ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

Fakulta elektrotechnická

Katedra řídící techniky

Diplomová práce

Prostředí pro vzdálenou výuku

Vypracoval: Vladimír Novotný

Prohlášení

Prohlašuji, že jsem svou diplomovou práci vypracoval samostatně a použil jsem pouze podklady (literaturu, projekty, SW, atd.) uvedené v přiloženém seznamu.

Nemám závažný důvod proti užití tohoto školního díla ve smyslu § 60 Zákona čl.121/2000 Sb. , o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon).

V Praze dne ……. …….
podpis

Abstrakt

Tato práce se zabývá problematikou distančního vzdělávání se zaměřením na inženýrské studium v oblasti řídící techniky. Práce navazuje na pilotní projekt katedry řídící techniky ČVUT-FEL. Hlavním cílem bylo rozšíření možností pro distanční vzdělávání zapojením různých modelů, které jsou k dispozici v laboratoři řídících systémů. To zahrnovalo vypracování nové analýzy databázového systému, který vzdálený přístup k modelům spravuje a který byl původně navržen pouze pro jeden model. Tato analýza potvrdila nutnost rozšířit databázový model a upravit webové rozhraní, které musí zajišťovat rezervaci více modelů a umožnit vzdálené připojení k různým modelům. Po vyřešení prvního cíle této diplomové práce jsme se zaměřili na některé problémy, se kterými se setkáváme v praktických úlohách vzdáleného řízení. Jeden z problémů se týká ovlivňování vizualizace změnami na procesní úrovni. Řešení problému bylo demonstrováno na aplikaci vzdáleného řízení pneumatického výtahu a bylo k němu použito produktů firmy Siemens – WinCC v.6 na straně vizualizace a SIMATIC Manager na straně procesní.

Abstract

This work is about remote education focused on study within a framwork of control engineering. The work maintains the way launched by a pilot project on Department of Control Engineering on CTU-FEE. The main task was to enlarge possibilities for remote education by adding different models, which are at disposal in control engineering lab. This included remake a new analysis of database system, which administrates remote access to models. Problem was, that the database system was originally created to be able to handle only one model. The analysis confirmed the necessity of enlargement of database model and adjusting web interface. New websites must be now capable to handle reservations for more models and have to enable remote access to different models as well. By finishing solution of the first goal of this thesis we've focused on some problems that we have to face in workaday remote control tasks. One of them is interference between visualization and process part of a project. This topic means what kinds of consequences have changes on process part towards visualization. The solution was demonstrated on application of remote controlling of pneumatic lift. We used Siemens products - WinCC v.6 on the visualization side and SIMATIC Manager on the process side.

Obsah

1 Úvod

1.1 Obecný úvod

S rozvojem informačních technologií, především sítě internet, v posledních letech se vzdálené řízení stalo jednou z nejžádanějších a nejrozšířenějších technologií. Zároveň ale již pominulo období, kdy bylo řízení na dálku považováno za žhavou novinku. Proto se v této diplomové prácí nespokojíme s pouhým použitím této metody řízení, ale zaměříme se také na řešení problémů, které s sebou tato technologie přináší. Inspirací pro tuto práci se staly skutečné aplikace v průmyslu. Prvním cílem naší práce je tedy demonstrace vzdáleného řízení. V našem případě byl k tomuto účelu vybrán model výtahu. Odložme prozatím popis tohoto modelu a podívejme se nejprve na motivaci k zadání této práce.

Ke vzdálenému řízení patří přirozeně vizualizace řízené technologie. Omezíme-li se na náš příklad řízení pneumatického výtahu ve školních podmínkách, bude ke kompletní vizualizaci postačovat několik procesních obrazovek. Budeme-li však chtít řídit nějakou skutečnou rozsáhlejší technologii – uvažujme například rafinerii ropy – počet procesních obrazovek dosáhne snadno stovek i tisíců. Často se pak v praxi stává, že zákazník má k předloženému řešení výhrady či drobné dodatečné požadavky, které nebyly nebo z principu ani nemohly být specifikovány předem. Tyto připomínky se mohou týkat jak požadované funkčnosti na procesní úrovni tak výsledného vzhledu vizualizace. Splnění prvního z požadavků sebou může přinést změnu adresace v programu, a tím nefunkčnost dosavadní vizualizace, která proměnné z těchto pevných adres čte či na ně zapisuje. Ani splnění druhého z požadavků nemusí být jednoduché. Vezměme pro představu přání zákazníka na jiný tvar či barvu zobrazovaných ventilů, kterých se na našich obrazovkách rafinerie vyskytují stovky. V obou výše uvedených příkladech by ruční úpravy vizualizace byly neúnosně nákladné především z časového hlediska.

Analýza a návrh řešení výše popsaných problémů je proto ihned po vyřešení základní úlohy vzdáleného řízení jeden z hlavních cílů této diplomové práce. V kapitole 5 této diplomové práce nalezneme ukázku jak řešení zmiňovaných problémů co nejvíce zautomatizovat. Nejprve však musíme vytvořit aplikaci pro vzdálené řízení našeho laboratorního modelu - pneumatického výtahu.

Dalším cílem této práce je zapojení našeho modelu do systému distančního vzdělávání. Správa modelů určených ke vzdálenému řízení spadá pod databázový systém, který však byl původně navržen pouze pro jeden model. Po krátké analýze jsme došli k názoru, že původní koncepci databáze bude nutné předělat a uzpůsobit pro více modelů. Změny se také nevyhnou webovému rozhranní, které má na starosti on-line rezervaci modelu a vzdálené připojení k němu. I webové rozhranní bylo však navrženo pouze pro jeden model. Tímto tématem se budeme zabývat až téměř na konci této publikace. Pro názornost budeme postupovat nejdříve od popisu modelu, který jsme si na demonstraci naší úlohy zvolili, přes ukázku jeho vzdáleného řízení a řešení problémů, které se při tom vyskytují. Dále se zaměříme, jak už bylo zmíněno výše, na automatizaci tvorby a úpravy vizualizačních obrazovek a nakonec zapojíme náš model do již rozšířeného systému pro distanční vzdělávání Lablink.

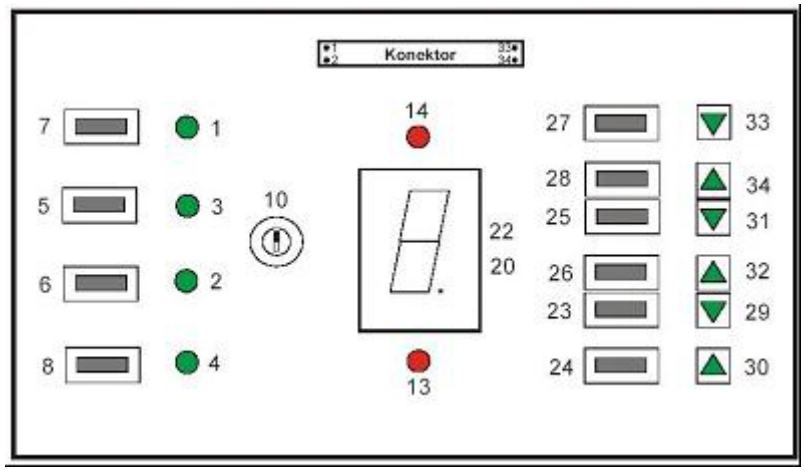
2 Model pneumatického výtahu

2.1 Popis použitého modelu

Jak již bylo řečeno dříve, k demonstraci vzdáleného řízení byl vybrán model výtahu. Model se skládá ze tří oddělených výtahových šachet, které tvoří trojice plexisklových trubek (Obr. 2.1.1). Kabinu výtahu představuje papírový váleček, který se v těchto trubicích může vertikálně pohybovat. Pod každou trubicí se nachází jeden ventilátor, který zajišťuje pohyb výtahu. Změnou otáček větráčku se mění intenzita proudění vzduchu a kabina výtahu se pohybuje nahoru nebo dolů. Fyzikální identifikace systému byla provedena v předchozí diplomové práci [1].

Obr. 2.1.1: Model pneumatického výtahu

Model lze ovládat manuálně pomocí ovládacího panelu (Obr. 2.1.2), který je umístěn u modelu, nebo vzdáleně z vizualizace. Ke vzdálenému řízení modelu z vizualizace se vrátíme v kapitole 4.

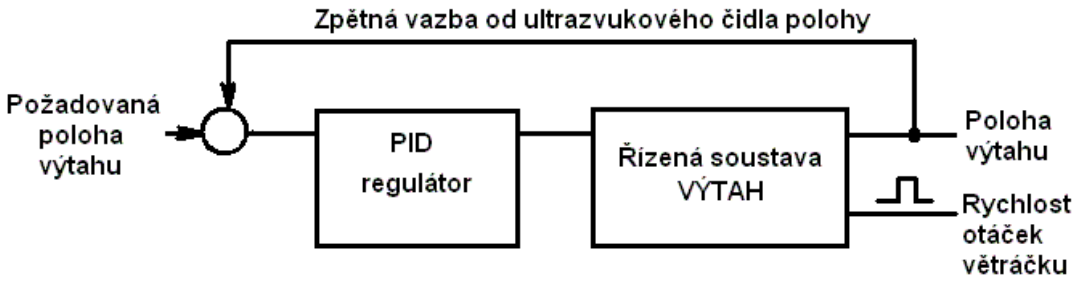


Obr. 2.1.2: Schéma ovládacího panelu jednoho z výtahů

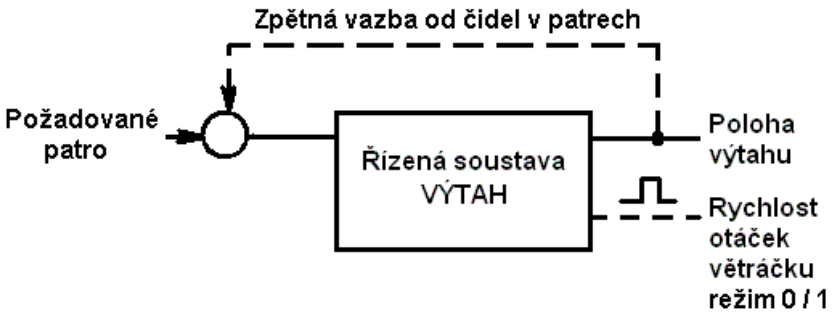
Očíslování prvků na obr. 2.1.2 vychází z označení pinů 34-pinového konektoru, pomocí kterého je panel připojen ke svorkám WAGO a odtud přes PROFIBUS k řídícímu automatu SIMATIC. Podrobnější popis zapojení lze nalézt v literatuře [2]. Každý ze tří výtahů má vlastní ovládací panel. Nyní si popíšeme jednotlivé ovládací prvky. Levá polovina panelu obsahuje tlačítka a signalizační prvky, které by se ve skutečnosti nacházely uvnitř výtahu. V pravé polovině jsou pak ovládací prvky představující tlačítka v jednotlivých patrech. Tlačítky 5-8 se volí požadované patro nacházíme-li se v kabině výtahu. Stisk tlačítek signalizují LED diody 1-4. Přepínač 10 simuluje obsazenost výtahu. Sedmisegmentový displej uprostřed panelu zobrazuje aktuální patro a LED diody 13-14 zobrazují směr jízdy výtahu dolů resp. nahoru. Tlačítka 23-28 slouží k přivolání výtahu do jednotlivých pater s úmyslem jízdy dolů nebo nahoru. Platnost požadavku na přivolání výtahu zobrazují šipky 29-34. Více o tom jak se získávají informace o aktuální pozici výtahu a o vlastním řízení výtahu se dovíme v následující kapitole. Nyní si pouze řekneme, že řízení výtahu je navrženo stejně jako u skutečných výtahů. Tzn., že výtah obsluhuje postupně takové požadavky na přivolání, které zachovávají směr jízdy výtahu. Programově lze zvolit také mód kooperace výtahů, což znamená, že přijede výtah, který je nejblíže a pohybuje se požadovaným směrem.

2.2 Popis řízení systému

Úlohu lze formulovat jako řízení tří paralelních proudem vzduchu poháněných výtahů. Podle druhu řízení můžeme rozdělit model na dvě části. U prostředního výtahu se jedná o skutečné zpětnovazební řízení, kde akční veličinu představuje výstup PID regulátoru a jako zpětnou vazbu o tom, kde se výtah momentálně nachází, využíváme signál ultrazvukového snímače polohy (obr. 2.2.5). Schéma použitého typu řízení je na obrázku 2.2.1. Naproti tomu oba krajní výtahy jsou řízeny binárním signálem, který určuje, zda se má výtah pohybovat nahoru nebo dolů. Stavu, kdy výtah stojí, je zde dosaženo rychlým přepínáním mezi stavy *„jeď nahoru“* a *„jeď dolů“*. Zpětná vazba je v tomto případě pouze digitálního charakteru, a sice jedná se o informaci z čidel v jednotlivých patrech (obr. 2.2.4). Schéma digitálního řízení zobrazuje obrázek 2.2.2. Pokud se takto řízený výtah pohybuje mezi patry nemáme informaci o jeho přesné poloze, víme jen v jakém patře se naposledy nacházel a jakým směrem se pohybuje. Lépe řečeno víme jakým směrem chceme, aby se pohyboval. Tj. posíláme-li výtahu povel *„jeď dolů“*, *„jeď nahoru“* či *„stůj“* předpokládáme, že povel bude vykonán. Pokud se však z nějakého důvodu vyskytne problém na fyzické části zařízení a výtah nebude správně následovat řídící pokyny, nemůžeme si být v takovém případě jisti, kde se výtah skutečně nachází, do chvíle než obdržíme informaci z některého z čidel v patrech.



Obr. 2.2.1: Zjednodušené schéma zpětnovazebního řízení prostředního výtahu



Obr. 2.2.2: Zjednodušené schéma digitálního řízení krajních výtahů

Na předchozích náčrtech je diskrétní tok signálu znázorněn přerušovanou čarou. Vidíme, že více možností a lepší kontrolu nad zařízením nám poskytuje kontinuální zpětnovazební řízení. Výhodou digitálního řízení je naproti tomu jednodušší ovládání výtahu pouze pomocí binárního signálu a využití jednodušších optických čidel v patrech oproti dražšímu ultrazvukovému čidlu použitému pro detekci přesné polohy prostředního výtahu. Optická čidla polohy v patrech jsou využita i při řízení prostředního výtahu.

Podrobný teoretický rozbor řízení výtahu a identifikaci tohoto systému lze nalézt v literatuře [1]. K vlastnímu řízení modelu s využitím průmyslových automatů se vrátíme v kapitole 3 *„Řízení a vzdálené řízení modelu“*. Nyní se však nejprve zaměříme na to, z jakých vstupů a výstupů se naše řízená soustava skládá.

2.2.1 Vstupy systému

Některé signály se mohou z jednoho pohledu jevit jako vstupy a z opačného pohledu jako výstupy. Příkladem budiž signál z ultrazvukového čidla polohy (viz schéma na obr.2.2.1). V našem případě se budeme držet standardního přístupu a jako referenční bod zvolíme řídící automat SIMATIC. Jako vstupy systému pak označíme takové signály, které se vůči řídící jednotce chovají jako vstupní proměnné. Jsou to tedy proměnné, které lze pouze číst a nelze na ně zapisovat. A jako výstupy systému nazveme výstupní proměnné, které lze rovněž číst ale hlavně na ně můžeme zapisovat.

Vstupy systému tvoří především požadavky na přivolání výtahu v různých patrech. Tyto vstupy jsou binárního pulsního charakteru, tj. jejich hodnota je „1“ nebo chcete-li *„high“* pouze po dobu stisknutého tlačítka. Pro využití těchto signálů v řídícím algoritmu je nutné tyto vstupy nejdříve zkopírovat do paměťového prostoru automatu, kde se ukládají do inteligentní fronty jako požadavky na přivolání výtahu. Více o zpracování požadavků a vytváření inteligentní fronty se můžeme dočíst v literatuře [1], a proto se tímto tématem nebudeme podrobněji zabývat. Řekněme si pouze, že požadavky na přivolání výtahu jsou řazeny tak, aby se výtah choval jako v reálném životě, tj. základním principem je zachování původního směru jízdy výtahu. Ve vizualizaci potom pracujeme pouze s obrazy vstupů v paměti automatu, proto v tabulce 2.2.1, která obsahuje seznam všech použitých vstupů, nalezneme také výpis paměťových buněk použitých pro uložení obrazů vstupů. Podobným způsobem jako požadavky na přivolání výtahu v patrech jsou zpracovány stisknutá tlačítka uvnitř výtahů.

Dalšími již zmiňovanými vstupy jsou signály z optických čidel v jednotlivých patrech (obr. 2.2.4). Jedná se o čidla firmy SENS vhodná pro detekci předmětu v rozmezí 10-550mm. V každém patře se nachází jedno čidlo a každý ze tří výtahů se může pohybovat od 0. do 3. patra. Celkem tedy používáme dvanáct těchto optických závor. Čidla umožňují různé režimy nastavení, v našem případě však používáme pro všechna čidla stejný režim a sice takový, kdy při přerušení světelného paprsku dostáváme na výstupu senzoru vysokou logickou úroveň +24V.

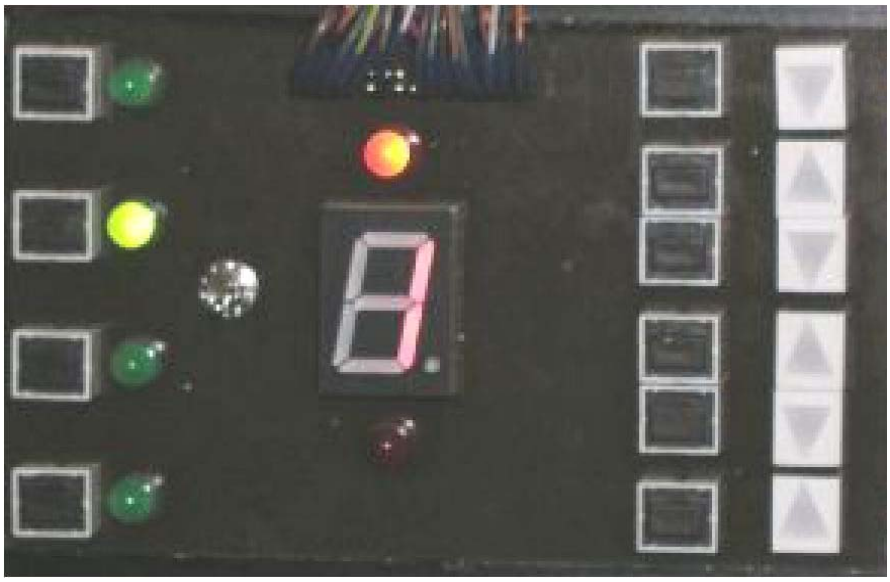
Pro přesné zjištění polohy prostředního výtahu a pro ukázku PID zpětnovazebního řízení používáme navíc ultrazvukový snímač polohy firmy MICROSONIC (obr. 2.2.5). Výstupem tohoto snímače je analogový signál 4–20mA a 2–10V. Tento snímač je pro správnou funkci nutné zkalibrovat. Při současném nastavení odpovídá poloze výtahu v 0.patře

hodnota napětí na výstupu senzoru 2.4 V a poloze výtahu v posledním třetím patře odpovídá výstupní napětí 9.2 V.

Posledními vstupy systému jsou přepínače simulující obsazenost jednotlivých výtahů, které se nacházejí na ovládacích panelech modelu (obr. 2.1.2 ovládací prvek č. 10). Detailnější technické parametry a informace o použitých součástí modelu lze nalézt v literatuře [2].

Následující obrázky zachycují jeden z ovládacích panelů výtahu, jedno z optických čidel umístěného v patře výtahu a ultrazvukový senzor na vrcholu prostřední výtahové šachty. Na ovládacím panelu si můžeme všimnout aktuálního stavu – bylo stisknuto tlačítko uvnitř výtahu s požadavkem *„jeď do druhého patra“*. Dále vidíme, že poslední registrované patro bylo první a oranžová LED dioda informuje o aktuálním směru výtahu – tedy jízdě nahoru. Vidíme také, že není registrován žádný požadavek z vnějšku na přivolání výtahu do některého patra.

V systému existuje ještě jeden prvek, který bychom mohli považovat za vstupní – je jím webová kamera. Signál z této kamery však v řídícím automatu nijak nezpracováváme, kamera dokonce není s řídící jednotkou propojena, a proto kameru mezi vstupy systému řadit nebudeme. Obraz snímaný touto kamerou však odesíláme jak do webového rozhranní vzdáleného přístupu k modelu tak do vizualizace ve WinCC, proto se blíže vrátíme k tomuto prvku v kapitolách 4 a 6 zaměřených na vzdálené řízení.



Obr. 2.2.3: Ovládací panel jednoho z výtahů

Obr. 2.2.4: Optické čidlo v patře výtahu

Obr. 2.2.5: Ultrazvukový snímač polohy

Tabulka 2.2.1 na následující straně zahrnuje všechny vstupy modelu včetně jejich adres a fyzického umístění na modelu. Význam použité syntaxe označení proměnných popisuje kapitola 3.4 *„SIMATIC – teoretický úvod“*. Jedinými vstupy, které nejsou fyzické, existují tedy pouze v programu, jsou přepínače kooperace výtahů. Při povolení kooperace bude daný výtah vyhodnocovat i požadavky na přivolání ostatních výtahů se zapnutou volbou kooperace. V případě, že pak tento výtah bude blíže než výtah, u kterého bylo přivolávací tlačítko stisknuto, požadavku vyhoví a přijede.

Popis vstupu	Levý výtah	Prostřední výtah	Pravý výtah	Referenční číslo*
tlačítko uvnitř, 0.patro	I5.0 M3.0**	I9.0 M4.0	I11.0 M5.0	8
tlačítko uvnitř, 1.patro	I5.1 M3.1	I9.1 M4.1	I11.1 M5.1	6
tlačítko uvnitř, 2.patro	I5.2 M3.2	I9.2 M4.2	I11.2 M5.2	5
tlačítko uvnitř, 3.patro	I5.3 M3.3	I9.3 M4.3	I11.3 M5.3	7
tlačítko venku, 0.patro nahoru	I4.0 M0.0	I8.0 M1.0	I10.0 M2.0	24
tlačítko venku, 1.patro dolu	I4.1 M0.1	I8.1 M1.1	I10.1 M2.1	23
tlačítko venku, 1.patro nahoru	I4.2 M0.2	I8.2 M1.2	I10.2 M2.2	26
tlačítko venku, 2.patro dolu	I4.3 M0.3	I8.3 M1.3	I10.3 M2.3	25
tlačítko venku, 2.patro nahoru	I4.4 M0.4	I8.4 M1.4	I10.4 M2.4	28
tlačítko venku, 3.patro dolu	I4.5 M0.5	I8.5 M1.5	I10.5 M2.5	27
tlačítko simulující obsazenost výtahu	I4.6	I8.6	I10.6	10
čidlo polohy 0.patro	I5.4	I9.4	I11.4	35, 39, 43
čidlo polohy 1.patro	I5.5	I9.5	I11.5	36, 40, 44
čidlo polohy 2.patro	I5.6	I9.6	I11.6	37, 41, 45
čidlo polohy 3.patro	I5.7	I9.7	I11.7	38, 42, 46
kooperace výtahu****	M100.0	M100.1	M100.2	47, 48, 49
ultrazvukový snímač	-	PIW320***		50

Tab. 2.2.1: Vstupy systému výtahu a jejich adresace

* Referenční číslo odpovídá popisu ovládacího panelu na obrázku 2.1.2 případně na obrazovce vizualizace na obrázku 4.2.1.

** Paměťová místa použitá pro uložení pulsních vstupů.

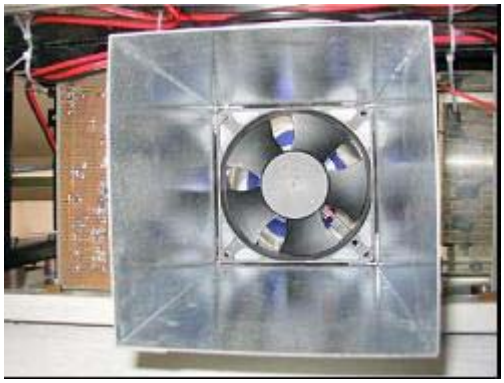
*** Adresuje se přímo WAGO periferní modul.

