tak, že se vybere pozice a potvrdí se stiskem tlačítka *Execute*, které je pomocí bitu *RobotStart* propojeno s robotem a spouští vykonání programu. Do robota se souběžně se startujícím bitem také propíše číslo pozice, kam má robot jet. Čísla pozic zohledňují podmíněné skoky v programu robota. Pokud má být míček vyjmut z pozice, tak je číslování 0-7. V případě vložení je nutné použít číslování 10-17. Jak už bylo zmíněno, vyplývá to z volání rozdílných funkcí v robotovi. Provedení jednotlivých akcí po aktivaci bitu *RobotStart* je znázorněno na obr. 19. Například na řádku 65 je vidět, předávání čísla programu do robota.



Obr. 19 - Pokyn z PLC pro start robota

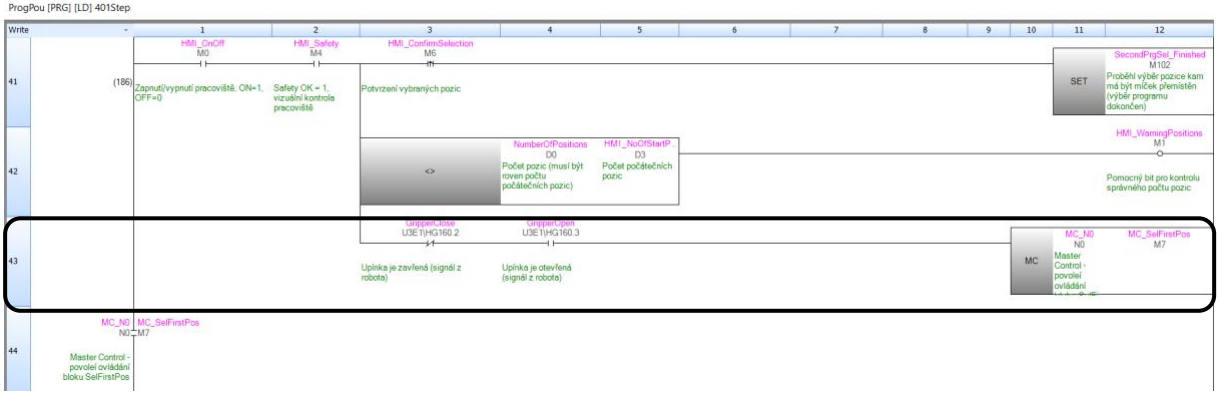
Po dobu chodu robota bliká dioda *Executing*, protože je provázána se stejnou proměnnou. Když program skončí, tak od robota přijde signál *RobotFinished*, který dává PLC

informaci o tom, že robot úspěšně dokončil program a je připraven na zadání dalšího programu. Přehled akcí, které vykoná PLC když robot dokončí je naznačen na obr. 20.



