

Monitorování dat z PLC využitím OPC serveru s OPC klientem

Data Acquisition from the PLC by Using OPC Server and OPC
Client

Bc. Roman Baťa

Diplomová práce
2014

 Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
akademický rok: 2013/2014

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení: **Bc. Roman Baťa**
Osobní číslo: **A12453**
Studijní program: **N3902 Inženýrská informatika**
Studijní obor: **Informační technologie**
Forma studia: **kombinovaná**

Téma práce: **Monitorování dat z PLC využitím OPC serveru s OPC klientem**

Zásady pro vypracování:

1. Zpracujte literární rešerši na problematiku týkající se OPC komunikace.
2. Navrhněte OPC klienta v prostředí Microsoft Visual Studio a proveďte jeho realizaci. Popište jednotlivé části vytvořené aplikace.
3. Popište výrobní linku, na které bude vytvořená aplikace využita.
4. Proveďte úpravy PLC programu pro čtení a zápis požadovaných dat, která budou využita při OPC komunikaci. Ověřte možnost propojení OPC klienta s vizualizačním SW WinCC.
5. Navrhněte a popište způsob práce s daty tj. způsob jejich volby, uložení a vyhodnocení. Daný postup ilustруйте na konkrétním příkladu.
6. Ověřte komunikaci OPC na výrobní lince s využitím vytvořené aplikace.

Rozsah diplomové práce:

Rozsah příloh:

Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam odborné literatury:

1. BERGER, H. Automatizace se STEPem 7 v AWL. Munich : Publicis MCD Werbeagentur GmbH, 1998. 440 s. Volně programovatelné automaty SIMATIC S7-300/400. ISBN 3-89578-089-8.
2. BLAŽEK, Jaroslav. BLAJA Automation Portal, Průmyslová automatizace, projekty, odborné texty [online]. 2014 [cit. 2014-01-10]. Dostupné z WWW: [http://www.blaja.cz].
3. MAHNKE, Wolfgang; LEITNER, Stefan Helmut; DAMM, Matthias. OPC Unified Architecture. Berlín: Springer, 2009. 362 s. ISBN 978-3-540-68898-3.
4. OPC FOUNDATION. The OPC Foundation - Unified Architecture [online]. 2014 [cit. 2014-01-10]. Dostupný z WWW: [http://www.opcfoundation.org]
5. SIEMENS A.G. WinCC - Examples of integrated engineering with STEP7 [pdf elektronický manuál]. Nurnberg: Siemens, release 4/2009 [cit. 2014-1-10]. Dostupný z WWW: [http://www.siemens.com].
6. SHARP, John. Microsoft Visual C 2010: krok za krokem. 1.vydání. Brno: Computer Press a.s., 2010. 696 s. ISBN 978-8-025-13147-3.
7. ŠÍMA, František; VILÍMEK, David. Visual Studio .NET: praktické programování krok za krokem. Grada Publishing a. s., 2006. 254 s. ISBN 978-8-024-71418-9.

Vedoucí diplomové práce:

Ing. Pavel Navrátil, Ph.D.

Ústav automatizace a řídicí techniky

Datum zadání diplomové práce:

21. února 2014

Termín odevzdání diplomové práce:

20. května 2014

Ve Zlíně dne 21. února 2014


prof. Ing. Vladimír Vašek, CSc.
děkan




doc. Mgr. Roman Jašek, Ph.D.,
ředitel ústavu

ABSTRAKT

Tato diplomová práce se zabývá použitím OPC rozhraní v prostředí SIMATIC.NET pro komunikaci se zařízením PLC a zpracováním dat na straně klienta v prostředí Microsoft Visual Studio. Pomocí jeho standardizovaných rozhraní je ukázána univerzálnost OPC komunikace. Teoretická část obsahuje základní popis OPC a popis použité technologie. Praktická část obsahuje úpravu stávajícího programu PLC s vizualizací WinCC a vytvoření klientské aplikace. Na závěr je otestována vytvořená aplikace na konkrétních příkladech. Výsledná realizace klienta by měla umožnit získat více znalostí o chování stroje, které mohou být dále využity ke zkvalitnění výstupů.

Klíčová slova: Microsoft Visual C#, OPC, PLC, STEP7, SIMATIC NET, WinCC.

ABSTRACT

This thesis deals with the use OPC interface in SIMATIC.NET environment to communicate with the PLC and data processing on the client side in Microsoft Visual Studio environment. By using its standardized interface is shown versatility OPC communication. The theoretical part provides a basic description OPC and description of the applied technology. The practical part contains a modification of an existing PLC program with WinCC visualization and creation of client application. Finally, created application is tested on the concrete examples. The resulting client implementation should allow to gain more knowledges about the behavior of the machine, which can be further used to improve outcomes.

Keywords: Microsoft Visual C#, OPC, PLC, STEP7, SIMATIC NET, WinCC.

Poděkování

Předně bych chtěl poděkovat vedoucímu práce, kterým je Ing. Pavel Navrátil, Ph.D za jeho podporu a mnoho cenných rad při vedení diplomové práce. Dále bych chtěl poděkovat společnosti Mitas Otrokovice, která mně umožnila vývoj aplikace včetně nasazení do provozu. Poděkovat bych chtěl i své rodině, přátelům a kolegům za podporu a trpělivost.

Prohlašuji, že

- beru na vědomí, že odevzdáním diplomové/bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že diplomová/bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk diplomové/bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji diplomovou/bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – diplomovou/bakalářskou práci nebo poskytnout licenci k jejímu využití jen s předchozím písemným souhlasem Univerzity Tomáše Bati ve Zlíně, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše);
- beru na vědomí, že pokud bylo k vypracování diplomové/bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky diplomové/bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem diplomové/bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na diplomové práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně

.....
podpis diplomanta

OBSAH

ÚVOD.....	9
I TEORETICKÁ ČÁST	10
1 ANALÝZA ZADÁNÍ A POPIS VÝROBNÍ LINKY ROLLER HEAD.....	11
1.1 ANALÝZA ZADÁNÍ	11
1.2 POPIS VÝROBNÍ LINKY ROLLERHEAD	12
2 ZÁKLADNÍ POPIS OPC	15
2.1 PŘEDSTAVENÍ OPC	15
2.2 NOVÁ SPECIFIKACE OPC UA	16
2.2.1 Architektura OPC UA	18
2.3 S7 KOMUNIKACE	20
2.3.1 Variabilní služby	21
2.3.2 Blokové služby.....	24
2.3.3 Namespace a NodeId.....	25
2.3.4 Rozhraní pro přístup do adresového prostoru	26
2.3.5 Adresování OPC UA serveru	26
2.4 TRANSPORTNÍ PROTOKOLY	27
2.4.1 UA TCP.....	28
2.4.2 SOAP/HTTP	29
3 POPIS POUŽITÉ TECHNOLOGIE.....	31
3.1 HARDWARE	31
3.1.1 PLC SIMATIC S7-400.....	32
3.1.2 CP 443-1.....	33
3.1.3 PC stanice.....	34
3.2 SOFTWARE.....	34
3.2.1 Microsoft .NET Framework.....	35
3.2.2 Microsoft Visual C#.....	36
3.2.3 SIMATIC NET.....	36
3.2.4 STEP7	38
3.2.5 WinCC.....	38
II PRAKTICKÁ ČÁST	40
4 KONFIGURACE OPC UA SERVERU	41
4.1 ÚPRAVA HARDWAROVÉ KONFIGURACE	41
4.2 KONFIGURACE OPC UA SERVERU SIMATIC.NET.....	44
4.2.1 Nastavení parametrů OPC	45

4.3	OPC UA SECURITY	45
4.3.1	Správa certifikátu	46
5	VYTVOŘENÍ PROGRAMU S7	49
5.1	FB300 – OPC DATA.....	50
5.1.1	FC301 – Write to DB	52
5.2	FB301 – OPC CTRL.....	53
5.2.1	Blok SFB12 (BSEND)	54
5.2.2	Blok SFB13 (BRCV)	56
5.3	KOMUNIKACE MEZI S7 A PC POMOCÍ VARIABILNÍ SLUŽBY	57
5.3.1	Použití absolutního přístupu.....	57
5.3.2	Použití symbolického přístupu	58
6	PROPOJENÍ VIZUALIZACE WINCC S OPC KLIENTEM.....	59
7	VÝVOJ KLIENSKÉ APLIKACE.....	60
7.1	CLIENTAPI.....	61
7.1.1	Třída Discovery	62
7.1.2	Třída Server.....	63
7.1.3	Třída Subscription.....	67
7.2	CLIENT	67
7.2.1	Třída MainForm	69
7.3	DATOVÝ VÝSTUP APLIKACE.....	71
8	TESTOVÁNÍ VYTVOŘENÉ APLIKACE S VYHODNOCENÍM.....	75
8.1	TESTOVÁNÍ PROMĚNNÝCH VARIABILNÍ SLUŽBY	75
8.2	TESTOVÁNÍ PROMĚNNÝCH BLOKOVÉ SLUŽBY	78
8.3	VYHODNOCENÍ PROJEKTU	81
ZÁVĚR	84	
ZÁVĚR V ANGLIČTINĚ.....	86	
SEZNAM POUŽITÉ LITERATURY.....	88	
SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK	90	
SEZNAM OBRÁZKŮ	92	
SEZNAM TABULEK.....	95	
SEZNAM PŘÍLOH.....	96	

ÚVOD

Díky rozvoji komunikace, počítačových sítí a různých metod strojové inteligence prochází proces výroby neustálým vývojem. Inteligentní zařízení a systémy si vzájemně dokážou vyměňovat informace. Sledování jednotlivých technologických kroků s okamžitým vyhodnocováním informací je standardem, který je dnes už samozřejmostí.

Dochází k propojování výpočetní techniky s oblastí automatizace. Vedle používání výpočetní techniky jako součást automatizační a řídicí techniky je vzrůstající tendence k používání síťového prostředí, internetu a dalších otevřených standardů. Tyto změny jsou přínosem, jak pro výrobce, tak i pro koncového uživatele. Klasické počítače jsou ve stále větší míře používány k vizualizaci nebo kontrole procesů, případně řešení dalších úkolů automatizace.

Rychlost, synchronizace v reálném čase a široká konektivita jsou klíčovými parametry pro systém řízení v automatizační technice. Nejčastěji se na úrovni procesní automatizace využívá propojení pomocí síťové architektury založené na protokolech, jako jsou například Profibus, CAN, Modbus, Profinet, EtherCAT apod. OPC je užitečným partnerem v tomto síťovém prostředí. Standardy OPC byly vyvinuty průmyslovou skupinou k podpoře výměny dat při řízení procesů prostřednictvím klientů a serverů Windows. OPC poskytuje otevřený standardizovaný přístup k propojení datových zdrojů, PLC, řadičů, vstupně výstupních zařízení, databází s klientskou aplikací HMI, jejíž součástí je grafika, trendy, alarmy apod.

I. TEORETICKÁ ČÁST

1 ANALÝZA ZADÁNÍ A POPIS VÝROBNÍ LINKY ROLLER HEAD

Primárním cílem je ukázat univerzálnost OPC komunikace mezi vybranými typy zařízení prostřednictvím jeho standardizovaných rozhraní, v této práci je využito OPC a PLC. Standard OPC specifikuje způsoby vzájemné komunikace v reálném čase, mezi zařízeními určenými pro řízení, monitorování a zaznamenávání technologického procesu, bez ohledu na výrobce softwaru a hardwaru. PLC neboli programovatelný logický automat je řídicí jednotka používaná pro řízení strojů nebo výrobních linek v reálném čase.

Základem práce je vytvoření OPC serveru v prostředí SIMATIC.NET a návrh včetně realizace OPC klienta v prostředí Microsoft Visual Studio. Vytvořená aplikace bude otestována a potom používána na výrobní lince RollerHead ve výrobním závodě Mitas a.s. provozovna Agro Otrokovice

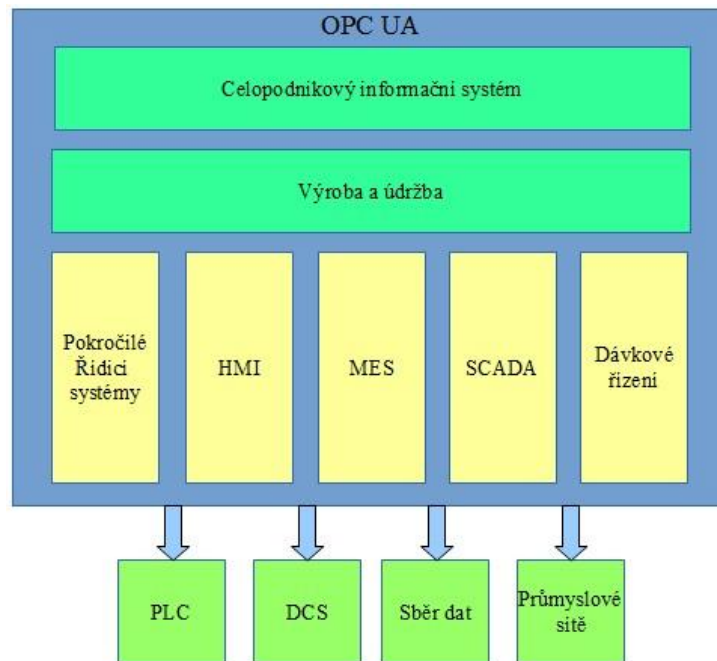
1.1 Analýza zadání

První částí práce by měla být analýza a postupné získávání znalostí o chování stroje na základě monitorování vstupně výstupních parametrů. Data budou využita k predikci nastavení parametrů stroje nebo k diagnostice poruchových stavů. Další částí práce by měla být analýza příčiny nedodržení nastaveného technologického procesu nebo eliminace množství vyrobeného odpadu. Cílem práce by měla být také i univerzálnost použití navrženého způsobu monitorování a řízení na více výrobních linek.

OPC server a OPC klient v této aplikaci poběží na stejném počítači spolu s vizualizací WinCC. Po navázání komunikace OPC serveru s PLC bude možné sledovat aktuální stav všech proměnných implementovaných v PLC. Standard OPC zahrnuje způsob předávání dat i specifické informace o dalších attributech doprovázejících tato data, například údaje o rozsahu, datovém typu, příznaku kvality a čase. OPC server získaná data převádí do formátu OPC a poskytuje je ke zpracování nadřazeným softwarovým aplikacím, kterými jsou připojení OPC klienti. Tito klienti mohou data pomocí OPC serveru nejen číst, zobrazovat, ale také zapisovat a to jak digitální tak i analogové hodnoty. Program OPC klienta bude naprogramován v prostředí Microsoft Visual Studio 2013, v jazyce C#. Výsledkem bude spustitelný soubor vybavený jednoduchým uživatelským rozhraním. Data

budou ukládány na pevný disk počítače, na kterém bude provozována aplikace obsluhující ukládání dat a zobrazování jejich hodnot. Data budou uložena v tabulkovém procesoru.

Pro účely provedení zápisu a čtení dat využitím OPC komunikace bude nutné upravit stávající program v daném PLC řídicí výrobní linky. Z důvodu možné vizualizace daného procesu bude přitom vhodné ověřit také možnosti SCADA systému, konkrétně WinCC.



Obr. 1. Cílové aplikace OPC UA

1.2 Popis výrobní linky RollerHead

Výrobní linka RollerHead slouží k vytlačení a válcování přesně dimenzovaných profilů různé tloušťky. Linka zajišťuje výrobu vnitřní gumy, která se stala důležitou součástí pneumatiky. Vnitřní guma je tenká fólie, která nahrazuje duši a zjednodušuje tak složení a manipulaci s celou pneumatikou. Základem linky je dvouválcový kalandr s vytlačovacím strojem RollerHead.

Vytlačovací stroj tlačí gumovou směs do vytlačovací hlavy, kde se pomocí vytvarované hubice tvoří profilový pás. Vytlačovaný profil postupuje dál do dvouválcového kalandru kalibrující tloušťku pásu vystupujícího z ploché hlavy vytlačovacího stroje. Pás se vytvaruje do podoby tenké fólie. Takto vytlačovaná fólie vstupuje do válcovacího stroje, který vytváří konečný tvar fólie. Intenzitu válcování ovlivňuje mezera mezi válci a rozdíl v obvodových rychlostech válců. Každým dalším průchodem materiálu mezi dvojicí válců se

zlepšuje kvalita prohnutí a kvalita povrchu. Podle velikosti štěrbin mezi válci se dosahuje předepsané tloušťky. Válce jsou navzájem napínány a kříženy z důvodu zamezení prohnutí válce ve středové části, aby nedošlo k nežádoucímu zesílení válcované fólie.



Obr. 2. Přední část linky RollerHead

Vytlačený pás se ořeže pomocí dvou ořezávacích nožů. Ořezané okraje z kalandru se vrací do násypky vytlačovacího stroje vratným dopravníkem.



Obr. 3. Ořezávání za kalandrem

Chlazení vytlačeného pásu zajišťují chladicí válce, které jsou naplněny vodou. Zbytek linky je tvořen soustavou dopravníků a navíjecích stanic. Ochlazená fólie putuje přes středící zařízení do navíjecí stanice, kde se navíjí do cívek. Takto připravený materiál se potom spojuje s kordovou vložkou a ochranným páskem nebo se přímo zpracovává na konfekci.



Obr. 4. Navíjecí stanice Miag

Řízení celého výrobního procesu se uskutečňuje přes operátorský panel s vizualizací WinCC. Přes toto rozhraní je možné přistupovat ke všem datům z PLC. Procesní proměnné se přenášejí z řídicích prvků do vizualizace a opačně.

Denní produkce je měřena podle celkové délky vytlačeného profilu, která se obvykle pohybuje okolo 4000m. Na ztrátách se vedle poruchových stavů, pohybujících se okolo 1 až 2 procent za měsíc, nejvíce podílí tzv. přehozy, kdy se vyměňují válce na kalandru podle požadovaného profilu. Průměrný čas pro přehozy je asi 210 minut za den, což odpovídá asi 29 procentům denní produkce. Na množství odpadu má vliv každá změna rozměru, kdy se musí odřezat úsek před a za spojem. Potom se po každém přehozy zařadí mezi odpad začátek vytlačeného profilu, protože nedosahuje požadovaných parametrů. Dále má na vykazování odpadu vliv proměnlivá kvalita kaučukové směsi a taky vliv kvality práce obsluhy linky. Množství odpadu se mění podle poměru vlivu těchto parametrů, takže ho není možné přesně statisticky vyčíslit. Dochází k velkým denním rozdílům.

2 ZÁKLADNÍ POPIS OPC

Původní přístup aplikačních programů k datům z koncového zařízení je realizován prostřednictvím ovladače (driveru). Protože ovladače nepodléhají žádné normě, dochází k odlišnostem u každého výrobce. To vede k těmto problémům:

- Každá aplikace musí obsahovat ovladače konkrétního hardware.
- Nefunkčnost ovladače může být způsobena například změnou vlastnosti hardware.
- Různé programové balíky mohou sdílet zařízení jenom, pokud každý z nich obsahuje nezávislý ovladač.
- Dochází k neshodám mezi ovladači od různých dodavatelů, když ne všichni dodavatelé podporují všechny vlastnosti daného hardwaru.

V klientských protokolech vznikají odlišnosti a to znemožňuje výrobcům řešení tohoto problému. Tuto situaci řeší vznik standardu OPC, který umožňuje získávat data z různých zdrojů a přenášet je nezávisle na hardwaru do kteréhokoliv klientského programu. OPC byl zaveden jako standard mezi výrobcí hardwaru a softwaru. V roce 1996 byl představen zakladatelskou skupinou společností OPC Foundation, aby poskytl obecnou specifikaci pro vzájemnou součinnost softwarových aplikací.

2.1 Představení OPC

OPC Foundation poskytuje specifikace pro výměnu dat v průmyslové automatizaci. Na začátku byla dlouhá historie založená na specifikaci COM/DCOM, zahrnující nejvýznamnější OPC Data Access (DA), OPC Alarms a Events (AE) a OPC Historical Data Access (HDA), které jsou hodně rozšířeny v průmyslu a implementovány na většině cílových systémů průmyslové automatizace [6].

OPC specifikace řešila způsob předávání dat z jedné aplikace do druhé prostřednictvím architektury klient-server. Termín OPC byl původně známý pod zkratkou OLE pro řízení procesů (OLE for Process Control), později se jeho význam změnil na otevřenou konektivitu (Open Connectivity) používající otevřené standardy. Technologie OLE umožňuje vkládání a propojování objektů, které jsou postaveny na systému COM

představující model Component Object Model. COM model umožňuje sdílet komponenty a vzájemnou komunikaci mezi aplikacemi u operačního systému Windows. Pro komunikaci více PC se nabízí rozšíření Distributed COM (DCOM), který byl původně navržen pro obchodní systémy, ne pro distribuovanou součinnost v reálném čase. OPC řešení vytváří jednotné rozhraní pro výměnu dat mezi aplikacemi reprezentovanými OPC klienty a koncovými zařízeními prostřednictvím OPC Serveru. Aplikace nejsou dále závislé na použití jednotlivých hardwarových ovladačů a tímto odpadají pro výrobce i uživatele další možné problémy. Standardizace umožnila klientským aplikačním programům konzistentní přístup k datům v technologických provozech. Výrobcům hardwaru stačí stejný soubor softwarových komponent pro všechny zákazníky a zároveň pro zákazníky vzniká svoboda volby mezi dodavateli různých součástí a zařízení. OPC je standardním rozhraním moderních programových produktů pro sledování a řízení technologických procesů a zařízení.

2.2 Nová specifikace OPC UA

V roce 2006 OPC Foundation vydala novou generaci specifikací OPC s názvem OPC Unified Architecture (OPC UA), která je na platformě nezávislý standard pro komunikaci různých druhů systémů a zařízení prostřednictvím přenosu zpráv mezi klienty a servery po různých typech sítí. S OPC UA OPC Foundation splňuje technický pokrok a přechází od COM/DCOM technologie, která byla přizpůsobena pouze na operační systém Windows, k architektuře orientované na služby poskytující data i pro jiné operační systémy prostřednictvím webových služeb nebo vlastním optimalizovaným protokolem na bázi TCP. OPC UA je postavena na technologii založené na architektuře orientované na službu (SOA). Protože se chová podobně jako každá webová služba, která spolupracuje s firewally, může být řízena správcem systému. Přímochárým a snadno předvídatelným způsobem umožňuje definici portů pro přístup ke službě a řízení síťového provozu. Tímto dochází k významnému zlepšení oproti OPC v minulosti, které bylo založeno na COM / DCOM. Tento objektově orientovaný systém vyžadoval mnoho změn v konfiguraci bezpečnosti Windows, bez kterých nebylo možné distribuované operace rozběhnout. DCOM také nezajišťoval dostatečnou kontrolu komunikace mezi počítači v oblasti portů, která omezuje možnost řídit komunikaci přes firewally.

Výhodou nového systému OPC UA oproti COM/DCOM je spojení ad hoc umožňující nezávisle na sobě měnit klienty, což ukazuje na vysokou flexibilitu řešení založených na SOA. Naopak v COM/DCOM byly jednotlivé služby pevně vázány na dané klienty.

OPC UA klade důraz na zabezpečené spojení mezi serverem a klientem za pomoci asymetrické kryptografie s veřejným klíčem. OPC UA poskytuje robustní a zabezpečenou komunikaci založenou na ověření identity klientů a serverů.

Základním prvkem OPC UA je UA server, který v sobě spojuje všechny funkce dřívějších klasických OPC serverů. Pozoruhodné zlepšení přinesl OPC UA model adresového prostoru, podle kterého mohou dodavatelé vystavit bohatý a rozšiřitelný informační model za použití objektově orientovaných technik. Specifikace OPC UA je rozdělena do několika částí. Zatímco v předchozí specifikaci OPC Foundation nebyla žádná spojitost mezi skutečnou hodnotou (OPC DA) s historickými daty (OPC HDA) nebo s událostmi (OPC Events), OPC UA poskytuje všechna data v jednotném adresovém prostoru. Takže aktuální údaje, historická data a události jsou ve vzájemném vztahu. Díky jednotnému adresovému prostoru již nemůže dojít k nekonzistenci dat mezi servery dat a servery událostí [6].

