ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ v Praze

Fakulta elektrotechnická
Katedra řídicí techniky

MODEL DOMOVNÍ AUTOMATIZACE

Diplomová práce

2006 Michal Slezák

České vysoké učení technické v Praze - Fakulta elektrotechnická

Katedra řídicí techniky *Školní rok:* 2004/2005

ZADÁNÍ DIPLOMOVÉ PRÁCE

Student: Michal Slezák

Obor: Technická kybernetika

Název tématu: Model domovní automatizace

Zásady pro vypracování:

1. Pokračujte v práci na realizaci modelu domovní automatizace. Realizujte vytápěcí systém modelu.
2. Navrhněte a realizujte různé režimy chování budovy. Simulujte přítomnost osob v místnostech a chování okolního prostředí (režim den/noc, počasí apod.).
3. Vytvořte Internetovou prezentaci modelu budovy a okolního prostředí.

Seznam odborné literatury: Dodá vedoucí práce.

Vedoucí diplomové práce: Ing. Martin Linhart

Termín zadání diplomové práce: zimní semestr 2004/2005

Termín odevzdání diplomové práce: leden 2006

prof. Ing. Michael Šebek, DrSc.
vedoucí katedry

L.S.

v zast.

prof. Ing. Vladimír Kučera, DrSc.
děkan

V Praze dne 08.03.2005

Prohlášení

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

V Praze dne: 26. 5. 2006 Podpis: Michal Sleva

Abstrakt

Diplomová práce se skládá ze dvou hlavních částí. První část popisuje návrh a hardwarovou realizaci ovládání termohlavice pro síť LonWorks, realizaci centrálního vytápění, návrh a implementaci regulačních algoritmů pro kotel, solární panel a integraci celého systému do modelu administrativní budovy pomocí softwarového nástroje LonMaker. Druhá část se věnuje vizualizaci administrativní budovy. Hlavním výsledkem je dynamický virtuální model budovy v běžném webovém prohlížeči s využitím VRML, jazyka Java a spojení této virtuální reality s reálným modelem skrze LNS server a komunikační rozhraní Javy.

Abstract

This diploma thesis consists of two main topics. In the first one is described design and HW realization of the heater control head for the LonWorks network, realization of the central heating, design and implementation of the regulation algorithms for furnace and solar panel and integration of the whole system to the model of automation building through use of the LonMaker SW tool. Second part deals with visualization of the automated building. Major goals were to bring a live virtual model of the building to common web browser using VRML and Java languages and connect this virtual reality with real model via LNS server and Java socket communication.

Poděkování

Chtěl bych poděkovat všem, kdo mi při vzniku této práce pomáhali. Jmenovitě Ing. Martinu Linhartovi, vedoucímu mé diplomové práce za cenné rady při její tvorbě, Ing. Pavlu Burgetovi při konzultaci s realizací diplomové práce, Ing. Ondřeji Dolejšovi za bezednou shovívavost při řešení problémů během mé diplomové práce týkajících se PLC Wago, Ing. Bohuslavu Kirchmanovi za rady při realizaci otopného systému v modelu administrativní budovy. Dále topenářské firmě Červinka, jmenovitě panu Heroutovi za praktické rady při návrhu otopného systému. Děkuji též Ing. Michalu Schoberovi při kontrole diplomové práce.

Nemenší díky patří rodičům a mé přítelkyni za podporu a trpělivost při mém studiu. Zvláště děkuji za jejich trpělivost při psaní této diplomové práce.

