UNIVERZITA PARDUBICE

Fakulta elektrotechniky a informatiky

Řízení dopravního uzlu pomocí PLC Siemens Simatic

Miroslav Jiránek

Bakalářská práce

2020

Prohlášení

Prohlašuji:

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

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

Beru na vědomí, že v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů, a směrnicí Univerzity Pardubice č. 9/2012, bude práce zveřejněna v Univerzitní knihovně a prostřednictvím Digitální knihovny Univerzity Pardubice.

V Pardubicích dne

Miroslav Jiránek

Poděkování

Tímto bych rád poděkoval vedoucímu bakalářské práce panu doc. Ing. Petru Doleželovi, Ph.D. za vždy ochotnou pomoc, cenné rady a připomínky.

V Pardubicích dne

Miroslav Jiránek

ANOTACE

Bakalářská práce se zabývá řízením a simulací dopravního uzlu. K řízení jednotlivých směrů byly semafory naprogramovány pomocí PLC a program vytvořen v prostředí TIA Portal od společnosti Siemens. Vizualizace dopravního uzlu byla provedena v prostředí MATLAB, včetně simulace vozidel. K propojení komunikace mezi fyzickým PLC a vizualizací v MATLAB byl vytvořen OPC Server pomocí softwaru KEPServerEX od společnosti Kepware.

KLÍČOVÁ SLOVA

PLC, TIA Portal, Simatic, řízení, křižovatka, Matlab, OPC Server

TITLE

Traffic control using a Siemens Simatic PLC

ANNOTATION

This bachelor thesis deals with the control and simulation of a transport hub. To control the individual directions, the traffic lights were programmed using a PLC and the program was created in the TIA Portal software by Siemens. The visualization of the transport hub was performed in the MATLAB software, including vehicle simulation. To connect the communication between the physical PLC and the visualization in MATLAB, there was created an OPC Server using the KEPServerEX software by Kepware.

KEYWORDS

PLC, TIA Portal, Simatic, control, crossroad, Matlab, OPC Server

.

OBSAH

SEZNAM ZKRATEK A ZNAČEK

CPU	Centrální procesorová jednotka
PLC	Programovatelný logický automat
DI/DO	Digitální vstup / Digitální výstup
AI/AO	Analogový vstup / Analogový výstup
HMI	Vizualizace, sběr dat a řízení technologických procesů
MES	Výrobní informační systém
ERP/MRP	Plánování podnikových zdrojů

SEZNAM ILUSTRACÍ

ÚVOD

K řízení procesů dochází již prakticky v každém odvětví, kterým se lidé zabývají. V mnoha případech je tak nahrazen lidský faktor, který může být často nespolehlivý nebo nedostatečně účinný. Před navržením řídicího procesu musí nejprve dojít k analýze a vyhodnocení současného stavu. Poté je možné navrhnout řešení, které umožní zefektivnit danou činnost. Jedním z mnoha odvětví, kde řízení procesů již plně nahradilo vliv člověka, je řízení dopravních uzlů neboli dopravních křižovatek. Právě této problematice je tato bakalářská práce věnována.

Cílem této bakalářské práce je navrhnout řídicí systém pro ovládání dopravní křižovatky pomocí programovatelného automatu, neboli PLC, který je nejrozšířenějším řídicím systémem napříč všemi průmyslovými obory. Pomocí PLC budou naprogramovány semafory řídící provoz na křižovatce. K tomu bude dále vytvořena vizualizace křižovatky včetně simulace projíždějících vozidel reagujících na změny semaforů v reálném čase. Součástí vizualizace bude také možnost uživatelem vybrat mezi třemi hustotami provozu v různých směrech a tím ovlivňovat průběh simulace.

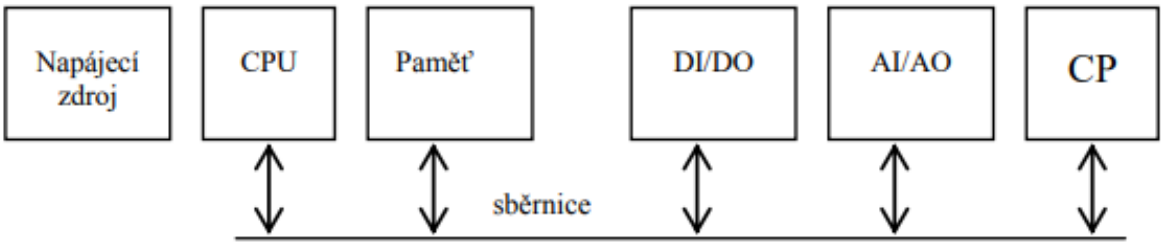
Pro vytvoření řídicího programu byl vybrán hardware i software od společnosti Siemens. Vizualizace dopravního uzlu a simulace vozidel je provedena v prostředí MATLAB od společnosti MathWorks. K propojení mezi fyzickým zařízením a vizualizací bylo využito softwaru KEPServerEX od společnosti Kepware.

1 TEORETICKÁ ČÁST

1.1 Programovatelné automaty

Jedná se o řídicí systémy, které jsou určeny převážně k řízení průmyslových a technologických procesů. Označují se jako PLC (Programmable Logic Controller). Původní záměr programovatelných automatů bylo nahrazení řízení strojů za již zastaralou a nedostatečně efektivní reléovou logiku. Na konci 60. let minulého století se začaly objevovat první jednoduché programovatelné automaty, které pracovaly zatím jenom s binární informací. V dnešní době je každé PLC velmi univerzální přístroj, kterým lze kompletně řídit daný proces od začátku až do konce bez větší nutnosti podpory ostatních zařízení (Švarc, 2002).

„Každý programovatelný automat se v podstatě skládá z centrální procesorové jednotky, systémové paměti, uživatelské paměti, souboru vstupních a výstupních jednotek pro připojení řízeného systému (technologického procesu, výrobního stroje, výrobního zařízení, výrobku) a souboru komunikačních jednotek pro komunikaci se souřadnými i nadřazenými řídicími systémy. Jednotky programovatelného automatu jsou navzájem propojeny systémovou sběrnicí“ (Koziorek, 2008). Propojení jednotlivých součástí automatu je znázorněno na obr. 1.1.



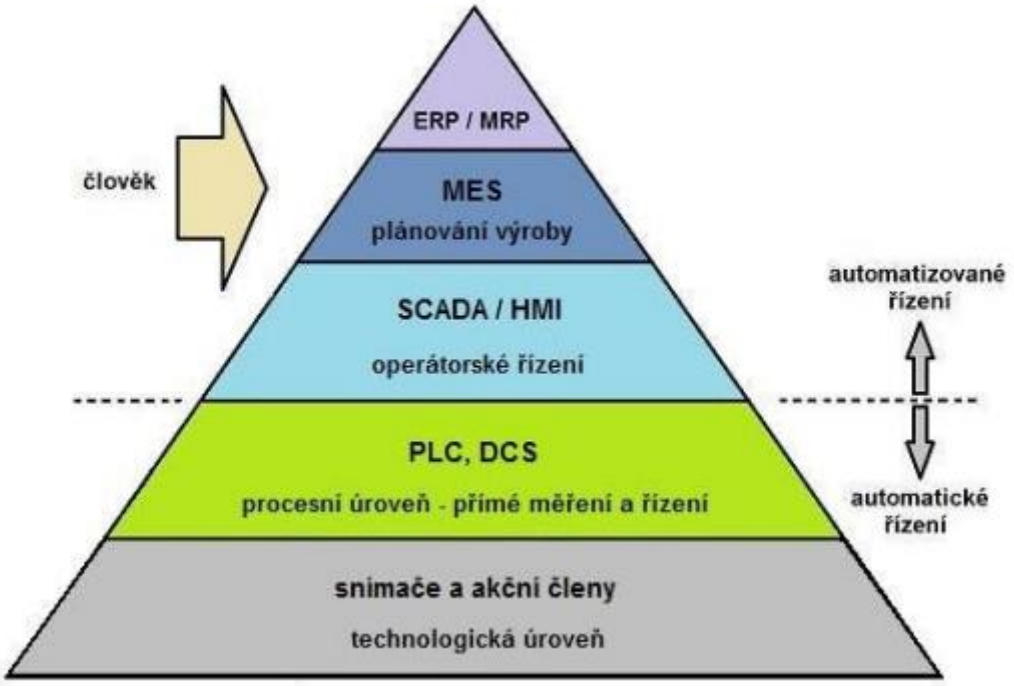
Obr. 1.1 – Blokové schéma programovatelného automatu (Koziorek, 2008)

Programovatelný automat může být využit mnoha způsoby. Vždy záleží na daném problému, který určuje, jak bude řídicí systém ovládán. Při dopředném řízení je řídicí systém ovládán bez zpětné vazby, není kontrolován dosažený stav. Tento způsob řízení je vhodný např. pro částečně automatizované prostředí, kde řídicí sytém ovládá člověk-operátor a programovatelný automat tak vykonává funkci akčního členu, který zvládne efektivněji obsluhovat dané zařízení (Koziorek, 2008).

Další způsob využití je zpětnovazební řízení. Řídicí systém získává prostřednictvím programovatelného automatu zpětnou informaci o důležitých stavech, které jsou na základě

žádané veličiny porovnávány. Snahou zpětnovazebního řízení je snížit odchylku mezi skutečnou a požadovanou veličinou na minimum. Příklad zpětnovazebního řízení pomocí logických operací může vypadat tak, že kontrolujeme například hladinu, tlak a teplotu v nádrži. Pomocí čidel získáváme informace o stavech jednotlivých veličin. Programovatelný automat potom sám vyhodnocuje, jak celý systém řídit. I přestože je programovatelný automat schopný většinu času pracovat samostatně, v praxi je alespoň občasná přítomnost operátora stále nezbytná pro monitorování celého procesu a případného seřízení či kontrole systému (Koziorek, 2008).

Na obr. 1.2. můžeme vidět hierarchické rozložení výrobního procesu.



Obr. 1.2 – Systém rozložení výrobního procesu (Šimral, 2011)

1.1.1 Rozdělení

Z konstrukčního hlediska dělíme programovatelné automaty na dvě kategorie. Jiný programovatelný automat je využit při jednoduchém ovládání domácí automatizace, jiné zařízení je naopak použito při komplexním řízení několika strojů v průmyslu. Dále rozdělujeme dle velikosti.

Kompaktní PLC

Uvnitř pouzdra je zabudován jak řídicí procesor (CPU), tak určitý počet vstupně výstupních zařízení, který je však značně omezen. Výhodou kompaktních programovatelných automatů je především nižší cena, rychlost přístupu k jednotlivým periferiím a také velmi nízká doba vykonávání cyklu, protože signály nemusí procházet přes žádný řadič sběrnice. Jako nevýhodu můžeme uvažovat nižší programovou paměť (Koziorek, 2008).

Modulární PLC

Jedná se o větší celky nejrůznějších periférií, kde je možné jednoduše přidávat moduly, které nejsou součástí základních celků. (Koziorek, 2008). Tímto je možné využít programovatelné automaty pro mnohem více úkolů najednou. S přídavnými moduly se zvyšuje programovatelná paměť, cena je však vyšší než u kompaktních programovatelných automatů.

Mikro PLC

Jedná se o nejmenší a nejlevnější programovatelné automaty na trhnu. Toto kompaktní PLC je využíváno převážně pro obsluhu jednoduchých strojů, u kterých je dostačující řízení pomocí binárních vstupů a výstupů. Počet vstupních a výstupních zařízení je v řádu jednotek. (Koziorek, 2008).

Malé PLC

Programovatelné automaty tohoto typu mohou být jak kompaktní, tak modulární. Záleží na velikosti a počtu přídavných modulů. Obvykle již obsahují větší množství funkcí než jenom binární logiku. Počet vstupně výstupních zařízení je několik desítek (Koziorek, 2008).

Střední PLC

Tento typ programovatelných automatů má již několik stovek vstupně výstupních zařízení, bývá většinou modulární (Koziorek, 2008).

Velké PLC

Pro nejsložitější projekty, které mají za úkol ovládat celé linky v průmyslových halách, se používají Velká PLC. Je zde využíváno až tisíce vstupů a výstupů s nejrůznějšími přídavnými moduly (Koziorek, 2008).

1.1.2 Princip činnosti CPU

Způsob zpracování programu se může lišit v závislosti na výrobci, nebo jaké PLC používáme. Pro tuto práci bylo vybráno PLC od firmy Siemens, S7 – 1200, které bude následně podrobně popsáno.