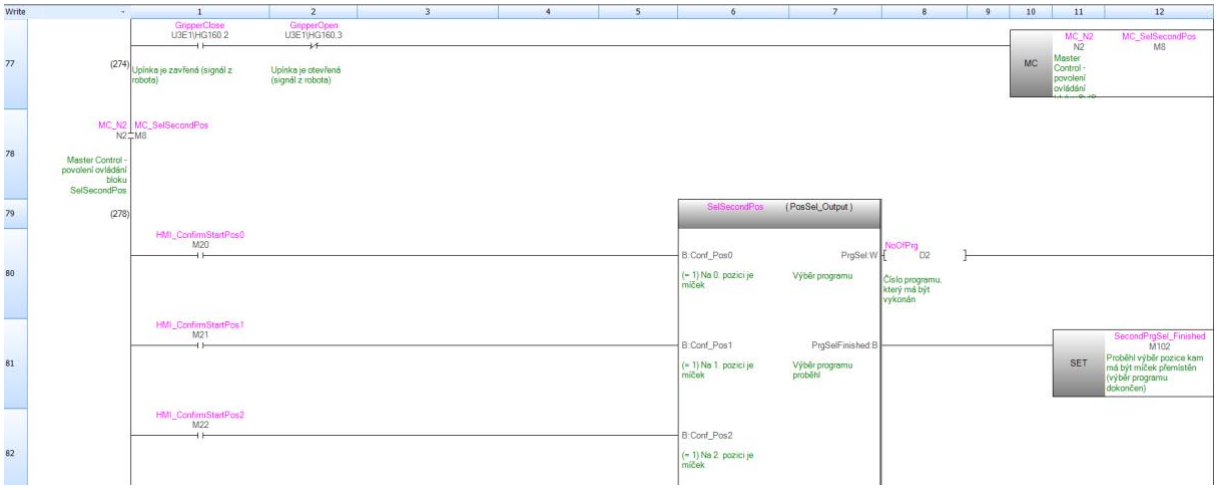
Obr. 20 - Program robota skončil

Na robotu je elektrická upínka, která je ovládána servomotory funkcemi *GripperOpen, GripperClose*, které jsou definovány v programu *Fuctions* v RT ToolBox3. Jsou to funkce, které mají booleovskou návratovou hodnotu, takže je možné získat informaci o aktuálním stavu upínky. Tato Informace je posílána do PLC a slouží k výběru funkce pro vložení míčku nebo pro vyjmutí míčku z dané pozice.



Obr. 21 - Podmínky výběru plné pozice

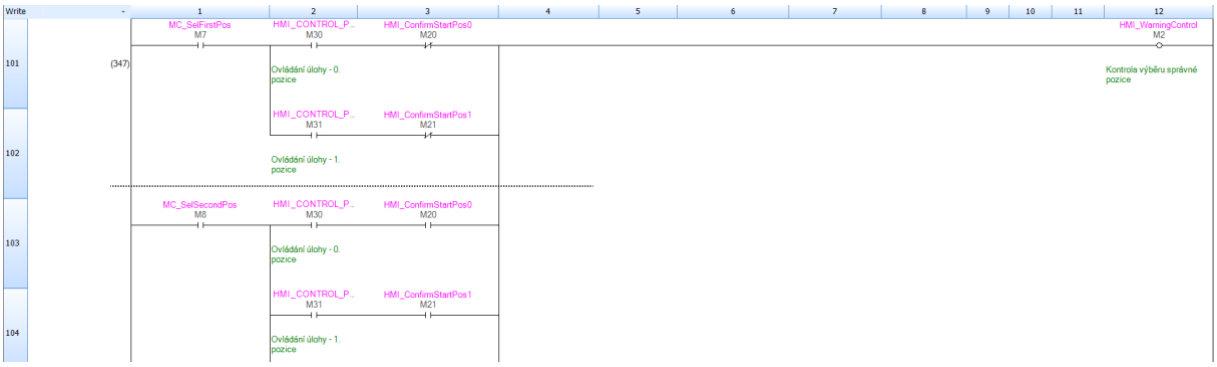
Pokud je upínka otevřená, tak je volán funkční blok s výběrem plné pozice, což je přehledně vidět na obr. 21, na řádku 43. Pokud je upínka zavřená, tak budou splněny podmínky pro volání funkčního bloku výběru prázdné pozice, to je vidět na obr. 22.



Obr. 22 - Podmínky výběru prázdné pozice

Po dokončení programu přestane blikat signalizace, přepíší se hodnoty aktuálně obsazených pozic a je možné vybírat další pozici.