Obsah

1 Úvod **1**

2 LonWorks **3**

2.1 Úvod do technologie LonWorks . . . 3

2.1.1 Technologie LonWorks . . . 3

2.1.2 Neuron Chip . . . 5

2.1.3 Programovací jazyk Neuron C . . . 8

2.1.4 Vývojové prostředí Node Builder . . . 9

2.2 Vhodné zařízení pro nod . . . 10

2.2.1 Výběr technologie . . . 10

2.2.2 Zvolená termohlavice . . . 11

2.3 Konstrukce LonWorks nodu . . . 11

2.3.1 Napájecí část . . . 11

2.3.2 Procesorová část . . . 12

2.3.3 Výkoná část . . . 12

2.3.4 Softwarová část . . . 12

2.3.5 Přepnutí režimu komunikace . . . 14

2.4 Regulátor . . . 16

2.4.1 Vhodné ovládání . . . 16

2.4.2 Softwarové ovládání . . . 17

2.4.3 Skokový regulátor . . . 17

2.4.4 Diferenční regulátor . . . 17

2.5 Zapojení nodu termohlavice a Wago do sítě LonWorks . . . 17

2.5.1 Node Builder . . . 17

2.5.2 Termohlavice v síti LonWorks . . . 18

3 VRML a internetová prezentace **19**

3.1 Úvod do VRML . . . 19

3.1.1 Technologie VRML . . . 20

3.1.2 Koncept VRML modelu . . . 28

3.1.3 Stručné využití a potenciál . . . 32

3.2 Abstrakce modelu . . . 33

3.3 Realizace . . . 33

3.3.1 Konstrukce z knihoven 33
3.3.2 Spojení jednotlivých komponent 33
3.3.3 Dynamické VRML 35
3.4 Java ve VRML 38
3.4.1 Zakomponování Javy do VRML 38
3.4.2 Konstrukce java tříd pro virtální svět 40
3.4.3 Ovládání modelu pomocí Javy 40
3.5 VRML a realný model administrativní budovy 44
3.5.1 Java komponenty firmy Echelon 44
3.5.2 Proces komunikace mezi virtuálním a reálným světem 44
3.6 Cortona VRML prohlížeč 45
3.6.1 Popis prohlížeče 45
3.6.2 Ovládání prohlížeče 45
3.6.3 Další alternativní prohlížeče VRML světů 46
3.7 Webový portál 46
3.7.1 Redakční systém 46
3.7.2 Rozvoj webového portálu 48

4 Otopný systém 49
4.1 Úvod do problematiky vytápění objektů 49
4.1.1 Vytápění budov 49
4.1.2 Tepelné zdroje 49
4.1.3 Tepelné ztráty 50
4.2 Vybrané otopné systémy 51
4.2.1 Elektrokotel 51
4.2.2 Solární kolektor 52
4.2.3 Vytápění radiátory 52
4.3 Tepelné ztráty modelu 52
4.3.1 Materiály pro výpočet tepelných ztrát modelu 53
4.3.2 Výpočet tepelných ztrát 53
4.4 Realizace 54
4.4.1 Konstrukce 54
4.4.2 Otopná tělesa 54
4.4.3 Elektrická konstrukce 55

4.4.4 Použité ovládací prvky 56
4.5 Regulátor 56
4.5.1 Hardware pro regulátor 56
4.5.2 Software regulátoru 57
4.5.3 Tepelné okruhy 59

5 Závěr **60**

A Dokumentace LonWorks **63**
A.1 Elektrická zapojení 63
A.1.1 Procesorová část 63
A.1.2 Ovládací část 64
A.2 Seznam součástek 65
A.2.1 Procesorová část 65
A.2.2 Ovládací část 65

B Použité zkratky **66**

C Schémata **67**
C.1 VRML model 67
C.2 Tepelná část 70

D Průvodci **71**
D.1 LonMaker 71
D.2 Přidání zařízení v LonMakeru 75
D.3 Přidání funkčního bloku v LonMakeru 80
D.4 Implementace programu do Neuron chipu 81
D.5 Wago Toplon-prio 85
D.6 Wago-IO-Pro 32 93
D.7 IPlongate 100

E Obsah přiloženého DVD **105**

Seznam tabulek

1 Rozdělení spotřeby elektrické energie v domácnostech. 1

2 ISO/OSI vrstvy implementované v protokolu LonTalk v síťi LonWorks . . 3

3 Popis Neuronchipu TMPN3120FE3M od Toshiby 6

1 Úvod

V dnešní době, kdy ceny všech běžných energií stále rostou, je stále důležitější jejich úspora. Podíváme-li se podrobněji na spotřebu energie, zjistíme, že v domácnostech, firmách či v kancelářích je největší část energie spotřebována na vytápění objektů. Níže uvedená tabulka dokládá rozložení spotřeby elektrické energie. Zdroj informací www.Infoenergie.cz [1].