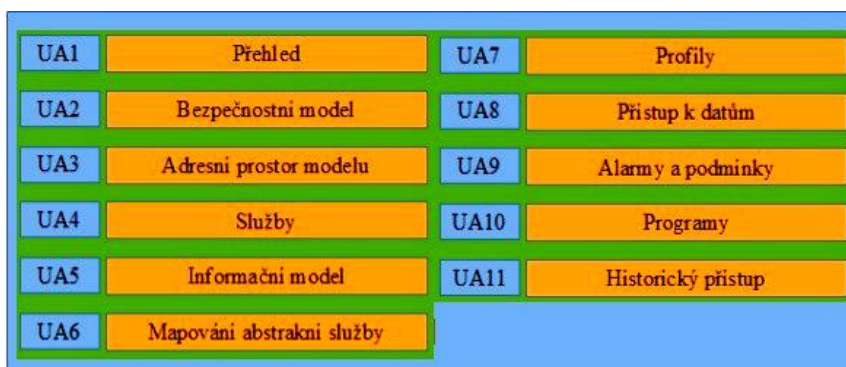
Další důvody, které motivovaly OPC Foundation rozvíjet její novou OPC UA specifikaci by se daly shrnout do těchto bodů:

- Zatímco stará specifikace pouze poskytuje jednoduchou hierarchii, OPC UA podporuje vysoko úroňový datový model. Server může data prezentovat v mnoha hierarchiích, přizpůsobených obvyklému pohledu klientů na konkrétní data.
- OPC UA umožňuje, aby server mohl poskytovat klientům definice typů objektů. Klienti se prostřednictvím adresového prostoru dotazují serverů na metadata, které popisují formát konkrétních dat.
- OPC UA specifikace je rozdělena do vrstev, aby oddělila návrh od výpočetních a síťových technologií. To usnadňuje integraci nových technologií. UA6 používá dva způsoby kódování: XML/text a UA Binary a dva způsoby přenosu: TCP (založené na SOAP) a Webové služby (založené na HTTP).
- OPC UA definuje standardní podmnožiny pro usnadnění vzájemné spolupráce. Na základě specifikace UA7 mohou klienti přistupovat k serveru podle zjištěného

profilu. Standardně lze UA server vybavit funkcemi pro řízení přístupu. UA klient předkládá serveru své oprávnění a ten kontroluje, zda je tento klient oprávněn odebrat danou službu. Zasílané zprávy je možné pro zvýšení bezpečnosti šifrovat.

OPC UA propojuje vše od inteligentních zařízení, regulátorů, DCS a SCADA systémů až do MES a ERP systémů [6].

OPC UA je účelově vytvořeno pro potřeby automatizačního průmyslu, stává se velmi efektivní v oblasti jednoduchosti použití, výkonnosti a bezpečnosti. Produkty založené na OPC UA jsou dnes dominantní v oblasti OPC a očekává se jejich další rozšíření.



Obr. 5. Specifikace OPC UA

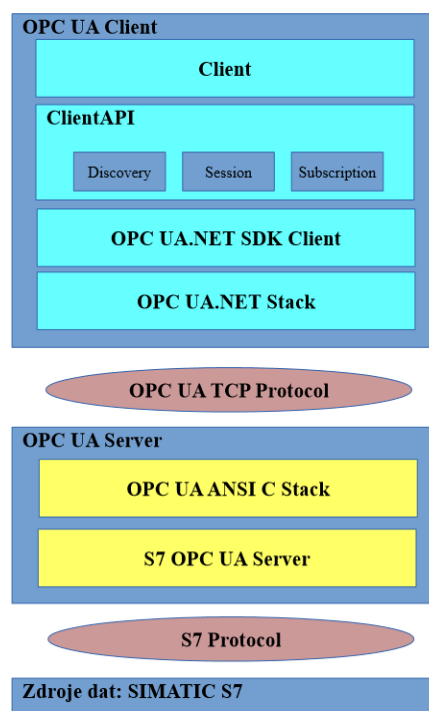
2.2.1 Architektura OPC UA

Na rozdíl od dřívější objektové koncepce OPC je OPC UA založená na architektuře orientované na služby (SOA). Základní myšlenka SOA je to, že jednotlivé aplikace se skládají ze služeb. Ke službám aplikace přistupují prostřednictvím standardizovaných rozhraní. OPC UA pracuje s jednotnou sadou služeb, například čtení a zápis dat, procházení a zobrazení adresového prostoru apod. V architektuře SOA existují tři základní softwarové prvky: poskytovatel, registr a odběratel. Poskytovatel posílá do registru informace o síťových službách, které je schopen zajistit a zároveň popisuje, jak vypadá rozhraní pro tyto služby. Odběratel hledá ve stejném registru vhodného poskytovatele dané služby.

OPC server je software komunikující buď s připojeným hardwarovým zařízením (např. PLC) pomocí jeho komunikačního protokolu (např. Profibus, MPI, Modbus, atd.) nebo s

jiným zdrojem dat (např. databáze). Data, která získá, převádí do formátu OPC a poskytuje je nadřazeným softwarovým aplikacím (OPC klienti) k dalšímu zpracování.

Dle specifikace je OPC server rozdělen na několik standardních částí. Na nejnižší úrovni řízení OPC serveru je umístěn ovladač vstupů a výstupů, který se stará o přenos dat do připojeného fyzického zařízení a naopak o získávání dat z něj. V prostřední části se OPC server stará o optimalizaci získávání dat, za účelem dosažení vyšší efektivity komunikace mezi ním a fyzickým zařízením. Nejvýše postavené vrstvy mají za úkol zprostředkovávat pomocí příslušných rozhraní OPC přenesená data připojeným OPC klientům. Celá architektura OPC UA je viditelná na následujícím obrázku.



Obr. 6. Architektura OPC UA

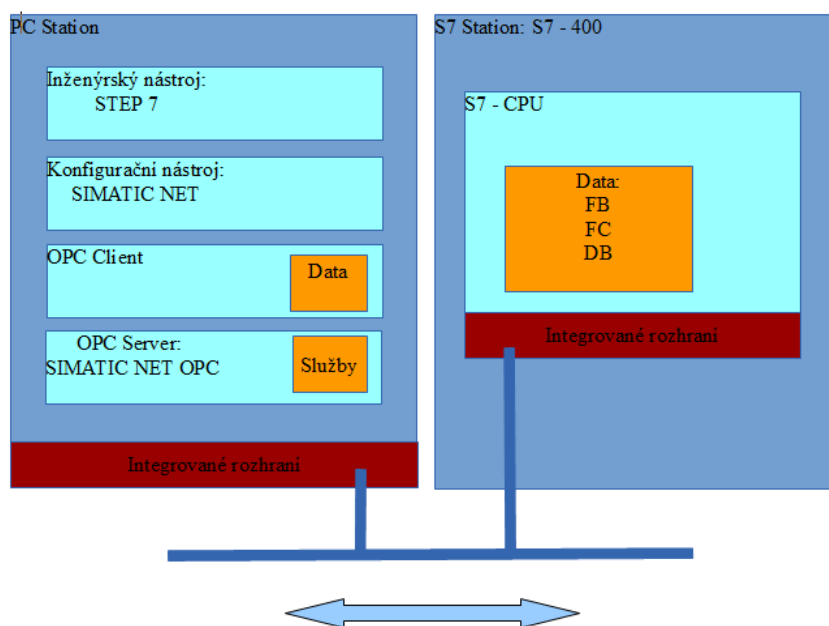
Význam jednotlivých částí viditelných na obrázku:

- ANSI C UA Stack - Server SIMATIC NET OPC UA používá optimalizovaný a přenositelný OPC UA ANSI C zásobník OPC Foundation.
- S7 OPC UA Server - Server SIMATIC NET OPC UA implementuje potřebnou serverovou logiku pro relace, podpisy a datové připojení k S7 stanicím.
- OPC UA .NET Stack - Na .NET založený OPC UA komunikační zásobník OPC Foundation.

- .NET Client SDK - Na .NET založený OPC UA klient SDK OPC Foundation.
- ClientAPI – Klient nabízející opakovaně použitelné C# třídy pro vyhledávání, relace a manipulace s podpisem. ClientAPI izoluje klienta od komunikačních vrstev.
- Client - Jednoduchý příklad pro použití rozhraní API klienta s funkcemi: připojení, odpojení, čtení, zápis a sledování dat. Tento příklad také ukazuje přímé adresování a manipulace se jmennými prostory.

2.3 S7 komunikace

SIMATIC NET OPC server umožňuje používat S7 komunikaci pomocí OPC UA. Protokol S7 pro PROFIBUS a průmyslový Ethernet se používá pro komunikaci mezi systémovými komponenty v rámci programovatelného automatu SIMATIC S7, případně pro komunikaci mezi komponenty systému SIMATIC S7 a programovacími zařízeními nebo PC.

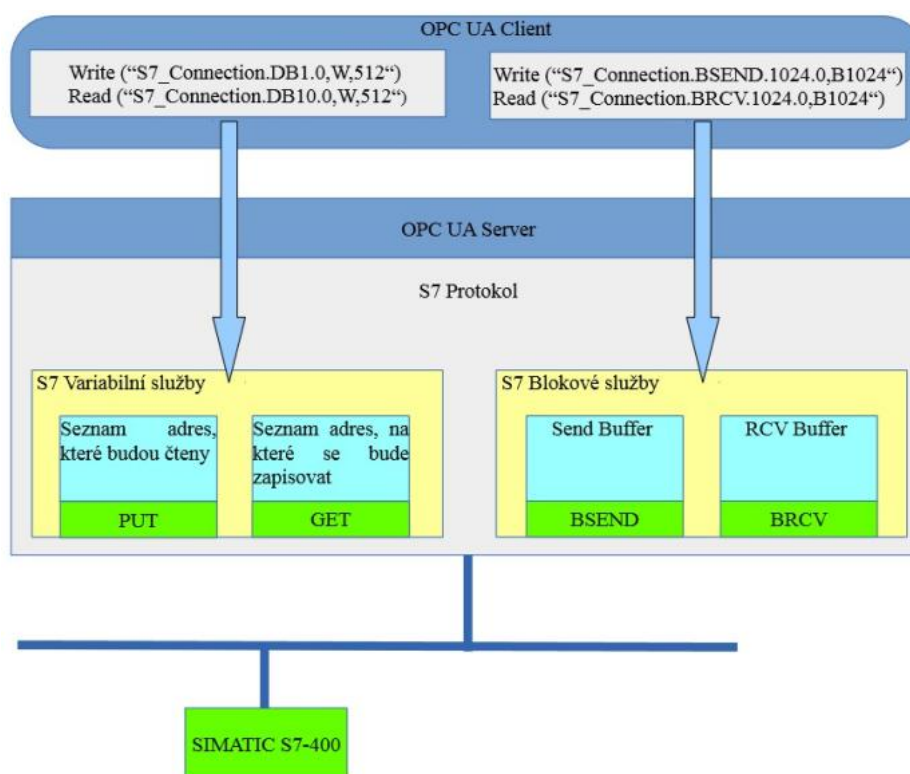


Obr. 7. Komunikační partneři OPC

Server SIMATIC NET OPC UA je dostupný pouze pro protokol S7 (přes Ethernet nebo Profibus). "Klasický" OPC server (COM / DCOM) je nadále k dispozici pro všechny ostatní protokoly: například DP, FDL, SR nebo SNMP. Komunikace S7 je rozdělena do dvou velmi odlišných komunikačních služeb, do variabilních služeb a blokových služeb. Pouze

na základě NodeId protokolu je možné rozpoznat, která komunikační služba je používána. Namespace "S7:" znamená, že se jedná o nový typ adresování, který je založen na syntaxi ukazatelů. Zejména v případě velkého počtu proměnných nabízí jasně výkonnější přístup. Pro dosažení kompatibility se starší syntaxí ItemIds u klasického OPC Data Access zůstává stará struktura identifikátoru "S7COM".

Server OPC UA odděluje interně NodeId svých komponent. Na základě struktury NodeId zjistí, přes které komunikační služby se uskuteční komunikace s S7. Název připojení identifikuje komunikačního partnera (mimo jiné toto jméno reprezentuje např. IP adresu) a klíčové slovo "BRCV" nebo "BSEND" zajistí použití blokových namísto variabilních služeb. Identifikátor typu S7 a offset adresy označují pozici dat uvnitř PLC. Datový typ specifikuje interpretaci těchto dat.



Obr. 8. Popis S7 komunikace

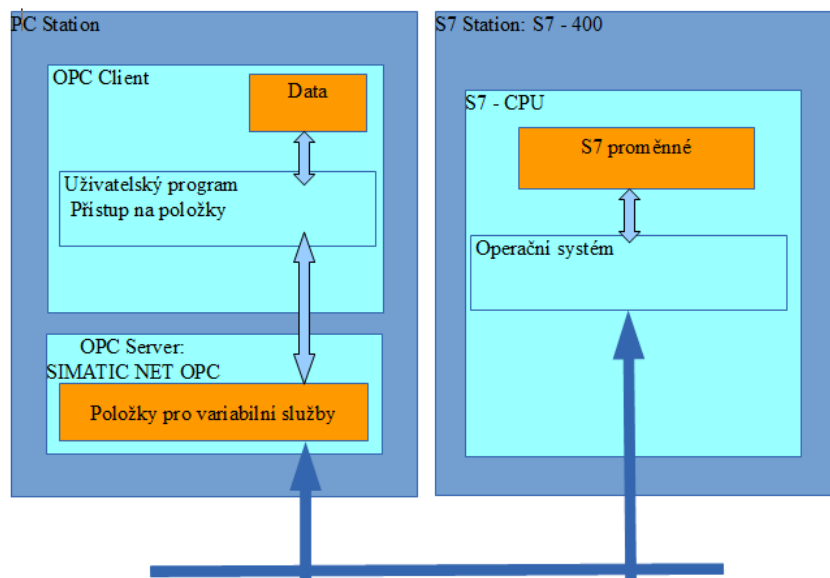
2.3.1 Variabilní služby

Pokud PLC S7 využívá variabilních služeb je nutné pouze jednostranně nastavené spojení. Použitím S7 variabilní služby může PC stanice přistupovat přímo k datům S7 stanice. Takto můžou být přímo přístupné všechny datové oblasti (I, Q, M, DB, atd.) CPU. Pro

správnou funkci je vyžadován pouze uživatelský program v PC stanici. Tato komunikační služba je velmi flexibilní a především snadno použitelná.

OPC klient může přistupovat k datům na S7 CPU různými způsoby:

- Monitorování proměnných - Při monitorování OPC UA server neustále kontroluje, jestli se změnila hodnota. Když je detekována změna, je tato hodnota přenesena z OPC UA serveru do klienta.
- Synchronní čtení/zápis - OPC klient čeká, až OPC UA server provede úlohu.
- Asynchronní čtení/zápis - OPC klient nemusí čekat, až OPC server provede úlohu.



Obr. 9. Přístup OPC klienta k položkám OPC serveru

Přístup OPC klienta k položkám OPC UA serveru pomocí absolutního přístupu používá tři typy syntaxe:

- S7:[<connectionname>] <object>{<type>} <address> {, <quantity>}
- S7:[<connectionname>] DB<no> {<type>} <address> {, <quantity>}
- S7:[<connectionname>] DI<no> {<type>} <address> {, <quantity>}

Tab. 1. Význam jednotlivých částí syntaxe variabilní služby

Komponenta	Stručný popis
S7:	Protokol S7 pro přístup k S7 proměnným.
<connectionname>	Název spojení, který je specifikován v konfiguraci.
DB	Identifikátor pro S7 proměnné v datovém bloku.
DI	Proměnné S7 v instančním datovém bloku.
<no>	Číslo datového bloku nebo instance datového bloku.
<object>	Specifikace typu oblasti v PLC.

Příklad syntaxe: S7_connection_1.DB1.10, W.

Proměnná typu Word (16 bitová hodnota) se nachází v datovém bloku 1 a začíná na adrese 10 bajtu (je složena z bajtů 10 a 11). Tato proměnná je načtena přes připojení s názvem "S7_connection_1", což znamená S7 CPU skrývající se za tímto připojením.

Další příklady syntaxe:

- S7:[S7_connection_1]DB1,B9 - identifikuje bajt číslo 9 v datovém bloku DB1.
- S7:[S7_connection_1]DI3,W10,9 – identifikuje 9 datových slov začínající na bajtu č.10 v instančním datovém bloku 3.
- S7:[S7_connection_]IB2 – identifikuje vstupní bajt č.2.

Vedle přímého absolutního adresování prostřednictvím nové syntaxe ("S7:" namespace), které je stejné jako staré kompatibilní adresování (S7COM: namespace), je zde možnost symbolického adresování. U tohoto typu adresování je ze STEPu 7 (HW Config nebo NetPro) generován adresový prostor. Obraz symbolického názvu pro přímé adresy je uložen v souboru s koncovkou ATI (Advanced Tag Informace). Generovaný soubor je buď nahrán do OPC UA serveru prostřednictvím STEP 7 nebo přes import XDB souboru.

2.3.2 Blokové služby

Pro výměnu velkých objemů dat je k dispozici účinnější bloková služba, kde mohou být vyměněny díky dvoustrannému konfigurovanému spojení velké objemy dat (až 64kbytes). Komunikace je založena na výměně datové vyrovnávací paměti. Pro tento účel musí být v řídicím programu volány příslušné funkční systémové bloky (BSEND / BRCV). Pokud byly vytvořeny odpovídající NodeIds, poskytuje Server OPC UA příslušné protějšky na PC.

Existuje dva typy syntaxe v namespace S7:

- S7:[<connectionname>] Brcv <RID>{, {<type>}<adress>{, <quantity>}}
- S7:[<connectionname>].Bsend<RID>.<length>.{, {<type>}<adresa>{, <quantity>}}

Tab. 2. Význam jednotlivých částí syntaxe blokové služby

Komponenta	Stručný popis
S7:	Protokol S7 pro přístup k procesním proměnným.
<connectionname>	Název spojení, který je specifikován v konfiguraci.
BRCV	Blok dat přijímaných od partnera.
BSEND	Blok dat odesílaných partnerovi.
<length>	Počet odesílaných bytů.
<RID>	ID spojení pro dvojici BSEND/BRCV.
<address>	Adresa první proměnné, která bude adresována.
<type>	Datový typ S7 je převeden na odpovídající OLE datový typ v OPC serveru.
<quantity>	Počet proměnných typu, který má být adresován.

Příklady syntaxe:

- S7:[S7-OPC-1]BRCV,4 - Blok dat je přijat v přijímacím bufferu s RID 4. Kompletní buffer je mapován na pole bajtů.

- S7:[S7-OPC-1]BRCV,1,W1,4 - Z přijatého bloku dat je mapován obsah začínající na adrese 1 s délkou pole 4 slov. Vztahuje se celkem na blok dat 8 bajtů.
- S7:[S7-OPC-1]BSEND32,1,REAL20 - Pole s čísly s plovoucí čárkou je adresován na datový blok o délce 32 bajtů s RID 1 začínající na adrese 20. Zapsaná hodnota je zadána na určené pozici v bloku a blok dat je odeslán.
- S7:[S7-OPC-1]BSEND10,1,D2 – Toto dvojitě slovo (Double word) má v datovém bloku délku 10 bajtů s RID 1 od adresy 2. Zapsaná hodnota je zadána na určené pozici v bloku a blok dat je odeslán.

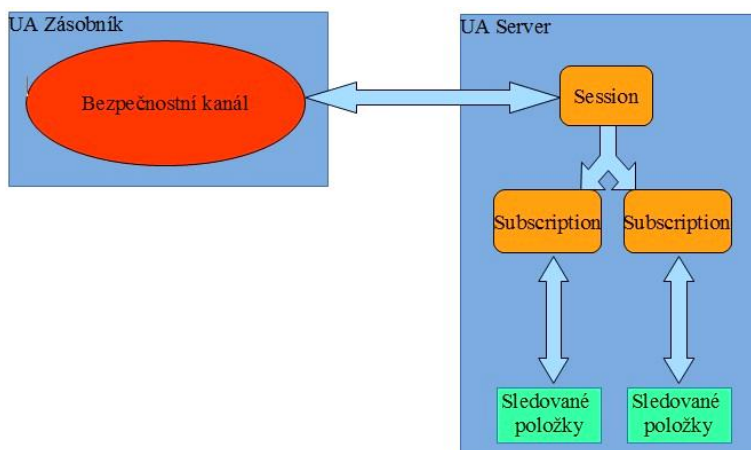
2.3.3 Namespace a NodeId

Uzel je základním stavebním kamenem adresového prostoru. Uzel je z určité třídy jako objekt, proměnná nebo metoda a je popsán seznamem atributů. Všechny uzly mají společné atributy jako je název nebo popis a konkrétní atributy jako jsou např. hodnota proměnné. Každý uzel v adresovém prostoru OPC UA je jednoznačně identifikován pomocí NodeId. Tento NodeId je složen ze jmenného prostoru (namespaces) pro odlišení identifikátorů z různých subsystémů a dále z identifikátoru, který může být buď číselná hodnota, řetězec nebo GUID. Jako identifikátor se používají nejčastěji řetězce. Číselné hodnoty se používají pro statické jmenné prostory, jako je např. typ systému. OPC UA definuje jmenný prostor pro uzly definované OPC. Servery OPC UA navíc definují jednu nebo více jmenných prostorů. Následující přehled uvádí příslušné jmenné prostory pro server SIMATIC NET OPC UA:

- <http://opcfoundation.org/UA/> - Používá se pro uzly, které jsou definovány v OPC UA (UA část 5). Jedná se o uzly tvořící základní strukturu adresového prostoru a uzly představující typy definované OPC UA.
- S7: Jmenný prostor pro přímé adresování S7 proměnných s optimalizovanou syntaxí.
- S7COM: Jmenný prostor pro přímé adresování S7 proměnných se syntaxí kompatibilní s OPC Data Access Server.
- SYM: Jmenný prostor pro symbolické adresování S7 proměnných. Informace o symbolu je exportována z projektu STEP 7.

2.3.4 Rozhraní pro přístup do adresového prostoru

Následující obrázek ukazuje komunikační kanál a aplikační objekty, které mohou být vytvořeny v průběhu výměny dat mezi klientem a OPC UA serverem.



Obr. 10. Aplikační objekty a komunikační kanál

Tyto objekty jsou dále podrobněji popsány:

- **Bezpečnostní kanál:** Bezpečný komunikační kanál je realizován v zásobníku OPC UA. Objekty na aplikační úrovni jsou nezávislé na sobě, ale mohou být vytvořeny, používány nebo měněny pouze v rámci bezpečnostního kanálu.
- **Session:** Relace na serveru je logické spojení mezi klientem a OPC UA serverem. Každá relace je spojena s bezpečnostním kanálem. Relace je odstraněna ze serveru, pokud není klientem přijato žádné volání v rámci časového limitu.
- **Subscription:** Objekt může být vytvořen klientem skupiny sledovaných položek. Sledované položky se používají pro sledování změn hodnot nebo k přijímání zpráv událostí. Podpis je odstraněn ze serveru, pokud nemohly být zaslány žádné údaje klientovi v rámci časového limitu.

2.3.5 Adresování OPC UA serveru

K adresování OPC UA serveru je možné využít adresování pomocí URL serveru. Pro nativní binární protokol TCP existují dva způsoby adresování serveru:

- Přímé adresování používá port 4845 pro OPC UA Server. Příklady syntaxe:
 - `opc.tcp://<hostname>:4845`

- opc.tcp://<IP-Adrese>:4845
- opc.tcp://localhost:4845
- opc.tcp://.\pipe\locals7

Výkonný lokální koncový bod vyžaduje použití zásobníku SIMATIC NET UA. Koncový bod může být dostupný jenom z lokálního PC. Koncový bod může být v programu "Configuration Console" aktivován výběrem protokolu OPC.

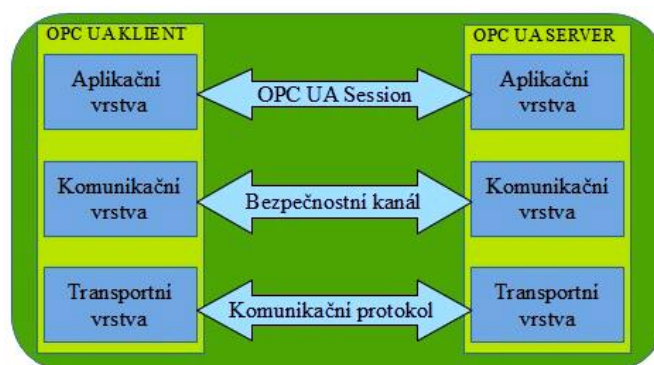
- URL serveru S7 OPC UA lze také nalézt pomocí služby OPC UA Discovery. Discovery server používá port 4840. Příklady syntaxe:

- opc.tcp://<hostname>:4840
- opc.tcp://<IP-Adrese>:4840
- http://<hostname>:52601/Discovery/
- http://<IP-Adrese>:52601/Discovery/

Dalším způsobem je adresování prostřednictvím IPv6 adresy, které mohou být také použity jako adresy IP. Adresa musí být v závorkách, např. [fd72 :: e522: B630: 5834:75 e9% 15].