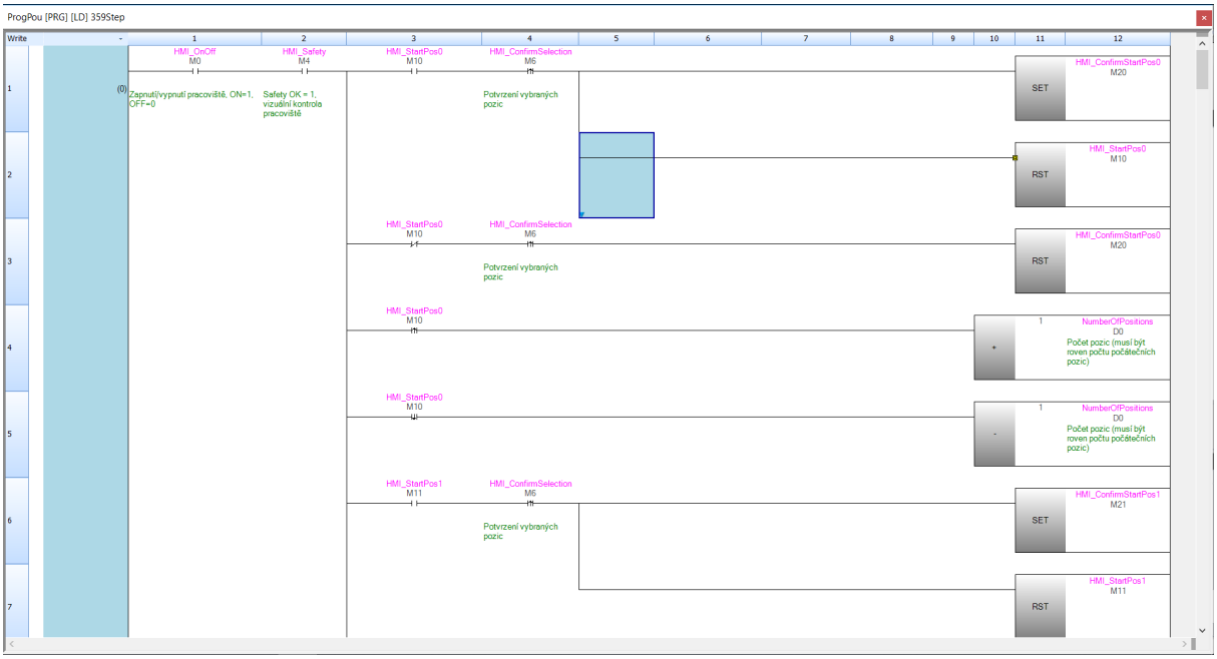
Kontrola výběru správné pozice je řešena podobně jako potvrzení výběru startovních pozic. Podle stavu robota se kontroluje, zda robot drží míček nebo ne. Takže snadno mohu kontrolovat výběr pozic. Když robot drží míček a já vyberu obsazenou pozici, tak tlačítko *Execute* zčervená a nemohu ho zmáčknout, protože v jeho nastavení jsem přidal na toto tlačítko *Trigger*, který kontroluje bit *HMI_WarningControl* a když je tento bit v 1, tak se znemožní zmáčknutí tohoto tlačítka. Po vybrání správné pozice opět obnoví svoji funkci. Jak je program řešen v PLC je vidět na obr. 23.



Obr. 23 - Kontrola výběru správné pozice

Další důležitou funkcí v programu je možnost změny výběru pozic. To je realizováno přechodem na předchozí obrazovku, kde se opět zadá počet pozic a vybere se příslušný počet pozic. Na tuto možnost je program připraven především díky programově řešenému výběru a správnému resetování pozic. Ukázka programu pro výběr počátečních pozic je na obr. 24. Je

vidět, že s náběžnou hranou tlačítka pro potvrzení pozic se aktuálně vybrané pozice potvrdí a pozice, které nebyly vybrány se pro jistotu resetují, protože mohly být aktivní v předchozím běhu programu. Na tomto obrázku je znázorněno i počítání vybraných pozic do registru *NumberOfPositions*.



Obr. 24 - Výběr startovních pozic

Poslední důležitou funkcí programu je funkce reset. Tato funkce se aktivuje s náběžnou hranou bitu HMI_Reset, který je provázán s tlačítkem na HMI panelu. Reset vymaže všechny vybrané pozice a vrátí se na začátek celé aplikace, to znamená, že je potřeba opět zapnout celé pracoviště.

5.3 Popis komunikace mezi robotem a PLC v rámci vybrané úlohy

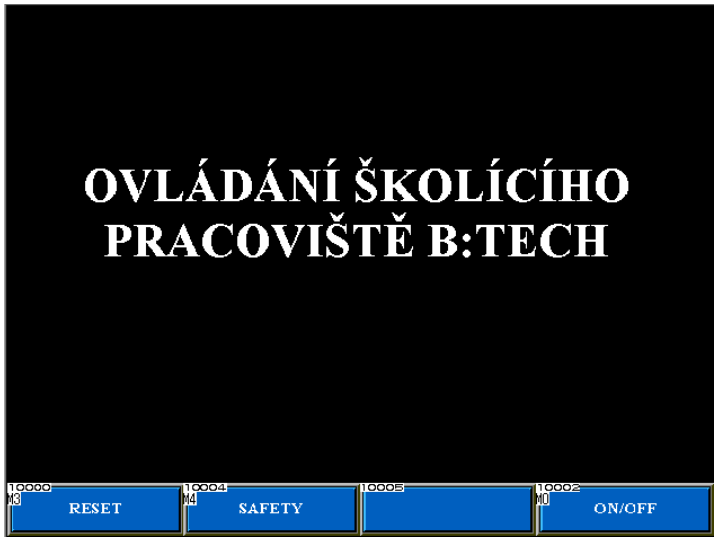
Komunikace mezi robotem a PLC je zprostředkována pomocí proměnných, které jsou definovány v PLC a jsou v tab. 2. Jsou to proměnné, které se automaticky propisují do robota. Na straně robota se komunikace řeší odlišným způsobem. Pomocí funkce M_Out a čísla bitu v programu pro robota se předávají informace do PLC, načtení hodnot z PLC se realizuje pomocí funkce M_In a příslušného bitu. Např. chci-li předat informaci do robota o startu programu, tak se v PLC bit U3E0\HG160.0 nastaví do hodnoty logické 1 a v programu pro robota bude instrukce M_In(12560). Je to velice dobrý způsob jak kontrolovat aktuální stav pracoviště na základě stavu robota.

