ZÁPADOČESKÁ UNIVERZITA V PLZNI

FAKULTA ELEKTROTECHNICKÁ

KATEDRA ELEKTROTECHNIKY A POČÍTAČOVÉHO MODELOVÁNÍ

BAKALÁŘSKÁ PRÁCE

Přípravek pro výuku Automatizační a řídicí techniky

Adam Samuel **2022**

ZÁPADOČESKÁ UNIVERZITA V PLZNI
Fakulta elektrotechnická
Akademický rok: 2021/2022

ZADÁNÍ BAKALÁŘSKÉ PRÁCE
(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Adam SAMUEL**
Osobní číslo:	**E19B0111P**
Studijní program:	**B2612 Elektrotechnika a informatika**
Studijní obor:	**Elektronika a telekomunikace**
Téma práce:	**Přípravek pro výuku Automatizační a řídicí techniky**
Zadávající katedra:	**Katedra elektroniky a informačních technologií**

Zásady pro vypracování

1. Popište programovatelný logický automat Siemens Simatic S7-1500 a jeho rozšiřující moduly.
2. Popište vývojové prostředí TIA Portal.
3. Popište programovací jazyky, kterými lze programovat PLC Siemens Simatic S7-1500.
4. Navrhněte a realizujte přípravek pro testování programů PLC. Přípravek má být vybaven alespoň 8 digitálními vstupy, 8 digitálními výstupy, 2 analogovými vstupy a 2 analogovými výstupy, a dále vhodnou indikací logické a analogové úrovně.
5. Navrhněte a realizujte alespoň 3 vzorové úlohy s využitím různých programovacích jazyků, na nichž bude možné demonstrovat funkci přípravku.

Rozsah bakalářské práce: **30 – 40**
Rozsah grafických prací: **dle doporučení vedoucího**
Forma zpracování bakalářské práce: **elektronická**

Seznam doporučené literatury:

Šmejkal, L.: PLC a automatizace 2, sekvenční logické systémy a základy fuzzy logiky. BEN – Technická literatura, Praha 2005
firemní materiály fy Siemens, s.r.o.

Vedoucí bakalářské práce: **Ing. Jiří Lahoda, Ph.D.**
Katedra elektrotechniky a počítačového modelování

Datum zadání bakalářské práce: **8. října 2021**
Termín odevzdání bakalářské práce: **26. května 2022**

Západočeská univerzita v Plzni
Fakulta elektrotechnická
L.S.

Prof. Ing. Zdeněk Peroutka, Ph.D.
děkan

Doc. Ing. Jiří Hammerbauer, Ph.D.
vedoucí katedry

V Plzni dne 8. října 2021

Abstrakt

Předkládaná bakalářská práce se zabývá návrhem a praktickou realizací přípravku pro výuku programování PLC primárně značky Siemens v prostředí TIA Portal. Dalším cílem práce je návrh, realizace a detailní popis řešení tří vzorových úloh, na kterých je funkčnost přípravku demonstrována. Zároveň se také práce zabývá stručným úvodem do problematiky PLC a konkrétním popisem automatu Siemens Simatic S7-1500, pro nějž je zároveň přípravek navržen. Součástí práce je i mimo jiné popis vývojového prostředí TIA Portal, v němž jsou vzorové úlohy řešeny.

Klíčová slova

Přípravek pro výuku programování PLC Siemens Simatic S7-1500, Řízení světelné křižovatky pomocí PLC, Řízení světelného železničního zabezpečovacího zařízení pomocí PLC, Řízení dopravníku ve výrobě pomocí PLC, Programování v prostředí TIA Portal

Abstract

Presented bachelor thesis deals with the design and practical implementation of the educational unit for teaching PLC programming, especially Siemens in TIA Portal IDE. Another goal of this thesis is design, programming and detailed description of three sample tasks, which are demonstrating functionality of the test unit. Simultaneously this thesis deals with breaf introduction into PLC problematics and specific description of Siemens Simatic S7-1500, for which the test unit is designed. The next part of this thesis is besides description of TIA Portal IDE, where the sample tasks are solved.

Key words

Test unit for teaching of programming PLC Siemens Simatic S7-1500, Controlling of traffic lights using PLC, Controlling of lights on a rail crossing using PLC, Controlling of conveyor in factory using PLC, Programming in TIA Portal IDE

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně, s použitím odborné literatury a pramenů uvedených v seznamu, který je součástí této diplomové práce.

Dále prohlašuji, že veškerý software, použitý při řešení této bakalářské práce, je legální.

.....

Podpis

V Plzni dne 25.5.2022 Adam Samuel

Poděkování

Tímto bych rád poděkoval VOŠ a SPŠE Plzeň za zapůjčení příslušného softwaru a možnost otestovat přípravek na tamějších PLC. Dále bych chtěl poděkovat vedoucímu bakalářské práce Ing. Jiřímu Lahodovi, Ph. D, za rady a profesionální přístup při konzultacích.

Obsah

OBSAH 8

ÚVOD 10

SEZNAM SYMBOLŮ A ZKRATEK 11

SEZNAM OBRÁZKŮ 12

SEZNAM TABULEK 14

1 PLC: PROGRAMOVATELNÉ LOGICKÉ AUTOMATY 15

1.1 POPIS A HISTORIE PLC 15
1.2 ČÁSTI PLC 15
1.3 DRUHY PLC 16
1.3.1 Kompaktní PLC 16
1.3.2 Modulární PLC 16

2 PROGRAMOVATELNÝ LOGICKÝ AUTOMAT SIEMENS SIMATIC S7-1500 17

2.1 ZÁKLADNÍ MODUL CPU 17
2.2 VSTUPNÍ A VÝSTUPNÍ MODULY 17
2.3 NAPÁJECÍ MODULY 18
2.4 KOMUNIKAČNÍ PROTOKOLY 18
2.5 HMI PANEL 19

3 VÝVOJOVÉ PROSTŘEDÍ TIA PORTAL 19

3.1 OBECNÝ POPIS PROSTŘEDÍ TIA PORTAL 19
3.2 KONFIGURACE PLC A JEDNOTLIVÝCH MODULŮ 20
3.3 STRUKTURA PROGRAMU 21
3.3.1 Hlavní část programu (Main) 21
3.3.2 Funkce (FC) 21
3.3.3 Funkční blok (FB) 21
3.3.4 Datový blok (DB) 21
3.4 PROGRAMOVACÍ JAZYKY V PROSTŘEDÍ TIA PORTAL 22
3.4.1 Jazyk reléových (kontaktních) schémat LAD 22
3.4.2 Jazyk logických schémat FBD 22
3.4.3 Jazyk mnemokódů STL 23
3.4.4 Jazyk strukturovaného textu SCL 23
3.4.5 Jazyk pro sekvenční programování GRAPH 23
3.5 VYBRANÉ DATOVÉ TYPY V PROSTŘEDÍ TIA PORTAL 24
3.6 TAGY 25
3.7 MOŽNOSTI SIMULACE CHODU PROGRAMU A DIAGNOSTIKA CHYB 26
3.8 TVORBA UŽIVATELSKÉHO ROZHRANÍ NA HMI PANELU 27

4 NÁVRH A REALIZACE PŘÍPRAVKU 28

4.1 NÁVRH PŘÍPRAVKU 28
4.2 PRAKTICKÁ REALIZACE PŘÍPRAVKU 32

5 VZOROVÉ ÚLOHY 33

5.1 ÚLOHA 1 – DOPRAVNĚ ŘÍZENÁ SVĚTELNÁ KŘIŽOVATKA 33
5.1.1 Zadání 33
5.1.2 Řešení 34
5.2 ÚLOHA 2 – ŘÍZENÍ SVĚTELNÉHO ŽELEZNIČNÍHO PŘEJEZDOVÉHO ZABEZPEČOVACÍHO ZAŘÍZENÍ 38
5.2.1 Zadání 38
5.2.2 Řešení 39

6 ÚLOHA 3 – ŘÍZENÍ DOPRAVNÍKU VÝROBKŮ S VÁHOU 45

6.1.1 Zadání 45
6.1.2 Řešení 46

ZÁVĚR 54

SEZNAM LITERATURY A INFORMAČNÍCH ZDROJŮ 1

Úvod

Předkládaná bakalářská práce se zabývá návrhem a praktickou realizací přípravku pro výuku programování PLC Siemens Simatic S7-1500 v prostředí TIA Portal. Přípravek byl vyvinut ve spolupráci s VOŠ a SPŠE Plzeň pro tamější PLC a jeho rozšiřující moduly. Studenti zde dosud ověřovali funkčnost svých programů pouze vytvořením uživatelského rozhraní na HMI panelu, což vedlo k neefektivnímu využití rozšiřujících modulů, kterými školní PLC disponují. Díky možnosti fyzického zapojení přípravku na samotné moduly PLC si studenti procvičí například správné nastavování adres jednotlivých tagů v programech a lépe pochopí problematiku rozšiřujících modulů.

Dalším cílem této práce je návrh, realizace a popis řešení tří vzorových úloh, na kterých je možné funkčnost přípravku demonstrovat. Úlohy jsou navrženy tak, aby si čtenář mohl situaci snadno představit. Jedná se konkrétně o úlohu s řízením světelné křižovatky, úlohu s řízením světelného železničního přejezdového zabezpečovacího zařízení a úlohu s řízením dopravníku výrobků s váhou.

K celé bakalářské práci je přistupováno tak, aby mohla případně sloužit jako podpůrný výukový materiál pro výuku programování PLC.

Seznam symbolů a zkratek

PLC ..... Programmable Logic Controller. Programovatelný logický automat.

LAD ..... Ladder Diagram. Jazyk reléových schémat.

FBD ..... Function Block Diagram. Jazyk logických schémat.

STL ..... Statement List. Jazyk mnemokódů.

SCL ..... Structured Control Language. Jazyk strukturovaného textu.

$GRAPH$ ..... Jazyk pro sekvenční programování.

CPU ..... Central Processing Unit. Centrální procesorová jednotka.

I/O ..... Input/Output. Vstup/výstup.

$TIA\ Portal$ ..... Totally Integrated Automation Portal. Vývojové prostředí.

$HMI\ Panel$ ..... Human Machine Interface Panel. Panel pro uživatelské rozhraní.

STP ..... Shielded Twisted Pair. Stíněná kroucená dvoulinka.

DI ..... Digital input. Digitální vstup.

DQ ..... Digital output. Digitální výstup.

AI ..... Analog input. Analogový vstup.

AQ ..... Analog output. Analogový výstup.

R_{pLED} ..... Předřadný odpor před LED diodou $[\Omega]$

U_{zdroje} ..... Výstupní napětí zdroje $[V]$

U_{LED} ..... Úbytek napětí na LED diodě $[V]$

I_{LED} ..... Proud odebíraný LED diodou $[A]$

P_{pLED} ..... Výkonová ztráta na předřadném odporu před LED diodou $[W]$

R_{pvolt} ..... Předřadný odpor před voltmetrem $[\Omega]$

R_{volt} ..... Vnitřní odpor voltmetru $[\Omega]$

R_{pot} ..... Odpor potenciometru $[\Omega]$

U_{volt} ..... Napětí na voltmetru $[V]$

Seznam obrázků

Obrázek 1: Okno části „Project view“ v prostředí TIA Portal 20
Obrázek 2: Nastavení jednotlivých modulů v části "Device configuration" 20
Obrázek 3: Ukázka jazyka LAD - Ovládání start/stop 22
Obrázek 4: Ukázka jazyka FBD - Ovládání start/stop 23
Obrázek 5: Ukázka jazyka STL - Ovládání start/stop 23
Obrázek 6: Ukázka jazyka SCL - Ovládání start/stop 23
Obrázek 7: Ukázka jazyka GRAPH - Blikání světla 24
Obrázek 8: Default tag table pro zapojení start/stop (nahoře), zobrazení, na kterých pinech se daný tag nachází v části "Device configuration" (dole) ... 26
Obrázek 9: Simulace chodu programu na PLC (vlevo nahoře), na HMI Panelu (vpravo nahoře), zobrazení cesty toku signálu v online režimu (dole) 27
Obrázek 10: Tvorba uživatelského rozhraní na HMI Panelu 28
Obrázek 11: Schéma zapojení pro výpočet předřadného odporu pro potenciometr na analogový vstup (AI) 30
Obrázek 12: Kompletní schéma zapojení přípravku 31
Obrázek 13: Fotografie realizovaného přípravku 32
Obrázek 14: Vnitřní zapojení přípravku 32
Obrázek 15: Úloha 1: Části Network 1 (nahoře) a Network 2 (dole) hlavní části programu (main) 34
Obrázek 16: Úloha 1: Část Network 3 hlavní části programu (main) 35
Obrázek 17: Úloha 1: Vnitřní struktura funkčního bloku "nocni_rezim" 36
Obrázek 18: Úloha 1: Vnitřní struktura funkčního bloku "denni_rezim" (vlevo) a detail na kroky S3, S4 a S5 (vpravo) 37
Obrázek 19: Úloha 1: Uživatelské rozhraní na HMI Panelu 38
Obrázek 20: Úloha 2: Vnitřní struktura funkčního bloku "klidovy_stav" 39
Obrázek 21: Úloha 2: Vnitřní struktura funkčního bloku "jede_vlak" 40
Obrázek 22: Úloha 2: Vnitřní struktura funkčního bloku "chybovy_stav" 41
Obrázek 23: Úloha 2: Část Network 1 hlavní části programu (main) 41
Obrázek 24: Úloha 2: Část Network 2 hlavní části programu (main) 42
Obrázek 25: Úloha 2: Část Network 3 hlavní části programu (main) 43
Obrázek 26: Úloha 2: Část Network 4 hlavní části programu (main) 44

OBRÁZEK 27: ÚLOHA 2: UŽIVATELSKÉ ROZHRANÍ NA HMI PANELU 45
OBRÁZEK 28: ÚLOHA 3: ČÁST NETWORK 1 FUNKČNÍHO BLOKU "STANDARDNI_REZIM" 47
OBRÁZEK 29: ÚLOHA 3: ČÁST NETWORK 2 FUNKČNÍHO BLOKU "STANDARDNI_REZIM" 48
OBRÁZEK 30: ÚLOHA 3: ČÁST NETWORK 3 FUNKČNÍHO BLOKU "STANDARDNI_REZIM" 50
OBRÁZEK 31: ÚLOHA 3: ČÁST NETWORK 4 FUNKČNÍHO BLOKU "STANDARDNI_REZIM" 51
OBRÁZEK 32: ÚLOHA 3: ČÁST NETWORK 5 FUNKČNÍHO BLOKU "STANDARDNI_REZIM" 51
OBRÁZEK 33: ÚLOHA 3: FUNKČNÍ BLOK "SERVISNI_REZIM" 52
OBRÁZEK 34: ÚLOHA 3: HLAVNÍ ČÁST PROGRAMU (MAIN) 53
OBRÁZEK 35: ÚLOHA 3: UŽIVATELSKÉ ROZHRANÍ NA HMI PANELU 53

Seznam tabulek

TABULKA 1: VYBRANÉ DATOVÉ TYPY V PROSTŘEDÍ TIA PORTAL [7] 25
TABULKA 2: ÚLOHA 1: TABULKA SE STAVY SEMAFORŮ, DOKUD U SEMAFORU NA VEDLEJŠÍ SILNICI NEČEKÁ ŽÁDNÝ AUTOMOBIL 33
TABULKA 3: ÚLOHA 1: TABULKA SE STAVY SEMAFORŮ, POKUD SE U SEMAFORU NA VEDLEJŠÍ SILNICI OBJEVÍ AUTOMOBIL 33

1 PLC: Programovatelné logické automaty

1.1 Popis a historie PLC

PLC neboli česky programovatelný logický automat je, dá se říct, počítač, jehož využití se uplatňuje především v průmyslu. S PLC se typicky setkáme v továrnách u řízení výrobních linek, dopravníků či regulace pohonů, ale lze se s nimi setkat i v jaderném inženýrství (po splnění příslušných předpisů a norem pro provoz v jaderném průmyslu). Předchůdcem dnešních PLC byla v prvopočátcích automatizace reléová logika, z níž také vychází základní programovací jazyk LAD (viz. kap. 3.4.1). Její nahrazení programovatelnými logickými automaty však přineslo mnoho podstatných výhod v podobě zvýšení spolehlivosti, snížení spotřeby, možnosti snadné změny funkce (přeprogramování) či komunikace po síti. Základním požadavkem na činnost PLC je především požadovaný výkon a plynulost jeho procesoru. Mezi nejvýznamnější výrobce programovatelných logických automatů patří například Siemens, ABB, Schneider Electric, Panasonic atd.