**** Pouze programový vstup. Při povolení této volby výtahy vzájemně spolupracují, tj. při požadavku na přivolání výtahu z venčí přijede nebližší z výtahů

2.2.2 Výstupy systému

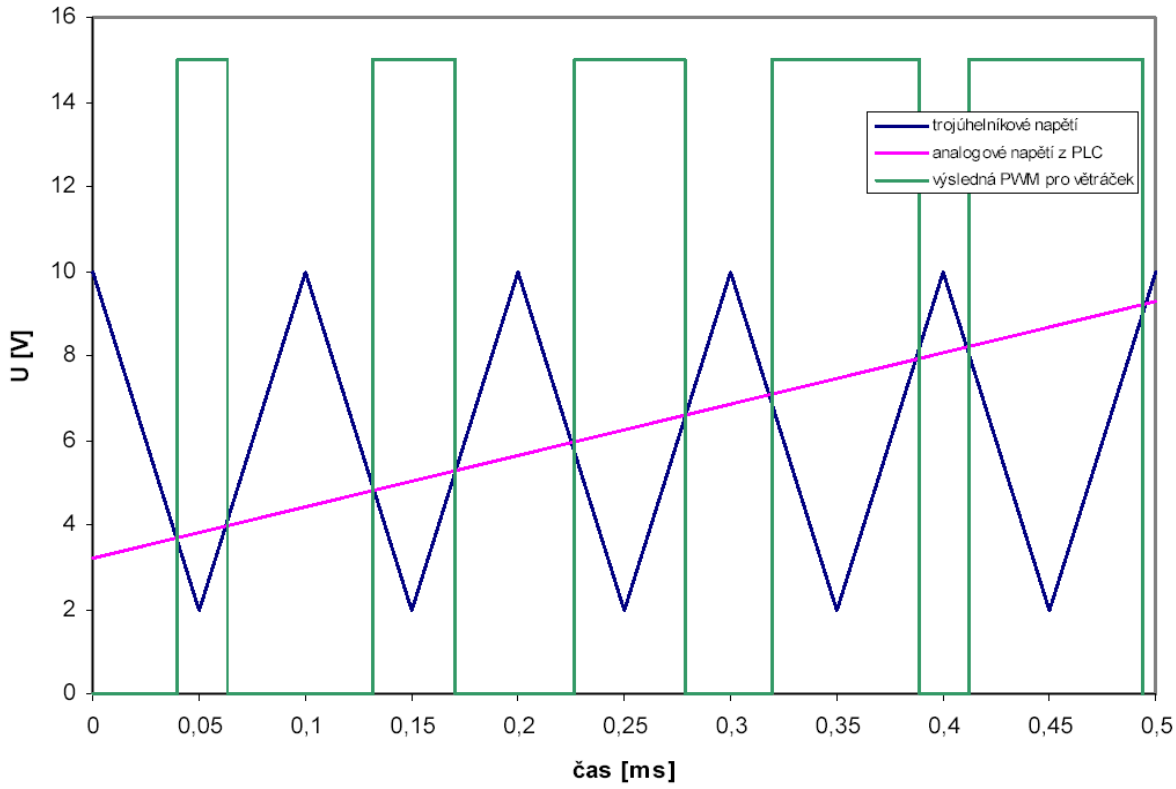
Mezi výstupy systému budeme řadit signalizaci stisknutých tlačítek vně i uvnitř výtahů, sedmisegmentové displeje zobrazující aktuální patra, ve kterých se jednotlivé výtahy nacházejí, LED indikace směru pohybu výtahů a především akční zásahy pro řízení pohybu výtahů.

Na tomto místě se podrobněji podíváme na vlastní řízení pohybu výtahů. Vyjdeme z toho, co již bylo popsáno v kapitole 2.2 *„Popis řízení systému“* a co znázorňují schémata na obr. 2.2.1 a na obr. 2.2.2. Pohyb každého z výtahů zajišťují větráčky umístěné pod každou ze tří výtahových šachet (obr. 2.2.6).

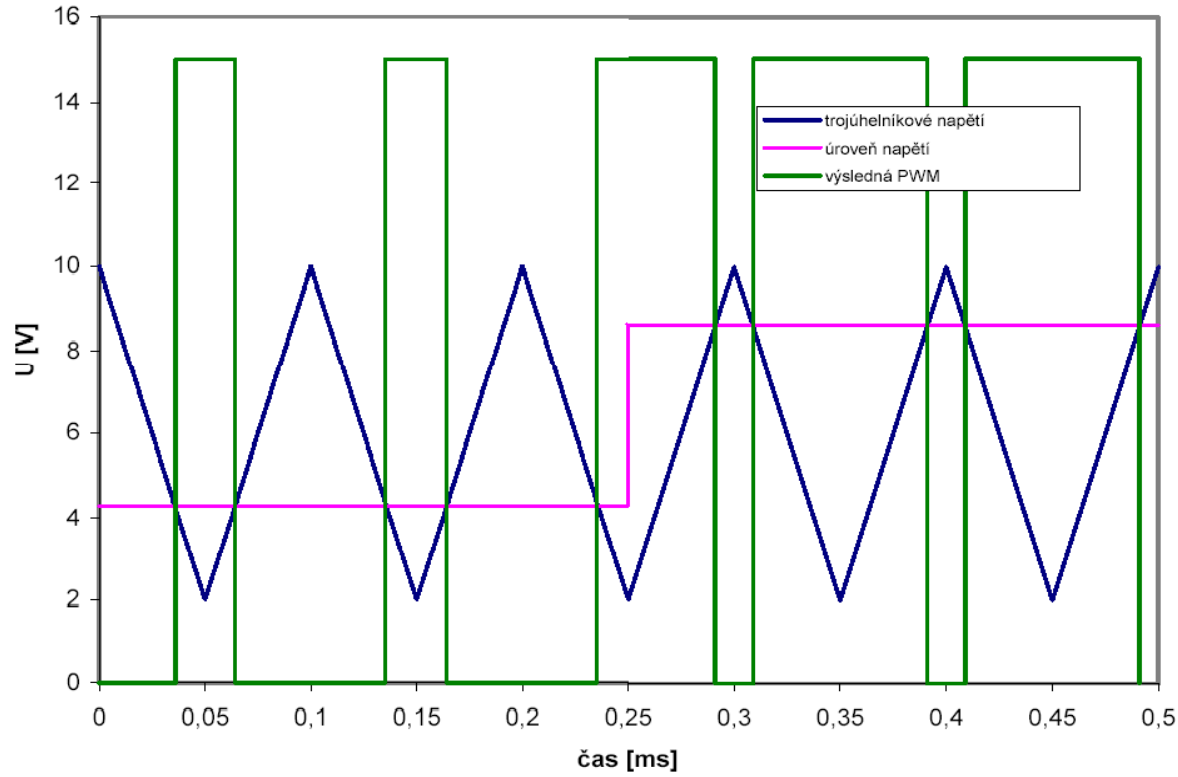


Obr. 2.2.6: Větráček pohánějící výtah (pohled zdola)

Znovu bychom měli upřesnit, že pohyb kabiny výtahu ve výtahové šachtě je zajišťován proudem vzduchu, který tyto větráčky generují. Díky nízké hmotnosti kabiny se pak tato ve vzniklém proudění řízeně vznáší. Pro řízení všech tří větráčků je použito PWM řízení. Princip změny střídy PWM průběhu se však u krajních výtahů liší od principu použitého pro prostřední výtah. Zatímco prostřední výtah je řízen kontinuální změnou analogové hodnoty napětí (schéma na obr. 2.2.1) a tím kontinuální změnou PWM průběhu (obr. 2.2.7), krajní výtahy jsou řízeny binárním signálem (obr. 2.2.8), který přepíná dva režimy PWM s různými střídami pro jízdu výtahu dolů a pro jízdu výtahu nahoru. Obrázky 2.2.7 a 2.2.8 vysvětlují vznik PWM průběhu (zeleně) komparací pilovitého signálu (modře) s komparačním napětím (fialově). Detailní popis HW pro tvorbu popsaných PWM průběhů lze nalézt opět v literatuře [1].



Obr. 2.2.7: PWM pro řízení prostředního výtahu s proměnným komparačním napětím



Obr. 2.2.8: PWM binárně přepínané mezi režimem pro pohyb výtahu dolů(čas < 0,25s) a pro pohyb výtahu nahoru (čas > 0,25s)

Následující tabulka 2.2.2 obsahuje seznam všech použitých výstupů systému pneumatického výtahu včetně jejich adres a fyzického umístění na modelu. Význam použité syntaxe označení proměnných popisuje kapitola 3.4 *„SIMATIC – teoretický úvod“*.

Popis výstupu	**Levý výtah**	**Prostřední výtah**	**Pravý výtah**	**Referenční číslo***
LED dioda uvnitř, 0.patro	Q5.2	Q13.2	Q15.2	4
LED dioda uvnitř, 1.patro	Q5.3	Q13.3	Q15.3	2
LED dioda uvnitř, 2.patro	Q5.4	Q13.4	Q15.4	3
LED dioda uvnitř, 3.patro	Q5.5	Q13.5	Q15.5	1
LED dioda venku, 0.patro nahoru	Q4.0	Q12.0	Q14.0	30
LED dioda venku, 1.patro dolů	Q4.1	Q12.1	Q14.1	29
LED dioda venku, 1.patro nahoru	Q4.2	Q12.2	Q14.2	32
LED dioda venku, 2.patro dolů	Q4.3	Q12.3	Q14.3	31
LED dioda venku, 2.patro nahoru	Q4.4	Q12.12	Q14.14	34
LED dioda venku, 3.patro dolů	Q4.5	Q12.5	Q14.5	33
LED dioda venku, 3.patro dolů	Q4.5	Q12.5	Q14.5	33
aktuální patro	DB1. ** DW18	DB2. DW18	DB3. DW18	20, 22
aktuální směr	DB1. DW16	DB2. DW16	DB3. DW16	13, 14
akční zásah - větráčky	Q5.7	PQW320***	Q15.7	51, 52, 53

Tab. 2.2.2: Výstupy systému výtahu a jejich adresace

* Referenční číslo odpovídá popisu ovládacího panelu na obrázku 2.1.2 případně na obrazovce vizualizace na obrázku 4.2.1.

** Syntaxe *„DB1.DW16“* se používá ve STEP7, ve vizualizace WinCC *„DB1,DW16“*

*** Adresuje se přímo WAGO periferní modul.

2.3 Závěry analýzy vybraného modelu

Před vlastním zapojením modelu výtahu do systému vzdáleného řízení byla provedena analýza SW i HW, jejímž cílem bylo ověřit vhodnost modelu výtahu pro vzdálené řízení.

Z hlediska funkčnosti řídícího SW byl model shledán bezvadným. Po HW stránce byly však nalezeny menší či větší nedostatky. Za hlavní nedostatek lze považovat HW řešení řízení krajních výtahů. Podívejme se na tento problém trochu podrobněji. Fyzikální princip řízení je stejný jako u prostředního výtahu – i zde je ke změně rychlosti otáček větráčku použito PWM řízení. Změny střídy PWM je dosaženo komparací pilovitého průběhu napětí s komparační úrovní napětí. V případě prostředního výtahu (obr. 2.2.7) je toto komparační napětí programově řiditelné – jedná se o výstup PID regulátoru. Tento způsob řízení si můžeme pracovně nazvat *„analogovým“*. V případě krajních výtahů je použito dvou PWM průběhů s odlišnými střídami. PWM s vyšší střídou pro jízdu nahoru, PWM s nižší střídou pro jízdu dolů. Přepínáním těchto dvou průběhů je dosaženo jízdy výtahu nahoru nebo dolů, rychlým přepínáním teoreticky i zastavení výtahu v libovolné poloze. Přepínání se děje na základě binárního řídícího signálu – nazvěme si pro jednoduchost tento druh řízení jako „digitální“.

Problém v použití „digitálního“ řízení vyvstává ve chvíli změní-li se parametry modelu v porovnání se stavem, pro který byly průběhy PWM navrženy. Změnami parametrů máme na mysli zejména posun převodní charakteristiky použitých větráčků. I přes snahu model co nejvíce zlinearizovat a zrobustnit dochází k této situaci, jak se ukázalo při praktických pokusech, relativně často. Mezi realizovanými zlepšeními modelu jmenujme například přidání *„nasávacích tunelů“* k zajištění co možná největší homogenity proudícího vzduchu ve výtahových šachtách nebo vyvážení válečku představující kabinu výtahu a optimalizace jeho tvaru tak, aby nedocházelo k vibracím a kmitání ve výtahové šachtě. Vraťme se nyní zpět do situace kdy zpozorujeme změnu v chování krajních výtahů. V takovém případě musíme zkalibrovat střídu PWM, což znamená změnit jednu nebo podle potřeby obě komparační úrovně napětí *„digitálního“* řízení podle toho jestli se výtah nechová podle očekávání při jízdě dolů, nahoru nebo oběma směry. Tyto napěťové úrovně však nejsou, jako v případě prostředního výtahu, programově ovlivnitelné. Komparační napětí zde vzniká na napěťovém děliči odvozeném z napájecího napětí. Kalibraci PWM průběhů je tak

nutné provést přímo na plošném spoji konkrétního výtahu a to experimentálně změnou odporů potenciometrů v napěťovém děliči.

Jak již bylo zmíněno, problémy s proměnlivostí parametrů výtahu souvisí mimo jiné s pohony výtahů – větráčky. Za další drobný nedostatkem můžeme tedy označit nepříliš vhodnou volbu pohonů výtahu. Použité větráčky vykazují velmi nelineární převodní charakteristiku. Převodní charakteristika vyjadřuje vztah mezi vstupním napětím a otáčkami větráčku. Pokusy bylo zjištěno, že otáčky větráčků lze měnit vstupním napětím jen v relativně malém rozsahu. Do přibližně 60% maximálního napájecího napětí se větráčky netočí (měřeno jako efektivní hodnota výstupního napětí – PWM průběhu). Tato mez však není pevná a závisí na tření vrtulky větráčku. Při používání výtahu se tato mez snižuje zahřátím větráčku a tím snížením tření. Převodní charakteristika se tak stává časově závislou, a proto řešení ovládání krajních výtahů pomocí pevně daných průběhů PWM je nevyhovující.

I přes tyto nedostatky v HW provedení byl model výtahu shledán postačujícím pro náš účel demonstrace rozšířené verze vzdáleného řízení. Pro budoucí použití ve výuce by bylo však vhodné tyto nedostatky odstranit a nahradit také nepříliš profesionální a nerobustní zhotovení plošných spojů odpovědných za řízení výtahů.

3 Řízení a vzdálené řízení modelu

Řízení modelu je rozděleno do několika úrovní. Na nejnižší úrovni je model řízen programovatelným automatem (PLC), na vyšší úrovni je řešena komunikace mezi PLC, PC a jejich integrace do sítě internet a celý projekt zastřešuje rezervační on-line systém na bázi relačních databází - Lablink.

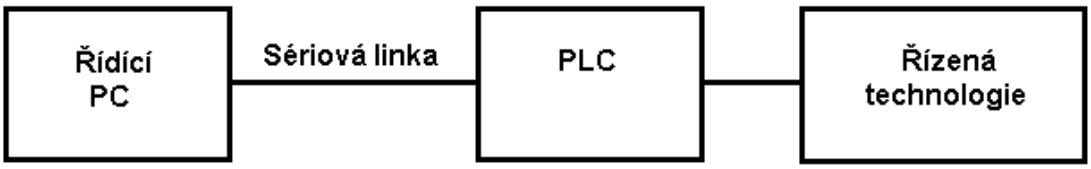
V předcházejících kapitolách jsme zatím mluvili pouze o tom, z čeho se fyzický model výtahu skládá a o vlastním řízení jsme se zmínili pouze okrajově v kapitole 2.2 *„Popis řízení systému“* a to především z pohledu teorie řízení. Komunikací mezi řídícím automatem a PC se budeme zabývat v kapitole 4.1.2 *„Komunikační kanály WinCC“* a k tomu, co vlastně je zmíněný systém Lablink a k integraci našeho modelu do tohoto on-line rezervačního systému se vrátíme v kapitole 6 *„Lablink – webové rozhranní“*. Posledním dosud opomíjeným avšak nejdůležitějším článkem v řetězci od řízeného modelu ke koncovému uživateli je průmyslový řídící automat.. Nadpis této kapitoly zní *„Řízení a vzdálené řízení modelu“*. Proto dříve než se budeme věnovat problematice využití průmyslových automatů v řízení, zamyslíme se nejdříve k jakému druhu řízení je vlastně chceme využít, co se skrývá pod pojmem *„řízení“* a co myslíme pojmem *„vzdálené řízení“*, a v neposlední řadě, co od průmyslových automatů očekáváme a jaké požadavky na řízení budeme klást.

3.1 Motivace ke vzdálenému řízení technologie

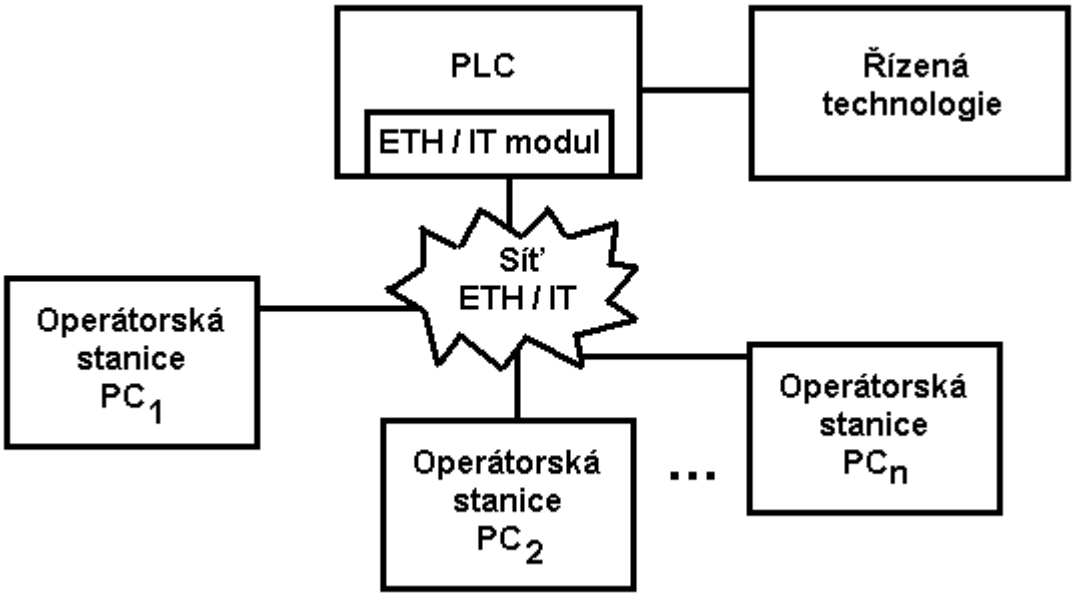
Nejjednodušší způsob, jak řídit nějakou technologii, je připojit PC přímo k PLC např. pomocí sériového rozhranní RS232. Toto rozhranní standardně obsahují všechny průmyslové automaty, avšak v praxi se používá pouze k inicializaci a počáteční parametrizaci zařízení a při provozu pouze k případné diagnostice. Jedním z hlavních důvodů, proč se komunikace po sériové lince v praxi používá jen omezeně je její malý dosah - běžně několik metrů a možnost spojení pouze typu point-to-point. Tyto nevýhody se projeví zejména v okamžiku, kdy chceme z nějakého důvodu změnit řídící program v PLC a technologie je daleko. Problém řeší vzdálené řízení například prostřednictvím lokální sítě Ethernet nebo v globálnějším rozsahu v rámci celosvětové sítě internet, kdy program lze měnit podle potřeby z libovolného vzdáleného PC. Dalším přínosem vzdáleného řízení je samozřejmě i možnost sloučit informace o dané technologii z různých PLC do jediného místa – velína.

Z této kontrolní a řídící místnosti lze pak technologii monitorovat a řídit. Zároveň lze informaci o stavu technologie distribuovat do různých monitorovacích místností. Současně jsou operátorské stanice odstíněny od nepříznivého prostředí technologie

Na následujících obrázcích vidíme rozdíl mezi komunikací po sériové lince a komunikaci v rámci sítí Ethernet nebo Internet. Na obr. 3.1.2 si můžeme všimnout, že standardní PLC musíme rozšířit o komunikační modul, který pak umožňuje sledovat a řídit technologii z libovolné operátorské stanice připojené k dané síti. Více o PLC a přídavných modulech se dozvíme v následující kapitole.



Obr. 3.1.1: Schéma sériové komunikace řídícího PC s PLC



Obr. 3.1.2: Schéma vzdálené komunikace různých operátorských stanic s PLC (ETH – Ethernet, IT - Internet)

3.2 Požadavky na vzdálené řízení

V okamžiku, kdy však umožníme přistup k technologii z lokální nebo dokonce vnější sítě, musíme například zabránit nedovolenému přístupu z vnějšku. Dále také musíme zajistit, aby do sdílených prostředků mohl v daný okamžik zasahovat maximálně jeden oprávněný uživatel. V případě našeho modelu řeší oba problémy již zmiňovaný rezervační systém Lablink, ke kterému se vrátíme v kapitole 6 *„Lablink – webové rozhranní“*.

Shrňme si nyní další obecné požadavky na vzdálenou komunikaci v řízení:

- **Časová odezva** – tj. čas, který uplyne od vzniku požadavku do vykonání skutečné akce v technologii. Délka této odezvy určuje, zda je možné řídit technologii v reálném čase.
- **Spolehlivost** – zaručení vykonání vzniklého požadavku a zobrazení správných hodnot z technologie. Tím je myšleno zamezení ztráty dat nebo jejich poškození při přenosu. To lze řešit použitím vhodných protokolů, které jsou založeny např.na potvrzování zpráv nebo takové, které zamezují kolizím na sběrnici.
- **Bezpečnost** – jak už jsme zmínili jde o zamezení zneužití vzdálené komunikace nežádoucími subjekty. Obvykle se řeší šifrováním přenosu dat, přičemž zvláště důležité je šifrování přihlašovacích údajů do systému.
- **Dostupnost** – možnost využití již hotových veřejných sítí bez jakýchkoliv omezení.

Splnění všech těchto požadavků zároveň není s dnešními dostupnými prostředky možné, neboť hlavním limitujícím faktorem je požadovaná časová odezva versus dostupnost připojení. V případe použití veřejných sítí, např. Internet, je dostupnost zajištěna téměř po celém světě, ale vesměs potřebné časové odezvy nelze vůbec zajistit.

V této diplomové práci využijeme ke vzdálenému přístupu k řídícímu PLC a k monitorování modelu vizualizaci a on-line webovou kameru. Pro vzdálenou komunikaci tedy použijeme síť Internet, která je nejrozšířenější a nejdostupnější sítí na světě. Pokud bychom měli rozebrat předcházející požadavky na vzdálené řízení, můžeme říci, že největší problém při použití komunikace přes Internet je časová odezva, která by v případě rychlejších procesů

mohla být překážkou. U našeho modelu je ale rychlost takovéhoto spojení dostatečná, protože samotné řízení obstarává PLC, do kterého se řídící algoritmus vzdáleně pouze nahraje a všechny výpočty a operace obstarává samotné PLC. U vizualizace je ovšem potřeba, aby se chování modelu dalo sledovat a řídit s co nejmenší časovou prodlevou. Ale i v tomto případě je rychlost komunikace pomocí Internetu dostačující. Více se k vybranému typu komunikace dostaneme v kapitole 4.1.2 *„Komunikační kanály WinCC“*, která se zabývá vzdáleným řízením z pohledu vizualizace, a v kapitole 6 *„Lablink – webové rozhranní“*, která řeší zabezpečení modelu před neoprávněným přístupem zvenčí. Popišme si nyní zvolený komunikační standard.

3.3 Internet

Síť internet využívá komunikačního standardu Ethernet. Ethernet je technologie, která byla vynalezena a uznána jako efektivní prostředek přenosu dat na poli kancelářské komunikace a informačních technologií, přičemž během krátké doby zaznamenal prudký vzestup v oblasti lokálních počítačových sítí na celém světě. Ethernet je vybaven vyšší úrovní komunikačního softwaru v souvislosti se standardem IEE 802.3, tj. TCP/IP (Transmission Control Protocol/Internet Protocol) pro umožnění komunikace mezi rozdílnými systémy. TCP/IP protokol nabízí vysoký stupeň spolehlivosti přenosu informace. Pro přenos dat v síti Ethernet se používá několik technologií s různými parametry, které se liší v typu přenosového média, přenosové rychlosti, délce segmentu a typu přenosu. Topologie sítě je závislá na použité přenosové technologii.

Ve standardu Ethernet mohou stanice přistupovat na sběrnici užitím metody CSMA/CD (Carrier Sense Multiple Access/Collision Detection).

- Carrier Sense: Vysílač odposlouchává provoz na sběrnici.
- Multiple Access: Na sběrnici může přistupovat několik vysílačů.
- Collision Detection: Jsou detekovány kolize.

Pracovní stanice sítě Ethernet může zahájit vysílání kdykoli, pokud zjistí, že žádná jiná pracovní stanice současně nevysílá. Pracovní stanice tak poslouchá *„éter“* a pokud je prázdný, zahájí vysílání. Odtud také název Ethernet - *„éterová síť“*. V průběhu přenosu pracovní stanice neustále naslouchá a zjišťuje, jestli mezitím nezačala vysílat stanice jiná.

Pokud dojde u vysílání ke kolizi (tedy pokud vysílají dvě stanice současně), počká každá pracovní stanice určitý náhodně zvolený časový interval a poté se o vysílání pokusí znovu. Při opakovaném vysílání může pochopitelně také dojít k další kolizi, pak nastupují různé speciální algoritmy. Jeden z takových algoritmu spočívá v čekání o exponenciálně narůstající interval.

Bohužel metoda CSMA/CD není pro průmyslové požadavky dostatečně spolehlivá, zvláště nelze vůbec zaručit časovou odezvu. Spolehlivost do jisté míry zajišťují až různé komunikační protokoly jako IP, TCP. Tabulka 3.3.1 zobrazuje pro přehlednost pozice protokolu ve vrstvách ISO-OSI.

Transportní vrstva	TCP a UDP
Síťová vrstva	IP
Spojová vrstva	Ethernet
Fyzická vrstva	

Tab. 3.3.1: Pozice protokolu ve vrstvách v rámci ISO-OSI

3.3.1 Protokol IP

Internet se skládá z celé řady hostitelských počítačů a také zařízení, kterým se říká směrovače (routers). Propojením některých počítačů vznikají sítě, přičemž některé ze sítí tvoří ostrovy. Směrovače pak propojují tyto ostrovy sítí. Každý jednotlivý hostitelský počítač i směrovač musí mít jedinečnou adresu, podle které se na toto zařízení a zpět z něj dají posílat datové pakety. Je nutné vhodně směrovat zprávy sítí, aby došly do cíle v co možná nejkratší době. To má na starosti protokol IP (Internet Protocol), který popisuje formát pro přiřazování IP adres. Současně definuje mechanismus pro nespolehlivý přenos dat, kterému se říká datagram. Datagramy IP jsou v protokolech nižších úrovní zapouzdřeny jako součást dat. Těmito protokoly jsou například Ethernet, Token-ring atd. Protokol IP verze 4 (IPv4) definuje síťové adresy jako 32 bitové číslo. Adresy IP přiděluje úřad Internet Assigned Numbers Authority (IANA). Adresy IP se dělí do celé řady kategorií, kterým se říká třídy. Avšak pro samotnou komunikaci mezi aplikacemi na dvou různých počítačích samotná IP adresa nestačí. Každá aplikace musí mít možnost identifikovat svůj protějšek, se kterým chce komunikovat, protože na jednom počítači běží obvykle souběžně několik různých aplikací.

