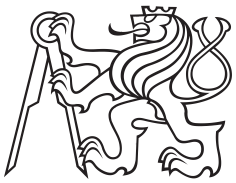
ČESKÉ VYSOKÉ
UČENÍ TECHNICKÉ
V PRAZE

FAKULTA
STROJNÍ

DIPLOMOVÁ PRÁCE

2020

PATRIK
DOLEŽAL



České
vysoké
učení technické
v Praze

F2

Fakulta strojní
Ústav přístrojové a řídící techniky
Odbor automatického řízení a inženýrské informatiky

Navrhování distribuovaných řídicích systémů

DIPLOMOVÁ PRÁCE

Bc. Patrik Doležal

Vedoucí práce: Ing. Mgr. Jakub Jura, Ph.D.
Studijní obor: Přístrojová a řídící technika
Září 2020

ZADÁNÍ DIPLOMOVÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Doležal** Jméno: **Patrik** Osobní číslo: **457665**

Fakulta/ústav: **Fakulta strojní**

Zadávající katedra/ústav: **Ústav přístrojové a řídicí techniky**

Studijní program: **Strojní inženýrství**

Studijní obor: **Přístrojová a řídicí technika**

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

Navrhování distribuovaných řídicích systémů

Název diplomové práce anglicky:

Design of distributed control systems

Pokyny pro vypracování:

- rešerše používaných způsobů programování distribuovaných úloh
- pro analýzu, popis a organizaci práce použít jazyk UML
- sestavit v laboratoři průmyslových automatů modelovou úlohu distribuovaného řídicího systému, která splňuje podmínky rekonfigurovatelnosti výroby v duchu vize Industry 4.0.

Seznam doporučené literatury:

1. V. Mařík, O. Štěpánková, and J. Lažanský, Umělá inteligence. Praha: Academia, 1997.
2. V. Mařík, O. Štěpánková, and J. Lažanský, Umělá inteligence. Praha: Academia, 2001.
3. V. Mařík, O. Štěpánková, and J. Lažanský, Umělá inteligence. Praha: Academia, 2003.
4. Automatizace: časopis pro automatizaci, měření a inženýrskou informatiku. Vydává: Automatizace, Praha, 1958-2010.

Jméno a pracoviště vedoucí(ho) diplomové práce:

Ing. Mgr. Jakub Jura, Ph.D., U12110.3

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

Datum zadání diplomové práce: **30.04.2020** Termín odevzdání diplomové práce: **07.08.2020**

Platnost zadání diplomové práce: _____

Ing. Mgr. Jakub Jura, Ph.D. podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Ing. Michael Valášek, DrSc. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

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

. _____ Datum převzetí zadání

_____ Podpis studenta

Poděkování

Děkuji všem lidem kolem mě, kteří mě podporovali při psaní této diplomové práce. Osobně bych pak chtěl poděkovat mému vedoucímu Mgr. Ing. Jakubovi Jurovi Ph.D., za jeho čas, rady a připomínky.

Bc.Patrik Doležal

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně s tím, že její výsledky mohou být dále použity podle uvážení vedoucího diplomové práce jako jejího spoluautora. Souhlasím také s případnou publikací výsledků diplomové práce nebo její podstatné části, pokud budu uveden jako její spoluautor.

V Praze dne 25. srpna 2020

_____

Bc. Patrik Doležal

Abstrakt

Cílem této práce je navrhnout distribuovaný řídicí systém za použití PLC. V práci je nejdříve rešerše na téma řídicích systémů centralizovaných, distribuovaných a následně systémů multiagentních a holonických. Rešerše obsahuje i možnosti komunikace v řídicích systémech a to jak mezi jednotlivými PLC, tak i na ostatních úrovních řídicích systémů. V Praktické části je analýza vytvářeného systému, jeho navržení, vytvoření softwaru pro generování PLC kódu využitím SFC v Javě a jeho realizace včetně vytvoření funkčního programu pro PLC a SCADA systému.

Klíčová slova: PLC, distribuované systémy, generování kódu, UML, SCADA, Java

Vedoucí práce: Ing. Mgr. Jakub Jura, Ph.D.
Ústav Přístrojové a řídící techniky
Odbor automatického řízení a inženýrské informatiky

Abstract

The aim of this thesis is to design distributed control system using PLCs. First part is research on the topic of centralized and distributed control systems a then multiagent and holonic control systems. The research also contains communication options between individual PLCs and at other levels of control systems. The practical part contains the analysis of created control system, its design, the creation of software for generating PLC code using SFC in Java and its implementation, including the creation of a functional program for PLC and SCADA system.

Keywords: PLC, distributed control system, code generating, UML, SCADA, Java

Title translation: Design of distributed control systems

Obsah

Obrázky

Tabulky

Kapitola 1

Úvod

Stále více se mluví o čtvrté průmyslové revoluci, která by měla přinést úplnou automatizaci výrobních procesů. Tato revoluce sice předpokládá celospolečenské dopady, ale v jejích středu stojí průmyslová výroba. Řídicí systémy se tak stávají středem pozornosti, protože jsou základem celé výroby.

Cílem této práce je sestavit distribuovaný řídicí systém, který splňuje podmínky rekonfigurovatelnosti výroby v souladu s principy čtvrté průmyslové revoluce.

Za tímto účelem je třeba se nejdříve seznámit s řídicími systémy (PLC) a jejich architekturami (jak centralizovanými, tak distribuovanými) a zjistit jejich vhodnost pro dané výrobní procesy. Speciální pozornost je třeba věnovat těm procesům, jejichž automatizace (a tedy nahrazení lidské obsluhy) vyžaduje nasazení umělé inteligence (například v oblasti percepční kontroly procesu operátorem). Nesmí se opomenout možnosti komunikace v takových systémech, protože hlavním předpokladem k bezchybnému chodu výroby jen rychlé a zejména bezchybné předávání informací.

V praktické části je za účelem splnění cílů této práce potřeba nejdříve provést analýzu výrobního procesu. Na tuto pasáž se nabízí využití lexikální analýzy (OMT) jejímž výstupem je UML Class diagram. Pro splnění maximální rekonfigurovatelnosti systému je nosnou myšlenkou generování kódu pro PLC v jazyce dle IEC 61131-3. Generátor kódu využívá SFC a je naprogramován v Javě s využitím grafického editoru SceneBuilder. Je navržena distribuovaná architektura. Jsou využity řídicí systémy Siemens S7-1200 a miniPLC Siemens LOGO! v režimu vzdálených vstupů a výstupů. Pro konfiguraci komunikace jsou využity proprietální prostředy Siemens.

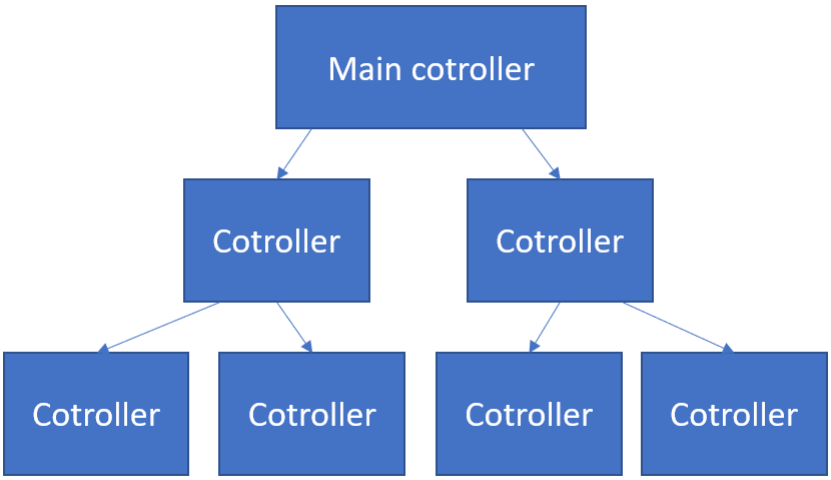
Kapitola 2

Teoretická studie řídicích systémů a nástrojů k jejich analýze

2.1 Centralizovaný řídicí systém

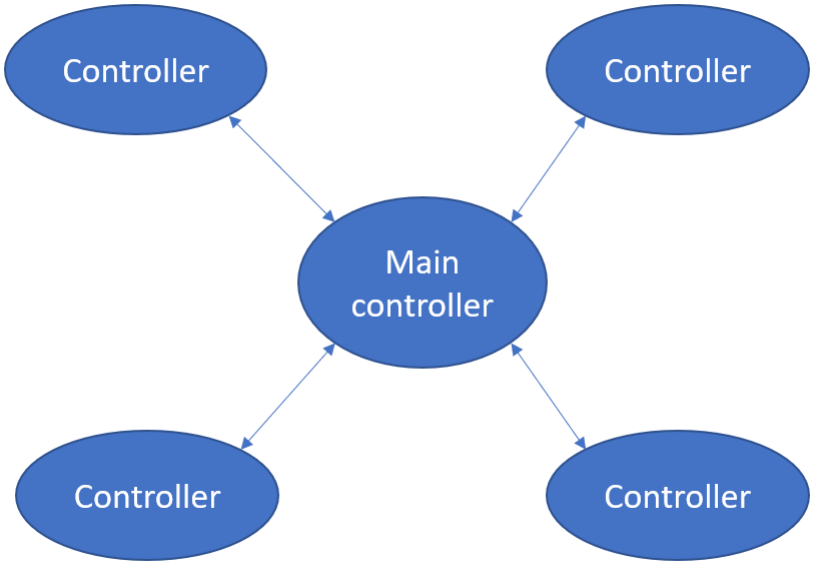
Centralizovaný řídicí systém (dále jen CCS) je systém, který využívá k řízení celého procesu pouze jeden centrální člen. Rozlišují se architektury *call-return* a *manager*, které se liší ve způsobu komunikace centrálního členu s ostatními komponentami.

The call-return model je hierarchicky uspořádaná architektura, ve které se řídicí člen vždy nachází na jejím vrcholu a rozesílá pokyny do nižší úrovně, kde se nachází 2 nebo více členů. Po přijmutí pokynů předají tito členové další pokyny do nižší úrovně a tento proces se opakuje dokud nedostanou instrukce koncová zařízení, tedy nejnižší úroveň této architektury.[1]



Obrázek 2.1: The call-return model

The manager model využívá hlavní řídicí člen jako manažera, který předává pokyny přímo dotyčným členům. Tato architektura tedy nevyužívá žádné mezičleny mezi hlavní řídicím členem a koncovými zařízeními, což přináší svoje výhody i nevýhody. Výhodou je například rychlost, kterou se pohybují informace mezi manažerem a koncovým zařízením, nevýhodou je, že celý řídicí program musí být obsažen v hlavním řídicím členu. [1]



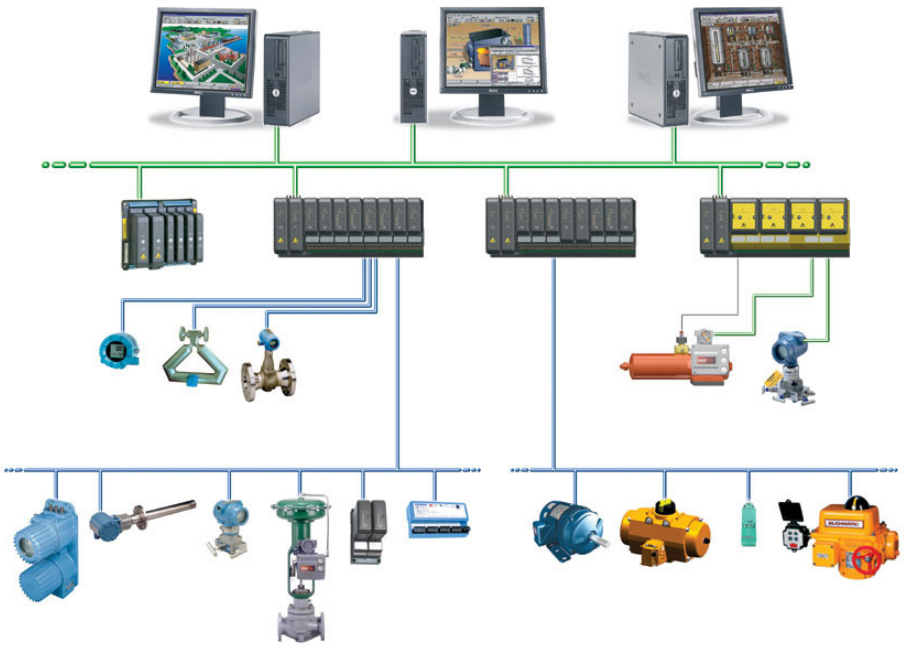
Obrázek 2.2: The manager model

2.2 Distribuovaný řídicí systém

Distribuovaný řídicí systém (dále jen DCS) je řídicí systém, který používá topograficky distribuované řídicí smyčky v celé továrně, stroji nebo kontrolní oblasti. Rozdíl oproti centralizovanému řídicímu systému, kde jeden kontrolér řídí sám všechny stroje, je možnost ke každé části výroby přiřadit vlastní kontrolér, který daný úsek kontroluje. DCS má rozmístěných po celé oblasti několik lokálních kontrolérů, které jsou propojeny dostatečně kapacitní komunikační sítí. Každý z nich pracuje autonomně, ale existuje centrální kontrolní dohled, který provádí dispečer. [2]

V minulosti se v DCS k řízení jednotlivých smyček používali pouze mikrokontrolery, protože programovatelné řídící automaty (PLC) měli velmi omezené možnosti komunikace, ale v současné době už jsou na takové úrovni, že se mikrokontrolerům vyrovnají. Mikrokontrolery se nicméně stále používají a během volby mezi nimi a PLC je třeba zvážit více aspektů dané výroby (jedním z důvodů je vyšší pořizovací cena PLC).

Jako příklad si uvedeme architekturu, která využívá 3 úrovní.



Obrázek 2.3: Příklad architektury DCS [3]

Úroveň 1
Nejnižší úroveň je úroveň polní instrumentace. Nachází se zde veškerá senzorika a akční členy. Vstupní (senzory) a výstupní (motory) zařízení jsou připojena ke vstupům a výstupům (I/O obvodům) řídicích jednotek (PLC).

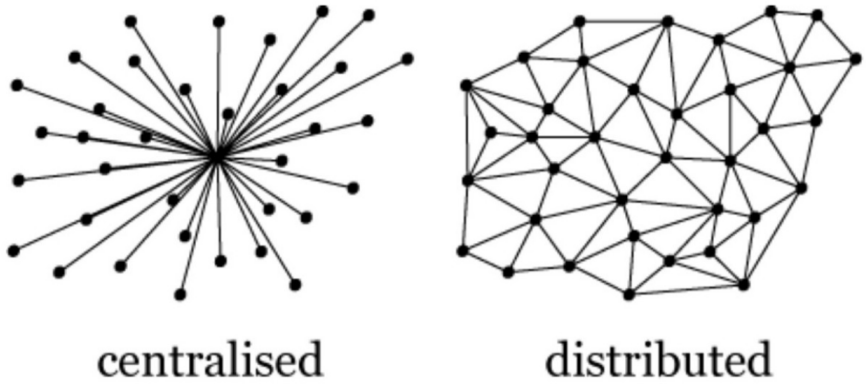
Úroveň 2
Toto je úroveň ovládání. Kontrolér přijímá data z průmyslové sběrnice a vykonává program, který v sobě má uložený. Každý kontrolér ovladá pouze k sobě připojená zařízení, nicméně může přijímat a odesílat údaje o jednotlivých zařízení ostatním kontrolérům, pokud je to pro jejich správné fungování nutné.

Úroveň 3
V nejvyšší úrovni nalezneme dispečerský dohled. Může zde být libovolný počet počítačů, umístěných většinou v kontrolní místnosti, kde dispečeři dohlíží jak na jednotlivé úseky výroby, tak na celý proces jako takový a mohou zde měnit potřebné parametry, pozastavit výrobu apod.

DCS může mít samozřejmě architekturu složenou z mnohem více úrovní, záleží zejména na velikosti jednotlivých částí a potřebě na ně dohlížet. Často bývá nějaká forma dohledu přímo u přístrojů pro lepší přehled pracovníků přímo se podílejících na výrobě nebo údržbě.

2.3 Srovnání využití centralizovaného a distribuovaného řídicího systému

Zda zvolit centralizované nebo distribuované řízení je otázka, se kterou se technici zabývají od počátku automatizačních systémů. Správně odpovědět je mnohem komplikovanější než to na první pohled vypadá a s vývojem technologií (zejména komunikačních) se může odpověď měnit. Rozhodně se nedá říci, že jeden typ je obecně lepší, než ten druhý. [4]



Obrázek 2.4: Schéma centralizovaného a distribuovaného systému [5]

Tato volba je základní částí návrhu, která definuje jak jsou kontrolované přístroje a operace propojeny se samotným řídicím systémem. Volba se odrazí ve všech vrstvách projektu a neměla by být zaměňována s výběrem použité platformy řídicího systému, ale měla by tento výběr řídit. [4]

Zaměřme se na proces. V prvním kroku návrhu je třeba vzít v potaz veškeré kontrolované vybavení a to jak z provozního, tak topologického hlediska. Z provozního hlediska jsou zařízení fungující relativně nezávisle na sobě dobrými kandidáty na distribuované řízení. Naopak zařízení a procesy vysoce propojené jsou obecně dobrými kandidáty na řízení centralizované. [4]

Podívejme se na příklad kandidáta na centralizované řízení. Jako příklad máme zařízení pro příjem tekutých přísad, které dodává tuto přísadu přes skupinu ventilů do nádrží. Systém musí ovládat 3 oblastí - přijímací zařízení, skupinu ventilů a skladovací zařízení. K provedení jakékoliv operace je zapotřebí všech 3 oblastí, což by u distribuovaného řídicího systému (tedy nezávislý kontrolér pro každou oblast) vedlo k vytvoření 3 potenciálních bodů selhání. Selhání kteréhokoliv z kontrolérů by kompletně ochromilo celý proces, kdežto pokud zde bude jeden společný kontrolér (tedy centralizovaný řídicí systém), tak zde tento bod bude pouze jeden. Použitím DCS by si navíc kontroléry musely neustále posílat velké množství informací, což je samozřejmě možné, nicméně to zvyšuje složitost systému a vytváří další potenciálně poruchové body. [4]

Řekněme, že tekutina byla ve zmiňované části rozdělena do tří nádrží a

na každou z těchto nádrží jsou napojeny jiné stroje pro přípravu různých produktů a každá z nich je na ostatních zcela nezávislá. Použití CCS by v tomto případě znamelo, že v případě výpadku kontroléru by nemohl být vyráběn ani jeden ze tří produktů a celá výroba by stála. Na druhou stranu s DCS je možné v případě výpadku jednoho kontroléru stále vyrábět zbylé 2 produkty, takže výhoda tohoto řešení je zřejmá. [4]

Dalším faktorem může být samotná instalace. Často bývají jednotlivé součásti procesu dodávány od různých dodavatelů a není vždy možné vše naplánovat na stejnou dobu, nebo může být už při pořízení jedné výrobní linky naplánováno rozšíření výroby v daném časovém horizontu. Je třeba brát v potaz, že i po implementování všech strojů je třeba vše otestovat, což v případě CCS znamená zastavit celý proces i v případě, že již funkční a otestovaná část vyrábí produkt. Naproti tomu v DCS může většina testů proběhnout bez zastavení již běžících kontrolérů a proto je třeba někdy zvážit jeho nasazení i tam, kde to z pohledu na proces není nezbytné. [4]

Je samozřejmě důležité zvážit i náklady spojené s řídicím systémem. V této oblasti jsou většinou ve výhodě CCS, protože jednoduše obsahuje méně kontrolérů, ale je třeba zvážit i případně pozastavení výroby, které bylo zmíněno v předchozím odstavci. [4]

2.4 Multiagentní systém

Multiagentní systém je systém, který je tvořen z decentralizovaných nebo distribuovaných inteligentních softwarových jednotek – inteligentních agentů. Snahou je, aby tyto autonomní jednotky byly schopné řešit určité problémy. Tento systém lze tedy popsat jako skupinu diskrétních volně propojených systémů (agentů), které spolupracují, aby dosáhli určitého společného cíle. [6]

Jaké má toto technické řešení výhody? Prakticky stejné jako týmová spolupráce. Díky možnosti paralelního postupu se zkrátí doba řešení, každý člen se zabývá pouze svou částí údajů a předává ostatním své závěry, takže se sníží nároky na komunikaci a vzhledem k možnosti týmu někoho přibrat, nebo se vzájemně zastoupit se zvyšuje operativnost a spolehlivost. [7]

Existuje spousta definic inteligentních agentů od různých autorů, pro lepší pochopení si některé uvedeme. Inteligentní agent (Farhoodi, Graham, 1996) je entita, která je zodpovědná za rozhodování, zda a jak reagovat na externí podněty. O agentovi platí, že:

- může posílat a přijímat informace od jiných agentů za použití vhodných protokolů,
- může zpracovávat přijaté informace a uvažovat o nich (tj. provádět odvozování, syntézu i analýzu),

- má soubor schopností provádět akce, které se mohou i dynamicky měnit; akce charakterizují úkoly, které dokáže agent provádět.