Parametr	Hodnota	Jednotka
Vytápění	57	%
Ohřev teplé užitkové vody	22	%
Příprava stravy	7	%
Chlazení a mražení potravin	5	%
Praní a žehlení prádla	4	%
Osvětlení	2	%
Ostatní energie	3	%

Tabulka 1: Rozdělení spotřeby elektrické energie v domácnostech.

Z tohoto důvodu se pro vytápěná nasazují energeticky šetrná zařízení. Požadavkem na technologii řízení je automatizace mnoha rozličných procesů a jejich optimalizace. Důležitou podmínkou souvisejícím s prudkým rozvojem internetu je kontrola procesu přes internet či vnitropodnikovou počítačovou síť.

Vedle alternativních zdrojů energií směřuje vývoj v této oblasti k její úspoře resp. k optimálnímu řízení celého objektu podle aktuálních požadavků. Z tohoto důvodu je nezbytné, aby čidla a akční členy vzájemně komunikovaly. V současné době je na trhu více druhů technologií. Jednou z nich je i síť LonWorks americké firmy Echelon. LonWorks neslouží pouze k úspoře energií, ale i k celkové automatizaci rozličných činností. Vzhledem k svému charakteru nachází své hlavní uplatnění v domovní automatizaci. Proto byla zvolena jako nosná část automatizace modelu administrativní budovy. Ovládání přes internetové rozhraní je realizováno pomocí softwaru firmy Axeda[1].

Firma Echelon poskytuje sběrnici LON[2] jako otevřený standard. To umožňuje výrobcům snadný vývoje zařízení od měření tepla, intentzity světla přes regulátory až po inteligentní ovládací prvky. K jejich vývoji slouží nástroj NodeBuilder výše uvedené firmy Echelon. Další organizací úzce spojenou s technologií LonWorks je Lonmark, která se stará o standardizaci objektů jednotlivých typů zařízení. Tato standardizace umožňuje vzájemnou bezproblémovou komunikaci zařízení od různých výrobců.

Diplomovou práci lze rozdělit do třech větších celků, které jsou v následujících kapitolách hlouběji rozebrány. První kapitola pojednává o technologii LonWorks a modulu pro ovládání radiátoru, který byl vytvořen v rámci diplomové práce. Druhá kapitola se věnuje virtuálnímu světu vytvořenému pomocí technologie VRML (Virtual Reality Modeling Language), která je zakomponována v běžném webovém prohlížeči. VRML pomocí programovacího jazyka Java ovlivňuje jak samotný virtuální svět, tak i reálný model administrativní budovy v laboratoři řídící techniky. Kapitola dále obsahuje popis

[1] Axeda je softwarový balíček, který obsahuje softwarové PLC, programovací prostředí, vizualizaci včetně generování webových stránek a webový server založený a vytvořený v jazyce Java

[2] LON je zkratka pro Local Operating Network

webové prezentace celé administrativní budovy. Poslední kapitola se věnuje otopnému systému celého domu. Typům zdroje tepla, topným tělesům, regulátorům a ovládacím prvkům umístěných v domě.

Součástí práce je přiložené DVD, které obsahuje schémata elektroinstalace, tepelných rozvodů, katalogové listy, všechny potřebné informace k administrativní budově, rejstřík zkratek, veškeré prezentace.