Tyto aplikace mohou zpravidla komunikovat s různými aplikacemi na různých vzdálených počítačích nebo i na jednom stejném počítači. Protokoly UDP a TCP identifikují odesílající a přijímající aplikaci pomocí portu, zdrojová a cílová IP adresa pak identifikuje počítače, na kterých tyto aplikace běží. Datový proud mezi dvěma aplikacemi potom identifikuje množina následujících čtyř hodnot:

- zdrojová IP adresa,
- číslo zdrojového portu,
- cílová IP adresa,
- číslo cílového portu.

Některým důležitým a známým aplikacím jsou čísla portu již pevně přidělena. Tato předem přiřazená čísla portu se používají pouze na serveru. Čísla portu na klientech se alokují dynamicky. Jediný port na serveru muže mít otevřených několik proudů, protože se na něj může připojit i několik klientů. Čísla portů získávají postupně na důležitosti, protože na jejich základě funguje filtrování firewallu.

3.3.2 Protokol TCP

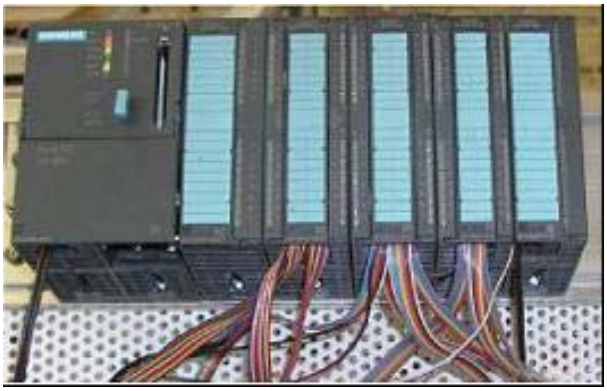
Protokol IP poskytuje poměrně nespolehlivý přenos dat. Protokol TCP oproti tomu představuje téměř již spolehlivý, bezchybný, plně duplexní kanál mezi dvěma počítači. Pro přenos se využívá služeb protokolu IP, současně však definuje určité další mechanismy, které se starají o ztracené a zdvojené datagramy IP. Protokol TCP zabezpečuje také správné složení jednotlivých datagramů IP, které dorazí mimo správné pořadí. Pro zabezpečení těchto funkcí rozděluje protokol TCP vstupní proud dat do paketu, které musí být natolik malé, aby se vešly dovnitř datagramu IP. Těmto paketům se říká segmenty. Datagramy IP se očíslují a odešlou se. Jestliže z cíle nedorazí potvrzení o jejich přijetí, odesílají se znovu. Odesílatel vždy po určitou dobu čeká a pokud neobdrží potvrzení o přijetí, odešle datagram znovu. Avšak když na sběrnici přistupuje současně velké množství počítačů (stanic), dochází k tak velkému množství kolizí, že ani protokol TCP není schopen vždy zabezpečit přenos dat, natož zaručit určité časové odezvy. Podrobnější popis fungování internetu a jednotlivých komunikačních protokolů a lze nalézt např. v [5].

3.4 SIMATIC – teoretický úvod

Pro případ, že by se tato publikace dostala do rukou nejen inženýrům v oboru řízení a automatizace, ale širšímu okruhu čtenářů, vložíme na toto místo stručný teoretický úvod do problematiky průmyslových automatů – PLC se zaměřením na PLC SIMATIC firmy SIEMENS.

3.4.1 Hardware

V naší aplikaci používáme průmyslový automat SIMATIC firmy Siemens řady S7-300 DP ve spolupráci s programovatelnými svorkami firmy WAGO. V této kapitole se budeme věnovat pouze průmyslovým automatů SIMATIC. K tomu, jakou roli hraje při řízení našeho modelu modul WAGO se vrátíme v kapitole 3.5 *„SIMATIC - praktická část“*. Firma Siemens dodává na trh několik typů PLC s různými výkony a samozřejmě i jinou cenou. Dříve se vyráběly automaty řady S5, které jsou v dnešní době nahrazeny nebo nahrazovány modernějšími automaty řady S7. Ty se dále dělí podle výkonu a nabízených možností na řady 200, 300 a 400. Nejvýkonnější je řada 400, nejméně výkonná je řada 200, tyto automaty se požívají jenom v jednoduchých aplikacích. Dále popisované skutečnosti se týkají pouze řady 300 a 400. Automaty se vyrábějí modulární nebo kompaktní technologií. Rozšířenějším typem je první z jmenovaných, jehož hlavní výhodou oproti kompaktnímu řešení je, že automat lze sestavit z různých modulů. Zákazník si tak může vybrat přesně takové HW řešení, jaké potřebuje. Základní jednotkou řídícího automatu je CPU s napájením. PLC standardně obsahuje sériové rozhranní a řady označené jako DP jsou navíc vybaveny rozhranním PROFIBUS. Na obr. 3.4.1 vidíme příklad modulárního PLC SIMATIC řady S7-300 s přídavnými moduly. Jmenujme si nejpoužívanější typy přídavných modulů, kterými lze rozšířit modulární PLC.



Obr. 3.4.1: Simatic 315-2 DP

Nabízené moduly:

- různé typy procesorů
- vstupní moduly
    - digitální
    - analogové
- výstupní moduly
    - digitální
    - analogové
- komunikační procesory
- funkční moduly, například čítače

Vstupní a výstupní karty se vyrábí s různými počty kanálů od 8 do 32 pro digitální karty a od 2 do 8 kanálů pro analogové karty. PLC použité v našem případě je sice vybaveno těmito vstupními a výstupními kartami, ale nejsou využity. Je tomu proto, že procesor neboli CPU je řady 315-2 DP, tj. je vybaven PROFIBUS konektorem. Pomocí tohoto rozhranní si naše PLC vyměňuje všechny vstupně-výstupní informace se vzdáleným modulem WAGO. Více se touto problematikou budeme zabývat v praktické části – v kapitole 3.*5 „SIMATIC - praktická část“*. Dalším typem přídavného modulu je komunikační procesor. Jeden z našich požadavků je řídit model vzdáleně, proto jsme vybavili PLC komunikačním procesorem CP 343-1 IT, který je určena pro řadu S7-300 a umožňuje komunikaci po síti Ethernet resp. Internet. Bližší popis použité CP karty bude uveden v odstavci 3.4.2.

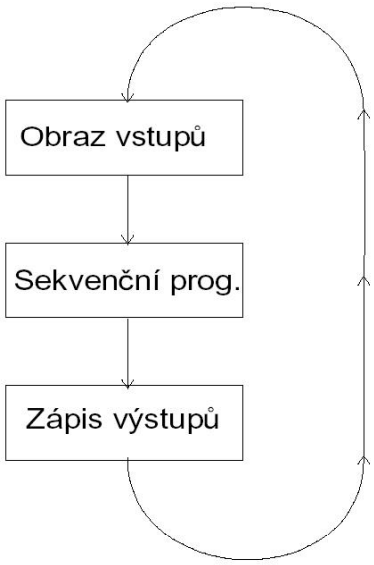
Pracovní módy PLC SIMATIC

Automat může pracovat v různých módech. K volbě módu slouží klíček, který je umístěný na přední části automatu. Možnosti jsou RUN, RUN-P, STOP a RESET. Pokud je PLC v RUN módu, automat cyklicky vykonává program. Pokud je PLC v RUN-P módu, chovaní automatu je stejné jako u RUN, můžeme ale nahrávat program do PLC. Pokud je v programu nějaká chyba, automat se přepne do STOP módu. Pokud je klíček v pozici STOP, automat nic neprovádí, lze nahrávat program. Pozice RESET umožňuje vymazat paměť PLC . V pozici RUN lze navíc vytáhnout klíček z automatu, a tím uzamknout RUN mód. Toho se využívá zejména při předávání hotového řešení k zamezení neautorizovaného zásahu do programu PLC ze strany zákazníka.

Scan cyklus PLC

Automaty PLC pracují na principu sekvenčního zpracování instrukcí. Automat neustále zpracovává takzvaný scan cyklus (obr. 3.4.2).

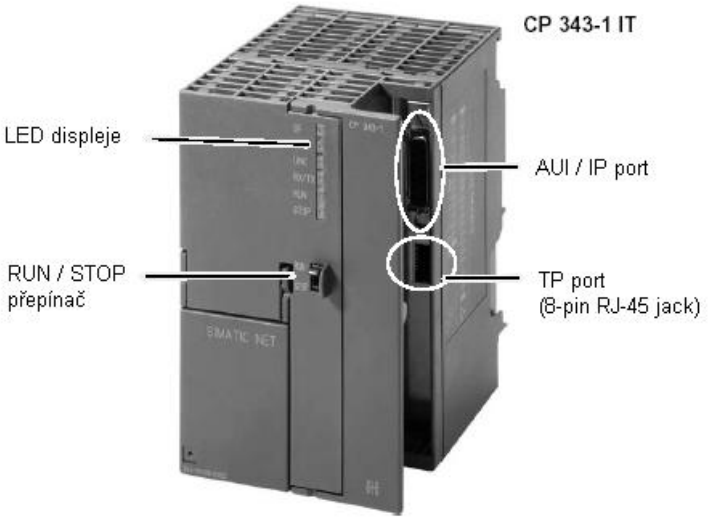
V každém cyklu se nejprve načtou fyzické vstupy a uloží se do tabulky vstupů. Provádí se to proto, aby v programu nevznikaly logické hazardy. Potom proběhne sekvence příkazů, pracuje se s obrazy vstupů, výstupy se zapisují do tabulky výstupů. Na konci cyklu se hodnoty z tabulky výstupů přepíší na fyzické výstupy. Dále se budeme zabývat pouze sekvenční častí. Na obr. 3.4.2 je pouze zjednodušená verze scan cyklu. Ve skutečnosti ještě přibývá diagnostika a komunikace.



Obr. 3.4.2: Zjednodušený scan cyklus

3.4.2 Komunikační procesor CP 343-1 IT

Internetový modul CP343-1 IT je komunikační procesor, který umožňuje vzdálenou správu dat na Simatic S7-300 na bázi Ethernet TCP/IP. Tato komunikační technologie zpřístupňuje provozní data z řízené technologie pro široký okruh uživatelů. K nahlédnutí do řídícího systému, osazeného tímto komunikačním modulem, pak stačí počítač standardu PC připojený do celosvětové informační sítě Internet. V případě poruchy řízené technologie je vyslána zpráva elektronickou poštou na libovolný počet adres (včetně mobilních telefonů) s přesnou specifikací poruch. To umožňuje operativnější servisní zásah. Tuto kartu použijeme pro vzdálenou správu modelu, a proto ji zde alespoň stručně popíšeme. Základní čášti této karty jsou označeny na obr. 3.4.3.



Obr. 3.4.3: Komunikační procesor CP 343-1 IT s popisem

3.4.2.1 Komunikační služby modulu CP343-1 IT

Na úrovni programovacího přístroje (PG) zajišťuje karta CP343-1 IT operátorské a monitorovací funkce, programování ve STEP 7 a změnu dat prostřednictvím komunikačních funkčních bloků (CFB).

V systémové komunikaci S7 realizuje rozhraní *„send/receive“* s použitím TCP nebo UDP a synchronizuje reálný čas prostřednictvím Ethernetu.

Mezi nejvýznamnější funkce informačních technologií patří funkce serveru webových stránek, zasílání zpráv elektronickou poštou, monitorovací funkce zařízení a technologických dat prostřednictvím Internetu nebo intranetu (protokolem HTML) a funkce FTP (File Transfer Protocol) pro přenos dat. V naší aplikaci využijeme například funkci webového serveru pro vytvoření vlastních webových stránek modelu a FTP protokol pro kopírování vytvořených stránek na web server CP karty.

3.4.2.2 Konfigurace karty CP

Komunikační kartu instalovanou do řídícího systému je nutné nejprve konfigurovat pomocí programu Step 7 (verze 5.0 a vyšší): nastavit adresu a specifikovat přenosové kanály, přístupová práva a hesla.

3.4.2.3 Ostatní služby a přínos modulu CP343-IT

Karta CP 343-IT se chová jako ostatní procesory TCP/IP. Protokolem S7 ji lze připojit např. k vizualizaci procesu jako v našem případě. K připojení použijeme standardní RJ-45 *„jack“* konektor (obr. 3.4.3). Pomocí funkcí *„send/receive“*, TCP/IP nebo UDP je možné řídící systémy propojit navzájem.

Tento komunikační modul není plnohodnotnou náhradou prostředků pro řízení procesů v reálném čase, ale je významným prvkem pro jednodušší přístup k informacím na manažerské i řídící úrovni.

3.4.3 Programové vybavení

Základní programové prostředí, ve kterém se realizuje celý návrh řízení, je *Simatic manager*. Po spuštění tohoto programu nás uvítá průvodce založení nového projektu. V projektu se musí vybrat typ procesoru, který máme k dispozici. Dále se v hardwarové konfiguraci vyberou veškeré moduly.

Struktura programu

OB - organizační blok

OB1 - základní blok , volá se v každém cyklu, z něho se volají další bloky.

OB100 - volá se pouze po restartu PLC

OBxx - dají se použít i další OB, např. opravy chyb, volání po určitém čase, atd.

FC - funkce

Funkce se používá jako v jiných programovacích jazycích. Lze volat s parametry. Parametry jsou vstupní, výstupní a vstupně/výstupní. Funkce se většinou napíše pro kód, který se opakuje ve více částech programu.

FB - funkční blok

Funkční blok se používá stejně jako funkce. Jeho výhoda je v možnosti použití statických proměnných. Ty jsou uložené v instančním DB a nemusí se FB předávat jako parametr. FB se může volat s různými datovými bloky.

DB - datový blok

Datové bloky slouží k ukládání proměnných. Jsou dvojího typu. Instanční, ty se přidělují k FB. Sdílené, slouží jako úložiště s různou možností použití. Především data z datových bloků budeme využívat pro vytváření vizualizace s použitím databáze – viz. kapitola 5 *„Propojení WinCC a SIMATICu pomocí databáze“*. Proto si nyní řekneme, jaká je syntaxe při adresování různých typů proměnných.

Příklad syntaxe adresace proměnných

I1.0	-	adresace vstupního bitu 0 bytu 1
Q1.0	-	adresace výstupního bitu 0 bytu 1
M10.5	-	adresace bitu 5 v paměťovém bytu 10
DB2. DW18	-	adresace datového wordu 18 v datovém bloku 2

Ukázka programovacího jazyka

Zdrojové kódy se píší v programu *Step 7*. Toto prostředí se spustí otevřením příslušných bloků (viz výše). Vývojové prostředí nám nabízí tři základní možnosti psaní programu. Uvedeme je za sebou podle toho jak se využívají a uvedeme ukázku pro stejný příklad.

STL

Zřejmě nejpoužívanější prostředí je STL (statement list). Toto prostředí se hodně podobá asembleru na PC. Na první pohled sice vypadá nepřehledně, ale zkušený programátor se v něm rychle zorientuje. Dají se v něm vytvořit nejpokročilejší techniky.

```
Network 1: Ukazka programu v STL
Logicky soucin v jazyce STL

      A     M     0.0
      A     M     0.1
      =     M     0.2
```

Obr. 3.4.4: Ukázka programu v STL

LAD

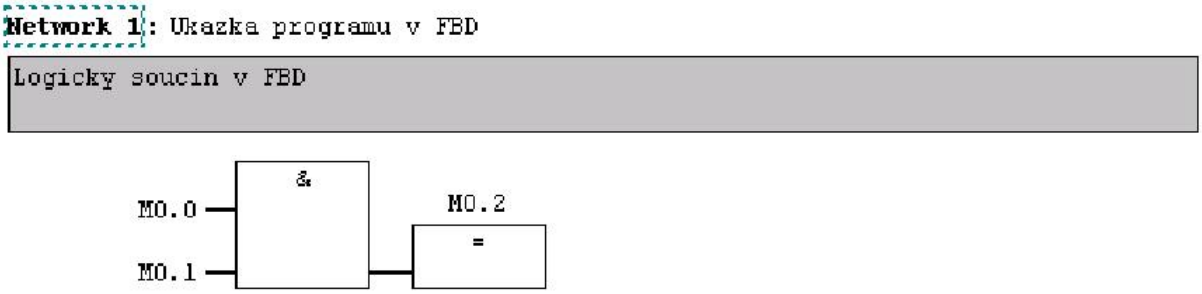
Další využívané prostředí je LAD (ladder = žebřík). Program se skládá ze žebříčkových diagramů, někdy také nazývané jako reléová schémata. Používají se zde kontakty relé. Na první pohled je velice přehledný. Složité programové konstrukce se zde již však nedají tak lehce vytvořit a program se stává nepřehledným.

```
Network 1: Ukazka programu v LAD
Logicky soucin v LAD

   M0.0        M0.1                    M0.2
 ---| |---------| |-------------------( )---|
```

Obr. 3.4.5: Ukázka programu v LAD

FBD

Další možnost je použít schémata funkčních bloků (neplést si s FB jak jsme si ho uvedli výše).



Obr. 3.4.6: Ukázka programu v FBD

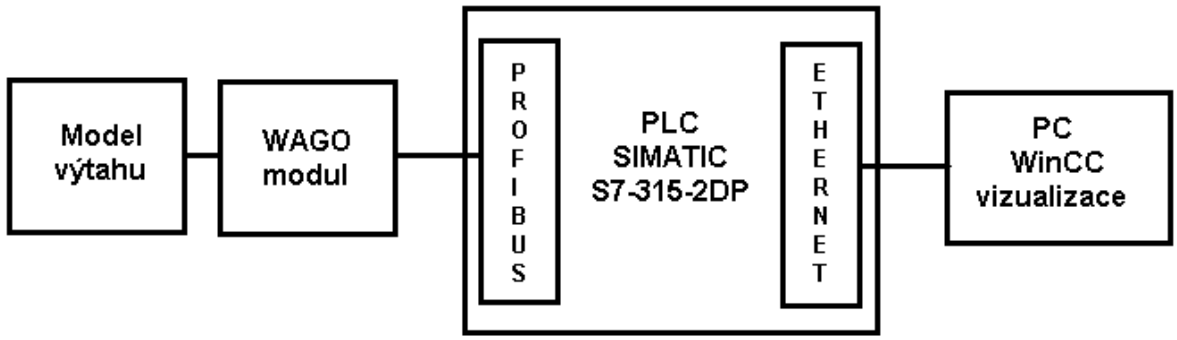
3.5 SIMATIC – praktická část

Podívejme se nejprve na to, jakou roli hrají PLC SIMATIC a WAGO modul při řízení našeho modelu. Hlavní řídící jednotkou, jak je zobrazeno na obr. 3.5.1, je průmyslový automat Simatic 315-2DP od firmy Siemens. Jedná se o automat řady S7-300 vybavený navíc PROFIBUS rozhranním. Pro komunikaci PLC s modelem využíváme místo vstupně-výstupních karet vzdálené svorky WAGO, které umožňují komunikaci po sběrnici PROFIBUS. To je uděláno proto, že model je od automatu poměrně daleko a při použití normálních vodičů by se zvyšovala pravděpodobnost chyb v přenosu signálu.

Profibus je průmyslová sběrnice, mající vlastní komunikační protokol, založená na standardu RS485. Veškerá komunikace se tedy mezi PLC a periférií uskutečňuje po dvou vodičích. Jinak bychom museli mít od PLC k technologii téměř 100 vodičů. Při psaní programu se periferní vstupy chovají jako obyčejné vstupy a proto to pro programátora nepřináší žádné nevýhody. Nevýhoda tohoto řešení je, že pokud se nám přeruší jeden ze dvou vodičů, ztratíme informaci o všech vstupech a výstupech.

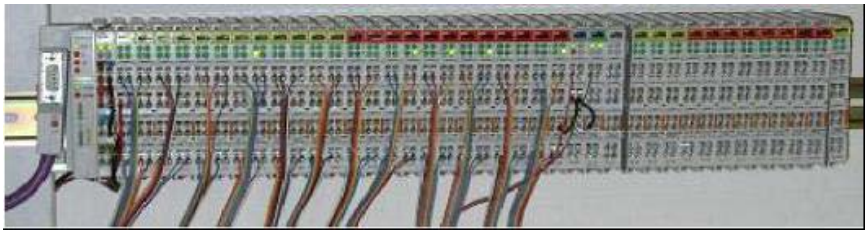
Pro komunikaci s vizualizací na řídícím PC je SIMATIC vybaven komunikační kartou CP 343-1 IT, která umožňuje zapojit automat do sítě Ethernet nebo Internet.

Obrázky na následující straně obr.3.5.2 a obr. 3.5.3 znázorňují použitý automat SIMATIC resp. modul WAGO.



Obr. 3.5.1: Simatic 315 2DP

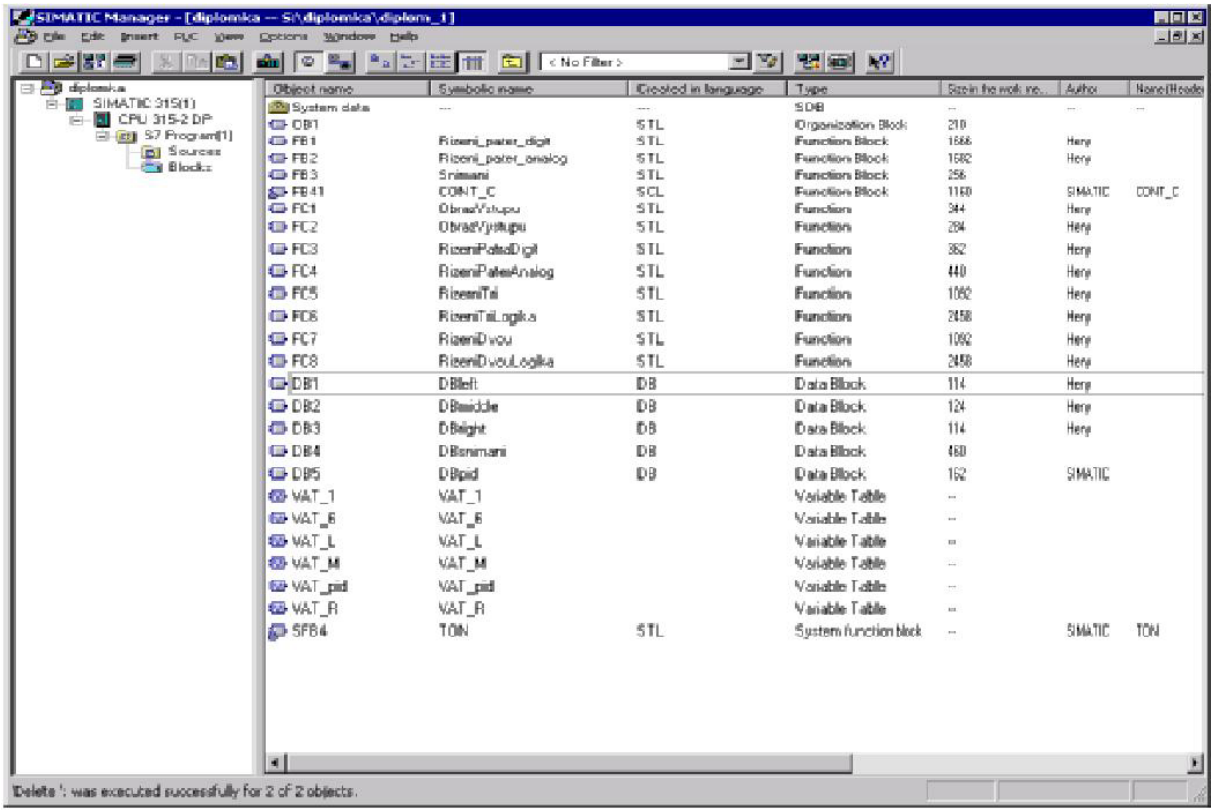
Obr. 3.5.2: Simatic S7-315 2DP s komunikační kartou CP 343-1 IT (vpravo)



Obr. 3.5.3: Periferní I/O WAGO 750-467

3.6 SIMATIC Manager

Základní aplikace vývojové prostředí pro SIMATIC programovatelné automaty je *Simatic manager*. Na následujícím obrázku je ukázka vývojového prostředí v pohledu s výpisem použitých bloků programu (obr. 3.6.1).