Inteligentní agent navíc dovede:

- uvažovat o svých schopnostech a o schopnostech ostatních agentů,
- generovat cíle nebo plány pro sebe a jiné agenty,
- účastnit se složitých interakcí s ostatními (např. za účelem vyjednávání a delegování úkolů), dynamicky se zapojovat do skupin či organizací (které mohou například sdružovat agenty s podobnou funkcí) nebo takové skupiny opouštět,
- získávat informace a používat jejich zdroje a
- udržovat explicitní modely důvěry pro sebe a ostatní agenty. [7]

Pojem agent (Wooldridge, Jennings, 1995) označuje systém, který je:

- autonomní, neboť agent pracuje bez přímého zásahu člověka a do jisté míry řídí své akce a vnitřní stav,
- sociální, protože agenti interagují s ostatními agenty (popř. s lidmi) prostřednictvím jazyka pro komunikaci agentů,
- reaktivní s ohledem na to, že agenti vnímají svoje okolí (fyzický svět, uživatele za grafickým uživatelským rozhraním, soubor jiných agentů, internet atd.) a reagují včas na změny v okolí,
- proaktivní, protože reakci agenta na podnět z okolí neurčuje pouze momentální stav, ale jeho chování je řízeno cíli v tomto stavu, že agent může sám převzít iniciativu při řešení. [7]

2.4.1 Základní charakteristiky agenta

- **Autonomnost** - agenti jsou samostatné na cíl orientované moduly, schopné samostatného řešení specifických úloh bez nutnosti komunikovat s okolím, schopnost komunikace je však zachována, mohou koordinovat činnosti a kooperovat s jinými agenty v rámci dané komunity. Mají možnost vstupovat do komunity, opouštět ji, poskytovat výsledky nebo o členství v komunitě požádat. [6]
- **Sociální chování** - agenti vytváří sociální vazby za účelem dosažení společných cílů, udržování informace o jiných agentech a vytváření úsudků o nich, sdružování do koalic a týmů, od nichž očekávají vzájemný prospěch. Nutnou podmínkou je schopnost upravovat model pohledu na okolní svět a to na základě obměňujících se vzájemných vazeb. [6]
- **Reaktivita** - agenti jsou aktivováni událostmi (obdoba událostního programování), schopni reakce v souladu s vnímáním reálného času. K dispozici má předem známou množinu akcí. Výběr obslužné akce je silně závislý na splnění podmínek, jak agent vnímá vnější svět. [6]

- **Intencionalita** - schopnost mít na paměti dlouhodobé cíle, rozklad problému na podproblém (strategie), organizace chování k dosahování těchto cílů, formulace vlastních plánů a využití svých úsudků v dalším rozhodování při dosahování cíle. Do volby cíle se promítá i osobní motivace agenta pro jeho dosažení. [6]

2.4.2 Typy Agentů

- **Agent inteligentní** - využívá metodu dedukce, má tedy schopnost plnit cíle s využitím vnitřní logiky. [6]
- **Agent reaktivní** - má schopnost reakce na podněty z okolí. [6]
- **Agent deliberativní** - rozvážný – má schopnost plánovat postup svých akcí, ovlivňovat prostředí, a to takovým způsobem, aby získal strategickou a taktickou výhodu. [6]
- **Agent kognitivní** - schopnost vyvozovat logické závěry z pozorování okolí, ukládání tohoto pozorování do interní báze znalostí. Taktéž sem patří schopnost mapování a vyhodnocování okolního prostoru (scény). [6]
- **Agent racionální** - má výše specifikované vlastnosti. Na základě získaných poznatků je schopen učit se a plánovat svou činnost s ohledem na dosažení známého cíle. [6]

2.4.3 Architektura agenta

- **Obal** - je zodpovědný za plánování a realizaci sociálních interakcí a tvoří jej komunikační vrstva a model sociálního chování [8]
- **Vlastní tělo** - nemá informace o komunitě [8]

2.4.4 Společné schopnosti agentů a jejich zájmy

Agenti musí mít společné vnímání pojmů, pravidla vzájemné komunikace a jazyk. [8]

Agenty lze dle jejich vzájemných zájmů dělit na:

- **Kooperativní** - mají společné cíle [8]
- **Kompetetivní** - mají protichůdné cíle [8]
- **Kolaborativní** - navzájem spolupracující [8]

2.4.5 Komunikace mezi agenty

Jeden agent může jiného agenta ovlivňovat buď přímo, což znamená s ním komunikovat, nebo nepřímo, kdy ovlivní jeho okolí a docílí tím změnu jeho postoje žádaným směrem. [8]

Během vzájemné komunikace může dojít k 6 různým druhům dialogu:

- **Dotazování** – agent se pokusí získat informaci tam, kde věří že je [8]
- **Hledání informace** – společné pátrání více agentů po informaci [8]
- **Přesvědčování** – snaha agenta o získání jiného ke splnění svého záměru [8]
- **Vyjednávání** - agenti spolu domlouvají podmínky sdílení prostředků nebo poskytnutí služeb tak, aby všechny strany dosáhli maximálního zisku [8]
- **Porada** - agenti poskytují potřebné znalosti a schopnosti a domlouvají se na dalším postupu s cílem nalezení řešení určitého problému, které je v zájmu všech [8]
- **Erestický dialog** - hádka za účelem dosažení svých záměrů [8]

Tyto dialogy se uskutečňují pomocí elementárních zpráv typu otázka, nabídka, zamítnutí a informování. [8]

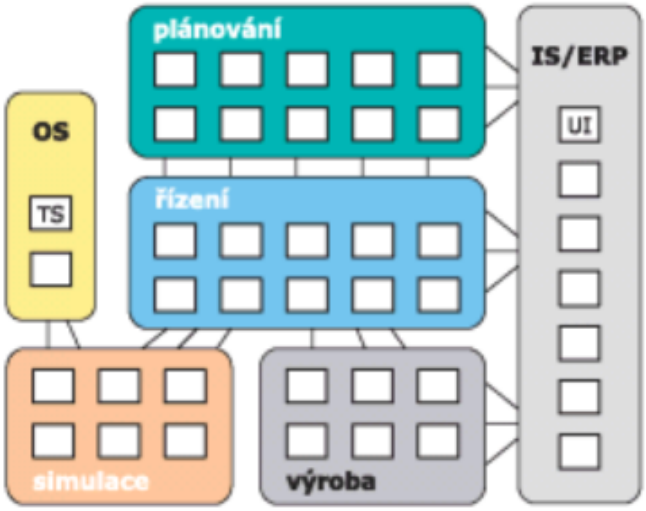
2.4.6 Multiagentní systémy ve výrobě

V multiagentním systému není výrobní postup a struktura výrobních linek pevně zadán ve struktuře řídicího systému, ale vzniká až při vzniku nové objednávky. [9]

Tento systém je také schopen velmi účelně reagovat na změny, které mu způsobí prioritní objednávka, změna v objednávce nebo výpadek některé z výrobních jednotek. Problém začne řešit jeden konkrétní agent, který si v případě neúspěchu vyžádá spolupráci okolní agenty a vzniklá odezva je úměrná závažnosti příčiny. [9]

Je třeba vzít v potaz, že jeden agent může reprezentovat libovolně velkou část reálného světa, takže je klidně možné reprezentovat celou výrobní halu jedním agentem a v druhé může být každý stroj reprezentován několika agenty. Obecně platí, že čím více máme agentů, tím je systém flexibilnější, ale také náročnější na hardware a komunikaci. [9]

Uvedeme si příklad architektury multiagentního systému pro řízení, simulaci a plánování výroby.



Obrázek 2.5: Příklad architektury multiagentního systému ve výrobě [9]

- **IS/ERP** – softwarové vybavení firmy, které je tzv. agentifikované, jedná se například o informační systém, systém ERP apod. [9]
- **Výroba** – představuje agenty obstarávající reálnou výrobu [9]
- **Plánování** – agenti zodpovědní za plánování výroby a za přizpůsobení plánů reálnému stavu výroby nebo její simulace [9]
- **Řízení** - v této mezivrstvě mezi plánováním a reálnou výrobou agenti zejména převádějí obecně formulované plány do konkrétních příkazů [9]
- **Simulace** - agenti zodpovědní za simulaci výroby jsou pro systémy bez přímého napojení na reálnou výrobu nepostradatelnou zpětnou vazbou pro agenty na vyšších úrovních [9]
- **OS - ovládání simulace** - agenti zodpovědní za řízení simulace [9]

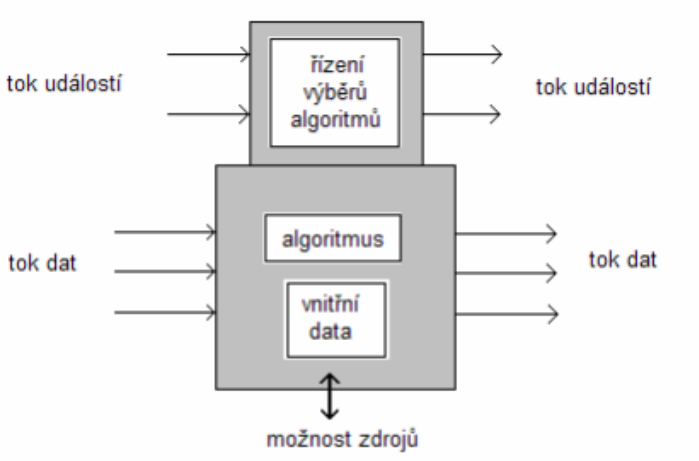
2.5 Holonický systém

Holonické systémy převzaly mnoho principů od multiagentních systémů, takže je zde velká podobnost. Největším a nejdůležitějším rozdílem je, že holonické systémy jsou systémy bezprostředně spojené s fyzickým zařízením.

Koncept holonů jako základních jednotek organizačních celků v biologických a sociálních systémech byl zaveden Arthurem Koestlerem v roce 1967. Koestler vycházel z myšlenky, že celky a části ve své podstatě nikdy neexistují samostatně. Slovo holon je složenina z řeckých slov holos (celek) a on (část). Tímto slovem lze tedy popsat jak celek, tedy soubor subsystémů, které tvoří systém, ale současně i část, tedy například jeden ze subsystémů tohoto

systému. Příkladem může být buňka, která se skládá mj. z jádra a plazmy a současně je částí většího celku např. tkáně. Rozdíl mezi holonem a agentem spočívá v tom, že holon může obsahovat další stejné či odlišné holony a i ty mohou obsahovat další holony. [10]

Z pohledu automatického řízení je holon základní stavební prvek, který slouží k dopravě, transformaci a ukládání informací a fyzických objektů, které mohou být organizovány v jiných strukturách. [11]



Obrázek 2.6: Model holonu [11]

Jednou z nejdůležitějších vlastností holonu je rekurzivita, což znamená že funkce je neustále opakovaně volána dokud není zastavena předem definovanou podmínkou. Holon může na rozdíl od agentů ve své struktuře obsahovat jiné holony, které mohou mít stejnou nebo jinou architekturu. Holony lze tedy definovat jako autonomní, kooperativní a částečně inteligentní modulární bloky, které jsou funkční v rámci decentralizovaného řízení. [11]

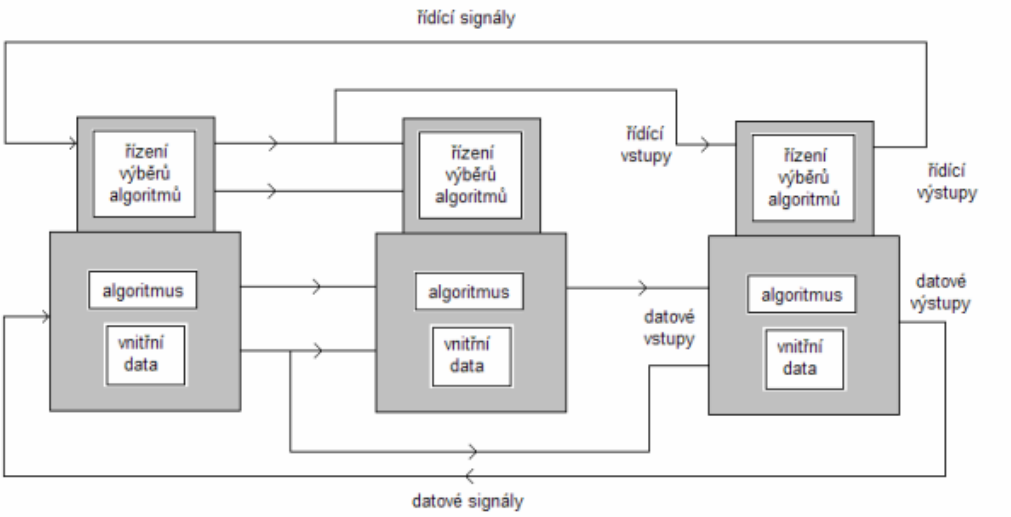
2.5.1 Komunikace v holonických systémech

K naplnění vize holonického výrobního podniku je nutné zpracovávat informace na různých úrovních:

- **Úroveň řízení v reálném čase** - úzké spojení s fyzickým výrobním zařízením [11]
- **Úroveň plánování a rozvrhování výroby** - toto plánování se musí provádět na úrovni celého závodu, jeho odděleních i jednotlivých výrobních úsecích [11]
- **Úroveň správy dodavatelsko-odběratelského řetězce** - zde je zahrnut přesun materiálu, polotovarů, produktů a služeb mezi výrobním závodem a externími entitami (dodavatelé, zákazníci apod.) [11]

Největší pozornosti se dostává úrovni řízení v reálném čase. Holony určené pro řízení v reálném čase jsou obvykle spojeny s fyzickým zařízením

a jsou schopny pomocí čidel pozorovat a ovlivňovat procesy v okolním prostředí. Bezprostřední fyzická vazba mezi holony a výrobní infrastrukturou je charakteristickým znakem holonů. [11]



Obrázek 2.7: Funkční bloky [11]

Nejpopulárnější je model holonu, který se skládá z jednoho či několika subsystémů nízké úrovně a obalu. Subsystémy na nižší úrovni jsou založené na funkčních blocích (pro řízení v reálném čase). V obalu je softwarový agent, který působí pomaleji. Jsou brány v úvahu tři komunikační úrovně: [11]

- **Komunikace uvnitř holonu** - mezi funkčními bloky a softwarovým agentem [11]
- **Komunikace mezi holony** - pro výměnu informací mezi agentními částmi holonů [11]
- **Přímá komunikace** - na úrovni funkčních bloků sousedních holonů [11]

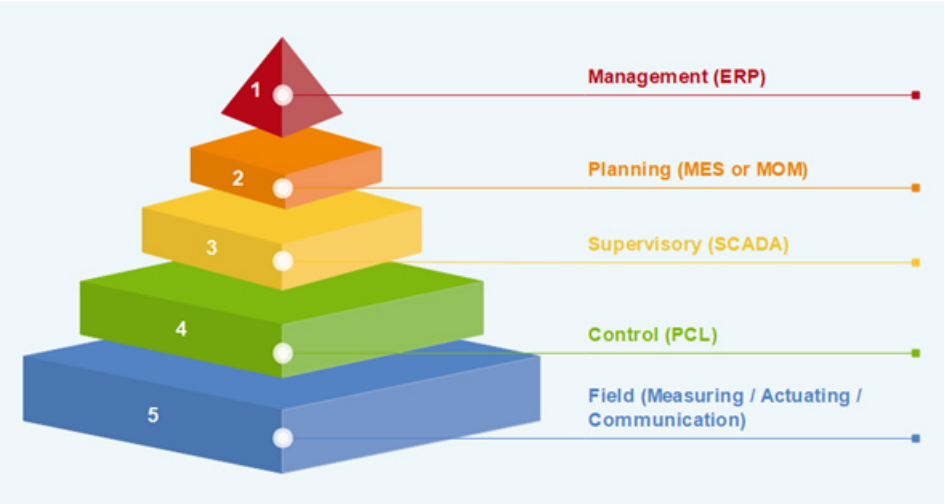
Holony definované tímto způsobem se ve skutečnosti chovají jako standardní agenti. Mohou navzájem komunikovat, provádět složité rozhodovací procesy, kooperovat, vytvářet výrobní plány a scénáře apod. Takové holony pak lze zařadit např. do globální komunity agentů v podniku, kde mohou např. zasahovat do správy logistického řetězce apod. Pro tyto účely je nutná standardizace komunikace mezi holony.[11]

2.5.2 Holonické systémy ve výrobě

V souvislosti s holonickými systémy se mluví o holonic factory, tedy holonickém výrobním závodu, kde jsou všechny části procesu, od objednání až po doručení k zákazníkovi, založeny pouze na holonických procesech. Jednotlivé autonomní holony tedy zastupují naprosto všechny fyzické i virtuální členy tohoto procesu a následně jsou díky komunikaci mezi sebou schopni zajistit automatický běh celé výroby.[11]

2.6 Komunikace v distribuovaném řídicím systému

V každém řídicím systému je komunikace extrémně důležitá, proto je zvolení správných komunikačních protokolů a sběrnic jedním ze základních bodů během návrhu systému. V DCS musí každé PLC komunikovat s jednotlivými stroji na úrovni polní instrumentace ať už přes I/O modul nebo přímo přes průmyslovou komunikační sběrnici, s dalšími PLC, s různými HMI (human-machine interface), kde běží SCADA (Supervisory Control And Data Acquisition) server(y), a výše položenými členy automatizační pyramidy, kteří jsou přítomni.

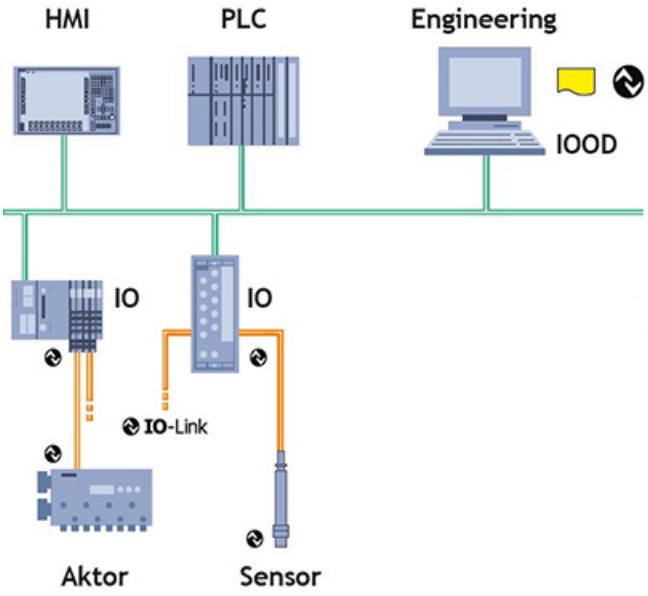


Obrázek 2.8: Pyramida automatizace [12]

2.6.1 Komunikace s úrovní polní instrumentace

Nejnižší úrovní komunikace je spojení s ovládanými stroji, které většinou bývají pomocí vodičů připojené k I/O modůlům. Některé komplexnější stroje mohou být připojeny přes některou z průmyslových sběrnic ke komunikačnímu modulu.

Jednou z možností je použít IO-Link, což je první standardizovaná technologie komunikace vstupů a výstupů se senzory a akčními členy (IEC 61131-9). Jedná se o point-to-point komunikaci probíhajícím na 3-vodičovém kabelu. [13] Nejedná se o průmyslovou sběrnici, ale o nesíťový komunikační standard připojení senzorů a akčních členů.[14]



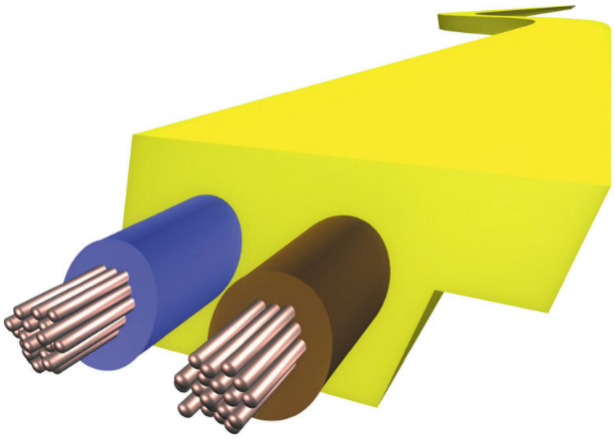
Obrázek 2.9: Příklad systémové architektury s využitím IO-Linku (oranžové vedení) [13]

IO-Link se dá použít s téměř jakoukoliv průmyslovou sběrnicí, nabízí diagnostiku senzorů i akčních členů a je podporován velkým množstvím výrobců. [14]

Existují i sběrnice speciálně dělané na tuto úroveň komunikace. Nejznámější z nich je AS-I.