1.2 Části PLC

Základní částí každého PLC je modul CPU, v němž je zpracováván program. Ten je zpracováván v podobě zpracování vstupů a ovládání výstupů tak, aby bylo dosaženo minimální odchylky od zadaného stavu. Operační systém PLC se stará o cyklický (opakovaný) chod programu. Cyklus chodu programu PLC se liší podle druhu PLC (například větší modulární PLC využívají řadič sběrnice a celkově je cyklus složitější). [1] Jednoduchý cyklus malého PLC může pro názornost vypadat následovně:

1. Vykonání systémových operací v CPU (vnitřní kontrola, komunikace s programátorem apod.)
2. Načtení stavů všech vstupních periferií a jejich uložení do paměti pro zajištění jednoznačnosti. Tento stav, kdy jsou stavy vstupních periferií načteny do paměti, se nazývá obraz procesních vstupů.
3. Zpracování programu uloženého v paměti shora dolů a vytváření obrazu procesních výstupů. Stavy výstupních signálů tedy nejsou zapisovány přímo na fyzické výstupy, ale do paměti.

4. Po skončení programu jsou stavy výstupních signálů zapsány z paměti na fyzické výstupy. [2]

Napájecí moduly, které zajišťují napájení, jak modulu CPU, tak celého PLC systému, jsou jeho nezbytnou součástí. V praxi je nejčastěji využívána 24 voltová logika, tudíž jsou tyto zdroje nejčastěji s výstupním napětím 24 V stejnosměrných.

Další podstatnou částí PLC jsou vstupní a výstupní moduly. Ty mohou být buďto analogové či častěji digitální. Jejich počet se odvíjí od složitosti dané úlohy, kterou PLC řeší, a především od druhu PLC.

1.3 Druhy PLC

1.3.1 Kompaktní PLC

Jedná se o levnější variantu, která v jednom modulu integruje modul CPU, digitální a analogové vstupy či výstupy, rozhraní pro komunikaci, napájecí modul apod. Kompaktní PLC jsou díky nižšímu výpočetními výkonu využívány pro jednodušší automatizační úlohy. Díky poměrně nízké ceně nachází uplatnění nejen v průmyslových aplikacích, ale i např. v chytrých domácnostech, kde mohou řídit třeba automatickou kotelnu. Nevýhodou je však omezený počet vstupů a výstupů a částečná nebo úplná nemožnost jeho rozšíření. Typickým příkladem kompaktního PLC jsou PLC řady Siemens LOGO.

1.3.2 Modulární PLC

Modulární PLC jsou hojně rozšířené zejména v průmyslových aplikacích. Oproti kompaktním jsou několikanásobně dražší, nabízí však vyšší výpočetní výkon, a především možnost přidávat moduly dle potřeby pro danou aplikaci. Výsledná cena celého PLC systému tedy závisí na požadavcích zákazníka, například kolik a jakých modulů požaduje. Typickým příkladem modulárního PLC systému je řada Siemens Simatic, se kterou budeme nadále v této práci pracovat.

2 Programovatelný logický automat Siemens Simatic S7-1500

Siemens Simatic S7-1500 označení pro řadu modulárních PLC, které umožňují k základnímu modulu PLC (u značky Siemens označeno jako CPU) přidávat další moduly dle potřeby. Typicky se jedná například o moduly analogových či digitálních vstupů a výstupů, komunikační moduly, technologické moduly apod. V této kapitole si řadu S7-1500 popíšeme.

2.1 Základní modul CPU

Řada Simatic S7-1500 nabízí tři základní procesorové jednotky s označením odstupňovaným dle výkonových charakteristik. Ty nesou označení 1511, 1513 a 1516 (s posledním jmenovaným modulem budeme dále pracovat v této práci). K modulu CPU lze připojit až 32 rozšiřujících modulů a disponuje výkonnou systémovou sběrnicí, která umožňuje rychlý přenos dat a použití efektivního komunikačního protokolu. Výrobce udává dobu odezvy mezi odpovídajícími I/O svorkami maximálně 500 µs a dobu potřebnou ke zpracování logické instrukce kratší než 10 ns. Kromě výkonnosti je další předností této řady především bezpečnost. Moduly CPU obsahují standardně funkce chránící program před kopírováním či modifikací a bývají také vybaveny mechanickými plombami či zámky pro omezení možnosti fyzicky se připojit k CPU. [3]

2.2 Vstupní a výstupní moduly

Jedná se o nejčastější rozšiřující moduly u modulárních PLC. V zásadě rozlišujeme, zda se jedná o analogové či digitální moduly vstupů či výstupů. U modulů digitálních vstupů pracujeme v našem případě pouze s 24 voltovou logikou, což znamená vysokou úroveň v podobě +24 V a nízkou úroveň 0 V. U digitálních výstupů máme možnost pracovat buďto s napěťovou úrovní 0 V a 24 V, nebo s proudovou úrovní až do 0,5 A.

U analogových vstupů je situace mírně složitější. Ty umožňují měřit analogovou hodnotu například v podobě napětí, proudu či přijímat hodnotu například z teplotního

senzoru. V našem případě budeme pracovat s napěťovým rozsahem ±10 V. Tyto rozsahy jsou předem definované výrobcem a je možné je dle potřeby nastavit přímo v prostředí TIA Portal. U modulů analogových výstupů je možné výstupní analogovou hodnotu vyjádřit pouze jako napětí nebo proud opět v předem definovaných rozsazích (i zde pracujeme s rozsahem ±10 V). Analogová hodnota z modulu je pak v programu načítána jako celé číslo formátu Int (obdobně číslo typu Int je možné zobrazit formou napětí na analogovém výstupu).

2.3 Napájecí moduly

Používají se k napájení celého CPU včetně modulů. Jedná se zpravidla o zdroje se stejnosměrným napětím 24 V o různých výkonech (v našem případě 190 W).

2.4 Komunikační protokoly

PLC řady Simatic S7-1500 disponují také podporou protokolů PROFINET a PROFIBUS pro připojení dalších periferií. Tyto protokoly nevyžadují žádné další moduly, jsou již součástí jednotky CPU. CPU 1516 disponuje hned třemi PROFINET porty pro snadné oddělení komunikace řízení procesu od informační sítě podniku. Dále také podporuje funkci webového serveru, který umožňuje vzdálený přístup k provozním a systémovým údajům či snadnou diagnostiku systému. [3]

V našem případě využijeme PROFINET k propojení CPU jednotky s HMI Panelem. Z technického hlediska se jedná o ekvivalent klasického Ethernetu a propojení je realizováno STP kabelem s konektory RJ-45.

Protokol PROFIBUS je využíván například k propojení modulu CPU s řadou frekvenčních měničů Siemens Sinamics. Díky tomu je možné pomocí PLC naprogramovat ovládání elektrického pohonu. Tomu také přispívá funkce PID regulátoru, kterou PLC řady Simatic S7-1500 nabízí.

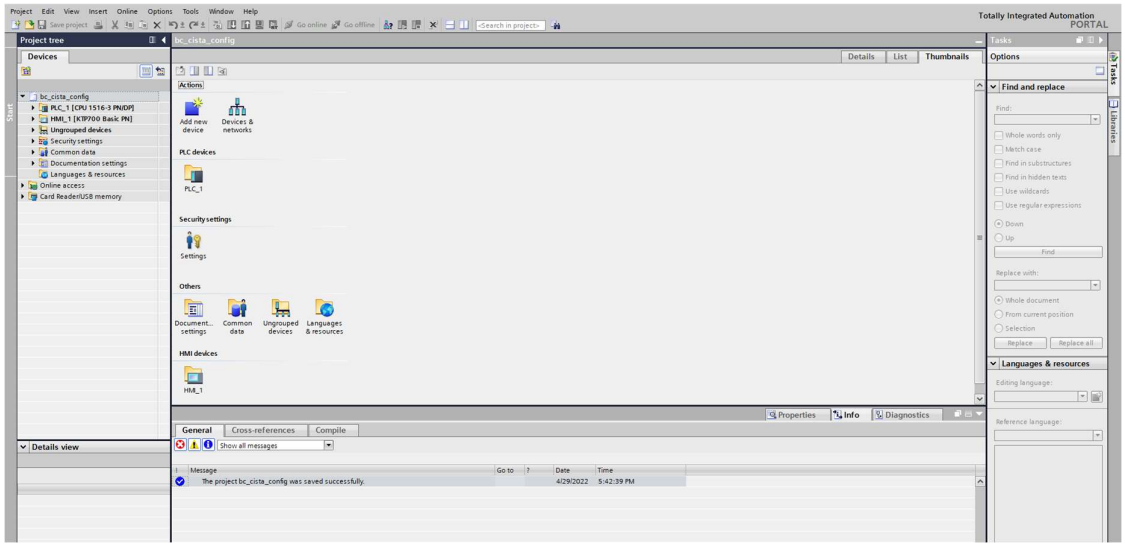
2.5 HMI Panel

HMI Panel neboli Human Machine Interface Panel je panel (resp. celá řada panelů) sloužící k ovládání chodu programu obsluhou. Typicky se jedná o zobrazování či zadávání číselných hodnot, signalizace logických stavů nebo dokonce u dotykových panelů možnost nadefinování tlačítek namísto fyzických tlačítek připojených k modulům PLC. V běžné praxi je celý PLC systém zabudován v rozvaděčové skříni, do které má přístup například údržba a HMI panel je připevněn zvenku. Méně odborná obsluha pak může pracovat pouze na HMI Panelu, aniž by musela jakkoliv manipulovat přímo s moduly PLC. V naší práci budeme pracovat s dotykovým HMI Panelem KTP700 Basic.

3 Vývojové prostředí TIA Portal

3.1 Obecný popis prostředí TIA Portal

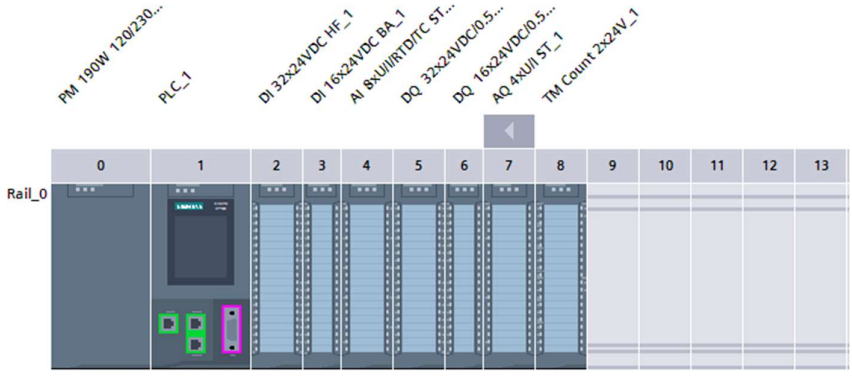
TIA Portal neboli Totally Integrated Automation Portal je vývojové prostředí vyvinuté společností Siemens. Již název napovídá, že v rámci jednoho prostředí nalezneme veškeré potřebné funkce pro komplexní návrh celého automatizovaného systému, jako například operátorské rozhraní pro dispečink, funkce pro návrh komunikačních sítí, elektrických pohonů apod. [4] Kromě samotného programování PLC tedy umožňuje konkrétně například nastavování parametrů jednotlivých modulů, nastavení komunikace mezi PLC a dalšími částmi systému (konkrétně například mezi PLC a inteligentními frekvenčními měniči), vytváření uživatelského rozhraní pro HMI Panely, diagnostiku a testování programů a mnoho dalších funkcí. Samotné prostředí TIA Portal je rozděleno na několik částí, v této práci však budeme výhradně pracovat v části „Project View“ (Obrázek 1). V této práci budeme pracovat v prostředí TIA Portal ve verzi v15.



Obrázek 1: Okno části „Project view“ v prostředí TIA Portal

3.2 Konfigurace PLC a jednotlivých modulů

Před začátkem vlastního psaní programu je nutné nastavit v prostředí TIA Portal modul CPU a další moduly, pro které program píšeme. To provedeme v části „Project view“, kde v levé straně nahoře nalezneme možnost „Device configuration“ (Obrázek 2). V ní máme dvě možnosti, jak jednotlivé moduly nastavit. Buďto vyhledávat jednotlivé moduly z nabídky ručně, anebo pokud jsme fyzicky připojeni k PLC, můžeme při výběru CPU vybrat možnost „Unspecified CPU 1500“ a TIA Portal sám pozná, s jakým CPU (včetně modulů k němu připojených) pracujeme. Je však nutné ještě manuálně vyhledat modul napájecího zdroje a HMI Panel.



Obrázek 2: Nastavení jednotlivých modulů v části "Device configuration"

3.3 Struktura programu

Struktura programu v TIA Portal je v zásadě obdobná, jako u vyšších programovacích jazyků, jako je například C++, C#, Python apod. Program zde můžeme skládat z dílčích celků, nebo jej psát přímo do hlavní části programu.

3.3.1 Hlavní část programu (Main)

Tato část je v programu přítomna vždy a zahrnuje veškeré funkce a funkční bloky. Zároveň také může řídit jejich jednotlivé ovládání (spouštění či vypínání bloků apod.) a vzájemnou komunikaci. Hlavní část programu bývá nejčastěji realizována v základním programovacím jazyku prostředí TIA Portal, tedy v jazyku LAD.

3.3.2 Funkce (FC)

Obdobně jako u jazyků jako například C, či C++ nemusí hlavní část programu vykonávat celý chod programu, ale můžeme jej složit z jednotlivých funkcí. U funkcí můžeme využít různých programovacích jazyků, které prostředí TIA Portal nabízí (viz. kap. 3.4) a můžeme v nich definovat lokální proměnné. Běžně se používá pro aritmetické výpočty.

3.3.3 Funkční blok (FB)

Funkční blok je specialitou prostředí TIA Portal. Stejně jako funkce je možné jej naprogramovat v různých jazycích a definovat v něm lokální proměnné. Na rozdíl od funkce však disponuje vazbou na svůj vlastní datový blok, zatímco obyčejná funkce žádný nemá. Funkce ztrácí v každém cyklu svůj obsah, ale funkční blok udržuje informace o proměnných právě v databázi. Další velkou výhodou je, že funkční blok umožňuje předávání parametrů dovnitř a ven z funkce. [5]

3.3.4 Datový blok (DB)

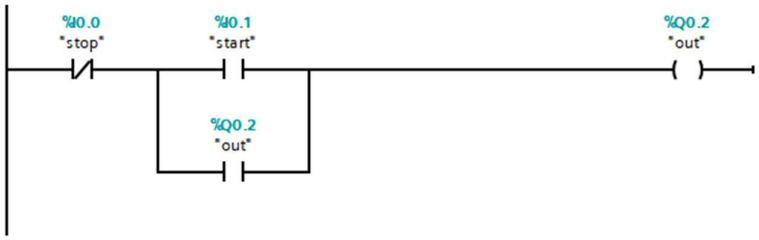
Ukládá informace o lokálních proměnných funkčního bloku, čítačích, časovačích apod.

3.4 Programovací jazyky v prostředí TIA Portal

K programování využívá TIA Portal několika programovacích jazyků. Některé jsou záležitostí PLC čistě od značky Siemens, s jinými (případně s jejich modifikacemi) se můžeme setkat i u jiných výrobců. Téměř všechny jazyky zde umožňují práci s logickými operacemi (logický součet, součin, negace apod.), práci s čítači a časovači, práci s matematickými operacemi, či práci s reálným časem. Jednotlivé jazyky a funkce, které nabízí, jsou velmi dobře popsány v nápovědě prostředí TIA Portal.

3.4.1 Jazyk reléových (kontaktních) schémat LAD

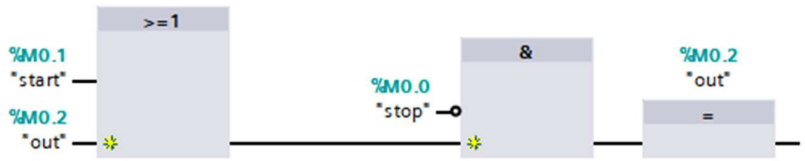
V PLC značky Siemens se jedná prakticky o základní jazyk, který vychází ze schémat reléové (kontaktní) logiky. Jedná se o grafický jazyk a s těmito schématy je prakticky totožný. Spínací kontakty jsou zde znázorněny dvěma svislými čarami, rozpínací kontakty mají mezi těmito dvěma čarami jednu čáru šikmou (Obrázek 3). Výstupy jsou znázorněny jakousi závorkou. Velikou výhodou jazyka LAD je poměrně snadná diagnostika chyb, jelikož v online módu umožňuje TIA Portal zobrazení, jakých hodnot, jaká proměnná nabývá, a především toku signálu.



Obrázek 3: Ukázka jazyka LAD - Ovládání start/stop

3.4.2 Jazyk logických schémat FBD

Jedná se opět o grafický jazyk, jehož jednotlivé logické operace jsou znázorněny, jako bloky. Proměnné jsou zde znázorněny jako vývody z jednotlivých bloků, negované proměnné (tzn. rozpínací kontakty) mají na vývodu z bloku kroužek (Obrázek 4). Jazyk FBD vychází z jazyka LAD a prostředí TIA Portal umožňuje jejich vzájemný překlad.