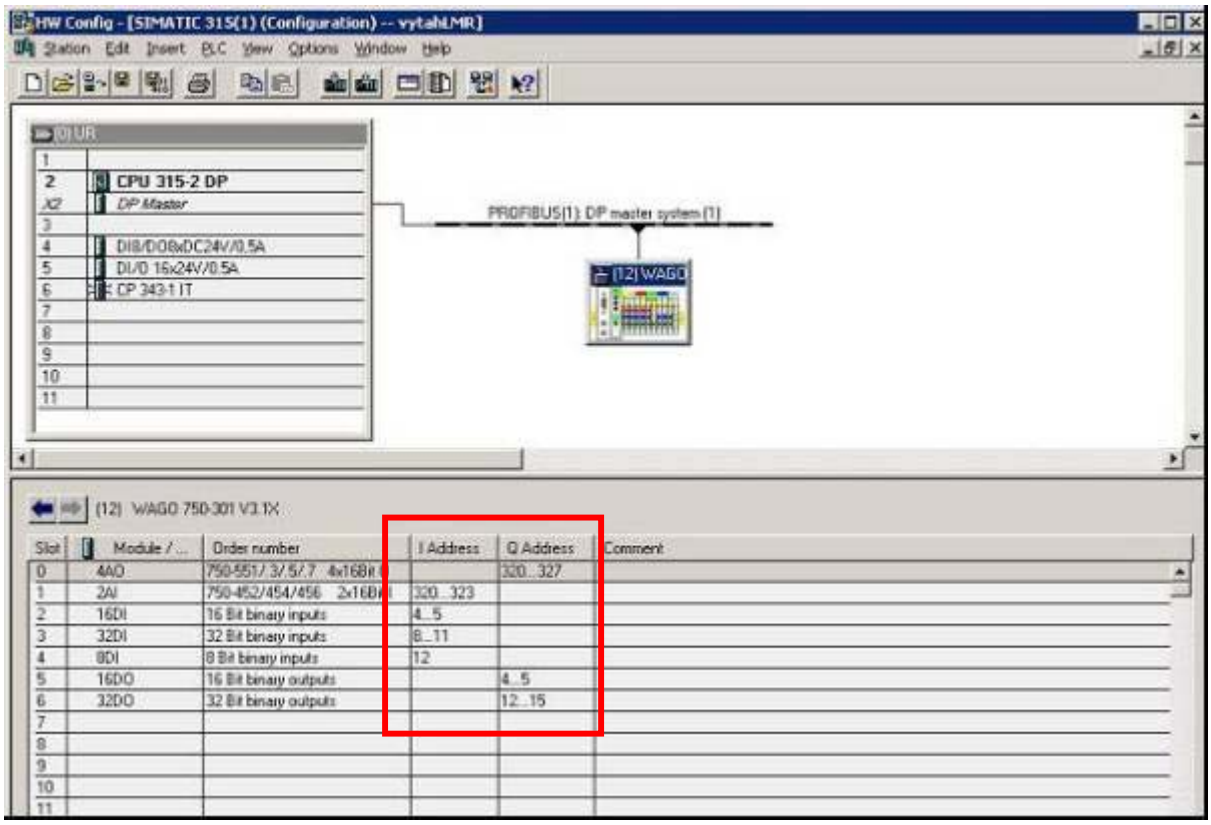


Obr. 3.6.1: Náhled na aplikaci Simatic manager

Simatic Manager je základním východiskem pro založení nového projektu, vytvoření HW konfigurace (kapitola 3.6.1) a tvorbu programu v prostředí STEP7. Při použití komplexního řešení PCS7 se stává Simatic Manager i místem pro založení WinCC aplikace a propojení programu v PLC s vizualizací. V našem případě není WinCC vizualizace součástí projektu v Simatic Manageru.

3.6.1 HW konfigurace

Po založení projektu v Simatic Manageru musíme vytvořit HW konfiguraci. Nutnost manuálního vytvoření HW konfigurace vychází z požadavků, jaké jsou na PLC kladeny. Na rozdíl od běžného PC musí být totiž průmyslové PLC schopno provozu okamžitě po zapnutí. Není tedy čas na *„bootování“* a zjišťování připojeného HW jak je tomu u PC. PLC po startu provede jen krátkou diagnostiku, zda je v paměti PLC nahraná nějaká HW konfigurace a zda odpovídá skutečnému stavu a zda je v paměti přítomen platný program instrukcí. Pokud inicializační diagnostika neobjeví žádný problém je PLC během několika vteřin připraveno k práci. Na obr. 3.6.2 vidíme HW konfiguraci použitou v našem projektu.

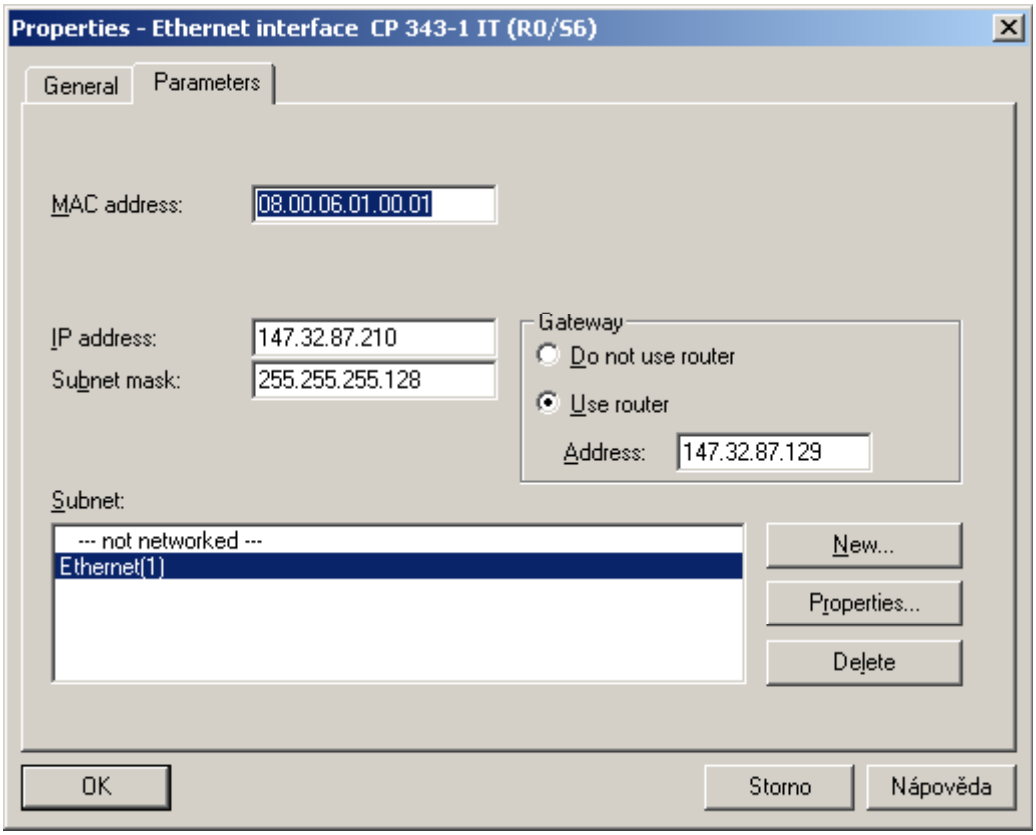


Obr. 3.6.2: HW konfigurace – adresace WAGO modulu

V levém horním okně si můžeme všimnout modulů, ze kterých se skládá naše PLC. Pořadí modulů v tabulce musí odpovídat skutečnému pořadí modulů na vodící liště automatu. Ve slotu 2 je umístěn procesor CPU 315-2 DP, dále ve slotech 4 a 5 vstupně-výstupní zásuvné moduly a konečně ve slotu 6 máme komunikační kartu CP 343-1 IT. Vpravo od horní tabulky

vidíme propojení našeho PLC s WAGO modulem přes sběrnici PROFIBUS. Tabulka v dolní polovině obrázku obsahuje adresaci vstupů a výstupů WAGO modulu. Toto namapování adres je velmi důležité (na obr. 3.6.2 vyznačeno červeně). Všechny proměnné v programu, které představují vzdálené vstupy a výstupy WAGO modulu využívají právě adres nastavených v této tabulce. Proto změnou mapování adres nebo nahráním neaktuální HW konfigurace můžeme v lepším případě docílit nefunkčnost programu, v horším případě zmatečnou funkčnost.

Pro možnost vzdáleně se připojit k našemu CPU musíme nejprve naparametrovat komunikační kartu CP. V kapitole 3.4.2 jsme zmínili možnosti CP karty, její kompletní konfigurace přesahuje rámec této diplomové práce. Co je ale pro nás důležité, je nastavení IP adresy karty a zapojení do existující sítě Ethernet jak je znázorněno na obr. 3.6.3. Ze zbylých konfiguračních možností zmiňme alespoň správu uživatelských účtů pro vzdálený přístup k nastavením CP karty - *„user-management“*, který lze nalézt pod záložkou *„General“*.



Obr. 3.6.3: HW konfigurace – adresace WAGO modulu

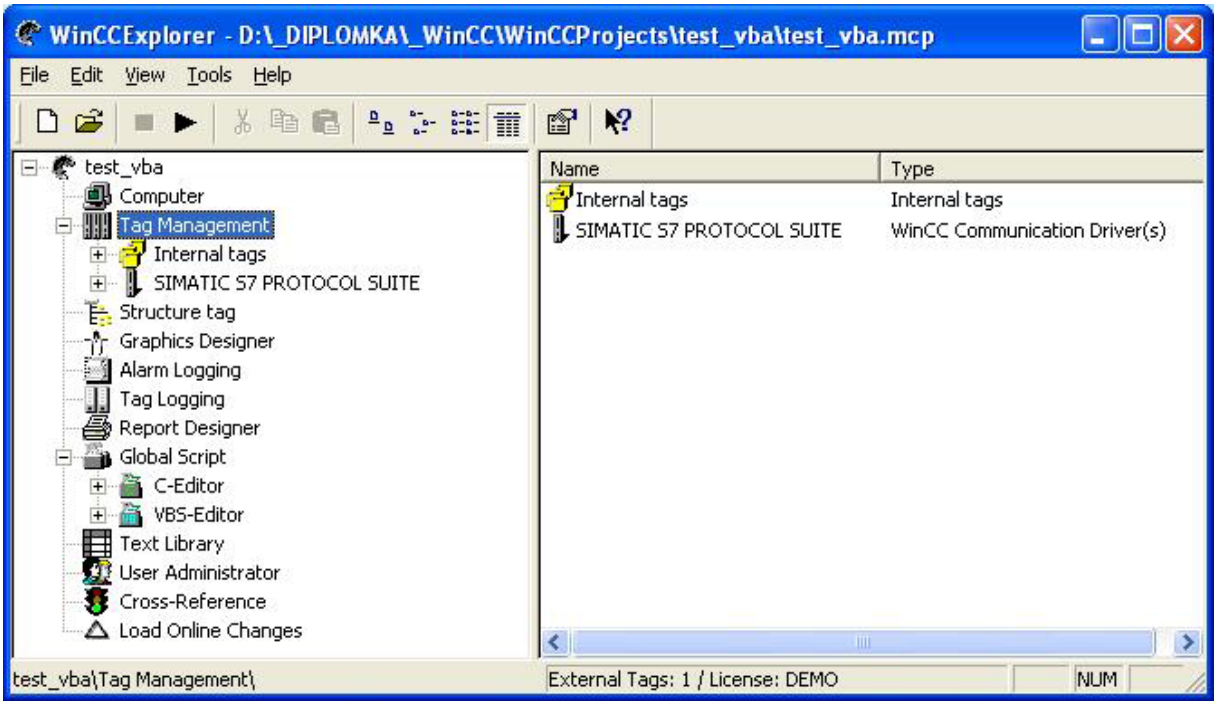
Podrobnější informace a návod jak používat Simatic Manager a programovat v prostředí STEP 7 lze nalézt například v [9].

4 Vizualizace ve WinCC v6.0

V této kapitole se zaměříme na to, jak technologii – v našem případě model výtahu – vzdáleně sledovat nebo přímo řídit pomocí vizualizace. Pro vizualizaci využíváme produkt firmy Siemens – WinCC verze 6. Jedná se o komunikaci typu *„tlustý klient“*, kdy na každém koncovém PC musí být nainstalováno prostředí WinCC. Vzhledem k omezenému počtu licencí, které máme k dispozici na tento produkt a poměrně vysoké ceně za multilicenci se budeme muset v našich školních podmínkách spokojit s jedním koncovým PC s prostředím WinCC. Toto omezení však není pro nás nijak dramatické, protože pro vizualizaci modelu výtahu jedna operátorská PC stanice v jeden časový okamžik plně postačí. Musíme však vyřešit problém sdílení této stanice mezi koncovými uživateli.

Tento problém řeší on-line rezervační systém Lablink. Více o tomto systému se můžeme dočíst v kapitole 6 – *„Lablink – webové rozhranní“*. Pro lepší představu o tom, kde se vizualizace nachází v našem schématu vzdáleného řízení (např. obr. 3.1.2, 4.2.1, 5.5.1), si však alespoň řekněme základní princip. Nejprve si koncový uživatel zarezervuje operátorskou stanici, na které běží WinCC vizualizace na určitou dobu prostřednictvím on-line rezervačního systému Lablink. Pomocí síťových nástrojů (Firewall, IP tables – více opět v kapitole 6 – *Lablink – webové rozhranní*) pro zamezení útoků z vnějšku jsou odmítnuty všechny neautorizované přístupy na tuto operátorskou stanici. Tzn., že pouze uživatel, který si model v daný čas rezervoval může využívat WinCC vizualizaci. Ostatní uživatelé mohou pouze sledovat model pomocí webové kamery. Vlastní připojení k operátorské stanici ze vzdáleného počítače probíhá pomocí nástroje pro sdílení plochy VNC server / klient. V tomto okamžiku se již tedy nacházíme na PC s WinCC vizualizací. Popišme si tedy nyní, jaké prostředky prostředí WinCC nabízí pro vzdálené řízení technologie.

4.1 WinCC Explorer

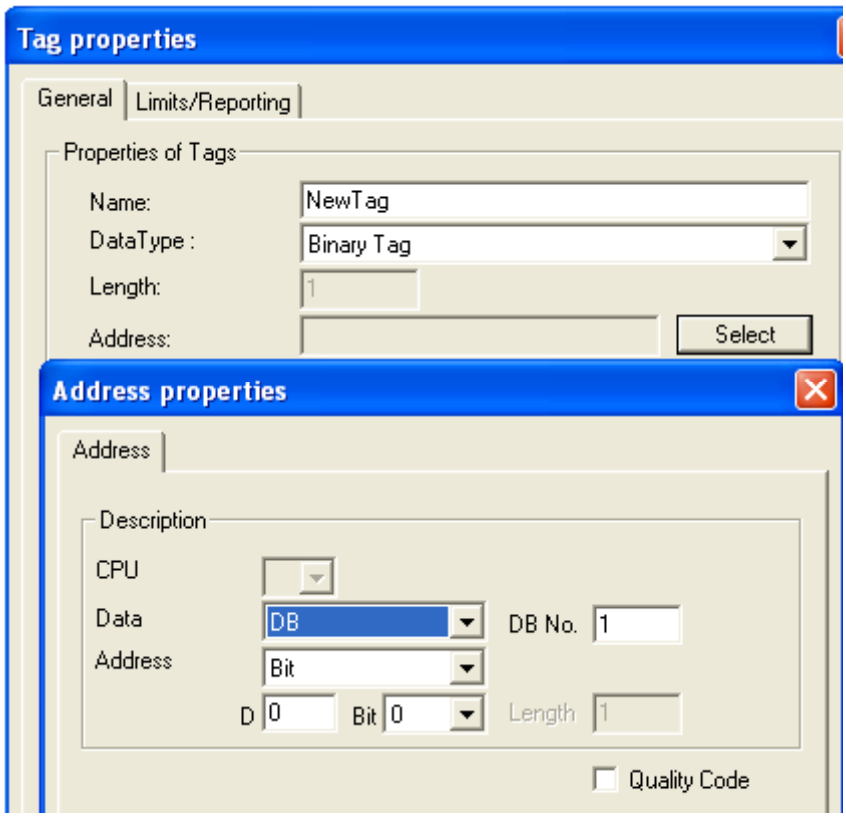


Obr. 4.1.1: WinCC Explorer

WinCCExplorer je výchozí aplikací pro tvorbu WinCC projektů obdobně jako SIMATIC Manager na procesní úrovni. Náhled úvodního okna WinCCExploreru je na obr. 4.1.1. Znovu bychom měli připomenout, že při použití softwarového balíku PCS7 v.6 může SIMATIC Manager zahrnovat i WinCC aplikaci. WinCCExplorer s odpovídajícím WinCC projektem pak můžeme spustit přímo ze SIMATIC Manageru a tento projekt může být určitým způsobem svázán s projektem na procesní úrovni. V této kapitole si popíšeme pro nás nejdůležitější části WinCC Exploreru.

4.1.1 Tag Management

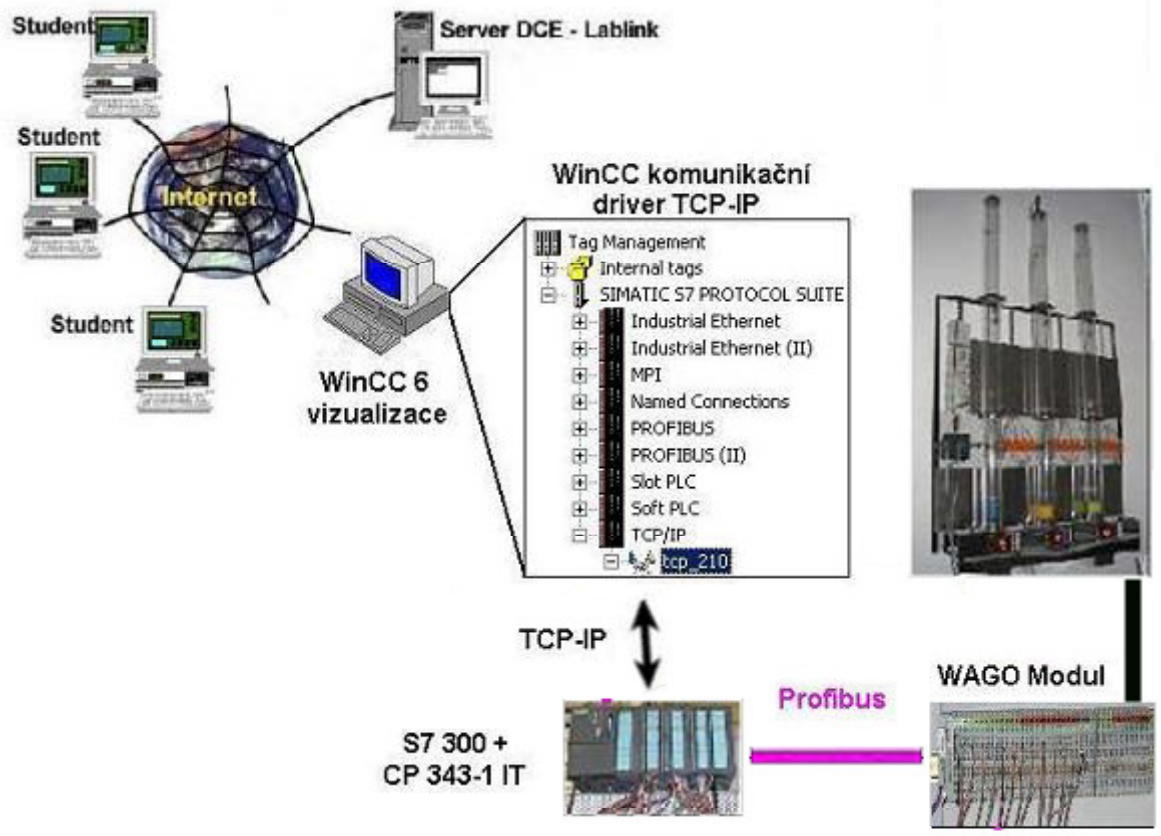
Tag Management je součástí aplikace WinCC Explorer a umožňuje definovat vizualizační proměnné, tzv. tagy. Tagy mohou být dvojího druhu. Prvním typem jsou interní proměnné, které lze definovat pro vnitřní potřebu WinCC projektu, a druhým typem jsou externí proměnné definované v rámci komunikačních kanálů. Vytvoření komunikačního kanálu mezi WinCC vizualizací a PLC bude popsáno v následující kapitole. Typy externích proměnných se shodují s proměnnými v PLC – jak už tedy bylo zmíněno v kapitole 3.4.3 proměnné rozdělujeme na vstupní, výstupní, paměťové buňky a proměnné datových bloků. Pro ilustraci nyní předpokládejme, že již máme nějakým způsobem vytvořen komunikační kanál do konkrétního PLC. Potom příklad vytvoření externí proměnné pomocí Tag Managementu popisuje obr. 4.1.2. Vytvořili jsme proměnnou *„NewTag“*, která odkazuje do datového bloku 1 na bit 0 bytu 0 daného PLC. Blíže se s externími tagy seznámíme v následující kapitole.



Obr. 4.1.2: Příklad vytvoření externí proměnné

4.1.2 Komunikační kanály WinCC

Na obr. 4.1.3 je znázorněna celá komunikační cesta mezi modelem a koncovým uživatelem. Zaměříme se zde na komunikační drivery WinCC.



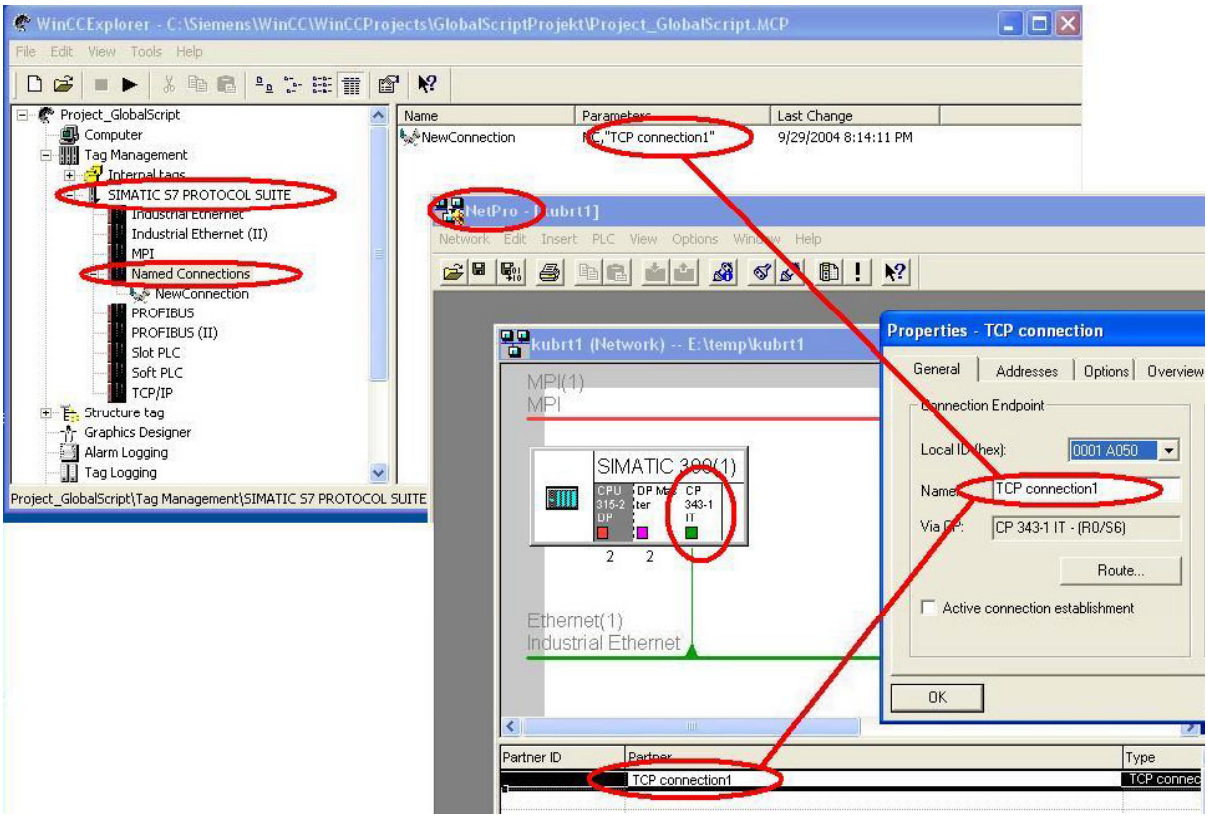
Obr. 4.1.3: Komunikační cesta od uživatele k modelu

Na straně WinCC máme k dispozici různé druhy komunikačních kanálů, které umožňují nadefinovat vizualizační proměnné. Prostřednictvím těchto kanálů pak můžeme přistupovat ke vstupům a výstupům, ale také přímo do paměťových buněk vzdáleného PLC nebo do datových bloků programu. Právě poslední z možností nás bude nejvíce zajímat. Nejdříve se ale zaměříme na volbu vhodného komunikačního kanálu.

Ke komunikaci použijeme některý z kanálů skupiny protokolů *„SIMATIC S7 PROTOCOL SUITE“*. Zde máme na výběr například protokol *„Industrial Ethernet“*, který vytváří spojení na základě MAC adresy zařízení. Dále je zde MPI spojení, které se využívá především k počáteční inicializaci zařízení, nahrání HW konfigurace a případné diagnostice

zařízení během provozu. Fyzická vrstva protokolu MPI se podobá sériovému kabelu PC samozřejmě v robustnějším průmyslovém provedení. Zvláštní skupinu protokolů tvoří tzv. *„Named Connections“*. K tomuto druhu komunikace se vrátíme později. Další často používanou formou komunikace je *„PROFIBUS“*. Naproti tomu následující dva protokoly Slot PLC a Soft PLC se příliš nepoužívají. První z nich je určen ke komunikace s PLC ve formě zásuvné karty do PC a druhý simuluje skutečné PLC na počítači. Skupinu uzavírá protokol TCP/IP.

U protokolů skupiny *„Named Connections“* se jako bod připojení používá klíčové jméno spojení (obr. 4.1.4). Nevyužíváme tedy předem nadefinovaný typ komunikace jako je např. MPI nebo TCP-IP, ale pouze odkaz na místo, kde je typ spojení definován. Tímto místem je nástroj *„Configuration Console“*, který se podobá HW konfiguraci v SIMATIC Manageru. Kromě jiného v něm můžeme nalézt část nazvanou *„Access points“*, která obsahuje všechny přístupové body pro navázání komunikace v síti SIMATIC NET. Na obr. 4.1.4 je zobrazena ukázková síť v editoru NetPro. Vidíme, že symbolické jméno spojení použité v našem případě je TCP connection1. Toto jméno bychom tedy nalezli v editoru Configuration Console a můžeme mu přiřadit libovolný typ komunikace. Pokud



Obr. 4.1.4: Využití komunikačních protokolů skupiny *„Named Connections“*

například bude náš přístupový bod *„TCP connection1“* nastaven v editoru Configuration Console na CP5611(MPI), bude se SIMATIC Manager snažit nalézt cestu, jak přistoupit na jednotlivé komponenty tohoto modulu pomocí MPI protokolu. Symbolický název spojení na obr.4.1.4 však napovídá, že pro komunikaci zvolíme typ spojení TCP-IP. V naší skutečné aplikaci však Named Connections nebudeme využívat.

Po analýze požadavků na vzdálené řízení provedené v kapitole 3.2 „*Požadavky na vzdálené řízení*“ jsme se rozhodli pro externí proměnné použít protokol TCP-IP. Problematika týkající se tohoto protokolu je velice rozsáhlá, nicméně alespoň stručný úvod do internetu a protokolů TCP-IP byl proveden v kapitole 3.3. *„Internet“*. Podrobnější popis fungování internetu a jednotlivých komunikačních protokolů a lze nalézt např. v [5] .