2 LonWorks

2.1 Úvod do technologie LonWorks

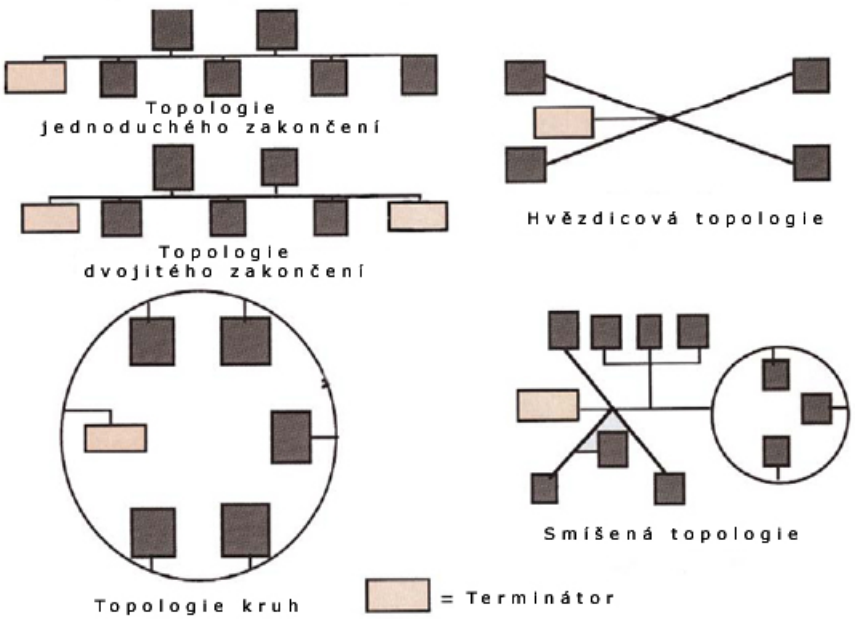
2.1.1 Technologie LonWorks

Technologie LonWorks je decentralizovanou sítí a je určena primárně pro nasazení v automatizaci budov. Nasasení je dáno jejím principem, přenosovou rychlostí a v neposlední řadě i způsobem přenosu informace. LonWorks umožňuje přenos pomocí radiových vln, může využít i napájecí vedení a mnoho dalších přenosových médií. Proto není fyzická vrstva v ISO/OSI modelu LonWorks definována.

	OSI vrstva	Účel
7	Aplikační	Aplikační kompatibilita
6	Prezentační	Interpretace dat
5	Relační	Vzdálené funkce
4	Transportní	Zajištění spolehlivosti spojení
3	Síťová	Cílové adresování
2	Linková	Přístup k médiu a rámcování
1	Fyzická vrstva	Elektrické spojení

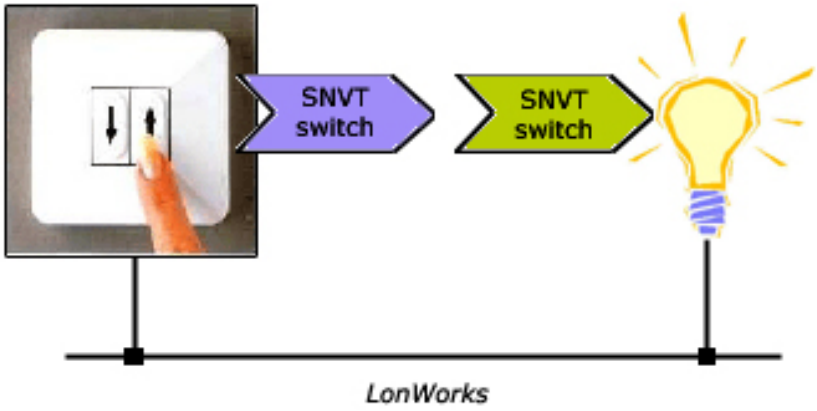
Tabulka 2: ISO/OSI vrstvy implementované v protokolu LonTalk v síťi LonWorks

Pro svoji rychlost, která je závislá na zvoleném přenosovém médiu, je vhodná především pro automatizaci budov, kde se nevyžaduje kritická doba odezvy. Celá síť je koncipována jako volná topologie.



Obrázek 1: Ukázky topologií sítě LonWorks

Síť umožňuje velmi komfortní spojení dvou či více zařízení pomocí SNVT[3] proměnných. Ty umožňují snadnou výměnu dat aniž by programátor musel zajišťovat jejich distribuci. Jsou to softwarová spojení uvnitř fyzické sítě.

