Fakulta elektrotechnická
Katedra aplikované elektroniky a telekomunikací

Bakalářská práce

SCADA systém Control Web a zpracování dat

Autor: David Grill
Vedoucí: Ing. Jiří Basl Ph.D

Plzeň 2014

Abstrakt

Tato bakalářská práce je zaměřena na sběr dat ze systému Control web a možnosti jejich následného zpracování. Práce pojednává převážně o možnostech získání dat z procesů a přenosu ke zpracování do databázových systémů. Práce obsahuje popis možností zpracování dat v databázích, stejně tak i popis pravidel pro správný návrh data-bází. Dále jsou zde popsány možnosti využití Control Wedu k archivaci dat. V praktické části práce se věnuji postupu tvorby aplikace pro PLC Amit, její vizualizaci, nastavení komunikace s PC a následnému propojení s SQL databází a demonstraci přístupu k uloženým datům přes internet.

Klíčová slova

SQL, ODBC, DBNet, NDB, MDB, RS-232, RS-485,

Na Fakultě elektrotechnické Západočeské univerzity v Plni.

Prohlašuji, že jsem svou závěrečnou práci vypracoval samostatně, pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce. Jako autor uvedené bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této závěrečné práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení § 270 trestního zákona č. 40/2009 Sb.
Také prohlašuji, že veškerý software, použitý při řešení této bakalářské práce, je legální.

V Plzni dne 6. června 2014

David Grill

.....
podpis

Obsah

1 Úvod

Obsahem mé bakalářské práce je popis postupu, principů a možností, jakými je možné získat data ze sledovaného procesu a zpracovat je a poskytnout dále ke zpracování. Konkrétně se hodlám zabývat získání dat pomocí OPC serveru a jejich další distribucí ostatním aplikacím a stanicím. Ke komunikaci systému ControlWeb a hardwarových prvků je možné použít poměrně rozsáhlé množství ovladačů, serverů a dalších softwarových prvků, z nichž každý má své výhody i nevýhody. Ve své práci budu porovnávat ovladač DBNet32 a DBNet/Ip OPC server od firmy Merz.

2 Proč kombinovat vizualizační systémy a databáze

V procesech využívajících vizualizačních systémů je možné získat poměrně velké množství různorodých dat. Je dobré vědět, jak ze sledovaných procesů získat potřebná data, to je ale jen jedna část úkolu. Důležité je i vědět, jak tato data správně zpracovat a poskytnout uživateli. K tomuto účelu Control Web využívá standard SQL. Díky kombinaci tohoto standardu a rozhraní ODBC můžeme pracovat s libovolným datovým formátem přístupným přes rozhraní ODBC a využít dotazů a funkcí poskytnutých SQL pro efektivní záznam a zpracování požadovaných dat. Pro představu zde popíši pár příkladů, jakými se tato možnost využívá v praxi. Např. ve žďárském podniku ŽĎAS a.s., stejně jako v mnoha dalších, používají tuto možnost k regulaci pecí a zároveň k údržbě zakázek. Zde k regulaci sběru dat využívají regulátor Watlow F4 s implementovaným protokolem Modbus, tento regulátor je nainstalovaný k pecím. Modbus zde pomocí RS-465 zpřístupňuje jednotlivé paměťové registry. Aplikační program tak může sbírat údaje o průběhu zpracování zakázky v peci, sbírat data z regulačního cyklu a zapisovat další program do regulátoru pro zpracování další zakázky. Tento zápis probíhá sekvenčním zápisem do registru zařízení. Nahrávání programů do regulátoru ale není úplně běžná věc, ke které by měly být vizualizační programy využívány. Většina současných systémů tuto možnost běžně neumožňuje.

3 Jemný úvod do databázových systémů

3.1 Seznámení s databázemi

Databáze je soustava navzájem souvisejících objektů spravovaných jako jeden celek. Každá společnost zabývající se vývojem databází je definuje odlišně, z tohoto důvodu je tato definice tak obecná. Objekt je datová struktura uložená v databázi. Jedná se například o tabulky, indexy a různé pohledy zobrazující data. Databáze je složena ze tří základních vrstev:

1. **Báze dat (DB)**
   V DB jsou uložena data, která zpracováváme a používáme je k požadovaným výstupům. Tato data jsou uložena v předem definované struktuře (viz pravidla pro návrh DB). Data jsou zde v podstatě pouze uložena a DB samotná s nimi nevykonává žádné další operace.

2. **Systém řízení báze dat (SŘDB)**
   SŘDB je programová vrstva, která řeší následné operace s daty uloženými

v DB. Základem pro jeho tvorbu je tzv. relační model (E-R model), který je popsán níže. Řídicí systém je mezičlánek mezi samotnými daty a uživatelem. Díky němu samotný koncový uživatel nemusí vědět vůbec nic o samotné struktuře databáze ani o tom, jak jsou data ukládána. SŘDB provádí operace jako např:

- Přesouvání dat mezi fyzickými datovými soubory
- Řízení přístupu k datům více uživateli zároveň
- Řízení transakcí tak, aby byly správně provedeny všechny změny
- Zálohování a obnova databáze
- Zamezení neoprávněné změně a přístupu k datům

Data lze zpracovávat dvěma způsoby

1. **Zpracování dat v dávkách** Při tomto typu se zpracování účastní velké množství dat a čas potřebný pro zpracování a přípravu výstupních sestav může být poměrně dlouhý. Z toho důvodu je tento typ vhodný spíše pro tvorbu přehledových sestav, mzdových podkladů nebo importů a exportů velkého množství dat.
2. **Konverzační způsob** Konverzační způsob je na rozdíl od dávkového vhodný pro aplikace vyžadující rychlou zpětnou vazbu, nebo v místech, kde je nutné co nejrychleji data editovat a poskytnou dalším uživatelům. Příkladem může být např. rezervační systémy v hotelových službách nebo služby v internetovém bankovnictví. [6]

3.2 Datové modely

Informační systém jako databáze má poskytovat informace o reálné situaci. To je jedním z důvodů, proč si při návrhu databáze resp. jejího modelu musíme vybrat *objekty*, které chceme použít a udržovat.

- Entita– je chápána jako libovolná věc, kterou chceme popisovat
- Hodnota– charakteristický popis pro danou entitu
- Atribut– je chápán jako vlastnost popisované entity, popisují vztahy entit a hodnot

3.3 Relační databáze

Jde o velmi flexibilní druh databáze, a to z toho důvodu, že její struktura vychází z relačního modelu. V něm jsou data reprezentována ve dvourozměrných tabulkách, ze kterých následně umožňuje vybírat, kombinovat a spojovat data to různých pohledů. Relace zde představuje provázanost (souvislost) mezi jednotlivými tabulkami. Tabulky totiž mohou existovat samostatně, ale tím nevyužijeme kouzlo relačních databází. To tkví v možnosti ukládání souvisejících dat a jejich spravování na základě jejich provázanosti. Při vytváření různých pohledů na data pak můžeme tyto provázanosti využít a vybrat pouze data, která pro daný pohled potřebujeme a uzpůsobit ho tak co nejvíce potřebám konkrétního uživatele nebo aplikace. Zde jsou základní typy relací:

1. Relace 1:1
   při při tomto vztahu patří k jednomu prvku z množiny A právě jen prvek z množiny B. *Příklad:* Uvažujme, jaký vztah platí např. mezi občanem a občanským průkazem. Každý občan může mít pouze jeden občanský průkaz, stejně tak ale jeden průkaz může patřit pouze jednomu občanovi.