Obrázek 4: Ukázka jazyka FBD - Ovládání start/stop

3.4.3 Jazyk mnemokódů STL

Jedná se již o jazyk, kde jsou příkazy psány formou textu a zároveň jde o jakousi obdobu Assembleru u klasických PC. Podobně jako Assembler u PC je i STL u PLC orientován strojově. [6] Všimněme si, že vedle řádků kódu se po pravé straně nachází tabulka obsahující informace o tagu dané proměnné (Obrázek 5).

```
1     A     "start"     %M0.1
2     O(
3     A     "out"       %M0.2
4     )
5     AN    "stop"      %M0.0
6     =     "out"       %M0.2
```

Obrázek 5: Ukázka jazyka STL - Ovládání start/stop

3.4.4 Jazyk strukturovaného textu SCL

Jedná se o jazyk, který vzdáleně může svou strukturou připomínat vyšší jazyky, jako například C. Veškeré instrukce jsou zde psány formou textových příkazů, díky čemuž se stal oblíbeným u programátorů, kterým přijdou jazyky LAD či FBD nepřehledné.

```
1 IF NOT "stop" AND ("start" OR "out") THEN
2     "out" := TRUE;
3 END_IF;
```

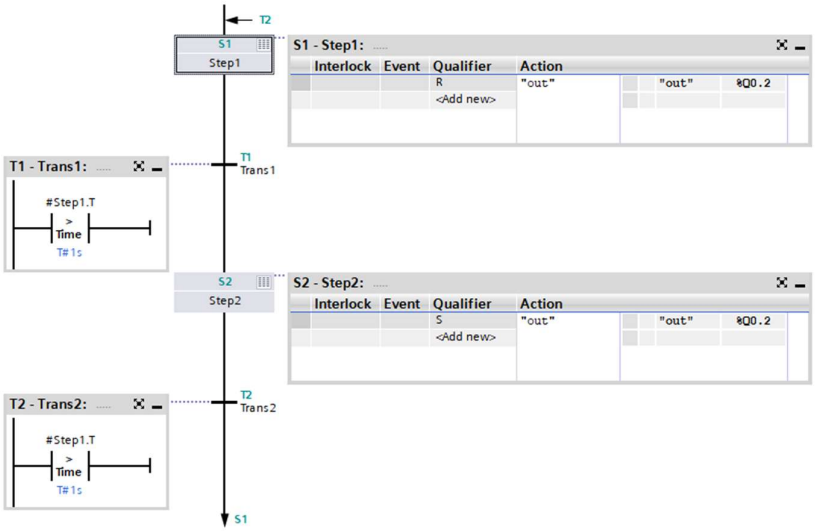
	"stop"	%M0.0
	"start"	%M0.1
	"out"	%M0.2
	"out"	%M0.2

Obrázek 6: Ukázka jazyka SCL - Ovládání start/stop

3.4.5 Jazyk pro sekvenční programování GRAPH

Tento jazyk je navržen výhradně pro sekvenční programy, které se vykonávají pořád dokola (například řízení světelné křižovatky – viz. kap. 5.1). Program je zde rozdělen na kroky, kde v každém z nich například nastavujeme konkrétní hodnoty pro dané proměnné a mezi nimi

jsou podmínky, které zajišťují při jejich vykonání přechod do dalšího kroku. Program zde můžeme různě větvit, případně mezi jednotlivými kroky realizovat skoky (například skok z konce programu zpět na začátek apod.).



Obrázek 7: Ukázka jazyka GRAPH - Blikání světla

3.5 Vybrané datové typy v prostředí TIA Portal

Datový typ všeobecně ve všech programovacích jazycích určuje rozsah hodnot, kterých může daná proměnná nabývat, kolik místa zabere v paměti a jaké operace lze s danou proměnnou provádět. Každý programovací jazyk může mít jiné datové typy, avšak v TIA Portal jsou všechny datové typy pro všechny jazyky stejné. Tabulka 1 obsahuje přehled základních, se kterými se mimo jiné setkáme i při řešení úloh v této práci (viz kap. 5).

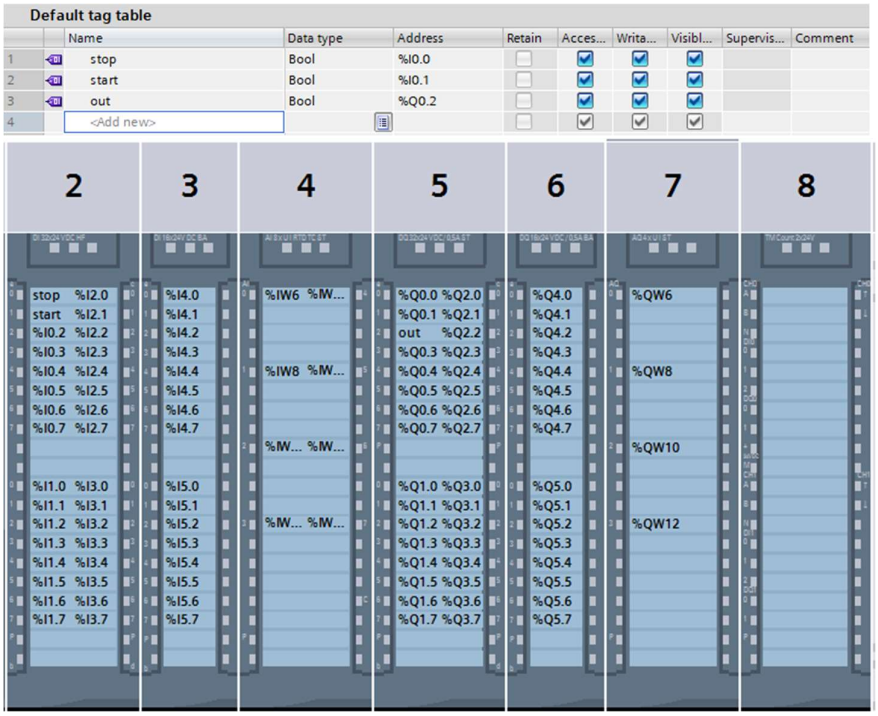
Tabulka 1: Vybrané datové typy v prostředí TIA Portal [7]

Datový typ	Konkrétní popis	Velikost [bit]	Rozsah hodnot
Bool	Zapnuto/vypnuto	1	True/False nebo logická ‚1'/logická ‚0'
Byte	Hexadecimální číslo	8	0 až FF
Int	Desítkové celé číslo se znaménkem	16	-32768 až 32767
UInt	Desítkové celé číslo bez znaménka	16	0 až 65535
Real	Desetinné číslo	32	$\pm 1{,}175495 \cdot 10^{-38}$ až $\pm 3{,}402823 \cdot 10^{38}$
Char	ASCII znak	8	

3.6 Tagy

Prostředí TIA Portal pracuje se všemi globálními proměnnými, jako s tzv. tagy. Ty obsahují informace o konkrétní proměnné. Kromě jejího datového typu se jedná především o informaci, zda se jedná o vstupní proměnnou (I), výstupní proměnnou (Q), nebo například o pomocnou proměnnou, uloženou v paměti PLC (M), která nemá vazbu na vstupní či výstupní modul PLC. Na základě těchto informací je vygenerována, či programátorem nastavena adresa dané proměnné například na konkrétní pin vstupního či výstupního modulu. V našem případě pracujeme s PLC, které je propojeno s HMI Panelem, pro který je nutné vytvářet vlastní tagy, které jsou následně propojeny s PLC tagy. Veškeré tagy pak lze upravovat v části „Default tag table" (PLC i HMI Panel má svoji vlastní).

Práci s jednotlivými tagy si můžeme vysvětlit na programu start/stop například z Obrázku 3. Tlačítka stop a start nadefinujeme jako vstupní proměnné a jelikož se jedná o proměnné typu bool, TIA Portal automaticky předpokládá, že se jedná o proměnné digitálních vstupů a každé přiřadí adresu volného digitálního vstupu. V části „Device configuration" (Obrázek 8 dole) můžeme po přiblížení vidět i na jakém konkrétním pinu modulu digitálních vstupů se jaká proměnná nachází podle toho, jakou má adresu. Obdobně výstupní proměnnou out můžeme nalézt na modulu digitálních výstupů. Adresy jednotlivých tagů může programátor libovolně měnit, avšak pokud bychom chtěli například proměnné typu bool přiřadit adresu na modul analogového vstupu, TIA Portal to automaticky vyhodnotí jako chybu.

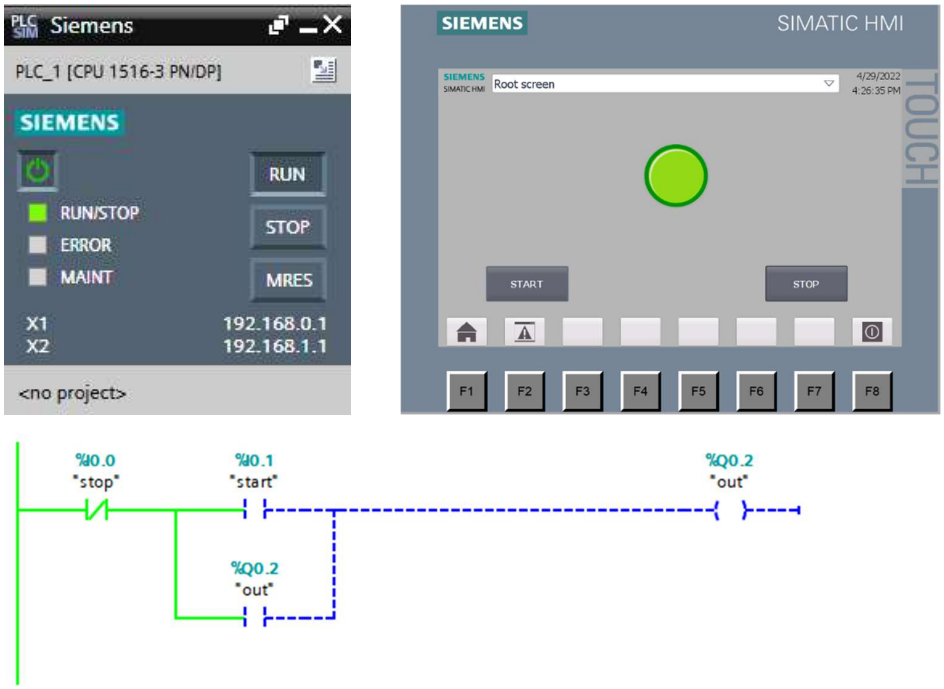


Obrázek 8: Default tag table pro zapojení start/stop (nahoře), zobrazení, na kterých pinech se daný tag nachází v části "Device configuration" (dole)

3.7 Možnosti simulace chodu programu a diagnostika chyb

Pro simulaci chodu programu PLC je TIA Portal vybaven nástrojem PLC SIM. Ten kromě simulace chodu programu PLC umožňuje i simulaci uživatelského rozhraní na HMI Panelu vytvořeného programátorem. Programátor si tedy může nasimulovat program přímo v prostředí TIA Portal ještě před tím, než jej nahraje fyzicky do PLC například ve výrobě. Je nutné však zdůraznit, že veškeré tagy při simulaci musí být adresovány jako paměťové (M). Na vstupní (I) či výstupní (Q) tagy PLC SIM nereaguje, respektive simuluje je jako fyzické vstupy a výstupy na modulech PLC. To znamená, že například u vstupní proměnné typu bool bude simulovat čekání na fyzický impulz do modulu PLC, který však nepřijde, jelikož jsme pouze v simulaci jeho chodu.

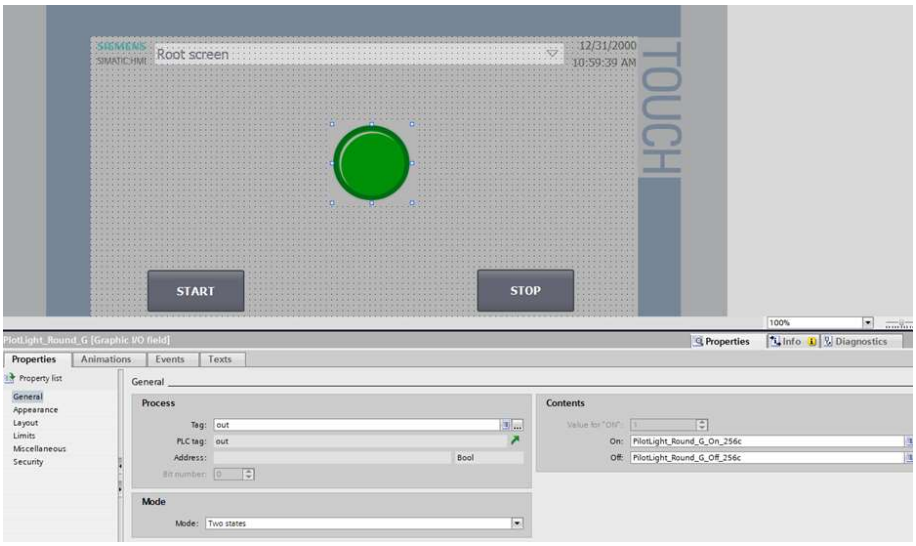
Prakticky u všech grafických jazyků je v online režimu prostředí TIA Portal možnost poměrně přehledně zobrazit tok signálu zapojením. Zároveň také zobrazuje hodnoty dané proměnné, což umožňuje programátorovi snadnou diagnostiku chyb.



Obrázek 9: Simulace chodu programu na PLC (vlevo nahoře), na HMI Panelu (vpravo nahoře), zobrazení cesty toku signálu v online režimu (dole)

3.8 Tvorba uživatelského rozhraní na HMI Panelu

Uživatelské rozhraní se na HMI panelu vytváří ryze graficky. Prostředí TIA Portal disponuje širokou knihovnou prvků, které je možné použít, nebo je možnost si vytvořit knihovnu vlastní. Typicky se jedná o tlačítka (button), přepínače (switch), světla různých barev (pilot light), prvky zobrazující číselné hodnoty apod. Prakticky u všech prvků je nejdůležitější nastavit správný HMI tag (případně PLC tag a TIA Portal mu vytvoří příslušný tag pro HMI), v případě tlačítek se ještě přiřazuje událost (event), co se má stát po stisku (ve všech našich programech se jedná o event „SetBitWhileKeyPressed“).



Obrázek 10: Tvorba uživatelského rozhraní na HMI Panelu

4 Návrh a realizace přípravku

V této kapitole si popíšeme návrh samotného přípravku, jehož realizace je jedním z hlavních cílů této práce. Tento přípravek by měl disponovat alespoň 8 digitálními vstupy (DI), 8 digitálními výstupy (DQ), 2 analogovými vstupy (AI) a 2 analogovými výstupy (AQ) a také vhodnou indikací logické a analogové úrovně.

4.1 Návrh přípravku

Při návrhu přípravku je nutno vzít v potaz požadavek na jeho robustnost a univerzálnost. Zároveň by na něm mělo být možné přehledně demonstrovat funkčnost vzorových úloh, které jsou dalším cílem této práce. Jelikož jedna z navržených úloh (viz. kap. 5) se zabývá dopravním řízením světelné křižovatky, kde je použito 10 světel různé barvy (tzn. 10 digitálních výstupů), disponuje náš přípravek celkem 11 digitálními výstupy s LED diodami v barvách semaforu. Jedenáctý digitální výstup, který indikuje logickou úroveň bílou LED diodou (označen DQ10), je zde kvůli další úloze týkající se světelného železničního přejezdového zabezpečovacího zařízení (viz. kap. 5.2).

K napájení celého přípravku využijeme modul napájecího zdroje u našeho PLC. Ten nám poskytne stejnosměrných 24 V o výkonu až 190 W, což k našemu účelu poslouží více než dostatečně.

Indikaci logické úrovně realizujeme LED diodami. Vysoká úroveň (resp. hodnota „true“) bude signalizována stavem, kdy LED dioda svítí, nízká úroveň naopak znamená, že daná LED dioda nesvítí. U digitálních vstupů jsou použity žluté LED diody, u digitálních výstupů jsou použity barvy semaforu a jedna bílá. Jelikož LED dioda vyžaduje zpravidla nižší napětí, než zmíněných 24 V, které nám poskytuje zdroj, je nutné vybavit všechny LED diody předřadnými odpory. Ačkoliv zde používáme různé typy LED diod, lze obecně říci, že ke svému rozsvícení potřebují všechny zhruba 1,5 V. Odběr diod bude pro každou také různý, ale obecně můžeme uvažovat 20 mA na LED diodu. Dostáváme se tedy k výpočtu předřadného odporu k LED diodě:

$$R_{pLED} = \frac{U_{zdroje} - U_{LED}}{I_{LED}} = \frac{24 - 1{,}5}{0{,}02} = 1125\ \Omega \qquad (1)$$

Vyšlo nám 1125 Ω, což však neodpovídá žádné odporové řadě, podle kterých se odpory vyrábí. Proto zvolíme hodnotu 1100 Ω, která odpovídá řadě E24. Tuto hodnotu odporu použijeme jak u digitálních vstupů, tak i u digitálních výstupů. Dále si vypočteme, na jaký výkon je nutno tyto rezistory dimenzovat.