2.4 Transportní protokoly

OPC UA specifikuje vrstvenou architekturu a přiřazuje zvláštní bezpečnostní funkce v různých vrstvách. Aplikační vrstva je odpovědná za autorizaci uživatelů a ověřování, komunikační vrstva ověřuje a používá aplikační ověřování a nakonec transportní vrstva udržuje jako bezpečnostní cíl důvěrnost údajů a integritu.



Obr. 11. Specifikace vrstvené architektury

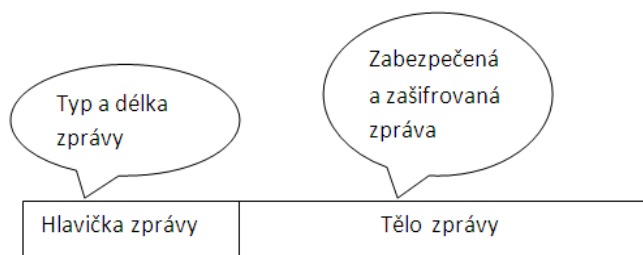
Na nejnižší transportní vrstvě OPC UA specifikuje dva transportní protokoly UA TCP a SOAP / HTTP. Tyto protokoly se používají pro vytvoření spojení mezi klientem a OPC

UA serverem. UA TCP je jednoduchý protokol, který poskytuje speciální mechanismy požadované OPC UA. Například speciální zotavení z chyb umožňuje relacím OPC UA přežít síťové přerušení. Ke komunikaci s webovými službami přes internet pro OPC UA aplikace slouží standardy SOAP / HTTP.

2.4.1 UA TCP

Použitím UA TCP na transportní vrstvě je dosažena rychlá a jednoduchá komunikační síť. UA TCP využívá komunikaci pomocí socketů. Socket je mechanismus, kterým je možné zprostředkovat lokální nebo vzdálenou komunikaci dvou uzlů mající charakter klient/server. Koncepce socketu umožňuje připojení několika klientů k jednomu serveru. Komunikace zahajuje strana serveru. Na začátku vyžádá pomocí systémového volání nový volný socket. Pokud klient ztratí připojení socketu, pak nový socket je vytvořen a přiřazen stávajícím bezpečnému kanálu, který má být nejprve znovu ověřen serverem. Nevyřízené žádosti od klienta a odpovědi serveru jsou ukládány do vyrovnávací paměti, dokud nový socket není dostupný.

Obecná struktura UA TCP zprávy se skládá z hlavičky a těla zprávy, který jsou znázorněny na následujícím obrázku.



Obr. 12. Struktura UA TCP

Hlavička zprávy obsahuje informace o typu a délce zprávy. Tělo zprávy obsahuje buď zakódované a zabezpečené zprávy služeb, které jsou předány do horní vrstvy nebo obsahuje specifické připojení zpráv používané pro vytvoření připojení socketu, případně výměnu informací o chybě připojení. K dispozici jsou tři typy UA TCP zpráv, které budou stručně představeny v následujícím textu:

- Aby bylo možné vytvořit připojení socketu na konkrétní koncový bod poskytnutý serverem, je odeslána OPC UA klientem na server zpráva *Hello*. Odesláním zprávy

klient požaduje určitou velikost vyrovnávací paměti pro odesílání a příjem dat ze serveru, stejně jako celkovou délku zprávy.

- Jako odpověď na zprávu *Hello* server odešle *potvrzovací zprávu*, kterou potvrdí nebo pozmění požadovanou velikost vyrovnávací paměti, stejně jako délku zprávy. Souhlas s těmito hodnotami je důležitý z hlediska spolehlivosti a bezpečnosti. Klient a server tak ví, co můžou od sebe očekávat a díky tomu odolávají určitým útokům, jako je přetečení zásobníku nebo Denial-of-Service.
- Třetí typ zprávy je *chybová zpráva*, poskytující informace o chybě. Například, pokud zprávu nelze zpracovat, protože velikost je příliš velká nebo je server přetížen.

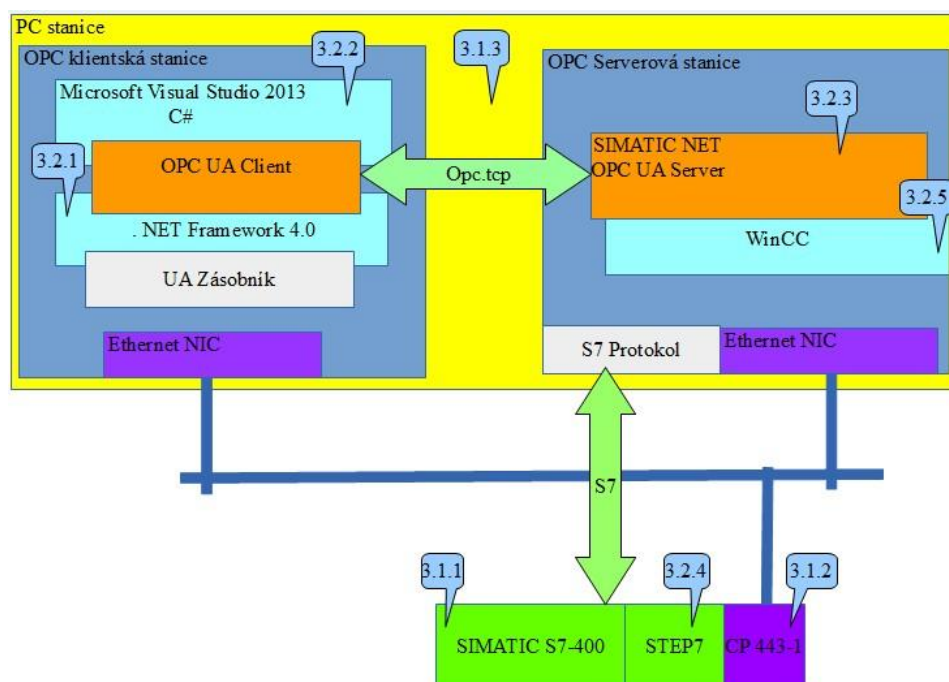
2.4.2 SOAP/HTTP

SOAP / HTTP je zkratka pro SOAP přes HTTP a je to široce přijímané komunikační schéma v prostředí webových služeb. OPC UA aplikace můžou bezpečně komunikovat spolu navzájem přes internet, stejným způsobem jako webové prohlížeče komunikují s webovými servery. Výhoda použití HTTP také spočívá v tom, že stávající síťová infrastruktura dovoluje neomezenou komunikaci na portu vyhrazeném pro HTTP (TCP port 80). Webové služby je možné používat bez nutnosti zásahu do konfigurace aktivních síťových prvků, jako jsou firewally. Při použití technologií DCOM je potřeba povolit komunikaci na portech, které používají příslušné přenosové protokoly.

SOAP je síťový protokol pro výměnu dat mezi systémy a volání vzdálených procedur. Je tedy založen na několika dalších standardech, např. XML pro reprezentaci dat a HTTP nebo TCP pro datové přenosy. SOAP požadavek se zasílá v těle HTTP požadavku. Používá metodu POST, která dovoluje posílat data v těle HTTP požadavku. Požadavek musí obsahovat HTTP hlavičku, která identifikuje SOAP požadavek. Tuto hlavičku mohou používat jednak firewally k filtrování požadavků a jednak může obsahovat URI s identifikací služby, která se má vyvolat. V zásadě je SOAP zpráva strukturována v hlavičce a v těle, která je vyjádřena na následujícím obrázku.

3 POPIS POUŽITÉ TECHNOLOGIE

V této části práce je popsán stav využití hardwarových a softwarových prostředků, které zajišťují propojení klientské aplikace a řízení výrobní linky. Oproti původnímu řešení se stává součástí PC stanice klientská aplikace a OPC server. Na následujícím obrázku jsou jednotlivé použité součásti vyobrazeny s odkazem na kapitolu, ve které je podrobnější popis.



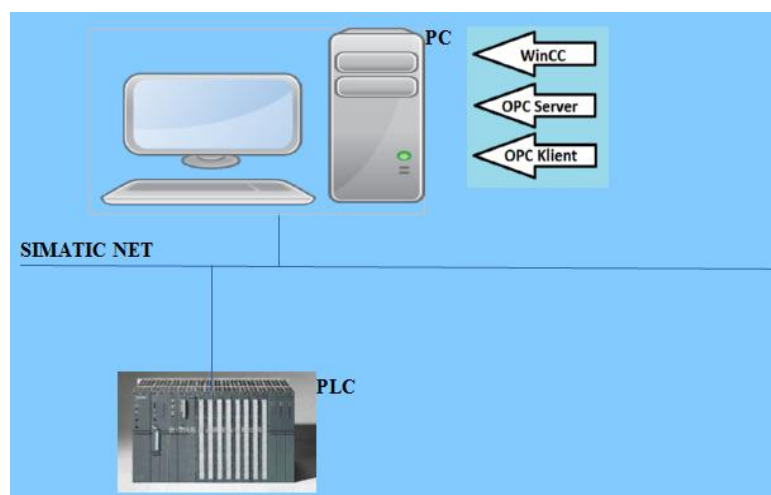
Obr. 14. Použité hardwarové a softwarové komponenty

3.1 Hardware

Obsluha aplikace běží na PC stanici, která plní funkci vizualizace a hlavního ovládacího místa linky. Součástí PC stanice se stane klientská aplikace, tak i OPC server. PC stanice je prostřednictvím běžné síťové karty, na které běží OPC a ethernetové komunikační karty CP 443-1 na straně PLC připojena k programovatelnému automatu PLC zajišťující řízení výrobní linky. Propojení je realizováno prostřednictvím sítě LAN podle standardu Ethernet 100BASE-T. Ke komunikaci je používán protokol TCP/IP. U tohoto protokolu musíme přiřadit každému zařízení vlastní IP adresu a masku sítě, která bude na všech stanicích v síti stejná (v našem případě 255.255.255.0).

3.1.1 PLC SIMATIC S7-400

PLC je zkratka pro Programmable Logic Controller neboli programovatelný automat. Je to přístroj, který řídí nějaký proces prostřednictvím svých výstupů. Na výstupy se připojují prvky jako například motory, ventily, vzduchové válce nebo signalizace stavu. Informace o procesu dostává ze vstupů, na které má připojeny signální čidla, tlačítka apod. Původní centrální způsob řízení, kdy všechny senzory a akční členy jsou připojeny na jednu centrální stanici, je dnes většinou nahrazen strukturou distribuovaných automatizačních systémů. Distribuovaný systém využívá k přenosu informací decentrálních periferních jednotek, které se umísťují v blízkosti senzorů a akčních členů.



Obr. 15. Propojení hardwarových komponent

Nejčastěji komunikují po průmyslové sběrnici Profibus nebo v poslední době častěji po sběrnici Industrial Ethernet s protokolem Profinet. Sestavení PLC programu se provede pomocí softwaru STEP 7 na PC a po připojení přes komunikační rozhraní PLC se pomocí zaváděcí funkce uloží do paměti PLC. V projektu použité S7-416 s objednacím číslem 6ES7416-3XJ00-0AB0 patří do kategorie PLC pro střední a vyšší výkonový rozsah. Díky modulárnímu designu, vysoké možnosti rozšiřování a rozsáhlým komunikačním schopnostem je vhodný pro náročné projekty. Periferie můžou být propojené sběrnici například MPI nebo Profibus.

Siemens pro své PLC používá normu, kterou vytvořila organizace PLC Open sjednocující požadavky na provedení a programování programovatelných automatů. V současnosti používaná norma označovaná IEC/EN 61131-3 je vytvořena z několika částí. První část obsahuje základní informace o PLC, druhá část je věnována požadavkům na provedení

elektroniky a na testování PLC. Třetí část řeší způsoby programování, popisuje syntaxi čtyř programovacích jazyků (LD, FBD, IL, ST) a způsoby aktivace úloh s nástroji SFC pro realizaci sekvenčních úloh. Podle této normy jsou programovány programovatelné automaty většiny významných světových výrobců včetně výrobců počítačových řídicích systému typu softPLC nebo embedded průmyslových počítačů. Programátor podle této normy vytvoří univerzální program použitelný pro libovolný systém PLC, ale pro jeho implementaci bude používat individuální vývojový systém. Většina výrobců PLC používá vlastní vývojový systém, pouze výjimečně se vyskytují výrobci nabízející univerzální vývojový systém (např. CoDeSys). I tento systém je ale nutné přizpůsobit pro konkrétní PLC.

3.1.2 CP 443-1

Komunikační procesor CP 443 – 1 je určen pro systémy S7-400. Přidává další funkcionalitu tím, že umožňuje připojit S7 – 400 k rozhraní Industrial Ethernet. CP samostatně zvládá přenos dat přes sběrnici Ethernet pomocí svého vlastního procesoru, který zprošťuje CPU od většiny komunikačních úkolů a manipulací s protokolem. Multiprotokolové provozní vrstvy 1 až 4 odpovídají mezinárodním normám ISO/OSI a TCP/IP s nebo bez RFC 1006.



Obr. 16. CP 443-1

Konfigurace CP je možná přes MPI nebo Ethernet LAN s maximálním počtem 48 připojených stanic. K dispozici jsou následující protokoly:

- S7 – pro komunikaci založenou na 7 vrstvách referenčního modelu ISO/OSI mezi

SIMATIC S7/M7/C7 a PC.

- SEND/RCV – pro komunikaci založenou na 4 vrstvě mezi SIMATIC S7, SIMATIC S5 a PC/PG.
- MMS/MAP – pro otevřenou komunikaci založenou na 7 vrstvě mezi SIMATIC S7, SIMATIC S5 a PC/PG a systémy nepatřícími Siemens.

3.1.3 PC stanice

PC stanice plní na stávající lince funkci hlavního operátorského panelu pro obsluhu linky. Rozhraní mezi strojem a operátorem zajišťuje počítač s vizualizačním software WinCC. Připojení PC stanice je provedeno přes síťovou kartu, která propojuje operátorský panel s komunikační kartou PLC. PC stanice bude plnit současně také funkci jak OPC serveru, tak na ní poběží i OPC klient. Bezpečnost aplikace je zajištěna tím, že není žádné spojení mezi PC stanicí a externí sítí.

3.2 Software

V této práci je použito více typů softwarových prostředků. Klientská aplikace je vytvořena v prostředí Microsoft Visual Studio jazykem C# běžící na platformě .NET Framework. Linka je řízena PLC SIMATIC S7-400 využívající programovací software STEP 7. Vizualizace je vytvořena pomocí software WinCC. Nezbytnou součástí použité komunikační sítě je SIMATIC NET.

Tab. 3. Softwarové komponenty

Komponenta	Stručný popis
.NET Framework 4.0	Běžné prostředí ke spuštění aplikací v prostředí Windows.
Microsoft Visual Studio 2013	Balík nástrojů a služeb určený k vývoji softwarových aplikací v prostředí Windows.
SIMATIC NET V 7.0	Standard pro komunikační protokoly společnosti Siemens.

Komponenta	Stručný popis
Step 7 Verze 5.5	Programovací software používaný u řídicích systémů SIMATIC od společnosti Siemens.
WinCC 6.2	Vizualizační software společnosti Siemens používaný v průmyslovém prostředí.

3.2.1 Microsoft .NET Framework

Microsoft .NET Framework jako nejrozšířenější platforma pro osobní počítače s operačním systémem Windows je nedílnou součástí mnoha aplikací běžících na Windows. Tento soubor ke stažení je určen pro lidi, kteří potřebují .NET ke spuštění aplikace na svém počítači. Pro vývojáře .NET Framework poskytuje komplexní programovací model pro vytváření aplikací, které mají vizuálně uživatelské zkušenosti a bezproblémovou a bezpečnou komunikaci. Tato technologie se skládá ze společného jazykového modulu runtime a knihovných tříd .NET Framework. Společný jazykový modul runtime je základem .NET Framework. Můžeme si runtime představit jako zprostředkovatele, který spravuje kód v době provádění kódu, poskytuje základní služby, jako je správa paměti, správa vláken, a vzdálené komunikace, a zároveň prosazuje bezpečnost a jiné formy přesnosti kódu. Ve skutečnosti, představa o řízení kódu je základním principem runtime. Kód, který se zaměřuje na runtime je znám jako řízený kód, zatímco kód, který necílí na runtime je známý jako neřízený kód. Knihovna tříd je komplexní, objektově orientovaná kolekce opakovaně použitelných typů, které můžete použít pro vývoj aplikací, od tradičního příkazového řádku přes aplikace grafického uživatelského rozhraní (GUI) až po aplikace založené na nejnovějších inovacích poskytované ASP.NET, jako jsou například webové formuláře a webové služby XML. .NET Framework může být hostitelem neřízených komponent, které načítají společný jazykový modul runtime do svých procesů a zahajují provádění řízeného kódu, čímž se vytváří softwarové prostředí, které může využívat řízené i neřízené funkce. Internet Explorer je příkladem neřízené aplikace, která je hostitelem modulu runtime (ve formě rozšíření typu MIME). Použití aplikace Internet Explorer jako hostitel runtime umožňuje vložit řízené komponenty nebo ovládací prvky Windows Forms v HTML dokumentech. V tomto projektu byla provedena aktualizace .NET Framework na verzi 4.0, kterou vyžadovala vytvořená klientská aplikace.

3.2.2 Microsoft Visual C#

Microsoft Visual C# je výkonný, ale přitom jednoduchý jazyk zaměřený především na vývojáře aplikací na platformě .NET Framework. Zdědil velké množství toho nejlepšího z jazyků C++ a Microsoft Visual Basic, ale jen málo z jejich nesrovnalostí a anachronismů, takže výsledkem je čistší a logičtější jazyk. Jazyk Microsoft Visual C# je výkonný, komponentně orientovaný jazyk. Hraje velmi důležitou roli v architektuře Microsoft .NET Framework a někteří lidé v tomto případě poukazují na podobnou roli jazyka C při vývoji systému Unix.[5]

Visual Studio podporuje Visual C# s celou řadou nástrojů, od plně vybaveného editoru kódu, přes kompilátor, šablony projektu, návrháře, průvodce kódu, výkonné ladící programy až po mnoho dalších nástrojů. Díky knihovně tříd rozhraní .NET Framework je možný přístup k řadě služeb operačního systému a dalším navrženým třídám, které dovolují výrazně urychlit vývoj.

3.2.3 SIMATIC NET

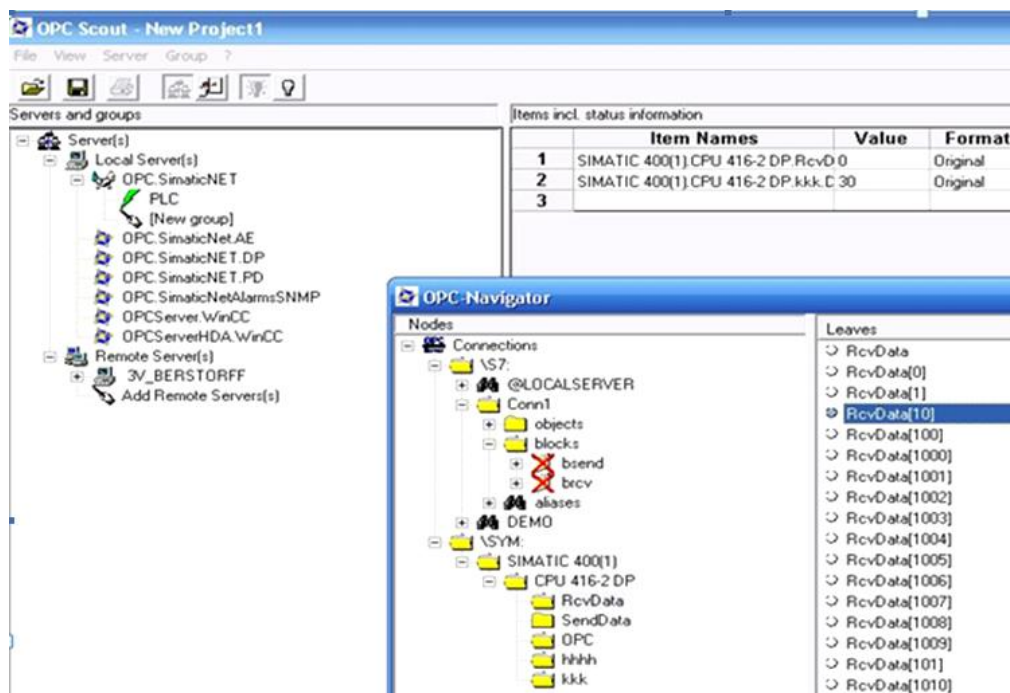
Otevřený sběrniceový systém SIMATIC NET je název rodiny průmyslových komunikačních sítí a produktů společnosti Siemens. Je to mezinárodní standard pro komunikační protokoly, který poskytuje jednotnou komunikaci od řízení pohonů přes čidla až po celosvětovou síť. Je založen na národních a mezinárodních normách v souladu s referenčním modelem ISO / OSI. SIMATIC NET poskytuje následující komunikační sítě:

- Průmyslový Ethernet (SINEC H1) - V široké míře uznávanou a používanou sběrnici pro přenosy dat od úrovně řízení procesů až po nadřazené lokální nebo rozlehlé sítě (více než 1 000 účastníků) je průmyslový Ethernet s úrovní řízení založené na standardu IEEE 802.3. Plný duplex dosahuje zdvojnásobení objemu dat s vyloučením kolizí. Přepojování dovoluje dynamicky rozdělit síť na několik segmentů a souběžně přenášet několik telegramů.
- PROFIBUS - Sběrnice Profibus je určena pro střední rozsah komunikační výkonnosti. Existuje několik variant určených pro různé typy komunikujících zařízení. Nejpoužívanější Profibus-DP je protokol pro připojování vzdálených jednotek s velmi rychlou odezvou. Umožňuje synchronizovat cyklus sběrnice s

přesností na zlomky milisekundy a díky tomu vyhovuje požadavkům regulačních úloh na vzorkování v přesných časových intervalech.

- AS-Interface – Actuator-Sensor-Interface je jednoduchá síť určená pro binární i analogové snímače a výkonné prvky. V síti, kde je řízeno až 62 zařízení typu slave, je dosažena reakční doba maximálně 10 ms. Výhodou je, že po běžné stíněné dvoulince mohou být přenášeny napájení i signál.

Připojení programovatelných automatů k sítím SIMATIC NET je fyzicky realizováno prostřednictvím rozhraní integrovaných v CPU (např. DP) nebo samostatných komunikačních procesorů (CP) nebo modulů (IM). Všechny komunikační prvky se konfigurují a parametrizují pomocí nástroje NetPro zabudovaného v prostředí STEP7. Podpora OPC UA je integrována do SIMATIC NET od verze 7.0.



Obr. 17. OPC Scout

Součástí instalace je jednoduchý OPC klient OPC Scout, který slouží ke kontrole funkčnosti OPC serveru. Ve své stromové struktuře nabízí všechny dostupné servery včetně OPC.SimaticNET., na který je možné po kliknutí přidat nové PLC. Dalším kliknutím na PLC se otevře "OPC-Navigator", který načte strukturu připojeného PLC. Pro další práci s OPC Scoutem se vybere konkrétní proměnná a přetáhne se do pravého sloupce. Tímto způsobem se sestavuje tabulka proměnných, která umožňuje monitorovat

nebo přepisovat hodnoty proměnných. OPC Scout jako klient s omezenými funkcemi, neumožňuje plnit další požadavky na vyhodnocování dat. Data můžeme pouze sledovat nebo případně přepisovat, ale už nemůžeme s nimi dále pracovat. Klient OPC UA Client, který je předmětem této práce, bude umožňovat datový výstup z aplikace. Navíc bude nabízet možnost data vzorkovat. Nespornou výhodou nové aplikace je přehlednější a praktičtější monitoring blokové služby. OPC Scout určitě není možné využít pro potřeby oddělení Průmyslového inženýrství nebo Technologie. Dalším problémem OPC Scout představuje složitější obsluha aplikace, kdy se musí provést několik úkonů, než se dostaneme k datům. Když k tomu připočteme i jazykovou nevybavenost potenciálních uživatelů, je pro běžné používání tento nástroj prakticky nepoužitelný.

3.2.4 STEP7

STEP7 je konfigurační a programovací software používaný u řídicích systémů SIMATIC společnosti Siemens. Se STEP 7 je možné programovat jak automatizační systémy založené na PLC (např. S7-300, S7-400 a C7), tak i řídicí systémy běžící na PC (WinLC).