AS-Interface je zkrácený název pro Actuator Sensor Interface neboli rozhraní akční člen - snímač. Jedná se o jednoduchou technologii, která umožňuje připojit snímače a akční členy k PLC pomocí jediné sériové linky, přesněji řečeno se jedná o dvoužilový kabel, na kterém probíhá vzájemná komunikace a současně slouží i jako napájecí vedení. Nevyžaduje přitom stínění ani ukončovací rezistory, čímž se drží své základní myšlenky, což je snadná instalace a minimální náklady. Tuto myšlenku podporuje i fakt, že přechod z klasického paralelního zapojení na tuto sériovou sběrnici nevyžaduje žádné dodatečné náklady na vybavení a zaškolení uživatele ani firmy projektující a instalující systém.[15][16]

Komunikace probíhá typem master/slave, je možno připojit až 248 vstupů či výstupů a kabel může být dlouhý až 300 metrů. Topologie je volná (strom, hvězda, liniová), cyklus sběrnice je menší než 5 ms a integrace do vyšších úrovní (např. Profibus) je snadná. [16]

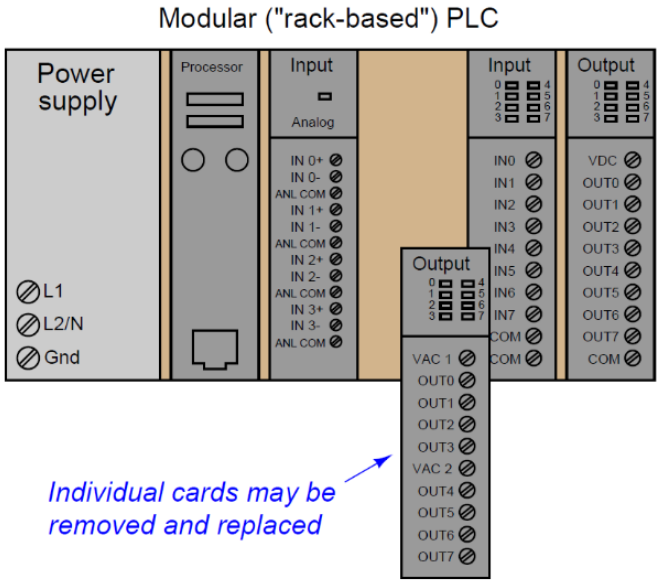


Obrázek 2.10: Pro AS-I kabel je typická žlutá barva [17]

2.6.2 Komunikace s moduly

Procesor PLC se připevňuje na nosič (rack), kde se nachází místo pro další moduly. Podle požadavků lze vybírat z různě velkých nosičů, s různě velkým prostorem pro další moduly. Tyto moduly mohou být komunikační, modul pro vstupy, pro výstupy, nebo kombinace vstupů a výstupů (analogových i digitálních). Už při výběru nosiče bychom měli mít rozmyšleno, zda budou vstupy a výstupy lokální (připojené přímo na nosiči), distribuované (propojené s nosičem pomocí komunikačního modulu a průmyslové komunikační sběrnice), nebo kombinací obou zmíněných možností.

Abychom mohli navrhnout optimální zapojení vstupů a výstupů pro daný proces, je třeba zvážit několik kritérií. Nejdůležitějším je optimální výkon stroje. Nemůžeme si dovolit, aby kvůli méně častému skenování kritických vstupů a výstupů docházelo k jakémukoliv nepříznivému ovlivnění procesu nebo strojů. Například enkodér musí být připojen tak, aby jeho skenování probíhalo co možná nejčastěji, naopak žárovka osvětlující stroj nepotřebuje sken tak často. Z těchto důvodů se doporučuje ponechat kritické vstupy a výstupy zapojené v modulu přímo na nosiči. Samozřejmě je i mnoho přístrojů, kde bohatě postačuje i rychlost skenování distribuovaně zapojených vstupů a výstupů, kterou omezuje rychlost průmyslové sběrnice. [19]



Obrázek 2.11: I/O moduly [18]

Dalším kritériem je snadná montáž stroje. U některých strojů je třeba některé součástky pravidelně měnit, některé potřebují častou údržbu a musí se kvůli ní rozebrat. V takových případech je dobré rozpojovat co nejméně kabelů, jak kvůli času, tak kvůli možnosti chyby při opětovném zapojení. Proto se i v případě distriubovaného rozmístění vstupů a výstupů musí zvážit, zda se modul se vstupy a výstupy umístí do skříně umístěné na stěně, nebo přímo do ovládaného stroje.[19]

Posledním ale rozhodně neméně důležitým kritériem je cena. U distribuovaných I/O je cena závislá zejména s komunikačním adaptéru, takže čím více vstupů a výstupů dokážeme dát na jeden adaptér, tím lépe. [19]

Je třeba si uvědomit, že hlavním cílem distribuovaného rozmístění vstupů a výstupů je úspora kabelů, což ušetří místo a zvýší bezpečnost na pracovišti, a při správném navržení může i snížit náklady na realizaci.

Jak již bylo řečeno, tak v distribuovaně rozložených vstupech a výstupech omezuje rychlost skenování rychlost přenosu průmyslové sběrnice. Z tohoto důvodu se již příliš nevyužívá protokolů využívajících seriovou komunikaci, které rychlostně nestačí na protokoly využivající IP adres.[19]

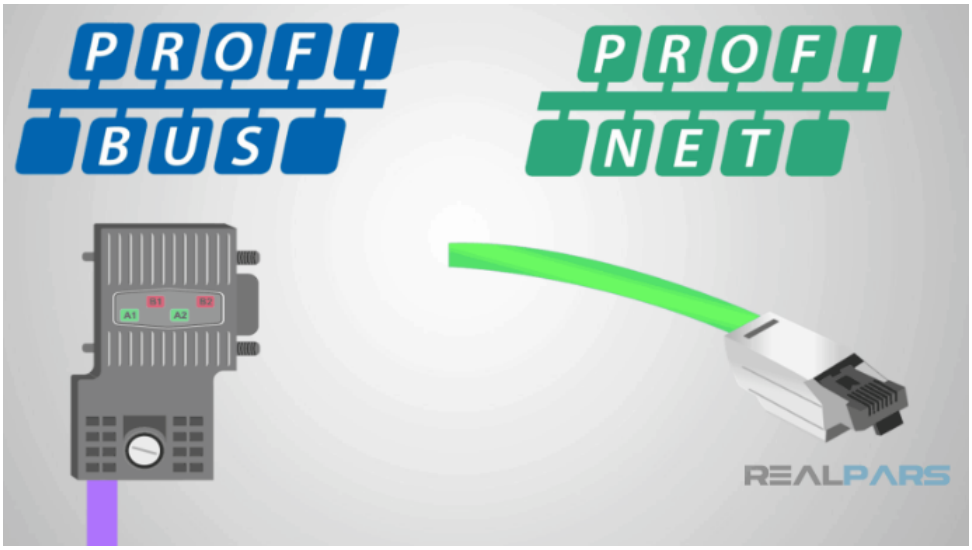
Díky dominanci PLC od společnosti Siemens na tuzemském trhu je nejvíce používanou sběrnicí Profinet, který využívá fyzickou vrstvu Ethernetu a má pevnou přenosovou rychlost 100 Mb/s. Není omezen na liniovou strukturu, lze použít i topologii hvězdy a stromu. Propojovací kabel může mít až 100 m a počet připojených stanic je omezen pouze počtem volných IP adres. Každé stanici je přiřazena unikátní MAC a IP adresa a uživatelem daný název. Spojení je typu point-to-point, navíc je zde funkce Fast-Start-Up, díky které je možné zkrátit potřebnou dobu k oživení modulu ze standartních 10 s na 500 ms.[20]

2.6.3 Komunikace mezi jednotlivými PLC

Nejprve je třeba PLC fyzicky propojit, tedy zvolit průmyslovou komunikační sběrnici, a následně podle toho nakonfigurovat komunikaci v daném softwaru.

Stejně jako u modulů je momentálně nejčastější volbou Profinet, který využívaný hlavně s PLC značky Siemens, které jak už bylo zmíněno prakticky ovládají tuzemský trh.

Samozřejmě je zde ale i mnoho dalších možností. Mezi stále používané patří například Profibus, který má 3 varianty. Konkrétně se jedná o Profibus DP, Profibus FMS a Profibus PA. Nejrozšířenějším a zároveň nejjednodušším z nich je Profibus DP, který umožňuje propojit stanice v liniové topologii. Pokud je délka jednoho segmentu sběrnice menší než 100 metrů, tak dosahuje přenosové rychlosti až 12Mb/s, nicméně pokud nevadí snížení rychlosti až na 9,6kb/s, tak můžeme segment prodloužit až na 1200 metrů. Včetně opakovačů signálu, kteří se používají při přenosu signálu přes velké vzdálenosti, může být ke sběrnici, která musí být opatřena zakončovacím odporem, paralelně připojeno až 32 zařízení na segment, celkově pak až 127. Komunikace probíhá po stíněné dvoulince, případně optickém kabelu a každé zařízení má pevně přidělenou adresu.[20]



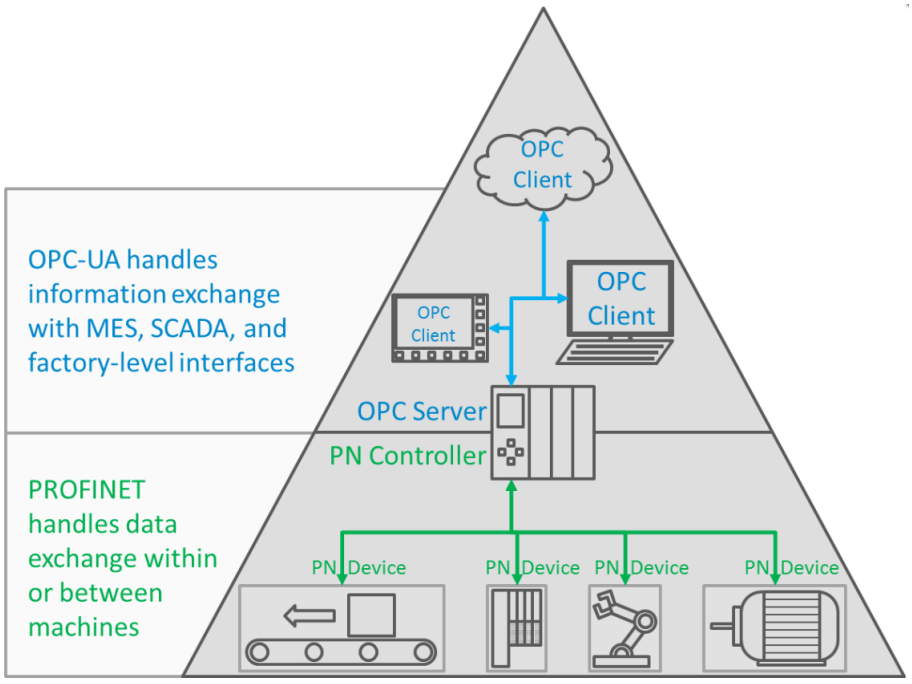
Obrázek 2.12: Průmyslové sběrnice Profibus a Profinet [21]

Někteří výrobci vybavují svá PLC konektorem k jimi vytvořeným sběrnicím, které se k tomuto účelu dají také využít. Příkladem je sběrnice MPI od společnosti Siemens. Tato sběrnice byla vytvořena pro programování a přenos programových paketů, ne pro sbírání dat z distribuovaných periferií. V síti musí být alespoň jeden master, který obstarává tok dat při volitelné rychlosti mezi 9kbit/s a 12Mbit/s. I když maximální vzdálenost není limitována, tak je tato sběrnice navržena pro přenos dat na vzdálenost v desítkách, případně stovkách metrů. Tyto vlastnosti nedělají z MPI ideálního kandidáta na propojení PLC, nicméně možné to je. [22]

Po zvolení sběrnice je třeba zvolit, pomocí čeho bude komunikace na síti nakonfigurována. Dle mého názoru nejlepší, v případě že máme PLC od různých výrobců dokonce jedinou mě známou, volbou je OPC UA. V ostatních případech záleží jaké možnosti výrobce připravil, u PLC Siemens jsou to bloky PUT/GET.

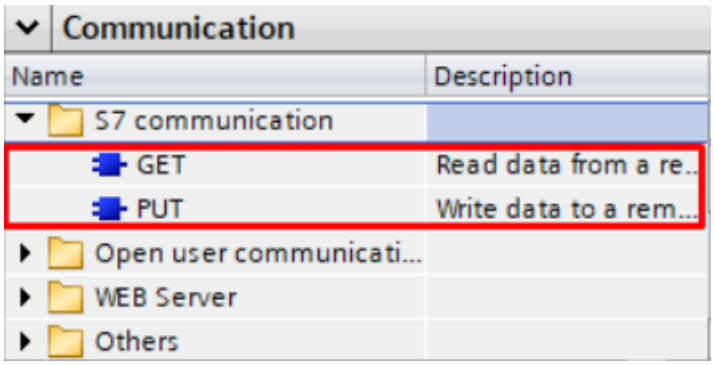
Když standart OPC v roce 1996 vznikl, jeho smyslem bylo převést abstraktní protokoly specifické pro PLC (např. Modbus, Profibus) do standartizovaného rozhraní, které by spojilo HMI/SCADA systém s prostředníkem, který konvertoval požadavky pro psaní/zápis do specifických požadavků pro zařízení a naopak. Myšlenka se ujala a standard si našel široké uplatnění v mnoha průmyslových odvětvích i když byl omezen pouze na operační systém Windows. Zkratka OPC tehdy znamenala OLE (Object Linking and Embedding) for Process Control, tedy v překladu vestavování a propojování objektů v řízení procesů a dnes je tato verze známá jako OPC Classic.[23]

Od roku 2008 totiž vyšla nová verze OPC, která nese název OPC UA. Tento standard je již nezávislý na operačním systému, což se promítlo do nového významu zkratky, která pro OPC UA znamená Open Platform Communication Unified Architecture, což se dá přeložit jako sjednocená architektura komunikace na otevřené platformě. Je to vlastně databáze proměnných, kam může každá připojená komponenta psát a odkud může číst, pokud má potřebná povolení. Velkou výhodou také je, že komunikace je šifrovaná, což není v průmyslové automatizace úplně běžné, navíc jsou aplikace tohoto standardu snadno rozšiřitelné, bez zásahu do již funkčních částí. Dříve musel OPC server běžet na počítači, ale dnešní moderní PLC již mají tento server zabudovaný (např. PLC Siemens Simatic S7-1500). [24]



Obrázek 2.13: Schéma komunikace v řídicím systém za použití OPC UA [25]

Pro PLC Siemens je další možností využít bloky PUT/GET. Jak je již z názvu patrné, tak každý blok stojí na jednom konci komunikace, blok PUT informaci dává a blok GET bere. Konfigurace těchto bloků je jednoduchá a pro menší projekty (ve smyslu počtu sdílených proměnných mezi PLC) naprosto dostačující.



Obrázek 2.14: Bloky PUT a GET se nachází v TIA portalu ve složce S7 communication

Obrázek 2.15: Bloky PUT a GET v jazyce LAD

2.6.4 Komunikace se SCADA systémem a dalšími členy

Zde je díky široké nabídce podporovaných sběrnic použít prakticky libovolnou. Následně se v použitém softwaru dodefinují parametry komunikace.

2.7 UML

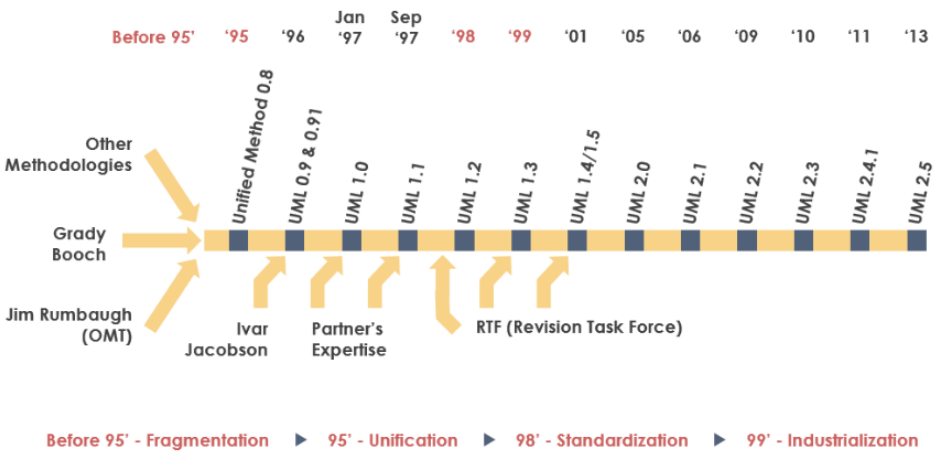
Unified Modeling Language (UML) je standardizovaný modelovací jazyk, který se skládá ze sady diagramů. Tyto diagramy pomáhají zejména vývojářům softwaru a systémů se specifikací, konstrukcí, vizualizací a dokumentací informačních systémů. UML tedy představuje soubor osvědčených technických postupů při modelování velkých a složitých systémů a je velmi důležitou součástí vývoje objektově orientovaného softwaru. K vyjádření designu informačních projektů UML využívá grafické notace a usnadňuje projektovým týmům komunikovat, zkoumat návrhy na potenciální úpravy systému a ověřovat architektonický návrh systému. [26]

2.7.1 Historie UML

Devadesátá léta byla obdobím vývoje objektově orientovaných jazyků (např. C++), které byly použity k vytvoření složitých a rozsáhlých systémů. Problémem byla jejich obtížná srozumitelnost, což vedlo k mnoha problémům s analýzou a samotnými návrhy při vysvětlování dalším lidem.[27]

Z tohoto důvodu v roce 1994 začali softvaroví inženýři Grady Booch, Ivar Jacobson a James Rumbaugh ze společnosti Rational software vyvíjet jazyk UML, jehož 1. verzi dokončili v roce 1996. Později byly do UML zavedeny behaviorální modely a stavové diagramy, které vynalezl David Harel.[27]

UML byla v roce 1997 uznána jako standard Object Management Group. Object Management Group je zodpovědná za správu UML od doby, kdy byla přijata jako standard. V roce 2005 schválila Mezinárodní organizace pro normalizaci UML jako normu ISO. Používá se v různých průmyslových odvětvích pro vytváření objektově orientovaných modelů. Poslední verze UML je 2.5.1, která byla vydána v prosinci 2017.[27]



Obrázek 2.16: Historie vývoje UML (bez poslední verze 2.5.1) [26]

2.7.2 Diagramy

UML diagramy jsou rezdělené do 3 kategorií:

1. **Strukturální diagramy** - používají se k znázornění statického pohledu na systém a představují tu část, která představuje strukturu systému a její objekty [26]
   - **Class diagram** - popisuje statické struktury systému pomocí tříd, jejich atributů, operací a vztahů mezi objekty
   - **Object diagram** - je instancí diagramu třídy, ukazuje snímek podrobného stavu systému v určitém časovém okamžiku, takže zahrnuje objekty a jejich vztahy [28]
   - **Package diagram** - může ukazovat jak strukturu, tak závislosti mezi podsystémy nebo moduly a různé pohledy na systém, například jako vícevrstvý aplikační model [29]
   - **Component diagram** - v podstatě diagramy tříd, které se zaměřují na součásti systému, které se často používají k modelování statického implementačního pohledu systému [30]
   - **Deployment diagram** - zachycují hardware, který bude použit k implementaci systému a vazby mezi různými položkami hardwaru [31]
2. **Behaviorální diagramy** - diagramy, které se zabývají pohyblivými nebo dynamickými částmi systému [26]
   - **Activity diagram** - popisují, jak jsou činnosti koordinovány za účelem poskytování služby, která může být na různých úrovních abstrakce [32]
   - **Use case diagram** - specifikují očekávané chování a ne přesnou metodu, jak toho dosáhnout [33]
   - **State machine diagram** - obvykle používá k popisu chování závislého na stavu objektu [34]
3. **Interakční diagramy** - používají se k zobrazení interakce mezi dvěma entitami a způsobu toku dat v nich [26]
   - **Timing diagram** - používají se k zobrazení interakcí, když primárním účelem diagramu je zdůvodnit čas [35]
   - **Sequence diagram** - zobrazují pořadí interakce vizuálně pomocí vertikální časové osy diagramu [36]
   - **Communication diagram** - se používají k ukázce toho, jak objekty interagují, aby provedly chování konkrétního use case [37]

Kapitola 3

Navržení a realizace distribuovaného řídicího systému

3.1 Prvotní analýza a návrh řídicího systému

Použití umělé intelegence v distribuovaných systémech zní jako skvělý nápad, ale z několika důvodů jsme se nakonec rozhodli ji upozadit. Hlavním důvodem je, že forma umělé inteligence, která je využívána v multiagentních a holonických systémech, je poměrně komplexní a systém by přišel o veškerou rekonfigurovatelnost, která je z našeho pohledu v dnešní průmyslové automatizaci důležitější.