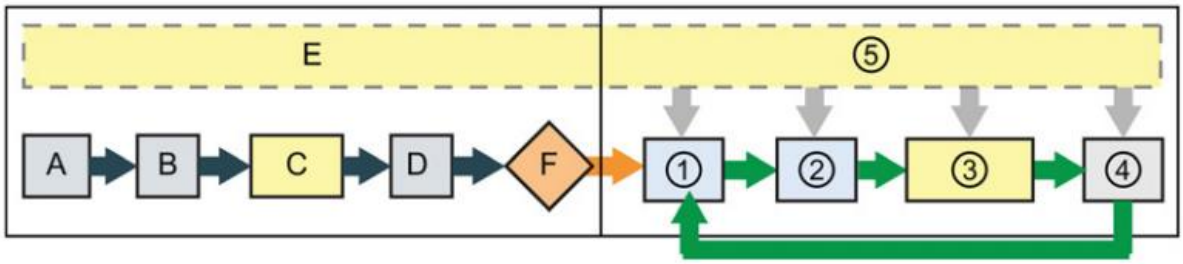
Po nahrání programu do PLC jsou procesorem zjištěny aktuální stavy na vstupech a na základě vykonání uživatelského programu jsou změněny jednotlivé výstupy. Tomuto procesu říkáme skenovací cyklus, ve kterém dojde k zapsání hodnot vstupně/výstupních periférií do vnitřní paměti CPU, takzvané *process image*. Skenovací cyklus slouží ke konzistentnímu vykonávání programu, kde změna vstupně výstupních zařízení je provedena pouze jednou za cyklus (SIEMENS, 2015). Samotné CPU se může nacházet ve třech režimech, které jsou indikovány pomocí RUN/STOP LED na přední straně PLC, jak je možné vidět na obr. 1.3.

Obr. 1.3 – Zobrazení aktuálního pracovního režimu PLC (SIEMENS, 2015)

Režimy CPU jsou následující:

- **STOP** – aktuální program není vykonáván, je možné zavést program z nového projektu. RUN/STOP LED svítí žlutou barvou,
- **STARTUP** – provedení inicializace. CPU v tomto režimu nemůže provést přerušení. RUN/STOP LED bliká žlutou a zelenou barvou,
- **RUN** – opakované provádění skenovacího cyklu. RUN/STOP LED svití zelenou barvou (SIEMENS, 2015).

Na obr. 1.4. jsou detailně popsány režimy STARTUP a RUN



Obr. 1.4 – Princip činnosti režimů CPU (SIEMENS, 2015)

STARTUP režim

A) Vymazání vstupních stavů z vnitřní paměti.
B) Načtení výstupních stavů z vnitřní paměti.
C) Provedení inicializace organizačních bloků.
D) Načtení aktuálních vstupních stavů do vnitřní paměti.
E) Uložení všech událostí přerušení do fronty. Při režimu RUN budou tato přerušení ve stejném pořadí volána.
F) Povolení zápisu výstupních stavů z vnitřní paměti na fyzický výstup.

RUN režim

1) Zápis výstupních stavů z vnitřní paměti na fyzické výstupy.
2) Uložení kopie vnitřních stavů z fyzického vstupu do vnitřní paměti.
3) Provedení programu.
4) Vlastní diagnostický test.
5) Provedení obsluhy přerušení, nebo komunikace, které byly při skenovacím cyklu zavolány (SIEMENS, 2015).

1.2 SIEMENS SIMATIC S7 – 1200

Jedná se o programovatelný automat poskytující dostatečný výkon a kompaktní design s flexibilní konfigurací přídavných modulů. Díky těmto vlastnostem a také příznivé ceně je série S7 – 1200 jedním z nejvyužívanějších programovatelných automatů od společnosti Siemens. „Hlavní důraz je v něm kladen na snadnou integraci a hladkou spolupráci jednotlivých částí řídicího systému, tj. komunikačních prostředků, operátorského rozhraní (HMI) a samotného programovatelného automatu (PLC). Přínosem pro uživatele je zejména možnost rychle a přehledně projektovat a snadno realizovat a uvádět do provozu nejrůznější automatizační systémy“ (Rakušan, 2012).

1.2.1 Technické parametry

Programovatelné automaty řady S7 – 1200 jsou vyráběny v pěti verzích. Ty se liší v rozměrech, počtu vstupně/výstupních zařízení a parametrů CPU. Na obr. 1.5 můžeme vidět porovnání jednotlivých modelů PLC. Pro tuto práci byl vybrán typ S7 – 1200 PLC CPU1212C.

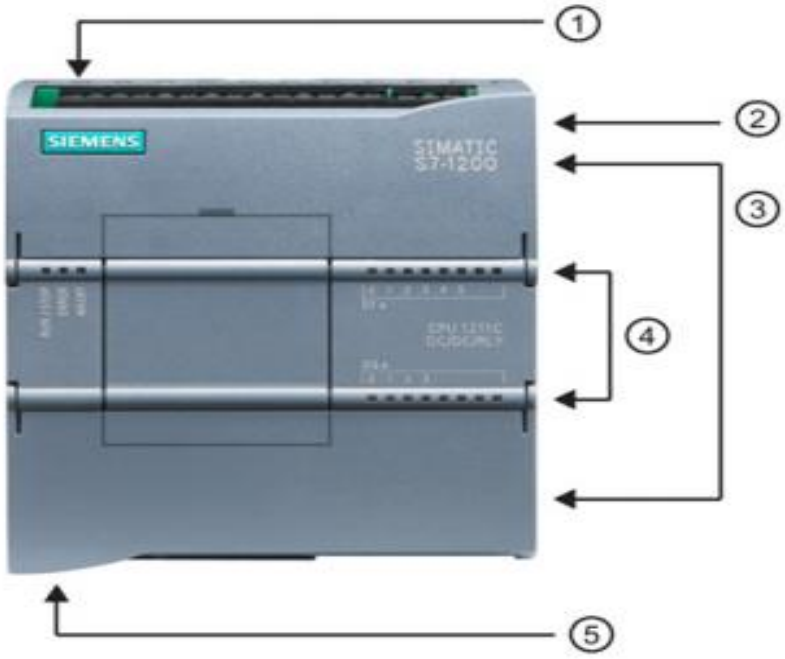
Model		**CPU 1211C**	**CPU 1212C**	**CPU 1214C**	**CPU 1215C**	**CPU 1217C**
Rozměry (mm)		90 x 100 x 75		110 x 100 x 75	130 x 100 x 75	150 x 100 x 75
Uživatelská paměť	Pracovní	50 KB	75 KB	100 KB	125 KB	150 KB
	Zaváděcí	1 MB		4 MB		
	Zálohovací	10 KB				
Vstupy/Výstupy	Digitální	6 vstupů/4 výstupy	8 vstupů/6 výstupy	14 vstupů/ 10 výstupů		
	Analogové	2 vstupy			2 vstupy/2 výstupy	
Velikost obrazu procesu	Vstupy (I)	1024 B				
	Výstupy (Q)	1024 B				
Bitová paměť (M)		4096 B		8192 B		
Signální moduly (SM) - rozšíření		Žádné	2	8		
Signální karta (SB), Záložní karta (BB), Komunikační karta (CB)		1				
Komunikační moduly (CM) - rozšíření		3				
Paměťová karta		SIMATIC Memory Card (volitelné)				
Hodiny reálného času - zálohování		20 dní/12 min. při 40°C				
PROFINET Ethernet komunikační port		1			2	
Rychlost matematických výpočtů		2,3 µs/příkaz				
Rychlost logických operací		0,08 µs/příkaz				

Obr. 1.5 – Technické parametry jednotlivých CPU (SIEMENS, 2015)

1.2.2 Možnosti komunikace

S7 – 1200 PLC má integrované rozhraní Profinet. Díky svému všestrannému využití, jako je jednoduché propojení s nadřazenými systémy nebo realizace síťové topologie, je tento protokol kvalitním nástupcem staršího protokolu Profibus. K propojení mezi dvěma zařízeními je potřeba znát unikátní adresu MAC a IP adresu, která byla zařízení přidělena (AUTOMA, 2012). Pomocí tohoto protokolu je možné propojit jednotlivé PLC mezi sebou pomocí průmyslového Ethernet konektoru dosahující rychlosti 100 Mb/s (Drahoš, 2006).

Na obr. 1.6 můžeme vidět S7 – 1200 PLC včetně grafického popisu



Obr. 1.6 – Popis S7 – 1200 PLC CPU1211C (SIEMENS, 2015)

Popis zařízení

1) Napájecí konektor,
2) slot pro paměťovou kartu,
3) odnímatelné připojovací konektory pro přídavné moduly,
4) LED diody indikující stav vstupně/výstupních zařízení,
5) Profinet konektor.

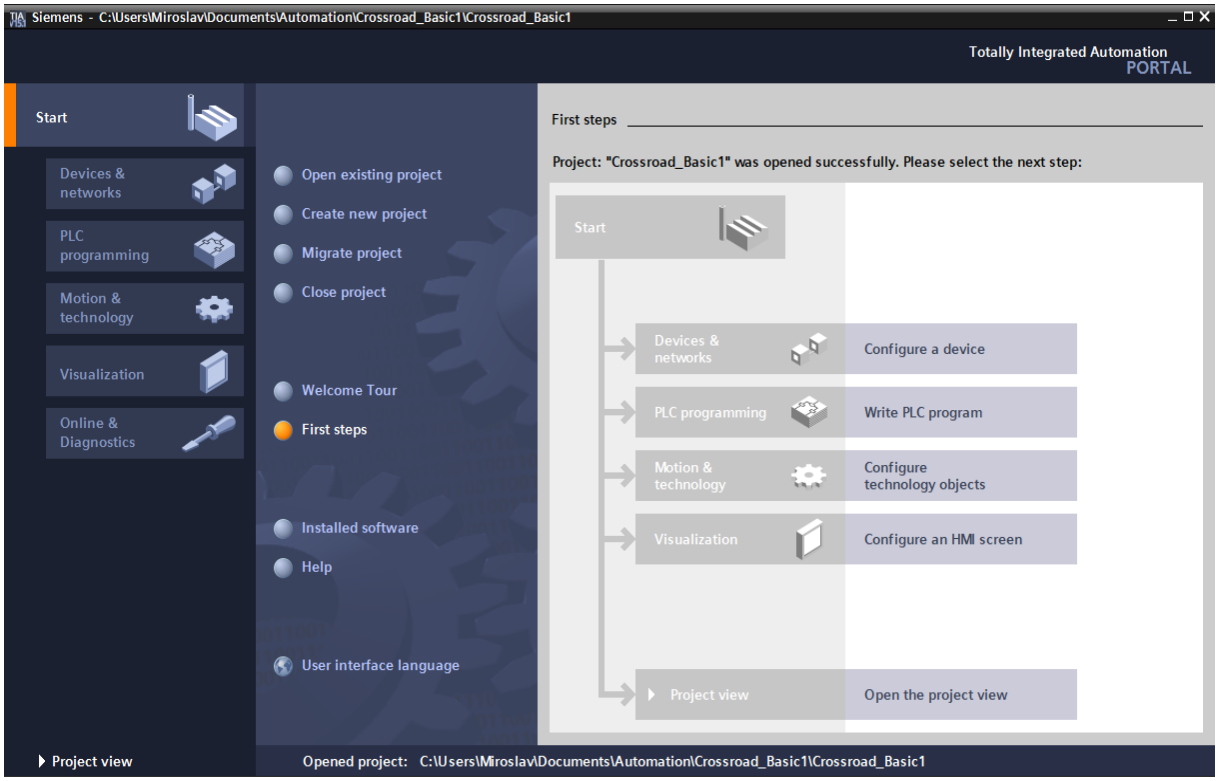
1.3 TIA Portal

Software Siemens TIA Portal je vývojové prostředí pro automatizaci v průmyslu. „V něm jsou ve společném softwarovém prostředí s jednotným ovládáním z jedné jediné

plochy integrovány veškeré nástroje potřebné k projektování a konfigurování řídicích systémů, operátorských rozhraní pro strojní celky i dispečerské systémy, komunikačních sítí a elektrických pohonů. Výsledkem je nejen rychlejší tvorba aplikačních programů, ale také zajištění konzistence dat v celém automatizačním projektu a transparentnosti jeho struktury, kterou lze snadno měnit či doplňovat a která usnadňuje diagnostiku, údržbu a servis automatizační techniky v běžném provozu“ (AUTOMA, 2011).

Součástí prostředí TIA Portal je také nástroj pro vytváření uživatelských operátorských rozhraní, Simatic WinCC, které slouží k obsluze strojů, nebo systémů, řízené pomocí PLC. Simatic WinCC je zároveň dodávaný v několika odstupňovaných variantách, což umožňuje dodávat software přizpůsobený k potřebám, kterým byl předurčen (AUTOMA, 2011).

Na obr. 1.7. je zobrazeno základní rozdělení projektu. Vše je zde logicky a přehledně poskládáno. Můžeme tak zde například nakonfigurovat PLC, podívat se na jednotlivé části programu, nebo přejít do vizualizační části (HMI).