4.1.3 Graphics Designer

Tento editor je vlastní aplikací pro grafický návrh a vytváření dynamického chování vizualizace. Na pozadí tohoto editoru můžeme spustit Microsoft Visual Basic, který výhodně použijeme pro zautomatizování tvorby a úpravy vizualizačních obrazovek (kapitola 5.2 *„Práce s databází pomocí VBA skriptu“*).

4.1.4 Global Script

Další pro nás zajímavou součástí WinCC Exploreru je Global Script. V této sekci nalezneme dva editory – C-Editor a VBS-Editor. Oba editory umožňují vytvářet skriptovací funkce, které lze poté využít ve vizualizaci. Podporované jazyky jsou ANSI C v prvním případě resp. VBS skript v druhém případě. Oba editory také obsahují standardní předem vytvořené funkce, které zjednodušují vytváření vizualizace.

V přílohách 11.3 a 11.4 můžeme nalézt ukázky C skriptu a VBS skriptu.

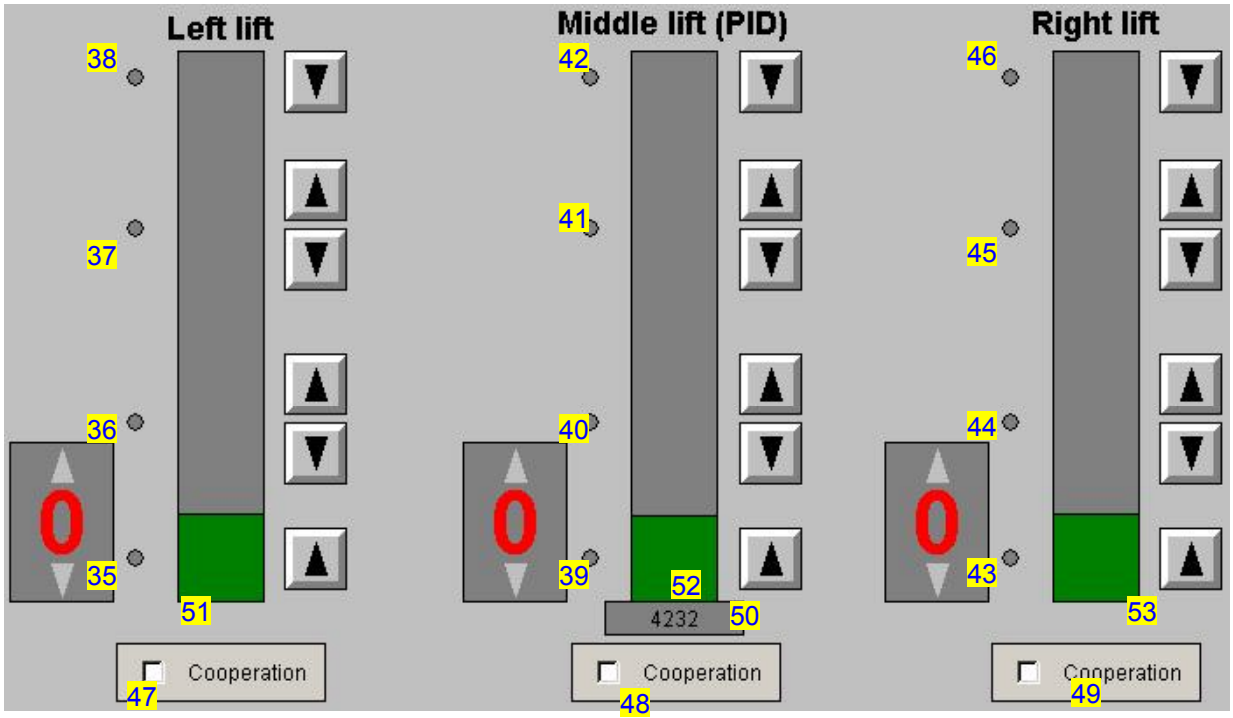
4.1.5 User Administrator

Poslední sekcí ve WinCC Exploreru, kterou využijeme v naší aplikaci je User Administrator, který nabízí možnost definice různých přístupových práv k projektu pro různé uživatele nebo skupiny uživatelů. Přístupová práva lze dokonce nastavit i jednotlivě pro různé procesní obrazovky. Např. pro uživatele, který má běžně přístup na úrovni operátora, můžeme nadefinovat procesní obrazovky, kde má administrátorské práva nebo naopak práva omezená. V naší aplikaci využijeme tuto možnost například na odlišení práv vyučujícího a studenta zasahovat do ukázkového projektu.

4.2 Popis procesních obrazovek

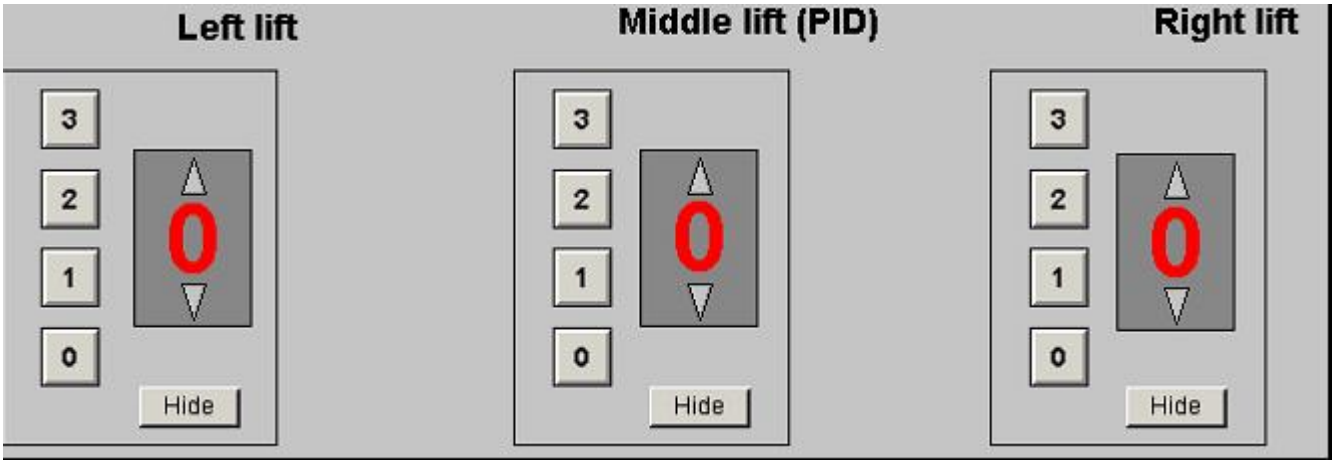
Po úspěšném vytvoření spojení mezi vizualizací a PLC pomocí komunikačního kanálu můžeme nadefinovat vizualizační proměnné. Procesní proměnné z tab. 2.2.1 a tab. 2.2.2 vytvoříme v sekci Tag_Management->SIMATIC_S7_PROTOCOL_SUITE->TCP-IP. V nejjednodušším případě se jedná zejména o vstupní nebo výstupní proměnné, popř. proměnné paměťových míst v PLC. V další kapitole se také budeme věnovat poslednímu druhu externích tagů – proměnným datového bloku, které budou pro nás nejvíce důležité. Pomocné proměnné potřebné pouze pro vizualizaci vytvoříme v sekci Tag_Management->Internal_Tags.

Na obr. 4.2.1 je znázorněna hlavní obrazovka vizualizace. Můžeme na ní vidět tři pneumatické výtahy s ovládacími tlačítky a signalizačními prvky. Číselné označení prvků odpovídá referenčním číslům v tab. 2.2.1 resp. tab. 2.2.2.



Obr. 4.2.1: Popis hlavní obrazovky vizualizace

Obr. 4.2.2 představuje ovládací panely uvnitř jednotlivých výtahů.



Obr. 4.2.2: Ukázka ovládacích panelů uvnitř výtahu

Jak jsme již říkali na začátku, samotné vytvoření vzdálené vizualizace modelu pro nás není cíl. Cílem je pro nás ukázat na takto vytvořené vizualizaci řešení potíží, které nastávají v praxi při dodatečných požadavcích zákazníka a ovlivňují procesní část, vizualizační část nebo obě části projektu.

5 Propojení WinCC a SIMATICu pomocí databáze

5.1 Motivace k použití databáze

Shrňme si nyní, jakých cílů jsme již dosáhli s naší demonstrační aplikací. Máme vytvořenou vizualizaci ve WinCC na vzdáleném serveru. Pro komunikaci mezi modelem a vizualizací jsme úspěšně vytvořilï komunikační kanál prostřednictvím protokolu TCP-IP. Vizualizace má finální vzhled a dynamické prvky jsou připojeny na dané adresy vstupů, výstupů, paměťových nebo datových míst v PLC. Jinými slovy aplikace je připravena pro předání zákazníkovi. Představme si však nyní, a téměř vždy se to v praxi stává, že zákazník má k předloženému řešení výhrady či drobné dodatečné požadavky, které nebyly nebo z principu ani nemohly být specifikovány předem. Tyto připomínky se mohou týkat jak požadované funkčnosti na procesní úrovni tak výsledného vzhledu vizualizace. Splnění prvního z požadavků sebou může přinést změnu adresace v programu, a tím nefunkčnost dosavadní vizualizace, která proměnné z těchto pevných adres čte či na ně zapisuje. Ani splnění druhého z požadavků nemusí být jednoduché. Vezměme pro představu přání zákazníka na jiný tvar či barvu zobrazovaných ventilů, kterých se může v praxi na procesních obrazovkách vyskytovat stovky. V obou výše uvedených příkladech by ruční úpravy vizualizace byly neúnosně nákladné především z časového hlediska. Proto bylo hledáno řešení, jak co nejefektivněji tyto úpravy zautomatizovat. Při analýze možných řešení bylo kladeno za podmínku, aby se změny prováděli jen na jednom místě a aby tyto změny odrážely jak změnu kódu v PLC tak požadovanou změnu vizualizace.

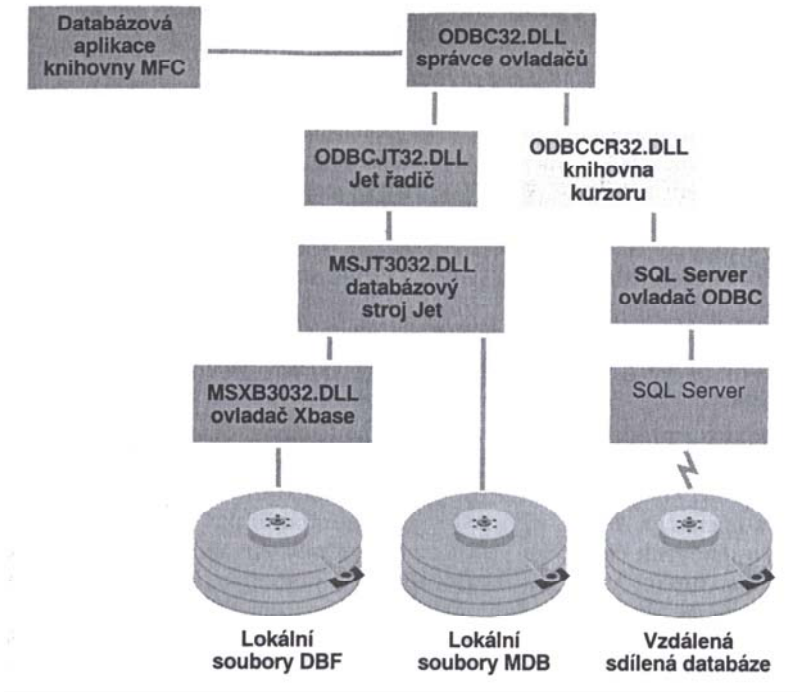
Jako možné řešení bylo navrženo použití externí databáze, která by obsahovala potřebné informace jak pro procesní stranu tak pro vizualizaci. K tomuto účelu byla vytvořena struktura tabulek (příloha obr.11.6.1) v Microsoft SQL Serveru 2000. Důvod pro vybrání právě tohoto databázového serveru byl ten, že MSQL Serveru 2000 je standardní součástí WinCC, primárně používaný pro archivaci procesních dat. Jak tedy můžeme vidět na obr. 11.6.1, z procesních informací si v databázi uchováváme fyzickou adresu parametru, v případě datových proměnných navíc číslo příslušného datového a funkčního bloku, dále typ parametru atd. Pro vizualizaci používáme jméno cílové procesní obrazovky pro umístění prvku, jméno tagu, typ vizualizačního prvku apod.

5.2 Práce s databází pomocí VBA skriptu

Parametry, které lze uložit do databáze mohou být jak vstupní nebo výstupní proměnné, pomocné paměťové proměnné, ale především proměnné datových bloků. Pro zachování konzistence dat v databázi bylo nutné zajistit přístup k datům pouze z jednoho místa. Vzhledem k těsnému propojení dat v databázi především s vizualizací, jak bude popsáno dále, bylo za vstupní bod do databáze vybráno vizualizační prostředí WinCC. Prostředí Microsoft Visual Basic (VBA) běží na pozadí Graphics Designeru a umožňuje nám v grafickém návrhu používat makra, která zautomatizují některé činnosti při vytváření a změnách vizualizačních obrazovek. V našem případě jsou informace o dynamické vlastnostech a vzhledu procesních obrazovek uloženy v MSQL databázi.

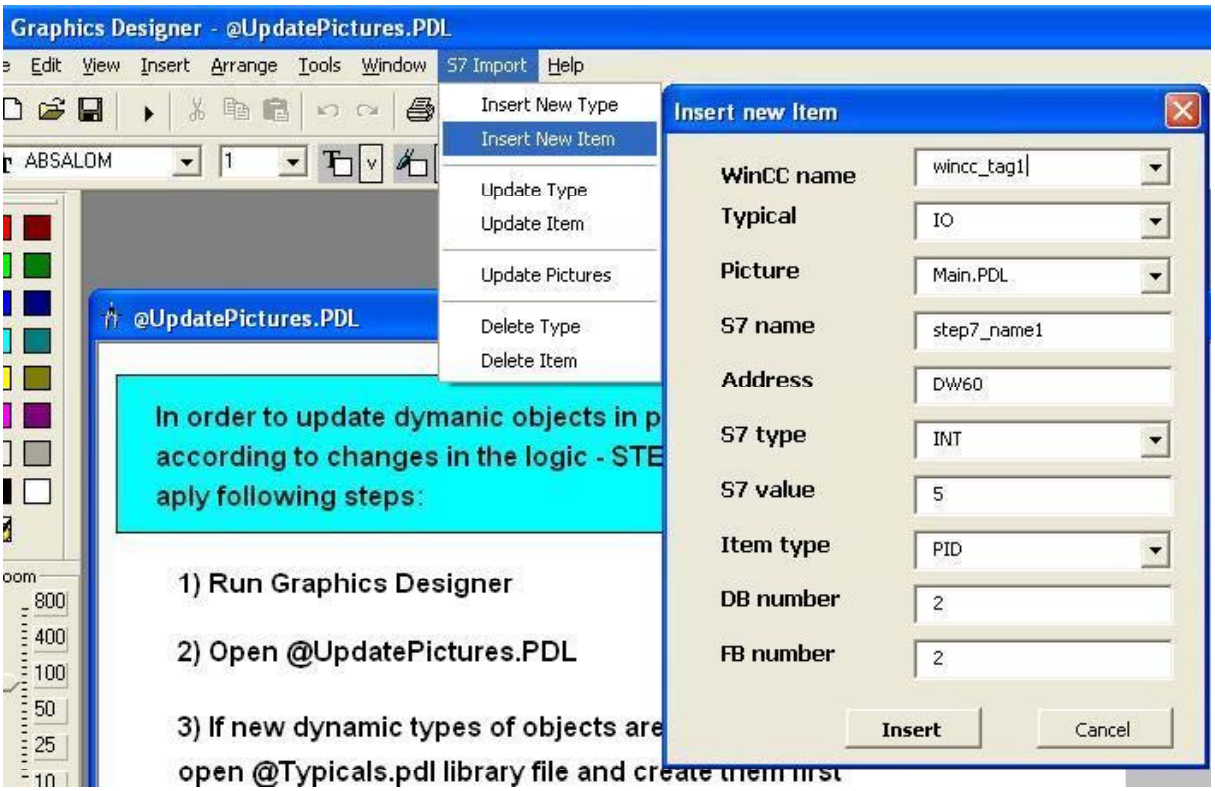
Microsoft ADO Data control

K připojení k databázi z prostředí VBA používáme ActiveX prvek ADO control (Microsoft Data Access Objects). Jedná se o nadstavbu ODBC (Open Database Connectivity) driveru. Jak znázorňuje obr. 5.2.1, ODBC umožňuje připojení k libovolnému zdroji dat – v našem případě je to MSQL Server 2000.



Obr. 5.2.1: 32bitová architektura ODBC

Aby náš projekt ve vizualizaci byl připraven na automatické změny v budoucnu, což bylo naším cílem, musíme při jeho vytváření vyplnit inicializační vlastnosti parametrů v databázi. Vzhledem k tomu, že na procesní úrovni nemáme k datům přiřazeny žádné informace o tom, jak se mají zobrazovat ve vizualizaci, musíme první inicializaci databáze provést ručně. Struktura tabulek databáze je uvedena v příloze na obr. 11.6.1. Následující obrázek (obr. 5.2.2) ilustruje vkládání nového parametru do databáze. Po otevření obrazovky



Obr. 5.2.2: Ukázka vkládání informací o parametrech do databáze

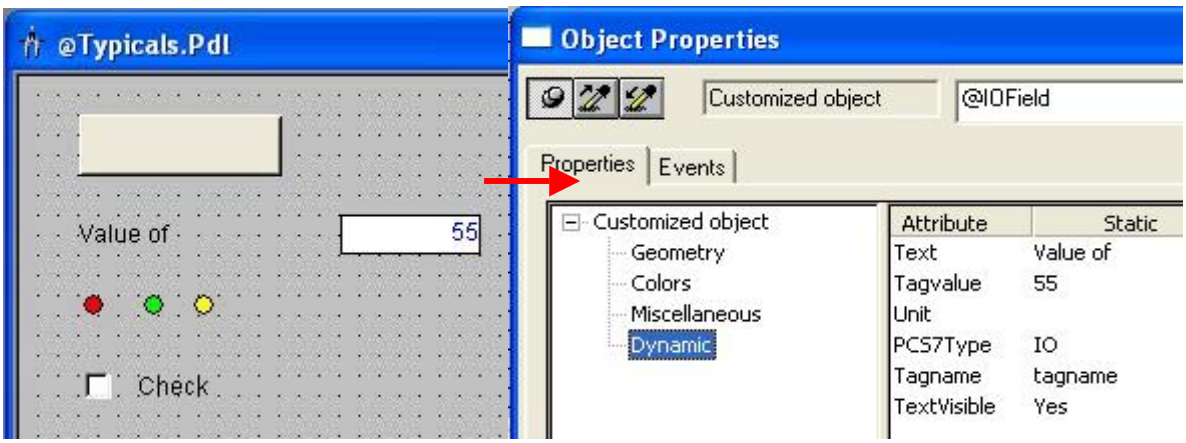
„@UpdatePictures.PDL“ se do menu Graphics Designeru přidá položka S7 import. V pravé části obrázku vidíme, že údaje o prvku v databázi obsahují jak informace pro procesní část (STEP7), tak informace potřebné pro vizualizaci těchto dat.

Proberme si jednotlivé údaje postupně. Prvním je WinCC name, což je jméno vizualizační proměnné, která bude automaticky vygenerována v Tag Managementu po vložení parametru do databáze. Její typ závisí na hodnotě S7 type, v našem případě INT – integer hodnota. Dalším údajem je Typical, který udává, jak bude tento parametr reprezentován na procesní obrazovce – více se tímto budeme zabývat v následujícím odstavci. Cílovou

obrazovku, kam bude Typical umístěn vybereme v listboxu Picture. S7 name je jméno parametru ve STEP 7 a může se lišit od pojmenování stejného parametru ve WinCC. V poli Address zadáme adresu parametru v PLC, syntaxe pro psaní adres byla popsána v kapitole 3.4.3 *„Programové vybavení“*. Hodnotu proměnné zadáme v poli S7 value a typ prvku z pohledu logického členění parametrů vybereme v listboxu Item Type. Pokud požadovaný typ neexistuje, je nutné ho nejprve přidat v menu S7_Import->Insert_New_Type. Pokud se jedná o proměnné datových bloků, vyplníme příslušné čísla DB a FB, jinak necháme hodnoty nulové. Pomocí menu Delete Type a Delete Item vymažeme typ resp. parametr z databáze. Při mazání typu budou smazány všechny parametry daného typu, proto příkaz mazání musíme několikrát kliknout na tlačítko OK, že jsme si opravdu jisti provedením operace. Po inicializaci databáze jsme připraveni případné změny na procesní úrovni velmi rychle zanést do připravené databáze aktualizací parametrů z menu Update Type a Update Item. Tím se zároveň i automaticky znovu vygenerují vizualizační proměnné a není potřeba žádný manuální zásah do již hotových procesních obrazovek. Nejčastějším příkladem je změna adresy již používaného parametru, např. přemapování vstupů a výstupů. Ještě bychom měli znovu zdůraznit, že jediný legální způsob, jak změnit údaje v databázi je právě pomocí VBA skriptu, resp. položek menu, které VBA skript do Graphics Designeru přidá. Jedině odtud totiž můžeme zaručit konzistenci dat v databázi. Vstupní formuláře pro zadávání údajů jsou totiž vybaveny vstupní kontrolou zadávaných údajů a nedovolí uživateli např. přidat již existující parametr, zadat neplatnou hodnotu parametru - např. údaji typu BOOL reálné číslo apod. Poslední položkou menu Update Pictures se budeme zabývat v následující kapitole 5.2.1.

5.2.1 Automatická aktualizace procesních obrazovek

V úvodním zamyšlení nad motivací této diplomové práce jsme se zmínili o problému hromadné aktualizace vizualizačních obrazovek při dodatečných požadavcích zákazníka na změnu vzhledu apod. K řešení tohoto problému použijeme kromě databáze ještě knihovnu standardních vizualizačních komponent.



Obr. 5.2.3: Knihovna vizualizačních prvků včetně jejich vnitřní struktury

Na obr. 5.2.3 je ukázka několika základních vizualizačních prvků v knihovně @Typicals.PDL – tlačítko, hodnota s popiskou, barevné LED diody a checkbox. Nyní se můžeme vrátit k obr. 5.2.2 k údaji Typical. V tomto formuláři jsou na výběr všechny prvky vytvořené v knihovně @Typicals.PDL a jsou dynamicky načítány z aktuální verze knihovny. Vidíme, že pro náš parametr jsme zvolili grafickou reprezentaci IO, což pohledem do knihovny zjistíme, že znamená hodnota s popiskou (IOField).

Princip využití knihovny prvků

Mějme prázdný WinCC projekt s několika prázdnými procesními obrazovkami. Po vyplnění údajů ve formuláři na obr. 5.2.1 se kromě zanesení daného parametru do databáze automaticky provedou následující operace:

- Vytvoří vizualizační proměnná wincc_tag1 typu integer odkazující na datový word DW60 v datovém bloku 2. Tato proměnná se vytvoří v sekci Tag Management, v sekci externích proměnných používající komunikační driver TCP-IP naparametrizovaný pro komunikaci s naším PLC.

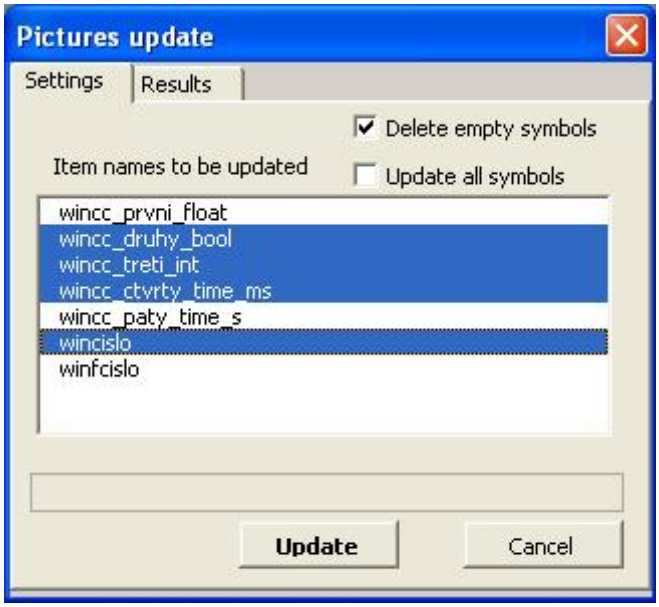
Nyní se můžeme rozhodnout, zda nejprve opravíme všechny údaje v databázi podle nových požadavků a teprve potom aktualizujeme procesní obrazovky nebo můžeme aktualizovat každý parametr zvlášť nebo libovolný počet parametrů současně podle vlastního výběru. Po vybrání položky Update Pictures z menu S7 import se objeví okno na obr. 5.2.4. Po stisknutí tlačítka Update se automaticky provedou následující operace:

- Pro každý vybraný parametr z databáze se nalezne odpovídající grafická reprezentace v knihovně @Typicals.PDL podle hodnoty *„Typical“* v databázi. Pokud necháme zaškrtlou volbu Update all symbols aktualizují se všechny parametry z databáze.
- Program postupně projde všechny doposud existující vizualizační obrazovky a hledá symboly s odpovídajícím klíčovým jménem – tím se zde myslí, zda údaj v databázi WinCC_name odpovídá vlastnosti Tagname daného symbolu (obr. 5.2.3 vpravo). Takto se projdou všechny dynamické prvky na všech obrazovkách a aktualizují se podle nových údajů v databázi. Pokud se na dané obrazovce odpovídající symbol nenalezne a je to zároveň cílová obrazovka parametru – hodnota Picture je shodná se jménem aktuálně otevřené obrazovky, a to je náš případ, protože jsme předpokládali prázdný projekt, nakopíruje se daný symbol z knihovny na cílovou obrazovku.
- Takto nakopírovaný symbol z knihovny není funkční, protože není dynamicky napojen na žádnou proměnnou. Proto se podle údajů z databáze musí naparametrizovat. Podívejme se na obr. 5.2.3 vpravo. Údaj *„Text“* se automaticky doplní o jméno tagu, tedy *„value of tagname“, „Tagvalue“* se nastaví podle hodnoty v S7_value databázi a *„Tagname“* se změní na hodnotu ve sloupci WinCC_name.