Label Name	Data Type	Assign	Comment
RobotStart	BOOL	U3E0\HG160.0	Začátek programu
RobotFinished	BOOL	U3E1\HG160.1	Skončil program
GripperClose	BOOL	U3E1\HG160.2	Upínka je zavřená
GripperOpen	BOOL	U3E1\HG160.3	Upínka je otevřená
NumberOfProgram	INT	U3E0\HG159	Číslo programu

Tab. 2 - Seznam proměnných, které se propisují z PLC do robota a zpět

5.4 Uživatelský návod pro ovládání školící aplikace

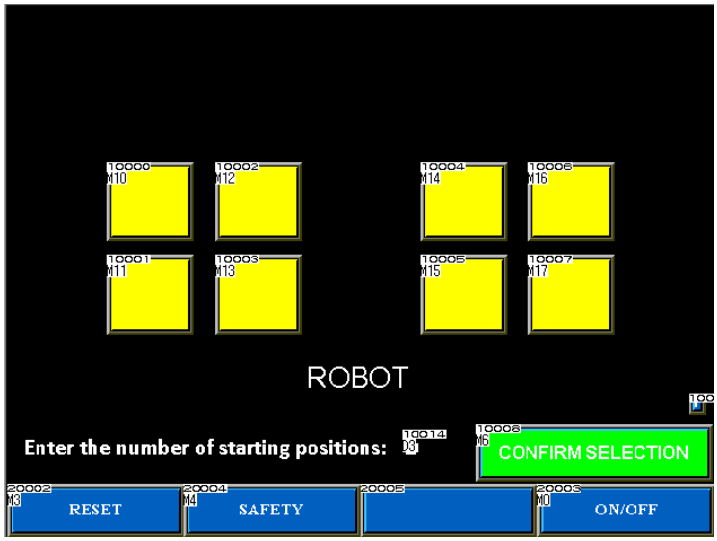
Celá úloha se ovládá pouze pomocí HMI panelu. V případě, že po obhlédnutí celého pracoviště bude zajištěn bezpečný provoz robota, tak se manuálně potvrdí *SAFETY* a po stisknutí tlačítka *ON/OFF* se zapne celé pracoviště. Pro splnění bezpečnosti by měl být robot oplocen, na školícím pracovišti toto není splněno, proto musíme manuálně potvrdit, že robotovi nic nepřekáží a může bezpečně pracovat.



Obr. 25 – První obrazovka

Po zapnutí se objeví další obrazovka (viz obr. 26) umožňuje volbu počátečních pozic. Nejdříve je nutné zadat počet počátečních pozic, kliknutím na číslo *Enter the number of starting positions*. Poté je možné volit dané pozice. Důležité je, že počet zvolených pozic musí souhlasit se zadaným číslem, jinak nebude umožněno pokračování v programu (*Confirm Selection* bude červené).

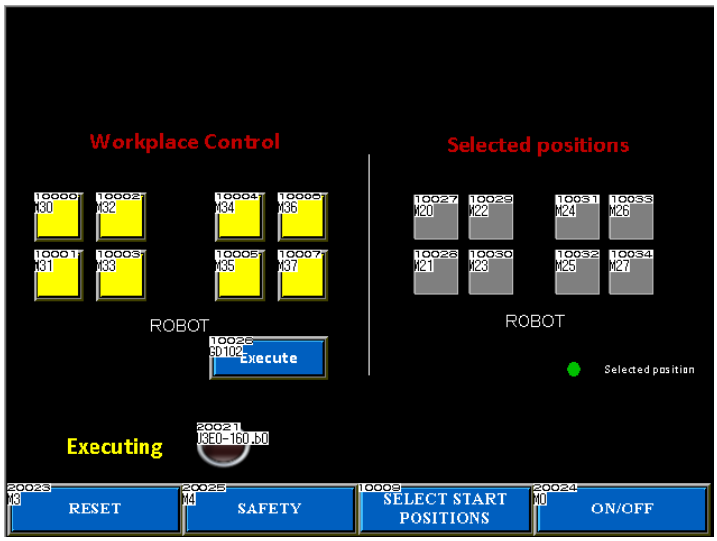
Jakmile tlačítko *Confirm Selection* zezelená, je možné ho stisknout a přejít k ovládání úlohy. Tuto skutečnost vysvětluje také informativní obrazovka, která se objeví po stisknutí tlačítka i.