Výstupem tedy bude software, který umožní zákazníkovi snadno vygenerovat PLC kód pomocí SFC. Pro veškeré použité komponenty bude vytvořena knihovna, takže se vytvořený systém stane snadno rekonfigurovatelný. Co se zmíněné umělé inteligence týče, tak jistou formu si bude moct uživatel pomocí SFC vytvořit, pokud se tedy spokojíme s definicí, že o umělou inteligenci se jedná, pokud stroj napodobuje svým chováním člověka. Jelikož se jedná o distribuovaný řídicí systém, tak bude samozřejmě obsahovat více PLC, mezi kterými bude třeba nastavit komunikaci, ale jelikož tato část se tvoří pouze při prvním programování a následně bývá drobně upravována, tak to na rekonfigurovatelnosti systému nijak neubere.

Jako vstupní data poslouží smyšlená objednávka na tento software, ze které bude podle za pomocí lexikální analýzy vytvořen Class diagram, který povede k výsledné verzi softwaru a samotného řídicího systému.

3.2 Zákazníkova objednávka a lexikální analýza

3.2.1 Objednávka

Chtěli bychom software, pomocí kterého můžeme rychle přeprogramovat náš stroj na úpravu plechů. Tento stroj může plechy ohývat, stříhat a vrtat do nich, ale ne vždy se využívají všechny 3 operace.

Podavač podá do přístroje plech a nejprve proběhne jeho upnutí pomocí 2 pneumatických pístů. První operací je stříhání, které je ralizováno jednou ze dvou dvojic pneumatických pístů. Následuje ohýbání, ke kterému jsou k

dispozici hned 3 pneumatické písty s ohýbacími přípravky, které určují úhel ohnutí, a vykonává ho nějaká jejich kombinace. Poslední dostupnou operací je vrtání, které má k dispozici 2 pneumatické písty, kde každý ovládá pohyb jiné vrtací hlavice. Po dokončení poslední operace si plech opět odebere podavač.

Software, který tvoří část objednávky by nám měl dovolit připravit programy pro různé kombinace operací. Chtěli bychom aby bylo možné ovládat stroj v případě nutnosti manuálně a také nějakou diagnostiku, která nás dovede rovnou k poruchové části stroje. Z bezpečnostního hlediska je nutné pracovní prostor stroje zabezpečit (optickými čidly), tak aby v případě výskutu člověka v pracovním prostoru byl stroj zastaven a byla signalizována chyba typu přerušení výroby. Obsluha/člověk má být upozorněn na skutečnost, že se blíží do pracovního prostoru stroje (a tedy k jeho zastavení) např. rozblikáním oranžového světla na semaforu. Systém je též potřeba zabezpečit proti práci (ohýbání) naprázdno, například čidlem přítomnosti polotovaru, aby se předešlo poškozování zařízení. Systém musí mít samozřejmě i nouzový stop, který bude zastoupen tlačítkem s aretací. V případě zmáčknutí se musí všechny stroje okamžitě zastavit a pokud bude zmáčknuto i tlačítko reset, tak se vrátí do výchozích poloh, pokud ne, tak bude stroj pokračovat tam, kde skončil.

3.2.2 Lexikální analýza

Je třeba si uvědomit, že cílem je analyzovat samotný systém, nikoliv software pro generování kódu pro PLC, ve kterém máme volnost, ale samozřejmě musí být přizpůsoben potřebám daného systému.

Prvním krokem je tvorba seznamu kandidátů na názvy tříd, což je výběr podstatných jmen a jmenných frází z textu objednávky.

Protože takových kandidátu je vždy mnoho a nakonec bude použit pouze zlomek z nich, je třeba je přefiltrovat, což se provádí na základě několika kritérií.

Kritéria filtrování:

- A) Nadbytečné třídy (např. synonyma)
- B) Irelevantní (beze vztahu k řešenému problému)
- C) Neurčité (příliš široký sémantický obsah)
- D) Atributy (vlastnosti a schopnosti jiných objektů) – může se stát atributem třídy
- E) Operace (vyloučení aktiv, které operují s atributy) – může se stát operací třídy
- F) Implementační konstrukty (produkty programátorského myšlení)

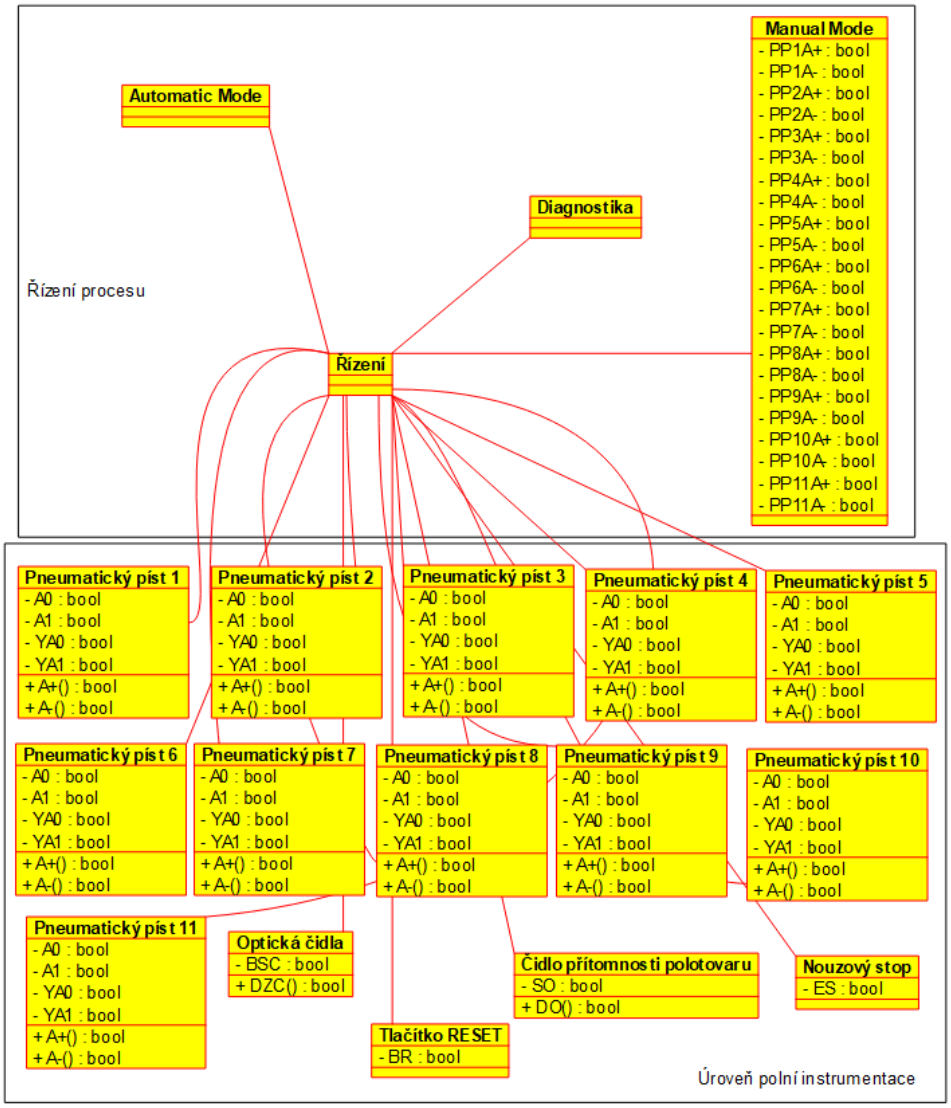
Kandidát na název třídy	Důvod vyřazení
Software	C
Stroj	C
Plech	B
Operace	C
Podavač	B
Přístroj	C
Upnutí	B
Pneumatický píst	
Stříhání	B
Dvojice pneumatických motorů	A
Ohýbání	B
Ohýbací přípravky	B
Úhel ohnutí	B
Kombinace	C
Vrtání	B
Program	C
Kombinace operací	C
Diagnostika	
Poruchová část stroje	A
Bezpečnostní hledisko	C
Prostor stroje	A
Optická čidla	
Výskyt člověka	D
Pracovní prostor	B
Chyba typu přerušení výroby	F
Obsluha	A
Skutečnost	C
Pracovní prostor stroje	A
Zastavení	E
Oranžové světlo na semaforu	A
Práce naprázdno	B
Čidlo přítomnosti polotovaru	
Poškozování zařízení	B
Nouzový stop	
Tlačítko s aretací	C
Zmáčknutí	B
Všechny stroje	C
Tlačítko reset	
Výchozí polohy	F

Tabulka 3.1: Seznam kandidátů na názvy tříd

Nyní vytvoříme seznam názvů tříd s atributy a operacemi v softwaru Umbrello za pomoci tabulky 3.1 a našich obecných znalostech v dané oblasti. Některé třídy budeme muset přidat, protože v objednávce byly obsaženy tak, že nebylo možné identifikovat pomocí lexikální analýzy (např. manuální režim) nebo nebyly obsaženy vůbec (např. třída Řízení). Některé třídy není možné momentálně blíže specifikovat, příkladem může být třída Diagnostika, kde je zatím předčasné hodnotit, jaké bude mít atributy a operace.

3.2.3 Výsledný diagram

Výsledkem je první návrh Class diagramu, ale je velice pravděpodobné, že po osobním seznámení se strojem bude upraven, protože objednávka jako taková není dostatečným zdrojem informací k vytvoření konečné verze diagramu.



Obrázek 3.1: První návrh class diagramu

Jak je patrné z diagramu, tak se zatím nepodařilo určit atributy a operace tříd Automatic Mode, Řízení a Diagnostika a je možné že se změní i ostatní po bližším seznámení se strojem. Pneumatické písty byly pouze očíslovány dle objednávky, protože bez bližšího seznámení se strojem není důležité který slouží k čemu. Pro operace a atributy byly z důvodu přehlednosti použité zkratky (A0, A1 - koncové senzory, YA0, YA1 - relé, A+,A- - vyjetí a zajetí pístu, PPxA+, PPxA- - proměnné pro manuální ovládání pneumatických pístů, BSC - button senzor člověk, DZC - dioda zastavení člověk, BR - button reset, SO - senzor obrobek, DO - dioda obrobek, ES - emergency stop).

3.3 Vývoj softwaru pro generování PLC kódu

Po dokončení Class diagramu je čas vytvořit software pro generování PLC kódu, jehož cílem je maximálně ulehčit a urychlit proces programování zákazníkova stroje. Původní plán byl použít jazyk STL, nicméně dostupná verze softwaru TIA Portal nepodporuje import externích souborů v tomto jazyce a jedinou dostupnou možností je jazyk SCL.

Software bude realizován v jazyce JavaFX ve vývojovém prostředí Netbeans a v grafickém editoru SceneBuilder. Tato kombinace byla zvolena čistě z osobních preferencí.

Samotný kód je připojen v příloze diplomové práce a jsou k němu připojeny komentáře pro základní orientaci, takže zde bude program představen spíše z funkčního hlediska.

Software je pojmenován DOCOGE, což je zkratka pro DOležal's COde GEnerator, a jedná se o verzi 1.0. Cílem je zatím bezchybné generování kódu a počítá se že nebude dokonale odladěný. Některé části sloužící pro lepší orientaci uživatele tak nejsou plně funkční.

3.3.1 Knihovna

Pro software na generování PLC kódu je logickým začátkem vytvoření knihovny. Jako její prvky poslouží všechny dostupné elektropneumatické ventily, tlačítko a dioda.

U každé položky knihovny je třeba připravit všechny použité části, takže nestačí pouze funkční blok, ale i tag table, případně countery a timery. Společnost Siemens vydala v jednom ze svých manuálů na programování PLC i návod, jak mají být tyto soubory vytvořeny a v jakém formátu, ale jako nejjednodšší způsob se jeví vytvořit krátký program pro pohyb pneumatického motoru s vybraným elektropneumatickým ventilem, který bude obsahovat všechny zmíněné části (tedy tag table, counter i timer), a následně tyto soubory z TIA Portalu vyexportovat. Tímto způsobem dostaneme formát všech potřebných souborů a stačí pouze přepsat programovou část. Díky exportní funkci jsme zjistili, že funkční bloky a funkce jsou vyexportovány s koncovkou .scl, datové bloky, timery a countery s koncovkou .db a všechny tyto souory lze otevřít v poznámkovém bloku, díky čemuž do nich bude moct program přímo zapisovat. Co se tag table týče, tak ta je vyexportována v

excelovském souboru .xlsx, takže bude mít každý element knihovny svůj soubor, který bude nakonec sloučen do tag table pro celý projekt.

Knihovna bude tedy složka Library, ve které bude podsložka AdditionalFiles a to z toho důvodu, že v programu bude potřeba načíst soubory knihovny, aby je mohl uživatel zařadit do projektu, což znamená že jedním tlačítkem přidá všechny potřebné soubory pro danou komponentu do projektové složky. Aby se do knihovny mohli i nadále přidávat položky (například nově pořízený elektropneumatický ventil) a program pracoval správně, je třeba přidat filtry, díky kterým se zobrazí v nabídce knihovny pouze jedno tlačítko pro každou komponentu a během testování programu se osvědčilo do podsložky AdditionalFiles umístit excelovské soubory komponent a některé další soubory potřebné v dalších fázích generování PLC kódu.

Tlačítko a dioda

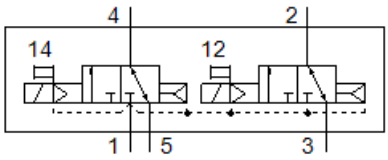
Tlačítko a dioda jsou nejjednodušími komponentami této knihovny, protože nepotřebují nic jiného, než přiřazení proměnné v tag table. Pro tlačítko bude tedy v knihovně excelovský soubor (z toho důvodu, že je jednodušší naprogramovat sloučení excelovských souborů, než do nich zapisovat), který bude obsahovat jednu proměnnou definovanou jako vstup. Dioda bude mít totožný soubor, pouze proměnná bude definována jako výstup. Už zde je třeba dbát na formát vyexportované tag table, protože nestačí pouze přidat název proměnné a její adresu, ale také Data Type, Logical Address, Comment, Hmi Visible, Hmi Accessible, Hmi Writeable a Typeobject ID (Version ID není třeba vyplňovat, při importu se pak objeví varování, že není toto pole definováno a tak se předpokládá, že se jedná o nejnovější dostupnou verzi).

Name	Tag table_1	Bool	%I1.1		True	True	True

Obrázek 3.2: Část tag table pro tlačítko

Bistabilní elektropneumatický ventil 4/2

První použitý elektropneumatický rozvaděč je bistabilní 4/2 pro dvojčinné pneumatické motory a bude v knihovně pod zkráceným názvem DABS_42.



Obrázek 3.3: Schématická značka ventilu

Jako první je třeba vytvořit fukční blok, který si bude proces volat a budou v něm definované jednotlivé stavy motoru, který tento rozvaděč ovládá.

```
FUNCTION_BLOCK "Enter name here"
{ S7_Optimized_Access := 'TRUE' }
VERSION : 0.1

BEGIN
        //Network 1 - Motor movement
        IF "Enter name here+" THEN
            "Enter name here_Plus" := 1;
            "Enter name here_Minus" := 0;
        END_IF;

        IF "Enter name here-" THEN
            "Enter name here_Plus" := 0;
            "Enter name here_Minus" := 1;
        END_IF;

//Network 2 - Diagnostics
        //Motor not in position
        "Enter name here_Timer_DB".TON(IN := NOT "Enter name here0" AND NOT "Enter name here1",
                              PT := T#2s,
                              Q => "Enter name here_Err_notInPosition");

        //Sensor fail
        IF "Enter name here0" AND "Enter name here1" THEN
            "Enter name here_Err_sensorFail" := 1;
        END_IF;


        //Sensor or motor fail
        "Enter name here_Counter1_DB".CTU(CU:="Enter name here0",
                                R:="Enter name here1",
                                PV:=2,
                                Q=>"Enter name here_Err_sensor0OrMotorFail");

        "Enter name here_Counter2_DB".CTU(CU:="Enter name here1",
                                 R:="Enter name here0",
                                 PV:=2,
                                 Q=>"Enter name here_Err_sensor1OrMotorFail");
END_FUNCTION_BLOCK
```

Obrázek 3.4: Funkční blok ventilu

Je třeba brát v potaz, že program bude do tohoto bloku zapisovat, takže část Enter name here bude nahrazena názvem motoru, který uživatel zadá. Pro vysvětlení funkce bloku předpokládejme, že se motor jmenuje A.

V první části (Network 1) je definován pohyb motoru, tedy jeho vyjetí a zajetí. Pokud se v procesu objeví A+, tak tento fukční blok zapíše hodnotu do proměnné tak, aby motor vyjel, pokud se objeví A-, tak motor zajede. Toho docílí změnou v tomto případě 2 proměnných, protože se jedná o bistabilní rozvaděč.

V druhé části (Network 2) se nachází diagnostika motoru, která upozorní uživatele pokud je něco špatně a přiblíží o jaký problém se může jednat. První proměnnou pro tento účel je A_Err_notInPosition, která se aktivuje, pokud motor není ani v jedné z koncových poloh déle než 2

vteřiny, takže je motor buď nefunční, nebo nedojíždí do koncových poloh (což může způsobit i špatně napsaný program). Druhou diagnostickou proměnnou je A_Err_sensorFail, která se aktivuje pokud jsou oba koncové senzory motoru aktivní ve stejný okamžik, což je samozřejmě nemožné a značí to poruchu na jednom ze senzorů. Posledními 2 proměnnými jsou A_Err_sensor0OrMotorFail a A_Err_sensor1OrMotorFail. Ty upozorní na chybu, pokud se jeden z krajních senzorů aktivuje dvakrát za sebou, bez toho aby byl aktivní druhý senzor, což může značit chybu motoru nebo daného senzoru. Aby zodpovědná osoba správně pochopila, s čím je ve skutečnosti problém, tak je třeba této diagnostice rozumnět, protože při některých poruchách se může aktivovat více diagnostických proměnných.

Jak si lze ve fuknčním bloku všimnout, tak je zde použit TON (Time On Delay) timer a dvakrát CTU (Count Up) counter, které je třeba také importovat, nicméně zde bude použit soubor exportovaný z TIA Portalu, kde je pouze pozměněno jméno bloku. Stejně tak je třeba importovat datový blok, který sice neobsahuje žádné potřebné proměnné, nicméně v jazyku SCL je tento blok třeba k volání funkčního bloku v procesu.

```
DATA_BLOCK "Enter name here_Counter1_DB"
{InstructionName := 'IEC_COUNTER';
 LibVersion := '1.0';
 S7_Optimized_Access := 'TRUE' }
AUTHOR : Simatic
FAMILY : IEC
NAME : CNTR
VERSION : 1.0
IEC_COUNTER

BEGIN

END_DATA_BLOCK
```

```
DATA_BLOCK "Enter name here_Timer_DB"
{InstructionName := 'IEC_TIMER';
 LibVersion := '1.0';
 S7_Optimized_Access := 'TRUE' }
AUTHOR : Simatic
FAMILY : IEC
NAME : IEC_TMR
VERSION : 1.0
NON_RETAIN
IEC_TIMER

BEGIN

END_DATA_BLOCK
```

```
DATA_BLOCK "Enter name here_DB"
{ S7_Optimized_Access := 'TRUE' }
VERSION : 0.1
NON_RETAIN
"Enter name here"

BEGIN

END_DATA_BLOCK
```

Obrázek 3.5: Soubory pro import counteru, timeru a datového bloku

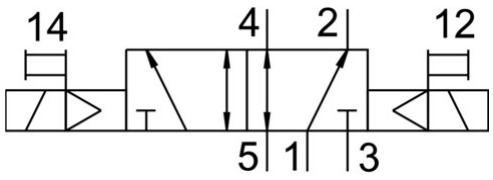
Posledním potřebným souborem je část tag table, která bude obsahovat veškeré proměnné použitých v souvislosti s tímto rozvaděčem. Zde je třeba stejně jako u tlačítka a diody dbát na správný formát a vyplnění všech potřebných polí.