2. Relace 1:N
   Při tomto typu relace patří jednomu prvku z množiny A několik prvků z množiny B. *Příklad:* Uvažujme teď vypůjčování knih v knihovně. Tady si může jeden člověk vypůjčit najednou několik knih, ale v jednu chvíli může být jedna kniha vypůjčena pouze jedním člověkem.

3. Relace M:N
   V tomto případě jde o nejobecnější relaci, která není nijak omezena.

Relační databáze podporují pouze relace 1:N. Minimální kardinalita tím vlastně říká, jestli je účast na relaci povinná nebo volitelná, kde na straně 1 jsou relace povinné a na straně N jsou relace volitelné. Není moc obvyklé mít povinné relace na straně N. To v podstatě znamená, že nadřízená tabulka musí mít minimálně jednu tabulku podřízenou. Pro tvorbu relace je nutné mít vytvořen nějaký odpovídající sloupec jako jednoznačnou identifikaci pro každý řádek. Takový sloupec se nazývá *primární klíč*. Primární klíč může být vždy jen jeden, ale může být tvořen více sloupci. Tyto klíče tvoří základ relací, protože umožňují spojovat data z více tabulek dle potřeby. Pokud použijeme primární klíč v jiné tabulce, tak ho označujeme jako *cizí klíč*.

3.4 Používaná omezení

Jedná se o pravidlo platící pro příslušný objekt, které nějakým způsobem definuje jeho hodnoty nebo datové typy. Tato omezení lze nastavit při tvorbě struktury objektů, a jakmile je jednou vytvořeno, bude jej SŘDB neustále vyžadovat a neumožní nám toto pravidlo obejít. Těchto omezení existuje několik:

- NOT NULL– Toto omezení znemožňuje použít hodnotu *null*. Touto hodnotou SŘDB označuje, že hodnota v buňce mu není známa. Jedná se o speciální hodnotu, která se nechová ani jako prázdná buňka, chybějící údaj nebo nula.

- PRIMARY KEY– toto omezení zajišťuje jedinečnost primárního klíče v tabulce. Pokud je jako primární klíč definováno více sloupců, pak musí být jedinečné jejich kombinace, tzn., že jeden sloupec, který je součástí širšího primárního klíče, může obsahovat duplicitní hodnoty. Primární klíče jsou v tabulce označeny indexy. Díky nim lze klíče rychleji prohledat, a SŘDB tak může zjistit, zda se nově zadávaná hodnota nachází, nebo nenachází z již zadaných. PK je důležitou součástí tabulky, z toho důvodu nesmí být mít nikdy nulovou hodnotu.

- Jedinečnost– Toto omezení se definuje pouze v tabulkách, kde vyžadujeme neopakující se hodnoty, na rozdíl od PK zde můžeme ale použít více požadavků na omezení. I zde jsou hodnoty kontrolovány pomocí indexů.

- CHECK– Zde dochází ke kontrole vkládaného údaje, který podle výsledku dotazu bude buďto přijat, nebo odmítnut s odpovídající chybovou zprávou. Toto omezení ověřuje hodnotu ve sloupci pomocí příkazu, jehož návratovými hodnotami jsou logické TRUE a FALSE.

- Referenční integritní omezení– Při tomto omezení SŘDB kontroluje, zda jsou veškeré zadané primární klíče použity v tabulkách propojených v relaci. Nedovolí nám například uložit údaj s novým primárním klíčem, pokud tento klíč není propojen s jiným údajem ve druhé tabulce, nebo pokud se nejedná o klíč správného typu. Stejně tak nám nedovolí odmazat klíč, který je používán jako cizí klíč v jiné tabulce propojené relací.

3.5 Pohledy

Pohled je jednou z forem databázových dotazů, umožňující uživateli zobrazit data v potřebné podobě. Jedná se o uložený dotaz, který uživateli zobrazuje

virtuální tabulku s podmnožinou dat z jedné, ale i několika tabulek. Pohled se i vlastnostmi podobá normální databázové tabulce. Neslouží však k ukládání a změně dat, pouze data zobrazuje. Zde jsou vypsány některé užitečné funkce pohledů:

- Skrývá sloupce a řádky, které uživatel není oprávněn vidět, nebo jsou nepotřebné, a tím zlepšuje přehlednost
- Zvyšuje výkon dotazu
- Skrývá komplexní databázové operace

3.6 SQL

SQL (Sequel Query Language) je jazyk navržený a nejčastěji používaný pro komunikaci s relačními databázemi. O jeho vývoj se ve značné míře zasloužila společnost Oracle a mnohé další. Oracle usilovala o standard akceptovatelný Tento jazyk funguje na základě dotazů. Uživatel vytvoří dotaz (požadavek), který odešle databázi, ta dotaz zpracuje a uživateli vrátí požadovaný výstup. Nutno dodat, že samotná databáze žádné operace neprovádí, o to vše se stará SŘDB. Pro běžnou tvorbu dotazů nepotřebujete znát ani vnitřní procesy funkcí a ani nemusíte nic počítat, jak tomu je u programovacích jazyků, jako je např. C. Jednoduše zapíšete příkaz a SŘDB se o vše postará. Pokud například chcete zjistit počet řádků, průměr apod., stačí tedy jen použít odpovídající příkaz jako např. *SUM* nebo *AVG*. Nemusíme tedy počítači určovat postup jak daný výsledek získat, pouze mu sdělíme, jaký výsledek požadujeme. Proto SQL není vhodné pro tvorbu samostatných programů, ale jen pro správu dat. Je ale možné SQL zkombinovat s jiným jazykem, jako je např. Java, PHP a jiné.

4 Na co je potřeba dbát při návrhu DB

Při tvorbě struktury DB musíme dbát na fakt, že jde většinou o velké množství dat. Proto by měla být strukturována podle určitých pravidel, které zajišťují jejich správnost ve všech aplikačních částech, stejně jako správnou manipulaci s nimi (výběr dat, jejich změny apod.). Protože je databáze tvořena z množin, platí pro i stejná pravidla jako pro práci s množinami. Jedním z těchto pravidel je dodržení **Normalizované formy** Jedná se o soubor pěti provázaných pravidel, která určují, jak správně organizovat data:

4.1 Normalizované formy

1.Normalizovaná forma
Každý atribut by měl obsahovat jen atomické údaje tzn. údaje již dále nedělitelné
Tím je míněno, že pokud je možné údaj v tabulce rozdělit na podrobnější údaje, mělo by se tak stát. Například pokud budeme mít tabulku zaměstnanců s jejich adresou. Měla by vypadat takto

Příklad:

Město	Ulice	ČP
Cheb	Slavice	1
Most	Hradební	12
Cheb	Jungmanova	8

a ne takto

Adresa
Cheb Slavice 1
Most Hradební 12
Cheb Jungmanova 8

První varianta umožňuje lepší výběr údajů a také vybrat pracovníky, kteří bydlí např. pouze v Chebu.