Obr. 26 – Druhá obrazovka

Když přejdeme k samotnému ovládání úlohy, tak na HMI panelu bude aktuální obrazovka 3, která je na obrázku 26. Nejdříve je potřeba vybrat pozici, na které má robot vykonat činnost (vložení/vyjmutí míčku) a potom se pozice potvrdí stiskem *Execute*. Řídící algoritmus hlídá, zda robot drží míček nebo ne, proto stačí pouze vybírat pozice. Když se vybere špatná pozice, tak tlačítko *Execute* bude červené a nebude možné potvrdit pozici. Po potvrzení pozice robot začne vykonávat program a začne blikat *Executing*, po skončení přestane blikat, přepíší se pozice a je možné další ovládání.

Změna pozic se provede stisknutím *Select start positions* a reset celého pracoviště zase stisknutím tlačítka *Reset*.



Obr. 26 – Třetí obrazovka

5.5 Simulace

První testování programu probíhalo offline – v simulátoru GX Simulator3 a v případě HMI panelu to byl GT Simulator3. Pomocí simulací jsem postupně ladil program a jeho nedostatky, finální simulaci jsem prováděl pouze pro PLC a HMI s tím, že jsem pozoroval, jestli se proměnné propisují do programu robota. Robota jsem nesimuloval z časových důvodů, ale také proto, že když jsem měl možnost testovat na reálném robotu, tak jsem toho využil a s užitečnými informacemi dalších kolegů jsem testoval na reálném pracovišti.

Problém simulací je ten, že v nich mohou být chyby. I v mém případě to tak bylo. Při ladění programu v simulaci a monitoringu proměnných nesepnula proměnná, která měla sepnout. Když jsem ten samý program odzkoušel na reálném pracovišti, vše fungovalo správně. Jinak byla simulace velice užitečná a čas, který jsem na pracovišti mohl strávit laděním programu, jsem trávil postupným spouštěním a tvorbou komunikace mezi jednotlivými komponenty pracoviště.

5.6 Testování na reálném robotickém pracovišti

Na reálném pracovišti bylo zapotřebí trochu více příprav než bylo možné spustit celý program. Nejdříve bylo třeba nastavení správných IP adres, protože komunikace probíhá přes CC-Link IE Field, který je zprostředkován pomocí ethernetového kabelu. IP adresa PLC je 172.16.16.1, pro HMI je to zase 172.16.16.2. Tyto IP adresy se nastaví v projektech příslušných vývojových prostředí.

Poté už bylo možné propojit projekty s reálnými komponenty. Na pracovišti byl i tzv. *síťový switch*, který propojuje jednotlivé prvky do společné sítě. Je to jednodušší v tom, že se stačí připojit ke switchi a není potřeba se připojovat k jednotlivým komponentům pracoviště. Switch má výhodu oproti podobnému zařízení – *hubu*. Tou výhodou je, že switch posílá informace pouze do směrů, do kterých je třeba.

Před nahráním programů do komponentů bylo také zapotřebí naučit robota příslušné pozice. Učení pozic probíhalo pomocí pendantu v modu Jog. Aby se robot mohl pohybovat, musel jsem držet bezpečnostní tlačítko, které spínalo servomotory robota. Robot se může pohybovat několika možnostmi. Prvním typem je pohyb v souřadnicích XYZ, dalším může být např. pohyb jednotlivých os. Pro doladění pozic je manipulace celým robotem nepraktická,

proto se využívá dalšího typu pohybu *Tool*. Tento typ umožňuje hýbat robotem okolo referenčního bodu, kterým je nejčastěji upínka.

Po připojení přišlo na řadu nahrání programů do komponentů. Při správně nastavené komunikaci bylo nahrání jednoduché. Následovalo ladění programu na reálném pracovišti. Několikrát jsem musel poupravovat především program pro robota, protože jsem v minulosti neměl zkušenosti s programováním robota a ten se často choval odlišně než jsem předpokládal.

5.7 Porovnání

Pokud se podívám na rozdíly mezi simulací a reálnou aplikací, tak to už jsem zmiňoval výše. Tyto rozdíly jsou především v rychlosti zpracování programu. Simulace trvá déle. Trvá déle také než se změní daná proměnná, takže je potřeba často po stisknutí tlačítka počkat, než se program aktualizuje. To je celkem nepříjemná věc, protože když budu pokračovat než se program přepíše, tak může dojít k chybě v programu.