Enter name here_Plus	Tag table_1	Bool	%Q0.3		True	True	True
Enter name here_Minus	Tag table_1	Bool	%Q0.2		True	True	True
Enter name here0	Tag table_1	Bool	%I0.5		True	True	True
Enter name here1	Tag table_1	Bool	%I0.4		True	True	True
Enter name here_Err_notInPosition	Tag table_1	Bool	%M5.0		True	True	True
Enter name here_Err_sensorFail	Tag table_1	Int	%MW6		True	True	True
Enter name here_Err_sensor0OrMotorFail	Tag table_1	Bool	%M5.3		True	True	True
Enter name here_Err_sensor1OrMotorFail	Tag table_1	Bool	%M5.4		True	True	True
Enter name here+	Tag table_1	Bool	%M8.3		True	True	True
Enter name here-	Tag table_1	Bool	%M8.4		True	True	True

Obrázek 3.6: Část tag table pro tento rozvaděč

Bistabilní elektropneumatický ventil 5/2

Další použitý elektropneumatický rozvaděč je bistabilní 5/2 pro dvojčinné pneumatické motory a bude v knihovně pod zkráceným názvem DABS_52.

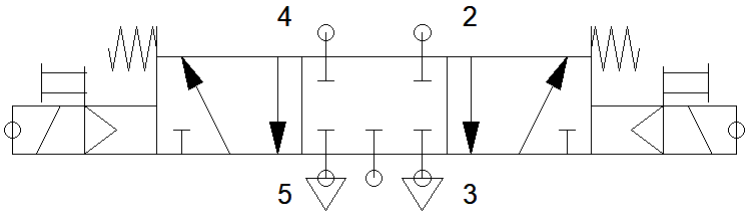


Obrázek 3.7: Schématická značka ventilu

Použité soubory pro tento ventil jsou totožné s předchozím (DABS_42), takže je není třeba uvádět. Rozdíl mezi nimi je v pneumatickém zapojení, z pohledu PLC jsou totožné.

Monostabilní elektrickopneumatický ventil 5/3

Elektropneumatický rozvaděč monostabilní 5/3 pro dvojčinné pneumatické motory bude v knihovně pod zkráceným názvem DAMS_53.



Obrázek 3.8: Schématická značka ventilu

Na rozdíl od předchozích ventilů má tento povolený stav, kdy jsou aktivované obě cívky, takže je nutné definovat i třetí stav.

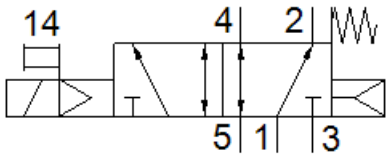
```
BEGIN
//Network 1 - Motor movement
        IF "Enter name here+" THEN
            "Enter name here_Plus" := 1;
            "Enter name here_Minus" := 0;
        END_IF;

        IF "Enter name here-" THEN
            "Enter name here_Plus" := 0;
            "Enter name here_Minus" := 1;
        END_IF;

        IF "Enter name here" THEN
            "Enter name here_Plus" := 1;
            "Enter name here_Minus" := 1;
        END_IF;
```

Obrázek 3.9: První část funkčního bloku ventilu

Ostatní soubory jsou prakticky totožné s již uvedenými.

Monostabilní elektropneumatický ventil 5/2 normálně uzavřený

Elektropneumatický rozvaděč monostabilní 5/2 normálně uzavřený pro dvojčinné pneumatické motory bude v knihovně pod zkráceným názvem DAMS_52_NC.



Obrázek 3.10: Schématická značka ventilu

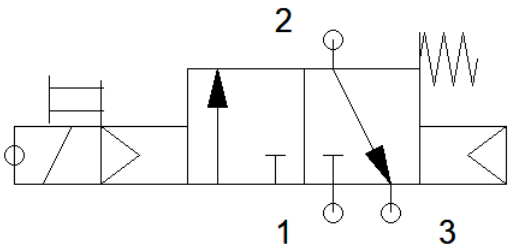
Ke změně oproti předešlím ventilům dochází opět pouze v první části fuknčního bloku. Tento ventil je monostabilní a určení jeho polohy tedy stačí pouze jedna proměnná.

```
BEGIN
        //Network 1 - Motor movement
        IF "Enter name here+" THEN
            "Enter name here_Plus" := 1;
        END_IF;


        IF "Enter name here-" THEN
            "Entername here_Plus" := 0;
        END_IF;
```

Obrázek 3.11: První část funkčního bloku ventilu

Monostabilní elektropneumatický ventil 3/2 normálně uzavřený

Poslední dostupný ventil a zároveň jediný pro jednočinný motor je elektropneumatický 3/2 normálně uzavřený bude v knihovně pod zkratkou SAMS_32_NC.



Obrázek 3.12: Schématická značka ventilu

Z pohledu PLC je tento ventil totožný s předchozím (DAMS_52_NC) a tudíž jsou soubory k importu do softwaru TIA Portal totožné.

Ostatní součásti knihovny

Kromě výše zmíněných souborů, které slouží pro použité součásti v programu, se v podložce AdditionalFiles nacházejí další 3 soubory nutné ke správnému generování kódu.

Prvním je začátek bloku function, kam se v průběhu vygeneruje programovaný proces. Nachází se zde pouze část, která je vždy stejná, protože slouží softwaru TIA Portal k identifici druhu bloku, a název, který je v průběhu přepsán. V průběhu testování programu byla ozkoušena i varianta, kdy si tento soubor program vytvoří, ale poté docházelo citelné prodlevě, což je samozřejmě nežádoucí.

```
FUNCTION "Name" : Void
{ S7_Optimized_Access := 'TRUE' }
VERSION : 0.1

BEGIN
```

Obrázek 3.13: Počáteční soubor procesu

Dalším je exlovský soubor, do kterého se umisťují jednotlivé součásti tag table. Je zde tedy první řádek, který definuje názvy sloupců a list je pojmenován PLC Tags, jak je nutné k importu tag table.

Name	Path	Data Type	Logical Address	Comment	Hmi Visible	Hmi Accessible	Hmi Writeable	Typeobject ID	Version ID

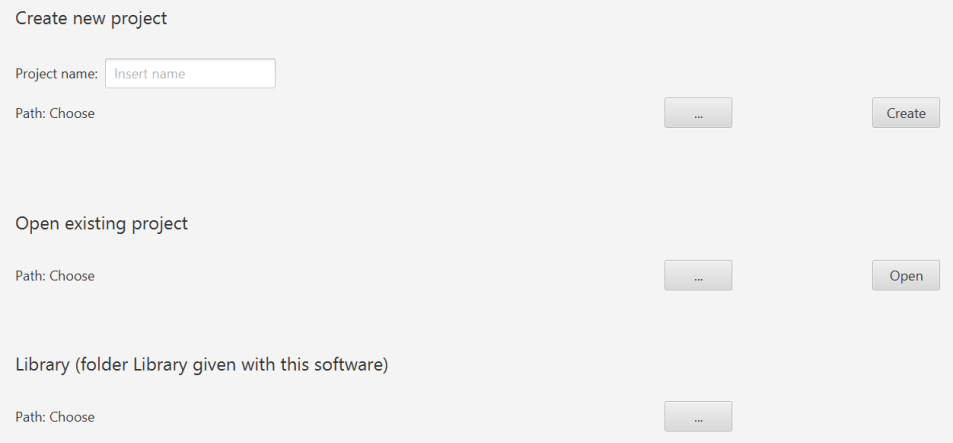
Obrázek 3.14: Tag table před sloučením s použitými komponentami

Posledním souborem je excelovký soubor s proměnnou, která je využita v CASE cyklu procesu a vypadá stejně jako soubor pro tlačítko nebo diodu, pouze se jedná o vnitřní proměnnou. Část NamePr bude nahrazena uživatelem zadaným názvem procesu.

TF_NamePr	Tag table_	Int	%MW20		True	True	True

Obrázek 3.15: Proměnná pro CASE cyklus

3.3.2 Vytvoření a otevření existujícího projektu



Obrázek 3.16: Část úvodní obrazovky softwaru DOCOGE

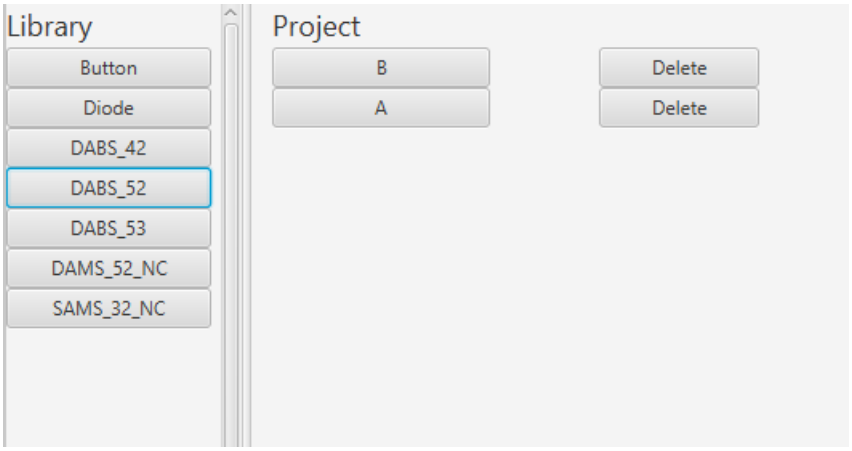
Po spuštění programu se otevře úvodní obrazovka, kde je možné vytvořit nový, nebo otevřít existující projekt. Pro vytvoření projektu je třeba vyplnit Project name a zvolit cestu v sekci Path, což se provede kliknutí na tlačítko ..., čím se otevře prohlížeč pro vybrání umístění. Tlačítkem Create se vytvoří složka se zvoleným jménem ve zvolené složce.

Tato cesta se pak automaticky napíše do cesty pro otevření projektu, případně lze tlačítkem ... tuto cestu změnit. Následně už chybí pouze vybrat knihovku v příslušné části opět tlačítkem ... a kliknout na tlačítko Open.

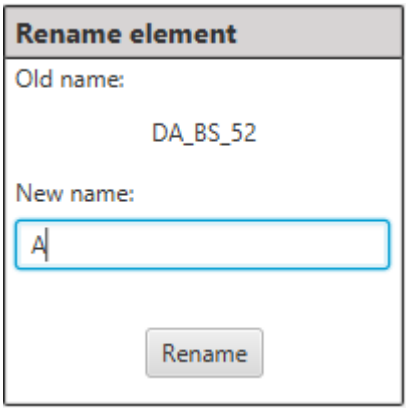
3.3.3 Umístění položek z knihovny do projektu

Po otevření projektu se v levém sloupku zobrazí nabídka knihovny ve formě tlačítek. Po kliknutí na tlačítko symblizující prvek, který chceme do projektu umístit, vyskočí Rename okno, kde zvolíme nový název přidávaného prvku. Nově pojmenovaný prvek se následně zobrazí v projektovém sloupci také jako tlačítko, které po kliknutí zobrazí funkční blok (pokud ho daný prvek má) v textovém poli umístěném v pravé části obrazovky. Společně s tímto tlačítkem se také zobrazí tlačítko Delete, kterým je možné prvek z projektové složky vymazat. Po umístění všech požadovaných prvků do projektů pokračujeme do další fáze kliknutím na tlačítko Step1 - Done.

Program během tohoto procesu zkopíruje veškeré soubory související s daným elementem do projektové složky, následně je přejmenuje podle nově zadaného názvu a přepíše i obsah souborů tam, kde je třeba. Tlačítko Step1 - Done nakonec sloučí veškeré excelovské soubory do jednoho, který jejíž v požadovaném formátu tag table pro TIA Portal.



Obrázek 3.17: Sloupec knihovny a projektu

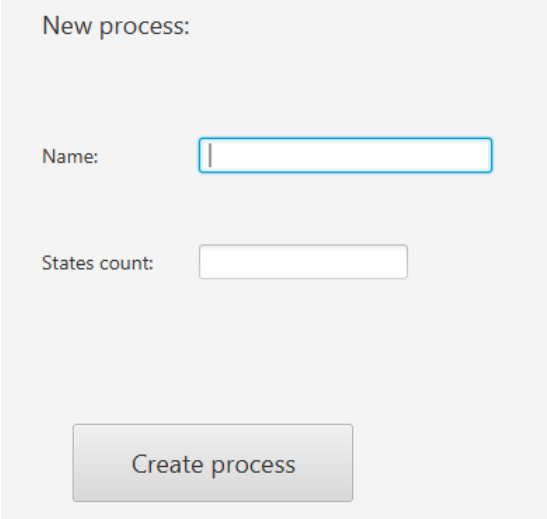


Obrázek 3.18: Renaming okno

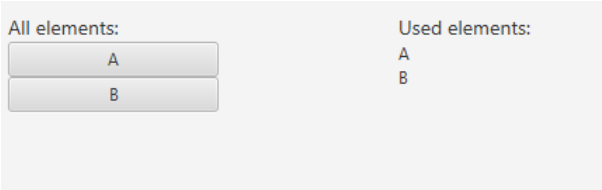
3.3.4 Vytvoření procesu

Dalším krokem je vytvoření samotného procesu. Proces je vytvořen pomocí CASE cyklu, přičemž program ho kompletně vytvoří a přidá proměnou pro tento cyklus do tag table.

Najříve je třeba proces pojmenovat a vyplnit počet stavů, které se v procesu nacházejí, a kliknout na Create process. Poté je třeba vybrat z tabulky All elements ty prvky, které jsou v procesu použity, to je zde z důvodu, aby proces volal pouze ty komponenty, které opravdu využívá. Na další obrazovku se poté dostaneme tlačítkem Add elements.



Obrázek 3.19: Vyvoření nového procesu



Obrázek 3.20: Přidání použitých elementů

Nyní se nacházíme v poslední a nejdůležitější části návrhu procesu. Zde je třeba vyplnit co se v jakém stavu má stát a jaké jsou podmínky k jeho přechodu. Jedná se tedy o realizaci SFC diagramu, nicméně pro složitější provedení diagramů je lepší mít po ruce jeho grafickou podobu, protože zde v textové podobě není návrh tak přehledný.

V levé části se nachází tabulka stavů a akcí, která dává uživateli nápovědu, co zde může použít, ale je třeba ji brát s rezervou, protože není ještě kompletně funkční a slouží tedy spíš pro přibližnou orientaci.

Uprostřed snímku je samotná tabulka přechodů. Zde je třeba postupovat ve správném pořadí, jinak se kód nevygeneruje správně. Pole je možné si předvyplnit libovolně, ale k samotnému procesu generování slouží tlačítka Set a Save a ty je třeba zmáčknout ve specifickém pořadí.

Vždy začínáme od stavu 0, vyplníme Actions in state (akce ve stavu) tedy například vyjetí motoru A zapíšeme jako A+, pokud je v daném stavu akcí více, je třeba je oddělit středníkem (například vyjetí motoru A a B bude tedy zapsáno A+;B+) poté klikneme na tlačítko Set, které tuto akci zapíše a již ji nelze dále upravit.

Zde jsou celkem 3 mechaniky, jak toto pole vyplnit:

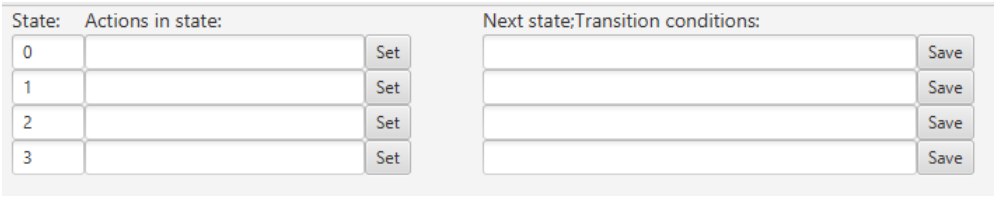
1. **Motor s dvoupolohovým rozvaděčem** - zde stačí napsat jméno motoru společně se znakem pro vyjetí, případně zajetí (A+/A-)

2. **Motor s třípolohovým rozvaděčem** - v tomto případě je třeba přidat za název ještě * a následně jsou rozlišeny 3 stavy, kdy pro krajní platí stejné znaky jako u dvoupolohového motoru a střední je bez znaku (A*+/A*/A*-)

3. **Proměnná** - pokud je třeba pouze setovat a resetovat proměnnou, tak stačí za název dopsat SET, případně RESET (dioda Ready se tedy bude spouštět příkazem ReadySET a vypínat příkazem ReadyRESET)

Následně přejdeme do pole Next state;Transition conditions (následující stav;podmínky přechodu), je třeba vyplnit číslo následujícího stavu a podmínku přechodu, kdy jedno od druhého dělí středník a podmínka musí být zapsána v jazyku SCL (přechod do stavu jedna, podmíněný senzory vyjetých motorů A a B tedy zapíšeme 1;"A1"AND "B1"). Po zapsání klikneme na Save, čímz se pole vyprázdní a pokud je třeba definovat přechod do jiného stavu, tak to provedeme stejně. Pro lepší orientaci je vpravo umístěno textové pole, kam se uloží všechny zapsané přechody i s podmínkami. Po dokončení pokračujeme stejným postupem postupně pro všechny stavy.

Po vyplnění tabulky se tlačítkem Process complete proces uloží a program se vrátí na předchozí obrazovku, takže je možné vytvořit další proces.



Obrázek 3.21: Tabulka stavů a přechodů

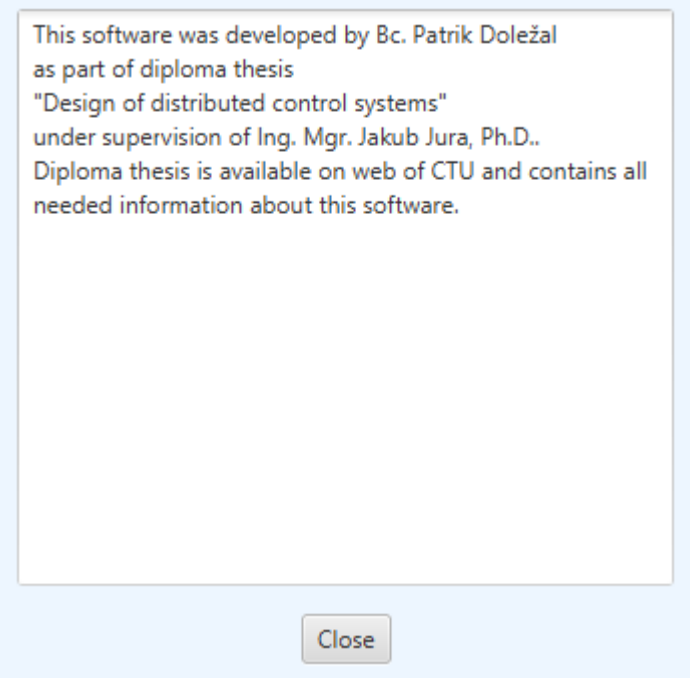
3.3.5 Menu

V horním menu jsou celkem tři možnosti Create/Open project, Open project folder a About. Všechny možnosti je třeba po rozkliknutí potvrdit tlačítkem, které se objeví pod tímto menu, protože SceneBuilder nenabízí možnost přiřazení akce přímo na tlačítko v liště menu, nicméně to alespoň zamezí případným překlepům.

Create/Open project nabízí přepnutí do úvodní obrazovky, kde je možné vytvořit nový, nebo otevřít jiný existující projekt, případně změnit složku knihovny.

Open project folder otevře složku projektu, na kterém momentálně pracujeme v softwaru, ve výchozím prohlížeči počítače.

About otevře pole s krátkým textem o programu a odkazem na tuto diplomovou práci.



Obrázek 3.22: Okno About

3.4 Realizace distribuovaného řídicího systému

Nyní je třeba celý stroj zprovoznit. Je třeba si prohlédnout harwarové zapojení a vytvořit potřebnou dokumentaci, díky čemuž bude možné upravit Class diagram. Následně je třeba hardwarovou konfiguraci nahrát do jednotlivých PLC, vytvořit program a vizualizaci pro SCADA server. Závěrečným krokem je test všech součástí systému.

3.4.1 Hardwarové zapojení

Veškeré členy systému bylo samozřejmě třeba zapojit, ale pro účel této práce předpokládejme, že hardware byl již zapojen, protože zákazník si objednal pouze softwarové řešení svého stroje, a není k dispozici žádná dokumentace.

Prvním zjištěním je, že stroj je ovládán třemi PLC Siemens S7-1200, přičemž jedno z nich využívá Siemens LOGO! jako vzdálené vstupy a výstupy. Ovládací panel navíc obsahuje dotykový monitor, kde běží SCADA server. Zkontrolujeme tedy pneumatické i elektrické zapojení a funkčnost jednotlivých komponent, zde je třeba si dát pozor na koncové senzory motorů, protože bylo objeveno hned několik nefunkčních.