2.Normalizovaná forma
Všechny objekty, které obsahují sloupce s duplicitními údaji, mezi sebou vytváří částečné závislosti. Ty je nutné upravit tak, aby v nich byl každý záznam uložen pouze jednou.

Příklad:
Každý záznam v objektu Zákazníci obsahuje informace o objednávce. Jestliže si zákazníci budou objednávat pouze jeden druh zboží, bude vše v pořádku. Jestliže si ale objednají více druhů zboží, bude se muset pro každý druh objednaného zboží vytvořit nový záznam v objektu Zákaznici. Lepší by bylo vytvořit nový objekt Objednávky a mezi objekty Produkty a Objednávky vytvořit relaci M:N. K tomu ale potřebujeme vytvořit nový objekt RozpisObjednavek.

3.Normalizovaná forma
Tato forma je splněna, pokud jsou splněny i normy 1 a 2. Dá se tedy říct, že je splněna, pokud je splněna i 2.NF a všechny atributy nepatřící do klíčových jsou nezávislé. *Příklad:*

Příklad: Každý záznam v objektu Zákaznici obsahuje informace o objednávce. Jestliže si zákazníci budou objednávat pouze jeden druh zboží, bude vše v pořádku. Jestliže si ale objednají více druhů zboží, bude se muset pro každý druh objednaného zboží vytvořit nový záznam v objektu Zákaznici. Lepší by bylo vytvořit nový objekt Objednávky a mezi objekty Produkty a Objednávky vytvořit relaci M:N. K tomu ale potřebujeme vytvořit nový objekt RozpisObjednavek.

5 Obecné podporované standardy

5.1 ODBC

ODBC je zkratka, která vznikla z názvu Open Database Contectivity. ODBC je psáno v jazyku C. Umožňuje přístup k datům nezávisle na systému pro řízení báze dat (SŘDB) v relačních i non-relačních databázích. Problém není ani v tom, že každý SŘDB používá pro data jiný formát. Toto rozhraní je dílem společnosti Microsoft a značně ulehčilo práci jak programátorům, tak koncovým uživatelům. Jednou z jeho výhod je nezávislost na platformě. ODBC je používáno jak na WIn32, tak i na platformách UNIX, MacOS,

OS/2 a dalších. Ovladač se konfiguruje pomocí *.ini souboru. Pokud chcete ODBC používat, pak budete potřebovat následující komponenty:

- ODBC klient
- ODBC ovladač

Pokud komunikujeme s databází za pomoci ODBC a chceme poslat SQL příkaz jako součást ODBC funkce, pak tento příkaz musí být převeden a upraven pro specifický ŠŘDB. Syntaxe SQL příkazů je založena na specifikaci X/Open a SQL Access Group z roku 1992. Pro maximální interoperabilitu ODBC aplikací je doporučováno používat syntaxi definovanou jako Appendix C, která obsahuje obě výše jmenované. [1]

5.1.1 Komponenty

ODBC se skládá z více komponent:

- Aplikace – zprostředkovává zpracování a volání funkce a obdrží výsledek
- Driver Manager – obsluhuje ovladač
- Driver – zajišťuje správnou komunikaci a překlad instrukcí od aplikace ke zdroji dat a zpět
- Zdroj dat – uchovává data přístupná uživateli a je závislý na operačním systému a ŠŘDB

Každá z těchto komponent má specifickou úlohu, díky které funguje ODBC jako celek. Např. aplikace inicializuje spojení se zdrojem dat, odesílá požadavky pro zdroj dat, zobrazuje požadované výstupy popřípadě chybová hlášení, Driver Manager vybírá správný zdroj dat a zavádí pro něj odpovídající ovladač, provádí validaci parametrů apod. V případě driveru se jedná o knihovnu zavedenou driver managerem. Tato knihovna se spojí se zdrojem dat a odešle do něj požadovaný příkaz. Pokud je zapotřebí, pak také převádí data z jiných formátu a výsledek odešle do aplikace. Zdroj dat je v podstatě kombinace SŘDB a operačního systému. Může tedy jít v jednom případě o Oracle běžící na systému Solaris a v jiném o Tandem NonStop SQL DBMS na systému Guardian 90. Jedná se tedy o ovladač, jehož úkolem je překládat příkazy z databázové aplikace na příkazy srozumitelné pro SŘDB. Aplikace tedy musí být schopna vytvořit příkaz srozumitelný pro ODBC a Systém pro řízení databáze ho musí být schopen na ně odpovídat. Jde tedy o dělení na systém serverový a klientský. Klient a server se mohou nacházet na jedné síti, nebo mohou být propojeny přes gateway a mohou být každý umístěn na jiné síti.

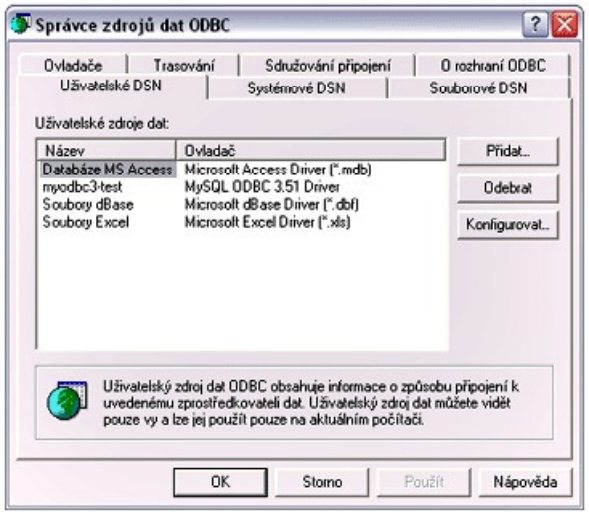
5.1.2 Nastavení komunikace přeš ODBC

ODBC je možné nakonfigurovat pro komunikaci se širokou řadou databázových platforem (MySQL, PostgreSQL, FireBird, MS SQL, MS Access, Oracle). Výhodou ODBC je jeho možnost využívat pro komunikaci s různými typy databází stejnou sadu příkazů. Prvním krokem pro funkční nastavení je použití správného ovladače. Já zde popíši nastavení pro MySQL a Windows XP a Windows 7. Ovladač pro Windows 7 a MySQL se nazývá MySQL Connector/OBDC. Cestu k MySQL databázi popisuje datový zdroj. Cesta k datovému zdroji se uloží do registru. Při pokusu o připojení k datovému zdroji se pak informace o spojení hledají v registrech. Každý zápis v registru musí obsahovat informace o datovém zdroji a k němu asociovaném ovladači. Pro správné vytvoření záznamů je potřeba ODBC správně nastavit. Postup k jeho nastavení ve Windows XP je následující:

1. otevřeme správce zdrojů dat ODBC, kterého nalezneme v ovládacích panelech
2. vybereme možnost Přidat a vybereme ze seznamu nainstalovaných ovladačů
3. zapíšeme název spojení
4. nastavíme požadované parametry spojení
5. Prvním parametrem je Host/Server Name nebo IP adresa. Zde je potřeba zadat buď platné DNS jméno, nebo IP adresu počítače (serveru), na kterém máme nainstalovanou platformu/databází, k níž se chceme přes ODBC připojovat. Implicitní nastavení je localhost
6. užitečnou volbou je položka SQL Command, ve které můžeme vložit SQL příkaz, jenž se spustí ihned po připojení
7. pokud máme vše nastaveno, pak zbývá jen vyzkoušet funkčnost. Použijeme tedy volbu test a v případě nefunkčního spojení nám vyskočí chybová hláška s popisem chyby

5.2 HTTP

Tento ovladač slouží ke komunikaci s jakýmkoliv HTTP serverem. Control Web pod ním umí pracovat jak v podobě klienta, tak v podobě serveru. HTTP ovladač funguje tak, že se připojí k určené URL stránce a pomocí řetězců do aplikace vrací její obsah. Ke komunikaci používá metody Get a Post. Stav komunikace je sledován ovladačem ve formě textového řetězce. Ty



Obrázek 1: správce zdrojů pro ODBC

se řadí do fronty a aplikace si je postupně zpracovává. Pro nastavení ovladače se používá *.ini soubor jehož parametry jsou rozděleny do několika sekcí.

5.2.1 Výjimky ovladače

Pokud při komunikaci ovladače vznikají výjimky, které ovladač může předat do aplikace, pak tato aplikace musí obsahovat objekt, který vzniklé výjimky aktivují. V objektu (virtuálním přístroji) musí být definovány parametry kde bude definován i název ovladače. Pokud dojde k výjimce, pak aktivuje virtuální přístroj. Při jeho činnosti je nutné zavolat i proceduru ovladače zvanou EnableExcetion, která povolí další výjimky.

- 0 stNone – žádná výjimka od ovladače
- 1 stConnect – signalizace spojení s URL
- 2 stGetComplete – signalizace dokončení operace Get
- 3 stPostComplete – signalizace dokončení operace Post

5.2.2 HTTP ve funkci serveru

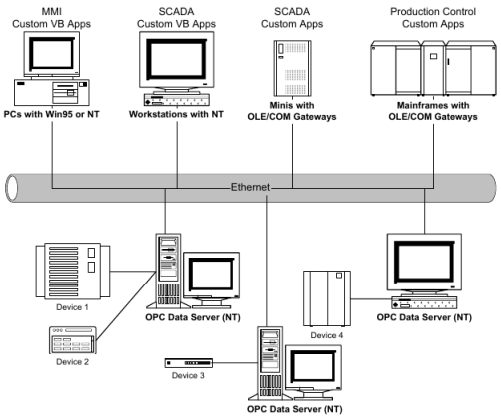
Oddělit od sebe funkci serveru a klienta není tak úplně jednoduché. To z toho důvodu, že jeden bez druhého prostě dost dobře nefungují. Doba, kdy pojem aplikace představoval většinou jeden spustitelný soubor nebo script, je navíc za námi. Dnes si každý, kdo má alespoň trošku ponětí o tvorbě

aplikací pod tímto pojmem představí více scriptů a souborů. Tyto soubory jsou provázané a jeden z druhého čerpají informace potřebné pro jejich běh. Teď k úkolu jaký v našem případě HTTP plní. Scripty a různé aplikace fungují jak na straně serveru, tak na straně klienta. Místo, odkud tyto aplikace čerpají a zároveň ukládají informace, je nějaký databázový stroj. HTTP server zde slouží jako zprostředkovatel těchto informací. Díky tomu je možné pak za pomoci nějaké WWW aplikace poskytovat informace o stavu různých procesů nebo např. telefonních seznamů, stavů objednávek. Dokonce jsme tímto schopni zpřístupnit i možnost ovládání a řízení strojů a procesů. Dále je tímto způsobem možné udržovat aktuální informace na různých klientech, kteří si jednoduše pomocí HTTP zažádají u databázového stroje o aktuální informace. Tento princip je funkční, a dnes je i tak používán jak na rozsáhle internetu, tak na intranetu. Dokonce i některé aplikace pracující na lokálním počítači využívají pro svou funkci řadu HTML stránek a tím pádem tedy i tento princip.

5.3 Přístroj HTTPD

Jedním z bodů této práce je zpřístupnit řízení a zobrazení dat pomocí webových stránek. K tomuto účelu je v prostředí Control Web vyžíván přístroj HTTPD. Příjemné na tomto virtuálním přístroji je fakt, že pro zobrazení požadovaných dat můžeme bez obav využít libovolný prohlížeč. Neměla, a u testovaného prohlížeče Mozilla Firefox 28.0 tak také nebyla, by být vyžadována ani žádná dodatečná instalace pluginu. Pro komunikaci používá právě zmiňovaný styl klient-server. Pokud tedy chceme zjistit nějakou informaci z procesu pomocí webového prohlížeče, pak server obdrží požadavek HTTP Request, a pokud vše proběhne správně, pak odpoví HTTP Response. My poté obdržíme nejčastěji URL adresu obsahující mimo jiné adresu serveru a cestu k požadované informaci na něm. Informace, kterou obdržíme, je ve většině případů prezentována nějakým souborem, na ten nás server odkáže a webový prohlížeč si jej poté načte. Tento soubor může být statický nebo dynamický. Výhodou dynamického souboru je, že nemusí v daný okamžik ani existovat, a server ho vygeneruje až podle našich požadavků. Oproti tomu statický soubor je již vytvořen a my s jeho formou a obsahem již nemáme možnost cokoliv udělat. Tento přístroj nám umožňuje i pro někoho užitečnější věci jako:

- vrátí klientovi obrázek zobrazující okamžitý stav přístroje
- aktivaci jiných přístrojů
- řídit proces pomocí html formulářů, kdy se námi zadaná data promítnou rovnou do řízeného procesu