Pokud si tedy zrekapitulujeme, co je potřeba udělat k vyřešení např. problému s odlišnou barvou ventilu zmíněného v úvodu diplomové práce, dá se to shrnout do dvou jednoduchých kroků. Otevřeme knihovnu @Typicals.PDL a upravíme barvu nebo případně další vlastnosti daného ventilu a poté spustíme Update Pictures pro všechny prvky, které máme v úmyslu změnit. Vidíme, že řešení je poměrně efektivní a hlavně nezávislé na celkovém počtu vizualizačních obrazovek.

Pro jednoduchost jsme opomněli ještě jednu věc při aktualizaci obrazovek. Možná jste si všimli, že symboly v knihovně začínají znakem „@“. Ve skutečnosti symboly, které se

automaticky aktualizují na obrazovkách musejí také začínat tímto znakem. Ostatní symboly jsou ignorovány. Zaškrtnutím volby *„Delete empty symbols“* ve formuláři na obr. 5.2.4 se vymažou všechny symboly na všech obrazovkách, které neexistují v knihovně @Typicals.PDL. Pravidlo se znakem „@“ zde platí také.



Obr. 5.2.4: Automatická aktualizace procesních obrazovek

Ukázku kódu VBA skriptu lze nalézt v příloze 11.5.

5.3 Přístup do databáze v Runtime módu

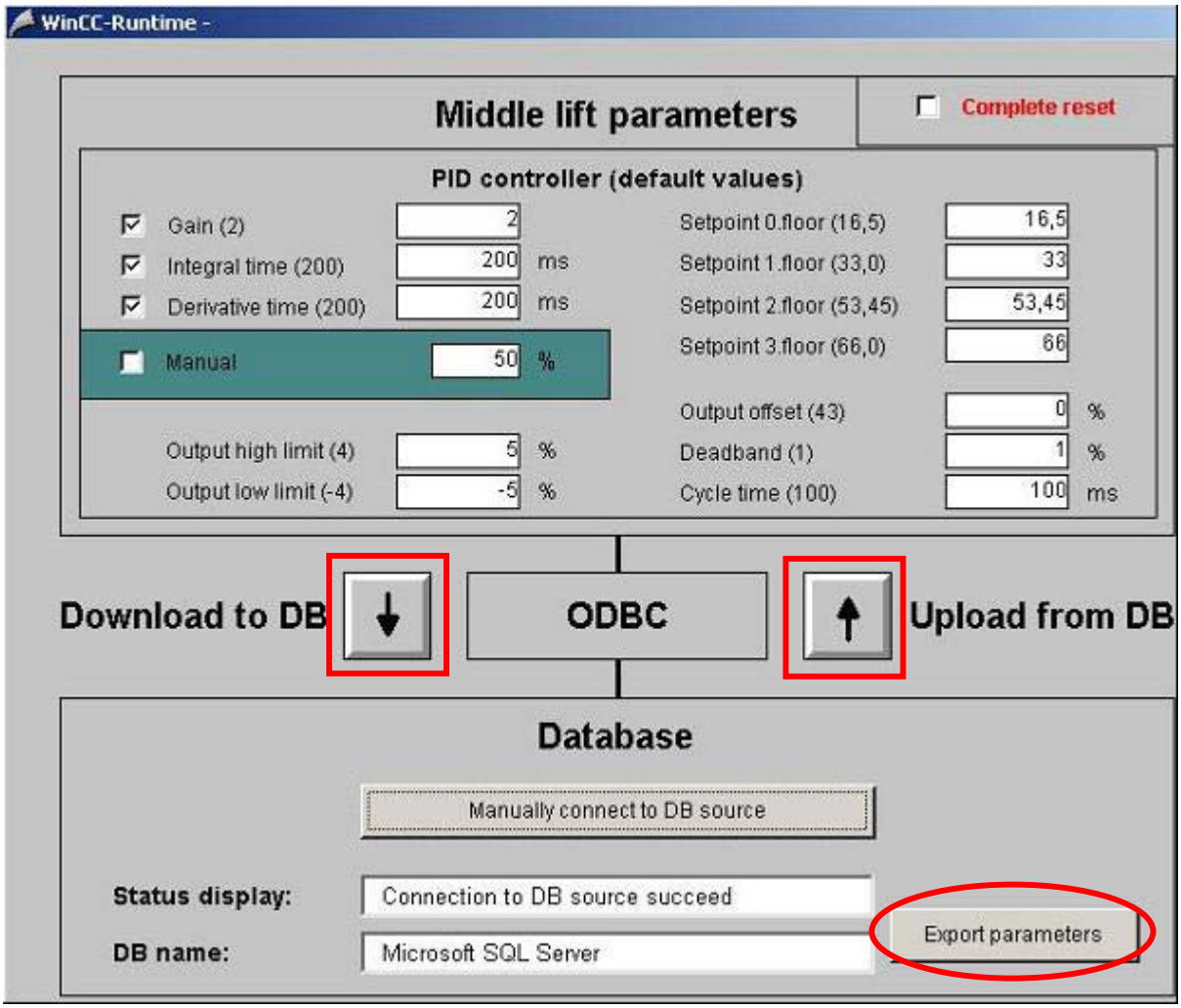
Doposud jsme se pohybovali v oblasti grafického návrhu vizualizace, pro který lze použít VBA skript. Vlastní vizualizace však běží v runtime módu, pro který VBA není určen a pokud chceme přistupovat do databáze i v tomto případě, musíme použít jiné prostředky. V runtime módu máme k dispozici nástroj pro skriptování – Global skript, který byl již zmíněn v kapitol 4.1.4. Zde můžeme používat buď ANSI C jazyk nebo VBS skript. Global skript je však určen pouze na jednodušší a časově nenáročné skripty. Prvním důvodem k tomu je to, že v praxi běží vizualizace v runtime módu nepřetržitě řádově měsíce a sebemenší chyba například v manipulaci s pamětí, řekněme při alokaci proměnných nebo chyba v nevhodně zvoleném volání funkce ve složitém a nepřehledném kódu vyvolá dříve nebo později zahlcení a kolaps aplikace. Druhým důvodem je velký počet dynamických skriptů, které se v runtime módu musejí vykonávat současně, a proto je nemyslitelné, aby byly příliš časově náročné. Naproti tomu VBA makra spouštíme pouze při potřebě nějakým způsobem aktualizovat vizualizaci, odděleně od runtime módu a tolik nezáleží na tom, za jak dlouho se vykonají. Běžně skript podobný našemu skriptu na aktualizaci procesních obrazovek by se při počtu obrazovek řádu stovek vykonával několik hodin. I přesto to není považováno za velký problém, protože VBA většinou automatizuje nějakou činnost v prostředí Graphics Designeru, která by manuálně trvala neporovnatelně déle.

Knihovna odbc32.dll

Global skript neposkytuje takové programátorské pohodlí jako VBA, a proto byl ke komunikaci s databází použit ODBC driver přímo na úrovni dll knihovny. Ukázka implementace ODBC driveru v C skriptu je v příloze 11.3.

Z odlišné orientace VBA skriptu a Global skriptu vyplývají i jejich různé možnosti. Zatímco v prostředí VBA můžeme vytvářet vizualizační proměnné, vytvářet uživatelské menu v Graphics Designeru nebo přistupovat na objekty jednotlivých vizualizačních obrazovek, umožňuje Global skript pouze čtení tagů nebo jejich zápis. To zase naopak neumožňuje VBA skript.

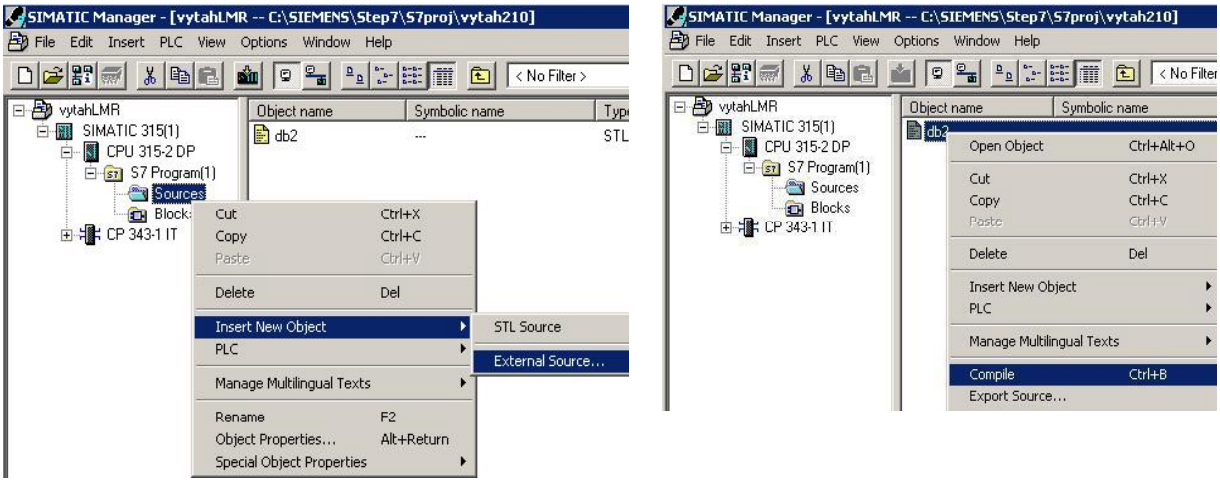
S použitím C skriptu se nám podařilo implementovat ODBC driver s využitím dll knihovny odbc32.dll. Tím jsme docílili propojení runtimu přímo s MSQL databází. Toho můžeme s výhodou využít při práci proměnnými datových bloků. Jak je znázorněno na obr. 5.3.1, PID regulátor, který řídí prostřední výtah, zahrnuje mnoho parametrů, které je výhodné uložit do databáze. Parametry lze jak nahrát do databáze, tak načíst naposledy uložené parametry z databáze. Nemůžeme sice vytvářet nové proměnné jako ve VBA, ale v případě datových bloků je nejvíce důležitá vlastní hodnota parametru. Díky použití komunikačního kanálu se hodnota změněná v runtimu okamžitě projeví v datovém bloku PLC. Nicméně můžeme dokonce datový blok vygenerovat celý se všemi parametry z údajů z databáze a pak ho nahrát on-line do PLC jako celek. To je ukázáno v následující kapitole.



Obr. 5.3.1: Propojení vizualizace s databází v runtime módu

5.4 Export parametrů do STEP 7

V této kapitole ukážeme dva způsoby jak vygenerovat datové bloky z údajů v naší MSQL databázi. Výsledkem obou exportů jsou datové bloky, resp. jejich zdrojové kódy, ze kterých lze kompilací v prostředí SIMATIC Manageru vygenerovat odpovídající datové bloky a ty pak nahrát do PLC (obr. 5.4.1).



Obr. 5.4.1: Vložení externího AWL zdrojového souboru do STEP 7 (vlevo) a vygenerování datového bloku kompilací zdroje (vpravo)

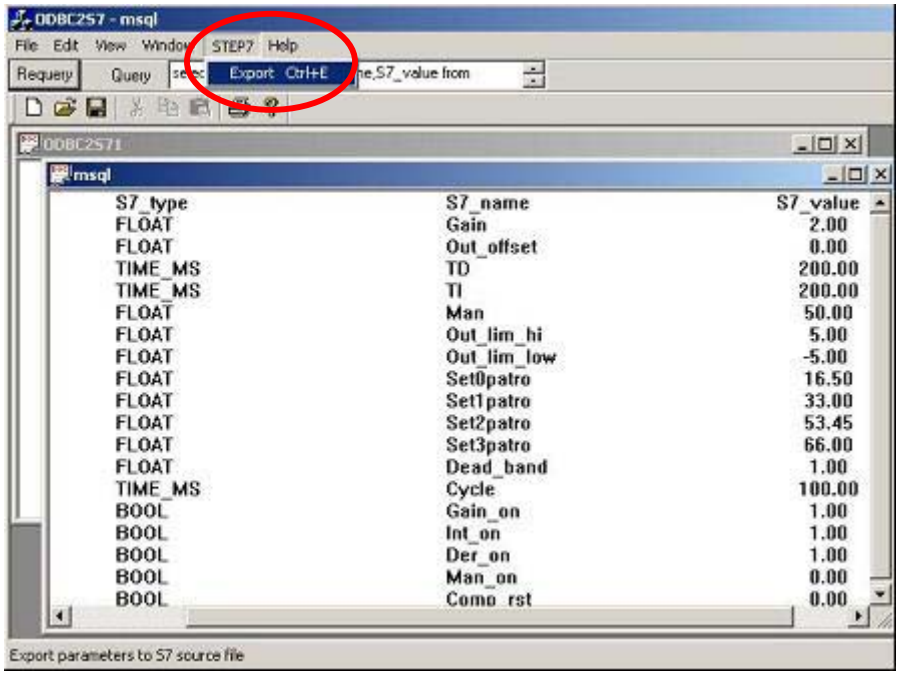
Teorií funkčních bloků a jejich spoluprací s datovými bloky jsme se zabývali již v kapitole 3.4.3 *„Programové vybavení“*. Snad jen znovu připomeňme, že hlavním důvodem, proč v kódu využíváme datové bloky je jejich univerzální využití ve spolupráci s funkčními bloky na způsob volání stejného kódu s různými parametry. Analogií v klasickém programování je volání stejné funkce s různými parametry. Více o použití datových bloků se lze dočíst například v [9].

5.4.1 Export parametrů v Runtime módu

Pro export parametrů se používá stejný přístup do databáze jako při uploadu čí downloadu parametrů v runtime módu – ODBC driver (obr. 5.3.1 – tlačítko Export parameters). Rozdíl je pouze ve zpracování dat. Při uploadu parametrů z databáze se načtené hodnoty použijí pro přepsání vizualizačních proměnných, naproti tomu při exportu parametrů se načtená data využijí pro vygenerování externího zdrojového AWL souboru pro PLC.

5.4.2 Export parametrů z externí aplikace

Pro případné budoucí širší využití generování datových bloků z údajů v databázi byla vytvořena univerzální externí aplikace – ODBC2S7.EXE. Tato aplikace využívá DAO ActiveX komponentu (Microsoft Data Access Objects) pro připojení k libovolnému datovému zdroji. Aplikace umožňuje otevřít více datových zdrojů najednou, pro ulehčení práce lze datový zdroj otevřít ze seznamu naposledy použitých, pomocí SQL příkazového řádku lze zadávat SQL dotazy na právě otevřený datový zdroj a má stavový řádek s kontextovou nápovědou. Po otevření datového zdroje je možné v menu STEP7->Export vygenerovat zdrojové AWL soubory pro PLC SIMATIC (obr. 5.4.2 vyznačeno červeně). Jejich počet není omezen, a vychází pouze z toho, kolik datových resp. funkčních bloků je v databázi zastoupeno. Protože je aplikace navržena univerzálně, jediný požadavek na použitý datový zdroj je, aby obsahoval veškerou informaci nutnou k vygenerování zdrojových souborů. Tzn., aby obsahoval sloupečky s údaji o čísle datového bloku, čísle funkčního bloku, jménu proměnné ve STEP7, typu proměnné ve STEP7 a hodnotě proměnné. Na pořadí ani pojmenování sloupců v databázi nezáleží, aplikace využívá k hledání potřebných údajů zadané jména sloupců. Dále v aplikaci vyplníme popisné informace o generovaných blocích jako je nadpis datového bloku, autor a verze. Na obr. 5.4.2 je hlavní okno této aplikace.

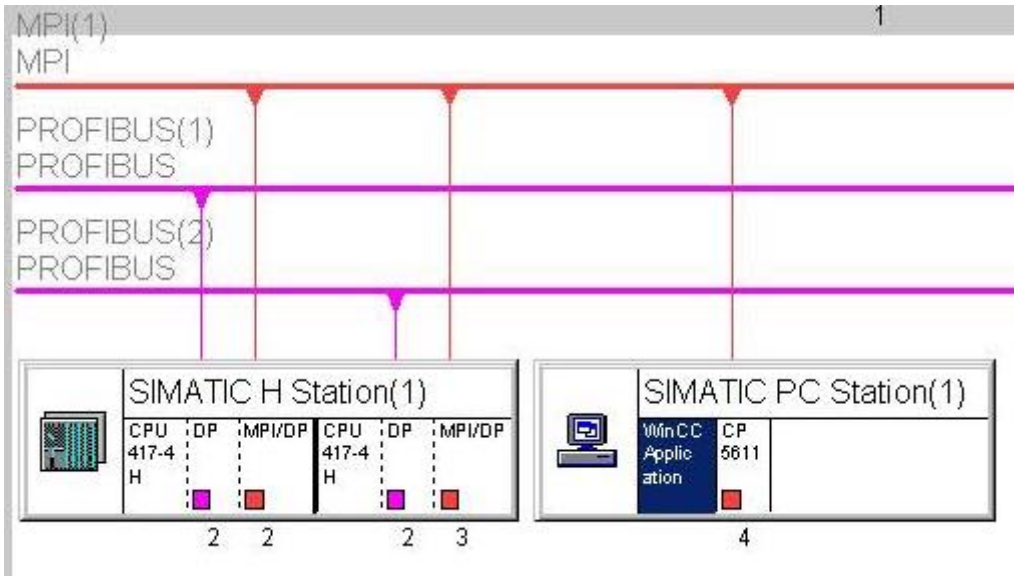


Obr. 5.4.2: Export parametrů do zdrojového souboru pro STEP 7 z externí aplikace

5.5 Porovnání s komplexním řešením PCS7 v6.0

V této kapitole budeme diskutovat rozdíl mezi řešením navrženým v této diplomové práci s PCS7 řešením. Jak již bylo řečeno softwarový balík PCS7 firmy SIEMENS kromě mnoha jiných rozšíření začleňuje WinCC projekt do SIMATIC Manageru a tím propojuje procesní část (AS) s vizualizační částí projektu (OS). Avšak licence na tento software je pro školní podmínky zatím nedostupná.

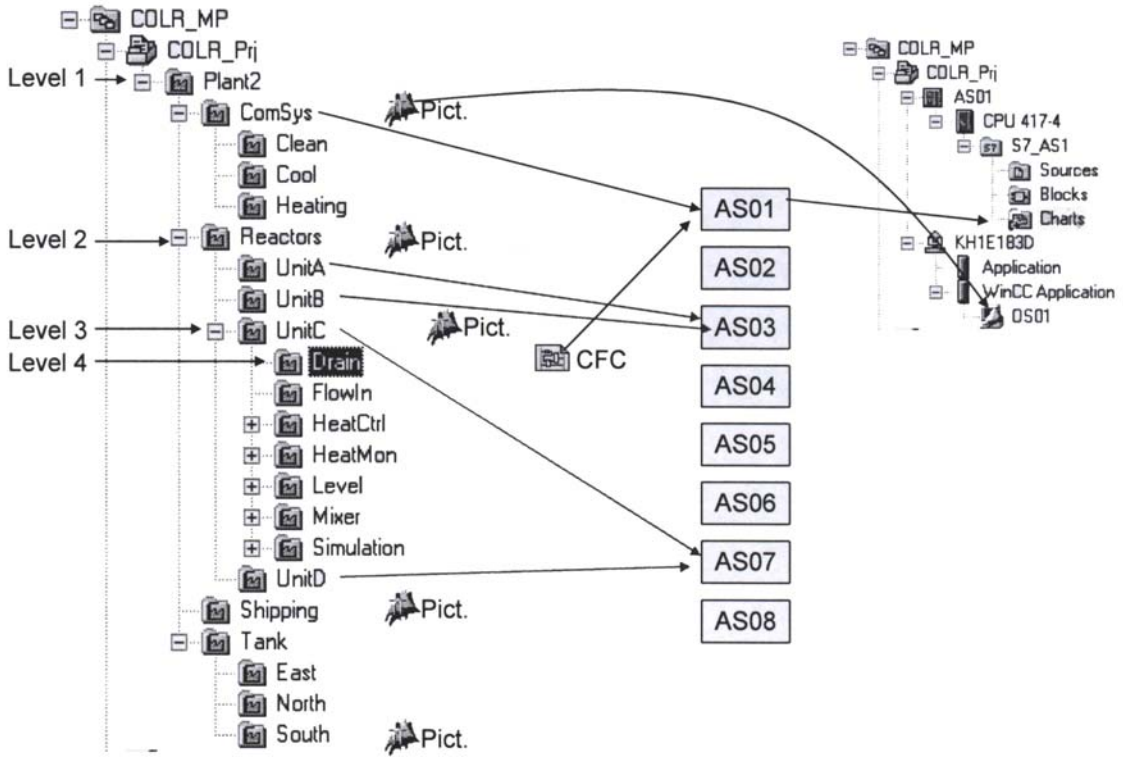
Na obr. 5.5.1 je znázorněna síť propojující PLC SIMATIC S7-400 s WinCC aplikací na řídícím PC. Je k tomu použita komunikace MPI, ale pokud obě komunikující strany budou vybaveny potřebným rozhranní, je možné použít libovolný přenos dat, např. komunikace protokolem TCP-IP na síti Ethernet.



Obr. 5.5.1: NetPro síť – propojení WinCC Aplikace s PLC SIMATIC S7-400

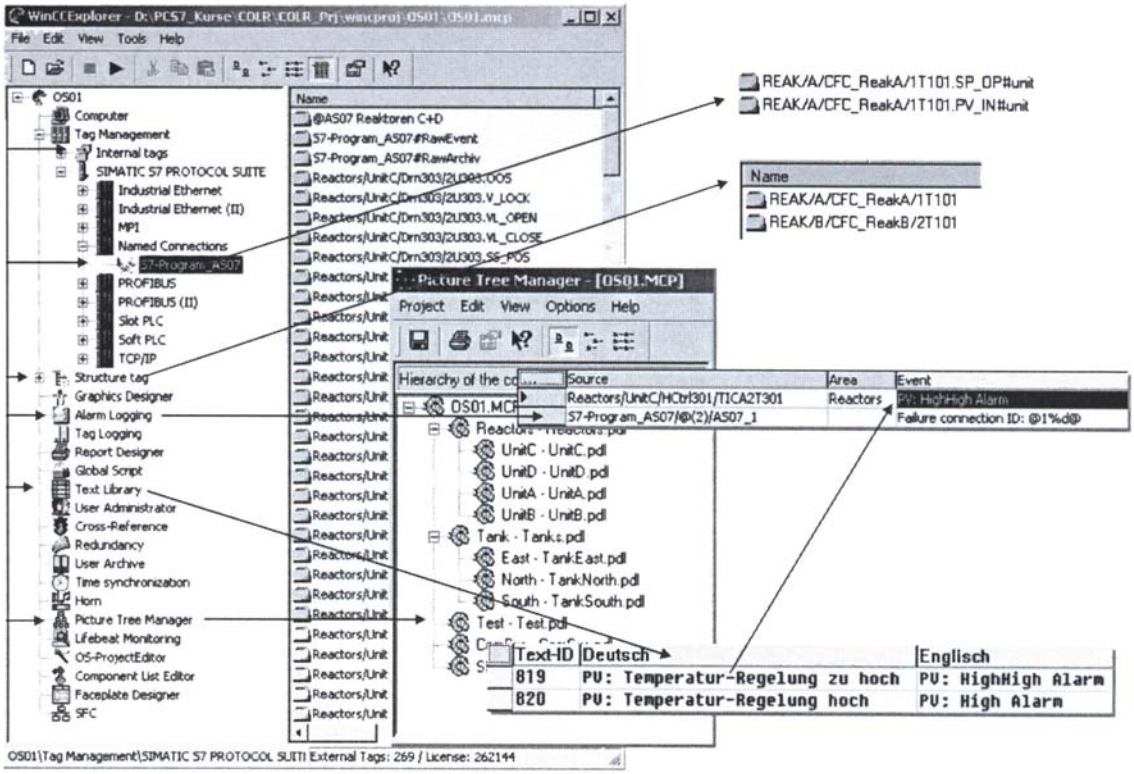
Po vytvoření odpovídající HW konfigurace a propojení WinCC aplikace se SIMATIC stanicí můžeme vytvořit v SIMATIC Manageru vazby mezi procesními daty a vizualizací. Pro tento účel nejlépe vyhovuje zobrazení projektu v pohledu. *„Plant Hiearchy“*.