Dalším rozdílem je určitě také to, že v simulaci je horší si představit reálné chování celého pracoviště. Toto jsem zjistil např. u chování robota, u kterého jsem měl o jeho chování ze simulace zcela jiné představy.

Celkově bych řekl, že obě metody ladění programu mají své přednosti. Simulace je užitečná v případě, že nejsou k dispozici reálné komponenty a používá se v případě základního ladění programu. Na druhou stranu, při uvádění do provozu pracoviště bychom se neobešli bez ladění na daném pracovišti.

Obr. 27 – Reálné pracoviště ovládané HMI panelem

6 Závěr

Cílem bakalářské práce bylo navrhnout a realizovat robotické pracoviště a naprogramovat vybranou školící aplikace. Pracoviště bylo připravené s dodaným hardwarem ve školící místnosti společnosti B:TECH, a.s. Nicméně v mém případě nebyla nutnost využití všech komponentů pracoviště, takže jsem přebytečné moduly odpojil od společné sběrnice. Celé pracoviště je zapojeno podle blokového schématu na obrázku 16 a náhled pracoviště je zdokumentován na obrázku 17.

Důležitou úlohu hraje v aplikaci robot, který je řízen PLC a ovládání úlohy zajišťuje HMI panel. Videa dokumentující funkčnost pracoviště jsou na přiloženém CD (viz přílohy), stejně tak jsou tam 2 programy pro PLC, vizualizace a program pro robota. Programy se liší pouze v řízení programu. Zatímco jeden program je řízen na základě pomocného bitu, tak druhý program je řízen sofistikovaněji – pomocí informací o elektrické upínce z robota. Postup při programování vychází z vývojového diagramu na obrázku 9 a je popsán v kapitole 5.2.

Dalším z cílů bylo nastudování možností komunikace mezi jednotlivými komponenty. Tyto možnosti jsou popsány v kapitole 2.5. Těchto možností je určitě více, já jsem vybral a popsal pouze ty nejvýznamnější. Protože všechen hardware byl od firmy Mitsubishi Electric, tak je i komunikační protokol používaný na pracovišti od stejné firmy a to sice CC-Link IE. Zařízení se připojují do CC-Linku pomocí ethernetového kabelu s konektory RJ-45.

Celá školící aplikace by se nechala vylepšit třeba přidáním optických senzorů na jednotlivé pozice. Tím by se zjednodušil program, protože pozice by nemusely být kontrolovány pomocí pomocných proměnných, ale kontrolovali by se na základě senzorických dat. Vylepšilo by to také zadávání startovacích pozic. Rozmístili by se míčky a pak by stačilo pouze aktualizovat data od senzorů. Senzory by se propojili se vstupně výstupním modulem PLC.

Dalším, tentokrát programovým vylepšením by mohlo být zjednodušení programu pro robota, který je psán pouze s účelem funkčnosti a není typický pro programátory. Vylepšit by se nechala například funkce *SELECT*, kde by místo napevno daných pozic mohlo být pouze pár základních pozic a pomocí roztečí by se měnily dané souřadnice. Tento způsob by byl po programátorské stránce vhodnější, ale vzhledem k tomu, že jsem s tímto prostředím pracoval poprvé, tak mi vyhovoval mnou zvolený způsob, protože je přehlednější při ladění programu.

Technologie PLC je bezpochyby do budoucnosti velice perspektivní. Používá se stále ve větší míře a důležitou roli hraje i v tzv. *průmyslu 4.0*. Tento trend digitalizace a robotizace

výroby odstartoval čtvrtou průmyslovou revolucí. Cílem průmyslu 4.0 je větší propojení digitálního světa se světem reálným, zrychlení výroby a snížení ceny výroby. Technologie PLC nahradila rozměrné a nepraktické reléové systémy a po půl století používání už vývoj nepokračuje takovým tempem jako v minulosti. Současný vývoj se zaměřuje na kompaktnost, menší rozměry, rychlost cyklů a samozřejmě také možné snížení ceny. Dalším důležitým směrem ve vývoji je určitě komunikace. Vyvíjejí se technologie, které jsou schopny komunikovat na stále větší vzdálenosti, popř. bezdrátová komunikace. Setkal jsem se také s PLC ovládaným přes mobilní aplikaci, což je také velice zajímavá možnost do vývoje.