Obr. 1.7 – Základní rozdělení projektu v TIA Portal

1.4 MATLAB

MATLAB, neboli Matrix Laboratory, je vysokoúrovňový programovací jazyk využívaný primárně pro rychlé numerické výpočty, programování a vizualizaci. Tento nástroj je vybaven velkým počtem knihoven a toolboxů, které je možné přidávat v závislosti na způsobu využití. Pomocí Matlabu je možné automaticky konvertovat napsaný kód do programovacích jazyků jako Java, Python, C++ a dalších. Při tvorbě vizualizace je využíváno GUI, neboli grafické uživatelské prostředí, kde je možné vytvořit aplikaci pouze pomocí předem vytvořených panelů, což je výhodné především pro začínající programátory, kteří se s prostředím teprve učí (Educba.com, nedatováno).

Systém MATLAB dále nabízí:

- Práce s maticemi,
- tvorba 2D a 3D grafů,
- algebraické rovnice,
- import a export dat z excelu,
- statistika,
- nelineární funkce,
- analýza dat,
- numerické výpočty,
- diferenciální rovnice,
- objektové programování (Educba.com, nedatováno).

1.5 OPC

OPC, neboli OLE for Process Control, je standard průmyslové komunikace, který byl vytvořen za účelem propojení různých zařízení, které monitorují a řídí technologické procesy. Smyslem OPC je vytvoření rozhraní, které zabraňuje nutnosti používat software a hardware od stejného výrobce. Díky OPC je tedy možné si vybrat ideální řešení, které bude jak z hlediska efektivity, financí a dalších faktorů tou nejlepší volbou. Další výhodou OPC je nepřetržitý přístup k datům v technologických provozech, které jsou dostupné i po změnách v hardwaru a odpadá tak potřeba vytváření stále nových ovladačů pro software (Stianko, 2004).

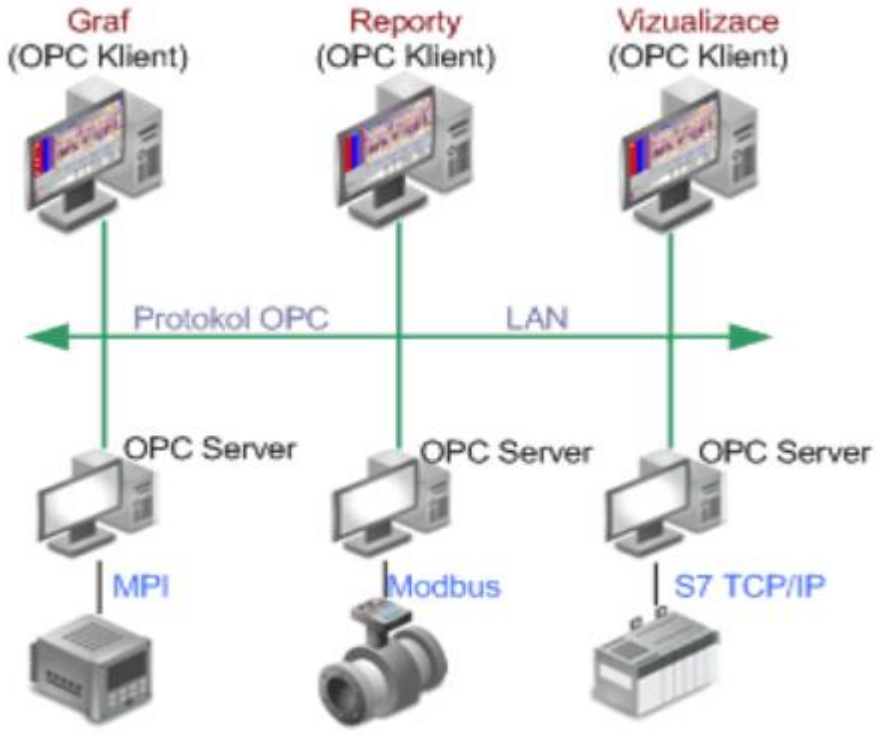
OPC Foundation

Je organizace zabývající se definováním a sdružováním tzv. OPC specifikací, které slouží k nastavení mezinárodních standardů rozhraní OPC. Mezi nejpoužívanější OPC specifikace můžeme zařadit:

- OPC Data Access (OPC DA – nejčastěji používaná specifikace),
- OPC UA (Unified Architecture),
- OPC AE (Alarm & Events),
- OPC HDA (OPC Historical Data Access) (Foxon.cz, nedatováno).

1.5.1 Architektura OPC

OPC standard využívá k výměně dat architekturu klient-server. Pojem server lze chápat jako zdroj signálu, od kterého získáváme data. Klient poté tato data zpracovává a vyhodnocuje. K jednomu vytvořenému serveru je možné připojit několik klientů od různých výrobců a naopak. Jeden klient může zpracovávat data z několika serverů najednou (Foxon.cz, nedatováno). Na obr. 1.8 můžeme vidět komunikaci mezi OPC klientem a OPC serverem prostřednictvím OPC protokolu



Obr. 1.8 – Princip komunikace pomocí OPC (Foxon.cz, nedatováno)

OPC klient

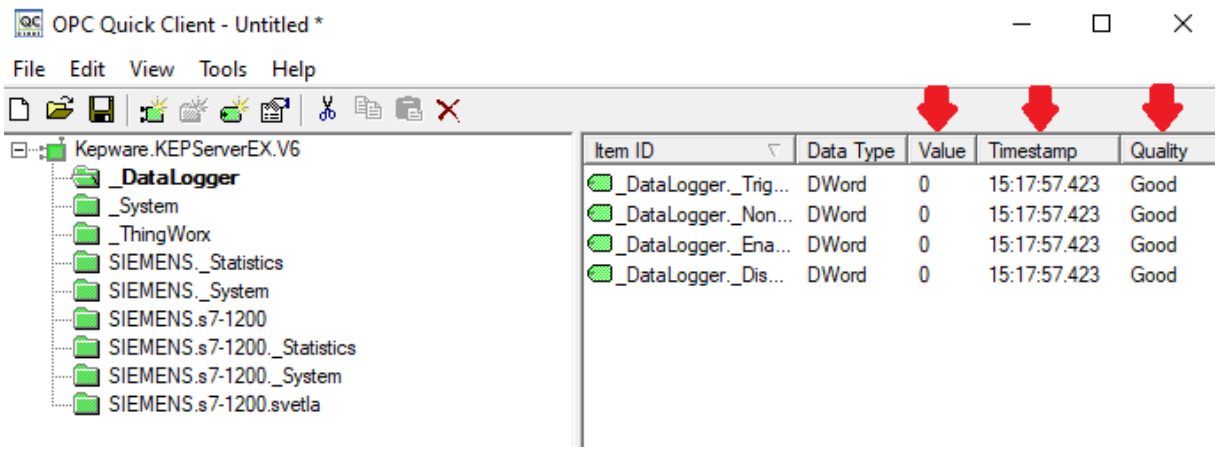
OPC klient je softwarový program, pomocí kterého je možné přijímat data z OPC serveru a tato data dále zpracovávat pro uživatelské účely, jako je například ovládání HMI, SCADA a další (Foxon.cz, nedatováno).

OPC server

Jedná se o softwarový program, který umožňuje navázat spojení mezi jednotlivými zařízeními pomocí průmyslových protokolů. Získaná data jsou poté předána OPC klientovi, který data dále zpracovává (Foxon.cz, nedatováno).

Software, na kterém je zprovozněný OPC server, musí nějakým způsobem navázat komunikaci s druhým zařízením, na kterém je zprovozněný OPC klient. K tomuto propojení se využívá tzv. OPC item (v dalších OPC specifikacích se můžeme také setkat s pojmenováním OPC tag, nebo I/O bod). Ze strany OPC klienta je pak OPC item jasně definován pomocí tzv. Item ID (Foxon.cz, nedatováno).

Jak můžeme vidět na obr. 1.9, OPC item je možné v reálném čase sledovat a kontrolovat jeho kvalitu připojení (Quality), momentální stav (Value) a čas, kdy byl poslední stav změřen (Timestamp).



Obr. 1.9 – Ukázka OPC itemů

2 PRAKTICKÁ ČÁST

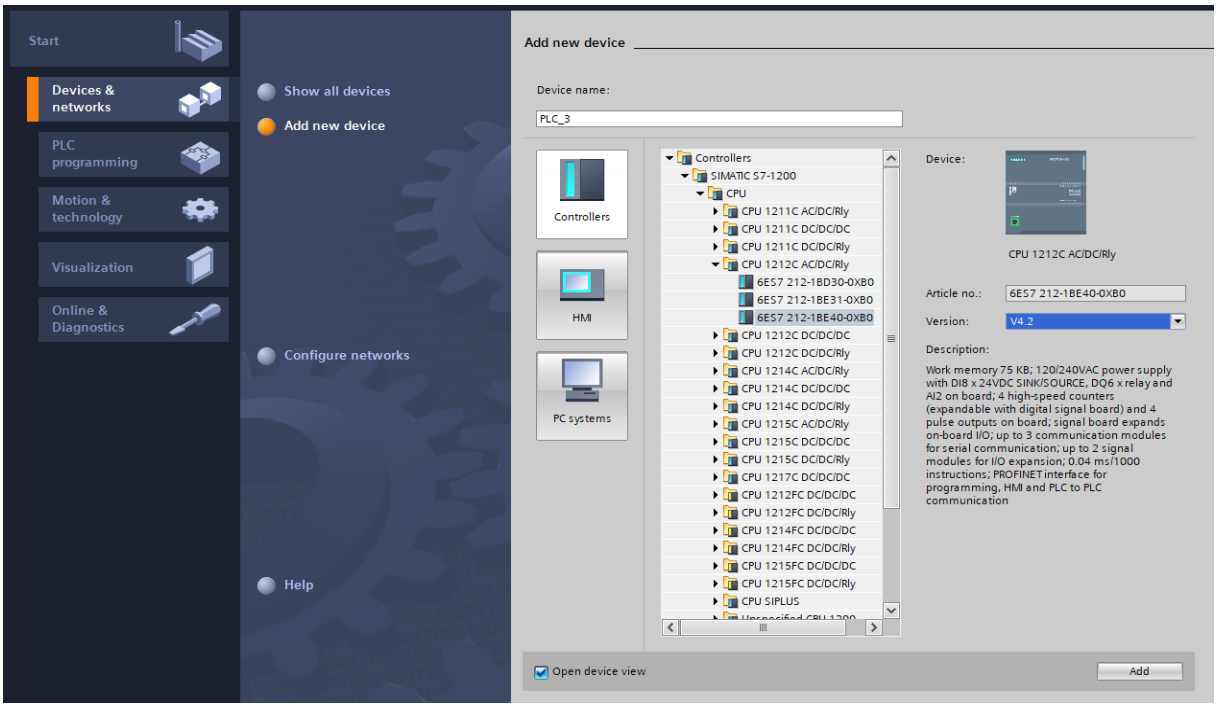
Praktická část se zabývá realizací modelu dopravního uzlu včetně simulace a vizualizace křižovatky s vozidly, kde nejdříve dojde ke konfiguraci programovatelného automatu, vybrání správného zařízení a testu komunikace v prostředí TIA Portal. Dále bude vytvořen řídicí program pro řízení semaforů na křižovatce a následně popsán princip vykonávání programu.

Aby bylo možné sledovat řízení dopravního uzlu v reálném čase, bude k tomuto účelu vytvořena křižovatka včetně simulace vozidel v prostředí MATLAB. Tato vozidla budou reagovat na změny stavů semaforů a budou si kontrolovat bezpečné rozestupy mezi sebou, aby nedocházelo ke kolizím. Součástí simulace bude také možná volba hustoty provozu v daných směrech. Na výběr bude mezi nízkou, střední a vysokou hustotou provozu, kterou bude možné měnit během simulace. Aby v simulaci bylo možné pracovat s aktuálními stavy semaforů řízené pomocí PLC, bude v MATLABu vytvořeno rozhraní založené na komunikačním protokolu OPC.

K propojení řídicího systému (TIA Portal) a vizualizačního programu (MATLAB) bude vytvořen OPC server pomocí softwaru KEPServerEX, díky kterému budou aktuální stavy semaforů posílány na server a následně použity pro řízení křižovatky.