STEP7 využívá centrální nástroj SIMATIC Manager. V programu SIMATIC Manager se pracuje s objekty ze světa STEP 7. Tyto „logické“ objekty odpovídají reálným objektům. Jeden projekt v sobě obsahuje celé zařízení, jedna stanice odpovídá jednomu řídicímu systému. Jeden projekt může obsahovat více stanic, které jsou vzájemně propojeny např. sítí MPI. V jedné stanici je zasunuta jedna CPU, která obsahuje program. Program je znovu „schránka“ pro další objekty, jako např. Objekt bloky (Blocks), který mimo jiné obsahuje přeložené bloky.[1]

Ve STEPu 7 jsou tři základní programovací jazyky LAD, FBD a STL, které je možné libovolně kombinovat.

3.2.5 WinCC

Vyšší úroveň v hierarchii řízení je nejčastěji označovaná zkratkou SCADA/HMI (Supervisory Control and Data Acquisition / human machine interface) nebo také vizualizační systém. Navazuje na řízení technologie pomocí PLC a slouží k vytvoření programů pro grafické operátorské rozhraní jednotlivých zařízení. Vizualizační nástroje

zviditelňují řízené procesy, poskytují obslužné funkce a monitorování dat. WinCC můžeme rozdělit na dva základní komponenty:

- Vývojové prostředí slouží ke konfiguraci a editaci projektu.
- Runtime je software pro zpracování vizualizace. Během procesních operací vykonává projekt a zajišťuje konečnou vizualizaci

WinCC v tomto projektu pracuje pod operačním systémem od Microsoft – Windows XP Professional.

II. PRAKTICKÁ ČÁST

4 KONFIGURACE OPC UA SERVERU

K vytvoření OPC UA serveru se musí provést několik činností. Na straně PLC bude provedena úprava hardwarové konfigurace, na straně PC nastavení OPC UA serveru.

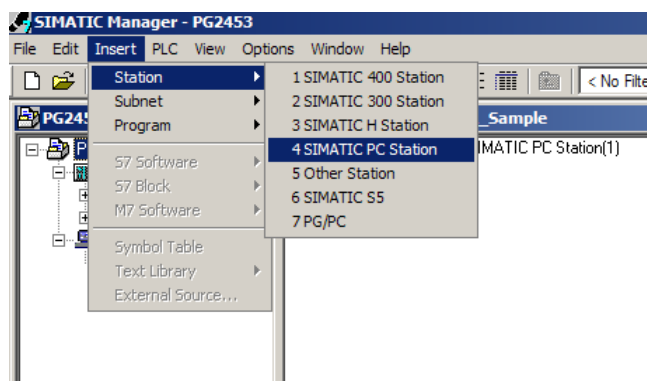
4.1 Úprava hardwarové konfigurace

K vytvoření hardwarové konfigurace slouží integrovaná součást programu SIMATIC Manager, nástroj HW Config. V původním projektu je v hardwarové konfiguraci vloženo CPU 416-2DP a CP443-1.

Tab. 4. Použité IP adresy

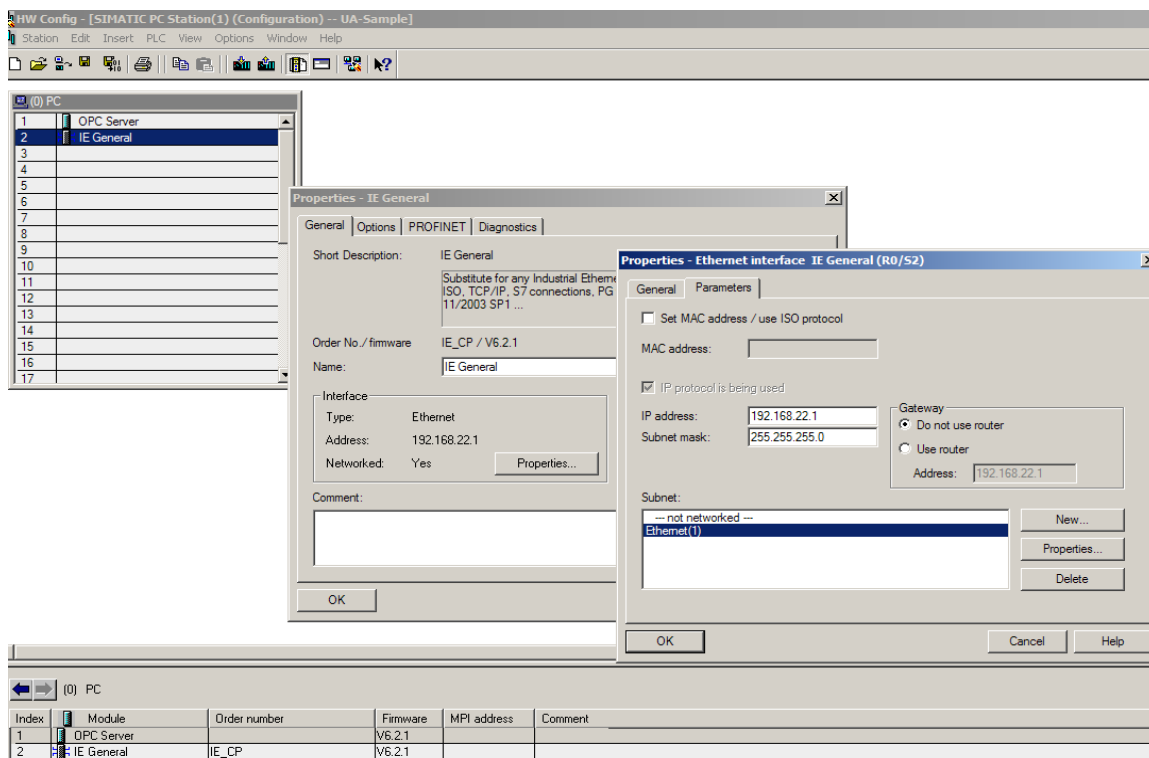
Adresa	Stanice	Popis
192.168.22.2	CP 443-1.	Komunikační modul PLC.
192.168.22.1	PC OPC Server.	Síťová karta PC.

Ve vlastnostech CP443-1 se nastavuje IP adresa (v tomto projektu 192.16.22.2), kterou bude používat tato komunikační karta a maska podsítě. MAC adresa je zadávána jenom v případech, kdy stanice komunikuje pře ISO 4 transportní úroveň.



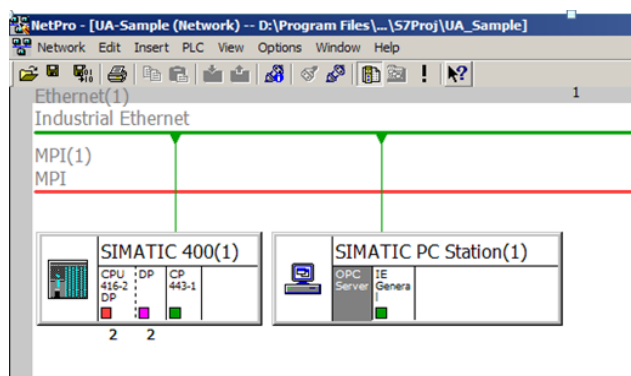
Obr. 18. Vložení PC stanice

Prvním krokem k vytvoření OPC UA serveru je vložení stanice SIMATIC PC Station do hlavní struktury projektu. Nová stanice se vkládá do aktuálního projektu přes menu Insert -> Station -> SIMATIC PC Station nebo dvojklikem na hardwarový objekt. Název PC stanice musí být stejný jako jméno počítače, na kterém má běžet OPC UA server.



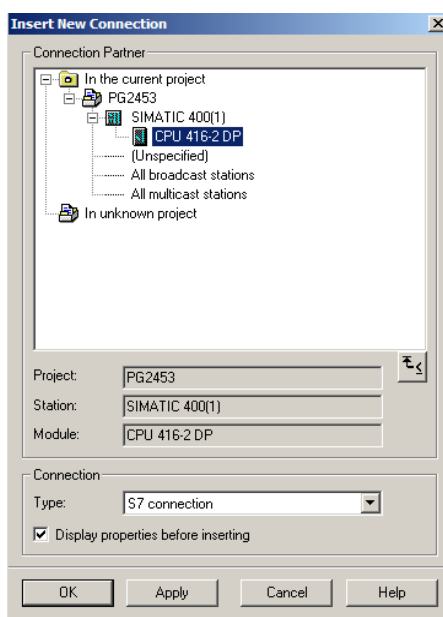
Obr. 19. Konfigurace PC stanice

Po otevření SIMATIC PC Station se otevře editor s připraveným rámem modulů nazývaným rack. Po výběru z hardwarového katalogu jsou do rámu vloženy moduly. Jednotlivé řádky v rámu, označované jako sloty, musí souhlasit s indexem, který je přiřazen v Station Configuration editoru konfigurovaného na straně PC Station. Do prvního slotu je vložen OPC server, do druhého ethernetová síťová karta pojmenovaná v projektu IE General. Otevřením vlastností objektu síťové karty se zobrazí dialogové okno, kde se nastaví IP adresa 192.168.22.1 a maska podsítě.



Obr. 20. Konfigurační nástroj NetPro

K vytvoření S7 spojení OPC UA serveru a programovatelného automatu slouží výkonný konfigurační nástroj projektů NetPro, který je určen k práci se síťovými objekty. NetPro



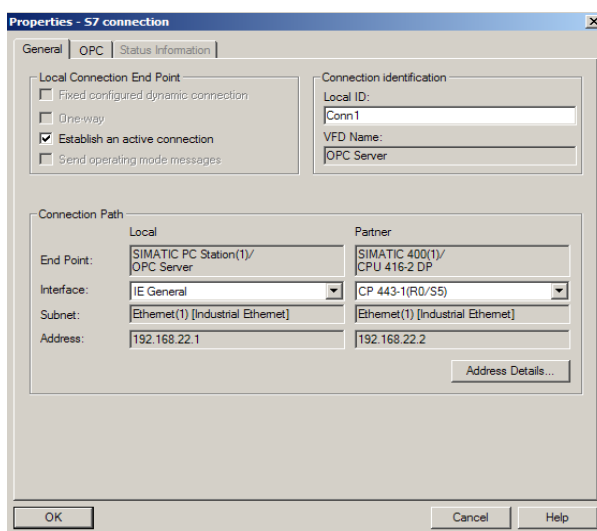
Obr. 21. Nové S7 spojení

má s HW Config společných většinu konfiguračních nastavení, ale jenom v NetPro se nastavuje propojení jednotlivých stanic v síti MPI, Profibus nebo Industrial Ethernet. V tomto projektu je spojení vytvořeno přes Industrial Ethernet.

Local ID	Partner ID	Partner	Type	Active connection p	Subnet
Conn1	1	SIMATIC 400(1) / CPU 416-2 DP	S7 connection	Yes	Ethernet(1) [IE]

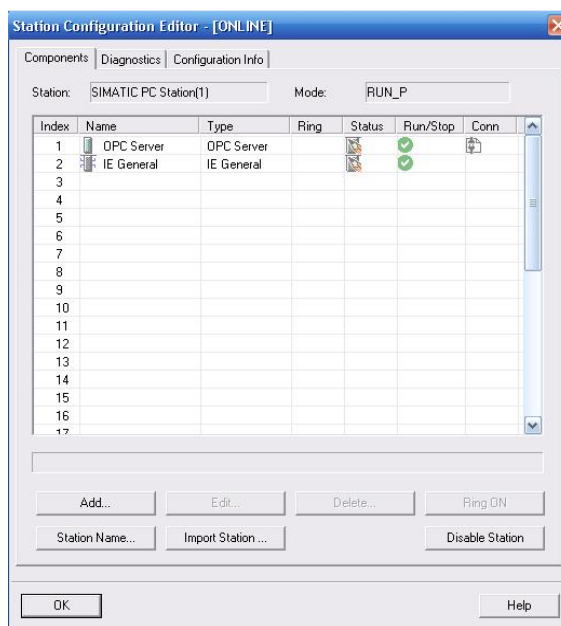
Obr. 22. Tabulka spojení

Kliknutím pravým tlačítkem na OPC Server se zobrazí kontextové menu, které nám nabídne možnost vložit nové spojení.



Obr. 23. Vlastnosti S7 spojení

Pro vložení stačí potvrdit CPU 416-2DP a typ spojení S7. Po potvrzení se vytvoří tabulka spojení, do které jsou zapsány všechny zvolené parametry spojení. Dvojklikem na tabulku spojení nebo přímo na OPC Server v grafické části NetPro se otevře dialogové okno s vlastnostmi S7 spojení, kde se ukazuje identifikátor spojení (v našem případě Conn1) a oba partneři komunikace. Po uložení a kompilaci se vytvoří hardwarová konfigurace uložená v adresáři projektu v binárním XDB souboru, který se bude dále využívat při importu do Station Configuration editoru na OPC UA server.



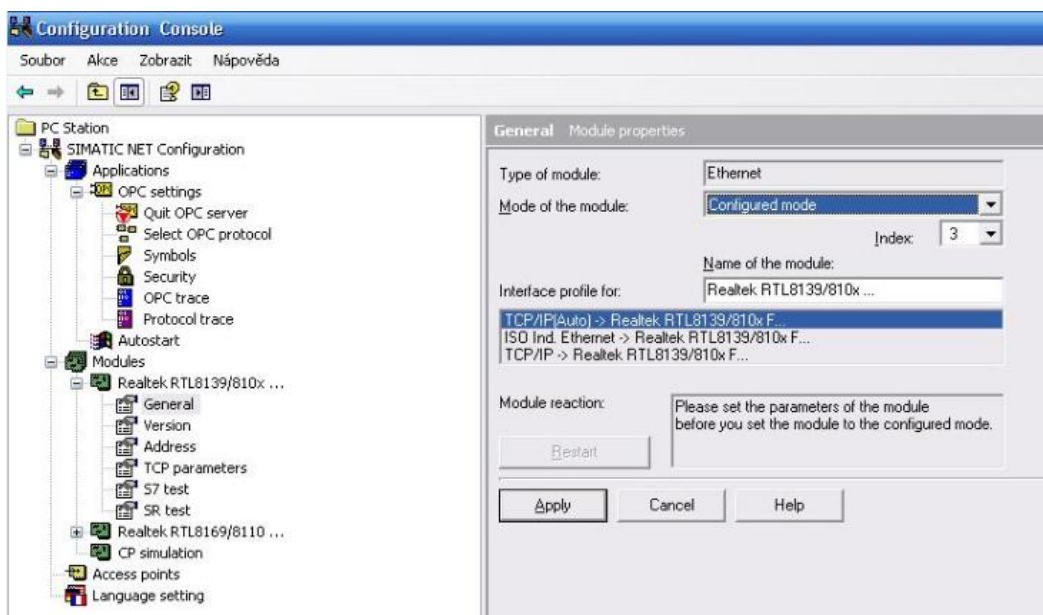
Obr. 24. Station Configuration Editor

4.2 Konfigurace OPC UA serveru SIMATIC.NET

Protože podpora OPC UA je integrována do SIMATIC NET až od verze 7.0, je nutné nejdříve přeinstalovat starší verzi SIMATIC NET. Ke konfiguraci stanice slouží nástroj Station Configuration Editor, kde se konfiguruje jednotlivé komponenty stanice. K tomu se využívá v předchozí kapitole vytvořený XDB soubor. Pomocí tlačítka *Import Station* se provede konfigurace do OPC UA serveru. Pokud všechno proběhne v pořádku, mělo by se v připojení online objevit OPC server i síťová karta v režimu Run a na řádku OPC UA serveru ve sloupci Conn zobrazit znak značící identifikátor spojení Conn1.

4.2.1 Nastavení parametrů OPC

Pro dokončení konfigurace je dalším krokem nastavení parametrů OPC pomocí Configuration Console, kterou najdeme pod nabídkou Start v adresáři SIMATIC NET. Nejdříve je nutné upravit „Mode of the module“ na Configured mode. Dále se musí vybrat konkrétní komunikační protokol, který se bude používat. Konzole umožňuje zkontrolovat nastavení sítě, IP adresu, masku sítě, bránu sítě, DHCP server, nastavení přístupového bodu S7ONLINE a taky otestovat spojení mezi serverem a PLC.



Obr. 25. Configuration Console

4.3 OPC UA Security

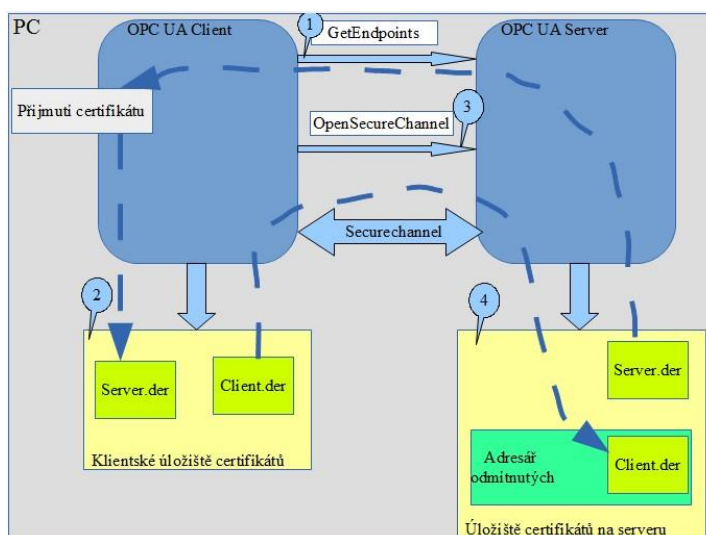
Zabezpečený komunikační kanál je na aplikační úrovni aktivní po celou dobu trvání relace. Proto jsou uživatelé autentizováni pouze při navazování relace, kdy se klient a server domluví na zabezpečeném komunikačním kanálu, vymění si softwarové certifikáty a informace o svých schopnostech. Výměna certifikátů mezi klientem a serverem je znázorněna na následujícím obrázku.

Na obrázku jsou označeny následující body:

1. Předtím, než se klient může připojit k serveru, potřebuje získat informace. Mezi důležité informace můžeme zařadit například bezpečnostní mechanismy, protokoly a adresu pro připojení požadované serverem. Tato informace naznačuje tzv.

koncový bod. K dispozici jsou koncové body serveru dodávány s voláním GetEndpoints. Při popisu koncových bodů server také poskytuje svůj certifikát.

2. Jakmile jsou koncové body vybrány s nastavením zabezpečení, uživatel je dotázán, zda chce přijmout certifikát. Pokud ano, je uložen v úložišti certifikátů klienta.
3. Při volání OpenSecureChannel je přenesen na server certifikát klienta. Jestliže certifikát není známý, na serveru bude uložen v adresáři odmítnutých.
4. Pomocí konfiguračního nástroje na serveru mohou být akceptovány certifikáty z adresáře odmítnutých a následně přesunuty do úložiště certifikátů serveru.



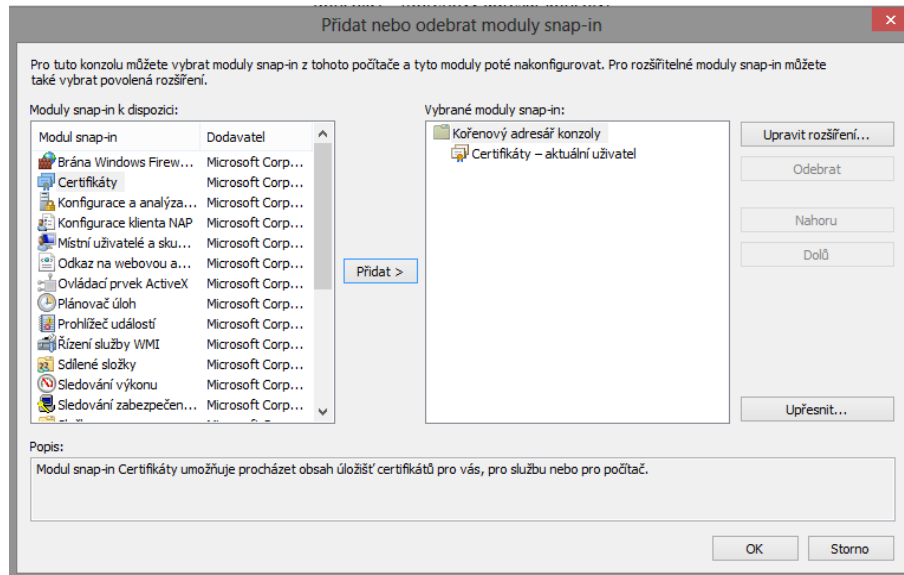
Obr. 26. Výměna certifikátů mezi klientem a serverem

OPC UA klient v tomto projektu používá úložiště certifikátů Windows. Na tomto místě se nachází veřejný certifikát klienta UA test Client, který klienti generují při prvním spuštění. Server OPC UA používá vlastní adresář s certifikáty, který je nezávislý na úložišti certifikátů Windows. Místo, ve kterém OPC UA server ukládá a spravuje vlastní certifikáty klientů OPC UA je datový adresář serveru OPC UA.

4.3.1 Správa certifikátů

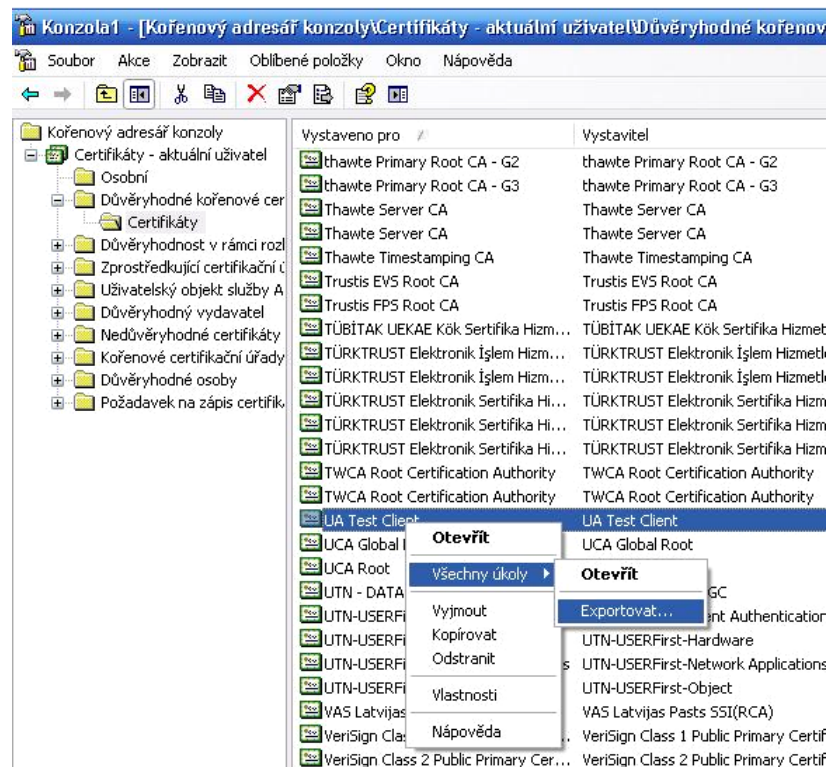
Importovaný certifikát se ukládá do úložiště na místě, které obsahuje klíče k certifikátu. K tomu se využívá konzole Microsoft Management Console (MMC). Konzole MMC hostí a zobrazuje nástroje pro správu, vytvořené společností Microsoft a dalšími dodavateli

softwaru. Tyto nástroje se nazývají moduly snap-in a používají se pro správu hardwaru, softwaru a síťových komponent systému Windows.



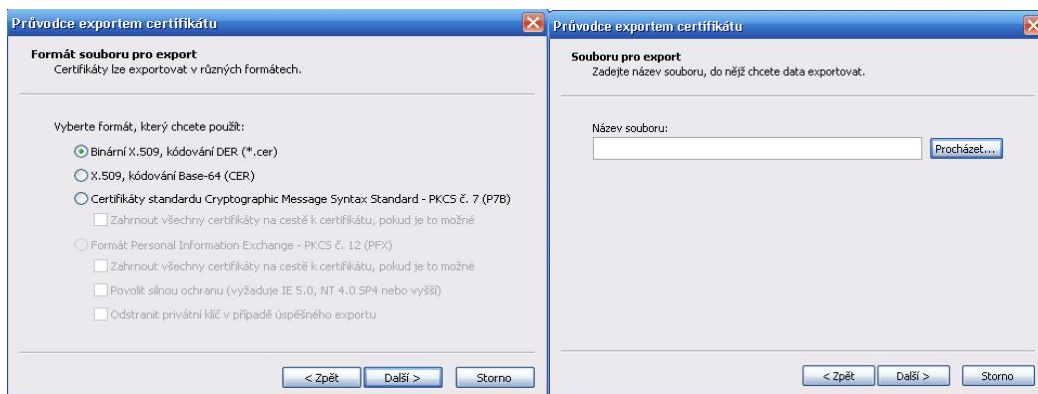
Obr. 27. Přidat nebo odebrat moduly snap-in

Prvním krokem je otevření Management console. Konzole se otevře na klientském počítači přes nabídku Start a funkci Spustit po zadání klíčového slova MMC do zadávacího pole.