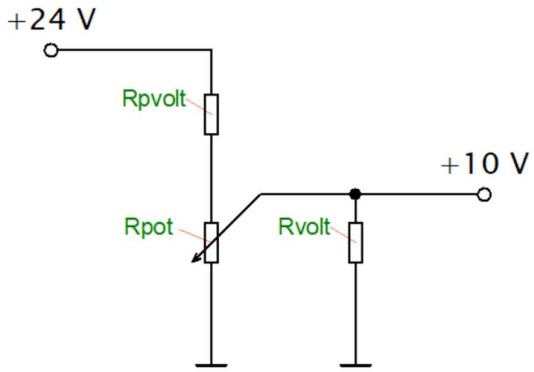
$$P_{pLED} = U_{zdroje} \cdot I_{LED} = 24 \cdot 0{,}02 = 0{,}48\ W \qquad (2)$$

Pro náš přípravek nám tedy postačí standardní rezistory dimenzované na výkon 0,6 W.

Co se týče digitálních vstupů, použijeme pro univerzálnost přípravku nastavování vysoké/nízké úrovně tlačítkem bez aretace nebo nastavení trvalé úrovně pomocí páčkového přepínače.

Analogové výstupy jsou zde realizovány jednoduše a sice jednoduchými stejnosměrnými voltmetry se stupnicí od 0 V do 15 V. Z dokumentace k námi použitému modulu analogových vstupů víme, že můžeme použít různé napěťové či proudové rozsahy vyjadřující analogovou úroveň. [8] V naší práci pracujeme výhradně s napěťovým rozsahem od -10 V do +10 V.

K regulaci úrovně pro analogové vstupy použijeme potenciometry, zde konkrétně lineární s hodnotou 10 kΩ. Modul analogových vstupů našeho PLC je dimenzován až na 28,8 V, nicméně stále pracujeme v rozsahu maximálně do +10 V. [8] Z toho důvodu zařadíme před potenciometr ještě předřadný odpor, který nám maximální napětí ze zdroje omezí právě na 10 V. Jelikož se pohybujeme řádově v desítkách kΩ, musíme do výpočtu předřadného odporu započítat i vnitřní odpor našeho voltmetru, který jsme naměřili ohmmetrem - 15 kΩ.



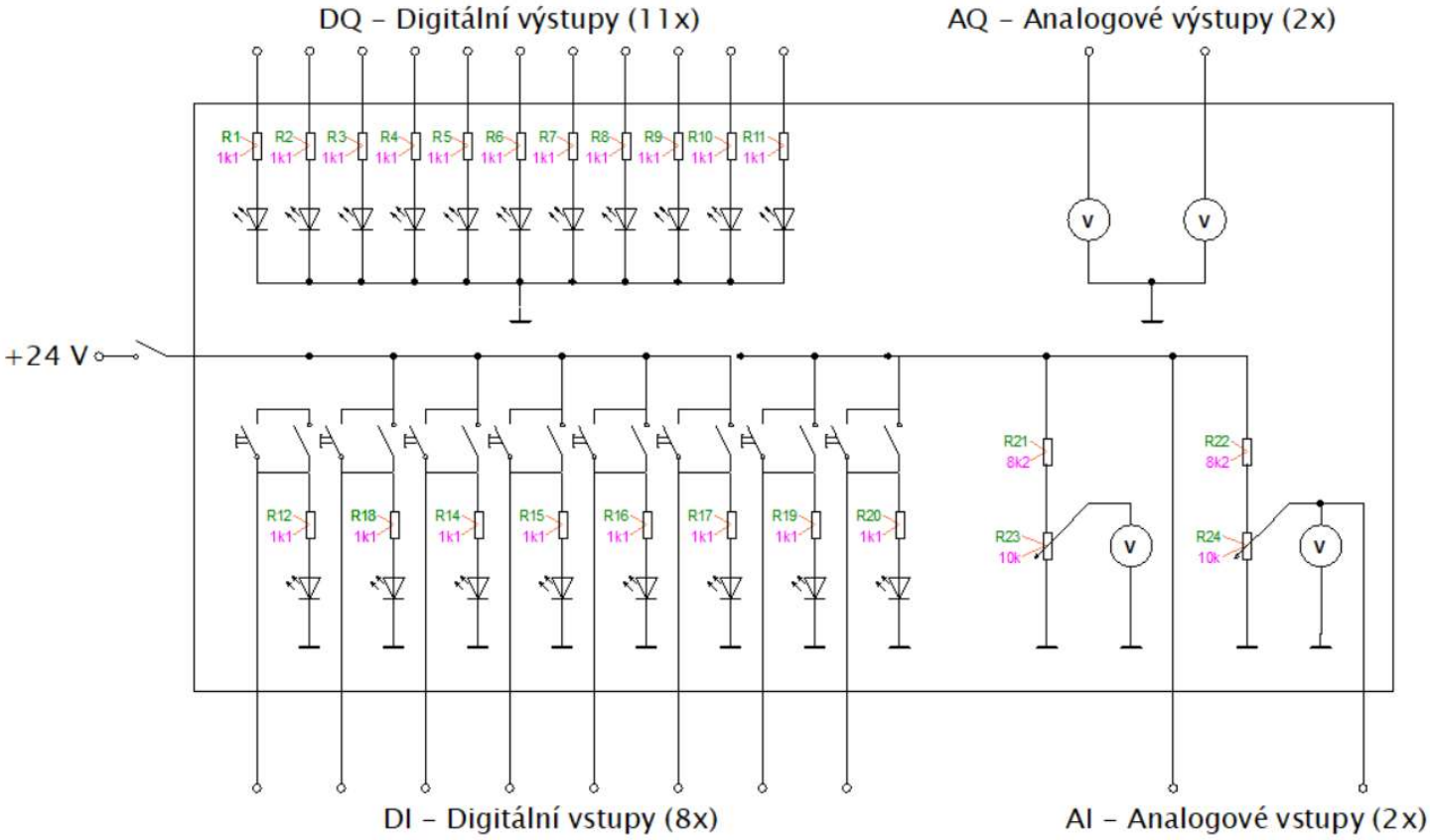
Obrázek 11: Schéma zapojení pro výpočet předřadného odporu pro potenciometr na analogový vstup (AI)

$$R_{pvolt} = \frac{U_{zdroje} \cdot \frac{R_{pot} \cdot R_{volt}}{R_{pot} + R_{volt}}}{U_{volt}} - \frac{R_{pot} \cdot R_{volt}}{R_{pot} + R_{volt}} = \tag{3}$$

$$= \frac{24 \cdot \frac{(10 \cdot 15) \cdot 10^3}{(10 + 15) \cdot 10^3}}{10} - \frac{24 \cdot \frac{(10 \cdot 15) \cdot 10^3}{(10 + 15) \cdot 10^3}}{10} = 8400\ \Omega$$

Opět zvolíme hodnotu podle řady, tentokrát podle E12 volíme hodnotu 8200 Ω.

Pro větší bezpečnost přidáme přípravku ještě kolébkový vypínač, kterým lze v případě nutnosti odpojit všechny vstupy od napájení najednou. Výsledné schéma přípravku je tedy následující:

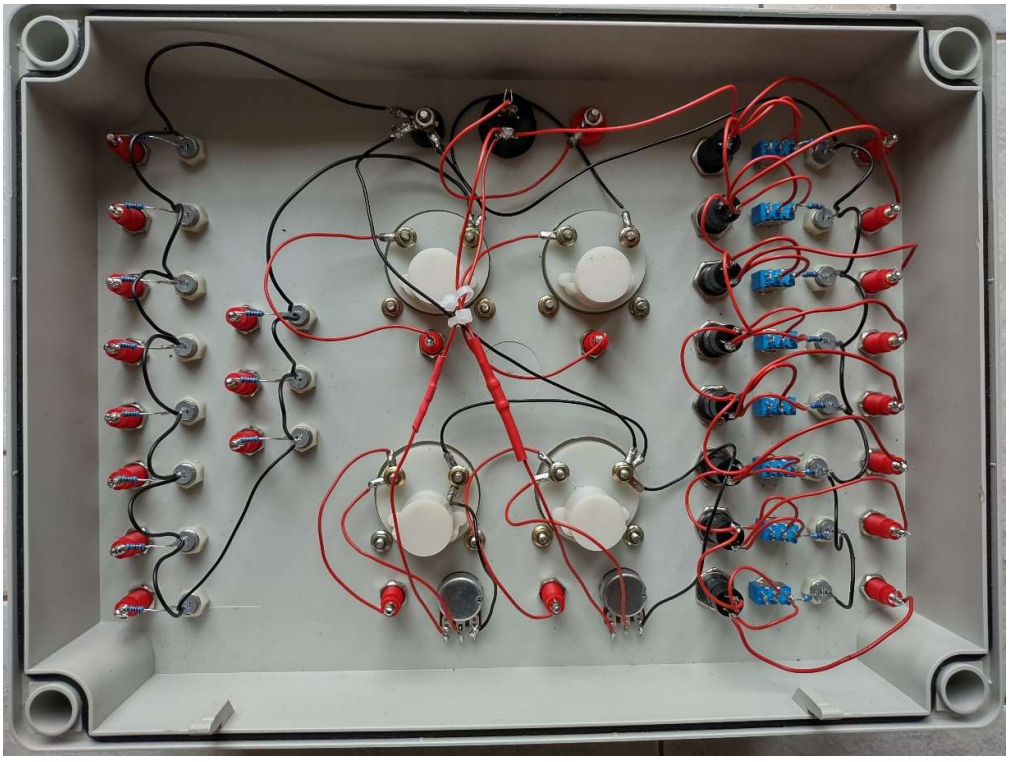


Obrázek 12: Kompletní schéma zapojení přípravku

4.2 Praktická realizace přípravku

Celý přípravek je realizován v boxu o rozměrech 300x220x120 mm s krytím IP65.

Obrázek 13: Fotografie realizovaného přípravku



Obrázek 14: Vnitřní zapojení přípravku

5 Vzorové úlohy

Dalším cílem této práce je návrh a praktická realizace vzorových úloh. Tyto úlohy jsou běžnými příklady použití PLC v praxi.

5.1 Úloha 1 – Dopravně řízená světelná křižovatka

5.1.1 Zadání

Realizujte řízení semaforů na tzv. dopravně řízené světelné křižovatce. Pro jednoduchost uvažujte pouze jednu hlavní a jednu vedlejší silnici. Stav, v jakém jsou semafory, pokud se před semaforem na vedlejší silnici nestojí žádný automobil je popsán v Tabulce 2.

Tabulka 2: Úloha 1: Tabulka se stavy semaforů, dokud u semaforu na vedlejší silnici nečeká žádný automobil

Čas [s]	Semafor pro automobily – hlavní silnice	Semafor pro automobily – vedlejší silnice	Semafor pro přechod – hlavní silnice	Semafor pro přechod – vedlejší silnice
0 až 3	Červená	Červená	Červená	Červená
3 až 4,5	Červená + Oranžová	Červená	Červená	Červená
4,5 až do stisku tlačítka	Zelená	Červená	Červená	Zelená

Pokud se před semaforem na vedlejší silnici objeví automobil (impulz v podobě tlačítka), nastává situace popsaná v Tabulce 3.

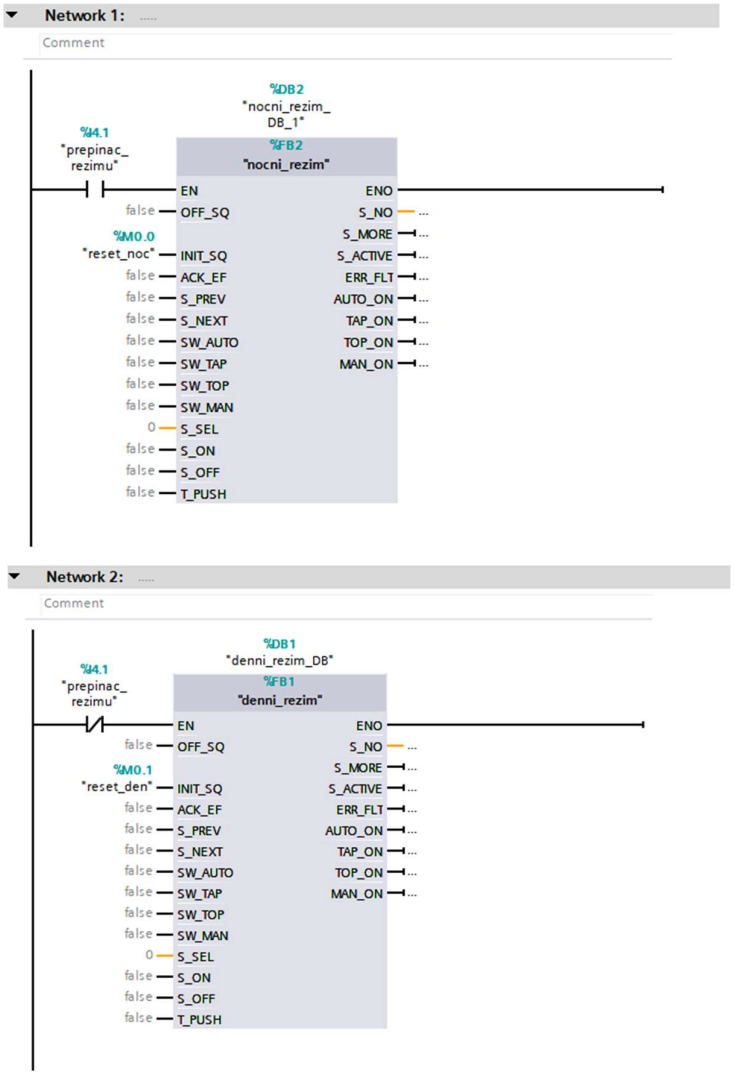
Tabulka 3: Úloha 1: Tabulka se stavy semaforů, pokud se u semaforu na vedlejší silnici objeví automobil

Čas [s]	Semafor pro automobily – hlavní silnice	Semafor pro automobily – vedlejší silnice	Semafor pro přechod – hlavní silnice	Semafor pro přechod – vedlejší silnice
0 až 5	Zelená	Červená	Červená	Zelená
5 až 6,5	Oranžová	Červená	Červená	Červená
6,5 až 9,5	Červená	Červená	Červená	Červená
9,5 až 11	Červená	Červená + Oranžová	Červená	Červená
11 až 21	Červená	Zelená	Zelená	Červená
21 až 22,5	Červená	Oranžová	Červená	Červená
22,5 až 25,5	Nastává stejný stav jako v Tabulce 2 pro čas 0 až 3 s			

Realizujte zároveň noční režim semaforů. Po přepnutí do nočního režimu se všude rozsvítí na tři sekundy červená. Po uplynutí těchto tří sekund začne na semaforech pro automobily blikat oranžová a semafory pro přechody zhasnou úplně. Po zapnutí denního režimu se opět všude rozsvítí červená a nastává stav podle Tabulce 2.

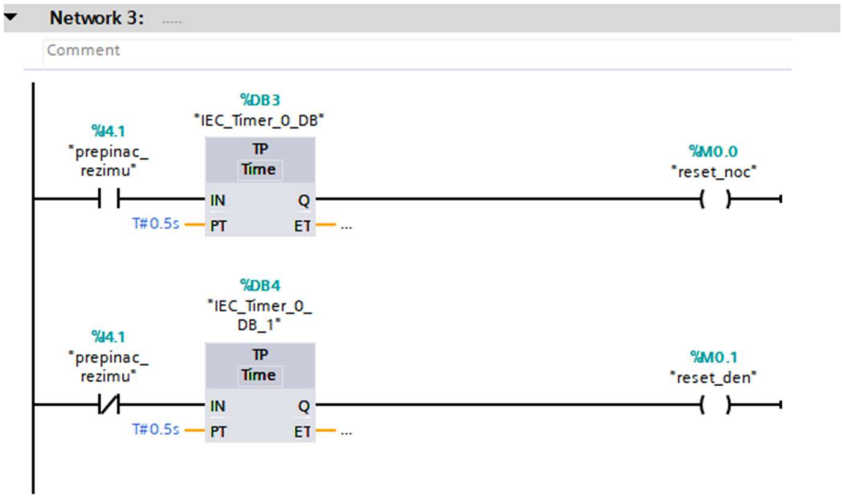
5.1.2 Řešení

Samotný program dopravního řízení křižovatky je z velké části realizován jazykem GRAPH. Tento jazyk je určen specificky pro sekvenční úlohy, nicméně pro začátek je asi nejsnáze pochopitelný. Úlohy podobného typu jsou typickým příkladem použití jazyka GRAPH.