HMI panely také neustále procházejí dalším vývojem. Co se týká těchto panelů, tak se vyrábějí stále větší a samostatnější panely, některé dokonce mohou pracovat bez PLC protože miniaturní PLC je zabudováno uvnitř těchto panelů. V dnešní době se stále více používají SCADA systémy, takže vývoj se také zaměřuje na tyto systémy.

Vývoj robotických manipulátorů probíhá v dnešní době velice intenzivně, poněvadž se jedná o poměrně novou technologii, navíc se ukazuje jako užitečná. Oblast využití robotů se neustále zvětšuje, používá se například pro sváření, šroubování, přesouvání věcí, atd.

7 Použitá literatura

[1] ZÁRYBNICKÝ, Tomáš, Miroslav JAROŠ, Dana TREFÍLKOVÁ, Bohumil VESELÝ a Lubomír JANYŠKA. Typy PLC. *ELUC Automatizace: Kompaktní a modulární PLC* [online]. Olomouc: MŠMT, 2015 [cit. 2020-03-23]. ISBN MSMT-7521/2015-40. Dostupné z: https://eluc.kr-olomoucky.cz/verejne/ucebnice/22/lekce

[2] Modulární PLC od firmy Mitsubishi Electric. In: *AutoCont: MELSEC iQ-R* [online]. Moravská Ostrava [cit. 2020-03-23]. Dostupné z: http://www.accs.cz/produkty-mitsubishi-electric/ridici-systemy/modularni/melsec-r

[3] Historie PLC. *PLC Automatizace* [online]. HaPeSoft [cit. 2020-03-23]. Dostupné z: http://plc-automatizace.cz/index.htm

[4] Programovatelné logické automaty Řada MELSEC FX, Příručka pro začátečníky, 2009, Mitsubishi Electric Corporation. Dostupné z: http://www.accs.cz/Files/FA/PLC/FX_Prirucka_pro_zacatecniky.pdf

[5] ČSN EN 61131-3. *PLC: Programovací jazyky*. Praha, 2003.

[6] *Automa - časopis pro automatizační techniku s.r.o.* [online]. Děčín, 2011, **2011**(11) [cit. 2020-03-23]. ISSN 1210-9592. Dostupné z: https://automa.cz/Aton/FileRepository/pdf_articles/44868.pdf

[7] JANEČEK, Josef, Jindřich KRÁL, Gunnar KÜNZEL, et al. *Automatizace a automatizační technika: systémové pojetí automatizace*. Brno: Computer Press, 2014. ISBN 978-80-251-3628-7.

[8] Logické funkce. *Mylms.cz* [online]. 2018 [cit. 2020-04-13]. Dostupné z: https://www.mylms.cz/logicke-funkce-and-nand-or-nor-xor-not-yes-znaceni-pravdivostni-tabulky/

[9] VOJÁČEK, Antonín. Programovací jazyky podle IEC 61131-3. *HW server: Automatizace* [online]. 2011 [cit. 2020-05-01]. Dostupné z: https://automatizace.hw.cz/programovaci-rezimy-pro-plc-dle-iec-611313-codesys

[10] HMI panely. In: *Siemens: Panely SIMATIC HMI Basic* [online]. [cit. 2020-04-25]. Dostupné z: https://new.siemens.com/cz/cs/products/automation/simatic-hmi/panels/basic-panels.html

[11] MICHALEC, Libor. Průmyslová komunikace. *Automatizace: Ethernet vs. sériové protokoly v průmyslu* [online]. Praha: HW server, 2020 [cit. 2020-05-01]. Dostupné z: https://automatizace.hw.cz/ethernet-vs-seriove-protokoly-v-prumyslu.html

[12] PROFIBUS PA. *Praktická teorie: Základní informace o průmoslové sběrnici PROFIBUS - část I.* [online]. Liberec: Foxon [cit. 2020-05-01]. Dostupné z: https://www.foxon.cz/blog/prakticka-teorie/162-zakladni-informace-o-prumyslove-sbernici-profibus-cast-i

[13] PROFINET. *Praktická teorie: A tohle jste o PROFINETU věděli? 2. díl průmyslové komunikace* [online]. Liberec: Foxon [cit. 2020-05-01]. Dostupné z: https://www.foxon.cz/blog/prakticka-teorie/206-a-tohle-jste-o-profinetu-vedeli-2-dil-prumyslove-komunikace

[14] CC-Link. *CC-Link technologie* [online]. Brno: Phoenix Contact [cit. 2020-05-01]. Dostupné z: https://www.phoenixcontact.com/online/portal/cz?1dmy&urile=wcm:path:/czcs/web/main/products/subcategory_pages/CC_Link_P-08-12-15/7fe3242f-c866-4984-baa6-e3bf4272962d