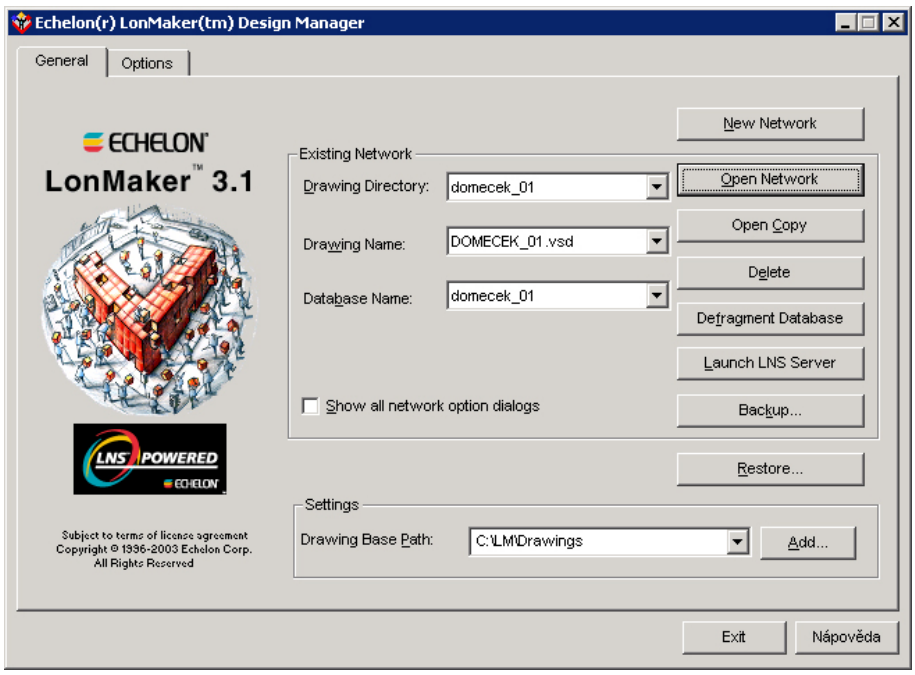


Obrázek 2: Ilustrace spojení síťových SNVT proměnných umístěných ve dvou zařízeních.

Všechny SNVT proměnné jsou objekty, které určují o jakou fyzikální veličinu se jedná, je zadán její rozsah a přesnost. Distribuce těchto proměnných je zajištěna pomocí směrovacích tabulek zapsaných do zařízení při konfiguraci celé sítě. V každé tabulce je uvedeno na jaké zařízení se budou odesílat výstupní proměnné a odkud se budou vstupní proměnné číst. Celá funkcionalita je zajištěna speciálním procesorem s pro programátora je tato ditribuce transparentní. Pro vývojáře je velkou výhodou, že LonWorks je otevřený systém, který umožňuje velice jednoduchou výměnu informací mezi výrobky jiných firem. Umožňují to standardizované profily. Každý typ či druh výrobku určeného pro provoz v síti LonWorks má definován jaké má mít vstupy a výstupy. Z tohoto důvodu je jednoduché stejné typy výrobků vyměnit bez nutnosti zásahu v topologii sítě, v programu či konfiguraci. O dodržování stanovených norem a rozvoj profilů se stará organizace Lonmark. Jedná se o organizaci sdružující významné výrobce zabývající se vývojem zařízení pro LonWorks. Mezi známé členy patří Honeywell Inc., Fuji Electric Co. Ltd., LOYTEC electronics GmbH atp. Další informace [2]. Vše co se týká LonWorks je dobře dokumentováno. Na internetu je veškerá potřebná dokumentace nabízena ke stažení zdarma. Jistou výhodou i nevýhodou této koncepce je, že LonWorks se musí při fyzické rekonfiguraci, též rekonfigurovat i v tabulkách dotčených zařízení. Virtuální sítě spojení SNVT proměnných se fyzická rekonfigurece nedotýká a proto není potřeba přepisovat kód aplikace.

Ke konfiguraci sítě se využít konfiguračních nástrojů. Běžně využívaným nástrojem pro konfiguraci sítě je program LonMaker zmiňované dříve firmy Echelon. Tento program využívá Microsoft Visio. V něm jsou zabudována makra, zásné moduly pro konfiguraci a práci se sítí LonWorks.