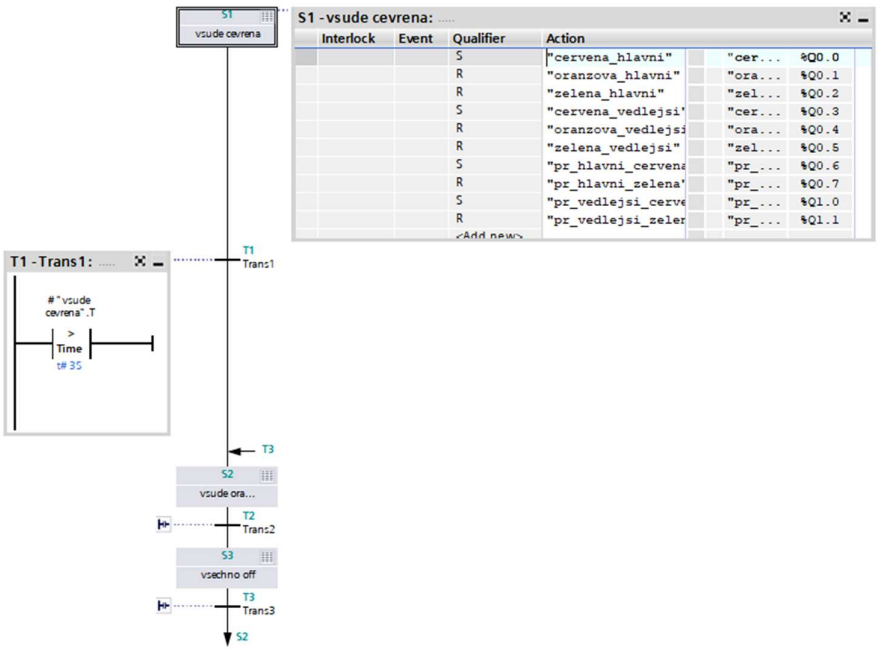
Obrázek 15: Úloha 1: Části Network 1 (nahoře) a Network 2 (dole) hlavní části programu (main)

Hlavní část programu (Obrázek 15) napsaná v jazyce LAD je jednoduchá. Zapínací kontakt „prepinac_rezimu“ zapne při stavu „true“ funkční blok „nocni_rezim“. Naopak ve stavu „false“ zapne funkční blok „denni_rezim“. Zde je nutno funkční bloky resetovat (resp. spouštět je od počátečního kroku) nastavením parametru „INIT_SQ“ na hodnotu „true“ při přepnutí mezi režimy. Pokud by zůstal celou dobu na úrovni „false“, běh programu ve funkčním bloku by nezačal od začátku, ale od stavu, ve kterém zůstal naposledy. To by v praxi znamenalo, že při přepnutí z nočního režimu na denní by se některé semafory rozsvítily ihned zeleně, což by mohlo vést k nebezpečné situaci.



Obrázek 16: Úloha 1: Část Network 3 hlavní části programu (main)

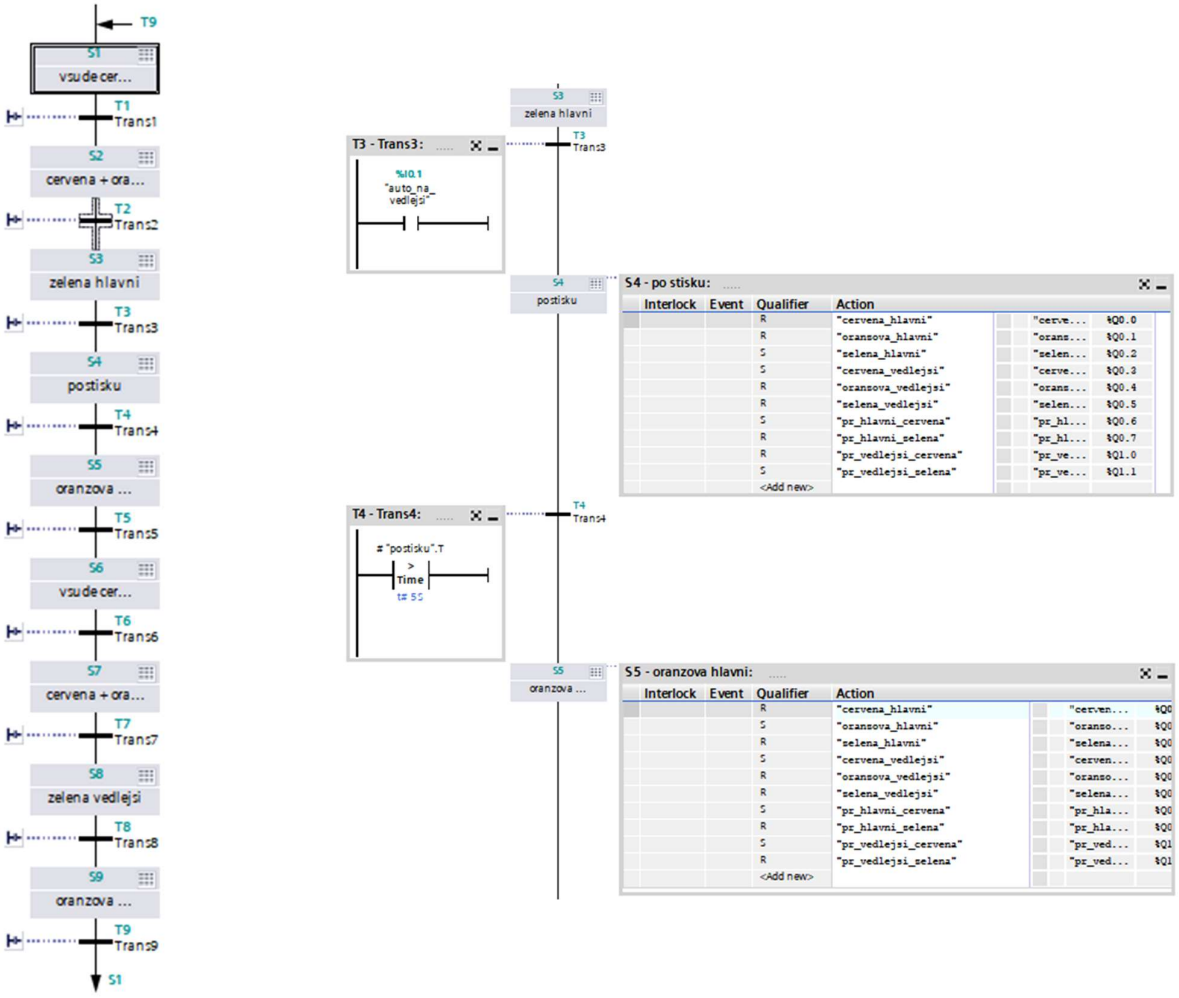
Část Network 3 (Obrázek 16) se stará o řízení resetů jednotlivých funkčních bloků. Pokud je zapnut noční režim (proměnná „prepinac_rezimu“ má hodnotu „true“), je na dobu 0,5 sekundy vygenerován impulz „true“ pro proměnnou „reset_noc“, který zajistí, že funkční blok „nocni_rezim“ se spustí od počátečního kroku, kdy na všech semaforech svítí červená. Obdobně je pak řízen reset bloku „denni_rezim“, kde pro vygenerování resetového impulzu stačí přepnout přepínač režimů na hodnotu „false“.



Obrázek 17: Úloha 1: Vnitřní struktura funkčního bloku "nocni_rezim"

Nyní si vysvětlíme chod funkčního bloku „nocni_rezim“ (Obrázek 17). V jazyce GRAPH je každý program rozdělen do kroků (v tomto konkrétním případě S1, S2, S3). Přechod do následujícího kroku je podmíněn podmínkou mezi nimi (zde „Trans1“, „Trans2“ a „Trans3“). V prvním kroku S1 jsou nastaveny proměnné „cervena_hlavni“, „cervena_vedlejsi“, „pr_hlavni_cervena“ a „pr_vedlejsi_cervena“ na hodnotu „true“, všechny ostatní jsou „false“. To znamená, že na všech semaforech svítí pouze červená. Po splnění podmínky „Trans1“, v tomto případě uplynutí požadovaného času 3 sekundy (resp. po uplynutí 3 sekund trvání kroku S1), se přejde do kroku S2.

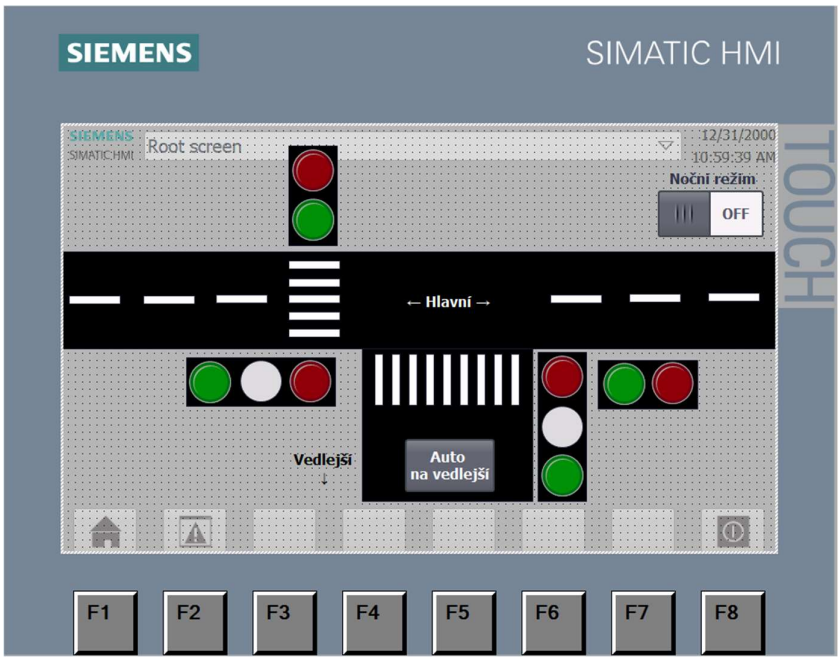
V tomto kroku jsou všechny proměnné nastaveny na hodnotu „false“, pouze hodnoty „oranzova_hlavni“ a „oranzova_vedlejsi“ na úroveň true. Následuje podmínka „Trans2“, která říká, že po uplynutí 1 sekundy trvání kroku S2 má program přejít dál do kroku S3. V něm jsou veškeré proměnné nastaveny na hodnotu „false“ a všechny semafory jsou zhasnuty. Jelikož ale chceme, aby na semaforech blikala oranžová, musíme po splnění podmínky „Trans3“ nastavit skok zpět do kroku S2. Program tedy nikdy neskončí a na semaforech cyklicky bliká oranžová, dokud nepřepneme přepínač „prepinac_rezimu“.



Obrázek 18: Úloha 1: Vnitřní struktura funkčního bloku "denni_rezim" (vlevo) a detail na kroky S3, S4 a S5 (vpravo)

Funkční blok „denni_rezim“ (Obrázek 18) je složitější, avšak princip je naprosto stejný, jako u bloku „nocni_rezim“. V kroku S1 je na všech semaforech nastavena červená (tzn. hodnoty „true“ jsou pouze na proměnných „cervena_hlavni“, „cervena_vedlejsi“, „pr_hlavni_cervena“ a „pr_vedlejsi_cervena“). Po uplynutí tří sekund (podmínka „Trans1“) následuje krok S2, kde je na hodnotu „true“ nastavena ještě navíc proměnná „oranzova_hlavni“. Tento krok trvá 1,5 sekundy a po něm následuje krok S3, v němž svítí zelená na semaforu pro hlavní silnici a na semaforu pro přechod na vedlejší silnici, zatímco na všech ostatních semaforech svítí červená. V tomto stavu naše křižovatka setrvává do chvíle, kdy se před semaforem na vedlejší silnici objeví automobil. V praxi se přítomnost auta čekajícího na semaforu detekuje například spínačem zabudovaným přímo v silnici.

Ten je v této úloze realizován spínačem „auto_na_vedlejsi“ v podmínce „Trans3“. Po splnění této podmínky (tzn. spínač je sepnut) následuje krok S4, který je totožný s krokem S3, nicméně je podmínkou „Trans4“ omezen na dobu trvání pěti sekund. V kroku S5 svítí na semaforu na hlavní silnici oranžová a na semaforu pro přechod na vedlejší silnici svítí červená. V kroku S6 svítí červená na všech semaforech. Po třech sekundách (podmínka „Trans6“) se rozsvěcuje spolu s červenou na semaforu na vedlejší silnici i oranžová (krok S7). V kroku S8 svítí na tomto semaforu zelená a zelená svítí i na semaforu na přechodu na hlavní silnici. Tato zelená však nečeká na impulz ze spínače na hlavní silnici, ale trvá pouze 10 sekund (podmínka „Trans8“). Stejně jako u funkčního bloku „nocni_rezim“ je i blok „denni_rezim“ zakončen skokem, zde však do kroku S1 (tzn. červená na všech semaforech).



Obrázek 19: Úloha 1: Uživatelské rozhraní na HMI Panelu

5.2 Úloha 2 – Řízení světelného železničního přejezdového zabezpečovacího zařízení

5.2.1 Zadání

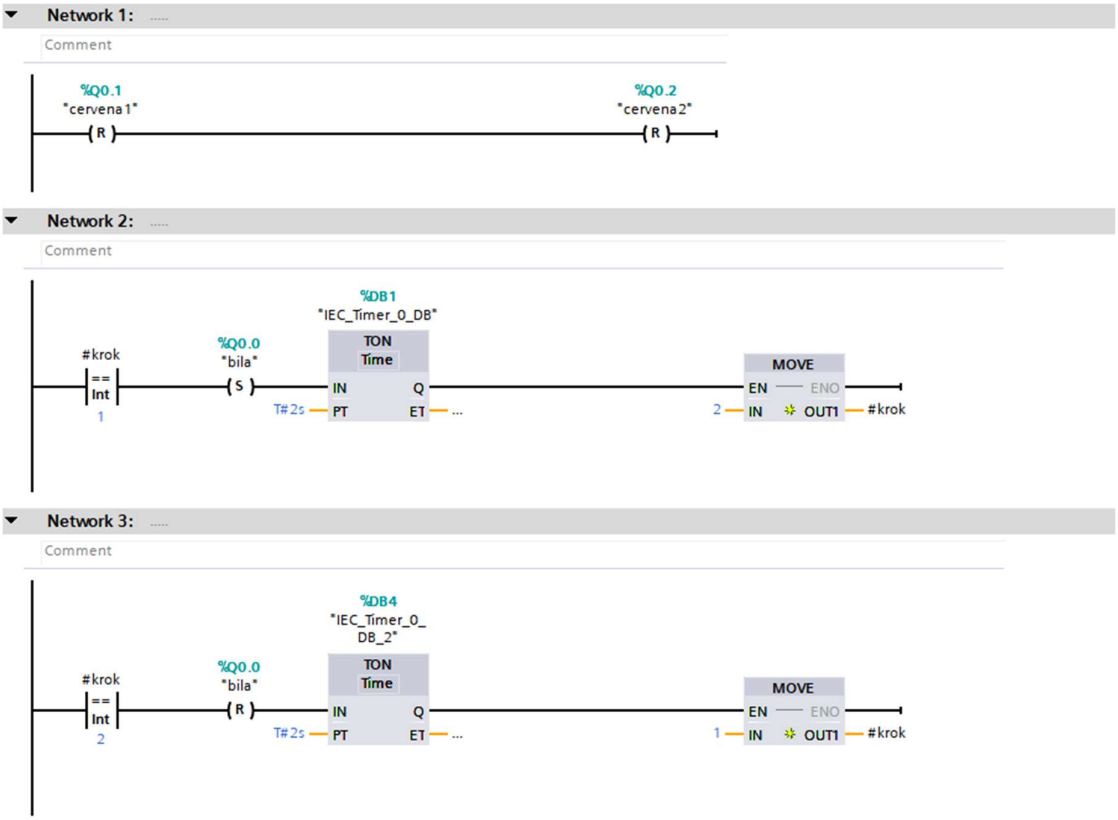
Realizujte řízení světelného železničního přejezdového zabezpečovacího zařízení v klidovém režimu (kdy neprojíždí vlak) a v režimu, kdy vlak projíždí. V klidovém režimu bliká jedno bílé světlo. Ve chvíli, kdy projíždí vlak, dojde před přejezdem k počítání počtu

náprav vlaku a začnou střídavě blikat dvě červená světla (počítání náprav realizujte počítáním stisků tlačítka). Takto je nastaven přejezd, kdy jím projíždí vlak, a to do chvíle, než se na druhé straně přejezdu napočítá stejný počet náprav, jako na straně první. Poté přestanou blikat červená světla a začne blikat opět bílé světlo jako v klidovém režimu. Zároveň dojde k vynulování počtu náprav na obou stranách přejezdu. Takto realizujte signalizaci z obou směrů přejezdu.

Ošetřete zároveň chybový stav, kdy je na konci přejezdu po projetí vlaku napočítán větší počet náprav než na začátku přejezdu. Chybový stav signalizujte například příslušným světlem. Z tohoto stavu se do klidového stavu lze dostat pouze stisknutím tlačítka resetu.