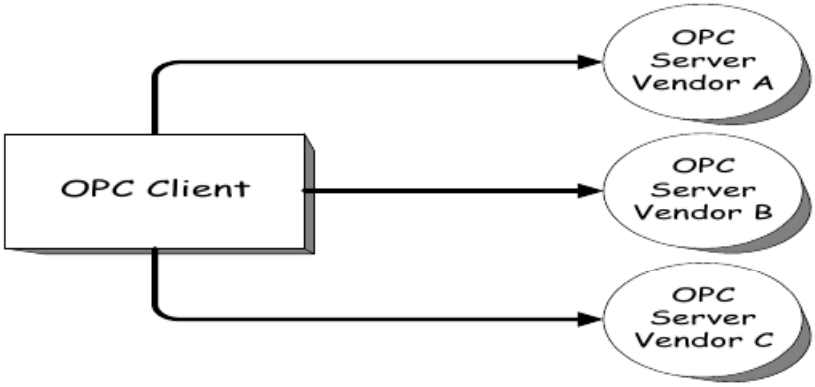
Obrázek 2: schéma použití OPC

- volání jiných akcí přístroje a procedury

[**?**]

5.4 OPC

Jedná se o zkratku OLE for Process Control. Tento standard je vyvíjen neziskovou organizací OPC Foundation, a je tedy dostupný bez jakýchkoliv licenčních poplatků. Jeho účelem je standardizovat rozhraní mezi prvky průmyslové automatizace, tedy řídícím systémem a průmyslovými informačními systémy na jedné straně a mezi průmyslovými automaty a akčními členy na straně druhé. Hlavním důvodem, proč vlastně OPC vzniklo, byla skutečnost, že každá aplikace a každé hardwarové zařízení potřebuje specifický ovladač, který je většinou vyvíjen právě jeho výrobcem. To vede k problémům, kdy například dochází k neshodám mezi ovladači. Změnou některého HW komponentu dochází k nefunkčnosti celku a nutnosti veškeré ovladače přeinstalovat. Zde se OPC využívá k vytvoření společného komunikačního spojení. Problém je, že výrobci nepodporují stejné funkce. Další problém spočívá v zajištění přístupu k HW. Dvě různé aplikace totiž nemohou zároveň používat jedno zařízení, pokud každá z nich nepoužívá jiný ovladač. Nemalou roli zde hraje i konkurenční boj, kdy zájmy výrobců brání ve vývoji nějakého univerzálního ovladače rozdílnost klientských protokolů. Proto vzniklo OPC, jehož úkolem je získávat data z datových zdrojů a zprostředkovávat je libovolné klientské aplikaci bez závislosti na výrobci hardware. Komunikace zde probíhá stylem klient-server a je založena na technologii OLE/COM vyvíjené společností Microsoft. Tento standard je specifikován v technické dokumentaci, která popisuje veškeré jeho vlastnosti a možnosti. Každá z těchto specifikací po-



Obrázek 3: architektura OPC klienta

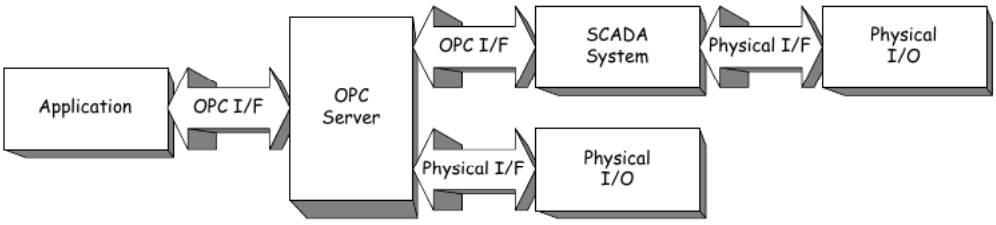
pisuje jinou problematiku jako např. způsoby komunikace a zpracování dat. [8]

5.4.1 Architektura

Jak jsem již uvedl výše, komunikace zde probíhá stylem klient/server. Nabízí se nám tedy možnost, kdy se k jednomu klientovi můžeme připojit pomocí OPC serverů různých výrobců a stejně tak se k jednomu serveru můžeme připojit pomocí klientů vyvíjenými více výrobci. Celá architektura se pak skládá z alespoň tří vrstev:

- Field management
- Process management
- Business management

Na nejnižší, tedy **Field management** úrovni, je prezentována jako typicky klientské stanice připojené na řízený proces, tzn. mají přímou vazbu s řídícími prvky, jako jsou PLC a jiné terminály. Tyto klientské stanice pak vstupují do LAN. Může se zde také nacházet komunikační a datový server určený pro uchovávání dat.
Process management úroveň představuje Human machine interface. Jedná se o připojené klientské počítače obsahující instalaci monitorovacích a vizualizačních aplikací, kdy jsou na těchto aplikacích prezentovány sledované technologické procesy.
Business managementstruktura parametrického souboru je již ta nejvyšší úroveň prezentována různými ERP a MES systémy, informačními



Obrázek 4: OPC vztah klient/server

a ekonomickými systémy. V tomto případě je OPC využíváno právě pro vytvoření společného komunikačního rozhraní pro přenosy mezi řídícími a databázovými systémy a systémy SCADA. [3]

5.4.2 Princip OPC komunikace

Princip komunikace je založen na technologii COM. Klient pro inicializaci komunikace potřebuje znát unikátní identifikátor serveru, se kterým se chce spojit. Server tento identifikátor získá při instalaci a následné registraci. Spuštěný program musí zaregistrovat svoji komponentu, která umožňuje vytvářet instance OPC serveru. Tento komponent následně požádá o vytvoření OPC serveru. Ten následně vrátí klientovi. Klient tak dostane ukazatel na objekt, kterému následně může zasílat dotazy. Proces OPC serveru je spuštěn vždy pouze jednou a musí umožnit vytvořit libovolný počet instancí objektu OPC serveru. Jedna instance je přístupná pouze jednomu klientovi. Server nerozlišuje přihlášení ze vzdáleného počítače nebo z lokální sítě. Z tohoto důvodu je oprávnění otázkou nastavení operačního systému a ne nastavení serveru.

5.4.3 Použití a účel OPC

OPC je primárně pro přístup k datům ze síťových serverů. Je použitelný na více místech aplikace. Může například získávat data ze zařízení umístěných přímo v procesu a posílat je do SCADA systému, nebo posílat data ze SCADA systému přímo do vizualizace. Díky své architektuře umožňuje jedné aplikaci získávat data z více OPC serverů nezávisle na výrobci připojených serverů. [2]

5.4.4 Možnosti datového přenosu

V závislosti na požadavcích klienta si u OPC standardu můžeme vybrat mezi několika způsoby komunikace:

- Synchronní komunikace – zde vždy čekáme na směr přenosu dat z/do zařízení
- Synchronní komunikace s vyrovnávací pamětí
- Asynchronní komunikace – zde je komunikace nezávislá na směru přenosu, data můžeme zároveň přijímat i vysílat
- Periodická komunikace – v tomto případě dochází ke komunikaci v předem definovaných časových intervalech, nebo pouze při změně požadovaných dat

V případě synchronní komunikace a asynchronní komunikace s vyrovnávací pamětí je nutné zajistit, aby samotný server dokázal vyvolat komunikaci s HW částí a přečtená data buď odeslat přímo klientovi, nebo je ukládat do vyrovnávací paměti. Bohužel i při sebelepší definici rozhraní a popisu jeho chování nelze dostatečně zajistit, aby se OPC této definice drželo. V důsledku toho je možné narazit na servery ignorující požadavky na četní/zápis dat ze zařízení a vracejí tak pouze data z vyrovnávací paměti. Takový server pak může aplikaci učinit nestabilní a nelze se na něj spolehnout ani v případě, kdy aplikace vyžaduje potvrzování komunikace nebo na časovou odezvu při komunikaci.