Obr. 28. Certifikáty

Otevře se dialogové okno pro výběr modulů Snap-in. V nabídce File je volba Přidat nebo odebrat modul Snap-in, kde se musí vybrat volba Přidat. Ke správě certifikátů v místním úložišti slouží možnost místní počítače. Pro nezbytnou bezpečnou komunikaci se vzdálenými servery SIMATIC NET OPC UA musí být nejprve tyto certifikáty exportovány ze správy certifikátů v klientském počítači.

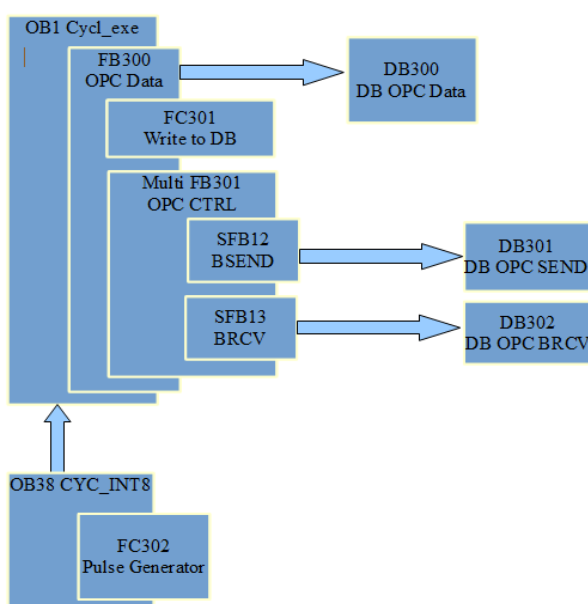


Obr. 29. Formát souboru pro export a uložení certifikátu

Dále je potřeba vybrat certifikáty pro místní účet počítače a certifikát, který má být exportován. Dalším krokem je kliknutí pravým tlačítkem myši a zvolení volby Všechny úkoly a Exportovat... V další fázi je nezbytné zvolit formát, který se bude používat. Po zvolení formátu "Binární X.509, kódování DER (*.cer)" je nutné pro dokončení exportu vybrat tlačítka *Další* a *Dokončit*. Soubor se umístí do složky se stejným názvem, ve kterém jsou také umístěny certifikáty serveru. Nakonec se provede zadání názvu souboru a uložení do souboru s certifikátem. Protože přípona souboru musí být také "DER", musí se ručně přepsat přípona z "*. Cer" na "*. Der".

5 VYTVOŘENÍ PROGRAMU S7

PLC program je sestaven pomocí softwaru STEP7 prostřednictvím nástroje SIMATIC Manager. Základními stavebními prvky programu jsou bloky, které se rozdělují podle účelu, na organizační bloky OB, Funkční bloky FB, funkce FC a datové bloky DB. Program pro PLC je sestaven z jednotlivých příkazů a instrukcí. Zpracování programu probíhá cyklicky a v sérii za sebou tak, že jednotlivé logické funkce jsou prováděny postupně.



Obr. 30. Struktura programu

Rozhraní mezi uživatelským programem a operačním systémem je tvořeno organizačními bloky (OB). Celý program je možné uložit v cyklicky volaném OB1, ale většinou se používá strukturované programování rozdělující program do bloků, kde každý z bloků řeší část programu. Organizační blok OB1 organizuje průběh ostatních bloků a zajišťuje cyklické volání všech částí programu umístěné v ostatních blocích. Ještě než dojde k cyklickému provádění programu, je proveden spouštěcí program pomocí OB100 - kompletní restart. Tento blok smaže tabulky vstupů a výstupů, bitové paměti a některé časovače. Následně se začne provádět program uložený v OB1. Cyklické provádění programu je možné přerušit v pevných časových intervalech pomocí organizačních bloků OB30 až OB38.

Stávající program S7 je doplněn o funkční bloky a funkce řešící přenos dat mezi PLC a OPC serverem. Program je v podstatě rozdělen do dvou částí. První část řeší používání

volaný FB 301 (OPC CTRL), který zajišťuje obsluhu řízení oboustranné komunikace blokové služby. Na obrázku Obr. 31 je vidět v deklarační tabulce deklaraci multifunkčního bloku FB301 (OPC CTRL) pojmenovaného commCTRL s datovým typem OPC CTRL a jeho zavolání v editoru. Proměnná *#commCTRL.run_COM* znamená probíhající přenos dat z WinCC. Po spuštění přenosu se prostřednictvím proměnné *#memWrite.writeStock_ON* začnou zapisovat sledovaná data do datového bloku DB300.

```
// Start přenosu dat z WinCC
A   #winCC.OPC_transfer
=   #commCTRL.run_COM

// Velikost požadované datové oblasti
L   #winCC.sizeArea
T   #commCTRL.Partner_COM.BSEND_LEN
T   #commCTRL.Partner_COM.BRCV_LEN

// Počáteční adresa pro zápis do DB - 1.blok dat
L   L#0
T   #memWrite.startAddress1

// Počáteční adresa pro zápis do DB - 2.blok dat
L   #winCC.sizeArea
L   2
/I
L   2
-I
ITD
T   #memWrite.startAddress2

// Sledovaná proměnná - 1.blok dat
L   #winCC.inpBlock1
T   #inpBlockTemp1

// Sledovaná proměnná - 2.blok dat
L   #winCC.inpBlock2
T   #inpBlockTemp2
```

Obr. 32. S7 program pro čtení dat z WinCC

Ke spuštění přenosu dat a obsluze aplikace z WinCC je určen obslužný panel ve WinCC. Před spuštěním přenosu je třeba nejdříve zvolit velikost požadované oblasti pro přenos dat. Ta je reprezentovaná proměnnou *#winCC.sizeArea*. Na základě této velikosti se vypočte počáteční adresa v datovém bloku pro druhou proměnnou *#memWrite.startAddress2*. Adresa první proměnné začíná na adrese nultého bajtu.

Dalším parametrem nutným pro správnou funkci aplikace je volba rychlosti zápisu dat do datového bloku DB300 (DB OPC Data). K tomu slouží generátor pulsu FC 302 (Pulse Generator) volaný v organizačním bloku OB38 a prováděný každých 10ms cyklu. Volba intervalu je součástí obslužného panelu ve WinCC, odkud se zvolená hodnota uloží do proměnné *"DB OPC Data"*. *winCC.setInterval*. Generátor pulsu na základě zvoleného

intervalu vyše puls *"pulsInterrupt"*. Každé vyslání pulsu zajistí zápis proměnné do datového bloku.

```

Network 1: Generátor pulsu
Generátor pulsu volaný cyklickým přerušením vytvoří puls "setNumbPuls" x 10 ms.

// Kontrola, jestli ve WinCC je zadaná hodnota
L 0
L "DB OPC Data".winCC.setInterval
==I
JNB x1

// Inicializační hodnota
L 100
T "DB OPC Data".winCC.setInterval

x1: CALL "Pulse Generator"
start := "DB OPC Data".winCC.OPC_transfer
setNumbPuls := "DB OPC Data".winCC.setInterval
firstPuls := "pulsInterrupt"
actNumbPuls := "actPuls"

```

Obr. 33. Generátor pulsu

5.1.1 FC301 – Write to DB

Prostřednictvím této funkce se plní datový blok DB300 (DB OPC Data) hodnotami sledovaných proměnných *#inpBlockTemp1* a *#inpBlockTemp2*.

```

Network 6: Zápis proměnných do DB
Hodnota pulsu v zásobníku odpovídá adrese v DB.
Rychlost změny v zásobníku určuje puls z OB38 (#pulsInterrupt).

L #memWrite.noPulsStock
T #noPulsStock_TempI

// 1.blok dat
CALL "Write to DB"
enable := #commCTRL.run_COM
db_No := "DB OPC Data"
startAdress := #memWrite.startAdress1
offsetIndex := #noPulsStock_TempI
valueInput := #inpBlockTemp1

// 2.blok dat
CALL "Write to DB"
enable := #commCTRL.run_COM
db_No := "DB OPC Data"
startAdress := #memWrite.startAdress2
offsetIndex := #noPulsStock_TempI
valueInput := #inpBlockTemp2
NOP 0

```

Obr. 34. Start komunikace s OPC a deklarace multifunkčního bloku

Vstupní parametr *offsetIndex* nastavuje adresu pro zápis proměnné. Tato adresa je inkrementována proměnnou *#memWrite.noPulsStock*, u které je možné nastavit rychlost inkrementace podle zvoleného intervalu v obslužném panelu WinCC.

5.2 FB301 – OPC CTRL

V bloku se řeší řízení oboustranné komunikace mezi S7 a PC stanicí. Základem oboustranné komunikace jsou systémové funkční bloky SFB 12 a SFB 13. Tyto bloky jsou integrovány v operačním systému S7-400 CPU. SIMATIC NET OPC server v PC stanici může být komunikačním partnerem pro tyto komunikační bloky. Pro tento účel se používají blokové služby S7 v OPC UA serveru, pomocí kterých jsou data vyměněny mezi PC stanicí a stanicí S7. Za tímto účelem jsou na obou stranách požadovány uživatelské programy:

- S7 stanice: volání komunikačního bloku S7 v uživatelském programu CPU
- PC stanice: přístup OPC klienta k položkám na OPC serveru

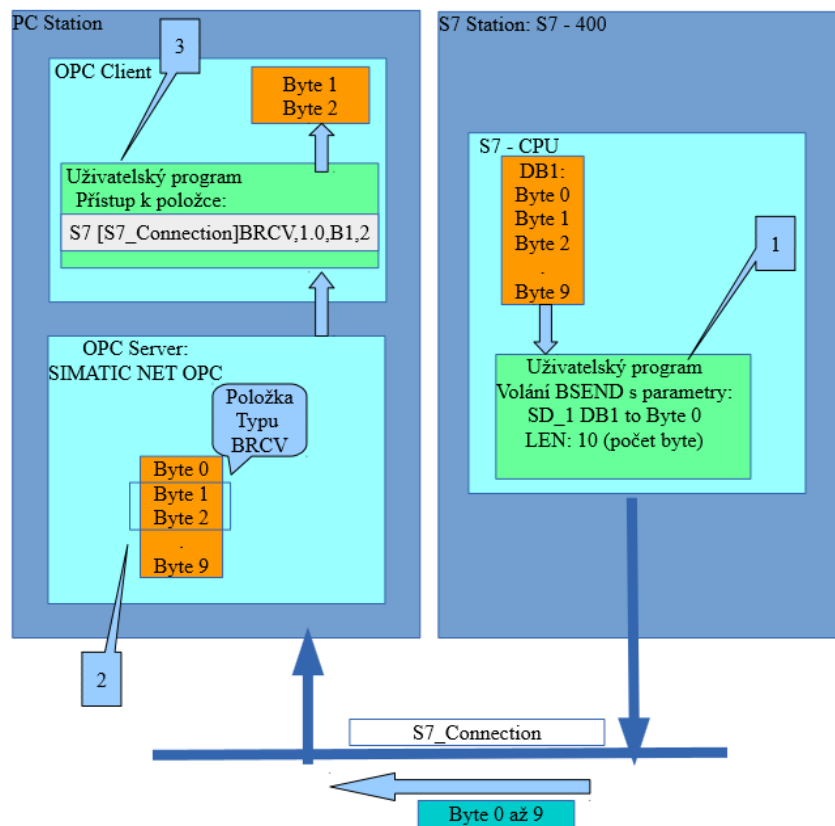
Jednotlivé části přenášených datových bloků je možné přiřadit položkám OPC serveru. Položky mohou být čteny nebo monitorovány klientem. Při čtení klient požaduje data z OPC UA serveru. Server posílá data na klienta, jestliže při monitorování došlo ke změně dat na OPC UA serveru.

V tomto projektu jsou použity bloky SFB 12 (BSEND) a SFB13 (BRCV), které představují tzv. blokované spojení pro příjem a vysílání. Tyto bloky musejí být vždy instalovány jako dvojice bloků. Na jedné straně může příjemce (Server) voláním bloku BRCV určit, kdy je připraven k přijetí nových dat od odesílatele pro další zpracování. Na druhé straně může příjemce pomocí parametru *#NDR* (nová přijatá data) zjistit, jestli data byla skutečně přijata. Tímto způsobem mohou být přenesena data až do 64KBytů. Přenos dat zatěžuje síť pouze, když komunikační partner spustí odesílání úlohy. Příjemce nemůže požadovat žádné údaje, ale musí počkat, až se data odešlou. Na následujícím obrázku je popsána sekvence komunikace při oboustranném spojení.

Na obrázku jsou popsány následující body:

1. Klient zapisuje do položky typu BSEND.
2. Celý datový blok nebo část datového bloku může být zastoupen v položce BRCV. V příkladu je znázorněna část datového bloku (bajt 1, bajt 2). Celý datový blok (10 bajtů) se přenáší do stanici S7.

3. Komunikační blok BRCV je volán uživatelským programem S7. BRCV vstupuje do datového bloku v cílové oblasti definované parametry BRCV (DB1, bajt 0 až 9).



Obr. 35. Sekvence oboustranné komunikace

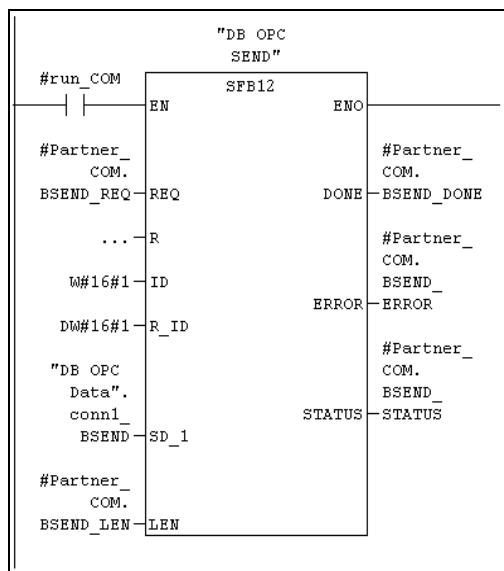
Tab. 5. Význam syntaxe S7

Syntaxe	Význam
S7:	Komunikační funkce
[S7_Connection]	Název spojení
BSEND	Typ položky
1	ID spojení
B1,2	Začátek na bajtu č. 1, délka 2 bajty

5.2.1 Blok SFB12 (BSEND)

BSEND (SFB12) odesílá data na partnerův blok SFB typu BRCV. Po nastavení proměnné #run_COM, která signalizuje probíhající komunikaci s OPC serverem, dojde k zavolání

bloku SFB12. Náběžná hrana na vstupu *REQ* zajistí přenášení dat. Parametr *SD_1* určuje adresu oblasti, do které se ukládají data z klienta. Oblast *conn1_BSEND* je definovaná v deklarační tabulce V FB300 (OPC Data). Pro správnou funkci je nezbytné zvolit stejnou délku dat pro odesílání i příjem dat. Parametr *LEN* se nastavuje ve WinCC pro oba bloky.



Obr. 36. SFB12 BSEND

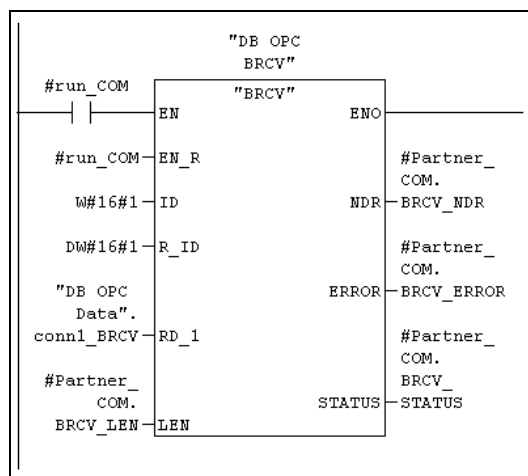
Přenos dat z uživatelské paměti je asynchronní oproti provádění uživatelského programu.

Význam jednotlivých parametrů:

- REQ – Aktivuje přenos přes náběžnou hranu.
- R – Přes náběžnou hranu aktivuje reset BSEND na inicializační stav.
- ID - Číslo spojení v tabulce spojení NetPro. V tomto projektu je ID = 1.
- R_ID – Přiřazení dvojice bloků. Parametr musí být stejný pro BSEND i BRCV.
- SD_1 – Data k odeslání do CPU (Datový blok 1, od bajtu 0, 10 bajtů).
- LEN – Délka přenesených dat (10 bajtů, od 0 do 9).
- DONE – Přes náběžnou hranu indikuje bezchybné ukončení požadavku BSEND.
- ERROR – Signalizace chyby.
- STATUS – Popis poruchového hlášení.

5.2.2 Blok SFB13 (BRCV)

BRCV (SFB13) přijímá data z bloku SFB typu BSEND vzdáleného partnera. Parametr *R_ID* musí být stejný jako u SFB12. Po zavolání bloku proměnná *#run_COM*, která signalizuje probíhající komunikaci s OPC serverem, nastaví řídicí vstup *EN_R* na 1. Po nastavení je blok připraven na přijetí dat. Parametr *RD_1* určuje adresu přijímací oblasti. Oblast *conn1_BRCV* je definovaná v deklarační tabulce v FB300 (OPC Data). Po každém přijetí dat je partnerovi odesláno potvrzení o přijetí a následně je aktualizován parametr *LEN*. Bezchybné přijetí indikuje stavový parametr *NDR* hodnotou 1.



Obr. 37. SFB13 BRCV

Význam jednotlivých parametrů:

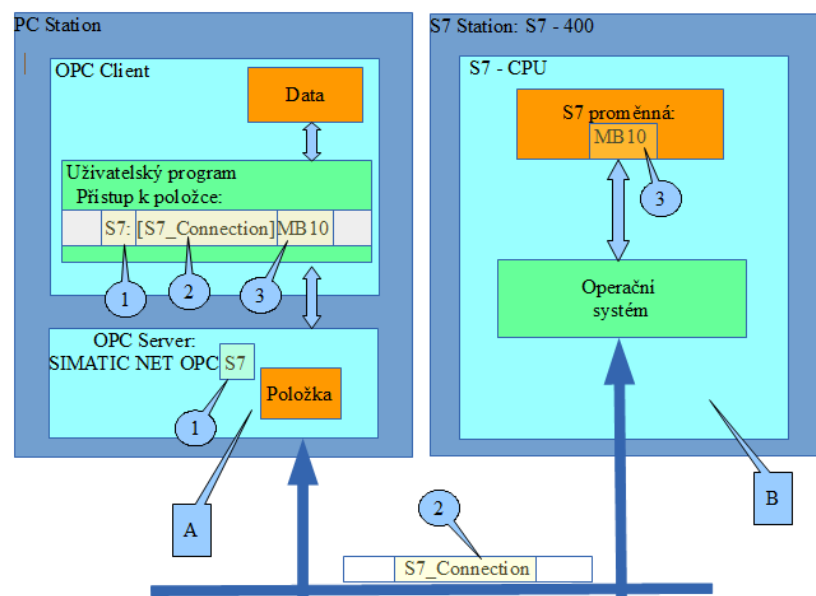
- EN_R – Blok připraven k příjmu, pokud RLO = 1.
- ID – Číslo spojení v tabulce spojení NetPro. V tomto projektu je ID = 1.
- R_ID – Přiřazení dvojice bloků. Parametr musí být stejný pro BSEND i BRCV.
- RD_1 – Přijatá data z CPU (Datový blok 1, od bajtu 0, 10 bajtů).
- LEN – Délka přijatých dat.
- NDR – Signalizace přijatých dat.
- ERROR – Signalizace chyby.
- STATUS – Popis poruchového hlášení.

5.3 Komunikace mezi S7 a PC pomocí variabilní služby

Použitím variabilních služeb může PC stanice přímo přistupovat k datům programovatelného automatu. K tomu využívá absolutního nebo symbolického přístupu.

5.3.1 Použití absolutního přístupu

Model funkce na konkrétním příkladu ukazuje absolutní přístup OPC klienta k S7 proměnné. V příkladu je použita S7 proměnná *MB10* (paměťový bajt 10). Následující obrázek a následující popis vysvětlují části položky ID a znázorňují, které z těchto částí mají vliv na funkci modelu.



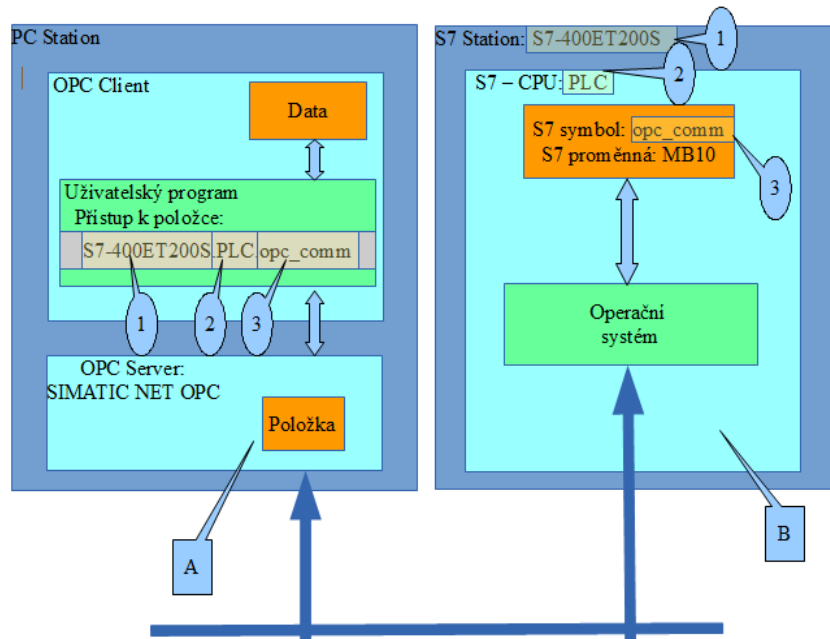
Obr. 38. Absolutní přístup

Význam jednotlivých bodů z obrázku:

- A) OPC klient přistupuje k proměnné v S7 CPU prostřednictvím položky. Za tímto účelem klient požaduje úlohy ze serveru, které server provádí. Možné úlohy jsou: čtení, zápis nebo monitorování. Předpokladem je, že položka byla vytvořena v OPC serveru.
- B) Pro tento přístup není požadovaný uživatelský program
 1. S7: komunikační funkce
 2. Název spojení
 3. Proměnná S7

5.3.2 Použití symbolického přístupu

Model funkce na konkrétním příkladu ukazuje symbolický přístup OPC klienta k S7 proměnné. V příkladu je použitý S7 symbol `opc_comm` a S7 proměnná `MB10` (paměťový bajt 10). Následující obrázek a následující popis vysvětlují části položky ID a znázorňují, které z těchto částí mají vliv na funkci modelu.



Obr. 39. Symbolický přístup

Význam jednotlivých bodů popisek z obrázku:

- A) OPC klient přistupuje k proměnné v S7 CPU prostřednictvím položky. Za tímto účelem klient požaduje úlohy ze serveru, které server provádí. Možné úlohy jsou: čtení, zápis nebo monitorování. Předpokladem je, že položka byla vytvořena v OPC serveru.
- B) Pro symbolický přístup musí být vytvořena v rámci projektu STEP 7 tabulka proměnných. Pro přístup není nutný uživatelský program.
 1. Jméno S7 stanice
 2. Jméno CPU
 3. Název proměnné

6 PROPOJENÍ VIZUALIZACE WINCC S OPC KLIENTEM

Pro jednodušší obsluhu OPC klienta byl vytvořen na úvodní obrazovce vpravo ve WinCC obslužný panel. Panel tvoří uživatelské rozhraní klientské aplikace s následujícími funkcemi:

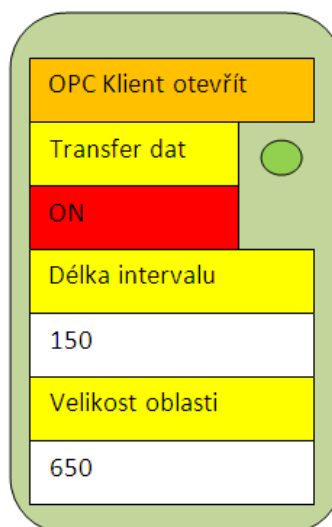
- *OPC Client otevřít* – Po kliknutí na tlačítko se otevře klientská aplikace.



```
#include "apdefap.h"
void OnClick(char* lpszPictureName, char* lpszObjectName, char* lpszPropertyName)
{
    ProgramExecute("C:\\opc\\client2.exe"); //Return-Type: unsigned long int
}
```

Obr. 40. Kód ve WinCC - otevření klientské aplikace

- *Transfer dat* – Hlavní spouštěcí tlačítko celé aplikace. Po kliknutí na tlačítko se spustí zápis sledovaných proměnných do datového bloku DB300.
- *Délka intervalu* - Zvolená hodnota představuje násobek 10ms intervalu pro zápis sledované proměnné blokovou službou.
- *Velikost oblasti* – Oblast představuje množství sledovaných dat.