Na obr. 5.5.2 vidíme ukázkový projekt v SIMATIC Manageru. V levé části je Plant Hiearchy, napravo projekt v klasickém pohledu a obdélníky AS0x uprostřed představují reálná PLC v technologii. Dále vidíme ikony *„Pict“,* které nám říkají, na které úrovni naší technologie je relevantní a rozumné zobrazovat procesní data na vizualizační obrazovce. Na úrovni *„Level4“* si můžeme všimnout *„CFC“* bloku. I přesto, že jsme se s pojmem CFC dosud nesetkali, přesahuje popis CFC problematiky rozsah této publikace, a proto se jí zabývat nebudeme. Řekněme si pouze, že CFC je obdobou funkčních a datových bloků má však mnohem větší možnosti. Pro porovnání s řešením navrženém v této diplomové práci je však dobré vědět, že k CFC lze přímo v SIMATIC Manageru mimo jiné definovat, jakým způsobem se mají tyto procesní data zobrazovat ve vizualizaci, zda je vůbec žádoucí, aby se zobrazovala a na jakých vizualizačních obrazovkách. Tuto možnost jsme u SIMATIC Manageru bez PCS7 postrádali, a proto bylo nutné přidat informaci při zakládání WinCC projektu do databáze. Ve skutečnosti na pozadí projektu v SIMATIC Manageru jsou data uložena také v databázi, a to v oddělené databázi pro procesní data a v samostatné databázi pro vizualizační data. Přesun dat z procesní části databáze do vizualizační se děje



Obr. 5.5.2: Plant Hiearchy – vazby mezi procesními daty a vizualizací

během tzv. OS kompilace, která se spouští ze SIMATIC Manageru. Výsledek můžeme vidět na obr. 5.5.3. Během kompilace se automaticky vygenerují vizualizační proměnné v sekci TagManagement->SIMATIC_S7_PROTOCOL_SUITE->NamedConnections->S7$Program, což je oblast tagů komunikačního kanálu definovaná v HW konfiguraci (obr.5.5.1). Pro jednoduchost řekněme, že vygenerované proměnné jsou proměnné datových bloků, které byly označeny pro použití ve vizualizaci. V pravém sloupečku podle jména proměnných, ve kterých je obsaženo umístění v Plant Hiearchy, můžeme vidět, že opravdu proměnné pocházejí z technologie z úrovně *„Level 4“,* kde jsme definovali jediný blok CFC (Reactors->UnitC->Drain). Dále si můžeme povšimnout vygenerovaných alarmů, které se také definují v SIMATIC Manageru na úrovni bloků CFC.



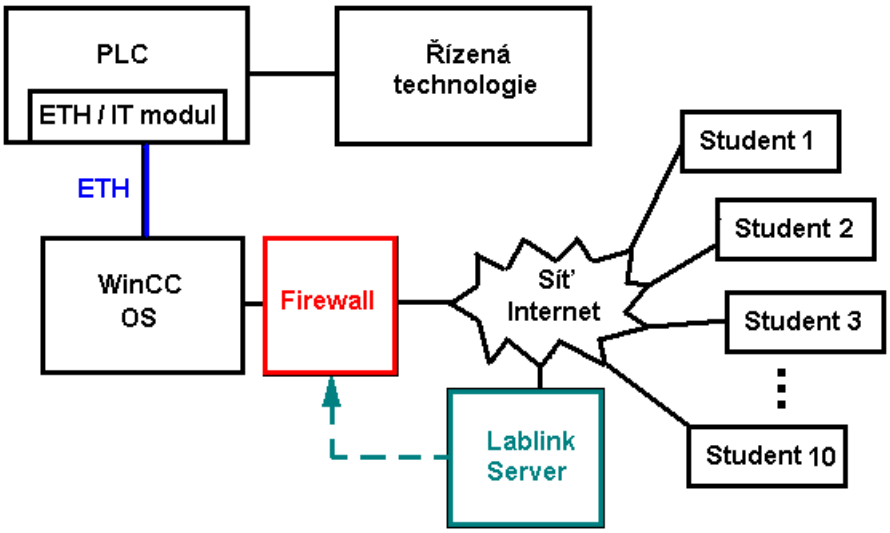
Obr. 5.5.3: Výsledek OS kompilace

V PCS7 WinCC projektu také existují standardní knihovny podobné naší @Typicals.PDL obsahující grafickou reprezentaci různých procesních dat - symboly. Z těchto knihoven se symboly při OS kompilaci kopírují na jednotlivé vizualizační obrazovky podobně jak je tomu v našem řešení. V naší aplikaci se o to však stará VBA skript, který jako zdroj dat nebere informace ze SIMATIC Manageru, protože tyto údaje nemáme bez PCS7 k dispozici, ale z naší externí MSQL databáze.

Problematika PCS7 je velmi rozsáhlá a více informací lze nalézt např. v [10].

6 Lablink – webové rozhranní

Pokud bychom vyvíjeli skutečnou aplikaci vzdáleného řízení technologie, čekala by nás nyní fáze testování a předání zákazníkovi. V našem případě však zákazníků je mnoho - jsou to studenti, kteří si na vytvořeném modelu budou osvojovat znalosti vývojového prostředí WinCC, SIMATIC Manageru a vzdálené řízení. Protože fyzický model je pouze jeden, musíme nějakým způsobem zajistit, aby v daný okamžik mohl s modelem pracovat pouze jeden student a na druhou stranu vytvořit systém, podle kterého by se studenti na modelu střídali. Oba problémy řeší rezervační on-line systém Lablink.



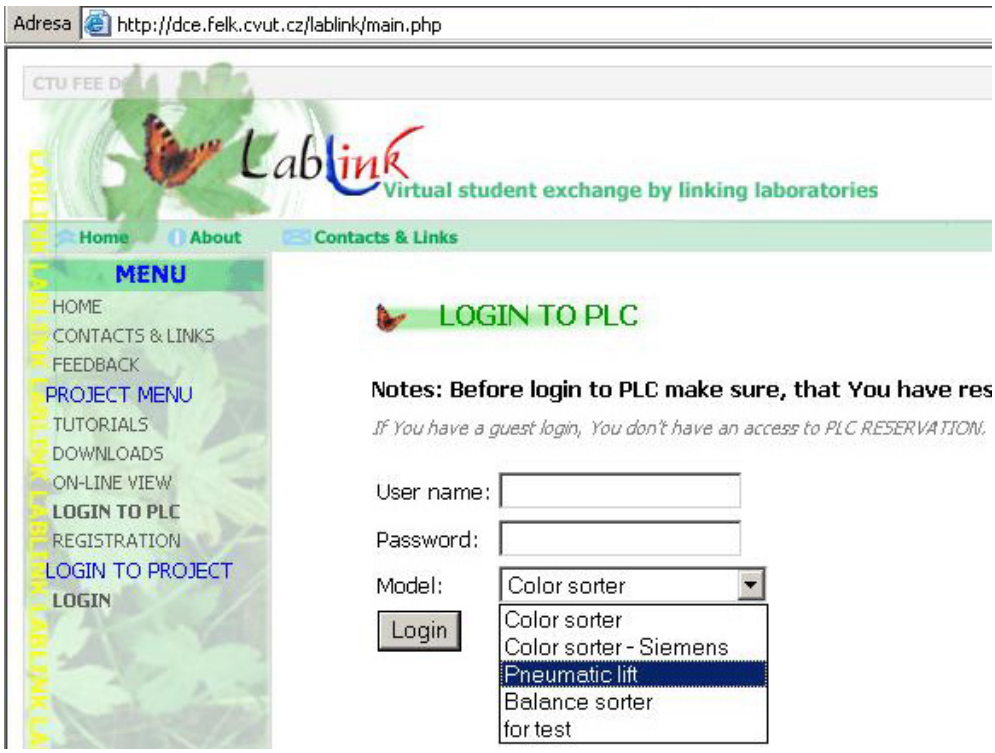
Obr. 5.5.1: Schématické znázornění funkce rezervačního serveru Lablink

Jak vidíme na obr. 5.5.1 studenti se připojují k modelu prostřednictvím sítě internet. Model včetně severu WinCC je umístěn na lokální síti a je od okolní sítě oddělen firewallem. O povolení přístupu přes firewall rozhoduje Lablink server. K definování, který uživatel může momentálně s modelem pracovat slouží tzv. IP tables. Ukázku kódu IP tables lze nalézt v příloze 11.8. Časový harmonogram změn nastavení firewallu, tedy změny v konfiguraci IP tables, je prováděn podle rezervací jednotlivých uživatelů. Rezervace probíhají a jsou ukládány na Serveru Lablink. Více se budeme autorizací vzdáleného přístupu k modelu zabývat v následující kapitole.

6.1 Popis rezervačního systému

Rezervační systém Lablink byl vytvořen v rámci předcházejících diplomových prací. Byl však navržen pouze pro rezervaci jednoho modelu. Proto před vlastní prací na dalším modelu bylo nutné rozšířit stávající systém tak, aby umožňoval vzdálenou rezervaci a připojování k různým modelům. To bylo také jedním z hlavních cílů této diplomové práce. Více si o rozšíření systému Lablink povíme v kapitole 6.2.

Systém Lablink je rezervační on-line prostředí skládající se z webového rozhranní a souhrnu relačních databází. Webové rozhranní je implementované v jazyce php a běží na serveru apache (příloha 11.7). Databáze spravuje MySQL server. V tabulkách databáze je uložen seznam všech uživatelů systému Lablink, dále tabulky obsahují informace o dostupných modelech, rezervační záznamy a další podpůrné informace. Struktura těchto tabulek je rozsáhlejší, a proto je zařazena až mezi přílohy. Uživatelé jsou rozděleni do skupin s různými pravomocemi. Na obr. 6.1.1 je ukázka webové stránky Lablinku. Vidíme, že tato stránka slouží k přihlášení k některému z modelů a vyžaduje autorizaci přístupu pomocí uživatelského jména a hesla.



Obr. 6.1.1: Jedna z webových stránek systému Lablink – přihlašování k modelům

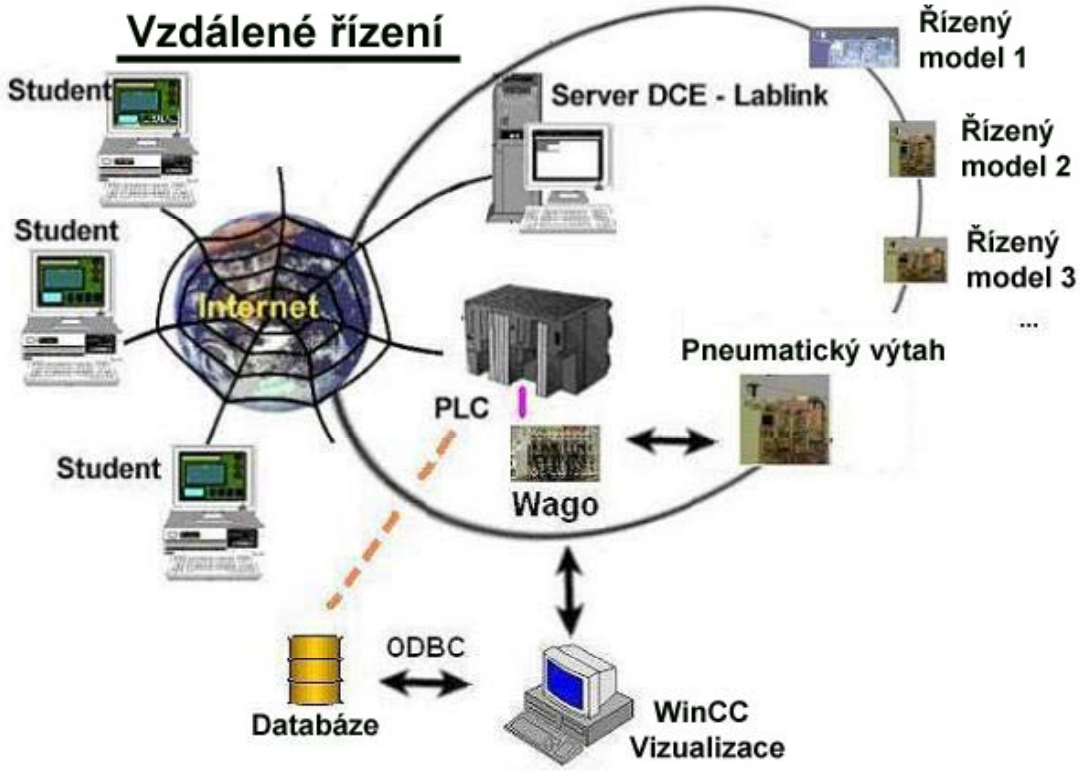
Nyní si podrobněji popíšeme fungování systému Lablink. Pro přihlášení do systému Lablink je nutné nejprve se zaregistrovat. Registraci provádí buď pověřený učitel, administrátor nebo je prováděna systémem automaticky. Při automatické registraci má uživatel pouze omezené pravomoci, ve zbylých případech záleží přidělené pravomoci na učiteli resp. administrátorovi. Po úspěšné registraci obdrží uživatel přihlašovací informace – uživatelské jméno a heslo. Tyto informace se použijí k přihlášení do systému. Poté si uživatel zvolí některý z nabízených modelů a rezervuje si den a hodinu, kdy bude s vybraným modelem pracovat. Pokud se pak daný uživatel přihlásí k modelu v dobu, kterou si zaregistroval, může s fyzickým modelem pracovat. V praxi to znamená, že uživatel (student) na svém vzdáleném PC vyvíjí řídící program, který nahraje do PLC, a chování modelu pak může sledovat přes webovou kameru, v našem případě navíc ještě ve WinCC vizualizaci. Pokud se model nechová podle očekávání, je nutné program upravit a znovu nahrát. To se opakuje až do dosažení požadovaného chování modelu.

Naše využití rezervačního systému Lablink pro model výtahu se liší od původní koncepce. Oproti původní situaci nám k modelu přibyla operátorská stanice s WinCC serverem. Proto jsme i tuto stanici zahrnuli do lokální sítě a firewall umístili před ní. Tím se studenti nepřihlašují přímo k modelu, jako tomu bylo v předchozím případě, ale právě k WinCC serveru (obr. 5.5.1). Tím se také rozšířilo spektrum úkolů, které lze distančně řešit a zadat jako úlohu pro studenty (viz. příloha 11.10).

Na WinCC serveru běží kromě WinCC aplikace také SIMATIC Manager, umožňující vytvářet program v prostředí STEP 7 a vzdáleně ho nahrát do PLC. Dále je zde k dispozici aplikace odbc2s7.exe pro generování zdrojových AWL souborů z databáze, ze kterých lze kompilací v SIMATIC Manageru vytvořit datové bloky programu ve STEP7. Poslední avšak pro uživatel nejdůležitější aplikací běžící na operátorské stanici je VNC server (kapitola 6.4).

6.2 Rozšíření rezervačního systému

Na obr. 6.2.1 je znázorněna současná celková situace prostředí pro vzdálenou výuku s detailnějším zobrazením našeho modelu pneumatického výtahu. Všechny modely se nacházejí v laboratoři K909 řídících systémů na Karlově Náměstí.



Obr. 6.2.1: Přehledové schéma zapojení modelu do existujícího systému

Pro dosažení našeho cíle umožnit vzdálený přístup k více modelům bylo nutné provést nejprve novou analýzu struktury tabulek databáze MySQL. Byly přidány nové tabulky a rozšířen počet údajů o modelech ukládaný do databáze. Výsledná struktura tabulek je značně rozsáhlá, a proto ji uvedeme až v příloze 11.9. Následně bylo nutné upravit webové stránky rezervačního systému Lablink. Zde bylo nutné změnit například rezervační rozvrh tak, aby byl použitelný pro více modelů. Dále byly zdynamizovány seznamy nabízených modelů při přihlašování k PLC podle aktuálního stavu v databázi. Při možnosti rezervovat si více modelů bylo náhle nutné kontrolovat, zda si studenti nerezervují více modelů ve stejný čas, což není přípustné. Byla přidána kontrola, zda student skutečně začal na modelu pracovat v době, kdy

si ho zarezervoval. V opačném případě je po určité době automaticky odhlášen a rezervace je uvolněná pro ostatní studenty, což značně zefektivňuje využití časového harmonogramu modelu. Pro náhodné uživatele, kteří si chtějí model pouze vyzkoušet a je pro ně neproveditelné z časových či jiných důvodů žádat o přidělení uživatelského účtu správce Lablinku, byl přidán speciální uživatelský účet guest. Uživatel guest má omezené pravomoci, nemůže například přistupovat do rezervačního rozvrhu, může se však přihlásit okamžitě k libovolnému modelu, pakliže je tento momentálně volný. Dále byla rozšířena sekce *„tutorials“* webových stránek Lablinku obsahující návody a manuály na dvoujazyčné stránky v češtině a angličtině. Byly provedeny ještě další drobnější úpravy webových stránek, které ale nejsou již tolik podstatné.

Všechny modely jsou vybaveny webovými kamerami, jejichž obraz lze sledovat na webových stránkách Lablinku [8]. Obr. 6.2.2 zachycuje kameru použitou pro model výtahu.

Obr. 6.2.2 : Webová kamera AXIS 2100

6.3 Důležité IP adresy

Webová kamera AXIS 2100:	**147.32.87.218** resp. http://147.32.87.218/admin
Model výtahu (CP 343-1 IT):	**147.32.87.210**
WinCC server:	**147.32.87.203**

6.4 VNC Server

Tato aplikace umožňuje práci na vzdáleném PC prostřednictvím sdílení plochy. Na operátorské stanici běží VNC server, ke kterému je možno se přihlásit pomocí VNC Vieweru (obr. 6.4.1), který si mohou studenti volně stáhnout z webových stránek Lablinkku. Jak jsme již zmínili, na operátorské stanici běží aplikace WinCC a je tam také k dispozici SIMATIC Manager a jiné nástroje, které může uživatel díky VNC používat jako by u vzdáleného počítače seděl. Přihlášení k operátorské stanici je podmíněno třemi autorizacemi. Vzhledem k tomu, že operátorská stanice je umístěna společně s modelem na lokální síti, je nutné si práci na této stanici zarezervovat pomocí systému Lablink stejně jako by se jednalo o běžný model. V rezervovanou dobu je potom nutné se k modelu z webových stránek Lablinku přihlásit opět obvyklým způsobem jako k obyčejnému modelu. Teprve potom firewall oddělující lokální síť od okolního světa dovolí danému uživateli přístup k modelu. Posledním třetím krokem je vlastní přihlášení k operátorské stanici pomocí VNC Vieweru (obr. 6.4.1). Zde již může být přihlašovací jméno a heslo společné pro všechny studenty, protože bez předchozích dvou kroků firewall pokus VNC Vieweru přihlásit se k operátorské stanici zamítne. Uživatelské jméno a heslo sdělí studentům vyučující.

Obr. 6.4.1: Přihlašování k operátorské stanici pomocí VNC serveru

Jedinou nevýhodou využití vzdálené plochy pomocí VNC Vieweru jsou vysoké požadavky na přenosovou rychlost našeho připojení k internetu. Vzhledem však k tomu, že parametry připojení k síti internet se v poslední době rychle zvyšují a v rámci ČVUT a přilehlých kolejí jsou na velmi dobré úrovni, je toto řešení postačující.

7 Závěr

Úkolem této diplomové práce bylo seznámit čtenáře s problematikou vzdáleného řízení technologického procesu a použití této metody řízení pro distanční vzdělávání. Protože se tento typ řízení již běžně používá, nespokojili jsme se s vlastní aplikací vzdáleného řízení, ale zaměřili jsme se především na řešení problémů, na které se v praxi naráží. Zabývali jsme se například návrhem řešení, jak co nejelegantněji vyřešit ovlivňování vizualizace změnami na procesní úrovni, ale také tím, jak co nejvíce zautomatizovat vlastní tvorbu a především následné úpravy vizualizace. Nyní můžeme říci, že se nám podařilo ve spolupráci s databází MSQL propojit procesní část projektu, zastoupenou SIMATIC Managerem, s WinCC vizualizací, a tím otevřít cestu k zautomatizování některých časově náročných úprav ve vizualizaci na základě údajů uložených v databázi. Naše řešení jsme také porovnali s obdobným řešení s použitím softwarového balíku PCS7, který se používá v průmyslu, ale na který bohužel zatím škola nemá licenci. Za největší přínos této práce považuji prozkoumání vizualizačních produktů nabízených firmou Siemens, které jsou bohužel zatím ve výuce nahrazeny konkurenčními značkami, avšak podle mého názoru mnohem méně profesionálními.

Pro ukázku vzdáleného řízení jsme si zvolili model pneumatického řízení, který byl vytvořen v rámci jedné z předchozích diplomových prací. V této části byla provedena analýza vhodnosti modelu pro vzdálené řízení, kterou model prošel pouze dostatečně pro naše momentální potřeby. Pro budoucí využití modelu v distančním systému byla navržena některá zlepšení týkající se především nerobustně vyrobené řídící logiky modelu.

V neposlední řadě byl model začleněn do on-line systému Lablink, což bylo vlastně jedním z hlavních cílů této diplomové práce. Tomuto kroku muselo předcházet rozšíření původního konceptu systému Lablink jak na úrovni struktury databáze tak na úrovni webového rozhranní. Hlavním zlepšením byla implementace podpory více modelů v systému distančního vzdělávání.

Závěrem bych chtěl poděkovat katedře řídících systémů resp. vedoucímu diplomové práce za udržování blízkých vztahů s firmami z oboru. Díky těmto kontaktům jsem měl možnost využít zkušenosti z mezinárodních projektů a odborného školení, bez kterých by pravděpodobně nebylo možné diplomovou práci dokončit.

8 Použitá literatura

[1] HENYCH, T. *Řízení a vizualizace technologického procesu pomocí PLC.* ČVUT Praha, 2004.

[2] HODNÝ, J. *Dálkové řízení technologického procesu.* ČVUT Praha, 2004.

[3] JOHN, J. *Systémy a řízení.* ČVUT Praha, 2004.

[4] BÍLEK, J. *Řídící systémy.* ČVUT Praha, 2002.

[5] NEPUSTIL, P. *Protokol TCP/IP,* ČVUT Praha, 2000.

[6] KRUGLINSKI, D. *Programujeme v Microsoft Visual C++.* Computer Press, 2000.

[7] *S7-CPs for Industrial Ethernet. Siemens, 2003.*

[8] ČVUT Praha. *Rezervační systém Lablink* [online]. Poslední revize 2005-05-01 http://dce.felk.cvut.cz/lablink

[9] ČVUT Praha. *Stránka předmětu řídicích systémů* [online]. Poslední revize 2005-05-01 http://dce.felk.cvut.cz/rs

[10] *SIMATIC PCS7 Systém Course.* Siemens, 2005.

9 Seznam obrázků

10 Seznam tabulek

11 Přílohy

11.1 Popis PID regulátoru

CALL "CONT_C", DB5	Comp_rst – kompletní restart, blok má inicializační rutinu, která se pustí při nastavení bitu na log 1
COM_RST :=#COMP_RST	Man_on - pokud je bit nastaven, rozpojí se smyčka a na výstupu je hodnota man
MAN_ON :=#MAN_ON	Pvper_on – při používání vstupu z I/O musíme nastavit na true
PVPER_ON:=TRUE	P_sel, I_sel, D_sel – zapojení do výpočtu příslušné složky P, I, D
P_SEL :=#GAIN_ON	Int_hold – zmražení výstupu integrační složky
I_SEL :=#INT_ON	I_int_on – inicializace integrační složky
INT_HOLD:=FALSE	Cycle – sample time, po totu dobu je výstup neměnný
I_ITL_ON:=FALSE	Sp_int – setpoint, žádaná hodnota
D_SEL :=#DER_ON	Pv_in – inicializační hodnota skutečné hodnoty
CYCLE :=#CYCLE	Pv_per – skutečná hodnota, vstup z I/O
SP_INT :=#setpoint	Man – manuální hodnota, na výstupu při Man_on
PV_IN :=	Gain, TI, TD – konstanty PID regulátoru
PV_PER :=#POZICE	Tm_lag – časové zpoždění derivační složky
MAN :=#MAN	Deadb_w – šířka pásma necitlivosti, pro výpočet PID
GAIN :=#GAIN	Lmn_HLM, Lmn_LLM – horní a dolní limit pro výstup
TI :=#TI	PV_fac, Lmn_fac – multiplikační konstanta pro vstup
TD :=#TD	PV_off, Lmn_off – hodnota, která se připočte k vypočítané hodnotě
TM_LAG :=	I_intval – inicializační hodnota integrační složky
DEADB_W :=#DEAD_BAND	Disv – poruchová veličina, přičítá se k vypočítané hodnotě
LMN_HLM :=#OUT_LIM_HI	Lmn – akční zásah
LMN_LLM :=#OUT_LIM_LOW	Lmn_per – akční zásah přepočtený na I/O
PV_FAC :=	Olmn_HLM, Olmn_LLM – signalizace překročení mezí
PV_OFF :=	Lmn_p, Lmn_i, Lmn_d – vypočítané hodnoty pro jednotlivé složky
LMN_FAC :=	Pv – výstup, skutečná hodnota která se jeví jako vstup
LMN_OFF :=#OUT_OFFSET	Er – skutečná odchylka, hodnota která vstupuje do výpočtu
I_ITLVAL:=	
DISV :=	
LMN :=	
LMN_PER :=#ZASAH	
OLMN_HLM:=	
OLMN_LLM:=	
LMN_P :=	
LMN_I :=	
LMN_D :=	
PV :=	
ER :=	

Tab. 11.1.1 : Popis parametrů PID regulátoru

Analogové napětí, které slouží jako reference pro komparátor je v případě prostředního PID výtahu čteno z výstupu PQW 320. Tuto hodnotu je nutné napojit na výstup PID regulátoru *„LMN_PER“* – akční zásah.

Pro oba krajní výtahy bohužel nelze měnit komparační napětí z PLC. Využívá se zde dvou předem nadefinovaných úrovní napětí pro pohyb výtahu nahoru a dolů. Pro použití těchto krajních výtahů pro vzdálené řízení je nutné vylepšit HW modelu.