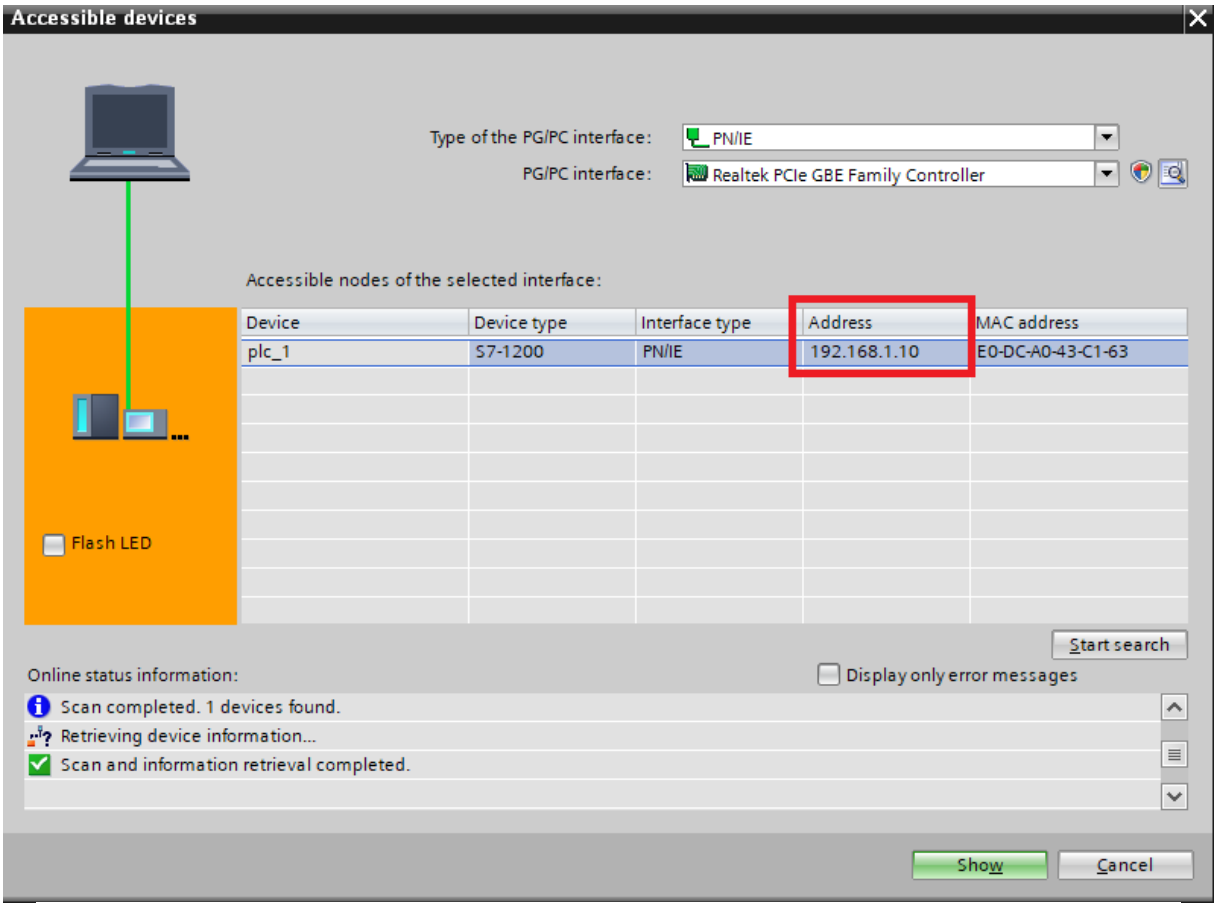
2.1 Konfigurace PLC

Abychom mohli začít vytvářet řídicí program v TIA Portal, je zapotřebí vybrat správný PLC hardware a následně provést konfiguraci. Na obr. 2.0 můžeme vidět výběr PLC v nově založeném projektu. V záložce *Device & Networks* byl vybrán model SIMATIC S7 – 1200 CPU 1212C. Po rozkliknutí vybraného modelu máme k dispozici další tři varianty CPU, které se od sebe liší především ve velikosti pracovní paměti.



Obr. 2.0 – Výběr fyzického zařízení PLC

Po nastavení správného typu PLC a po připojení PLC pomocí průmyslového Ethernet konektoru do počítače bude proveden test dostupných zařízení. K tomuto slouží v oblasti panelu nástrojů ikona *Accessible devices*. Při rozkliknutí je nutné vybrat správný typ komunikace mezi PC a PLC. Úspěšné propojení zařízení je zobrazeno na obr. 2.1. Pro otestování, že TIA Portal s PLC komunikuje, je možné na levé straně zakliknout políčko *Flash LED* a dojde k rozblikání diod na PLC. Pomocí tohoto testu jsme také zjistili velice důležitou IP adresu zařízení, kterou budeme následně potřebovat k propojení s OPC serverem.



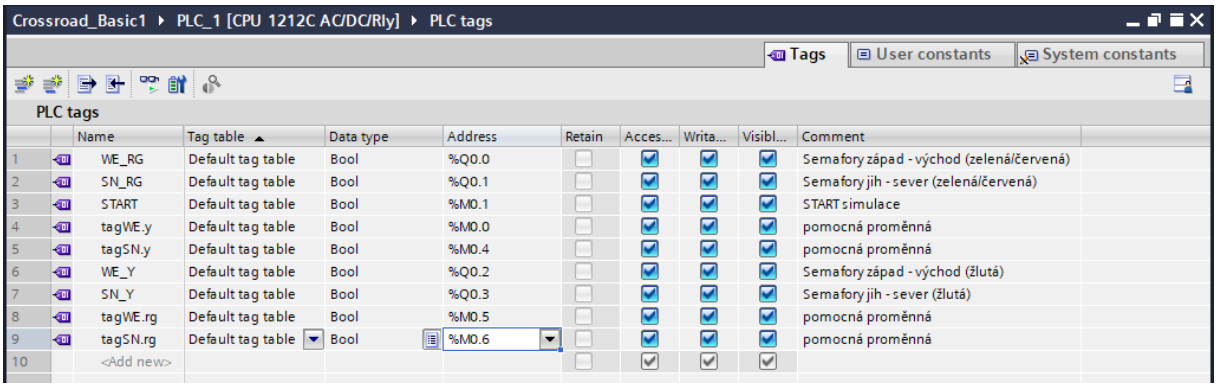
Obr. 2.1 – Test dostupných zařízení a zjištění IP adresy PLC

2.2 Řízení semaforů

Nyní bude popsán princip řízení semaforů pomocí PLC. Křižovatka je vytvořena mezi dvěma vozovkami, kde auta jezdí v obou směrech (vždy pouze rovně). Aby nedocházelo ke kolizím s auty z druhé vozovky, jsou vytvořeny řídicí semafory, které cyklicky mění stav průjezdnosti jednotlivých směrů vozovek.

Aby bylo možné semafory pomocí PLC řídit, je zapotřebí v TIA Portal nadefinovat proměnné, neboli *PLC tagy*. U těchto proměnných je důležité vybrat správný datový typ. Vždy je důležité si rozmyslet, k čemu konkrétní proměnná bude využívána. Pokud má proměnná sloužit například k přičítání hodnot na čítači, budeme uvažovat o číselném datovém typu, např. *Integer*. Pro řízení semaforů bude vhodný datový typ binární, neboli proměnná, která může nabývat pouze dvou stavů (True – False). Při definování nových proměnných je také potřeba si uvědomit, jak bude proměnná deklarována. Pokud bude proměnná ovlivňovat vnější chování řídicího systému, bude v PLC deklarována jako výstupní (např. Q0.1). Pokud je naopak proměnná z řídicího systému posílána do PLC, je nutné tuto proměnnou deklarovat jako vstupní

(např. I0.3). Dalším datovým typem může být *Memory bit* (např. M0.5), což je proměnná pracující pouze uvnitř paměti PLC a může sloužit např. jako pomocná proměnná pro udržení určitého stavu. Existuje mnoho dalších datových typů a možných adresací, v této práci však používání jiných datových typů nebude potřeba. Na obr. 2.2 můžeme vidět deklaraci proměnných pro řízení semaforů včetně komentářů, které popisují funkci proměnné.

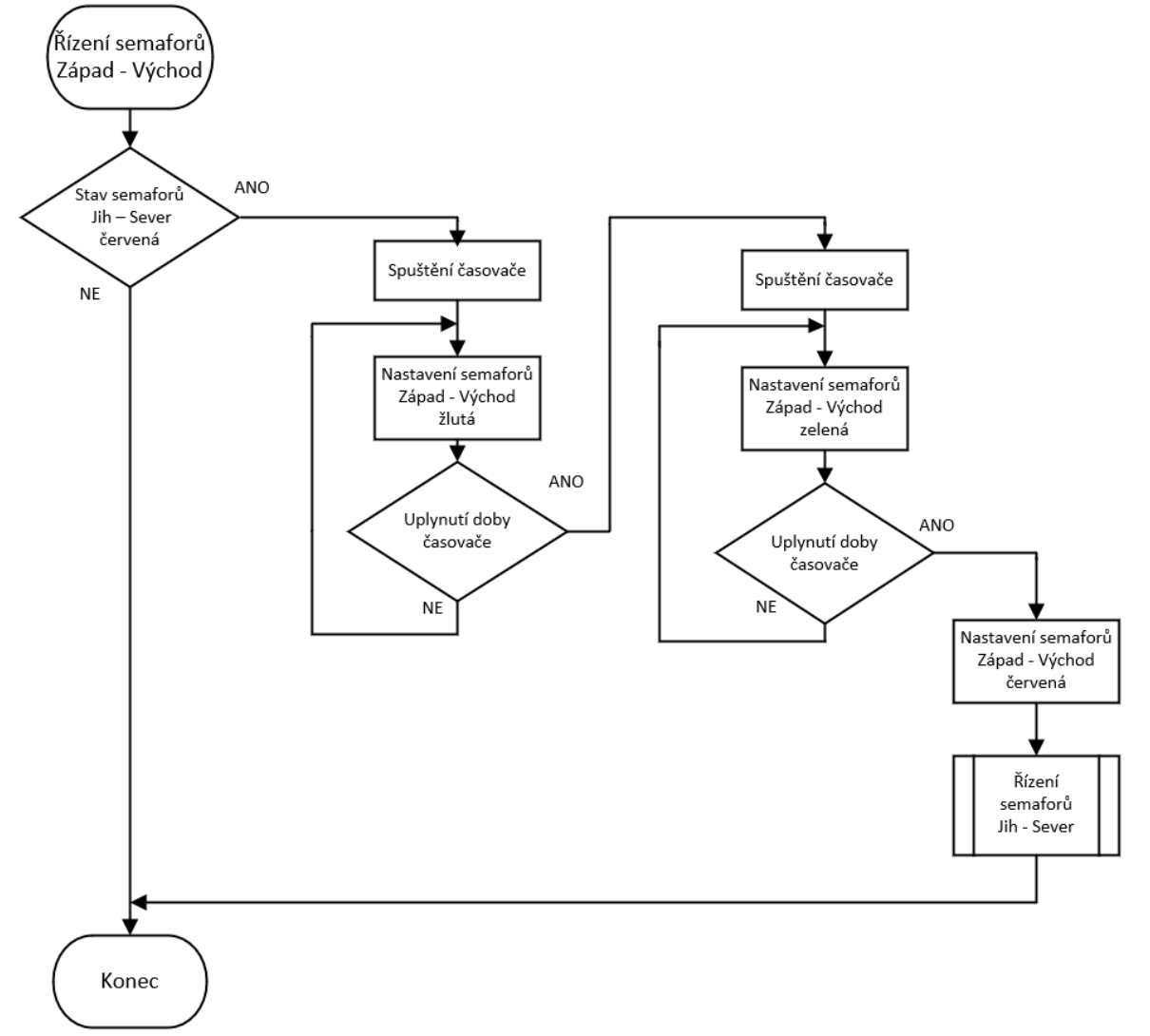


	Name	Tag table	Data type	Address	Retain	Acces...	Writa...	Visibl...	Comment
1	WE_RG	Default tag table	Bool	%Q0.0	☐	☑	☑	☑	Semafory západ - východ (zelená/červená)
2	SN_RG	Default tag table	Bool	%Q0.1	☐	☑	☑	☑	Semafory jih - sever (zelená/červená)
3	START	Default tag table	Bool	%M0.1	☐	☑	☑	☑	START simulace
4	tagWE.y	Default tag table	Bool	%M0.0	☐	☑	☑	☑	pomocná proměnná
5	tagSN.y	Default tag table	Bool	%M0.4	☐	☑	☑	☑	pomocná proměnná
6	WE_Y	Default tag table	Bool	%Q0.2	☐	☑	☑	☑	Semafory západ - východ (žlutá)
7	SN_Y	Default tag table	Bool	%Q0.3	☐	☑	☑	☑	Semafory jih - sever (žlutá)
8	tagWE.rg	Default tag table	Bool	%M0.5	☐	☑	☑	☑	pomocná proměnná
9	tagSN.rg	Default tag table	Bool	%M0.6	☐	☑	☑	☑	pomocná proměnná
10	<Add new>				☐	☑	☑	☑	