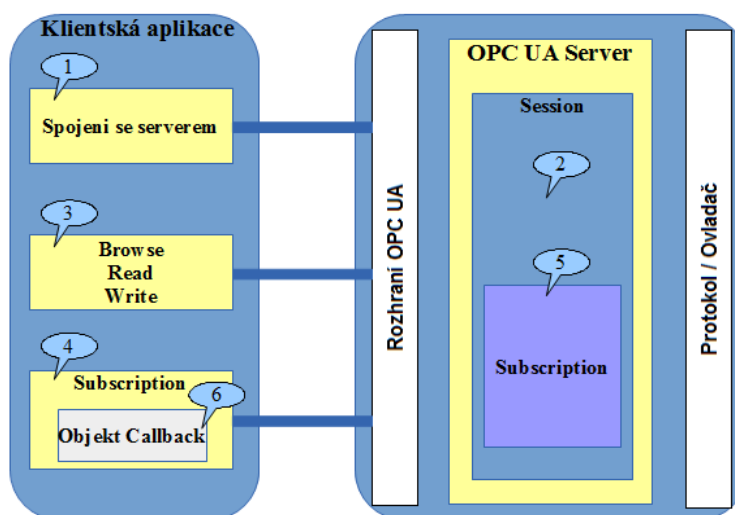


OPC Klient otevřít	
Transfer dat	<input type="checkbox"/>
ON	<input type="checkbox"/>
Délka intervalu	
150	
Velikost oblasti	
650	

Obr. 41. Obslužný panel ve WinCC

7 VÝVOJ KLIENŤSKÉ APLIKACE

V tomto projektu OPC Server zajišťuje ve funkci informačního serveru bezpečný přístup ke zpracování dat přes nové OPC UA rozhraní SIMATIC NET OPC Serveru. Vytvořený OPC UA Client v PC stanici ukazuje všechny základní funkce realizované v prostředí .NET pomocí programovacího jazyka C#. Cílem SIMATIC .NET OPC Client API je poskytnout uživatelům jazyka C# jednoduché a optimalizované knihovny tříd, které mohou být použity intuitivně a s kterými mohou být rychle vytvořeny OPC klientské aplikace pro přístup k OPC serverům produktové řady SIMATIC NET. Ke spuštění aplikace je vyžadován .NET Framework ve verzi 4.0.



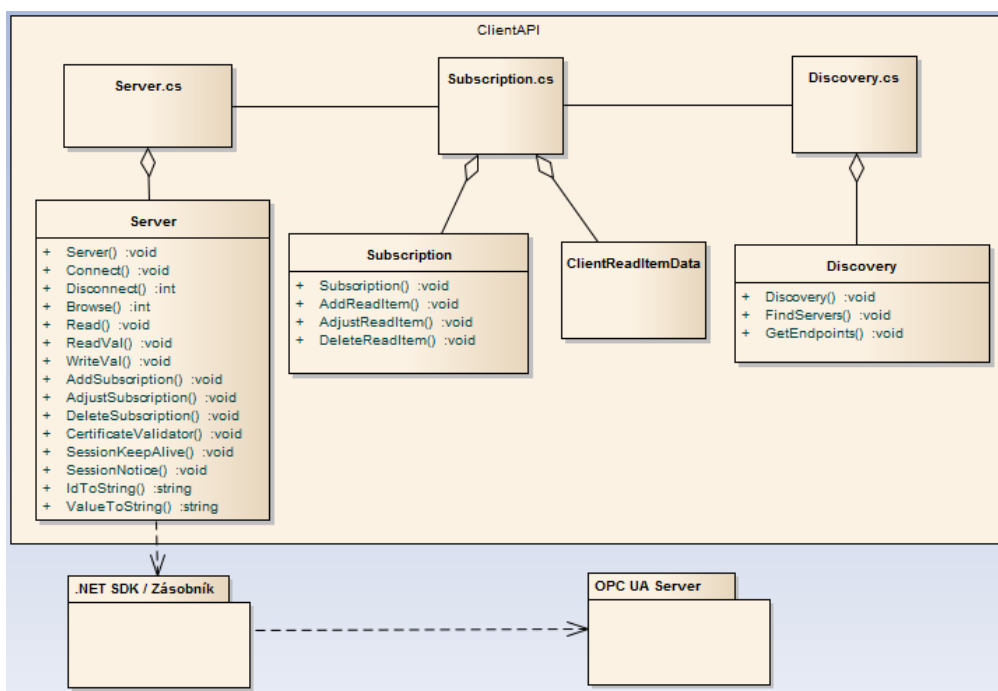
Obr. 42. Interakce OPC klienta s OPC UA

Výše uvedený obrázek ukazuje funkční bloky OPC klienta a interakci s OPC UA serverem. Význam jednotlivých bodů je následující:

1. Při vytváření spojení mezi uživatelským rozhraním a serverem OPC UA je generován na straně klienta objekt ClientAPI. Tento objekt řídí připojení k serveru a poskytuje další služby OPC UA, s výjimkou služeb spojených s podpisem.
2. Objekt relace (session) je generován na serveru prostřednictvím rozhraní OPC UA.
3. Při navázání spojení mezi uživatelským rozhraním a serverem OPC UA je také

zobrazen první stupeň adresového prostoru OPC UA serverů. V tomto procesu je použito procházení služby OPC UA rozhraní. Pokud je vybrán uzel, hodnoty atributu uzlu jsou zobrazeny v jiném okně pomocí služeb funkce Read.

4. Při registraci proměnných je vytvořen objekt Subscription ke sledování změn hodnot, který poskytuje veškeré služby OPC UA týkající se podpisu.
5. Objekt Subscription, řídící všechny podpisové nastavení, je generován na serveru prostřednictvím rozhraní OPC UA.
6. Aby bylo možné přijímat změny hodnot ze serveru, je založeno připojení zpětného volání (callback). Objekt Callback je vytvořen v klientovi a připojen k podpisu na serveru.



Obr. 43. Diagram tříd ClientAPI

7.1 ClientAPI

Diagram tříd na předcházejícím obrázku ukazuje třídy klienta ClientAPI. Tyto třídy zapouzdřují přístup k OPC UA serveru v jednoduchém a opakovaně použitelném .NET API. Třídy jsou shrnuty v knihovně .NET Siemens.OpcUA.dll, která má vazbu na .NET

SDK sestaveného v knihovně Opc.Ua.Client.dll a .NET zásobníku sestaveného v knihovně Opc.Ua.Core.dll. Knihovny Opc.Ua.Client.dll a Opc.Ua.Core.dll jsou naimportovány ze SIMATIC NET CD. Vložení těchto knihoven do projektu je nezbytné pro správnou funkci OPC UA klienta.

7.1.1 Třída Discovery

Třída je implementována v souboru Discovery.cs v projektu ClientAPI. Zapouzdřuje potřebné metody k hledání serverů FindServers a GetEndpoints. Metoda FindServers vrací seznam OPC UA serverů. Obsahuje tyto parametry:

- *url_D* - Zjištěná adresa URL serveru
- *servers* - Nalezené servery

```
public void FindServers(Uri url_D, ref ApplicationDescriptionCollection servers)
{
    try
    {
        // Vytvoření discovery klienta
        DiscoveryClient clientD = DiscoveryClient.Create(url_D);

        // Hledat servery
        servers = clientD.FindServers(null);
    }
    catch (Exception e)
    {
        throw e;
    }
}
```

Obr. 44. Metoda FindServers

Metoda GetEndpoints vrací seznam koncových bodů na serveru. Obsahuje tyto parametry:

- *url_D* - zjištěná adresa URL serveru.
- *endpoints* – zjištěné koncové body

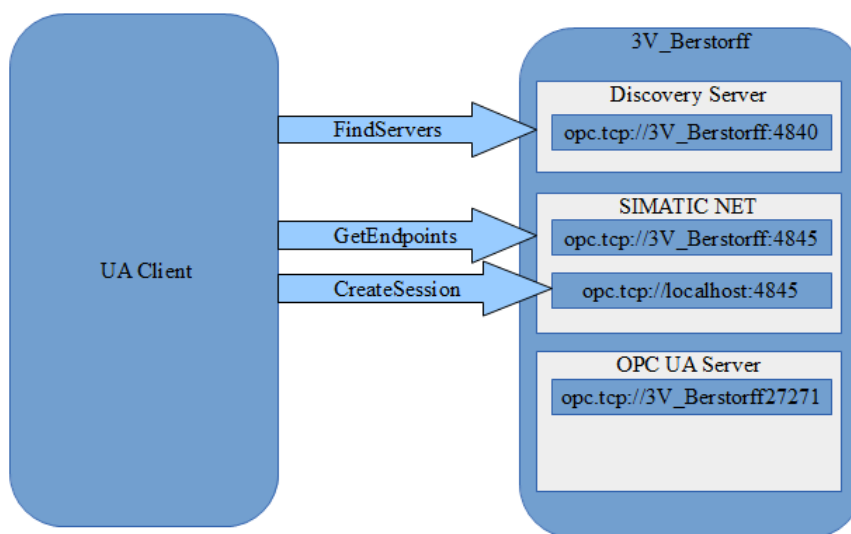
```
public void GetEndpoints(Uri url_D, ref EndpointDescriptionCollection endpoints)
{
    try
    {
        // Vytvoření discovery klienta
        DiscoveryClient clientE = DiscoveryClient.Create(url_D);

        // Získat koncové body
        endpoints = clientE.GetEndpoints(null);
    }
    catch (Exception e)
    {
        throw e;
    }
}
```

Obr. 45. Metoda GetEndpoints

Pro OPC UA Discovery byla definována služba Local Discovery Service (LDS), která svojí funkcí je srovnatelná s OPCEnum klasického OPC. LDS poskytuje seznam OPC UA serverů, který je k dispozici na místním uzlu sítě. Ve výchozím nastavení LDS používá registrovaný port 4840, proto LDS je vždy adresovatelné jako adresa URL pomocí `opc.tcp:// <Node> :4840`. Servery na PC jsou registrovány s LDS. Klient může zvolit server a navázat spojení s následujícími kroky:

- Zřízení připojení s portem 4840 a volání `FindServers`. Toto volání poskytuje seznam dostupných serverů a jejich adresy URL.
- Zřízení připojení pomocí zjištěné adresy URL požadovaného serveru a volání `GetEndpoints`. Toto volání poskytuje seznam koncových bodů s adresou koncového URL a nastavení zabezpečení koncových bodů.
- Navázání spojení s URL koncového bodu a požadované nastavení zabezpečení. Následně relace aplikace může být otevřena s `CreateSession`.



Obr. 46. Nalezení serveru

7.1.2 Třída Server

Třída Server zapouzdřuje funkce pro připojení k OPC UA serveru a přístup k jednotlivým službám. Zjednodušuje používání těchto OPC UA služeb, které jsou potřeba pro klientskou aplikaci, s výjimkou služeb pro podpis. Třída je implementována v souboru `Server.cs` v projektu `ClientAPI`. Ve třídě `Server` se volají tyto metody:

- Connect. Vytvoří zabezpečený komunikační kanál a relace na OPC UA server. Metoda má jeden parametr string *Url*, kterým je adresa URL koncového bodu.

```
public void Connect(string Url)
{
    try
    {
        endpoint.UpdateFromServer(bindingFactory);
        endpointD = endpoint.Description;
        endpointC = endpoint.Configuration;
    }

    X509Certificate2 clientCertificate
        = config.SecurityConfiguration.ApplicationCertificate.Find();

    // Nastavit callback pro handle chyby platnosti certifikátu.
    config.CertificateValidator.CertificateValidation
        += new CertificateValidationEventHandler(CertificateValidator);

    // Inicializace kanálu, který bude vytvořen serverem.
    SessionChannel channel = SessionChannel.Create(
        config,
        endpointD,
        endpointC,
        bindingFactory,
        clientCertificate,
        null);

    // Obálka kanálu s objektem relace.
    // Volání bude chybné, jestliže server neověří pravost certifikátu klienta.
    mSession = new Session(channel, config, endpoint);
    mSession.ReturnDiagnostics = DiagnosticsMasks.All;

    UserIdentity identity = new UserIdentity();

    // Vyrošení relace.
    mSession.Open("OPC UA client", identity);
}
```

Obr. 47. Ukázka kódu metody Connect

- Disconnect. Odstraní relaci na serveru a odpojí připojení zabezpečeného kanálu.

```
public int Disconnect()
{
    int result = 0;

    // Uzavře relaci
    try
    {
        mSession.Close(10000);
    }
    catch (Exception e)
    {
        return -1;
        throw e;
    }

    // Vymaže prohlížeč
    mBrowser = null;

    return result;
}
```

Obr. 48. Metoda
Disconnect

- Browse. Slouží k procházení adresového prostoru serveru. Poskytuje seznam uzlů, které je možné získat z přenesené startovacího uzlu prostřednictvím reference. Seznam výsledků lze ovlivnit pomocí nastavení filtrů.

```
public int Browse(NodeId nodeId, out ReferenceDescriptionCollection browseResults)
{
    int result = 0;

    if (mBrowser == null)
    {
        // Instance prohlížeče.
        mBrowser = new Browser(mSession);
    }

    // Filtr prohlížení: sledovat pouze hierarchické reference.
    mBrowser.ReferenceTypeId = ReferenceTypes.HierarchicalReferences;
    browseResults = null;

    try
    {
        // Aktuální volání do serveru.
        browseResults = mBrowser.Browse(nodeId);
    }
}
```

Obr. 49. Metoda Browse

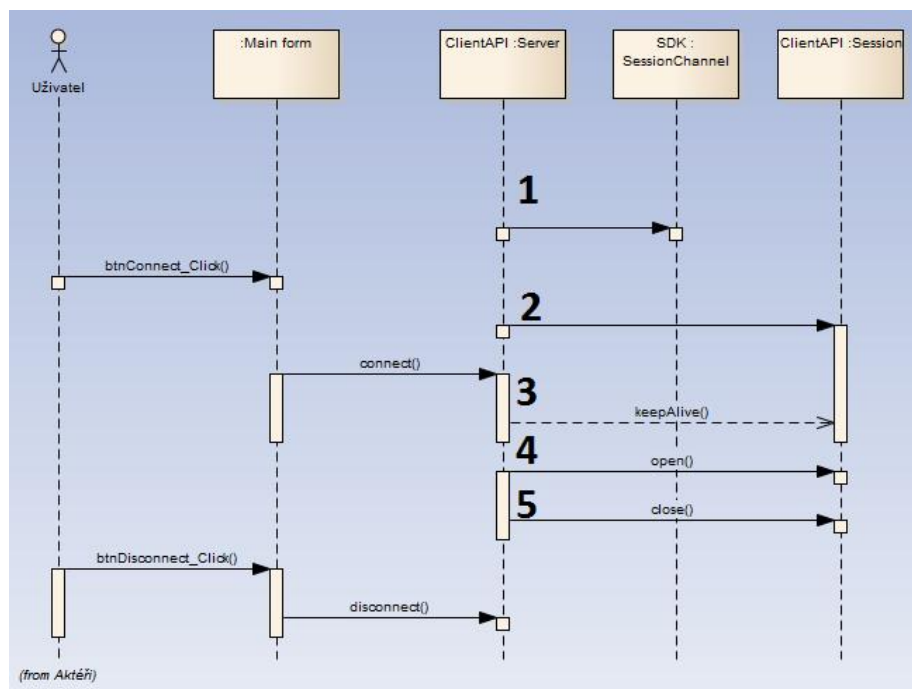
Metoda obsahuje dva parametry:

- *nodeId* - Startovací uzel
- *browseResults* - Seznam výsledků pro startovací uzel a filtr

Další metody používané ve třídě Server:

- Read. Dodává hodnoty do seznamu atributů uzlu.
- ReadVal. Dodává hodnoty atributů ze seznamu uzlů.
- WriteVal. Zapiše hodnotu atributu jedné nebo několika proměnných
- AddSubscription. Vytvoří podpis a připojí je do relace.
- AdjustSubscription. Změní nastavení podpisu
- DeleteSubscription. Odstraní existující podpis
- SessionNotice. Volána, když server OPC UA odešle odpověď (zveřejnění zprávy).
- CertificateValidator. Volána, když je certifikát serverů považován za nedůvěryhodný.

Následující sekvenční diagram ukazuje nezbytné kroky, které jsou nutné k navázání relace. Vysvětluje procesy během založení připojení k OPC UA serveru v kontextu klienta ClientAPI a jeho odpojení.



Obr. 50. Navázání relace

Na obrázku jsou popsány tyto body:

1. Metoda Connect() vytvoří objekt SDK::SessionChannel pro založení bezpečného připojení k serveru
2. V dalším kroku je generován objekt ClientAPI::Session, který zapouzdřuje kanál na server.
3. Následně objekt ClientAPI::Session registruje zpětné volání KeepAlive() na OPC UA Server.
4. Metoda Open() volá aktuální spojení mezi klientem a serverem.
5. Během provádění metody Disconnect() je zavolána metoda Close() v objektu ClientAPI::Session.

7.1.3 Třída Subscription

Třída Subscription zapouzdřuje použití podpisu pro výměnu hodnot mezi serverem a klientem. Třída je implementována v souboru Subscription.cs v projektu ClientAPI. Ve třídě jsou volány tyto metody:

- AddReadItem. Vytvoří sledované položky pro sledování změn hodnot a připojí je k podpisu.
- AdjustReadItem. Změní nastavení sledované položky.
- DeleteReadItem. Odstraní sledované položky z podpisu.

```
public void AddReadItem(NodeId varNodeId, object clientHandle,
    valueChanged callback, uint samplingRate, out object serverHandle)
{
    serverHandle = null;

    try
    {
        MonitoredItem readItem = new MonitoredItem(mSubscription.DefaultItem);
        ClientReadItemData clientData = new ClientReadItemData();
        clientData.callback = callback;
        clientData.clientHandle = clientHandle;

        // Nastavení monitorovaných položek
        readItem.StartNodeId = varNodeId;
        readItem.AttributeId = Attributes.Value;
        readItem.MonitoringMode = MonitoringMode.Reporting;
        readItem.SamplingInterval = (int)samplingRate;
        readItem.QueueSize = 1;
        readItem.DiscardOldest = false;
        readItem.Handle = clientData;

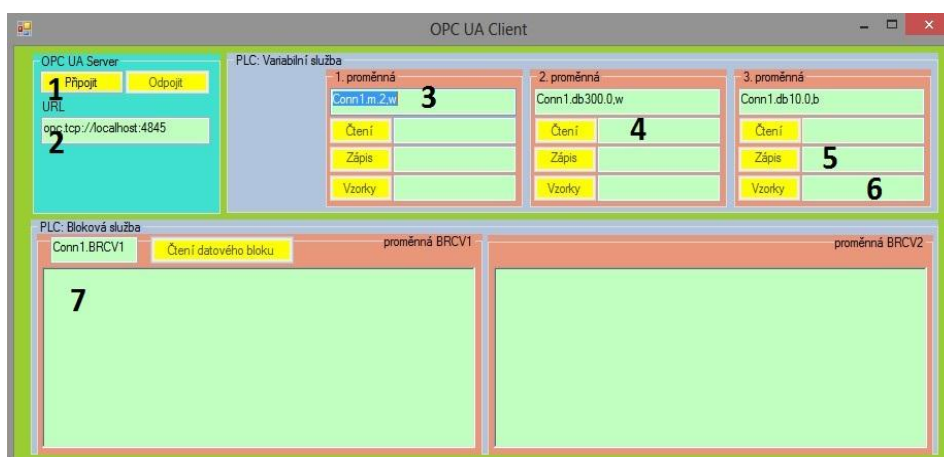
        // Přidat položky do podpisu.
        mSubscription.AddItem(readItem);
    }
}
```

Obr. 51. Metoda AddReadItem

7.2 Client

Jednoduchý příklad použití ClientAPI představuje projekt Client, který tvoří uživatelské rozhraní klienta. Pomocí klienta je možné sledovat až tři proměnné pomocí variabilní služby, které je možné dále vzorkovat nebo případně některé i přepisovat.

Další variantou je použití blokové služby se sledováním dalších dvou proměnných, s možností hlubší analýzy dat. V dialogové třídě jsou uvedeny nejdůležitější metody: například Connect, Disconnect, Read, Write. Kód lze nalézt v souboru MainForm.cs. Rozhraní jednoduchého OPC UA klienta (OPC UA Client) se ovládá pomocí tlačítek pro jednotlivé funkce. Jednoduchý příklad ukazuje použití přímého adresování S7 proměnné.



Obr. 52. Uživatelské rozhraní klienta

Na obrázku jsou označeny tyto body:

1. Tlačítka *Připojit* a *Odpojit* zajišťují připojení nebo odpojení k OPC UA serveru.
2. Do textového pole pro adresu URL serveru lze zadat URL serveru. Pro SIMATIC NET OPC server je tvořen syntaxí `opc.tcp://<název počítače>:4845`.
3. V textových polích pro identifikátor proměnné je zadána část NodeId identifikátoru. Pro jmenný prostor S7 je tvořena například: `<S7Connection>.<data Area>.<offset>,<data Type>`. NodeId pro čtení a zápis se skládá z identifikátoru a indexu jmenného prostoru. Index jmenného prostoru vyplývá z pozice jmenného prostoru v tabulce jmenného prostoru serveru. Tuto tabulku lze číst ze serveru pomocí metody Read.
4. Pomocí tlačítka *Čtení* je vytvořen podpis s NodeId a sledovaná položka v podpisu. V textových polích vedle tlačítka se zobrazí změny dat. Pokud nastane chyba, je zobrazena místo hodnot v textovém poli.
5. Tlačítkem *Zápis* se přepisuje hodnota z textového pole vedle tlačítka na proměnnou, která byla identifikována v NodeId. Aby bylo možné zapisovat, musí být nejdříve provedeno čtení, protože text z textového pole musí být převedený na datový typ vhodný pro proměnnou. Konverze je na základě datového typu, který je dodáván funkcí čtení.
6. Tlačítko *Vzorky* přečte hodnoty ze tří proměnných s uvedeným NodeId a zobrazí je

v textových polích vedle tlačítka.

7. Blokovou službou mohou být přijímány data, které jsou z CPU vysílány službou bloku BSEND. Projekt řeší sledování dvou proměnných pomocí blokové služby.

7.2.1 Třída MainForm

Funkce ve třídě MainForm lze nalézt v souboru MainForm.cs. Třída implementuje funkcionalitu hlavního dialogu klientské aplikace. Obsahuje zejména funkce řešící jednotlivé tlačítka hlavního formuláře se zpracováním událostí. Pokud dojde při volání OPC UA k výjimce, objeví se dialogové okno s chybou. Pokud dojde k chybě v jedné nebo více proměnných s voláním souvisejících proměnných, chyba se zobrazí v příslušných textových polích.

```
private void btnConnect_Click(object sender, EventArgs e)
{
    try
    {
        // Spojení s URL z textového pole URL
        mServer.Connect(txtUrl.Text);
    }
    catch (Exception ex)
    {
        MessageBox.Show("Chyba připojení:\n\n" + ex.Message);
        return;
    }

    // Přečtení tabulky jmenného prostoru.
    try
    {
        NodeIdCollection nodesRead = new NodeIdCollection();
        DataValueCollection results;

        nodesRead.Add(Variables.Server_NamespaceArray);

        // Přečtení pole z jmenného prostoru
        mServer.ReadVal(nodesRead, out results);

        if ((results.Count != 1) || (results[0].Value.GetType() != typeof(string[])))
        {
            throw new Exception("Čtení z tabulky jmenného prostoru vrátilo neočekávaný výsledek");
        }
    }
}
```

Obr. 53. Ukázka metody btnConnect_Click

Obsahem třídy jsou metody, jejichž funkce se dá vysvětlit pomocí následujících bodů:

- Metoda btnConnect_Click() zajistí připojení k OPC UA serveru prostřednictvím metody Server::Connect() z klienta ClientAPI. Řetězec URL se přenáší z příslušného textového pole. Pokud bude spojení úspěšně navázáno, bude tabulka jmenného prostoru přečtena pomocí metody Server::ReadVal(). Ve vrácené tabulce

je hledaný jmenný prostor z textového pole Namespace URI. Index v tabulce je uložen v třídě proměnných.