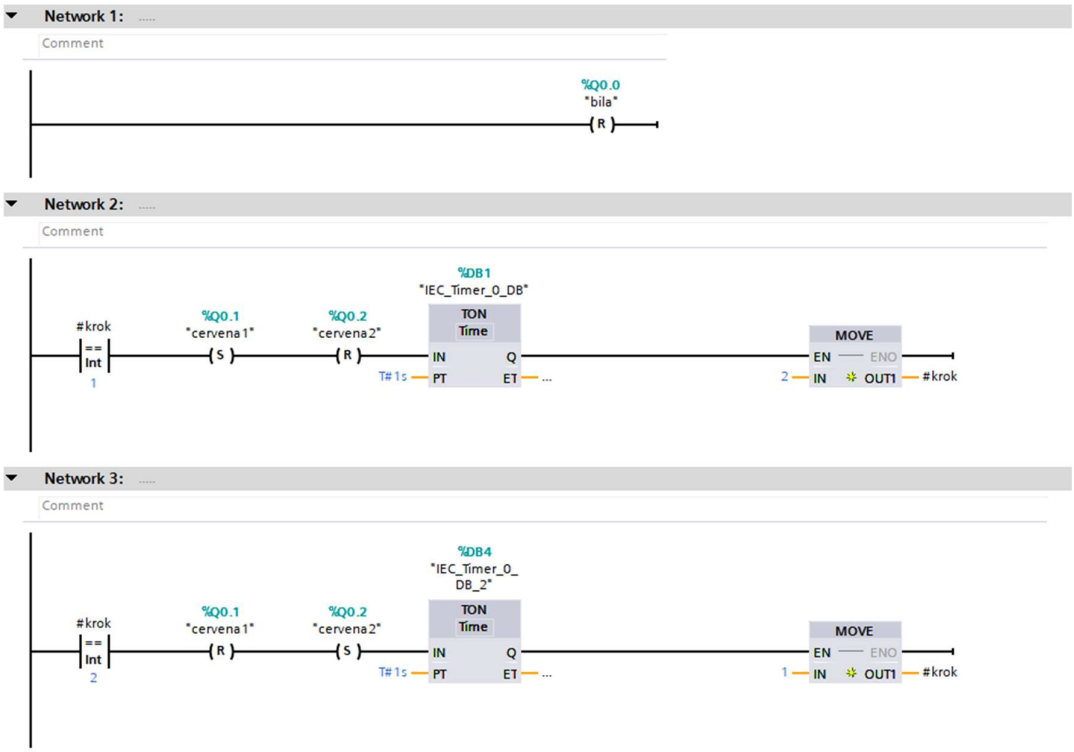
5.2.2 Řešení

Úloha je zde řešena v jazyce LAD, což je v podstatě základní jazyk pro programování PLC značky Siemens.



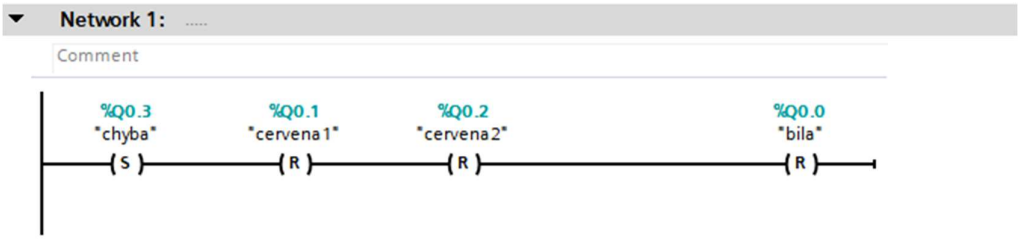
Obrázek 20: Úloha 2: Vnitřní struktura funkčního bloku "klidovy_stav"

Nejprve si popišme funkční blok „klidovy_stav“ (Obrázek 20), ve kterém jde v podstatě jen o to, rozblikat bílé světlo na světelném zařízení. Jelikož v tomto stavu pouze cyklicky bliká bílé světlo, nabízí se zde použít jazyk GRAPH. Sekvenční úlohu lze ale realizovat i jazykem LAD (případně i dalšími jazyky) a zde si ukážeme, jakým způsobem. Program je zde pro přehlednost rozdělen do tří částí (Networks). V první části Network 1 jsou vyresetovány proměnné „cervena1“ a „cervena2“ (tzn. je na nich nastavena hodnota „false“ – tato světla nesvítí). V části Network 2 je nejprve lokální proměnná „#krok“ porovnána s hodnotou 1. Pokud je tato podmínka splněna, je výstupní proměnná „bila“ nastavena na hodnotu „true“ a bílé světlo svítí. Tento stav je časovačem TON nastaven na dobu dvou sekund a následně je lokální proměnné „#krok“ přiřazena hodnota 2 (blok MOVE zde plní funkci přiřazení hodnoty k dané proměnné). Část Network 3 je téměř shodná s Network 2, pouze je zde hodnota proměnné „bila“ nastavena na „false“. Tento stav opět trvá dvě sekundy a lokální proměnné „#krok“ je poté opět přiřazena hodnota 1. Na venek se tedy funkční blok chová tak, že zde cyklicky bliká bílé světlo.



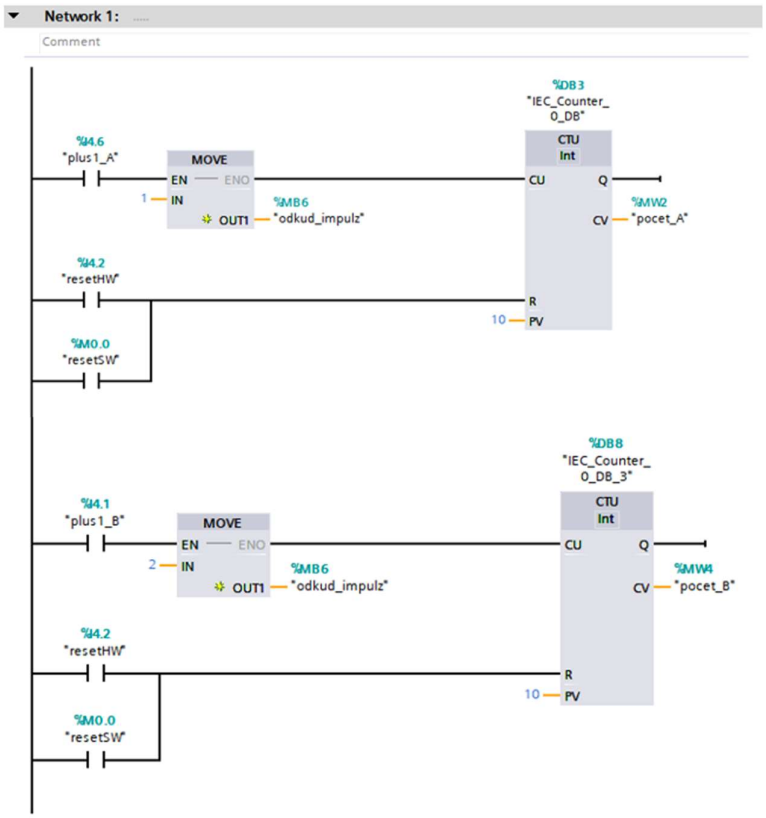
Obrázek 21: Úloha 2: Vnitřní struktura funkčního bloku "jede_vlak"

Obdobným způsobem je realizován funkční blok „jede_vlak“ (Obrázek 21), kde střídavě blikají dvě červená světla. V části Network 1 je nastavena hodnota „false“ na proměnnou „bila“, v Network 2 je proměnná „cervena1“ nastavena na „true“, „cervena2“ na „false“ a v Network 3 naopak.



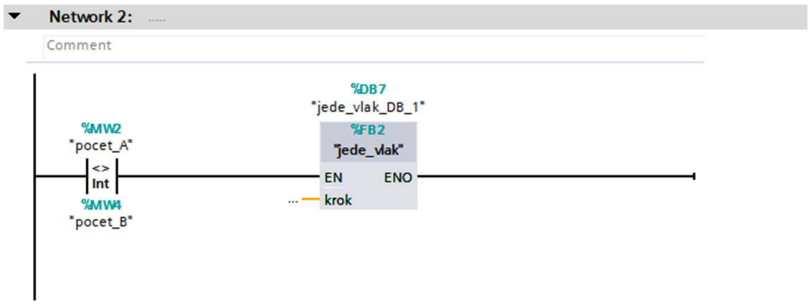
Obrázek 22: Úloha 2: Vnitřní struktura funkčního bloku "chybovy_stav"

Funkční blok „chybovy_stav“ (Obrázek 22) je jednoduchý a jeho funkce je pouze v rozsvícení chybového světla a zhasnutí všech ostatních.



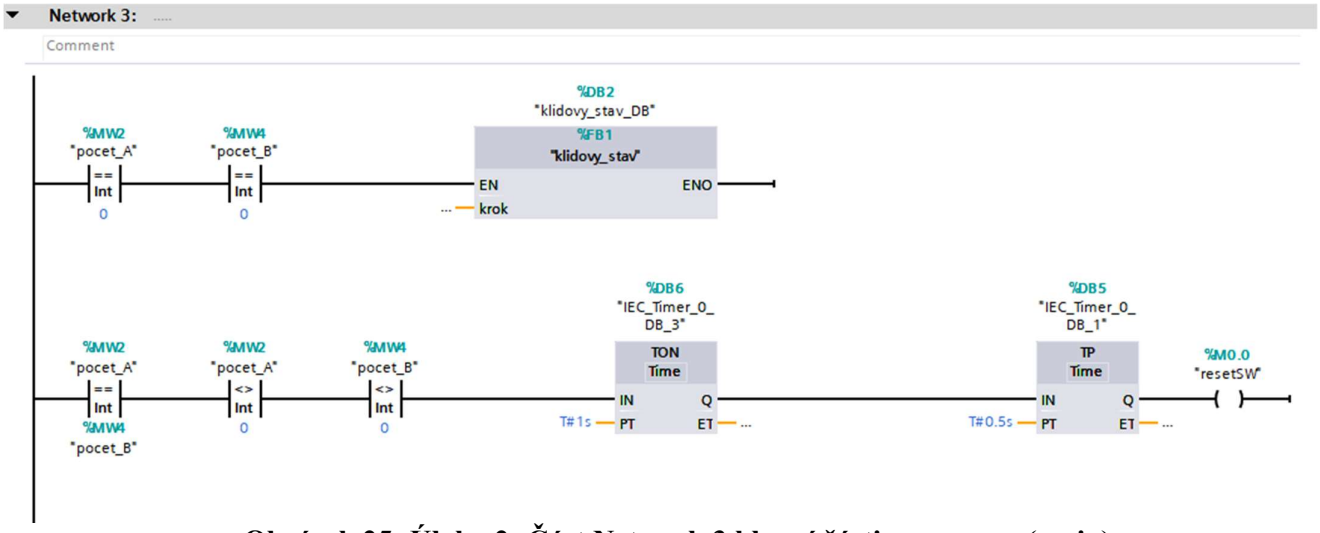
Obrázek 23: Úloha 2: Část Network 1 hlavní části programu (main)

V hlavní části programu je realizováno vlastní řízení celého světelného zařízení. V části Network 1 (Obrázek 23) je dvěma čítači počítán počet náprav na jednotlivých stranách přejezdu. Čítače zde počítají impulzy ze spínacích kontaktů „plus1_A“ resp. „plus1_B“ a počty impulzů ukládají do proměnné „pocet_A“ resp. „pocet_B“. Čítače jsou resetovány na hodnotu 0 buďto fyzickým tlačítkem na přípravku (proměnná „resetHW“), nebo jinou částí programu (proměnná „resetSW“). Zároveň s každým impulzem dojde k přiřazení konkrétní hodnoty proměnné „odkud_impulz“. Tu využijeme při detekci chybového stavu, detailně si ji popíšeme v dalším textu. Maximální hodnoty na čítačích jsou zde nastaveny na hodnotu 10, nicméně lze použít jakoukoliv hodnotu v rozsahu datového typu Int.



Obrázek 24: Úloha 2: Část Network 2 hlavní části programu (main)

Část Network 2 (Obrázek 24) spouští funkční blok „jede_vlak“ a to tehdy, je-li splněna podmínka, že hodnoty proměnných „pocet_A“ a „pocet_B“ jsou rozdílné (tedy jsou rozdílné počty náprav před a za přejezdem).

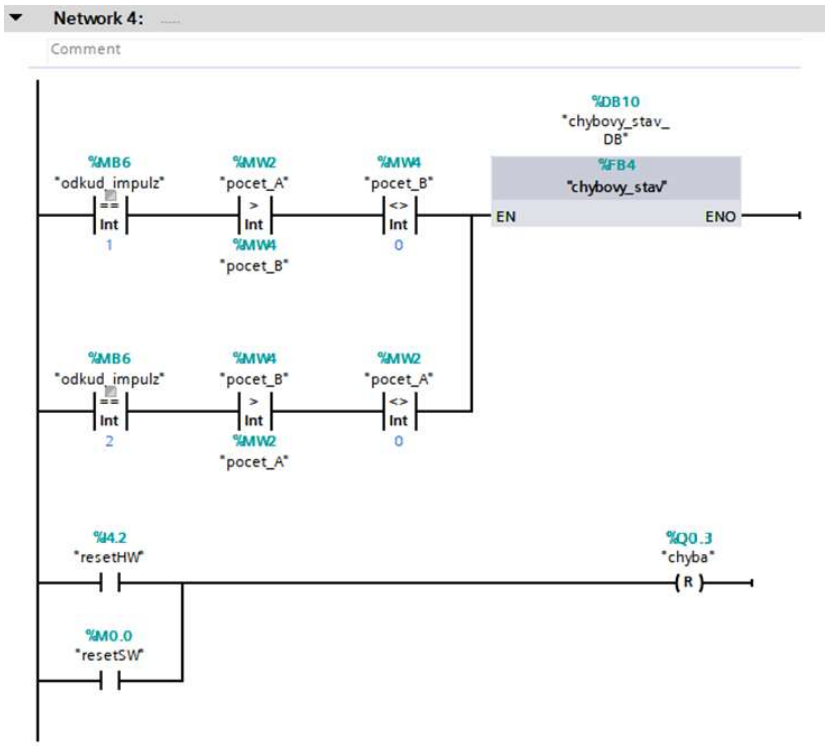


Obrázek 25: Úloha 2: Část Network 3 hlavní části programu (main)

V části Network 3 (Obrázek 25) je řízen chod funkčního bloku „klidovy_stav“, který je spuštěn, je-li splněna podmínka, že hodnota proměnné „pocet_A“ je rovna nule a zároveň je rovna nule hodnota proměnné „pocet_B“. Druhá větev této části řídí reset čítačů z části Network 1 a tím návrat do klidového stavu poté, co přejezdem projede vlak. Pokud se rovnají hodnoty proměnných „pocet_A“ a „pocet_B“ a zároveň je každá z těchto hodnot různá od nuly, je po uplynutí jedné sekundy (toto zpoždění zajišťuje časovač TON) časovačem TP na dobu 0,5 sekundy nastavena hodnota „true“ proměnné „resetSW“, čímž dojde k vynulování čítačů a je opět zajištěn klidový stav. Takto realizovaný přejezd je průjezdný v obou směrech (tzn. z bodu A do bodu B a naopak).

V případě světelného železničního signalizačního zařízení je nutno detekovat i chybový stav, kdy je po průjezdu vlaku přejezdem na konci přejezdu napočítán větší počet náprav než na začátku. Jelikož máme ale přejezd průjezdný z obou směrů, je nutné detekovat, z jakého směru vlak jede. Pro pochopení se vraťme k části Network 1 (Obrázek 23). V případě, že vlak jede z bodu A do bodu B, načítají se na straně A impulzy pomocí spínacího kontaktu „plus1_A“. Tím se zároveň načte hodnota 1 do proměnné „odkud_impulz“. Spustí se funkční blok „jede_vlak“, přejezdem projíždí vlak, pomocí kontaktu „plus1_B“ se začnou načítat impulzy do čítače na straně B a hodnota proměnné „odkud_impulz“ je přepsána na hodnotu 2. Pokud by vlak projížděl z bodu A do bodu B, měla by po projetí vlaku přejezdem proměnná „odkud_impulz“ hodnotu 1. Tímto jednoduchým způsobem jsme schopni rozlišit, z jakého směru přejezdem projíždí vlak a

detekovat chybový stav se zachováním obousměrné průjezdnosti přejezdu. Časovač TON v části Network 3 je pro detekci chybového stavu nezbytný. Pokud by zde nebyl, došlo by k vynulování čítačů ihned ve chvíli, kdy se počty náprav na obou stranách přejezdu rovnají a pokud by přišel další impulz, nebyl by detekován jako chyba, ale pouze jako další náprava například dalšího vlaku.