Obr. 2.2 – Deklarace PLC tagů

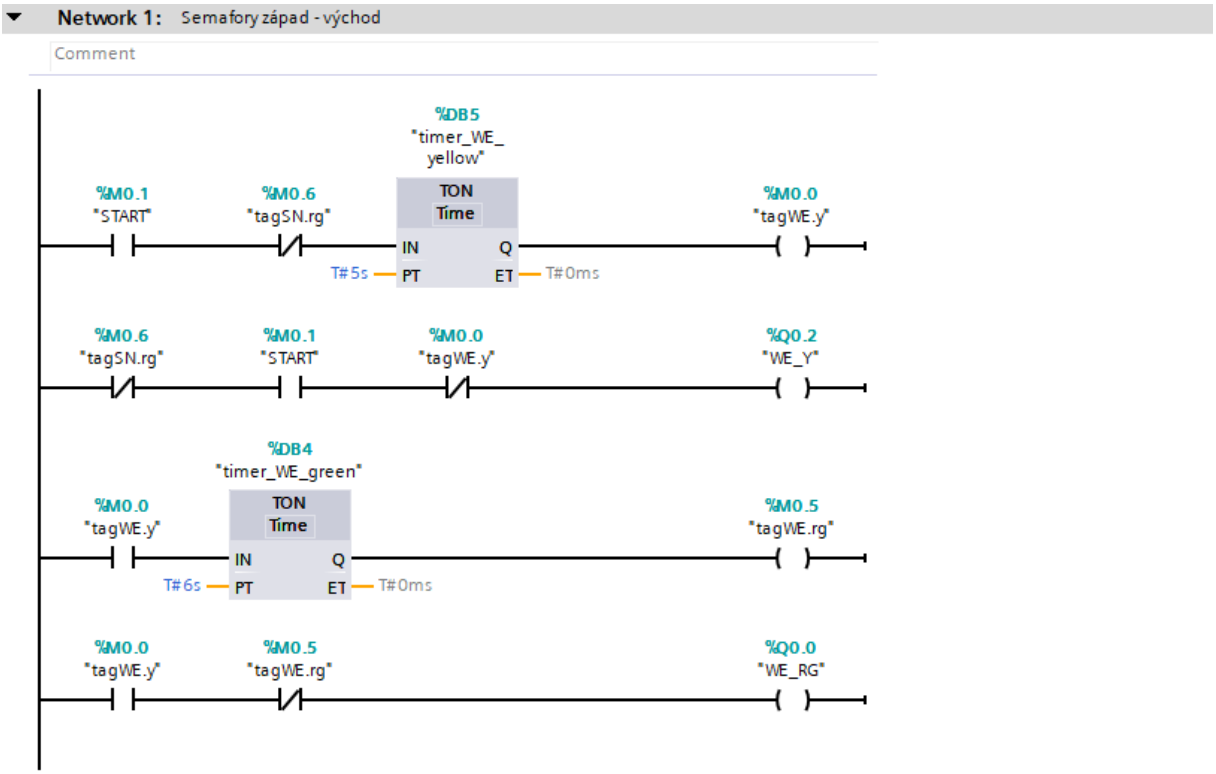
Popis funkce řídicího programu

Program je vykonáván v nekonečné smyčce, kdy se stavy semaforů střídají pomocí časovačů, které jsou nastaveny tak, aby vozidla měla dostatečný čas na opuštění křižovatky. Na obr. 2.3 můžeme vidět algoritmus řízení semaforů pro směr Západ – Východ. Stavy semaforů jsou stejné i pro vozidla v opačném směru, tedy Východ – Západ, proto je jejich řízení společné.



Obr. 2.3 – Vývojový diagram pro řízení semaforů

Na obr. 2.4 je ještě možné vidět část řídicího programu vytvořeného v grafickém programovacím jazyku Ladder v prostředí TIA Portal.

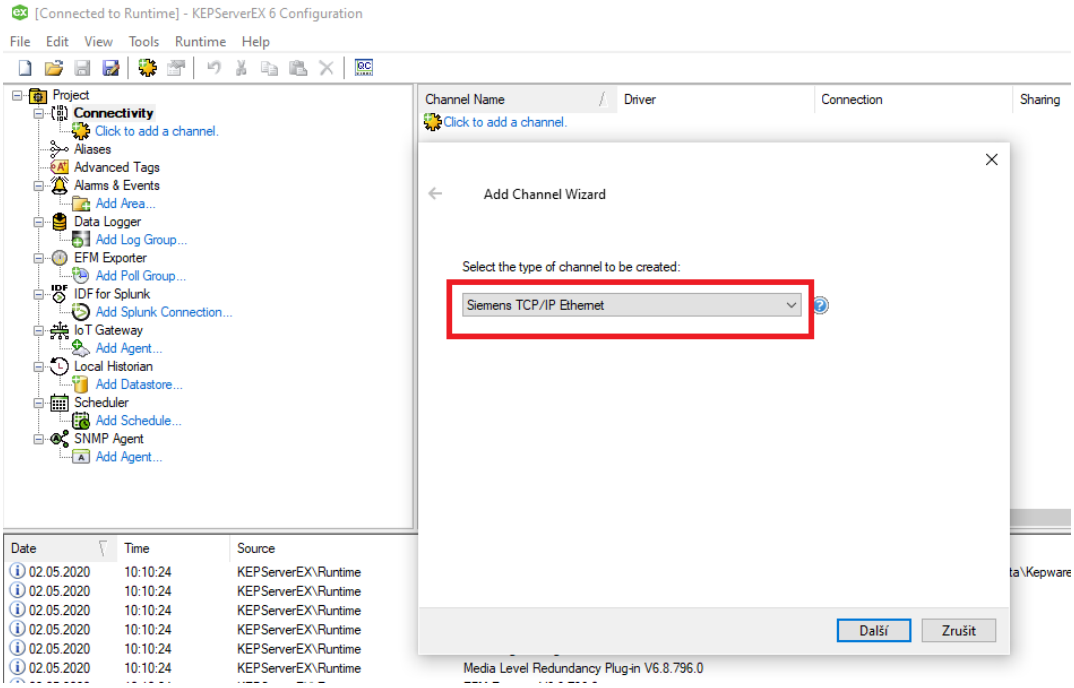


Obr. 2.4 – Realizace algoritmu řízení pomocí Ladder

2.3 Nastavení OPC serveru

K propojení řízení semaforů s vizualizací křižovatky je potřeba připojit se k OPC serveru. Pro vytvoření OPC serveru byl vybrán software od společnosti Kepware, která nabízí bezplatnou demo verzi softwaru KEPServerEX. Omezení demo verze je, že OPC server může být připojený maximálně po dobu dvou hodin. Po opětovném zapnutí komunikace je však možné server dále obsluhovat, což je pro potřeby této práce naprosto dostačující.

Aby bylo možné vytvořit komunikaci mezi počítačem a externím zařízením, je nejprve zapotřebí vytvořit tzv *Channel*, který je možné najít v levé části pod záložkou *Connectivity*. Po rozkliknutí se otevře okno, které vyzve k výběru vhodného driveru. Do vytvořeného *Chanellu* bude později možné přidávat pouze zařízení, která jsou podporována tímto driverem, proto je důležité předem zjistit, jaký driver vybrat. V tomto případě bude komunikace zajištěna prostřednictvím průmyslového Ethernet konektoru, který je podporován driverem uvedeným na obr. 2.5.

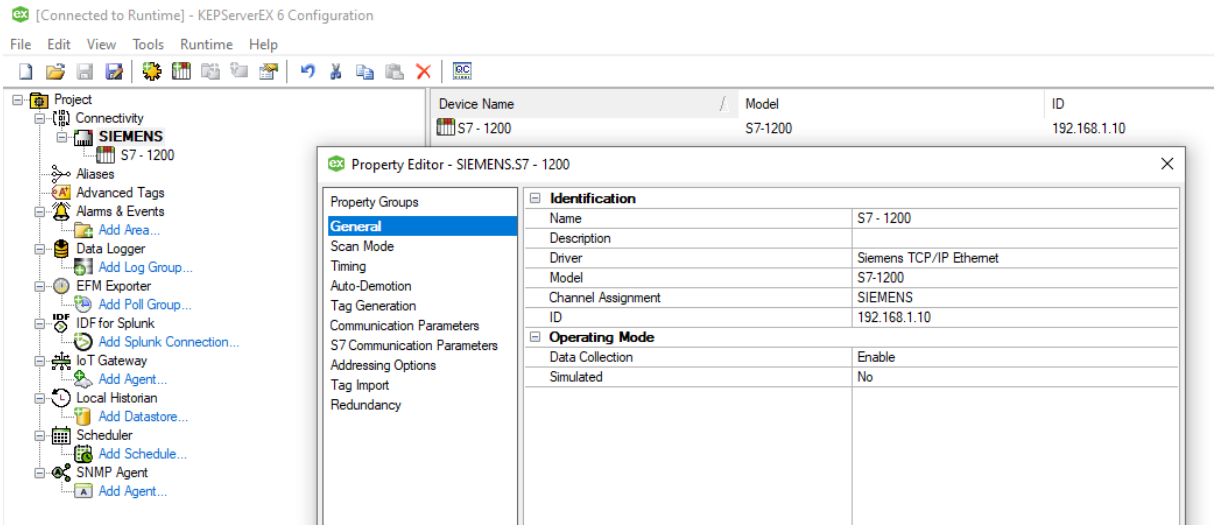


Obr. 2.5 – Výběr vhodného driveru pro komunikaci

V dalším nastavení *Channelu* je možné vybrat způsob zápisu dat nebo poměr mezi počtem zápisů ku čtení dat. Je však možné ponechat přednastavené vlastnosti a případně je později upravit.

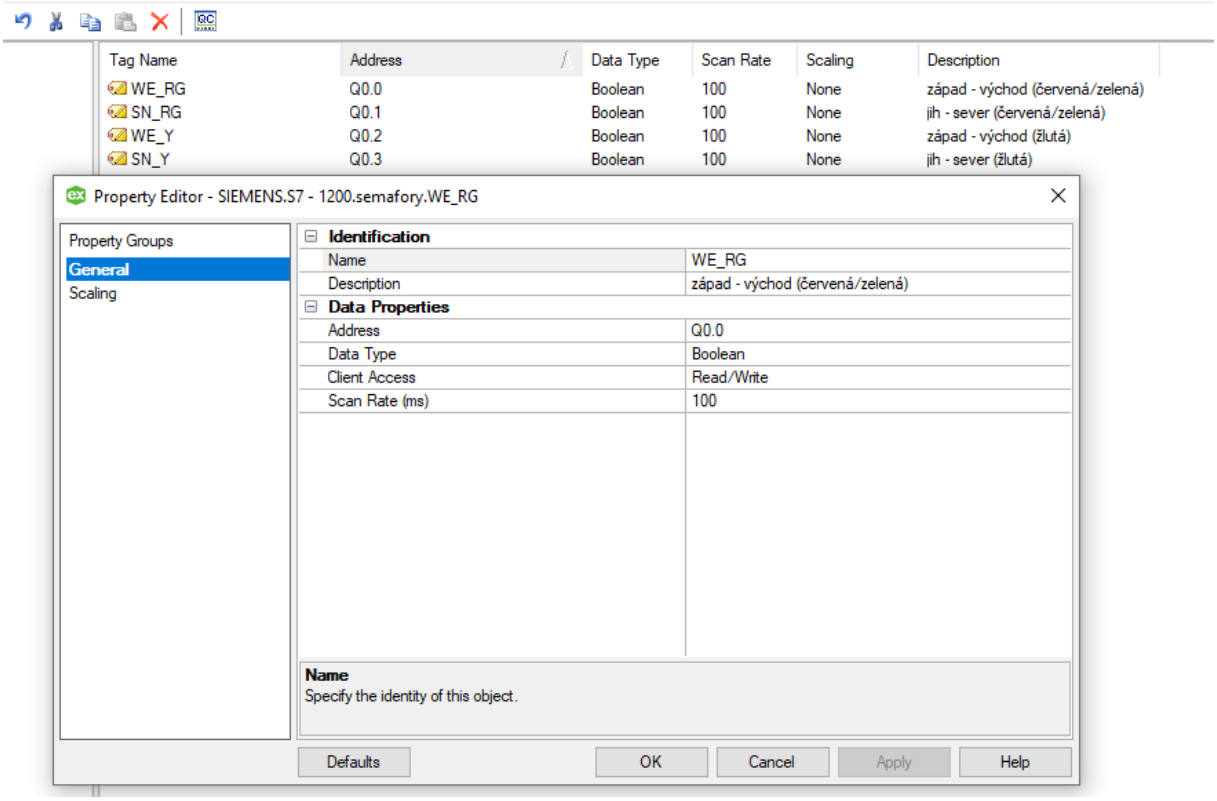
Pod vytvořeným *Channelem* se nyní objevila možnost přidat nové zařízení. Po rozkliknutí dojde nejprve k pojmenování a následně k výběru zařízení od firmy Siemens, která jsou kompatibilní se zvoleným driverem. Po výběru zařízení je nutné nastavit správnou adresu PLC, která byla v předchozí kapitole zjištěna prostřednictvím testu dostupných zařízení v prostředí TIA Portal.

Na obr. 2.6 je možné vidět vytvořené zařízení pojmenované S7 – 1200 včetně rozpisu vlastností, které byly nastaveny. Před dalším pokračováním je dobré si vše znovu zkontrolovat, aby nedošlo k případnému problému v komunikaci.