Nejprve si vytvoříme tabulku, kde bude určeno jakým rozvaděčem je ovládán který motor a jakou funkci motor plní. Motory rovnou pojmenujeme podle abecedy v pořadí v jakém byly zmíněny v objednávce a rozvaděče označíme zkratkami, které jsou použité v knihovně vytvořeného softwaru.

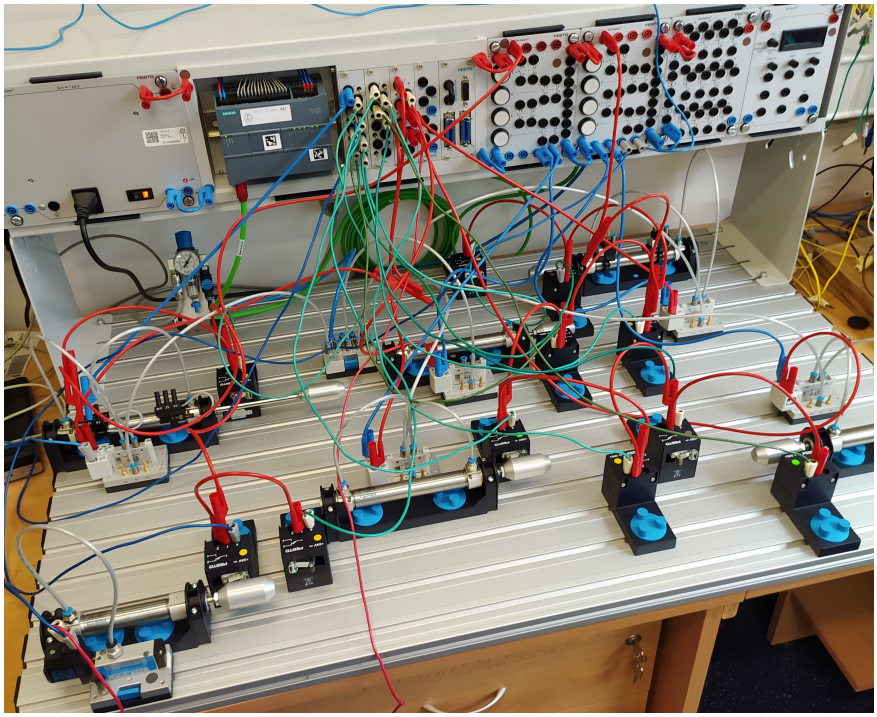
Dále vytvoříme další tabulku, kde budou uvedené vstupy a výstupy jednotlivých PLC společně s adresou a označením.

Motor	Rozvaděč	Operace
A	DAMS_52_NC	Upínání
B	SAMS_32_NC	Upínání
C	DAMS_52_NC	Stříhání - 1. pár
D	DAMS_52_NC	Stříhání - 1. pár
E	DAMS_52_NC	Stříhání - 2. pár
F	DAMS_52_NC	Stříhání - 2. pár
G	DAMS_52_NC	Ohýbání
H	DABS_52	Ohýbání
I	DAMS_53	Ohýbání
J	DABS_42	Vrtání
K	DABS_42	Vrtání

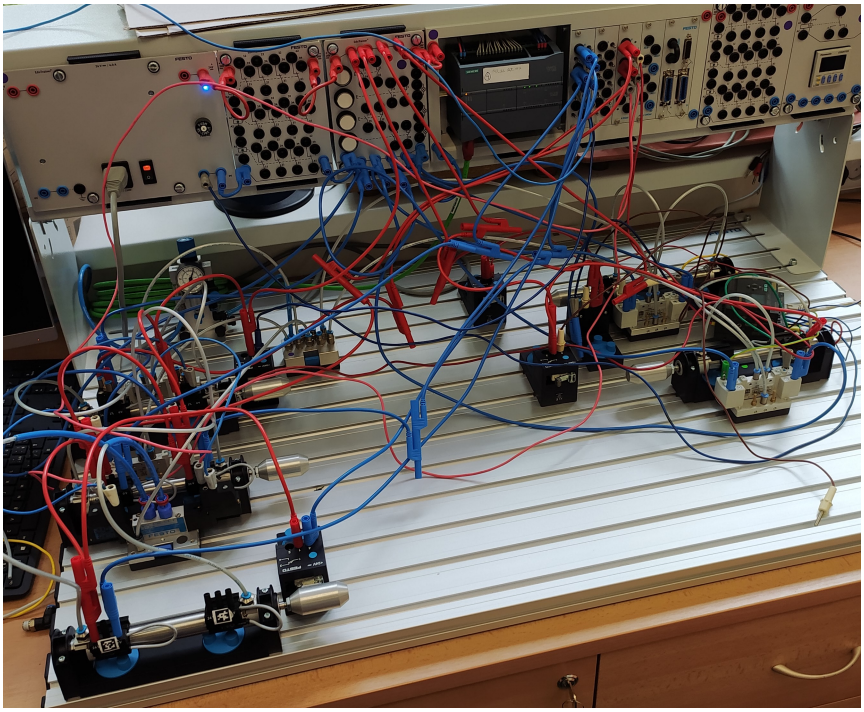
Tabulka 3.2: Přiřazení rozvaděčů k motorům

PLC	Označení	Adresa	PLC	Označení	Adresa
PLC1	A+	Q0.0	PLC2	G0	I0.5
PLC1	A0	I0.0	PLC2	G1	I0.4
PLC1	A1	I0.1	PLC2	H+	Q0.2
PLC1	B+	Q0.1	PLC2	H-	Q0.1
PLC1	B0	I0.2	PLC2	H0	I0.2
PLC1	B1	I0.3	PLC2	H1	I0.3
PLC1	C+	Q0.2	PLC2	I+	Q0.4
PLC1	C0	I0.4	PLC2	I-	Q0.3
PLC1	C1	I0.5	PLC2	I0	I0.1
PLC1	D+	Q0.3	PLC2	I1	I0.0
PLC1	D0	I0.6	PLC2	Tlačítko RESET	I1.0
PLC1	D1	I0.7	PLC2	Senzor Člověk	I1.1
PLC1	E+	Q0.4	PLC2	Dioda PLC2 Done	Q0.5
PLC1	E0	I1.0	PLC2	Dioda Člověk Varování	Q0.6
PLC1	E1	I1.1	PLC2	Dioda Člověk Zastavení	Q0.7
PLC1	F+	Q0.5	PLC3	J+	Q1
PLC1	F0	I1.2	PLC3	J-	Q2
PLC1	F1	I1.3	PLC3	J0	I1
PLC1	Dioda Obrobek	Q0.6	PLC3	J1	I2
PLC1	Dioda PLC1 Done	Q0.7	PLC3	K+	Q3
PLC1	Senzor Obrobku	I1.4	PLC3	K-	Q4
PLC1	Emergency stop	I1.5	PLC3	K0	I3
PLC2	G+	Q0.0	PLC3	K1	I4

Tabulka 3.3: List vstupů a výstupů



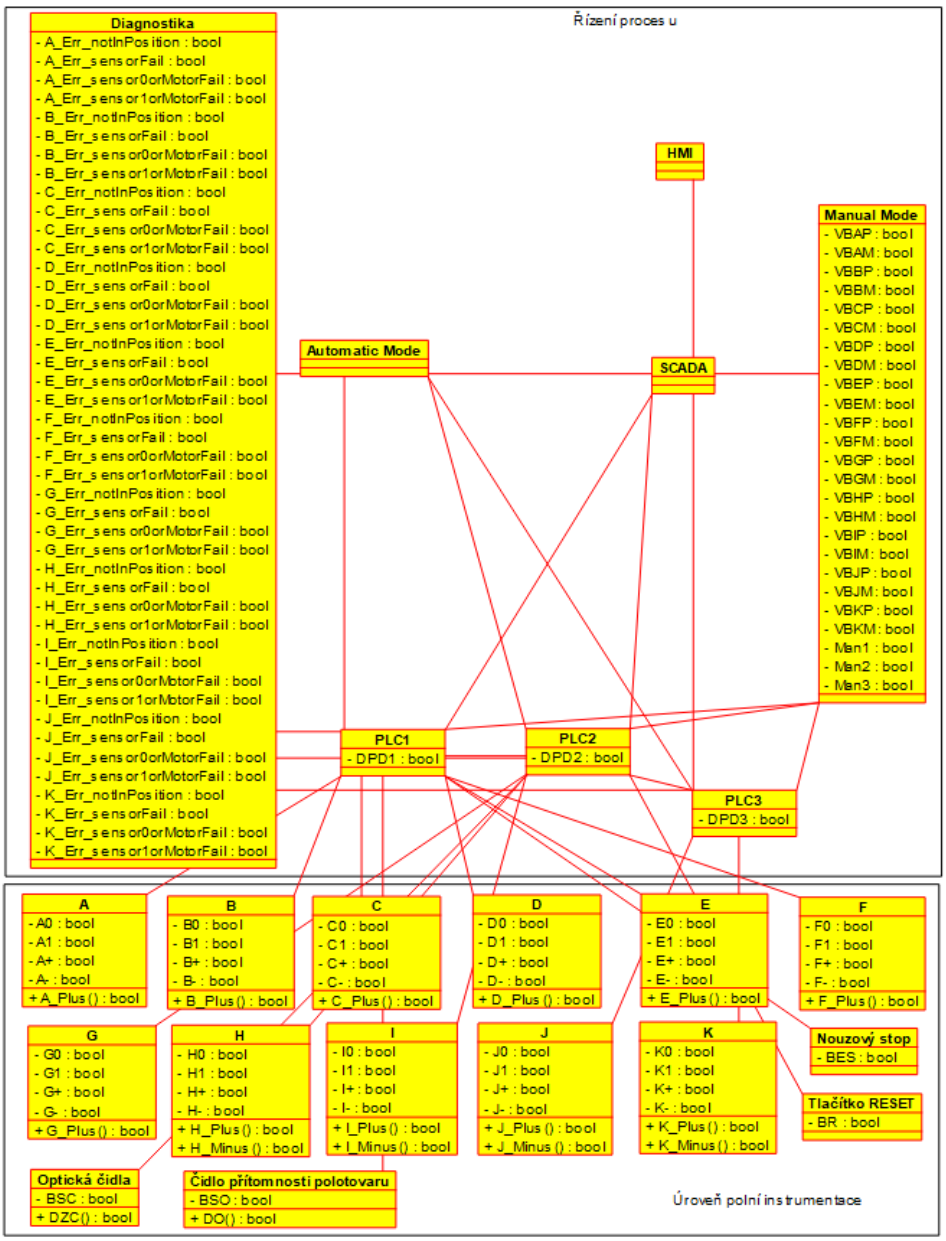
Obrázek 3.23: Zapojení PLC1



Obrázek 3.24: Zapojení PLC2 a PLC3

3.4.2 Úprava class diagramu

Protože již proběhlo základní seznámení se strojem, tak můžeme upravit Class diagram dle skutečné podoby stroje.



Obrázek 3.25: Konečná podoba class diagramu

Kvůli příjemnějšímu generování kódu v softwaru DOCOGE muselo být porušeno názvosloví v operacích a atributech pneumatických pístů. Zkratky v atributech třídy Manual Mode popisují tlačítka pro manuální ovládání motorů (VB - virtual button, x - označení pístu, P - plus, M - mínus), ostatní jsou již popsané v předchozí kapitole.

3.4.3 Hardwarová konfigurace

Vzájemnou komunikaci PLC je třeba nadefinovat i softwarově, nestačí pouze fyzické propojení přes průmyslovou sběrnici.

Po vytvoření projektu v softwaru TIA Portal tedy přidáme všechna PLC. V našem případě tedy 3x SIMATIC S7-1200, CPU verze 1215C DC/DC/DC 6ES7 215-1AG40-0XB0 a vybereme nejnovější dostupnou verzi (což je momentálně V4.2).

Vzájemná komunikace se nastavuje v sekci *Devices & networks*, nejprve je třeba přiřadit všechna PLC na stejnou subnet, což znamená že jsou na stejné síti propojené ethernetovým kabelem. Protože se toto nastavuje přímo v nastavení PLC, tak je výhodné rovnou přiřadit každému PLC svoji IP adresu, povolit PUT/GET komunikaci pro výměnu informací mezi PLC a aktivovat hodinové paměťové bity, které spínají s danými frekvencemi a velice často je pro ně užitečné využití.

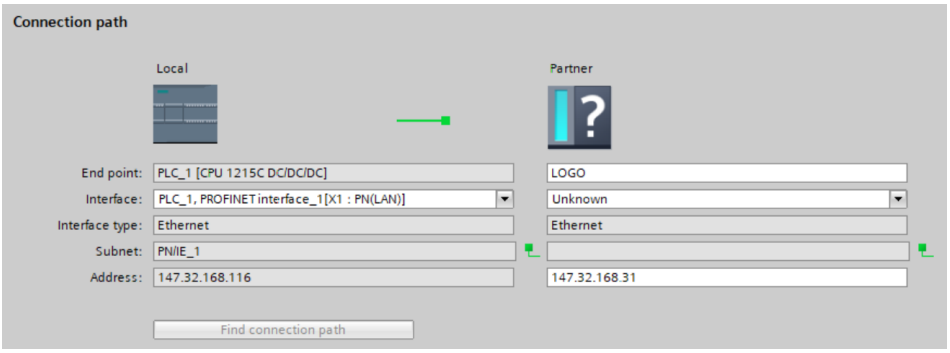
Nyní je třeba PLC spojit pomocí S7 Connection, což se provádí pravým kliknutím na dané PLC a výběrem *Add new connection* po výběru partnera je třeba zadat ID tohoto spojení, které budeme následně potřebovat přímo v PUT/GET blocích. Tento krok je nutné provést pro všechny dvojice PLC a je dobré zvolit přehledné ID spojení.

Network overview | Connections | I/O communication | VPN | TeleControl

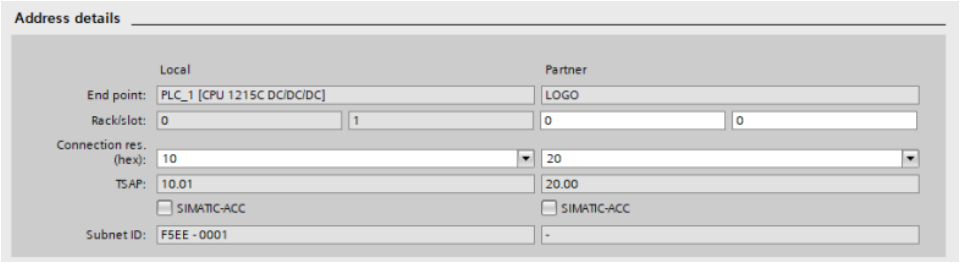
Local connection name	Local end point	Local ID (hex)	Partner ID (hex)	Partner	Connection type
S7_Connection_1	PLC_1 [CPU 1215C ...	100	100	PLC_2 [CPU 1215C ...	S7 connection
S7_Connection_1	PLC_2 [CPU 1215C ...	100	100	PLC_1 [CPU 1215C ...	S7 connection
S7_Connection_2	PLC_1 [CPU 1215C ...	102	102	PLC_3 [CPU 1215C ...	S7 connection
S7_Connection_2	PLC_3 [CPU 1215C ...	102	102	PLC_1 [CPU 1215C ...	S7 connection
S7_Connection_3	PLC_2 [CPU 1215C ...	104	104	PLC_3 [CPU 1215C ...	S7 connection
S7_Connection_3	PLC_3 [CPU 1215C ...	104	104	PLC_2 [CPU 1215C ...	S7 connection

Obrázek 3.26: Definovaná S7 Connections

Tímto je nakonfigurovaná komunikace mezi PLC S7-1200, nicméně v tomto systému využívá PLC3 LOGO! jako vzdálené vstupy a výstupy, proto se zde musí definovat i toto spojení. Vytvoříme tedy stejným způsobem další S7 Connetion pro PLC3, ale tentokrát nebudeme definovat partnera spojení, protože TIA Portal nenabízí v katalogu LOGO!. Po vytvoření spojení je nutné přenastavit několik věcí v jeho nastavení. Nejprve změníme pro přehlednost jméno endpointu na LOGO a vyplníme jeho IP adresu. Následně přejdeme do sekce *Special connection properties*, kde je nutné odškrtnout *Active connection establishment*. Poslední změnu je nutné provést v *Address details*, kde pro lokální PLC odškrtneme *SIMATIC-ACC* a nastavíme hex adresy - pro LOGO! se musí tato adresa nastavit 20. Rovnou vytvoříme i datové bloky, kam bude LOGO! odesílat informace o vstupech a výstupech, protože budeme potřebovat vědět, jaké mají tyto bloky číslo. Vytvoříme tedy 2 datové bloky, jeden pro vstupy a jeden pro výstupy a v jejich vlastnostech odškrtneme *Optimized block access*, protože do takto optimalizovaných bloků není možné zapisovat ani z nich číst z jiného PLC.



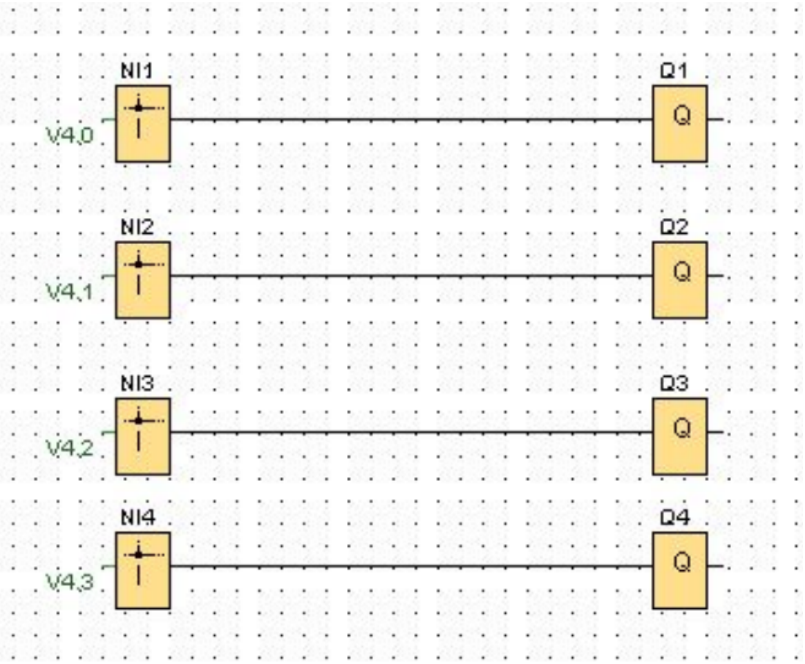
Obrázek 3.27: Sekce General S7 Connection pro komunikaci s LOGO!



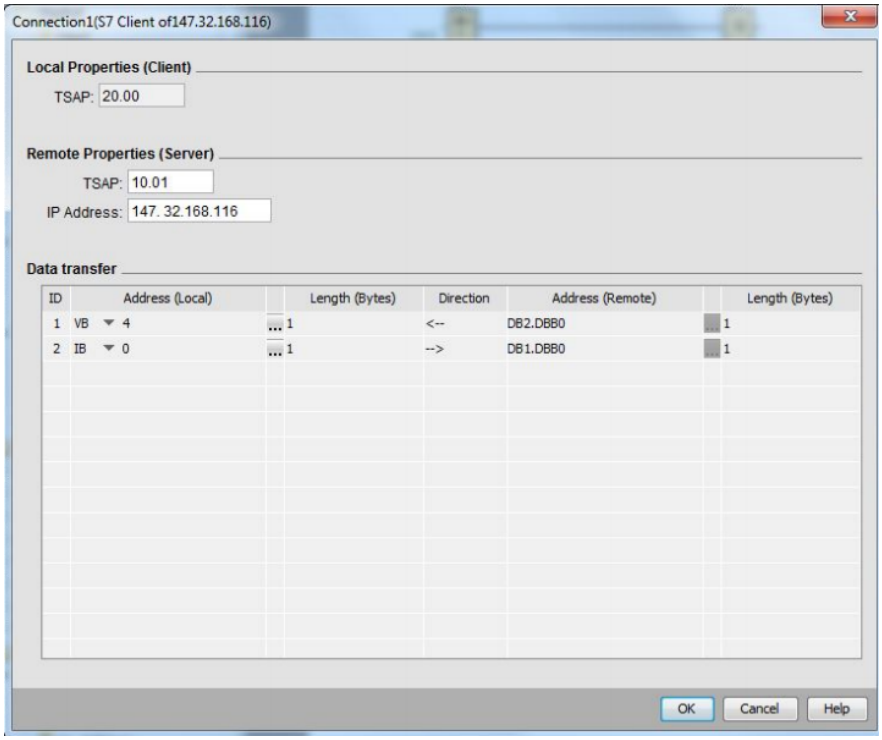
Obrázek 3.28: Sekce Address details S7 Connection pro komunikaci s LOGO!