[3]SNVT je zkratka pro Standart Network Variable Type



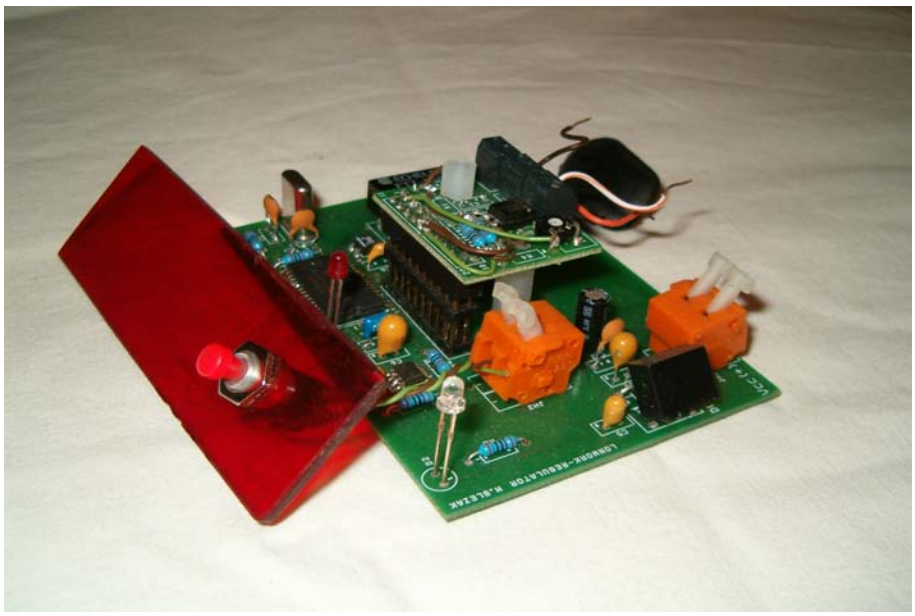
Obrázek 3: Ilustrační obrázek prostředí LonMaker

Uživatelské programy včetně konfigurace sítě se dají do nodů[4] distribuovat pomocí LonWorks. To umožňuje z jediného bodu v síti spravovat a konfigurovat celou síť. Pro správnou identifikaci mají všechny nody v síti celosvětově unikátní adresu nazývanou Neuron ID (lze přirovnat k MAC[5] adrese v síti Ethernet). Tento identifikátor je vložen výrobcem při výrobě procesoru. Nod po stisku servisního tlačítka vysílá do sítě své Neuron ID, aby bylo možné zařízení identifikovat v síti a následně nakonfigurovat nebo nahrát program.

2.1.2 Neuron Chip

Každé zařízení v síti LonWorks se nazývá nod. Vzhledem k tomu, že LonWorks je decentralizovaná síť, je uvnitř každého nodu umístěn procesor a spojení se sítí zajišťuje tzv. transceiver (v našem případě FTT-10A). Nod minimálně obsahuje Neuron chip, příslušný transceiver, paměť a ovládané zařízení. Vývoj procesorů zajišťuje mateřská firma Echelon. Výrobu procesorů zajišťuje více výrobců například Toshiba, Motorola a další.

[4] Nod je zařízení zapojené do sítě LonWorks, schopné komunikovat s touto síťí
[5] Media Access Control



Obrázek 4: Vytvořený nod pro termohlavici pro LonWorks

Neuron chip je složen ze třech 8 bitových procesorů. První procesor obstarává přístup na médium. Druhý zajišťuje síťový provoz, autorizace, správu síťových proměnných, řízení paketů aj. Třetí procesor obsluhuje uživatelské aplikace a též operační systém v něm zabudovaný. Podrobnější popis a funkce procesoru jsou v [10]. Neuron chipy se dají rozdělit do dvou skupin. Procesory řady 3120 neumožňují připojení externí paměti, obsahují RAM, ROM a EEPROM. V případě EEPROM je počet přepisů omezen přibližně na 10000 zápisů. Vestavěný operační systém hlídá jaká data se zapisují do paměti EEPROM a pokud jsou shodná, nedojde k jejich zápisu. Tím se výrazně omezí počet přepisů. Oproti řadě 3150 ještě navíc obsahuje paměť ROM, kde je uložena obsluha komunikačního protokolu LonTalk, knihovnu funkcí a TaskScheduler[6]. Druhá skupina procesorů řady 3150 umožňuje připojit externí paměť typu EEPROM. Tato paměť obsahuje uživatelskou aplikaci, data a firmware procesoru. Bližší popis typů a využití pamětí je uvedeno v [10].