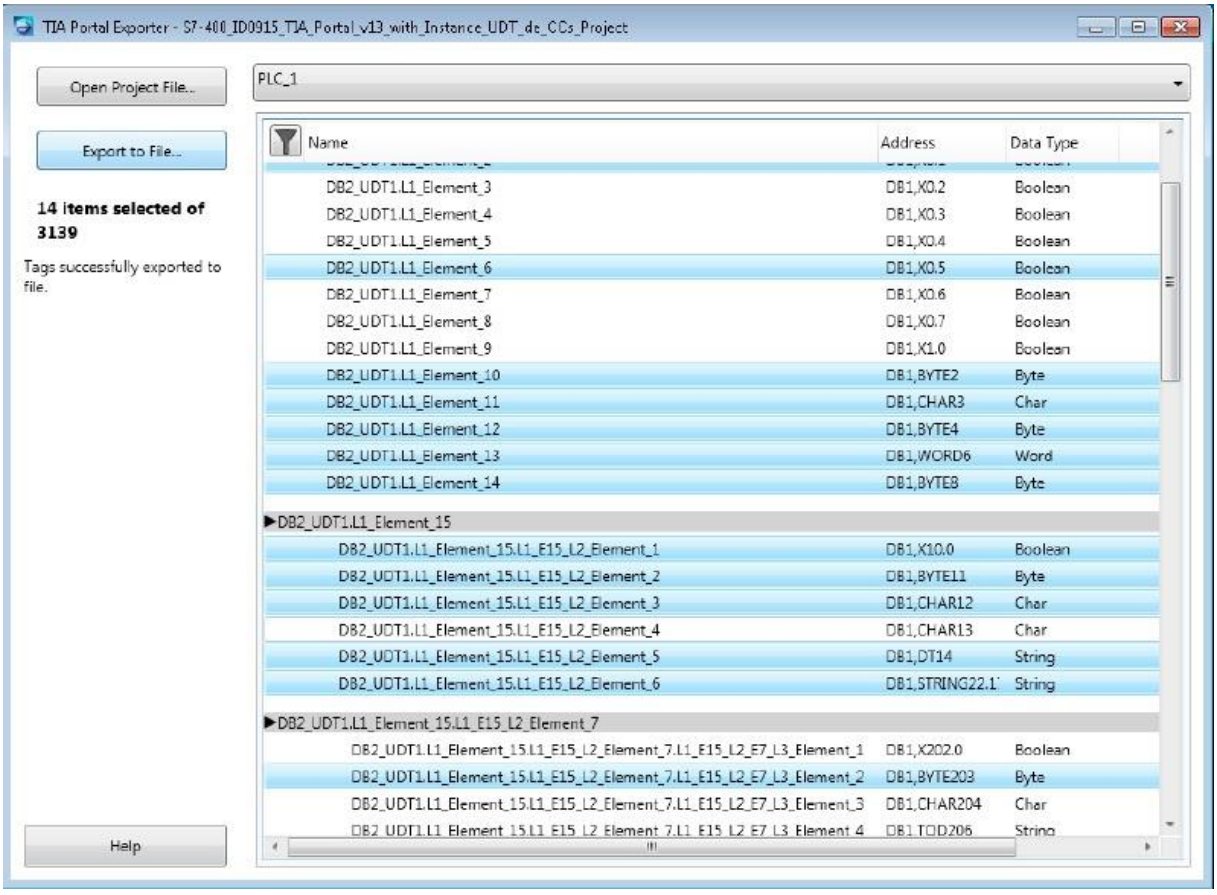
Obr. 2.6 – Zobrazení vlastností vytvořeného zařízení

Nyní je možné vytvořit skupinu, ve které se následně vytvoří tzv. tagy, potřebné pro komunikaci mezi dvěma zařízeními. Toho lze docílit kliknutím pravým tlačítkem na vytvořené zařízení, pojmenované S7 – 1200 a výběrem *New Tag Group*. Vytváření nových tagů se provádí stejným způsobem. Na obr. 2.7 jsou znázorněné tagy, které jsou potřeba pro řízení semaforů, včetně zobrazení vlastností jednoho tagu. Aby vytvořené tagy mohly přijímat data z PLC, je nutné, aby byly deklarovány na stejnou adresu a měly stejný datový typ. Pojmenování tagů stejně jako v PLC není nutné. Defaultní nastavení rychlosti snímání aktuálního stavu je 100 ms, což je pro účely této úlohy dostačující.



Obr. 2.7 – Zobrazení tagů včetně nastavených vlastností

Výše zmíněným způsobem byly nadefinovány tagy manuálně. Pokud by se však jednalo o robustnější úlohu obsahující stovky, nebo i tisíce tagů, bylo by přepisování tagů velice zdlouhavé a přehlédnutí v adresaci proměnných snadné. KEPServerEX nabízí možnost automatického generování tagů z TIA Portal. K tomu je zapotřebí ve vlastnostech vytvořeného zařízení v záložce *Tag Generation* mít povolené generování tagů a dále v záložce *Tag Import* mít nastavenou cílovou adresu pro soubor *TIA Portal Exporter file*. Tento soubor, který obsahuje tagy vytvořené v TIA Portal, je generován pomocí nástroje *TIA Portal Exporter*, který je součástí softwaru KEPServerEX. Na obr. 2.8 je možné vidět ukázku výpisu tagů z PLC pomocí nástroje *TIA Portal Exporter*.



Obr. 2.7 – Ukázka TIA Portal Exporter (KEPServerEX Version 6.2, 2017)

2.4 Komunikace MATLABu s OPC serverem

K propojení mezi OPC serverem a prostředím MATLAB je zapotřebí vytvořit komunikaci, která bude sloužit k výměně dat. Aby bylo možné jednoduše přistupovat k datům získaným z OPC serveru, je k tomuto účelu vytvořena v MATLABu třída, která se zároveň stará o připojení k OPC serveru. Do konstruktoru třídy jsou zadány proměnné, které chceme použít pro získaná data a dále je zde vytvořena komunikace pro připojení k serveru. MATLAB má implementovaný toolbox, který je potřeba zkontrolovat, zda je nainstalován. Poté je možné se připojit k softwaru KEPServerEX, pomocí příkazu *opcda*. Aby došlo k úspěšnému propojení, je nutné zadat IP adresu místní sítě a název softwaru. Dále je nutné zadat přesný název a umístění, kde se nachází tagy, které chceme načíst do vytvořených proměnných. Nejdříve se přidá skupina, ve které se tagy nachází, pomocí příkazu *addgroup* a názvu skupiny. Nyní je možné pomocí příkazu *additem* tagy přidat. Na obr. 2.8 je možné vidět deklaraci proměnných včetně konstruktoru a připojení k OPC.

```
classdef MatlabPLC < handle
    properties
        WE_RG;
        SN_RG;
        WE_Y;
        SN_Y;
       OPCdata;
       OPCserver; %OPC server
        end
       methods
           function this = MatlabPLC()
               this.WE_RG = false;
               this.SN_RG = false;
               this.WE_Y = false;
               this.SN_Y = false;
               if mpccheckboxinstalled('opc')
                   opcreset
                   clear mpcopcPlatnStep;
                   clear mpcopcMPCstep;
                   try
                       this.OPCserver = opcda('192.168.0.101','Kepware.KEPServerEX.V6');
                       connect(this.OPCserver)
                       catch ME
                       disp('The Kepware.KEPServerEX.V6 must be running on the local machine.')
                       return
                   end
               end
               semafory = addgroup(this.OPCserver);
               this.OPCdata.WE_RG = additem(semafory,'SIEMENS.S7-1200.semafory.WE_RG');
               this.OPCdata.SN_RG = additem(semafory,'SIEMENS.S7-1200.semafory.SN_RG');
               this.OPCdata.WE_Y  = additem(semafory,'SIEMENS.S7-1200.semafory.WE_Y');
               this.OPCdata.SN_Y  = additem(semafory,'SIEMENS.S7-1200.semafory.SN_Y');
           end
           function delete(this)
               %destruktor
               disconnect(this.OPCserver);
         end
```

Obr. 2.8 – Zdrojový kód třídy MatlabPLC – konstruktor

K načtení hodnot z OPC serveru je zapotřebí ještě přečtení těchto dat a uložení do vytvořených proměnných v konstruktoru. K tomu jsou využívány gettery, které je možné vidět na obr. 2.9

```
% WEST - EAST (RG,Y - GETTER)
        function WE = getWE(this)
            pom = read(this.OPCdata.WE_RG);
            this.WE_RG = pom.Value;
            WE = this.WE_RG;
        end
         function WE_Y = getWE_Y(this)
            pom = read(this.OPCdata.WE_Y);
            this.WE_Y = pom.Value;
            WE_Y = this.WE_Y;
        end
% SOUTH - NORTH (RG,Y - GETTER)
            function SN = getSN(this)
            pom = read(this.OPCdata.SN_RG);
            this.SN_RG = pom.Value;
            SN = this.SN_RG;
            end
            function SN_Y = getSN_Y(this)
            pom = read(this.OPCdata.SN_Y);
            this.SN_Y = pom.Value;
            SN_Y = this.SN_Y;
            end
```

Obr. 2.9 – Zdrojový kód třídy MatlabPLC – gettery

2.5 Řízení vozidel

Pro každý ze čtyř směrů křižovatky byla vytvořena třída, charakterizující vlastnosti, které každé vozidlo musí obsahovat. Mezi základní vlastnosti patří:

- Aktuální poloha X,
- aktuální poloha Y,
- povolení/zakázání pohybu.

Při deklaraci nového objektu (vozidla) je třídou zajištěno náhodné vybrání jednoho ze čtyř barev vozidel a následné vložení na začátek silnice. Na obr. 2.10. je možné vidět třídu *Move_LR* včetně nastavení vlastností a konstruktoru deklarující počáteční stavy.

```
classdef Move_LR < handle
 properties
    CurrentY; % aktuální poloha Y
    CurrentX; % aktuální poloha X
    Stop;     % false == pohyb povolen / true == pohyb zakázán
    Start;    % true == auto na silnici / false == auto není na silnici
 end
 properties (Access = private)
    vehicles; %seznam modelů aut
    vehicle;  %vybrané auto
 end
 methods
     function [this] = Move_LR() %konstruktor - deklarace počátačních stavů
         this.CurrentY = 487 + (493-487)*rand(1,1);
         this.CurrentX = (-66);
         this.Start = false;
         this.Stop = true;
         global handles       %globální proměnná handles pro přístup ke GUI ze všech tříd
         this.vehicles = dir(['C:\Users\Miroslav\Desktop\Crossroad\vehicles' '/*.png']); %seznam aut
         this.vehicle =imagesc(handles.axes8,imrotate(imread(fullfile('C:\Users\Miroslav\Desktop\Crossroad\vehicles',...
                               this.vehicles(randi(4)).name)),270),'XData',this.CurrentX(),'YData',this.CurrentY());
     end
```

Obr. 2.10 – Zdrojový kód třídy Move_LR

Poslední dva řádky konstruktoru popisují vytvoření seznamu vozidel. Dále dojde k náhodnému vybrání jednoho vozidla a vložení jej do GUI s nastavenými X,Y souřadnicemi. Vlastnosti *vehicles a vehicle* mají nastavený přístup pouze z aktuální třídy (*Access private)*, protože slouží pouze k vybrání typu vozidla a nikde, než v dané třídě, s nimi není manipulováno.