Tímto je hardwarová komunikace pro všechna PLC S7-1200 hotová a je třeba ji nahrát do PLC přímo ze sekce *Devices & networks* kliknutím pravým tlačítkem na dané PLC a zvolením *Download hardware configuration*. Jedná se o důležitý krok, protože hardwarová konfigurace se do PLC nenahraje při nahrávání programu a bez ní nebudou jednotlivá spojení aktivní.

Posledním krokem v hardwarové konfiguraci je nastavení komunikace s PLC3 ze strany LOGO!. Tu je třeba nastavit v softwaru LOGO!Soft Comfort. Začneme vybráním správného typu LOGO! z katalogu a vytvoříme krátký program, kde pouze spojíme síťové vstupy s výstupy a přiřadíme jim patřičné adresy. Nakonec je třeba definovat samotné spojení v *Tools - Ethernet Connections*. Po vyplnění IP adresy a Subnet Mask našeho PLC LOGO! klikneme pravým tlačítkem myši na *Ethernet Connections - Add client connection - S7 Connection* a v zobrazeném okně vypníme IP adresu a TSAP PLC3 ado tabulky *Data Transfer* přidáme všechny údaje které chceme přenášet, tedy vstupy a výstupy.



Obrázek 3.29: Program pro LOGO!



Obrázek 3.30: Konfigurace spojení pro LOGO!

3.4.4 Tvorba programu

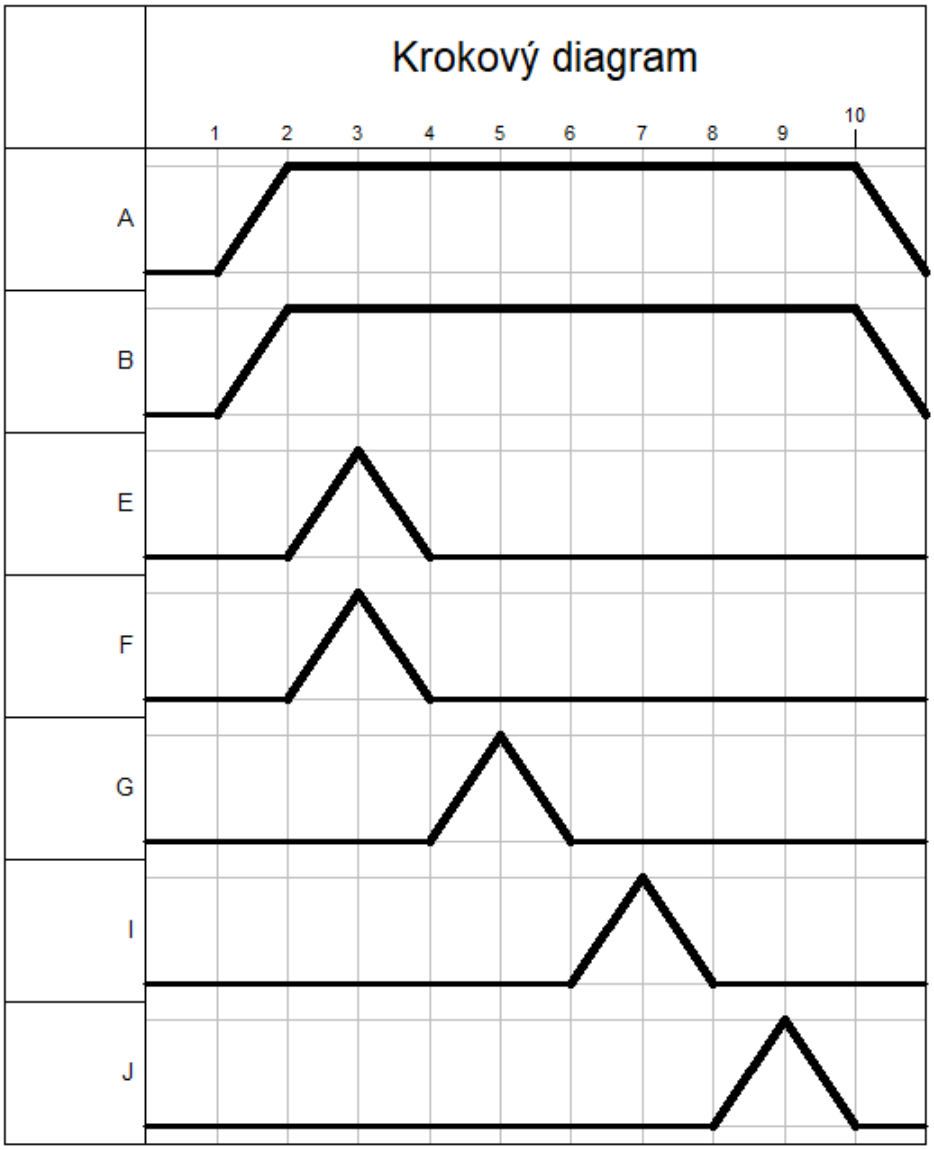
Zákazník si přeje se kromě softwaru na generování kódu také program pro jednu danou sekvenci stroje. V této sekvenci se plech nejprve upne, následně dojde ke střihu 2. párem na stříhání, ohnutí motorem G poté motorem I a vyvrtání díry pomocí motoru J.

Pro pojmenování proměnných tlačítek a diod použijeme zkratky (senzory obrobku a přítomnosti člověka byly nahrazeny tlačítky).

Celý název	Zkratka
Dioda Obrobek	DO
Dioda PLC1 Done	DPD1
Senzor Obrobku	BSO
Emergency Stop	BES
Tlačítko RESET	BR
Senzor Člověk	DSC
Dioda PLC2 Done	DPD2
Dioda Člověk Varování	DBC
Dioda Člověk Zastavení	DZC

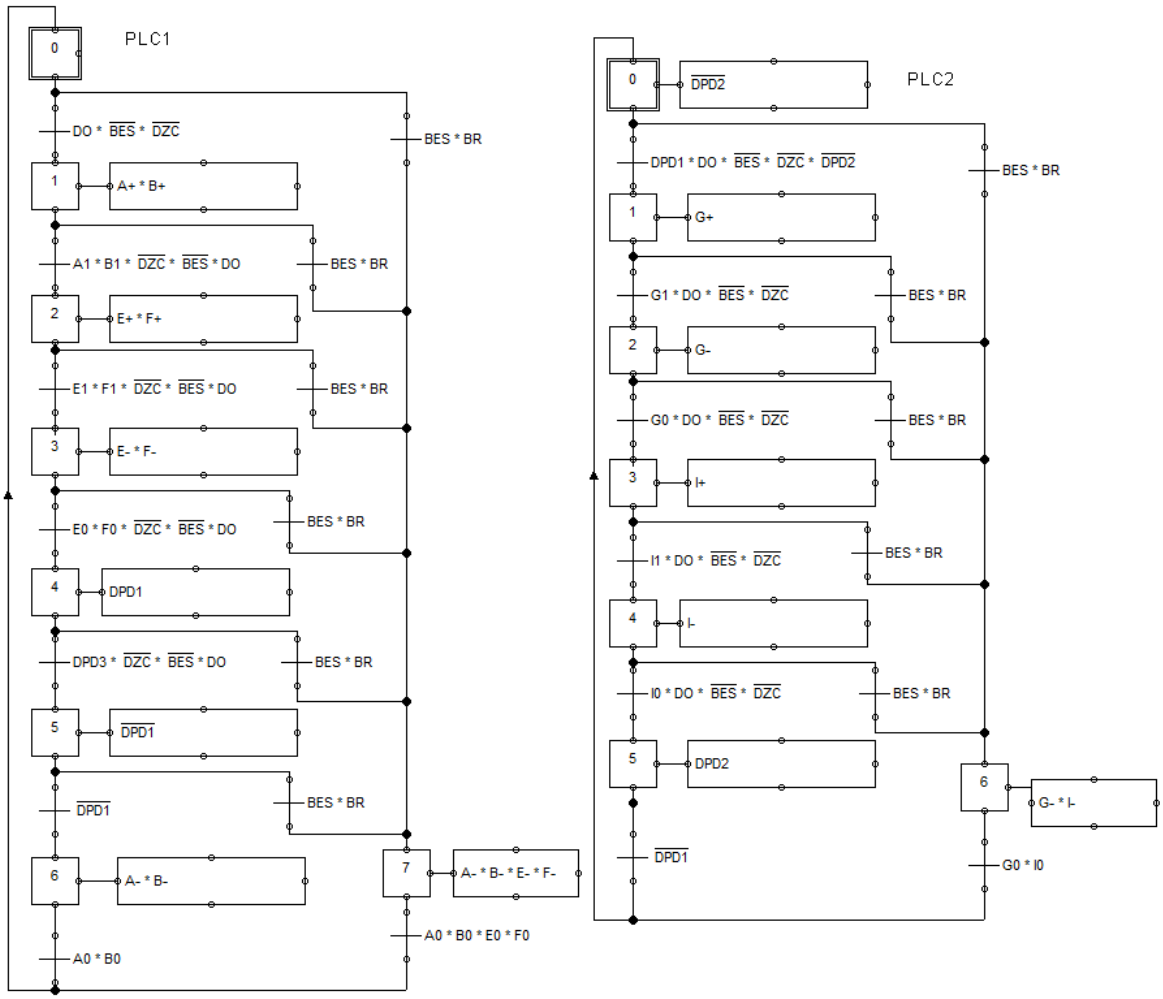
Tabulka 3.4: Použité zkratky pro tlačítka a diody

Prvním krokem je vytvoření krokového diagramu pro tuto sekvenci.

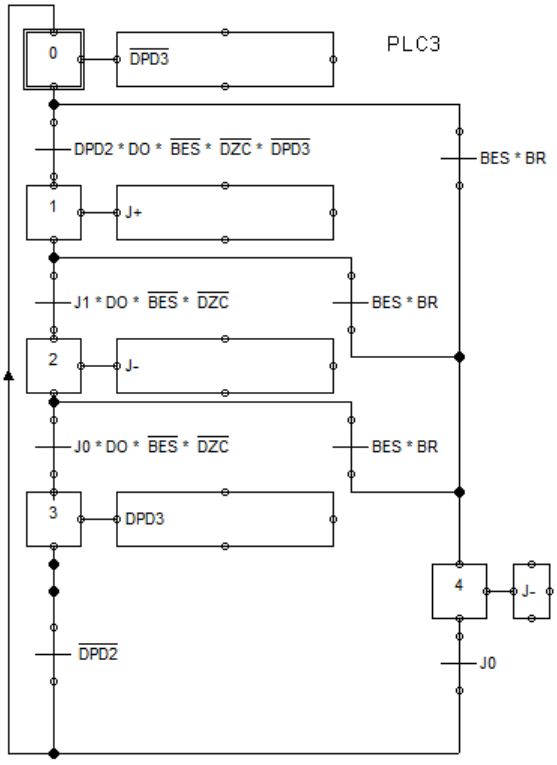


Obrázek 3.31: Krokový diagram pro zadanou sekvenci

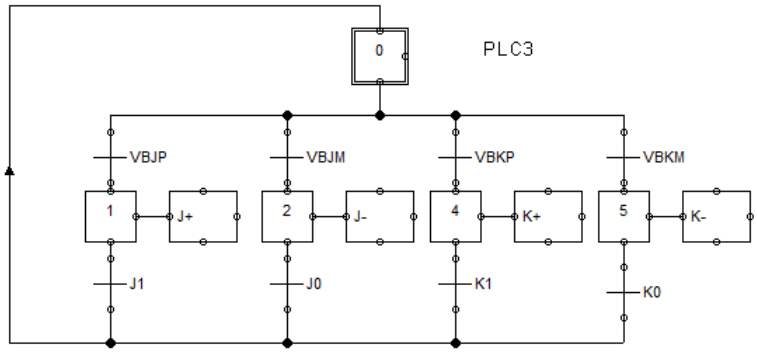
Následně využijeme krokový diagram pro vytvoření SFC pro všechna 3 PLC a také pro manuální ovládání jednotlivých motorů.



Obrázek 3.32: SFC pro zadanou sekvenci - PLC1 a PLC2



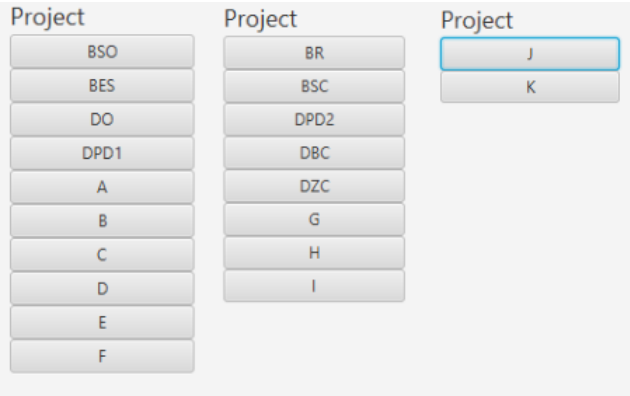
Obrázek 3.33: SFC pro zadanou sekvenci - PLC3



Obrázek 3.34: SFC pro manuální ovládání motorů - PLC3

Nyní využijeme tyto diagramy v softwaru DOCOGE, jehož fuknce byla již vysvětlena v předchozích kapitolách, takže příprava této sekvence zde bude popsána stručně.

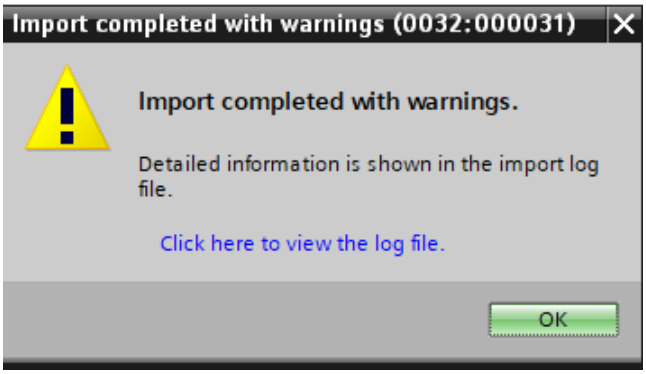
Po založení projektové složky přidáme použité komponenty na daném PLC, rozvaděče přiřadíme k motorům pomocí tabulky 3.1.



Obrázek 3.35: DOCOGE použité komponenty pro PLC1, PLC2 a PLC3

Následně vytvoříme pro každé PLC 2 procesy, jeden pro zadanou sekvenci a jeden pro manuální ovládání motorů. DOCOGE využívá k vytvoření procesů textové zadaní SFC, takže stačí využít diagramů z obrázků 3.30 a 3.31.

Vygenerované soubory nahrajeme do TIA Portal projektu k příslušným PLC. Pro lepší přehlednost je lepší začít tag table, čímž definujeme veškeré použité proměnné. V opačném případě by se bloky vygenerovaly, nicméně by zde byly zmíněny všechny nedefinované proměnné a bylo by těžké lokalizovat chyby, které způsobí problémy v kódu. Vytvoříme tedy novou tag table a importujeme tagy z vygenerovaného excelovského souboru. Po úspěšném importu vyběhne varování, v jehož detailech nalezneme informaci, že nebyla definována ID verze proměnných a TIA Portal jim tedy přidělil nejnovější, což je uděláno záměrně a nemusíme se tedy tímto varováním zabývat.



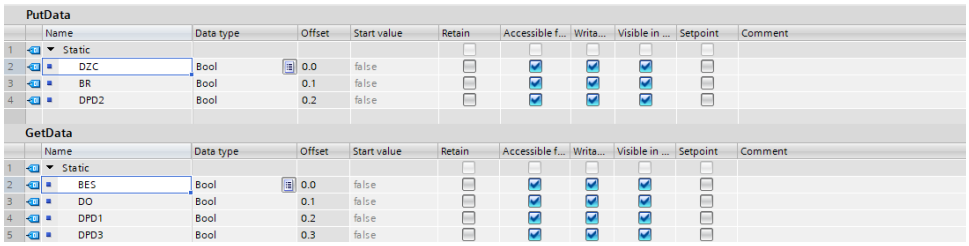
Obrázek 3.36: Varování po importu tag table

Bloky vygenerujeme v sekci *External source files*. I přesto, že máme definované proměnné se zde objeví chyby spojené s countery a timery a to z důvodu, že fukční bloky motorů byly generovány dříve než datové bloky timerů a counterů, což znamená že v čase generování funkčních bloků nebyly definované proměnné couterů a timerů. Timery a countery se uloží do podsložky *System blocks* a je nutné je přemístit do nadřazené složky *Program blocks* k ostatním blokům, protože jinak dojde při kompilaci k jejich smazání. Nejspíš se tak

děje kvůli shodě bloků a TIA Portal je vyhodnotí jako zbytečné duplikáty, nicméně bez nich nebudou funkční bloky fungovat, takže je nutné je takto "ochránit".

Po těchto krocích je třeba otevřít všechny tag table, kam jsme importovali proměnné, a manuálně jim přiřadit adresy, protože DOCOGE tuto fukci neobsahuje a ponechal proměnným adresu, která byla uvedena v jeho knihovně. Zároveň je třeba přidat vnitřní proměnné jako například DPD3 (Dioda PLC3 Done, která nebyla realizována jako dioda kvůli nedostatku výstupů LOGO!) a tlačítek pro manuální ovládání motorů (budou ovládány zze SCADA systému, takže není nutné vytvářet fyzické vstupy).

Před otestováním funkčnosti sekvence musíme udělat ještě pár kroků. Prvním je definování proměnných, které se budou přenášet mezi PLC. Na každém PLC vytvoříme 2 datové bloky, první se bude jmenovat PutData a vytvoříme v něm všechny proměnné, o jejichž stavu bude PLC odesílat informace, ve druhém vytvoříme všechny proměnné, o kterých bude PLC přijímat data a pojmenujeme ho GetData. V jejich vlastnostech je třeba opět odškrtnout *Optimized block access*, čímž si umožníme přesně popsat adresu proměnné v daném datovém bloku.


PutData

	Name	Data type	Offset	Start value	Retain	Accessible f...	Writa...	Visible in ...	Setpoint	Comment
1	Static									
2	DZC	Bool	0.0	false		✓	✓	✓		
3	BR	Bool	0.1	false		✓	✓	✓		
4	DPD2	Bool	0.2	false		✓	✓	✓		

GetData

	Name	Data type	Offset	Start value	Retain	Accessible f...	Writa...	Visible in ...	Setpoint	Comment
1	Static									
2	BES	Bool	0.0	false		✓	✓	✓		
3	DO	Bool	0.1	false		✓	✓	✓		
4	DPD1	Bool	0.2	false		✓	✓	✓		
5	DPD3	Bool	0.3	false		✓	✓	✓		

Obrázek 3.37: Datové bloky PutData a GetData na PLC2

Následně vytvoříme v každém PLC nový funkční blok pro bloky PUT a GET v jazyce LAD (můžeme samozřejmě zvolit jiný jazyk, ale bloky PUT a GET vypadají v grafickém provedení nejpřehledněji) a pojmenujeme ho putget. Pro předání informace o proměnné z jednoho PLC na druhé je třeba použít blok PUT v PLC, které informaci odesílá, a blok GET v PLC, které informaci přijímá. Jeden blok může posílat/přijímat informace o více proměnných, ale může se spojit pouze s jedním PLC, takže na každém PLC musíme použít 2 bloky PUT a 2 bloky GET. Blokům je samozřejmě nutné přiřadit vstupy a výstupy.