[15] CC-link IE. *CC-Link IE: Vyrábět rychlostí světla* [online]. Ostrava: Mitsubishi Electric [cit. 2020-05-08]. Dostupné z: https://cz3a.mitsubishielectric.com/fa/cs/dl/5614/213387.pdf

[16] CANbus. *TSS Group: Co je to CANbus?* [online]. Zlín: TSS Group [cit. 2020-05-08]. Dostupné z: https://www.tssgroup.cz/item/co-je-to-can-bus-/

[17] CANbus. *MotoFocus.cz: Jak a proč diagnostikovat datovou sběrnici CAN-BUS* [online]. Bohumín: MotoFocus EU [cit. 2020-05-08]. Dostupné z: https://motofocus.cz/technika/26453,jak-a-proc-diagnostikovat-datovou-sbernici-can-bus

[18] VOJÁČEK, Antonín. DeviceNet. *Automatizace: Průmyslová sběrnice DeviceNet* [online]. Praha: HW server, 2014 [cit. 2020-05-08]. Dostupné z: https://automatizace.hw.cz/prumyslove-sbernice-a-komunikace/prumyslova-sbernice-devicenet.html

[19] FIGINI, Alessandro. EtherCAT. *Automatizace: EtherCAT Automation Protocol* [online]. Praha: HW server, 2017 [cit. 2020-05-08]. Dostupné z: https://automatizace.hw.cz/ethercat-automation-protocol.html

[20] VOJÁČEK, Antonín. MODBUS. *Automatizace: MODBUS* [online]. Praha: HW server, 2004 [cit. 2020-05-08]. Dostupné z: https://automatizace.hw.cz/clanek/2004070701

[21] VACEK, František a Holger ZELTWANGER. CANopen. *Automa: CANopen – vyšší komunikační protokol pro vestavné sítě*[online]. Děčín: Automa - časopis pro automatizační techniku [cit. 2020-05-08]. Dostupné z: https://automa.cz/cz/casopis-clanky/canopen-vyssi-komunikacni-protokol-pro-vestavne-site-2004_04_32279_854/

[22] ROMÁNEK, David. Co je CANopen. *Vývoj: Co je CANopen a jak na něj?* [online]. Praha: HW server, 2006 [cit. 2020-05-08]. Dostupné z: https://vyvoj.hw.cz/produkty/co-je-canopen-a-jak-na-nej.html

[23] KLOS, Oldřich. *Automatizace, regulace: Co je systém AS-Interface* [online]. Praha: MM průmyslové spektrum, 2007 [cit. 2020-05-08]. Dostupné z: https://www.mmspektrum.com/clanek/co-je-system-as-interface.html

[24] GX Works3 Manuál. *Help GX Works3* [online]. Praha: Mitsubishi Electric [cit. 2020-05-10]. Dostupné z: https://ie3a.mitsubishielectric.com/fa/en/dl/10994/sh081215.pdf

[25] GT Designer3 Manuál. *Help GT Designer3* [online]. Praha: Mitsubishi Electric [cit. 2020-05-10]. Dostupné z: https://dl.mitsubishielectric.com/dl/fa/document/manual/got/sh081220eng/sh081220engae.pdf

[26] RT ToolBox3 Manuál. *RT ToolBox3 Help* [online]. Praha: Mitsubishi Electric [cit. 2020-05-10]. Dostupné z: https://eu3a.mitsubishielectric.com/fa/en/dl/12817/RV-FR_Series_-_Standard_Specifications_Manual_bfp-a3470h.pdf

8 Seznam Použitých zkratek

- AI/AO – Analog Input/Analog Output
- CPU – Central Proessing Unit
- DI/DO – Digital Input/Digital Output
- FBD – Function Block Diagram
- GOT – Graphic Operation Terminal
- HMI – Human-Machine Interface
- IL – Instruction List
- LD – Ladder Diagram
- LED – Light-Emittnig Diode
- PLC – Programmable Logic Controller
- SCADA - Supervisory Control And Data Acquisition"
- ST – Structured Text

9 Seznam příloh

- Příloha 1 - CD s programy pro školící aplikaci a videi dokumentující funkčnost reálného pracoviště
    - Program pro PLC 2x, stejné programy, pouze jeden program obsahuje vstupy od robota, v druhém je to řešeno pomocným bitem
    - Vizualizace pro HMI
    - Program pro robota
    - 2 videa dokumentující činnost pracoviště