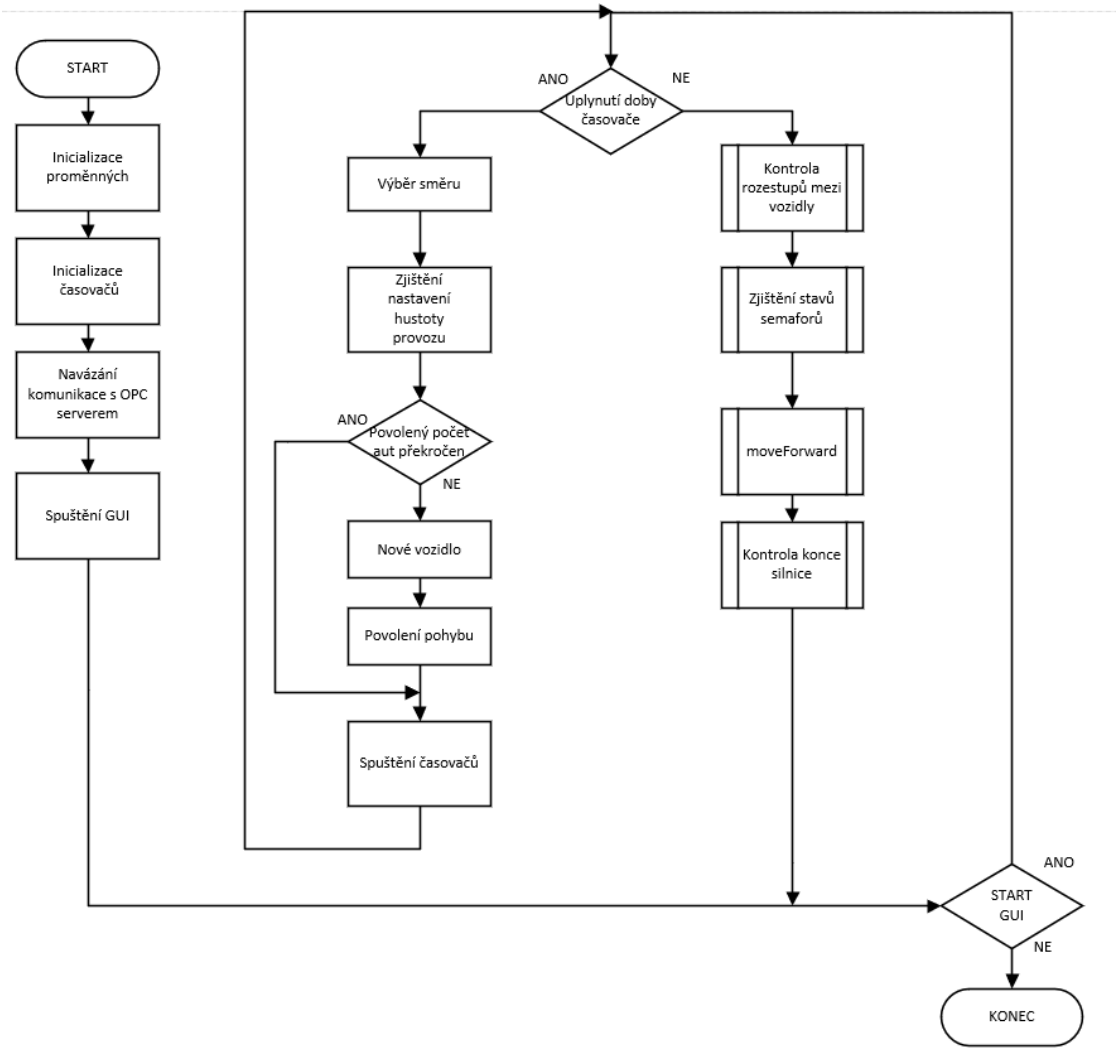
Uvnitř třídy je dále funkce, která zajišťuje pohyb vozidla vpřed a aktualizaci polohy v GUI. Tuto funkci je možné vidět na obr. 2.11.

```
function moveForward(this)
 if  this.Stop() == false && this.Start() == true %jsou-li splněny podmínky pro pohyb
     this.CurrentX = this.CurrentX + 2;           %pohyb vpřed
     set(this.vehicle,'XData',this.CurrentX());   %aktulizace souřadnic
 end
 end
```

Obr. 2.11 – Zdrojový kód funkce moveForward

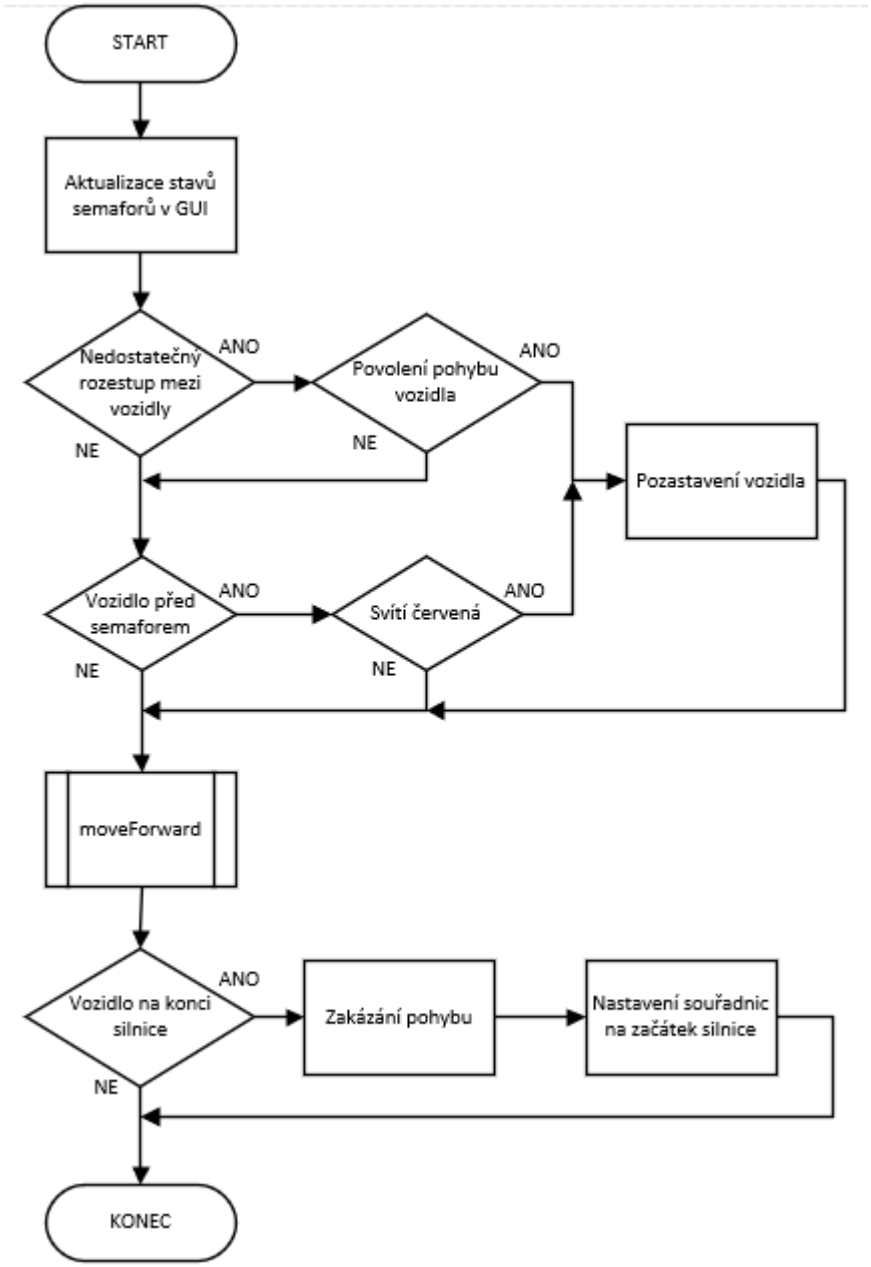
Popis funkce řídicího programu

Nyní je možné vytvořit řídicí program, pomocí kterého bude řízen provoz na křižovatce. Po inicializaci proměnných a zpuštění uživatelského rozhraní začíná simulace. Program je vykonáván v nekonečné smyčce, kde jsou průběžně aktualizovány souřadnice jednotlivých vozidel a také stavy semaforů. Vývojový diagram je možné vidět na obr. 2.12.



Obr. 2.12 – Vývojový diagram celého programu

Významnou složkou v programu je funkce tří časovačů, které určují, zda dojde k poslání nového vozidla na silnici nebo ne. Tyto tři časovače reprezentují nízkou, střední a vysokou hustotu provozu a vždy, když dojde k uplynutí času jednoho z nich, dojde k přidání nového auta do směru, který má tuto hustotu provozu zvolenou. V době, kdy jsou všechny časovače aktivní, je prováděna kontrola rozestupů mezi vozidly, kontrola stavů semaforů a při následném volání funkce z třídy (*moveForward)* je vozidlo případně pozastaveno, pokud je to potřeba. To je možné vidět na obr. 2.13 prostřednictvím vývojového diagramu.



Obr. 2.13 – Vývojový diagram pro řízení vozidel

Pokud mají vozidla jedoucí za sebou nedostatečný rozestup a zároveň byly již vpuštěny na silnici (mají povolení pohybu), je pohyb druhého vozidla pozastaven na dobu, dokud tato vzdálenost nebude opět dostačující. Vozidlo může být také pozastaveno, pokud stojí před semaforem, což je možné vidět na zdrojovém kódu (obr. 2.14).

```
if LR(n).getCurrentX() == (200) %pokud stojíš na semaforu
    if com.getWE() == false     %svítí červená?
        LR(n).Stop = true;      %pozastavení
    else                        %svítí zelená
        LR(n).Stop = false;     %pohyb povolen
    end
end
```

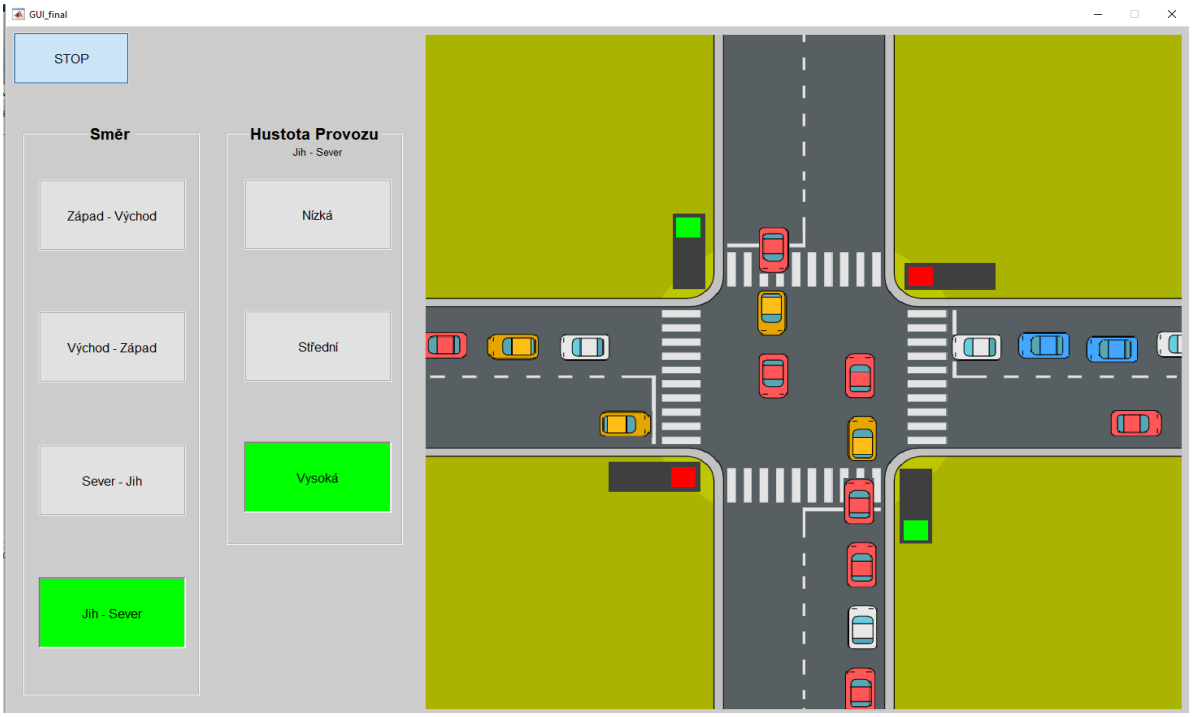
Obr. 2.14 – Zdrojový kód pro kontrolu stavu před semaforem

2.6 Uživatelské rozhraní

Vizualizace křižovatky je vytvořena v prostředí MATLAB pomocí nástroje GUIDE, který umožňuje jednoduché vytváření GUI (Graphical User Interface), neboli grafického uživatelského rozhraní. GUIDE je možné vytvořit zapsáním příkazu *guide* do příkazového řádku MATLABu a funguje na principu vkládání již vytvořených objektů na obrazovku, což výrazně zjednodušuje práci a není nutné vše ručně programovat. Při vložení nového objektu je automaticky vygenerována funkce (*callback function*), která tento objekt reprezentuje. Vytvoří se v novém, automaticky vygenerovaném souboru s příponou .m, ve kterém je možné dohledat každou část aplikace a programově ji upravit podle představ. Aby vytvořená aplikace byla interaktivní a uživatel ji měl možnost ovládat, je možné pomocí *callback function* měnit vlastnosti vytvořených objektů, jako např. změna barvy tlačítka po jeho stisknutí.

Pro potřeby této práce byla vytvořena vizualizace křižovatky, kterou můžeme vidět na obr. 2.15 se čtyřmi směry jízdy, pojmenované následujícím způsobem:

- Západ – Východ,
- Východ – Západ,
- Sever – Jih,
- Jih – Sever.



Obr. 2.15 – Model křižovatky

U každého ze čtyř směrů je možné ovládat hustotu provozu. Na výběr je mezi nízkou, střední a vysokou hustotou. Aby bylo možné hustotu změnit v určitém směru, je zapotřebí vybrat v panelu *Směr* požadovanou silnici. Ta se po stisknutí zabarví zeleně a zobrazí se panel *Hustota Provozu* pro směr, který byl zvolen. Hustotu provozu je možné měnit i v průběhu simulace. Při přepnutí směru silnice dochází k automatickému odkrytí panelu požadovaného směru a zakrytí panelů všech ostatních směrů. Princip přepínání mezi směry je znázorněn na zdrojovém kódu (obr 2.16), včetně komentářů. Takto je naprogramován i zbytek směrů včetně hustot provozu.