- Metoda `btnDisconnect_Click()` vázaná na tlačítko *Odpojit* odpojí připojení k serveru OPC UA pomocí metody `Server::Disconnect()` z klienta `ClientAPI`.
- Provedením metody `btnSample_Click()` jsou nejprve vytvořeny tři `NodeIds` z indexu jmenného prostoru a texty identifikátorů. Poté jsou přečteny hodnoty pomocí metody `Server::ReadVal()`. Výsledek je zapsán do příslušného textového pole. Výsledkem může být buď zapsaná hodnota, nebo taky chybový kód.

```
private void btnSample_Click(object sender, EventArgs e)
{
    try
    {
        NodeIdCollection nodesRead = new NodeIdCollection();
        DataValueCollection results;

        // Přidáme tři proměnné NodeIds do seznamu uzlů ke čtení
        nodesRead.Add(new NodeId(txtVar1.Text, mNameSpaceIndex));
        nodesRead.Add(new NodeId(txtVar2.Text, mNameSpaceIndex));
        nodesRead.Add(new NodeId(txtVar3.Text, mNameSpaceIndex));

        // Čtení hodnot
        mServer.ReadVal(nodesRead, out results);

        // Tisk výsledku pro první proměnnou - kontrola prvního výsledku kódu
        if (StatusCode.IsBad(results[0].StatusCode))
        {
            // Uzel selhal - vytiskne se symbolický název stavu kódu
            txtSample1.Text = StatusCode.LookupSymbolicId(results[0].StatusCode.Code);
            txtSample1.BackColor = Color.Orange;
        }
        else
        {
            // Uzel uspěl - vytisknout hodnotu jako řetězec
            txtSample1.Text = results[0].Value.ToString();
            txtSample1.BackColor = Color.Gray;
            currentValue1 = results[0].Value;
        }
    }
}
```

Obr. 54. Ukázka metody `btnSample_Click`

- Pokud nebyl vytvořen podpis, bude v metodě `btnActual_Click()` vytvořen podpis nejprve pomocí `Server::AddSubscription()`. Následně jsou vytvořeny `NodeIds` z indexu jmenného prostoru a texty identifikátorů. Poté jsou vytvářeny sledované položky prostřednictvím `Subscription::AddReadItem()`. Obdobný kód je i pro `btnActual_Click2()` a `btnActual_Click3()`.
- Metoda `ClientApiValChanged` je indikována jako funkce zpětného volání v metodě `Subscription::AddReadItem()`. V metodě se nejprve zkontroluje, zda volání prochází hlavním vláknem dialogu. Pokud tomu tak není, pak se volání přepojí do hlavního vlákna dialogu prostřednictvím `BeginInvoke`. V opačném případě není možný přístup k dialogu.

- Metoda `btnWrite1_Click()` nastavuje `NodeId` pro proměnnou 1 a volá metodu `writeNextValue()` s `NodeId` nové hodnoty jako text z textového pole a s hodnotou poslední přečtené hodnoty. Stejnou funkci mají i `btnWrite2_Click` a `btnWrite3_Click`.
- V metodě `writeNextValue()` je převedena nová hodnota z textového tvaru do datového typu, který byl doručen během posledního čtení. Potom je hodnota zapsána pomocí metody `Server::WriteVal()` a výsledek je zkontrolován.
- Pokud nebyl vytvořen podpis, bude v metodě `btnReadBlock_Click()` nejprve vytvořen podpis pomocí metody `Server::AddSubscription()`. Následně jsou vytvořeny `NodeIds` z indexu jmenného prostoru a texty identifikátorů. Poté jsou vytvářeny sledované položky prostřednictvím `Subscription::AddReadItem()`.

7.3 Datový výstup aplikace

Data zpracovávaná blokovou službou budou dále ukládány pomocí vytvořené tabulky. K tomu bude použita .Net knihovna EPPlus, která čte a zapisuje soubory Excel 2007/2010 pomocí otevřeného formátu Open Office xml (xlsx). Před vytvořením tabulky je nutné přidat do referencí projektu knihovnu EPPlus.dll a pomocí klíčového slova `using` vložit odkazy na jmenný prostor `OfficeOpenXML` a `OfficeOpenXML.Style`. K uložení dat používá třídu `FileInfo`, která zajistí vytvoření souboru s příponou `xlsx` pojmenovaného `Conn1_BRCV`.

```
////// Uložení dat do tabulkového souboru ////  
FileInfo XLSFile = new FileInfo("c:\\Documents and Settings\\Uzivatel\\Dokumenty\\Conn1_BRCV.xlsx");  
if (XLSFile.Exists)  
{  
    XLSFile.Delete();  
}  
  
using (ExcelPackage XLSPack = new ExcelPackage(XLSFile))  
{  
    ExcelWorksheet ws = XLSPack.Workbook.Worksheets.Add("Sesit");
```

Obr. 55. Vytvoření tabulky

Konstruktor této třídy obsahuje parametr typu `string` reprezentující jméno souboru s úplnou cestou. Instance třídy `ExcelPackage` nám umožní přístup k listům sešitu. Na začátku je vytvořen první list sešitu s názvem `Sesit` a doplněny některé základní údaje. `ExcelPackage`

dělá všechnu těžkou práci na vytvoření XML prvků, které jsou potřebné k reprezentaci listu aplikace Excel, vytvoří řádky, buňky atd. Třída `ws` má všechny vlastnosti a metody potřebné k vytvoření a manipulaci s listy. Podpůrné třídy (např. `ExcelCell`, `ExcelRow`, `ExcelColumn`, `ExcelHeaderFooter` atd.) poskytují vlastnosti a metody jednotlivých komponent listu. Poskytují také pomocné funkce, které usnadňují manipulaci s daty. Naplnění tabulky stejně jako výpis hodnot do klienta OPC UA Client provádí smyčka „for“. Smyčka využívá pole bajtů `valBRCV`, které reprezentuje pole hodnot blokové služby OPC serveru a zapisuje tyto hodnoty do tabulky. Pro snadnější orientaci v tabulce je první sloupec tabulky určen k rozlišení jednotlivých náměrů. Ke každému náměru je přiřazena hodnota obou proměnných blokové služby. První sloupec slouží ke spárování hodnot z tabulkového procesoru s hodnotami z klientské aplikace a hodnotami v PLC.

```
byte[] valBRCV = (byte[])value.Value;

for (int i = 0; i < valBRCV.Count() / 2; i++)
{
    // 1. sloupec tabulky
    ws.Cells[i + 1, 1].Value = i;
    ws.Cells[1, 1].Value = "Náměry";
    ws.Cells[1, 1].Style.Fill.PatternType = ExcelFillStyle.Solid;
    ws.Cells[1, 1].Style.Fill.BackgroundColor.SetColor(Color.Red);
    ws.Cells[i + 1, 1].Style.Fill.PatternType = ExcelFillStyle.Solid;
    ws.Cells[i + 1, 1].Style.Fill.BackgroundColor.SetColor(Color.Aqua);
}
```

Obr. 56. Smyčka „for“ pro výpis hodnot

Datový osmibitový typ `byte` má velikost omezenou pro celé číslo od 0 do 255, což je použitelný typ jenom pro málo proměnných. Navíc v datovém bloku DB300 (DB OPC Data) jsou sledované proměnné datového typu `Integer`. Z těchto důvodů se po přenesení pole bajtů provede sloučení vedlejších bajtů a uložení do proměnné `val`, která je datového typu `Integer`.

```
// Byte-> Int
byte lowByte = valBRCV[1 + (2 * i)];
byte hiByte = valBRCV[2 * i];
int val = (int)((hiByte << 8) | lowByte);
```

Obr. 57. Sloučení sousedních bajtů do proměnné `val`.

Uvnitř smyčky se prostřednictvím podmínek `if` rozdělí pole hodnot na dvě poloviny reprezentující dvě sledované proměnné. Výsledná data se ukládají do sešitu `Conn1_BRCV.xlsx`. Každá proměnná se vypisuje do samostatného sloupce tabulky.

```

if (i < valBRCV.Count() / 2)
{
    g++;
    // 1. blok dat
    if (g < (valBRCV.Count() / 4))
    {
        lineLength++;
        // 2. sloupec tabulky
        ws.Cells[g + 1, 2].Value = val;
        ws.Cells[1, 2].Value = "BRCV1";
        ws.Cells[1, 2].Style.Fill.PatternType = ExcelFillStyle.Solid;
        ws.Cells[g + 1, 2].Style.Fill.BackgroundColor.SetColor(Color.Red);
        ws.Cells[1, 2].Style.Fill.PatternType = ExcelFillStyle.Solid;
        ws.Cells[g + 1, 2].Style.Fill.BackgroundColor.SetColor(Color.Blue);
        stringValue += string.Format("{0:D1} ", val);
    }

    // 2. blok dat
    if ((g >= (valBRCV.Count() / 4)) && (g < (valBRCV.Count() / 2)))
    {
        h++;
        lineLength10++;

        // 3. sloupec tabulky
        ws.Cells[h + 1, 3].Value = val;
        ws.Cells[1, 3].Value = "BRCV2";
        ws.Cells[1, 3].Style.Fill.PatternType = ExcelFillStyle.Solid;
        ws.Cells[h + 1, 3].Style.Fill.BackgroundColor.SetColor(Color.Red);
        ws.Cells[1, 3].Style.Fill.PatternType = ExcelFillStyle.Solid;
        ws.Cells[h + 1, 3].Style.Fill.BackgroundColor.SetColor(Color.Blue);
        stringValue10 += string.Format("{0:D1} ", val);
    }
}

```

Obr. 58. Výpis dat do tabulky

Současně s ukládáním dat do datového bloku DB300 se po stlačení tlačítka *Čtení datového bloku* u klientské aplikace zobrazí hodnoty obou proměnných v textových polích blokové služby txtReadBlock (BRCV1) a txtReadBlock2 (BRCV2).

```

// Nový řádek - OPC UA Client
if (lineLength > 9)
{
    stringValue += "\n";
    lineLength = 0;
    stringValue += "Conn1_BRCV1[";
    stringValue += i + 1;
    stringValue += "]-->";
}

if (lineLength10 > 9)
{
    stringValue10 += "\n";
    lineLength10 = 0;
    stringValue10 += "Conn1_BRCV2[";
    stringValue10 += i + 1;
    stringValue10 += "]-->";
}

}

// Tisk hodnoty - OPC UA Client
txtReadBlock.Text = stringValue;
txtReadBlock2.Text = stringValue10;

```

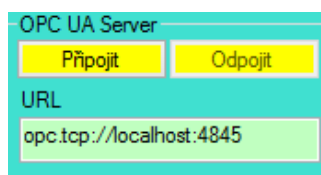
Obr. 59. Výpis hodnot do klienta
OPC UA Client

Náměry	BRCV1	BRCV2
1	47	94
2	23	31
3	100	69
4	77	11
5	55	54
6	31	97
7	6	38
8	83	77
9	59	14
10	34	53
11	11	91
12	88	47
13	62	34
14	39	21
15	14	8
16	91	97
17	67	85
18	41	73
19	18	60
20	95	47
21	69	35
22	45	22
23	22	8
24	97	98
25	72	86
26	48	74

Obr. 60. Uložená data

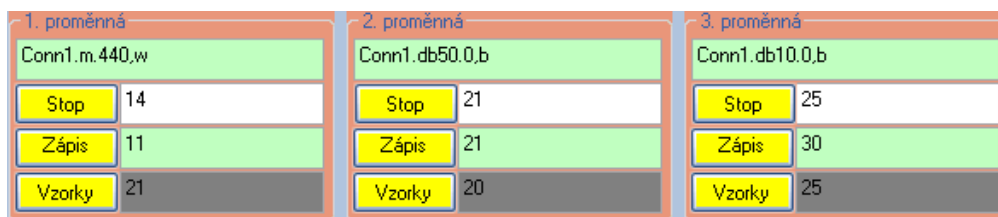
8 TESTOVÁNÍ VYTVOŘENÉ APLIKACE S VYHODNOCENÍM

Při vývoji klientské aplikace jsem kladl důraz na jednoduchost ovládání. Proto si v prvním kroku obsluha aplikace pouze vybere předvolenou lokální adresu využívající port 4845. V případě potřeby existuje možnost si zvolit jinou adresu URL.



Obr. 61. Předvolená lokální adresa

Dokud nedošlo k připojení serveru je aktivní pouze tlačítko *Připojit* a je možné zvolit adresy proměnných. Ostatní tlačítka nejsou aktivní.



Obr. 62. Volba proměnných

Po kliknutí na tlačítko *Připojit* a po úspěšném připojení serveru se aktivují všechny zbývající tlačítka. Funkci jednotlivých tlačítek bude představena na konkrétním příkladu.

8.1 Testování proměnných variabilní služby

Protože linka RollerHead je tvořena soustavou dopravníků a navíjecích stanic je nejčastějším problémem, který trápí obsluhu i oddělení Technologie udržení optimální regulační smyčky. Regulační smyčka je tvořena průvěsem mezi dvěma řízenými pohony. U dopravníků je to přechod mezi dopravníky, u navíječky přechod mezi dopravníkem a cívkou navíjecí stanice. Vlastnosti smyčky ovlivňují zejména tyto parametry:

- Poloha vahadla určuje rychlost následujícího pohonu a tím i velikost smyčky.
- Nastavení tlaku vahadla určuje sílu, která působí proti taženému profilu. Správné

nastavení tlaku zabraňuje protažení nebo naopak slepení profilu.

- U navíjecí stanice má na výslednou rychlost pohonu navíječky vliv průměr namotané cívky.

Při nedodržení některého z parametrů může dojít k protažení taženého pásu a způsobit zmetkovost výrobku.

```

Network 3: vypocet hodnoty OUT

A #PLUS // "H" : KOR = KOR
JCN M301

//;
L #INP // INT = 16 bit
ITD // INT na DINT
L #AUX_KOR // INT = 16 bit
ITD // INT na DINT
*D // DINT = 32 bit
L L#1000 // DINT = cislo 100,0%
/D // DINT = 32 bit
T #AUX_OUT // DINT = 32 bit : OUT = (INP x KOR)/1000

//;
JU M351

//;
M301: L 2000 // "L" : KOR = 2000 - KOR (doplnek)
L #AUX_KOR // napr: 200,0 - 120,0 = 80,0 %
-I
T #AUX_KOR

//;
L #INP // INT = 16 bit
ITD // INT na DINT
L #AUX_KOR // INT = 16 bit
ITD // INT na DINT
*D // DINT = 32 bit
L L#1000 // DINT = cislo 100,0%
/D // DINT = 32 bit
T #AUX_OUT // DINT = 32 bit : OUT = (INP x KOR)/1000

//;
M351: NOP 0

```

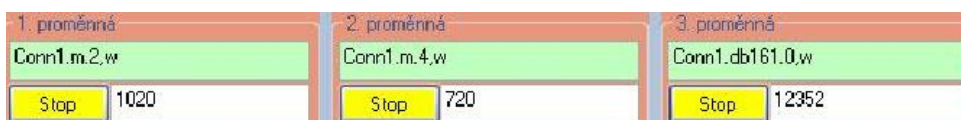
Obr. 63. Výpočet regulační smyčky

Pro otestování možností klientské aplikace byly zvoleny proměnné související s regulační smyčkou u navíječky Miag. Rychlost navíjení vychází ze vzorce: $\#AUX_OUT = (\#INP \times \#AUX_KOR) / L\#1000$, kde jednotlivé parametry představují tyto proměnné:

- $\#AUX_OUT$ – Požadovaná rychlost navíjení
- $\#INP$ – Rychlost předchozího dopravníku
- $\#AUX_KOR$ - Součin polohy vahadla a průměru cívky

Vlastní výpočet rychlosti navíjení dále pokračuje kalibrací pohonu. Protože aplikace nabízí možnost zvolit tři proměnné, budou zvoleny proměnné, které představují polohu vahadla, průměr cívky a rychlost navíjení před kalibrací.

Pro sledování datových změn, čtení a zápis mohou být indikovány tři proměnné. Sledování proměnné může být zapnuto přes tlačítko *Čtení*, kde každá proměnná má své vlastní tlačítko. Zaznamenané datové změny se zobrazí v textových polích vedle tlačítka.



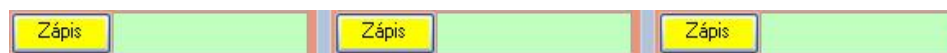
Obr. 64. Čtení proměnných

Protože zobrazované hodnoty se dynamicky mění, je možné pro porovnání hodnot využít tlačítka *Vzorky*. Stiskem tlačítka jsou okamžitě přečteny aktuální hodnoty tří proměnných a jsou zapsány do vedlejšího textového pole. Tímto způsobem jsou monitorovány hodnoty ve stejných časových okamžicích.



Obr. 65. Vzorkování proměnných

Pomocí tři tlačítek *Zápis* lze samostatně přepsat proměnné. Podmínkou pro zápis proměnné je, že před zápisem musí proběhnout čtení, protože konverze dat pro zápis se provede na základě naměřených hodnot.



Obr. 66. Zápis proměnných

Problémem na navíječce Miag je protahování taženého profilu od poloviny namotané cívky. Když si porovnáme hodnoty do poloviny namotané cívky, tak je vidět že, poloha vahadla se pohybuje okolo hodnoty 1000. Tato hodnota odpovídá optimálnímu stavu, ve kterém by se vahadlo mělo pohybovat v pracovním režimu. V tomto stavu průměr cívky odpovídá skutečnosti. Při překonání průměru odpovídající polovině cívky začíná docházet ke zkreslení průměru cívky oproti skutečnosti a zároveň ke změně polohy vahadla. Z toho vyplynulo, že problém primárně vzniká v odměřování průměru cívky. Po výměně ultrazvukového snímače pro snímání průměru byl problém vyřešen.

8.2 Testování proměnných blokové služby

Ke znázornění funkcionality blokové služby byla vytvořena v PLC programu testovací sekvence, která přiřazuje proměnné *#inpBlock1* sekvenci čísel v rozsahu 1 až 100 a proměnné *#inpBlock2* stejnou sekvenci posunutou o hodnotu 1.

```

L      "DB OPC Data".winCC.inpBlock1
L      L#100
>=D
JNB   x2

L      L#0
T      "DB OPC Data".winCC.inpBlock1
T      "DB OPC Data".winCC.inpBlock2
JU    x3

x2:   L      "DB OPC Data".winCC.inpBlock1
      L      1
      +I
      T      "DB OPC Data".winCC.inpBlock1
      L      1
      +I
      T      "DB OPC Data".winCC.inpBlock2
x3:   NOP   0

```

Obr. 67. Testovací sekvence BSEND

Bloková služba se spouští tlačítkem *Transfer dat* na obslužném panelu ve WinCC, které zajistí zápis hodnot do datového bloku v PLC. K zobrazení dat obou proměnných blokové služby v textových polích klientské aplikace je určeno tlačítko *Čtení datového bloku*.



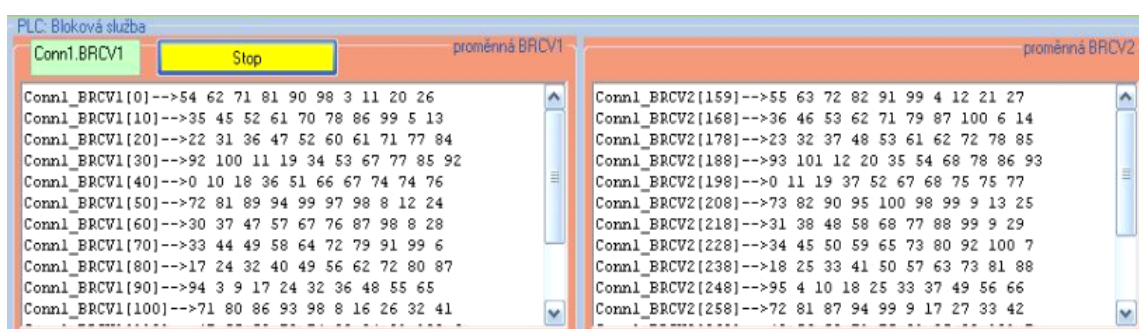
Obr. 68. Čtení datového bloku

Během zápisu hodnot do textového pole je tlačítko označeno textem *Stop*. Současně se zápisem do textového pole klientské aplikace se data zapisují i tabulkového výstupu.

	Address	Declaration	Name	Type	Initial	@Actual value
1	0.0	stat	conn1_BSEND[0]	INT	0	54
2	2.0	stat	conn1_BSEND[1]	INT	0	62
3	4.0	stat	conn1_BSEND[2]	INT	0	71
4	6.0	stat	conn1_BSEND[3]	INT	0	81
5	8.0	stat	conn1_BSEND[4]	INT	0	90
6	10.0	stat	conn1_BSEND[5]	INT	0	98
7	12.0	stat	conn1_BSEND[6]	INT	0	3
8	14.0	stat	conn1_BSEND[7]	INT	0	11
9	16.0	stat	conn1_BSEND[8]	INT	0	20
10	18.0	stat	conn1_BSEND[9]	INT	0	26
11	20.0	stat	conn1_BSEND[10]	INT	0	35
12	22.0	stat	conn1_BSEND[11]	INT	0	45
13	24.0	stat	conn1_BSEND[12]	INT	0	52
14	26.0	stat	conn1_BSEND[13]	INT	0	61
15	28.0	stat	conn1_BSEND[14]	INT	0	70
16	30.0	stat	conn1_BSEND[15]	INT	0	78
17	32.0	stat	conn1_BSEND[16]	INT	0	86
18	34.0	stat	conn1_BSEND[17]	INT	0	99
19	36.0	stat	conn1_BSEND[18]	INT	0	5
20	38.0	stat	conn1_BSEND[19]	INT	0	13
21	40.0	stat	conn1_BSEND[20]	INT	0	22
22	42.0	stat	conn1_BSEND[21]	INT	0	31

Obr. 69. Datový blok DB300

Sledovaná data se ukládají do datového bloku DB300 jako pole typu INT pod jménem *conn1_BSEND[.]*, kde pro každou proměnnou je inkrementován index uvnitř hranatých závorek. Data z datového bloku DB300 jsou viditelná v klientské aplikaci pod názvem *Conn1_BRCVx[.]* ($x=1, 2$). Při porovnání hodnot v datovém bloku s daty v klientské aplikaci nejsou vidět žádné rozdíly. Rozdíl není ani mezi daty *Conn1_BRCV1* a *Conn1_BRCV2* u klientské aplikace, kde je u všech hodnot vidět stejný posun hodnot odpovídající testovací frekvenci. Stejná data jsou i u vytvořené tabulky *Conn1_BRCV.xlsx*.



Obr. 70. Data z DB300 v klientské aplikaci

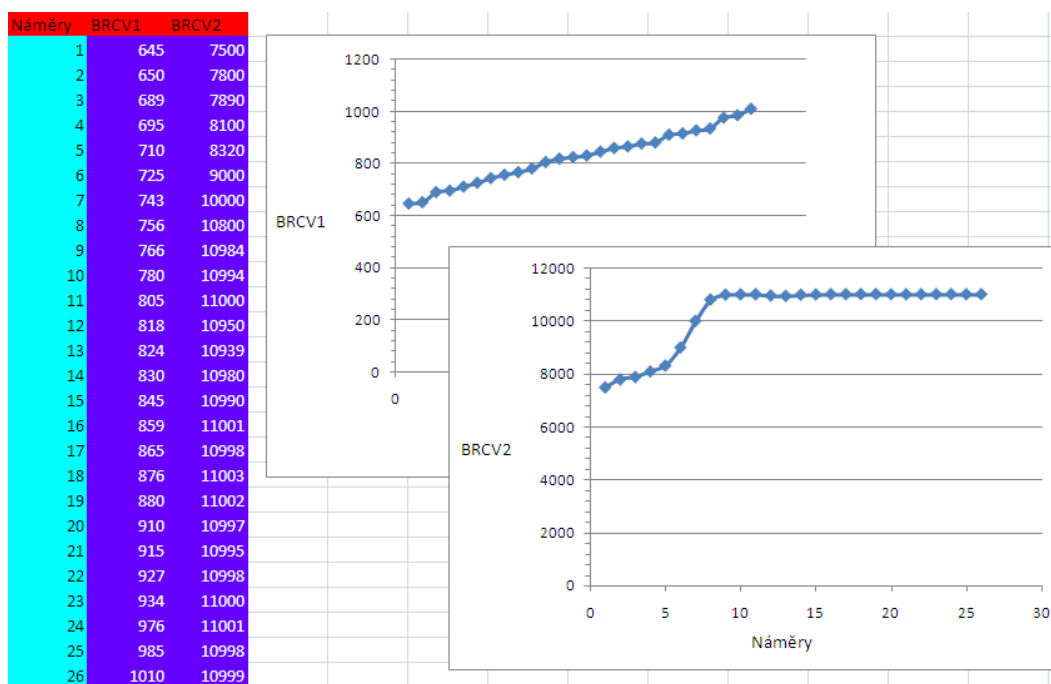
Když se porovnájí konkrétní data, tak například v datovém bloku DB300 na řádku označeném číslem 10 je proměnná *conn1_BSEND[9]* s hodnotou 26.