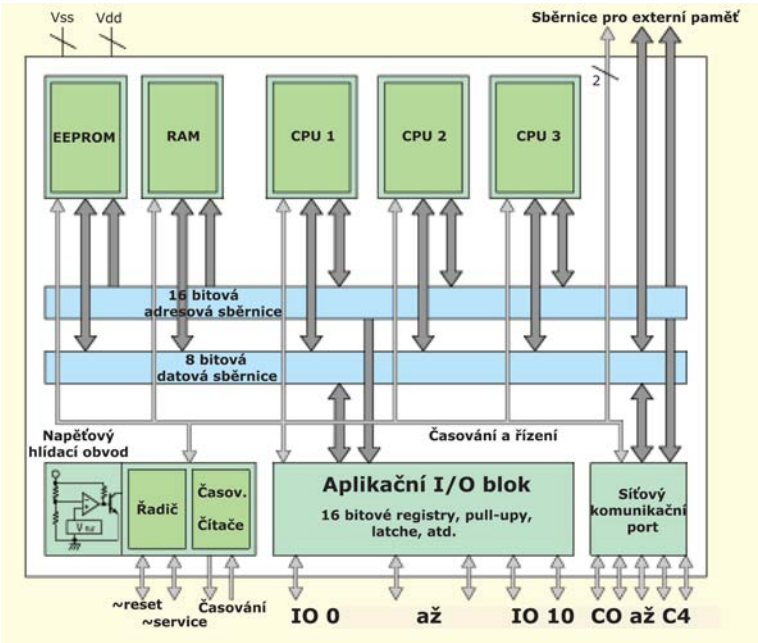
Části Neuron chipu	TMPN3120FE3M
CPU	8-bit CPU(3)
RAM	2,048 bytů
ROM	16,384 bytů
EEPROM	2,048 bytů
16-bit časovač / čítač	2-kanálový
Rozhraní pro připojení ext. paměti	Neobsahuje
Pouzdro	32-pin SOP
Maximální frekvence	20MHz