Obrázek 26: Úloha 2: Část Network 4 hlavní části programu (main)

K chybovém stavu dojde ve chvíli, kdy je po průjezdu vlaku přejezdem napočítán větší počet impulzů než před průjezdem. Na základě toho je sestavena část Network 4 (Obrázek 26). Chybový stav zde nastane při splnění dvou podmínek:

1. Hodnota proměnné „odkud_impulz“ je rovna 1, zároveň hodnota proměnné „pocet_A“ je větší, než hodnota „pocet_B“ a zároveň hodnota „pocet_B“ je nenulová.

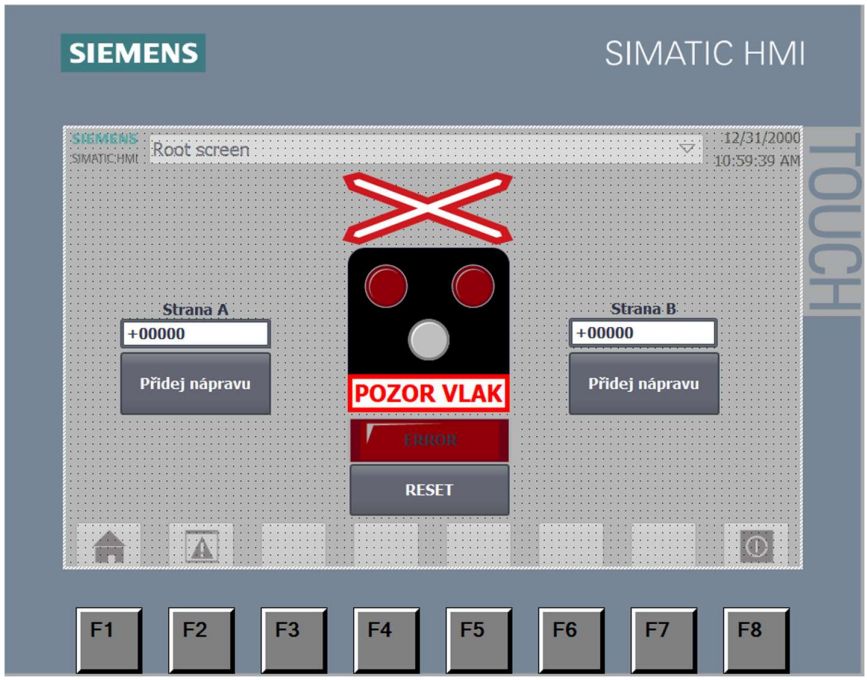
NEBO

2. Hodnota proměnné „odkud_impulz“ je rovna 2, zároveň hodnota proměnné „pocet_B“ je větší, než hodnota „pocet_A“ a zároveň hodnota „pocet_A“ je nenulová.

Pokud dojde k resetu ať už fyzickým stiskem tlačítka na přípravku, nebo k resetu

vyvolanému programem, je nastavena proměnná „chyba“ na hodnotu „false“.

Povšimněme si, že v jazyku LAD je logický součin několika podmínek realizován sériovým zapojením, zatímco logický součet je realizován paralelním zapojením.



Obrázek 27: Úloha 2: Uživatelské rozhraní na HMI Panelu

6 Úloha 3 – Řízení dopravníku výrobků s váhou

6.1.1 Zadání

Realizujte řízení dopravníku výrobků v továrně s váhou na konci. Dopravník po spuštění dopravuje výrobky po jednom a na konci je každý výrobek zvážen (hmotnost výrobku je analogový vstup do programu). Chod dopravníku je značen příslušnou diodou (spuštěn je vždy, pokud není na váze výrobek, nebo není sepnut stop vypínač). Pokud hmotnost výrobku odpovídá hmotnosti zadané (± tolerance), rozsvítí se příslušná signalizace a obsluha může výrobek umístit do přepravky. Pokud je naopak hmotnost mimo toleranci, rozsvítí se opět příslušná signalizace. Po odebrání výrobku z dopravníku a jeho umístění do přepravky (nebo zahození mezi zmetky) stiskne obsluha tlačítko, které opět uvede dopravník do chodu.

Pro univerzálnost celého systému přidejte ještě jeden analogový vstup, který bude

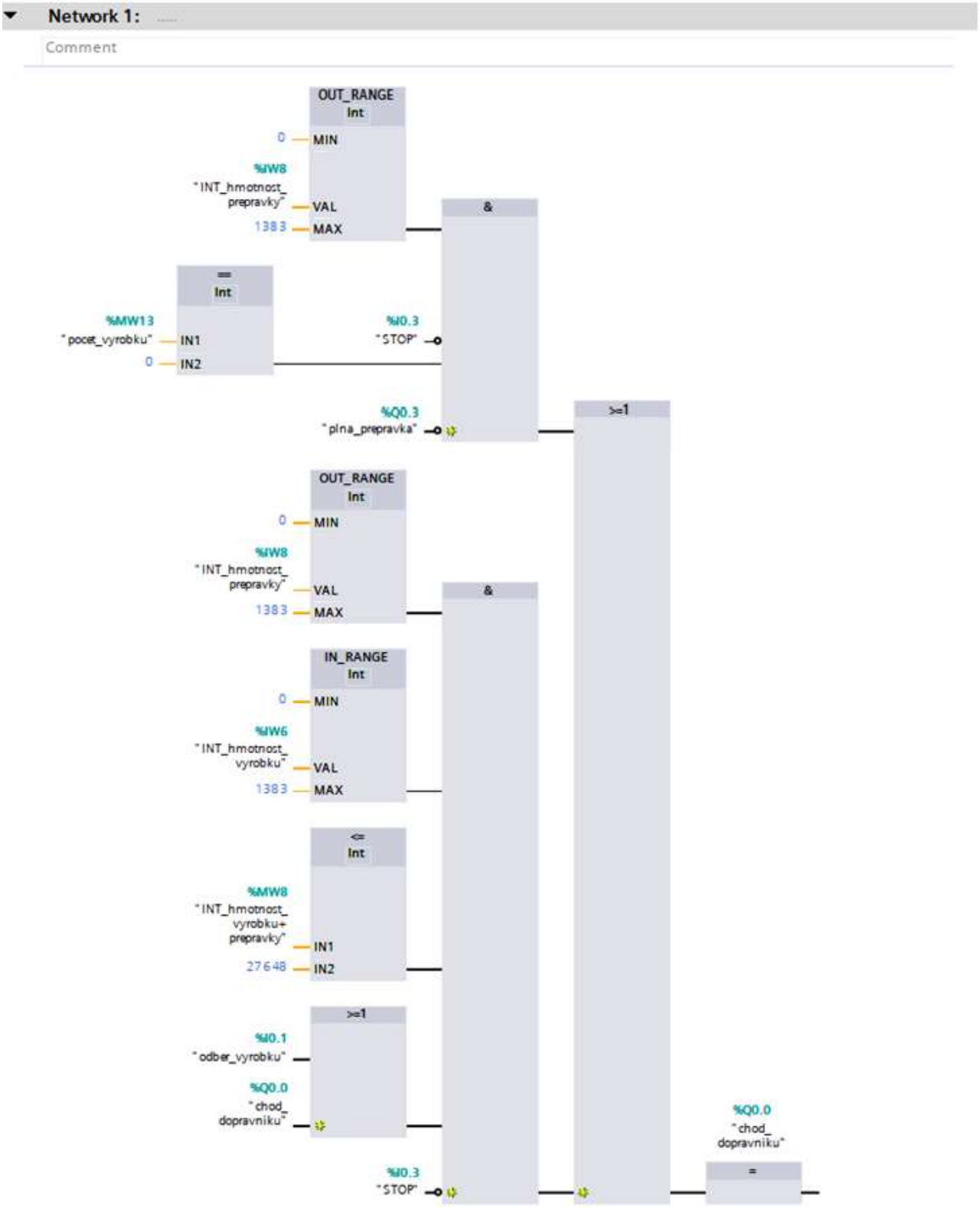
symbolizovat hmotnost přepravky za dopravníkem. Dosáhne-li hmotnost přepravky s výrobky hmotnosti 10 kg, rozsvítí se signalizace pro obsluhu, že má plnou přepravku odebrat a dát prázdnou. Pokud za dopravníkem přepravka není (resp. není zadána její hmotnost), není možné uvést dopravník do chodu.

Realizujte v tomto systému také servisní mód, kde půjde popojíždět dopravníkem při podržení tlačítka.

Poměr mezi napěťovým výstupem přípravku a danou hmotností udržujte 1:1 (tzn. 1 V na přípravku odpovídá hmotnosti 1 kg zobrazené na HMI Panelu). Normu hmotnosti výrobku pro tuto úlohu uvažujte mezi 1 kg a 2 kg.

6.1.2 Řešení

V této úloze se budeme zabývat prací s analogovými vstupy a výstupy našeho PLC. Úlohu budeme řešit v jazyce FBD, který je velmi podobný již vysvětlenému jazyku LAD. Ještě před začátkem vlastního řešení úlohy je nutné si uvědomit, jak s analogovými moduly pracovat. Z dokumentace k našim modulům můžeme vyčíst, že konkrétnímu napětí na analogovém vstupu (nebo výstupu) odpovídá konkrétní číselná hodnota typu Int. Například hodnotě 1 V na modulu analogového vstupu odpovídá dekadická hodnota 2765. Obdobně dekadické hodnotě 1 typu Int odpovídá napětí 362 μV. [8] Je zřejmé, že tuto hodnotu bude velmi obtížné přesně nastavit, ať už vlivem nedokonalosti potenciometrů na přípravku, příliš hrubě cejchované stupnici voltmetrů, či různých parazitních vlastností ať už na našem přípravku, nebo přímo na modulu PLC. Z toho důvodu je při návrhu programu prakticky nutné pracovat s širšími rozsahy hodnot proměnných, namísto porovnávání proměnných s konkrétními hodnotami typu Int



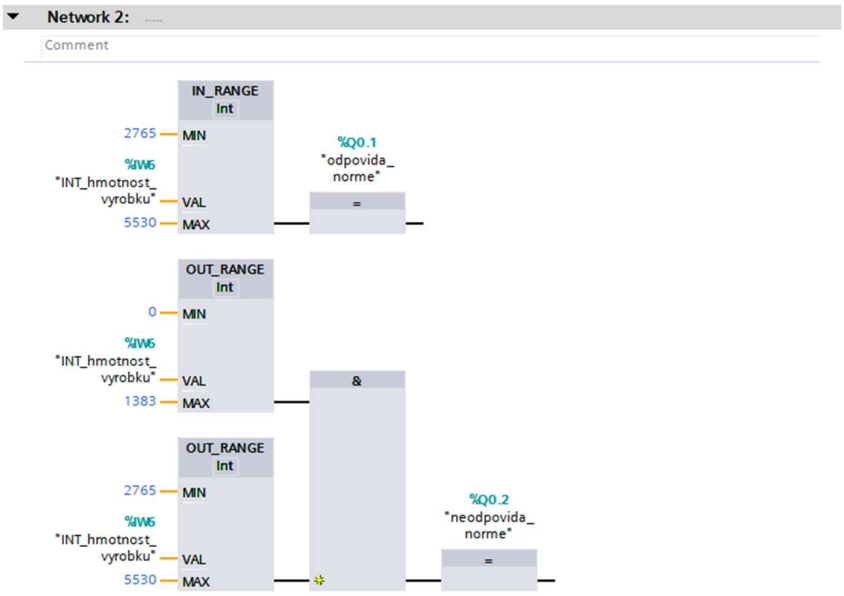
Obrázek 28: Úloha 3: Část Network 1 funkčního bloku "standardni_rezim"

Konkrétní příklad práce s širším rozsahem hodnot si můžeme uvést hned v části Network 1 funkčního bloku „standardni_rezim“ (Obrázek 28). Na první pohled můžeme vidět, že tato část se skládá ze dvou paralelních větví, které jsou vzájemně logicky sečteny. Pokud jsou tedy splněny následující podmínky, dopravník je v chodu (proměnná „chod_dopravniku“ je nastavena na hodnotu „true“):

1. Hodnota proměnné „INT_hmotnost_prepravky“ je mimo rozsah 0 až 1383 (což odpovídá napětí 0 V až 0,5 V – tedy 0 kg až 0,5 kg – viz. zadání úlohy) a zároveň není stisknut nouzový vypínač „STOP“, zároveň je počet výrobků roven nule a zároveň nesvítí signalizace plné přepravky (hodnota proměnné „plna_prepravka“ je „false“).

NEBO

2. Hodnota proměnné „INT_hmotnost_prepravky“ je mimo rozsah 0 až 1383 a zároveň je hodnota proměnné „INT_hmotnost_vyrobku“ v rozsahu 0 až 1383, zároveň je součet hmotnosti výrobku s hmotností přepravky menší, nebo roven hodnotě 27648 (což odpovídá napětí 10 V), došlo ke stisknutí tlačítka „odber_vyrobku“ (s přídržným kontaktem „chod_dopravniku“) a zároveň není stisknut vypínač „STOP“.



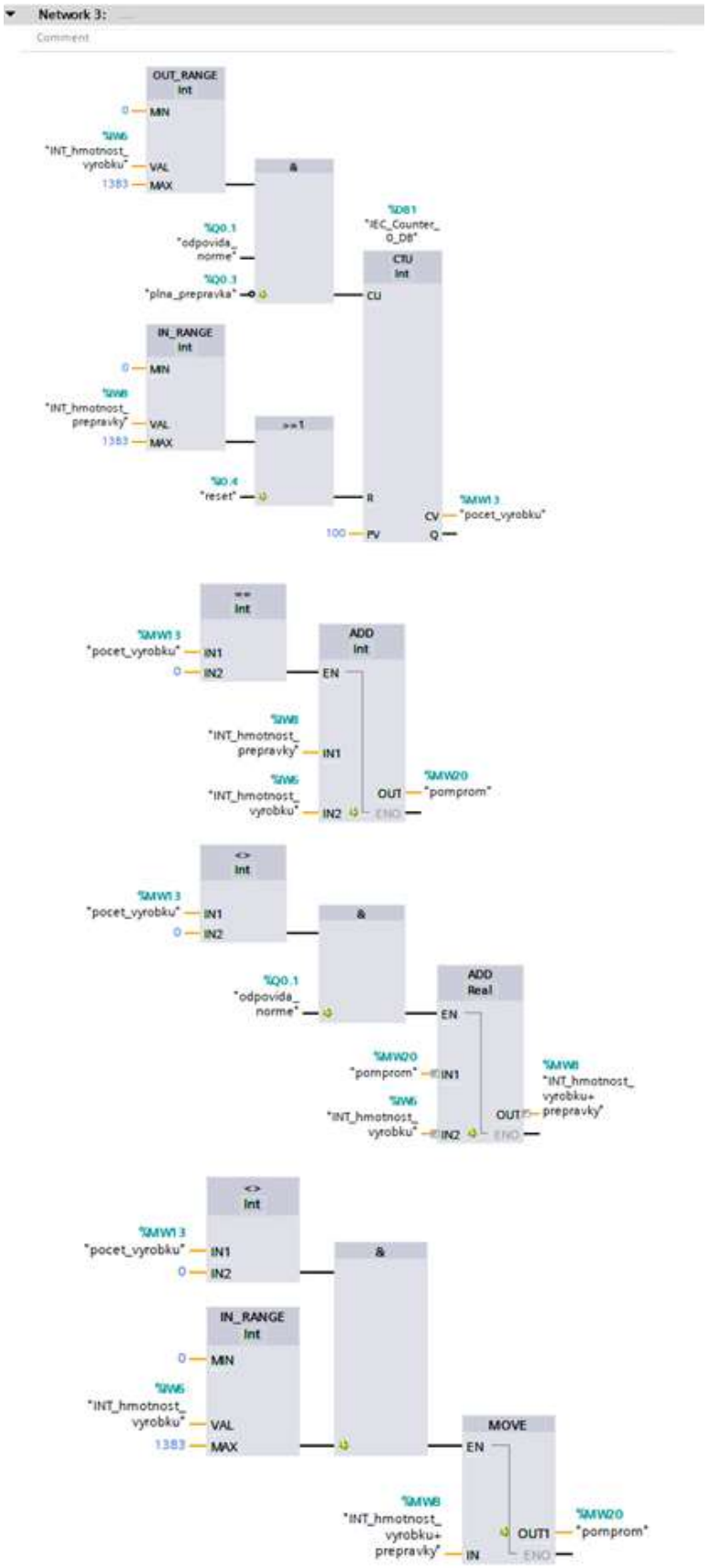
Obrázek 29: Úloha 3: Část Network 2 funkčního bloku "standardni_rezim"

Část Network 2 (Obrázek 29) se zabývá pouze rozhodováním, zda hmotnost výrobku odpovídá ormě, či nikoliv. Pokud je hodnota „INT_hmotnost_vyrobku“ v rozsahu 2765 až 5530 (což odpovídá napětí na vstupu 1 V až 2 V), pak svítí signalizace „odpovida_norme“. Pokud je však hodnota této proměnné mimo tento rozsah a zároveň je nenulová (tzn. mimo rozsah 0 až 1383), pak hmotnost výrobku normě neodpovídá a svítí signalizace „neodpovida_norme“.