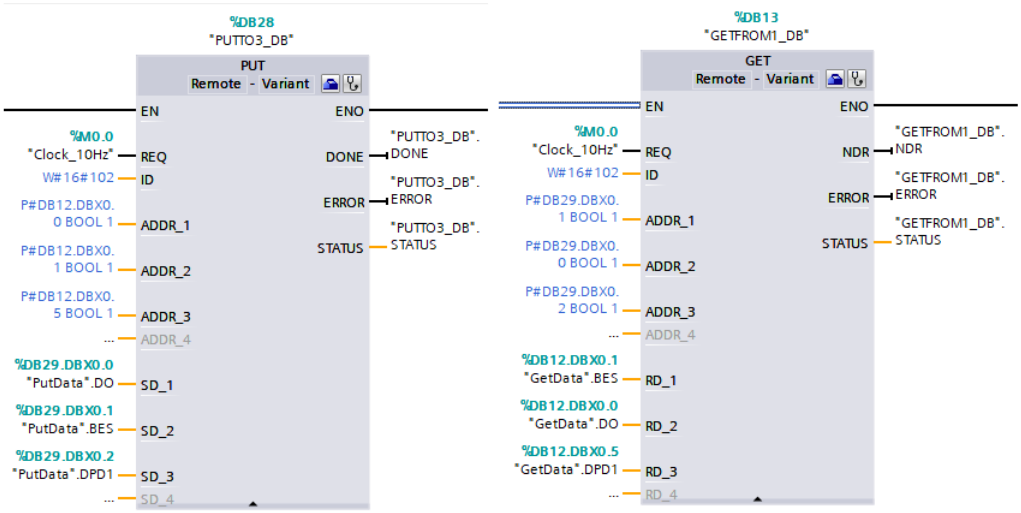
Vstupy:

- **REQ** - při náběžné hraně dochází k přenosu informací, proto je výhodné využít hodiný paměťový bit
- **ID** - zde je třeba uvést ID S7 connection s partnerským PLC
- **ADDR** - adresa partnerského PLC, kam bude toto PLC zapisovat v případě PUT bloku, nebo odkud bude PLC číst v případě GET bloku

- **SD** - adresa proměnné o které má PUT blok odesílat informaci v lokálním PLC
- **RD** - adresa proměnné o které má GET blok číst informaci v lokálním PLC

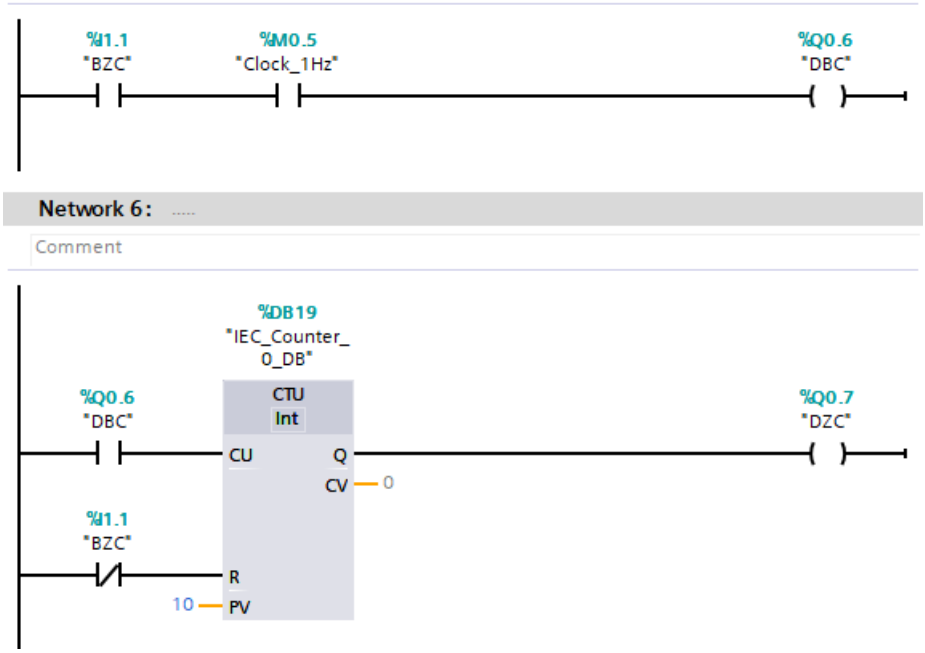
Výstupy (všem bývá obvykle přiřazena stejnojmenná proměnná z automaticky vytvořeného datového bloku během vložení bloku):

- **DONE** - aktivuje se pokud přenos proběhl v pořádku
- **ERROR** - aktivuje se při chybě při přenosu
- **STATUS** - pro specifické situace posílá kód, tyto kódy lze dohledat v nápovědě pro daný blok



Obrázek 3.38: PUT a GET blok umožňující PLC3 číst informace z PLC1

Další část těchto funkčních bloků využijeme pro propojení proměnných se stejnojmennými proměnnými, které jsou uloženy v datovém bloku PutData, a vytvoříme závislosti senzorů na diodách, které bylo zbytečné generovat v DOCOGE, tedy že senzor obrobku aktivuje diodu obrobku a nakonfigurujeme také senzor člověka v okolí stroje. Tento senzor rozbliká diodu člověk varování s frekvencí 1 Hertz a po 10 bliknutích této diody se rozsvítí dioda člověk zastavení.



Obrázek 3.39: Konfigurace senzoru přítomnosti člověka v blízkosti stroje

Následně je nutné projít kód sekvence, kde jsou chyby v názvech proměnných, které se nenacházejí na lokálním PLC, protože jsou umístěné v datových blocích, což se projeví na jejich značení v programu.

Posledním krokem je úprava organizačního bloku Main[OB1]. Tento blok se využívá k volání požadovaných funkcí, což v našem případě znamená volit mezi automatickým a manuálním provozem a k tomu vždy volat funkční blok putget pro výměnu potřebných informací mezi jednotlivými PLC a udržení funkčnosti bezpečnostních prvků.

```
1
2  IF "Man2" THEN
3      "Manualplc2_Proc"();
4  END_IF;
5
6  IF NOT "Man2" THEN
7      "Vzorplc2_Proc"();
8  END_IF;
9
10 "putget_DB"();
```

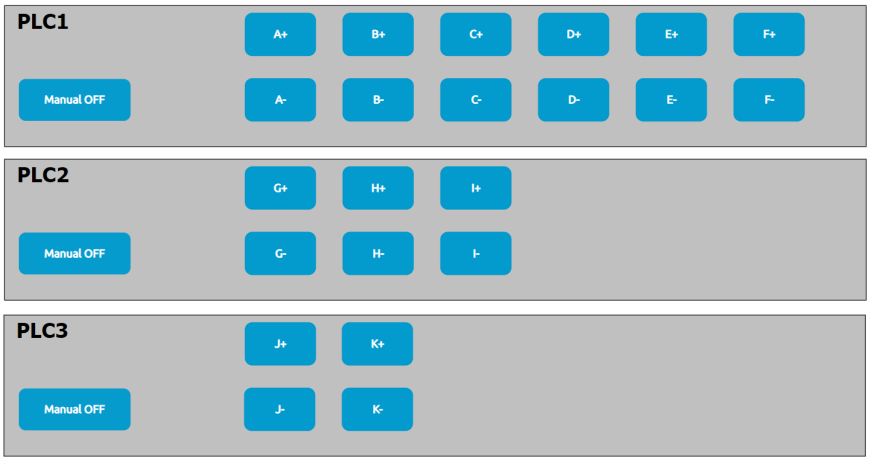
Obrázek 3.40: Organizační blok Main PLC2

3.4.5 SCADA

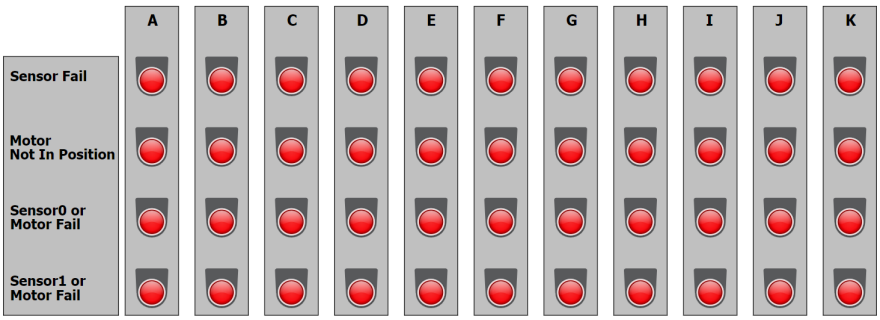
Součástí stroje je i dotykový monitor, kde je umístěn SCADA server, kam nahrajeme vizualizaci manuálního ovládání a diagnostiky. Z osobních preferencí byl zvolen software MyScada verze 8.

Protože zákazník definoval v objednávce že ne vždy je využíváno všech operací, tak se bude manuální ovládání zapínat na každém PLC zvlášť. Díky tomu bude mít stroj možnost například stále produkovat plechy, které nepotřebují vyvrtávat díry, a zároveň testovat PLC 3 v manuálním režimu.

Vytvoříme tedy projekt v programu MyDesigner8, připojíme všechna 3 PLC, přidáme všechny potřebné tagy a vytvoříme vizualizaci pro ovládání manuálního režimu a diagnostický přehled.



Obrázek 3.41: SCADA - vizualizační okno pro manuální ovládání



Obrázek 3.42: SCADA - vizualizační okno pro diagnostiku

3.4.6 Zkouška funkce systému

Nakonec je třeba systém řádně otestovat. První test je zkouška sekvence v automatickém režimu, která proběhla úspěšně bez výraznějších komplikací. Následně je třeba otestovat bezpečnostní prvky, tedy že emergency stop a přítomnost člověka v okolí stroje po určitém čase zastaví proces a rozvítí se příslušné diody. I tento test dopadl úspěšně. Posledním testem je zkouška manuálního režimu a dignostiky ve SCADA systému. Tento test musel být opakovám kvůli chybám v adresování tagů ve SCADA systému, ale nakonec proběhl také úspěšně.

Kapitola 4

Závěr

Tuto práci jsem začal rešerší na téma centralizovaných a distribuovaných řídicích systémů včetně popisu rozhodnutí kdy které architektury využít. Probrány byly také systémy multiagentní a holonické. Pokračoval jsem teoretickým rozborem komunikace v těchto systémech a to mezi všemi úrovněmi, což obsahuje přehled používaných sběrnic a standardů. Podíval jsem se i na možnosti distribuovaných vstupů a výstupů a nakonec stručně představil softwarový standard UML.

V praktické části jsem začal analýzou systému, který byl definován objednávkou zákazníka, jejímnž výstupem je Class diagram, který byl vytvořen ná základě lexikální analýzy zadání. V této podobě byl diagram nekompletní, protože zákazníkova objednávka nebyla dostatečným zdrojem informací, a proto byly dohledány/dovyžádány informace vedoucí k doplnění diagramu.

Dále jsem vytvořil software DOCOGE pro generování PLC kódu v jazyce SCL z popisu výrobního procesu v sekvenčním funkčním diagramu. Tento software dokáže bezchybně generovat kód, nicméně se nedá považovat hotový pro průmslové použití. Uživatelské rozhraní by se dalo dále vylepšovat, protože nezaškolený uživatel by nejspíš kód správně vygenerovat nedokázal. Některé chyby a překlepy nelze opravit a v některých momentech nelze práci uložit pro pozdější dodělání. Některé pomocné prvky (jako například text zobrazující dostupné vstupy a výstupy) by potřebovaly ještě odladit, protože jsou situace, kdy se zde objevují menší chyby. Dále by bylo možné do určité míry začlenit i přenos informací o proměnných mezi jednotlivými PLC a to alespoň vytvořením datových bloků, které obsahují potřebné proměnné. I přes tyto skutečnosti považuji vytvořený software za zdařilý, protože plní svou nejzákladnější funkci, což je bezchybné generování kódu pro PLC, a obsahuje i pomocné prvky pro uživatele jako je otevření projektové složky nebo přehled již vytvořených přechodů mezi jednotlivými stavy SFC.

Dále se v práci zabývám sestavením funkčního modelu výrobního procesu z elektropneumatických komponent. Sestavil jsem "zákazníkův stroj", vytvořil potřebnou dokumentaci, která obsahovala seznam použitých hardwarovýchh komponent a jejich zapojení. Připravil jsem hardwarovou konfiguraci řídicího systému, kde byla PLC spojena pomocí S7 Connection, což jim umožňuje výměnu informací. Pokračoval jsem vytvořením kódu pomocí softwaru DOCOGE a nakonfiguroval bloky PUT a GET, které byly použity pro výměnu

informací o proměnných. Změny přímo v generovaném kódu nebyly nijak rozsáhlé, ani časově náročné. Bylo také nutné vytvořit menší vizualizační projekt pro ovládání stroje v manuálním režimu a zobrazení diagnostiky jednotlivých elektropneumatických pohonů, jejíž výstupy budou pomocí SCADA serveru zobrazeny na dotykové obrazovce ovládacího panelu stroje.

Diplomovou práci považuji celkově za úspěšnou protože jsem dokázal vytvořit funkční software pro generování PLC kódu. Počítal jsem s tím, že nebude kompletně připraven pro průmyslové využití, protože na takových softwarech pracují vícečlené týmy programátorů a mají delší časový horizont pro kompletní odladění. Navržený řídicí systém funguje správně a to včetně diagnostiky a bezpečnostních prvků.

Kapitola 5

Použité zdroje

[1] SOMMERVILLE, Ian. Centralized Control [online]. 2008 [cit. 2020-02-12]. Dostupné z: https://ifs.host.cs.st-andrews.ac.uk/Books/SE9/Web/Architecture/ArchPatterns/CentralControl.html

[2] Distributed control system (DCS). Whatis.com [online]. [cit. 2019-09-15]. Dostupné z: https://whatis.techtarget.com/definition/distributed-control-system

[3] What is Distributed Control System (DCS)? ELECTRICAL TECHNOLOGY [online]. [cit. 2020-08-17]. Dostupné z: https://www.electricaltechnology.org/2016/08/distributed-control-system-dcs.html

[4] Choosing between centralized and distributed control system designs. Control Engineering [online]. [cit. 2020-08-17]. Dostupné z: https://www.controleng.com/articles/choosing-between-centralized-and-distributed-control-system-designs/

[5] The Radical Transformation from Centralized to Distributed Network Systems. In: Medium [online]. 2017 [cit. 2020-04-28]. Dostupné z: https://medium.com/@sushmita.kalashikar/the-radical-transformation-from-centralized-to-distributed-network-systems-93d3a3c008d6

[6] Multiagentní systémy (MAS). Https://is.mendelu.cz [online]. [cit. 2019-09-15]. Dostupné z: https://is.mendelu.cz/eknihovna/opory/zobraz_cast.pl?cast=32363

[7] MAŘÍK, Vladimír, Olga ŠTĚPÁNKOVÁ a Jiří LAŽANSKÝ. Umělá inteligence. Praha: Academia, 1993-. ISBN 80-200-0504-8.

[8] Úvod do problematiky multiagentních systémů [online]. Ing. Arnoštka Netrvalová, Ph.D., ZČÚ v Plzni, 28 [cit. 2019-09-28]. Dostupné z: https://www.kiv.zcu.cz/ netrvalo/phd/MAS.pdf

[9] Multiagentní řízení, simulace a plánování výroby. Automa [online]. 2004 [cit. 2020-04-28]. Dostupné z: https://automa.cz/cz/casopis-clanky/multiagentni-rizeni-simulace-a-planovani-vyroby-2004_05_32334_1681

[10] MAŘÍK, Vladimír, Olga ŠTĚPÁNKOVÁ a Jiří LAŽANSKÝ. Umělá inteligence. Praha: Academia, 1993-. ISBN 80-200-1044-0.

[11] Holonické systémy [online]. , 3 [cit. 2019-09-28]. Dostupné z: https://vendulka.zcu.cz/Download/Free/IRS2/01/Holonicke%20systemy.pdf

[12] MANTLE, Jakes. The 5 Layers of the Automation Pyramid and Manufacturing Operations Management. In: SYSPRO [online]. 2019 [cit. 2020-04-28]. Dostupné z: https://www.syspro.com/blog/erp-for-manufacturing/the-5-layers-of-the-automation-pyramid-and-manufacturing-operations-management/

[13] What is IO-Link? IO-Link [online]. [cit. 2020-04-28]. Dostupné z: https://www.io-link.com/en/Technology/what_is_IOLink.php?thisID=76

[14] IO-Link – popis digitální komunikace pro senzory. HW [online]. [cit. 2020-04-28]. Dostupné z: https://automatizace.hw.cz/iolink-popis-digitalni-komunikace-pro-senzory

[15] Nová generace technologie AS-Interface. PEPPERL+FUCHS [online]. [cit. 2020-08-17]. Dostupné z: https://www.pepperl-fuchs.com/czech_republic/cs/ASi-5.htm

[16] Průmyslová sběrnice AS-Interface. Automa [online]. [cit. 2020-08-17]. Dostupné z: https://automa.cz/cz/casopis-clanky/prumyslova-sbernice-as-interface-2001_02_33464_2573/

[17] AS-Interface Special. JOKARI [online]. [cit. 2020-08-17]. Dostupné z: https://www.jokari.de/en/ASInterface-Special-2.htm

[18] PLC Input Output Modules. HW [online]. [cit. 2020-04-28]. Dostupné z: https://instrumentationtools.com/plc-inputoutput-modules/

[19] Does distributed I/O architecture make sense? Control design [online]. 2017 [cit. 2020-04-28]. Dostupné z: https://www.controldesign.com/articles/2017/does-distributed-io-architecture-make-sense/

[20] Profinet versus Profibus. Automa [online]. 2012 [cit. 2020-04-28]. Dostupné z: https://automa.cz/cz/casopis-clanky/profinet-versus-profibus-2012_05_0_9618/

[21] WHAT IS THE DIFFERENCE BETWEEN PROFIBUS AND PROFINET? RealPars [online]. [cit. 2020-04-28]. Dostupné z: https://realpars.com/difference-between-profibus-and-profinet/

[22] BYSTRICANOVA, Anna a Andrej RYBOVIC. DATA COMMUNICATION BETWEEN PROGRAMMABLE LOGIC CONTROLLERS IN THE INDUSTRIAL DISTRIBUTION APPLICATIONS [online]. June 2011, , 7 [cit. 2020-04-28]. Dostupné z: https://www.researchgate.net/publication/266162472_Data_Communication_between_Programmable_Logic_Controllers_in_the_Industrial_Distribution_Applications/fulltext/54ad4cf30cf24aca1c6ef9e8/Data-Communication-between-Programmable-Logic-Controllers-in-the-Industrial-Distribution-Applications.pdf

[23] What is OPC? OPC foundation [online]. [cit. 2020-04-28]. Dostupné z: https://opcfoundation.org/about/what-is-opc/

[24] Unified Architecture. OPC foundation [online]. [cit. 2020-04-28]. Dostupné z: https://opcfoundation.org/about/opc-technologies/opc-ua/

[25] OPC UA and PROFINET. Profinet university [online]. [cit. 2020-04-28]. Dostupné z: https://profinetuniversity.com/industrial-automation-ethernet/opc-ua-profinet/

[26] What is Unified Modeling Language (UML)? Visual Paradigm [online]. [cit. 2020-08-06]. Dostupné z: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-uml/

[27] UML Diagrams: Versions, Types, History, Tools, Examples. GURU99 [online]. [cit. 2020-08-06]. Dostupné z: https://www.guru99.com/uml-diagrams.html

[28] What is Object Diagram? Visual Paradigm [online]. [cit. 2020-08-06]. Dostupné z: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-object-diagram/

[29] What is Package Diagram? Visual Paradigm [online]. [cit. 2020-08-06]. Dostupné z: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-package-diagram/

[30] What is Component Diagram? Visual Paradigm [online]. [cit. 2020-08-06]. Dostupné z: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-component-diagram/

[31] What is Deployment Diagram? Visual Paradigm [online]. [cit. 2020-08-06]. Dostupné z: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-deployment-diagram/

[32] What is Activity Diagram? Visual Paradigm [online]. [cit. 2020-08-06]. Dostupné z: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-activity-diagram/

[33] What is Use Case Diagram? Visual Paradigm [online]. [cit. 2020-08-06]. Dostupné z: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-use-case-diagram/

[34] What is State Machine Diagram? Visual Paradigm [online]. [cit. 2020-08-06]. Dostupné z: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-state-machine-diagram/

[35] What is Timing Diagram? Visual Paradigm [online]. [cit. 2020-08-06]. Dostupné z: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-timing-diagram/

[36] What is Sequence Diagram? Visual Paradigm [online]. [cit. 2020-08-06]. Dostupné z: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-sequence-diagram/

[37] What is UML Collaboration Diagram? Visual Paradigm [online]. [cit. 2020-08-06]. Dostupné z: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-uml-collaboration-diagram/

Kapitola 6

Seznam použitých zkratek

AS-I	Actuator Sensor Interface
CCS	Centralized Control System
DABS_42	Double Acting Bistable 4/2
DABS_52	Doble Acting Bistable 5/2 Normally Close
DAMS_52_NC	Single Acting Monostable 3/2 Normally Close
DAMS_53	Double Acting Bistable 5/3
DCS	Distributed Control System
ERP	Enterprise Resource Planning
HMI	Human Machine Interface
I/O	Input/Output
OPC Classic	OLE(Object Linking and Embedding) for Process Control
OPC UA	Open Platform Communication Unified Architecture
PLC	Programmable Logic Controller
SAMS_32_NC	Single Acting Monostable 3/2 Normally Close
SCADA	Supervisory Control And Data Acquisition
SFC	Sequential Function Chart
UML	Unified Modeling Language