11.2 Ukázka STL programu

```
Network 1: nacteni hodnot pro levy vytah
Comment:

      OPN   "DBleft"
//*********************************************************************
      L     P#I 5.0                     //nacteni cidel do db
      LAR1
      L     P#DBX 4.0
      LAR2
      L     B [AR1,P#0.0]
      SRW   4
      T     B [AR2,P#0.0]

//*********************************************************************
      L     P#I 5.0                     //nacteni tlacitek ve vytahu do db
      LAR1
      L     P#DBX 0.0
      LAR2
      L     B [AR2,P#0.0]
      L     B [AR1,P#0.0]
      OW                                // zachovej v pameti vsechna zmacknuta tlacitka
      L     MB     3
      OW
      T     B [AR2,P#0.0]
```

Obr. 11.2.1: Ukázka STL programu – scan vstupů a uložení do datového bloku

```
      OPN   "DBleft"

//************************************
      L     P#DBX 8.0                   //nacteni vystupu diod venku z db
      LAR1
      L     P#Q 4.0
      LAR2
      L     B [AR1,P#0.0]
      T     B [AR2,P#0.0]


//******************************
      L     P#DBX 6.0                   //nacteni vystupu diod ve vytahu z db
      LAR1
      L     P#Q 5.0
      LAR2
      L     B [AR1,P#0.0]
      SLW   2
      T     B [AR2,P#0.0]

//************************************
      A     DBX   10.1                  //nacteni vystupu 7-seg a smeru  z db
      =     "L_Seg_A"
      A     DBX   10.2
      =     "L_Seg_B"
      A     DBX   10.3
      =     "L_Di_Down"
      A     DBX   10.4
      =     "L_Di_Up"

//**********************************
      A     DBX   10.5
      =     "L_B"
```

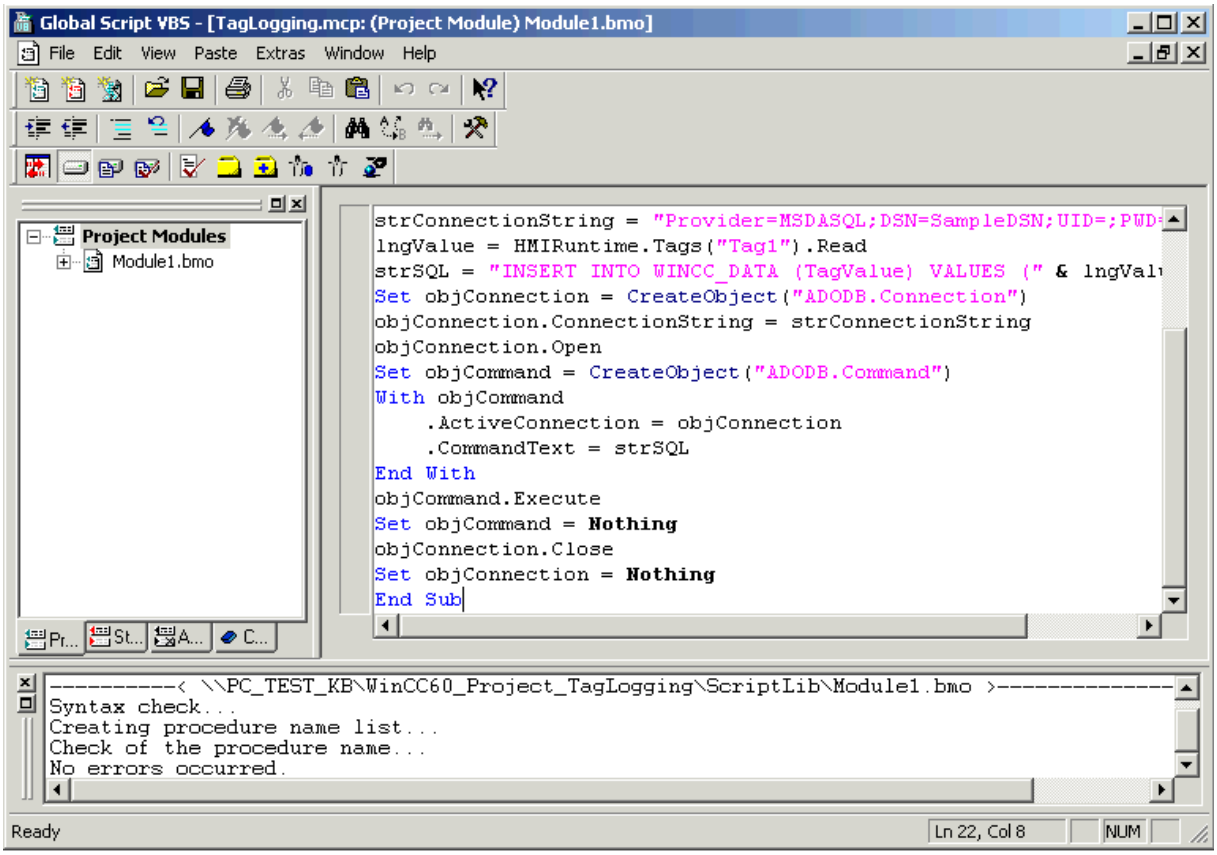
Obr. 11.2.2: Ukázka STL programu – zápis na výstupy

11.3 Ukázka C skriptu WinCC

Výňatek z kódu funkce, která inicializuje odbc spojení s databází. Funkce využívá dynamické knihovy odbc32.dll a je volána z runtimu WinCC při komunikaci s MSQL databází.

```
#include "apdefap.h"

//————————global variables————————————
SQLHSTMT   hstmt = NULL;
char pict_name[30];

void odbc_init(char* lpszPictureName,  char* ResObjectName, char* DBNameObject)
{


//————————————vlozeni dll knihovny s ovladaci odbc————————————
#pragma code("odbc32.dll")
SQLRETURN  SQL_API SQLAllocHandle(SQLSMALLINT HandleType,
     SQLHANDLE InputHandle, SQLHANDLE *OutputHandle);

SQLRETURN  SQL_API SQLSetEnvAttr(SQLHENV EnvironmentHandle,
     SQLINTEGER Attribute, SQLPOINTER Value, SQLINTEGER StringLength);

SQLRETURN  SQL_API SQLGetDiagRec(SQLSMALLINT HandleType, SQLHANDLE Handle,
       SQLSMALLINT RecNumber, SQLCHAR *Sqlstate,
       SQLINTEGER *NativeError, SQLCHAR *MessageText,
       SQLSMALLINT BufferLength, SQLSMALLINT *TextLength);

SQLRETURN  SQL_API SQLAllocHandle(SQLSMALLINT HandleType,
       SQLHANDLE InputHandle, SQLHANDLE *OutputHandle);

SQLRETURN  SQL_API SQLSetConnectAttr(SQLHDBC ConnectionHandle,
       SQLINTEGER Attribute, SQLPOINTER Value,
       SQLINTEGER StringLength);

SQLRETURN  SQL_API SQLConnect(SQLHDBC ConnectionHandle,
       SQLCHAR *ServerName, SQLSMALLINT NameLength1,
       SQLCHAR *UserName, SQLSMALLINT NameLength2,
       SQLCHAR *Authentication, SQLSMALLINT NameLength3);

SQLRETURN  SQL_API SQLGetInfo(SQLHDBC ConnectionHandle,
       SQLUSMALLINT InfoType, SQLPOINTER InfoValue,
       SQLSMALLINT BufferLength, SQLSMALLINT *StringLength);
|
#pragma code()
  ⋮

//————————————ziskej jmeno akt. obrazovky kam se bude zobrazovat vysledek——
p_pict_name = GetLocalPicture(lpszPictureName);
if (p_pict_name != NULL)
 strcpy(pict_name,p_pict_name);

//————————————alokuj handle————————————————————————
retcode = SQLAllocHandle(SQL_HANDLE_ENV, SQL_NULL_HANDLE, &henv);
if (retcode == SQL_SUCCESS || retcode == SQL_SUCCESS_WITH_INFO){
        SetText(pict_name,ResObjectName,"Alloc ENV handle succeed");

        //————————————nastav enviroment atributy————————————————
        retcode = SQLSetEnvAttr(henv, SQL_ATTR_ODBC_VERSION, p_sql_ver, 0);
        if (retcode == SQL_SUCCESS || retcode == SQL_SUCCESS_WITH_INFO){
                SetText(pict_name,ResObjectName,"Set ODBC version succeed");

                //———————————— Allocate connection handle ————————
                retcode = SQLAllocHandle(SQL_HANDLE_DBC, henv, &hdbc);
                if (retcode == SQL_SUCCESS || retcode == SQL_SUCCESS_WITH_INFO){
                        SetText(pict_name,ResObjectName,"Alloc DBC handle succeed");

                        //————————————nastav connection atributy————————————
                        SQLSetConnectAttr(hdbc, SQL_LOGIN_TIMEOUT, (void*)5, 0);
                        if (retcode == SQL_SUCCESS || retcode == SQL_SUCCESS_WITH_INFO){
                                SetText(pict_name,ResObjectName,"Connect attributes succeed");
                                #ifdef __MYSQL__

                                        //————————————pripojeni k databazi mysql————————————
                                        retcode = SQLConnect(hdbc, (SQLCHAR*) "myodbc", SQL_NTS,
                                        (SQLCHAR*) "root", SQL_NTS,
                                        (SQLCHAR*) "admin", SQL_NTS);
                                        retcode =SQLGetDiagRec(SQL_HANDLE_DBC, hdbc, 1, sqlstate, &native_err, err_msg, sizeof(err_msg), &err_msg_len )
                                #else
                                #endif
  ⋮
```

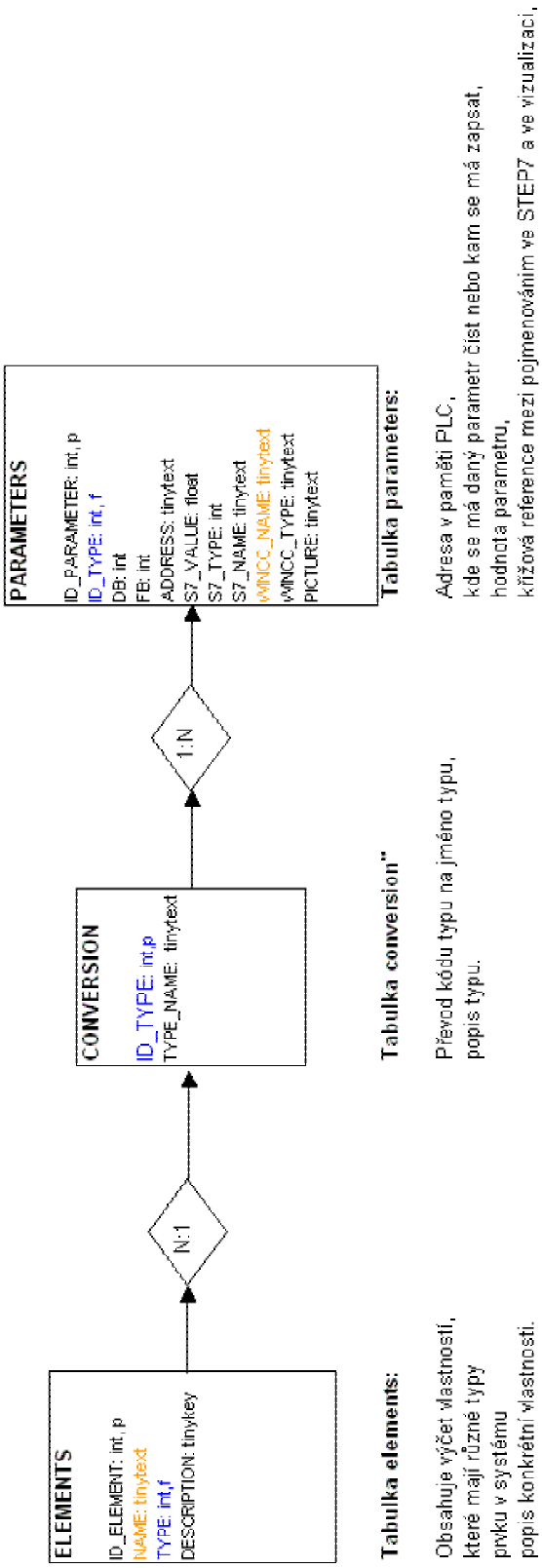
11.4 Ukázka VBS skriptu WinCC



11.5 Ukázka VBA skriptu WinCC

```
Private Sub cmdUpdate_Click()

    Dim f As Folder
    Dim fs As FileSystemObject
    Dim fc As Files
    Dim fSymbolFile As File

    Set fs = New FileSystemObject
    Dim strActiveDir, strSymbolsDir, strFilter As String

    Dim objGDApplication As grafexe.Application
    Dim objDest, objSource, objDestTempl, objDestActive As grafexe.Document
    Dim objHMIObject, objTypObject, objHMINew As grafexe.HMIObject
    Dim objProperty As grafexe.HMIProperty
     ⋮

    Set objDest = objGDApplication.ActiveDocument
    Set objDestActive = objGDApplication.ActiveDocument
    strActiveDir = objDest.Application.ApplicationDataPath     ' Active GraCS path
    strSymbolsDir = strActiveDir

    Set f = fs.GetFolder(strSymbolsDir)
    Set fc = f.Files
     ⋮

   ' Open a connection.
   Set cnn1 = New ADODB.Connection
   strCnn = "Provider=MSDASQL.1;Persist Security Info=False; Data Source=msql;
   cnn1.Open strCnn

   ' Open table.
   Set rstParameters = New ADODB.Recordset
   Set rstConversion = New ADODB.Recordset
   Set rstCheck = New ADODB.Recordset
    ⋮

   retcount = rstParameters.RecordCount
   Dim i As Integer
   If retcount Then
       rstParameters.MoveFirst
   Else

       If bClear Then

           For Each fSymbolFile In fc

               is_symbol = False

               'updatuj vsechny takove objekty na vsech obrazovkach
               If ((InStr(1, UCase(fSymbolFile.Name), ".PDL", vbTextCompare) <> 0)) _
                 And (InStr(1, UCase(fSymbolFile.Name), "@", vbTextCompare) = 0) Then

                   Set objDest = objGDApplication.Documents.Open(fSymbolFile)
                   For Each objHMIObject In objDest.HMIObjects

                       If (InStr(objHMIObject.Properties("ObjectName").value, "@") = 1) Then

                           'smaz vsechny objekty pripojene na jiz neexistujici polozky v databazi

                               strTag = objHMIObject.Properties("Tagname").value
                               rstCheck.Filter = "WinCC_name = '" & strTag & "'"
                               rstCheck.Open "select * from parameters", cnn1
                               If (rstCheck.RecordCount < 1) Then

                                   CountDelete = CountDelete + 1
                                   formGatherSymbols.txtLogText.Text = formGatherSymbols.txtLogTex
                                   formGatherSymbols.strProgress.value = "Deleting " & objHMIObjec
                                   formGatherSymbols.Repaint
                                   objHMIObject.Delete
                               End If
                               rstCheck.Close

                       End If
                   Next
                   objDest.Save
                   objDest.Close
               End If
           Next
    ⋮
```

11.6 Struktura tabulek pro automatizaci – MSQL 2000 Server



Obr. 11.6.1: Struktura tabulek pro automatizaci vizualizace ve WinCC

11.7 Ukázka PHP skriptu v rezervačním systému Lablink

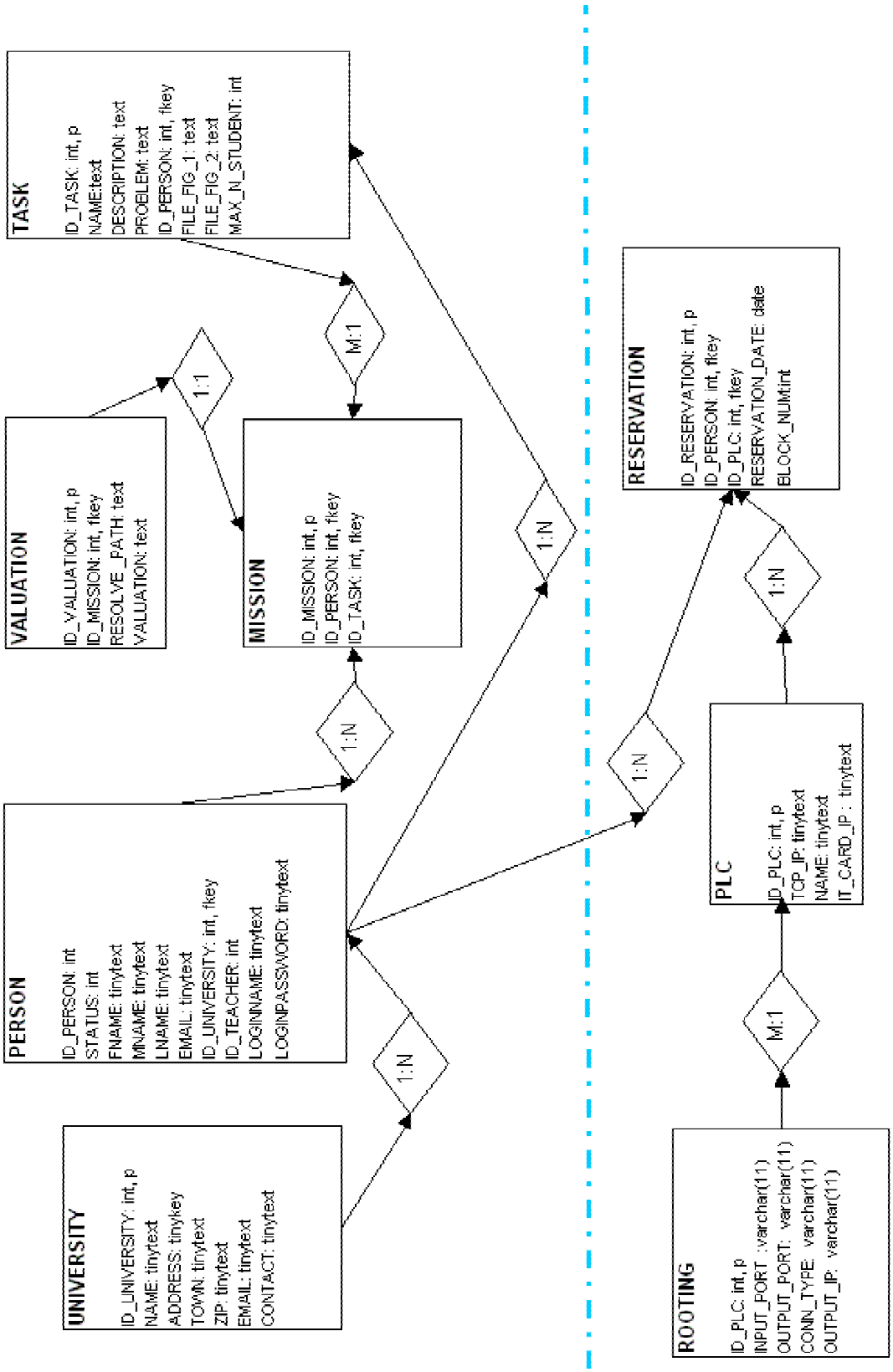
```
<?
#
# mysql.php - prace s Mysql
#

#
include('config_path.php');

# init mysql variables
$link = 0;


################################################################################
# fOpenMySQL()
#
function fOpenMySQL()
  {
    include( BASE."config.php" );
    $link = mysql_connect( $hostname , $username , $pass)
    or die ("Could not connect to database");
    //print ("Connected successfully");
    mysql_select_db ( $database ) or die ("Could not select database");
    return ($link);
  };

################################################################################
# fQueryGetItem()
#

function fQueryGetItem( $item, $table, $eq )
   {
       global $link;
     $link = fOpenMySQL();
     $query = sprintf('SELECT %s FROM %s WHERE %s', $item, $table, $eq );
     //echo $query;
     $result = fQueryMySQL( $query );
     while($line = mysql_fetch_array($result))
       {
        list($res) = $line;
       }
     return ($res);
   };
 ⋮
<?
##################################################################################################
# main
##################################################################################################
include('../config.php');
include_once('../html_oper.php');
include_once('../mysql_oper.php');
include_once("../time_oper.php");

/* head of the html docoment*/
sHeader ( "LABLINK Reservation - submint" , array(0, 0, "table.php?var1=$week_num&var2=$plc_num" ), 0, 0)

/* access isn't ready - access refused*/
if ( ($c_visitor_status != 1) ){
    echo "\n\tYou aren't able to make changes in DB<br>";
    exit;
  };
 ⋮
/* update the item in DB */
//echo "UPDATE reservation SET id_person=$person WHERE block_num='$block' AND reservation_date='$day'";
fQueryUpdate( "reservation","id_person=$person","block_num='$block' AND reservation_date='$day'" );
 ⋮
```

11.8 Ukázka IP tables Firewallu

V kódu php vidíme volání systémového příkazu iptables, který nastavuje firewall.

```
⋮

$query6 = "select output_ip, output_port  from rooting where id_plc = $id_plc";
$v6 = mysql_query($query6, $db) or exit ("Query failed:" . mysql_error());
if($r6 = mysql_fetch_row($v6)){
    $model_ip = $r6[0];
    $model_port = $r6[1];
}

if ($r6[0]){
    echo("<html><head><meta http-equiv=\"Refresh\" content=\"10; url=http://".$r6[0].":".$r6[1].
#    echo("query: $query<br>");
    echo("<DIV class=\"text1\">");
    echo("<h1>Welcome guest $r2[0] $r2[1] $r2[2]!</h1><br>");
    echo("<h3>Start time: ".Date("H:05")."</h3><br>");
    echo("<h3>End time: ".Date("H:55")."</h3><br>");
    echo("<h3>IP address: $ip</h3><br>");
    echo "Note: Till 10 seconds you will be refreshed to model's camera web site.";
    $s = "iptables -D fwModel".$id_plc." 1"; //smazani minuleho povoleni pristupu
    $s1 = system($s);
    $s = "iptables -A fwModel".$id_plc." -s $ip -j ACCEPT\n"; //povoleni pristupu pro nove ip

    #fwModel1 = wago u tabule - micky
    #       2 = S7 u tabule - micky
    #       3 = koule u schodu

    system($s);
    echo("</body></html>");
  ⋮
   else echo "<B>We are sorry, this model is under reconstuction.</B>".$r7[0];
 }
 else
 {
   echo("<html><head><meta http-equiv=\"refresh\" content=\"2; url=index.php\"></head><body>\n");
   echo("access denied<br>");
   echo("</body></html>");
 }
?>
```

11.9 Lablink - Struktura tabulek - MySQL



Obr. 11.9.1: Lablink – struktura tabulek

11.10 Ukázkové zadání úlohy pro studenty

1. Seznamte se s prostředím SIMATIC Manager.
2. Seznamte se s vývojovým prostředím STEP 7.
3. Seznamte se s vývojovým prostředím pro vizualizaci WinCC v.6.
4. Zarezervujte si model pneumatického výtahu pomocí on-line rezervačního systému Lablink. Pokud nemáte uživatelský účet pro přístup do Lablinku, postupujte podle pokynů na webových stránkách.
5. Pomocí VNC Vieweru se připojte k operátorské stanici – WinCC serveru. VNC Viewer je k dispozici ke stažení na webových stránkách Lablinku v sekci download. Přístupové jméno a heslo obdržíte od vyučujícího, IP adresa serveru je 147.32.87.203.
6. Na operátorské stanici spusťte SIMATIC Manager a nahrajte do PLC ukázkový program na řízení výtahu – manualne.zip. Využijte k tomu protokol TCP-IP. IP adresa modelu je 147.32.87.210. Ukázkovou aplikaci lze opět stáhnout z webových stránek Lablinku ze sekce download.
7. Spusťte WinCC projekt s ukázkovou aplikací v runtime módu a sledujte odezvy reálného modelu. Chování modelu můžete také přímo sledovat na webových stránkách Lablinku pomocí webové kamery.
8. Upravte ukázkový kód v prostředí STEP 7 tak, abyste dokázali bezpečně řídit pneumatický výtah. Tento kód můžete průběžně nahrávat do PLC a sledovat odezvy.
9. Vytvořte některé nové proměnné ve STEP 7, které mají být přidány do vizualizace a změňte adresu některých doposud používaných proměnných – např. paměťových buněk, které se předávají do vizualizace.
10. Spusťte Graphics Designer a otevřete knihovnu vizualizačních prvků – symbolů - @Typicals.PDL. Tento soubor se nachází v adresáři GraCS aktuálního projektu.
11. Vytvořte vlastní symbol dle vaší volby podle vzoru již předpřipravených symbolů v knihovně. Knihovnu uložte a zavřete.
12. Otevřete soubor @UpdatePictrures.PDL , který se nachází opět v adresáři GraCS.
13. Pomocí položek menu S7 Import->Insert New Item a S7 Import->Update Item vytvořte resp. aktualizujte všechny proměnné v databázi, které jste ve STEP 7 kódu přidali resp. změnili. Novým proměnným přiřaďte jako grafickou reprezentaci ve WinCC váš nový typ symbolu a zvolte cílovou obrazovku.

14. Ve WinCC Exploreru otevřete Tag Management a zkontrolujte, zda se vytvořily nové vizualizační proměnné podle vámi zadaných údajů a zda se změnily existující proměnné podle aktuálního stavu v databázi.
15. Spusťte znovu runtime mód vizualizace nebo pokud stále běží obnovte obrazovku přepnutím na jinou obrazovku a zpět. Povšimněte si, že proměnné, u kterých jste změnili např. adresu a aktualizovali jste je v databázi ve vizualizaci nefungují nebo pokud fungují, nezobrazují správná data, protože stará adresa proměnné již není platná.
16. Zavřete všechny soubory Graphics Designeru kromě @UpdatePictures.PDL a zvolte v menu S7 Import položku Update Pictures. Pro aktualizaci označte ze seznamu vámi nově přidané prvky a změněné proměnné a spusťte update obrazovek.
17. Přepněte se opět do runtime módu a obnovte vizualizační obrazovku. Ověřte, že nyní jsou změněné proměnné již funkční a zobrazují data podle nového kódu ve STEP 7.
18. Otevřete nyní v Graphics Designeru, že na obrazovce, kterou jste zvolili jako cílovou pro nové prvky jsou nakopírovány vaše nové symboly z knihovny a jejich dynamické vlastnosti jsou správně vyplněné. Tzn. symboly jsou napojené na nové proměnné ve STEP 7 a funkční, bez manuálního zásahu do vizualizace.
19. Testujte další možnosti využití knihovny symbolů a zkoušejte aplikovat různé změny v databázi. Zkuste například vytvořit nový symbol v knihovně funkcí podobný už nějakému hotovému a změňte například jen jeho barvu. Pokuste se hromadně změnit barvu všech takových symbolů na vizualizačních obrazovkách s použitím knihovny symbolů a databáze. Dále testujte například hromadnou záměnu podobných prvků na obrazovkách. Např. všechna tlačítka, které přepínají nějaký režim mezi zapnuto a vypnuto hromadně zaměňte za checkboxy opět pomocí databáze a knihovny prvků apod.

Při vypracování postupujte podle návodu který je umístěn na webových stránkách modelu.

11.11 Obsah přiloženého CD