Náměry	BRCV1	BRCV2
1	54	55
2	62	63
3	71	72
4	81	82
5	90	91
6	98	99
7	3	4
8	11	12
9	20	21
10	26	27
11	35	36
12	45	46
13	52	53
14	61	62
15	70	71
16	78	79
17	86	87
18	99	100
19	5	6
20	13	14
21	22	23
22	31	32
23	36	37
24	47	48

Obr. 71. Data z DB
300 v tabulce

Stejná hodnota je v klientské aplikaci označena proměnnou Conn1_BRCV[9] a v tabulce na řádku 10 ve sloupci BRCV1. Na tomto příkladu je vidět snadná dohledatelnost stejných dat u různých typů výstupů.

Využití blokové služby je možné znázornit na problému s protažením taženého pásu mezi horním a spodním středícím dopravníkem v situacích, kdy docházelo k větším změnám požadované rychlosti linky. V těchto případech docházelo k diferencí mezi rychlostí spodního a horního středícího dopravníku. Protože rychlost spodního dopravníku závisí na poloze kompenzátoru, byla do proměnné *#inpBlock1* vložena poloha kompenzátoru. Do proměnné *#inpBlock2* byla vložena rychlost horního středícího dopravníku. Při správné funkci by měly hodnoty obou proměnných spolu korespondovat.



Obr. 72. Poloha kompenzátoru a rychlost horního středícího dopravníku

Jak ukazuje graf na předcházejícím obrázku, zaznamenaná data mají rozdílný bod dosažení optimálního stavu. Zatímco rychlost dopravníku byla na požadované hodnotě u 11 náměru, kompenzátor dosáhl tohoto stavu až u 26 náměru. Tento rozdíl způsoboval diferencí mezi rychlostmi obou sledovaných dopravníků. Problém vyřešen překalibrováním polohového snímače kompenzátoru.

8.3 Vyhodnocení projektu

Předmětem práce je sledování dat z PLC využitím OPC serveru s OPC klientem, kteří poskytují novou funkcionalitu. Jak je znázorněno v následující tabulce, OPC server a OPC klient jsou nové části podílející se na řízení linky. V tabulce je i stručný popis toho, co bylo dopracováno oproti původnímu stavu řízení.

Tab. 6. Přehled částí podílejících se na řízení linky

Původní části řízení	Současné části řízení	Popis provedených úprav
WinCC 6.2	WinCC 6.2	Vizualizační software – vytvořen obslužný panel pro ovládání klientské aplikace
SIMATIC NET 5.0	SIMATIC NET 7.0	Standard pro komunikační protokoly – instalace nové verze kvůli podpoře OPC UA
.NET Framework 3.0	.NET Framework 4.0	Běžové prostředí ke spuštění aplikace – aktualizováno na novější verzi
	OPC UA Server SIMATIC .NET	OPC Server – nakonfigurován OPC Server
	OPC UA Client	Klientská aplikace – vytvořen klient OPC UA Client s tabulkovým výstupem
PLC SIMATIC S7-400	PLC SIMATIC S7-400	Programovatelný automat – provedeny úpravy programu
CP 443-1	CP 443-1	Komunikační procesor – bez úprav

Pro sledování dat je použita variabilní i bloková služba. Práce prokázala význam obou typů služeb. Pro Využití variabilní služby je určeno primárně pro servisní údržbu strojů, která se jednoduše dostane ke stavovým informacím všech proměnných s možností tyto data vzorkovat nebo přepisovat. Před realizací klienta nebylo ve schopnostech údržbáře se k těmto informacím dostat. Je pravda, že některé proměnné jsou zobrazeny vizualizačním

prostředkem WinCC. Ale v případě, že proměnná, která je nutná pro vyřešení poruchy, není zobrazena, musí se údržbář připojit k řídicímu systému. V tomto místě většina údržbářů není schopna pokračovat. Díky vytvořené klientské aplikaci může údržbář přistupovat ke všem proměnným. Klientská aplikace simuluje tabulku proměnných (VAT tabulku) používanou programem SIMATIC Manager, do které se zapisují sledované proměnné. Hlavní výhodou tabulky je možnost sledovat vybrané proměnné z jednoho místa. Výhodou variabilní služby je navíc oproti blokové službě snadnost použití, protože se nemusí dělat žádné programové úpravy. Díky aplikování klienta byl vyřešen problém s protažením na navijedce Miag. Když se podíváme na potenciální řešení poruchy s využitím klientské aplikace, tak při porovnání s původním řešením je zde poznat zřejmý posun.

Naopak bloková služba nabízí širší možnosti využití i pro další oddělení pracující s provozními daty. Bloková služba je určena k monitorování velkého objemu dat. V tomto projektu je možné zvolit současné sledování dvou proměnných. Tímto způsobem je možné sledovat například rychlosti všech pohonů, polohy vahadel nebo dalších analogové snímače. Na rozdíl od použití variabilní služby je nutné u blokové služby přípravná fáze programátora, který specifikuje proměnnou úpravou programu v PLC nebo úpravou skriptu ve WinCC. Důležitost použití blokové služby se nejlépe vysvětlí na následujících příkladech:

- Tento typ služby bude využíván oddělením Průmyslového inženýrství, které pro své výpočty potřebuje znát celodenní průběh určitých proměnných na stroji. V současnosti data zapisuje brigádník, který tyto data odečítá přímo z operátorského panelu. Jedním z možných příkladů využití blokové služby je monitorování rychlostí linky sloužící k normování výkonu obsluhy stroje.
- Další možné využití je pro oddělení Technologie, pro které je jednou z nejdůležitějších parametrů teplota profilu za vytlačovací hlavou. Pokud dojde k překročení maximální dovolené teploty, začne se směs napalovat a dojde ke znehodnocení výrobku. K napálení může dojít u nekvalitní kaučukové směsi i při nižších teplotách. Problém je způsoben tím, že se tady střetávají zájmy oddělení Technologie, případně oddělení Kvality a zájmy obsluhy stroje. Pro Technologii i Kvalitu jsou důležité parametry výrobku, naopak pro obsluhu strojů množství

vyrobených výrobků a čas ke splnění plánu. U nové aplikace na základě zaznamenaných dat je možné zpětně zjistit, jestli došlo k napálení směsi z důvodu vysokých teplot nebo špatnou kvalitou směsi.

- Nejčastějším uživatelem bude samozřejmě servisní údržba strojů a zástupci Továrního inženýrství, kteří budou blokovou službu využívat pro monitoring poruchových stavů. Hlavní přínosem je možnost sledovat v delším časovém úseku proměnnou bez přítomnosti u stroje. Protože pomocí klientské aplikace je možné sledovat dvě proměnné blokové služby, nabízí se pro údržbáře nová funkcionality, která přináší nové možnosti využití. Zejména v případech, kdy je potřeba zjistit, která událost nastala před poruchou. Potom stačí zvolit proměnnou, která způsobila poruchový stav a přiřadit ji k blokové službě Conn1.BRCV1. Pro zjištění příčiny poruchového stavu je nejdůležitější specifikovat „správnou“ proměnnou a přiřadit ji k blokové službě Conn1.BRCV2. Tato proměnná se určuje na základě studia vytvořeného programu.

Správnou funkčnost blokové služby je také možné ověřit prostřednictvím porovnání hodnot u všech výstupů aplikace. Jak se ukázalo v předchozí kapitole, zobrazená data v tabulce byly shodné s daty v datovém bloku v PLC a také u klientské aplikace. Díky nasazení aplikace byl vyřešen problém s protažením tažené fólie mezi horním a spodním středícím dopravníkem.

ZÁVĚR

V rámci této diplomové práce je řešena problematika využití OPC serveru. Úkolem tohoto projektu bylo realizovat sledování PLC dat využitím OPC komunikace. Vytvořená práce rozšiřuje již existující způsob řízení o klientskou aplikaci s rozsáhlými možnostmi využití. Obsahem práce je uživatelský program zahrnující rozsáhlé použití softwarových prostředků. Je zde použitý vývojový software na programování PLC Step7, vizualizační prostředek WinCC, software k vývoji aplikací v prostředí Windows Visual Studio 2013, případně standard pro komunikační protokoly Siemens SIMATIC NET. Prvním úkolem bylo zpracovat literární rešerši na téma OPC komunikace. Ukázaly se výhody nové specifikace OPC UA oproti staršímu COM modelu. Dalším úkolem bylo realizovat OPC klienta. Do klienta byla zakomponovaná jak jednodušší variabilní služba, tak i bloková služba. Dalším dílčím úkolem bylo vytvoření popisu linky RollerHead. Pro správnou funkci byl upraven stávající PLC program s napojením na vizualizaci WinCC. Na závěr jsou na příkladech využití aplikace vyhodnoceny přínosy.

Zpracování této DP by mělo pokrývat vybrané situace vyskytující se ve výrobním závodě Mitas Agro Otrokovice, ve kterých nebyla údržba strojů schopná řešit problémy z důvodu nedostatečných informací týkající se výroby. Tyto informace nebylo možno získat, neboť ne všechny požadované parametry byly monitorovány. V takových případech byl nucen přijet zástupce Továrního inženýrství. Protože servisní podpora z Továrního inženýrství nebyla vždy k dispozici, docházelo občas k nemalým prostojům, které způsobovaly ekonomické problémy. Z tohoto důvodu nové řešení přináší výhody zejména pro servisní pracovníky, kterým tato aplikace usnadní práci. Klientská aplikace poskytuje funkcionalitu, která u původního způsobu řízení nebyla zastoupena. Pro výrobní úsek by klientská aplikace v některých případech mohla snížit ztrátovost výroby vzniklou z nuceného prostoje zkrácením doby zásahu údržby. Dalšími potenciálními uživateli by mělo být oddělení Průmyslového inženýrství nebo Technologie, případně Kvality. Pro tyto oddělení vzniká možnost monitorovat konkrétní data, na jejichž základě se dostanou k přesnějším datům a odhalí nedostatky v parametrech výrobku. Využití blokové i variabilní služby tímto přispívá ke zkvalitnění výrobního procesu.

Klientská aplikace je určena k monitorování PLC hodnot, které nám umožní sledovat data přímo na obrazovce operátorského panelu. Vedle sledování dat je nespornou výhodou

jejich uložení v tabulkovém formátu. Právě uložení dat ve formátu MS Excel nabízí další možnosti využití sledovaných dat. Protože PC stanice s vizualizací WinCC jsou ve výrobním závodě Mitas Agro Otrokovice rozšířeným standardem, nabízí se implementace na další výrobní linky a stroje. Protože na lince dochází k velkým statistickým výkyvům při vyčíslení ztrát je možné vyhodnocení přínosů zpracovat až v delším časovém horizontu.

ZÁVĚR V ANGLIČTINĚ

In this thesis is solved issue of the use OPC server. The implementation monitoring PLC data using OPC communications was task of this project. The created work extends the existing way of managing the client application with extensive use. The thesis includes user program involving extensive use of software tools. There is a development software used for programming PLC Step7, WinCC visualization tool, software for application development in Windows environment, Visual Studio 2013, or standard communication protocols for Siemens SIMATIC NET. The first task was to process a literature review on the topic of OPC communication. They were shown the advantages of the new OPC UA specification compared to the older COM model. The next task was to implement the OPC client. Into client was integrated both a simpler variable service, and block service. Another task was to create a description RollerHead line. For the correct function was modified existing PLC program with a connection to WinCC visualization. In conclusion, the benefits are evaluated on the examples of application use.

Processing of the DP should cover selected situations occurring in the factory Mitas Agro Otrokovice in which maintenance have not been able to solve problems due to insufficient of information regarding production. Such information could not possible to obtain because not all required parameters were monitored. In such cases had come from the representative Plant engineering. Because service support from Plant engineering was not always available, there sometimes to downtime, which caused economic problems. For this reason, a new solution brings benefits especially for service personnel, which this application will facilitate work. The client application provides functionality that the original method of control is not represented. For the production department would the client application in some cases, reduce the loss rate of production resulting from the forced downtime by reducing the time the maintenance action. Other potential users should be the department of Industrial engineering or Technology or Quality. The possibility to monitor specific data exist for these departments, on the basis of which they get more accurate data and reveal defects in the products. Use a block and variable service contributes to improving the production process.

The client application is designed to monitor PLC values that enabling us to monitor data directly on the screen of the operator panel. Besides monitoring data is an advantage to

store them in a spreadsheet format. Storing data in MS Excel format offers more ways of using monitored data. Because the PC station with WinCC visualization are factory Mitas Agro Otrokovice's standard, it is possible implementations on other production lines and machines. Because there are large statistical fluctuations in the quantification of losses is possible to process the evaluation of the benefits in the longer term.

SEZNAM POUŽITÉ LITERATURY

- [1] BERGER, H. *Automatizace se STEPem7 v AWL*. Munich : Publicis MCD Werbeagentur GmbH, 1998. 440 s. Volně programovatelné automaty SIMATIC S7-300/400. ISBN 3-89578-089-8.
- [2] WEIGMANN, J.; KILIAN, G. *Decentralization with Profibus DP/DPV1*. Erlangen : Publicis Kommunikations Agentur GmbH, GWA, 2003. 251 s. Structure, Configuration and Use of PROFIBUS DP with SIMATIC S7. ISBN 3-89578-218-1.
- [3] BLAŽEK, Jaroslav. *BLAJA Automation Portal, Průmyslová automatizace, projekty, odborné texty* [online]. 2014 [cit. 2014-01-10]. Dostupné z WWW: <<http://www.blaja.cz>>.
- [4] ŠÍMA, František; VILÍMEK, David. *Visual Studio .NET: praktické programování krok za krokem*. Grada Publishing a. s., 2006. 254 s. ISBN 978-8-024-71418-9.
- [5] SHARP, John. *Microsoft Visual C# 2020: krok za krokem*. 1.vydání. Brno: Computer Press a.s., 2010. 696 s. ISBN 978-8-025-13147-3.
- [6] MAHNKE, Wolfgang; LEITNER, Stefan Helmut; DAMM, Matthias. *OPC Unified Architecture*. Berlín: Springer, 2009. 362 s. ISBN 978-3-540-68898-3.
- [7] SIEMENS AG. *Communication SIMATIC NET Industrial Ethernet*. Karlsruhe, 2001. Interní materiál pro Course K0-7KETHER. Siemens .
- [8] BERGER, H. *Automating with SIMATIC: Controllers, Software, Programming, Data Communication, Operator Control and Process Monitoring*. Second Edition. Erlangen : Publicis Corporate Publishing, 2003. 221 s. ISBN 3-89578-223-8.
- [9] SIEMENS A.G. WinCC - Examples of integrated engineering with STEP7 [pdf elektronický manuál]. Nurnberg: Siemens, release 4/2009 [cit. 2014-1-10]. Dostupný z WWW: [<http://www.siemens.com>].
- [10] SIEMENS A.G. *SIMATIC NET: S7-300/400 – Industrial Ethernet/PROFINET Configuring and comissioning S7 CPs for Industrial Ethernet* [pdf elektronický

manuál]. Nurnberg: Siemens, release 09/2013 [cit. 2014-1-10]. Dostupný z WWW: [<http://www.siemens.com.text>].

- [11] OPC FOUNDATION. *The OPC Foundation - Unified Architecture* [online]. 2014 [cit. 2014-01-10]. Dostupný z WWW: [<http://www.opcfoundation.org>].

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

API	Application Programming Interface, rozhraní pro programování aplikací.
BRCV	Blok dat přijímaných od partnera.
BSEND	Blok dat odesílaných partnerovi.
COM	Component Object Model, implementuje nějaké rozhraní.
CP	Komunikační procesor.
DB	Datový blok.
DCS	Distributed Control System, distribuovaný způsob řízení.
DP	Decentrální periferie.
ERP	Enterprise Resource Planning, podnikový informační systém.
FB	Funkční blok.
FC	Funkce.
HMI	Human Machine Interface, rozhraní člověk - stroj.
HTTP	Hypertext Transfer Protocol, Internetový protokol.
HW	Hardware, technické vybavení počítače.
IEEE	Institut pro elektrotechnické a elektronické inženýrství.
LAD	Ladder Diagram, liniové schéma.
MES	Manufacturing Execution Systems, výrobní informační systém.
OB	Organizační blok.
OPC UA	OPC Unified Architecture, nová specifikace OPC
PC	Personal Computer, počítačová sestava.
PLC	Programmable Logic Controller, programovatelný automat.

RT	Runtime, běhové prostředí.
SCADA	Supervisory Control and Data Acquisition, dispečerské řízení a sběr dat.
SDK	Software Development Kit, sada nástrojů pro vývoj softwaru.
SFB	Systémový funkční blok.
SOA	Service Oriented Architecture, architektura orientovaná na službu.
STL	Statement list, programovací jazyk.
SW	Software, programové vybavení.
TCP	Transmission Control Protocol, transportní vrstva protokolu.
WinCC	Vizualizační SCADA software.
XML	Extensible Markup Language, značkovací jazyk.

SEZNAM OBRÁZKŮ

Obr. 1. Cílové aplikace OPC UA.....	12
Obr. 2. Přední část linky RollerHead	13
Obr. 3. Ořezávání za kalandrem	13
Obr. 4. Navíjecí stanice Miag	14
Obr. 5. Specifikace OPC UA	18
Obr. 6. Architektura OPC UA.....	19
Obr. 7. Komunikační partneři OPC	20
Obr. 8. Popis S7 komunikace.....	21
Obr. 9. Přístup OPC klienta k položkám OPC serveru.....	22
Obr. 10. Aplikační objekty a komunikační kanál	26
Obr. 11. Specifikace vrstvené architektury	27
Obr. 12. Struktura UA TCP	28
Obr. 13. Zpráva SOAP	30
Obr. 14. Použité hardwarové a softwarové komponenty	31
Obr. 15. Propojení hardwarových komponent	32
Obr. 16. CP 443-1	33
Obr. 17. OPC Scout	37
Obr. 18. Vložení PC stanice.....	41
Obr. 19. Konfigurace PC stanice	42
Obr. 20. Konfigurační nástroj NetPro.....	42
Obr. 21. Nové S7 spojení.....	43
Obr. 22. Tabulka spojení.....	43
Obr. 23. Vlastnosti S7 spojení	43
Obr. 24. Station Configuration Editor	44
Obr. 25. Configuration Console.....	45
Obr. 26. Výměna certifikátů mezi klientem a serverem	46
Obr. 27. Přidat nebo odebrat moduly snap-in	47

Obr. 28. Certifikáty	47
Obr. 29. Formát souboru pro export a uložení certifikátu	48
Obr. 30. Struktura programu	49
Obr. 31. Start komunikace s OPC a deklarace multifunkčního bloku	50
Obr. 32. S7 program pro čtení dat z WinCC	51
Obr. 33. Generátor pulsu.....	52
Obr. 34. Start komunikace s OPC a deklarace multifunkčního bloku	52
Obr. 35. Sekvence oboustranné komunikace	54
Obr. 36. SFB12 BSEND	55
Obr. 37. SFB13 BRCV	56
Obr. 38. Absolutní přístup	57
Obr. 39. Symbolický přístup	58
Obr. 40. Kód ve WinCC - otevření klientské aplikace	59
Obr. 41. Obslužný panel ve WinCC	59
Obr. 42. Interakce OPC klienta s OPC UA serverem	60
Obr. 43. Diagram tříd ClientAPI.....	61
Obr. 44. Metoda FindServers.....	62
Obr. 45. Metoda GetEndpoints	62
Obr. 46. Nalezení serveru	63
Obr. 47. Ukázka kódu metody Connect.....	64
Obr. 48. Metoda Disconnect	64
Obr. 49. Metoda Browse.....	65
Obr. 50. Navázání relace.....	66
Obr. 51. Metoda AddReadItem.....	67
Obr. 52. Uživatelské rozhraní klienta	68
Obr. 53. Ukázka metody btnConnect_Click	69
Obr. 54. Ukázka metody btnSample_Click	70
Obr. 55. Vytvoření tabulky	71

Obr. 56. Smyčka „for“ pro výpis hodnot	72
Obr. 57. Sloučení sousedních bajtů do proměnné val.....	72
Obr. 58. Výpis dat do tabulky	73
Obr. 59. Výpis hodnot do klienta OPC UA Client	73
Obr. 60. Uložená data	74
Obr. 61. Předvolená lokální adresa	75
Obr. 62. Volba proměnných.....	75
Obr. 63. Výpočet regulační smyčky.....	76
Obr. 64. Čtení proměnných.....	77
Obr. 65. Vzorkování proměnných.....	77
Obr. 66. Zápis proměnných.....	77
Obr. 67. Testovací sekvence BSEND	78
Obr. 68. Čtení datového bloku.....	78
Obr. 69. Datový blok DB300	78
Obr. 70. Data z DB300 v klientské aplikaci	79
Obr. 71. Data z DB 300 v tabulce	79
Obr. 72. Poloha kompenzátoru a rychlost horního středícího dopravníku	80

SEZNAM TABULEK

Tab. 1. Význam jednotlivých částí syntaxe variabilní služby.....	23
Tab. 2. Význam jednotlivých částí syntaxe blokové služby	24
Tab. 3. Softwarové komponenty	34
Tab. 4. Použité IP adresy	41
Tab. 5. Význam syntaxe S7	54
Tab. 6. Přehled částí podílejících se na řízení linky.....	81

SEZNAM PŘÍLOH

PŘÍLOHA P I: OBRAZOVKY VE WINCC.....	97
PŘÍLOHA P II: FUNKCE FC300 WRITE TO DB.....	98
PŘÍLOHA P III: FUNKCE FC302 PULSE GENERATOR.....	99

PŘÍLOHA P II: FUNKCE FC300 WRITE TO DB

Funkce pro zápis hodnot do datového bloku DB300.

FC300 : Zápis hodnot na požadované adresy v DB

Funkce zapiše hodnoty na požadované adresy v DB.
Požadovaná adresa je dána číslem počáteční adresy #startAdress a číslem indexu #offsetIndex (hodnota = index), který udává posun od počáteční adresy. Na nové adrese je zapsaná dynamicky měnící se hodnota #valueInput

Network 1: Odblokování funkce

Comment:

```

A    #enable                // Pokud je enable vykoněj blok
JNB  kon

OPN  #db_No                 // Otevři datový blok

```

Network 2: Číslo indexu počáteční adresy

Comment:

```

L    #startAdress
L    2
/D
T    #indexStartAdress

```

Network 3: Výpočet požadované adresy

Comment:

```

L    #offsetIndex
L    #indexStartAdress
+I
T    #indexSetAdress

L    #indexSetAdress
L    16
+I
T    #startBitAdress

```

Network 4: Zápis hodnoty na požadovanou adresu

Comment:

```

L    #valueInput
T    DBW [#startBitAdress]

```

PŘÍLOHA P III: FUNKCE FC302 PULSE GENERATOR

Funkce pro vytvoření pulsu volaného cyklickým přerušením.

FC302 : Pulse Generator

Vytvoření pulsu volaného cyklickým přerušením

Network 1: Vytvoření pulsu

Comment:

```

        AN    #start
        JNB   dl
        JC    ini
ini:     L     0
        T     #actNumbPuls           // Inicializace
        R     #firstPuls

dl:     A     #start
        JNB   kon

        L     #actNumbPuls           // Aktuální číslo vygenerovaného pulsu
        L     #setNumbPuls           // Počet požadovaných vygenerovaných pulsů
        >=I
        JC    ini                     // Pokud je act větší než set
                                           // Proveď inicializaci

        L     #actNumbPuls           // Aktuální číslo pulsu
        L     1                       // Přičti 1
        +I
        T     #actNumbPuls           // Aktuální číslo pulsu

        L     #actNumbPuls
        L     1
        ==I
        =     #firstPuls             // První vygenerovaný puls

kon:    NOP  0
```