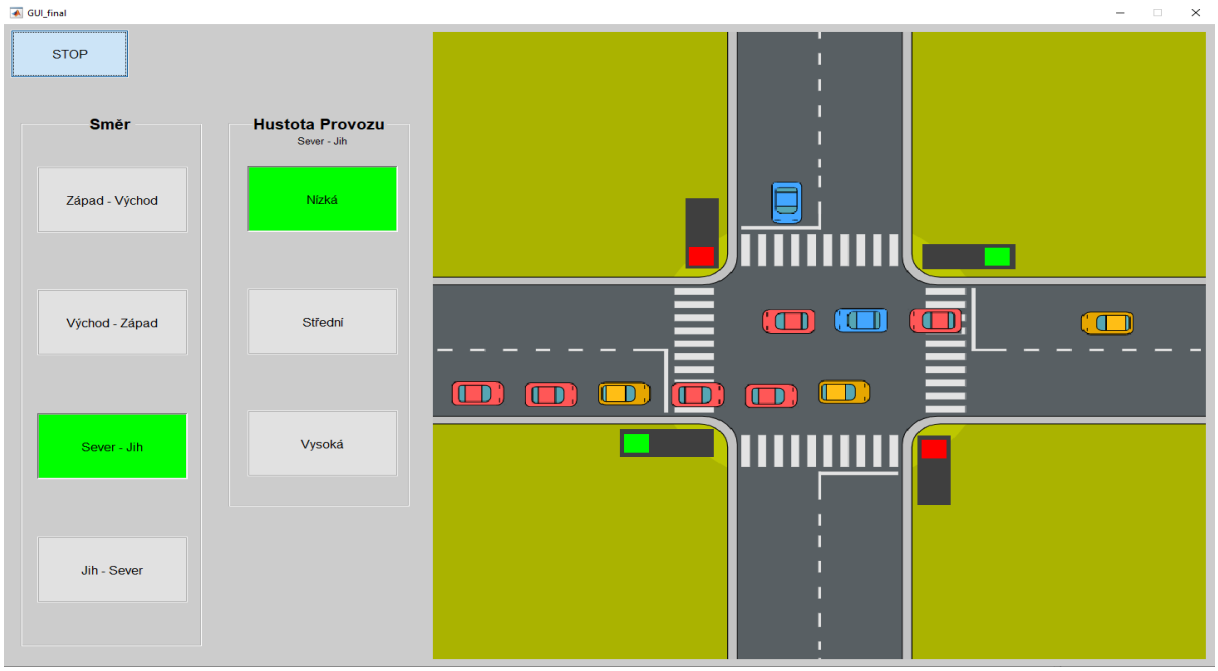
```
function TB_ZV_Callback(hObject, eventdata, handles)
smer_ZV = get(hObject, 'Value');                        %handler(ukazatel) na objekt

if smer_ZV                                              %pokud došlo ke stisknutí(smer_ZV == true)
 set(handles.TB_VZ, 'Value', 0);                        %nastavení ostatních tlačítek na false
 TB_VZ_Callback(handles.TB_VZ, eventdata, handles)      %volán callback dané funkce, aby došlo ke změně stavu

 set(handles.TB_SJ, 'Value', 0);                        %nastavení ostatních tlačítek na false
 TB_SJ_Callback(handles.TB_SJ, eventdata, handles)      %volán callback dané funkce, aby došlo ke změně stavu

 set(handles.TB_JS, 'Value', 0);                        %nastavení ostatních tlačítek na false
 TB_JS_Callback(handles.TB_JS, eventdata, handles)      %volán callback dané funkce, aby došlo ke změně stavu

 set(handles.TB_ZV,'Backgroundcolor','g');              %nastavení barvy tlačítka
 %zapnutí viditelnosti panelu tlačítek pro nastavení hustoty provozu
 set(handles.PANEL_ZV, 'Visible', 'on')
else                                                    %pokud nedošlo ke stusknutí (smer_ZV == false)
 set(handles.TB_ZV,'Backgroundcolor',[0.94, 0.94, 0.94]); %nastavení barvu tlačítka na defaultní
 %vypnnutí viditelnosti panelu tlačítek pro nastavení hustoty provozu
 set(handles.PANEL_ZV, 'Visible', 'off')
end
```

Obr. 2.16 – Zdrojový kód pro přepínání řízení směru

2.7 Popis průběhu simulace

Simulace vozidel je spuštěna, když je vybrána velikost hustoty provozu v daném směru. Pokud není zvolena žádná ze tří možností, vozidla v daném směru nejezdí. Program v MATLABu je napsán tak, aby počet vozidel jedoucích v daném směru, bylo možné jednoduše upravit na libovolný počet. Vzhledem však k rozměrům křižovatky v porovnání s rozměry vozidel byl povolený počet v jednom směru nastaven na 8 vozidel. Při nastavení vysoké hustoty provozu je vozidel na silnici prakticky neustále nejvyšší možný počet a lze tento stav chápat jako dopravní špičku. Nastavení sekvence semaforů pak umožňuje projet až čtyřem vozidlům najednou, poté dojde ke změně stavu. Při vysoké hustotě provozu tedy nikdy nedojde ke stavu, kdy by se před semaforem netvořily kolony a provoz byl plynulý.

Při nastavení střední hustoty provozu dochází k výrazně pomalejšímu vkládání vozidel na silnici. Zda se budou před semaforem tvořit kolony či nikoliv, však záleží na předchozí hustotě provozu. Pokud v daném směru dojde k přepnutí z vysoké hustoty provozu do střední, potrvá určitý čas, než se provoz ustálí. Za ustálených podmínek by se však neměly před semaforem tvořit kolony více než tří vozidel. Toto nastavení lze považovat za běžný denní provoz. Při výběru nejnižší možné hustoty provozu jsou v daném směru vždy maximálně dvě vozidla a kolony před semaforem se prakticky žádné netvoří. Takto může vypadat např. noční provoz, kdy je vozidel nejméně. Na obr. 2.7 je možné vidět simulaci všech tří nastavení, kde směr Západ – Východ reprezentuje vysokou hustotu provozu, směr Východ – Západ střední hustotu provozu a směr Sever – Jih nízkou hustotu provozu.



Obr. 2.17 – Ukázka jednotlivých hustot provozu

Možnosti rozšíření úlohy

Model dopravního uzlu může mít mnoho dalších rozšíření, které by k této úloze mohly být přidány. Je například možné přidat ovládání přechodů pro chodce, kdy by z vizualizační části byl poslán požadavek řídicímu systému na odklonění dopravy tak, aby v daném směru bylo umožněno přejít chodcům. Dále by bylo možné rozšířit logiku řídicího systému na tzv. „inteligentní semafory“, kdy by docházelo k měření hustoty dopravy na základě počtu vozidel vyskytujících se před semaforem. Z těchto dat by řídicí systém určoval sám, jaké směry budou prioritně pouštěny, aby došlo k co nejplynulejšímu provozu ve všech směrech.

3 ZÁVĚR

Cílem této bakalářské práce byl návrh řídicího systému pro ovládání dopravního uzlu, dále vytvoření vizualizace křižovatky, simulace projíždějících vozidel a umožnění uživateli ovládat hustotu provozu v jednotlivých směrech.

V teoretické části byl kladen důraz na stručný popis všech komponent, které byly potřeba pro dokončení práce. Byly zde vysvětleny pojmy jako programovatelný automat, princip činnosti vykonávání programu a dále podrobnější popis programovatelných automatů od společnosti Siemens, které byly k této práci použity. Dále zde byly stručně popsány jednotlivé softwary, které byly použity a nakonec princip OPC serveru, který zprostředkovává komunikaci mezi řídicím systémem a vizualizací.

V praktické části je dále podrobně popsáno, jak práce vznikala. Je zde popsána konfigurace fyzického zařízení, včetně postupu nastavení a zprovoznění komunikace mezi PLC a softwarem, ve kterém byl vytvořen řídicí program (TIA Portal). V této části jsem se nesetkal s žádnými většími problémy. V praktické části je dále vysvětleno vytvoření a nastavení OPC serveru. Software KEPServerEX má velmi kvalitně zpracovaný manuál, pomocí kterého jsem postupoval při nastavení OPC serveru. K vytvoření uživatelského prostředí, včetně simulace vozidel v prostředí MATLAB bylo využíváno online podpory MATLAB Central, kde je možné dohledat řešené příklady a návody pro práci s programem.

Budoucí vývoj práce je možné směřovat k rozšiřování modelu dopravního uzlu o další uživatelské funkce, jako je ovládání přechodu pro chodce, přidání dalších režimů řízení semaforů, nebo zdokonalení simulace vozidel.

4 POUŽITÁ LITERATURA

ŠVARC, Ivan, 2002. *ZÁKLADY AUTOMATIZACE* [online]. Brno: CERM, [cit. 2020-04-21]. Dostupné z: http://media1.wgz.cz/files/media1:5100dca403912.pdf.upl/ZakladyAutomatizace.pdf

KOZIOREK, Jiří a Libor Chromčák, 2008. LOGICKÉ SYSTÉMY ŘÍZENÍ: Programovatelný automat – technické vybavení. *Logické systémy řízení a programovatelné automaty* [online]. Ostrava: Vysoká škola báňská - Technická univerzita [cit. 2020-04-21]. ISBN 978-80-248-1490-2. Dostupné z: http://www.elearn.vsb.cz/archivcd/FEI/LSA/Logicke%20systemy%20rizeni.pdf

ŠIMRAL, Jaroslav, 2011. Tepelné zásobování v Mostě s využitím řídicího systému Metasys. Bakalářská práce. [online]. Ostrava: Vysoká škola báňská, Technická univerzita Ostrava, Hornicko-geologická fakulta, Institut ekonomiky a řízení. Vedoucí práce Oldřich Kodym. [cit. 2020-04-22]. Dostupné z: https://dspace.vsb.cz/bitstream/handle/10084/86881/SIM514_HGF_B2102_6209R013_2011.pdf?sequence=1.

SIEMENS: S7-1200 Easy Book Manual [online], 2015. NÜRNBERG: Siemens [cit. 2020-04-25]. Dostupné z: https://euroec.by/assets/files/siemens/s71200_easy_book_en-US_en-US.pdf

RAKUŠAN, Ondřej, 2012. AUTOMA: Simatic S7-1200 – brána do světa automatizace budoucnosti [online]. Siemens [cit. 2020-04-27]. Dostupné z:https://automa.cz/cz/casopis-clanky/simatic-s7-1200-brana-do-sveta-automatizace-budoucnosti-2012_12_0_10064/

DRAHOŠ, Peter a Juraj GABRIEL, 2006. AUTOMA: Komunikačný systém Profinet IO [online]. Fakulta elektrotechniky a informatiky STU Bratislava [cit. 2020-04-27]. Dostupné z:https://automa.cz/cz/casopis-clanky/komunikacny-system-profinet-io-2006_07_31231_560/

AUTOMA: Profinet versus Profibus [online], 2012. Murrelektronik CZ [cit. 2020-04-27]. Dostupné z:https://automa.cz/cz/casopis-clanky/profinet-versus-profibus-2012_05_0_9618/

AUTOMA: Siemens TIA Portal – jednotné vývojové prostředí pro automatizaci v průmyslu [online], 2011. Siemens [cit. 2020-04-27]. Dostupné z:https://automa.cz/cz/casopis-clanky/siemens-tia-portal-jednotne-vyvojove-prostredi-pro-automatizaci-v-prumyslu-2011_03_43212_6058/

Introduction to Matlab, *Educba.com* [online]. [cit. 2020-04-28]. Dostupné z: https://www.educba.com/introduction-to-matlab/

STIANKO, Martin a Jaromír PETERKA, 2004. *AUTOMA: OPC v průmyslové komunikaci* [online]. Liberec: Fakulta strojní, katedra aplikované kybernetiky TUL Liberec [cit. 2020-04-29]. Dostupné z: https://automa.cz/cz/casopis-clanky/opc-v-prumyslove-komunikaci-2004_06_32378_2345/

Foxon.cz: CO JE OPC? OPC SERVER? OPC KLIENT? [online], [cit. 2020-04-29]. Dostupné z: https://www.foxon.cz/blog/prakticka-teorie/159-co-je-opc-opc-server-opc-klient

Foxon.cz: CO JE OPC TAG? (OPC GROUP A OPC SERVER BROWSING) [online], [cit. 2020-04-29]. Dostupné z: https://www.foxon.cz/blog/prakticka-teorie/101-co-je-opc-tag-opc-group-a-opc-server-browsing

KEPServerEX Version 6.2, 2017. In: *Opcturkey* [online]. ASP Otomasyon, 8 June 2017 [cit. 2020-05-02]. Dostupné z: http://www.opcturkey.com/kepserverex-version-6-2-now-available