5.4.5 Konfigurace pro Windows XP

Stejně jako ostatní ovladače, tak i OPC využívá pro konfiguraci dva soubory, soubor *.pad a soubor *.dmf. Na rozdíl od ovladače DBNet se struktura těchto souboru liší podle typu a výrobce daného serveru. Z toho důvodu bylo vyvinuto grafické rozhraní umožňující pohodlnou konfiguraci bez nutné znalosti konfiguračních souborů. Parametrický soubor má strukturu stejnou jako konvenční *.ini soubory.

5.5 Konfigurace OPC

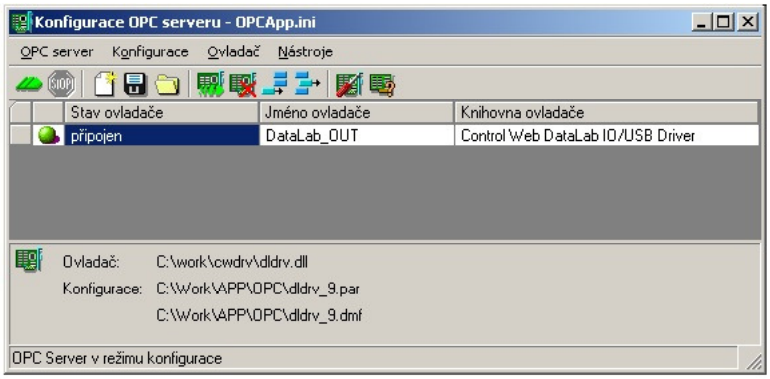
OPC, stejně jako jiné ovladače pro ControlWeb, využívá ke své konfiguraci *.ini soubory. Jeden parametrický a druhý mapový. Mapový soubor je pro OPC strukturálně totožný s mapovým souborem pro ovladač DBNet32. Parametrický soubor obsahuje několik sekcí označených hranatými závorkami [server] a [channels].

```
[comment]
OPCDRV PAR soubor vytvořen konfigurátorem OPC ovladače
Soubor: C:\BUFF\opc.par
Vytvořen: 04/30/2002 16:31:21

[server]
CLSID={F8582CF2-88FB-11D0-B850-00C0F0104305}
host=\\remotehost

[channels]
100=Random.Int4
101=Random.String
102=Random.UInt4
103=Random.Real4
104=Random.Real8
105=Square Waves.Real8
106=Triangle Waves.Real8
107=Write Only.UInt1
108=Write Only.UInt2
109=Write Only.UInt4
110-119=Random.StringArray
120-129=Random.Real8Array
```

Obrázek 5: struktura parametrického souboru OPC



Obrázek 6: náhled na konfiguraci OPC

```
číslo_kanálu=identifikátor položky OPC serveru
```

Obrázek 7: zavedení jednoho kanálů do OPC

```
první_číslo_kanálu-poslední_číslo_kanálu=identifikátor položky pole OPC serveru
```

Obrázek 8: zavedení pole kanálů do OPC

5.5.1 server

Tato sekce obsahuje identifikátor třídy serveru označený jako CLSID. Jedná se o unikátní 128-bitový identifikátor (GUID). Toto číslo se používá k veškerých tříd, rozhraní a knihoven používaných v COM. Formát čísla je definován v COM standardu. Toto číslo je poté uloženo v registrační databázi HKEY CLASSES ROOT. Pokud má být server vytvořen na vzdáleném počítači, pak musí být nadefinován i parametr host. V tomto parametru je nadefinován i parametr UNC představující jméno serveru s nainstalovaným OPC serverem. OPC ovladač implicitně komunikuje synchronně se servery verze 1 a asynchronně se servery verze 2 a 3. V případě, že asynchronní komunikace není podporována, pak servery komunikují synchronně. Pokud chceme u verzí 2 a 3 použít z nějakého důvodu synchronní komunikaci, pak musím v parametrickém souboru použít parametr **sync=true**. Pokud je tento parametr nastaven na hodnotu **false**, pak OPC ovladač upřednostňuje asynchronní komunikaci.

5.5.2 Channels

sekce channels obsahuje definice jednotlivých kanálů. Vždy obsahuje identifikátor kanálu a identifikátor položky OPC serveru. Pokud chceme zavést pole kanálů, pak zavedeme do sekce celý interval kanálů.

6 DBNET32

Pro komunikaci s ControlWebebem použijeme ovladač DBNet32. Jedná se v podstatě o ovladač ATOUCH32 uzpůsobený pro prostředí ControlWebu. Tento ovladač slouží pro přímé spojení mezi PLC a Stanici nebo PC. Tento ovladač je opc client konfigurován pomocí dvou souborů. Jeden s příponou *.par a druhý s příponou *.dmf. Souboru *.par se nazývá parametrický a obsahuje specifikaci veškerých komunikačních kanálů, používaných aplikací nahranou v PLC. Tyto kanály jsou automaticky definovány v DetStudiu a nedoporučuje se je měnit. Jejich změna vede k nefunkčnosti spojení a bez

```
========== HW_BEGIN ==========
[General]
MyStation=31
========== HW_END ============

========== DB_BEGIN ==========
========== DB_END ============

========== AP_BEGIN ==========
[system]
Channel=100000

[text]
   ...

[dwtext]
   ...

========== AP_END ============
```

Obrázek 9: struktura parametrického souboru

vlastní aplikace pro PLC je téměř nemožné je zpětně dohledat. Druhý soubor s příponou *.dmf se nazývá mapový a obsahuje veškeré nastavení ohledně komunikace. Tento soubor je důležitý pro vlastní ustanovení komunikace, obsahuje totiž specifikace veškerých COM portů a identifikace všech PLC a PC, ze kterých vizualizace snímá hodnoty. Pokud máme nadefinované oba tyto soubory, pak v prostředí ControlWebu vybereme záložku "datový inspektor"a do položky ovladače vložíme ovladač DBNet, kde musíme vybrat právě tyto dva soubory a knihovnu s názvem DDBNET32.DLL. Takto zavedený ovladač se poté převede do zdrojového textu vizualizace.