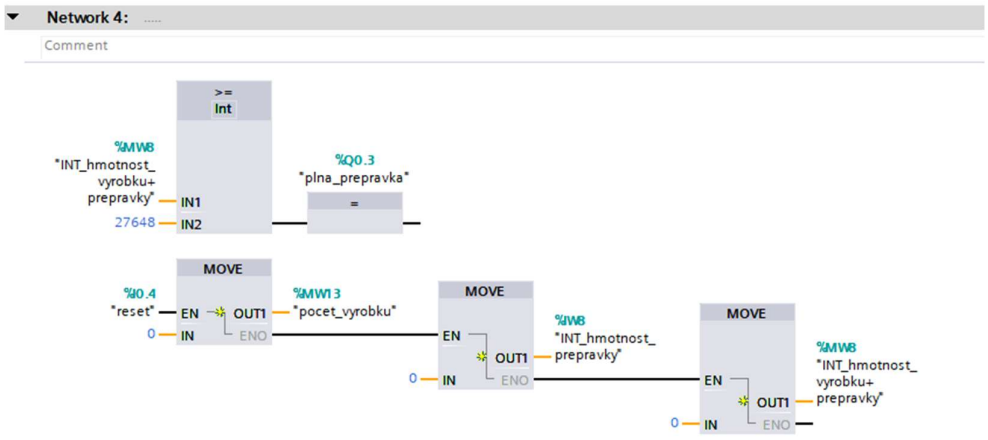
V části Network 3 (Obrázek 30) je prováděno počítání výrobků čítačem CTU a započítávání hmotnosti výrobku do celkové hmotnosti i s přepravkou. Čítač přičte hodnotu 1 do proměnné „pocet_vyrobku“, pokud je hodnota „INT_hmotnost_vyrobku“ nenulová (mimo rozsah 0 až 1383), zároveň hmotnost daného výrobku odpovídá normě a zároveň nesvítí signalizace, že je plná přepravka. Čítač je vynulován buďto, pokud je hmotnost přepravky nulová (obsluha odebrala plnou přepravku a dává novou), nebo pokud bylo stisknuto tlačítko „reset“.

Pokud je počet výrobků nulový, je proveden součet hmotnosti přepravky s hmotností výrobku a výsledek je uložen do pomocné proměnné „pomprom“. Pokud je počet výrobků naopak nenulový a zároveň hmotnost výrobku odpovídá normě, je proveden součet hodnoty proměnné „pomprom“ s hodnotou proměnné „INT_hmotnost_vyrobku“ a výsledek je uložen do proměnné „INT_hmotnost_vyrobku+prepravky“. Tento zdánlivě nelogický způsob součtu má své opodstatnění a sice z důvodu toho, aby byla hmotnost přepravky do celkové hmotnosti započítána pouze jednou.

Následuje poslední podmínka, která přiřadí proměnné „pomprom“ hodnotu proměnné „INT_hmotnost_vyrobku+prepravky“ právě tehdy, když je počet výrobků nulový a zároveň je hodnota proměnné „INT_hmotnost_vyrobku“ v rozsahu 0 až 1383 (tedy na vstupu je napětí v rozsahu 0 V až 0,5 V).

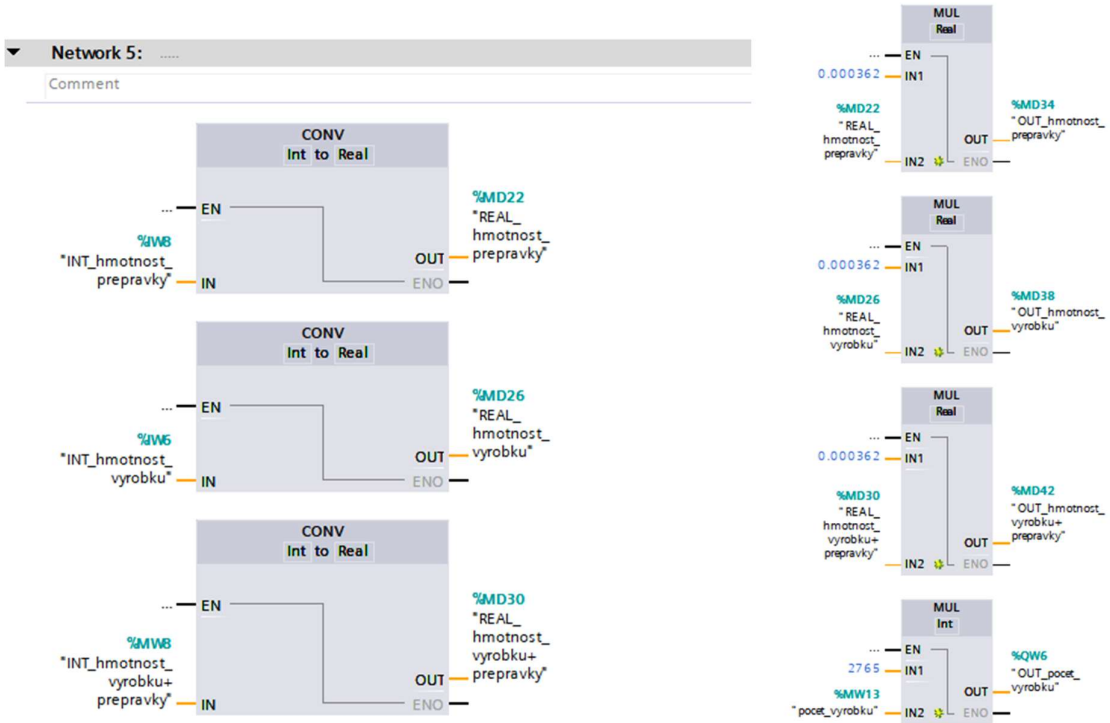


Obrázek 30: Úloha 3: Část Network 3 funkčního bloku "standardni_rezim"



Obrázek 31: Úloha 3: Část Network 4 funkčního bloku "standardni_rezim"

V části Network 4 (Obrázek 31) je řešena signalizace stavu, kdy je přepravka plná. Tento stav je vyvolán při splnění podmínky, že hodnota proměnné „INT_hmotnost_prepravky+vyrobku“ je větší, nebo rovna hodnotě 27648. Dále je zde řešen reset celého zařízení. Pokud je tedy stisknuto tlačítko „reset“, dojde k přiřazení hodnoty 0 proměnným „pocet_vyrobku“, „INT_hmotnost_prepravky“ a

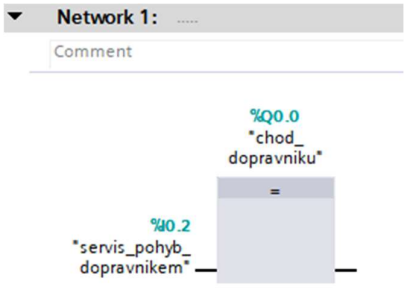


„INT_hmotnost_vyrobku+prepravky“.

Obrázek 32: Úloha 3: Část Network 5 funkčního bloku "standardni_rezim"

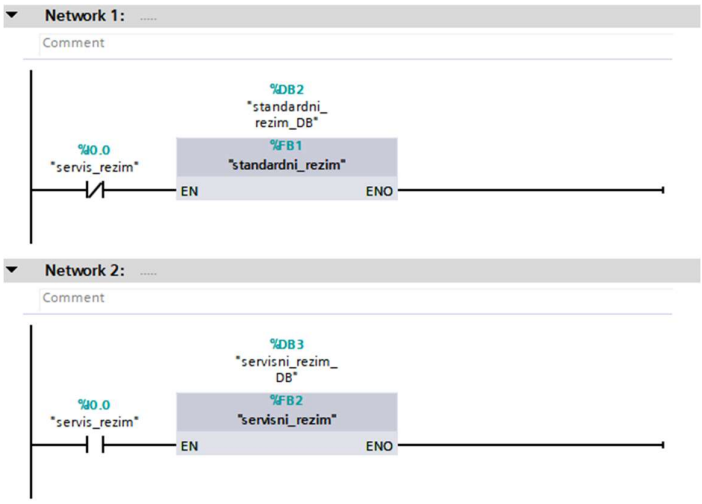
Část Network 5 (Obrázek 32) se stará o převod hodnot proměnných typu Int na Real a následný přepočet na hodnoty odpovídající příslušným napětím. Například hodnota proměnné „INT_hmotnost_prepravky“ získaná z analogového vstupu je zde převedena na desetinné číslo typu Real, poté přenásobena hodnotou 0,000362 a uložena do proměnné „OUT_hmotnost_prepravky“, která je následně zobrazena na HMI panelu. Tímto přepočtem je zajištěno, že pokud je na analogovém vstupu napětí 1,0 V (tzn. hodnota proměnné „INT_hmotnost_prepravky“ je 2765), je na HMI panelu zobrazena hodnota 1,0 kg a je splněna podmínka, že napětí na vstupu je v poměru 1:1 k simulované hmotnosti na váze. Analogicky je pak proveden přepočet proměnných „INT_hmotnost_vyrobku“ a „INT_hmotnost_vyrobku+prepravky“ na „OUT_hmotnost_vyrobku“ a „OUT_hmotnost_vyrobku+prepravky“.

Pro názornost použití analogového výstupu je zde navíc proveden přepočet hodnoty proměnné „pocet_vyrobku“ opět tak, aby byl počet výrobků k výstupnímu napětí v poměru 1:1.



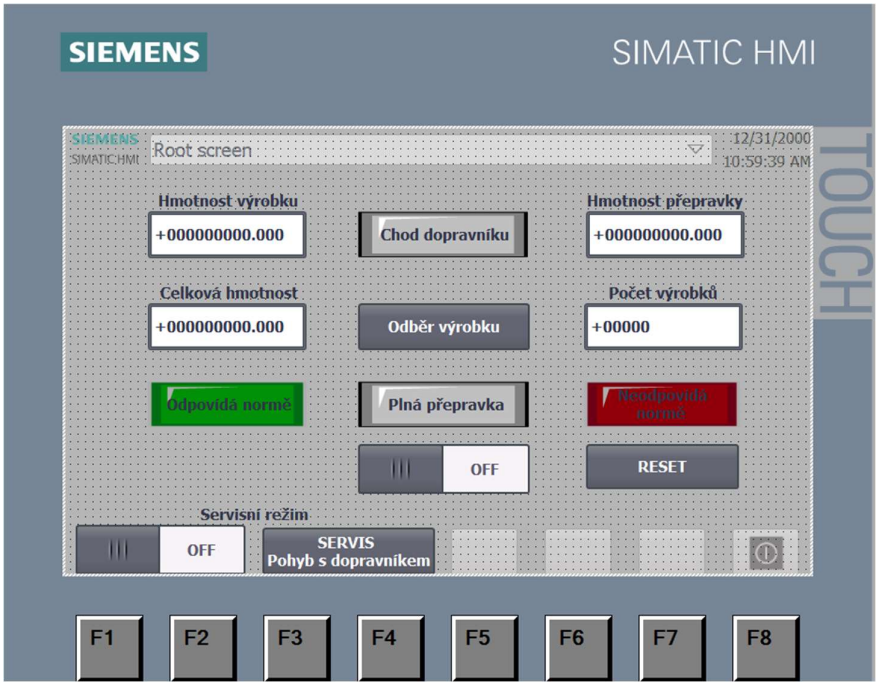
Obrázek 33: Úloha 3: Funkční blok "servisni_rezim"

Servisní režim (Obrázek 33) je v tomto případě realizován velmi jednoduše a sice, že pokud údržba drží tlačítko „servis_pohyb_dopravnikem“, je dopravník v chodu a údržba s ním může libovolně popojíždět.



Obrázek 34: Úloha 3: Hlavní část programu (main)

Hlavní část programu (Obrázek 34) je taktéž velmi jednoduchá a sice, pokud je přepínač režimů na hodnotě „false“, je spuštěn blok „standardni_rezim“, pokud naopak na hodnotě „true“, je spuštěn blok „servisni_rezim“.



Obrázek 35: Úloha 3: Uživatelské rozhraní na HMI Panelu

Závěr

Prvním cílem této bakalářské práce byl obecný popis PLC a posléze popis konkrétního programovatelného logického automatu Siemens Simatic S7-1500. Dále se také práce zabývá popisem vývojového prostředí TIA Portal, kde byly řešeny vzorové úlohy. Popis prostředí je napsán tak, aby i programátor bez zkušeností s tímto prostředím dostal alespoň základní představu o struktuře programu, o vybraných datových typech, se kterými se v prostředí setká a o jazycích, ve kterých může program vytvářet.

Dalším cílem práce byl návrh a praktická realizace přípravku, na kterém je možnost ověřit funkčnost vzorových úloh s připojením na PLC. Přípravek měl být vybaven alespoň osmi digitálními vstupy, osmi digitálními výstupy, dvěma analogovými vstupy a dvěma analogovými výstupy. Všechny tyto náležitosti realizovaný přípravek splňuje, dokonce je pro účely našich vzorových úloh vybaven až jedenácti digitálními výstupy. Zároveň byl při jeho návrhu kladen důraz na univerzálnost použití, což také splňuje.

Posledním a zároveň hlavním cílem práce byl návrh a realizace tří výukových programů, na kterých je demonstrována funkčnost výukového přípravku. První úloha týkající se řízení světelné křižovatky je realizována v kombinaci jazyků GRAPH a LAD a slouží jako ukázka typické úlohy pro sekvenční programování. Druhá úloha se zabývá řízením světelného železničního přejezdového zabezpečovacího zařízení čítači náprav na každé straně přejezdu a detekcí chybového stavu. Úloha je realizována čistě v jazyce LAD. Poslední úloha týkající se řízení dopravníku výrobků ve výrobě a jejich následného zvážení je navržena především na procvičení jazyka FBD, procvičení práce s analogovými vstupy a výstupy a jako ukázka složitějších matematických operací, které prostředí TIA Portal v jednotlivých jazycích nabízí. Funkčnost všech úloh byla ověřena jak simulačním nástrojem PLC SIM v prostředí TIA Portal, tak i na našem přípravku po připojení k PLC. Součástí všech úloh je také vizualizace na HMI Panelu, při jejichž návrhu byl kladen důraz na interaktivitu a co nejvěrohodnějším vyobrazení situace. Je však vhodné zdůraznit, že se jedná o zjednodušená řešení jednotlivých úloh. To znamená, že například u prvních dvou zmiňovaných úloh, kde na funkčnosti programu závisí lidské životy, by realizace v praxi vyžadovala plnění řady bezpečnostních požadavků, standardů a certifikací hardwaru i softwaru.

Seznam literatury a informačních zdrojů

[1] PLC – Programovatelný logický automat. *PLC-Automatizace.cz* [online]. Bakov nad Jizerou: HaPeSoft [cit. 2022-05-09]. Dostupné z: http://www.plc-automatizace.cz/knihovna/plc.htm

[2] Cyklus PLC. *PLC-Automatizace.cz* [online]. Bakov nad Jizerou: HaPeSoft [cit. 2022-05-09]. Dostupné z: http://www.plc-automatizace.cz/knihovna/plc/plc-cyklus.htm

[3] KOSEK, Rostislav a Jakub VOJANEC. Nová generace řídicích jednotek střední a vyšší výkonnosti Simatic S7-1500. *Automa* [online]. 2013, **8**(1) [cit. 2022-05-09]. Dostupné z: https://automa.cz/Aton/FileRepository/pdf_articles/10114.pdf

[4] Siemens TIA Portal – jednotné vývojové prostředí pro automatizaci v průmyslu. *Automa* [online]. 2011, **6**(3) [cit. 2022-05-09]. Dostupné z: https://automa.cz/Aton/FileRepository/pdf_articles/43212.pdf

[5] Differences between Function Block (FB) and Function (FC). *Support.Industry.Siemens.com* [online]. 2007 [cit. 2022-05-09]. Dostupné z: https://support.industry.siemens.com/forum/ww/en/posts/differences-between-function-block-fb-and-function-fc/11089

[6] Programovací jazyky PLC. *Elektroprumysl.cz* [online]. Hajany: Elektroprumysl.cz, 2013 [cit. 2022-05-09]. Dostupné z: https://www.elektroprumysl.cz/software/programovaci-jazyky-plc

[7] TIA Portal data types. *OB121* [online]. [cit. 2022-05-09]. Dostupné z: https://www.ob121.com/doku.php?id=de:s7:tia_datatypes

[8] *Katalogový list k modulu analogových vstupů AI 8xU/I/RTD/TC ST* [online]. [cit. 2022-05-16]. Dostupné z: https://cache.industry.siemens.com/dl/files/205/59193205/att_112065/v1/s71500_ai_8xu_i_rtd_tc_st_manual_en-US_en-US.pdf