6.1 Parametrický soubor

Parametrický soubor, ostatně jako všechny konfigurační soubory, je jen prostý text, popisující ale důležité parametry. Jako prostý text může být tedy upravován v libovolném textovém editoru nebo ho lze upravovat přímo v ControlWebu. Základ tohoto souboru vypadá takto:

Parametrický soubor je postaven podle následujících pravidel:

- Každý klíč je definován jako jednoduchý parametr a musí být uveden na kraji samostatného řádku
- Parametry uvedené ve stejné sekci tvoří logické celky

- Jména sekcí musí být uvedená od kraje na samostatných řádcích
- Jednotlivé názvy a sekcí nejsou casesensitivní
- Mezi jednotlivými sekcemi mohou být vloženy prázdné řádky

6.2 Mapový soubor

Mapový soubor s příponou *.dmf je součástí konfigurace ovladače DBNet32. Tento soubor obsahuje deklaraci proměnných, tzn. čísla kanálu, na kterém můžeme dané proměnné odposlouchávat, datový typ a jejich směr. Datové typy zde mohou být:

- longint
- integer
- shortint
- longreal
- real
- shortreal
- longcard
- cardinal
- shortcard
- string
- boolean

Přiděleným datovým typem říkáme samotné vizualizaci, se kterým chceme pracovat. Všechny kanály se přizpůsobují požadovanému typu včetně přetypování na typ string. Můžeme klidně kanálu zpřístupňující proměnnou typu FLOAT přidělit typ longcard a ovladač provede potřebné konverze. Riskujeme tím ale ztrátu informací. Pokud např. konvertujeme z typu float na typ int, pak dojde k ořezání desetinných míst.

Definované směry kanálů mohou nabývat hodnot:

- input– povoluje pouze vstup informací do aplikace, ale už ne zpět do terminálu. Umožňuje tedy pouze čtení informací

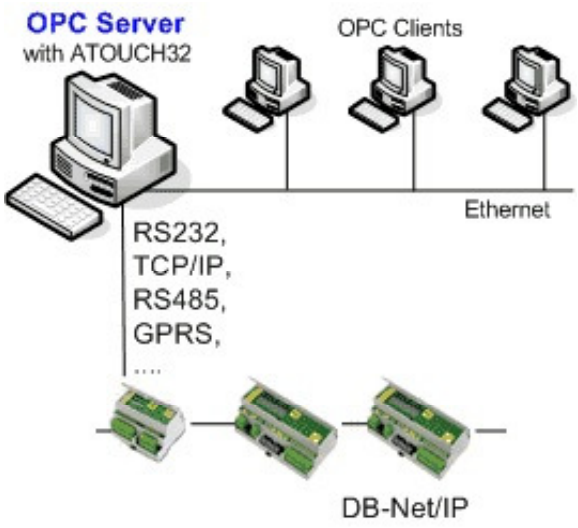
```
begin
 100070 - 100071 longcard  input
 100072          boolean   input
 100073          boolean   output
 100080 - 100084 cardinal  input
 100085          longcard  input
 100086 - 100087 string    input
 100090          cardinal  output
 100100 - 100355 cardinal  input
 100400 - 100655 string    bidirect
 100700 - 100955 cardinal  input
end.
```

Obrázek 10: struktura mapového souboru

- output – povoluje pouze zápis do terminálu, ale nepovoluje čtení dat. Je tedy použitelný pouze pro řízení
- bidirectional – tento kanál je obousměrný a umožňuje tedy jak čtení, tak zápis do terminálu

7 Merz OPC DBNet/IP

Jedná se o programové vybavení umožňující zapojit řídicí systém do internetové sítě za použití TCP/IP protokolů. Souhrnně se tato síť označuje jako "Průmyslový Ethernet". Obousměrný přenos dat pomocí průmyslového Ethernetu zajišťují speciální funkční moduly. Tyto moduly zajišťují definici řídicího systému na Ethernetu, identifikaci vzdálených stanic, statické směrování a přenos proměnných po Ethernetu. Komunikační ovladače, které poté zajišťují samotný přenos dat, využívají protokolu OPC. OPC vrstva, která je u tohoto ovladače použita odpovídá specifikaci DA 2.05. Dále umožňuje připojení, jak pomocí sítě, tak běhu na jedné stanici, kde je spuštěn jak OPC klient, tak i server. Díky základům z ovladače ATOUCH32 je možné využít ke komunikaci např. RS232/485, Ethernet, GPRS a modemu. DBNet/IP je konfigurován taktéž pomocí dvou *.ini souborů, podobně jako DBNet/IP. Jeden soubor je použit ke konfiguraci hardware a pomocí druhého se definují proměnné. [4]



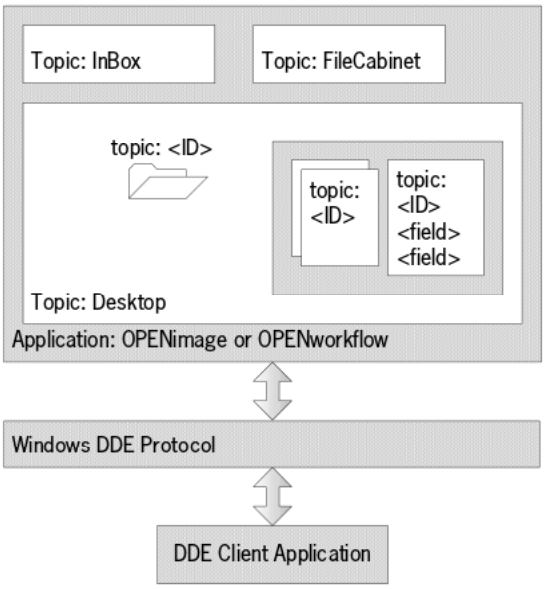
Obrázek 11: DBNet/IP struktura

8 DDE

DDE je standard sloužicí ke sdílení dat a je určen k výměně dat mezi několika aplikacemi na jednom počítači. Ke správné funkci DDE je potřeba minimálně DDE server, přes který bude sdílena potřebná množina dat a proměnných, a alespoň jeden DDE klient. Klient je vlastníkem veškerých proměnných a klienti se na základě udělených oprávnění odkazují na tyto proměnné. Proměnné se skládají ze tří částí:

- service – název aplikace
- topic – název oblasti dat
- item – název konkrétní datové položkyBakalářská práce

V průmyslové automatizaci je standard DDE využívá k získávání dat z PLC automatů. K tomu je potřeba správný ovladač pro PLC. Ten je dodávaný výrobcem. DDE je rozhraní v podstatě totožné s OPC. V principu totiž, stejně jako OPC, získává data z PLC a překládá je do formy srozumitelné naší aplikaci. Rozdíl je ovšem v rychlosti, kdy přenos dat mezi serverem a klientem je DDE mnohem pomalejší. V jeho prospěch mluví ale jednoduchost použití, kdy není potřeba nijak zvlášť složité konfigurace. [**?**]



Obrázek 12: DDE client/server

8.0.1 Client/Server model DDE

Tento model zobrazuje, jakým způsobem vlastně DDE pracuje. Server, který využívá, je vlastně aplikace, poskytující specifický druh služeb, v tomto případě zpracování a překlad dat. Naproti tomu je zde klient, což je další aplikace, která tyto služby požaduje pro svou funkci.

9 Tvorba praktické části

První aplikaci vytvoříme pro komunikaci pomocí DBNet32. Prvním krokem bude tedy založení nového projektu, kdy vybereme možnost *aplikace řízená reálným časem*. Vybereme rozložení hlavního panelu, a tím máme připraven naprostý základ. Dále musíme v datovém inspektoru nadefinovat používané proměnné, komunikační kanály, alarmy a ostatní datové prvky. Musíme také vybrat vhodný ovladač. V datovém inspektoru v záložce ovladače vybereme *přidat ovladač* a zvolíme libovolné jméno. Pod tímto jménem bude ovladač zobrazován v celé vizualizaci a budeme ho používat i při nastavování činnosti virtuálních přístrojů, proto je vhodné zvolit název tak, abychom i později poznali, o jaký ovladač se jedná, obzvláště pokud jich používáme více současně. Důležitou součástí je navolení správného ovladače, v našem případě DBNET v4.09 a správných *.par a *.dmf souborů do položky Parametrický a Mapový

soubor. Dále nadefinujeme komunikační kanály. Těm navolíme opět vhodný název a vybereme odpovídající WID (číslo kanálu) zapsaném v mapovém souboru.

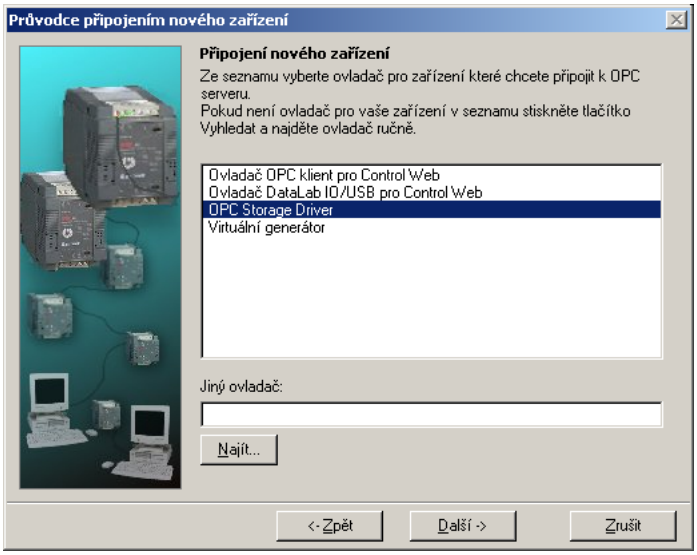
9.1 Instalace OPC serveru

OPC se spouští ve stylu LocalServer. To znamená, že server je spuštěn jako dva samostatné procesy. Tento server pak komunikuje s ostatními nainstalovanými klienty. Při registraci je tedy zapsán unikátní identifikátor OPC serveru a COM komponenty. Při používání ActiveX. Tyto komponenty jsou spuštěny jako InProcServery, server a klienti pak běží jako jeden proces. Při samotné instalaci pak musí dojít k registraci serveru do systému. Při instalaci je pak Server rozpoznatelný i pro ostatní klienty. Při instalaci OPC serveru jsou nainstalované i komunikační knihovny dodávané organizací OPC Foundation. Tyto knihovny zprostředkovávají meziprocesorovou komunikaci mezi klientem a serverem. Další součástí instalace je služba OpcEnum.exe, která umožňuje klientům zjistit všechny nainstalované OPC servery.

9.2 Konfigurace OPC serveru

První věcí, kterou po instalaci serveru musíme udělat, je nastavení datových skladů. Definujeme tedy datové elementy, sloužící k předávání dat. Po první instalaci serveru se nám konfigurační rozhraní otevře automaticky. Vybereme tedy *Průvodce připojením nového ovladače*. Zde zadáme libovolné jméno a poté i typ našeho ovladače **OPC storage driver**.

Dalším krokem bylo definování datových elementů. Pro test jsem zvolil logický typ boolean, dále real a string.



Obrázek 13: konfigurace OPC Storage driver

9.3 Konfigurace OPC

Pro konfiguraci OPC jsem použil konfigurační nástroj, který umožňuje nastavení serveru v grafickém rozhraní. Umožňuje nám zjistit CLSID a jména položek serveru. Vybral jsem položky, které mají být viditelné v naší aplikaci v ControlWebu. Tento nástroj nám umožní i vygenerovat náš DMF a PAR soubor.

9.4 Konfigurace klientského ovladače

V aplikaci určené ke konfiguraci se připojíme k našemu OPC serveru. Ze seznamu hodnot v levé části aplikace jsem vybral všechny tři nadefinované proměnné. Tím máme nadefinované komunikační kanály stejně jako v případě ovladače DBNet32. Tímto způsobem všechny tři proměnné získaly podobu kanálů, přes které budou přenášena data. Každý kanál je označen unikátním číslem.

Závěr

Účelem mé práce bylo vyzkoušet a porovnat několik druhů komunikačních ovladačů pro komunikaci s hardwarovým zařízením, konkrétně Amit ART 4400, a porovnat způsoby této komunikace a zpracování dat při ní získaných. Pro přímou komunikaci jsem tedy zvolil ovladač DBNet32 a pro komunikaci umožňující spojení se serverem, a tím pádem i získávání dat po internetové síti jsem zvolil ovladač DBNet/IP OPC Amit. K tomuto ovladači se mi podařilo získat licenci od firmy Merz. Princip konfigurace obou ovladačů probíhá pomocí *.ini souborů obsahujících základní prvky pro spojení. Výhodou spojení přes OPC je možnost přístupu k datům a řízení celého procesu v podstatě z libovolného místa. Důvodem je fakt, že v tomto případě nepotřebujeme přímé spojení s řízeným procesem. Naproti tomu u použití ovladače DBNet32 je nutné přímé připojení do procesu např. pomocí RS232. Z porovnání komunikace přes ovladač DBNet32 a OPC serveru jsou zřejmé následující rozdíly. Komunikace pomocí OPC serveru umožňuje současné sdílení dat mezi více aplikacemi i mezi více klienty současně. Komunikace pomocí DBNet32 umožňuje komunikovat data pouze s jedním klientem.

Reference

[1] J. Hirsch: *Open Database Conectivity version 0.8*. ODBC Reference Manual,

[2] OPC foundation: *OPC OLE for Process Control* . OPC overview, Version 1 October 27.1988

[3] OPC foundation: *OPC Unified Brochure* . OPC Brochure, 2013

[4] Jan Kovář: *Komunikační možnosti systému Amit*. Liberec: Technická univerzita v Liberci Fakulta mechatroniky, informatiky a mezioborových studií, 2011

[5] Unisys Corporation: *DDE Interface*. Guide, Unisys Corporation 1999

[6] Andy Oppel: *SQL bez předchozích znalostí*. Computer Press a.s. 2008, Vydání první

[7] Control Web Help: *DDE*

[8] Control Web Help: *HTTPD*

[9] Moravské přístroje: *Co je OPC?*.
WWW: http://www.mii.cz/art?id=214lang=405 1.3.2005