Tabulka 3: Popis Neuronchipu TMPN3120FE3M od Toshiby

Vnitřní struktura Neuron chipu je uvedena na obrázku [5]. Výměna dat mezi třemi

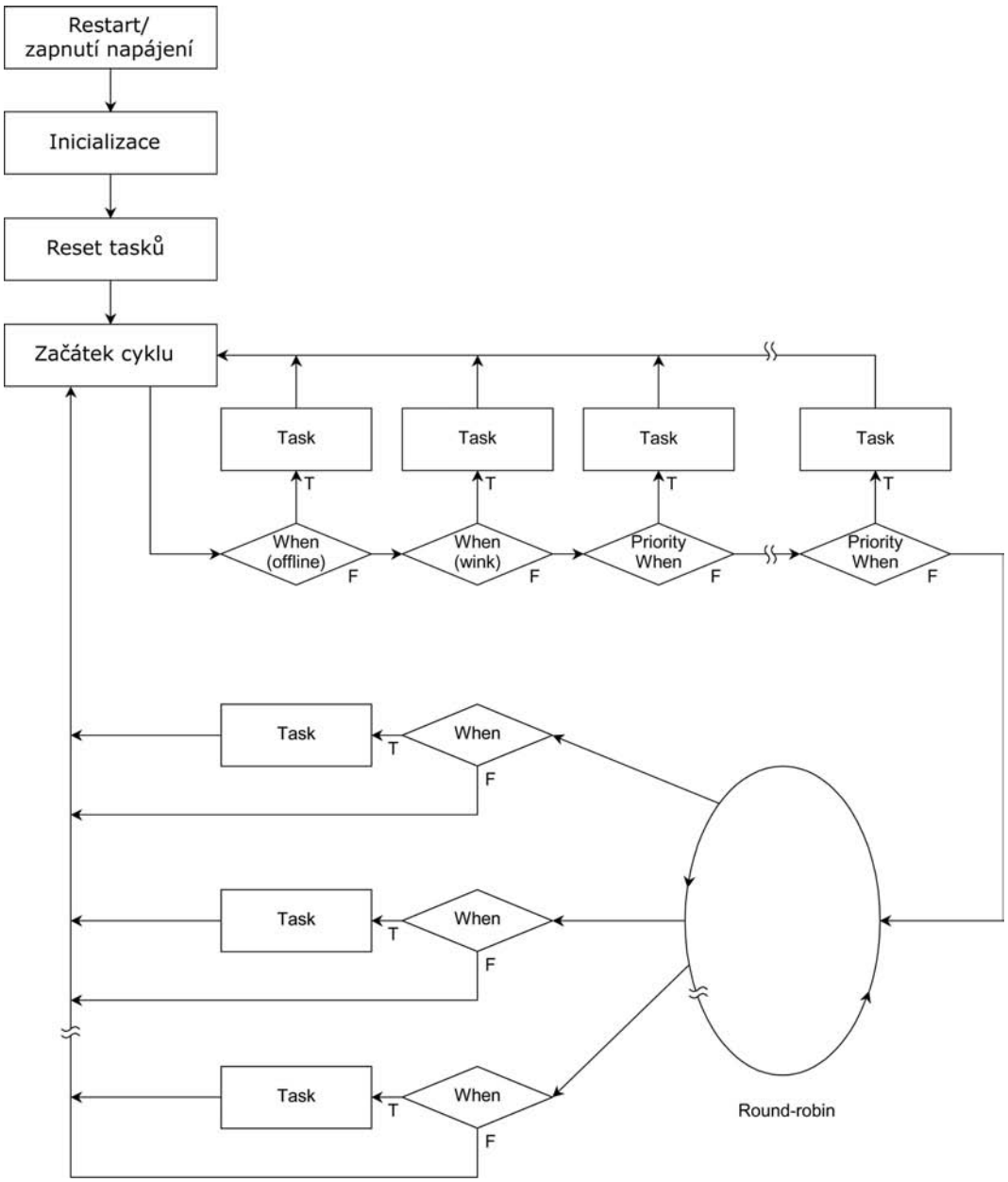
[6] TaskScheduler je jednoduchý operační systém pro řízení událostí

procesory je zprostředkována přes dva zásobníky. Procesor pro přístup na médium je spojen přes síťový zásobník se síťovým procesorem. Síťový procesor přijíma a vysílá data přes aplikační zásobník. V paměti je již zabudován jednoduchý operační systém a řada předdefinovaných funkcí. Z tohoto důvodu není nutné programovat komunikaci na sběrnici. Vše je zajištěno síťovým a sběrnicovým MAC[7] procesorem. Programátor má úlohu lehčí a může se plně věnovat uživatelskému programu. Operační systém podle programového kódu ukádá konfigurační a programové proměnné do paměti RAM či do EEPROM při použití speciálních instrukcí. Samotný program se ukládá jako vytvořený obraz programu od kompilátoru do paměti EEPROM. Paměť RAM slouží k zápisu proměnných u kterých se přepokládá rychlý přepis jejich hodnot. Pokud by se využilo paměti EEPROM dojde k rychlému vyčerpání počtu přepisovacích cyklů a pamět přestane fungovat. Pomocí speciálních direktiv v programovacím jazyce Neuron C je možné zapsat proměnnou do paměti EEPROM. Mezi vestavěné funkce v operačním systému patří i autorizace pomocí 64 bitové šifry. Významný vliv na funkci procesoru má TaskScheduler, protože Neuron chip pracuje podobně jako PLC v periodicky se opakujících cyklech. Všechny události jsou obsluhovány tak, jak ukazuje obrázek [6]. Pokud je Neuron chip po resetu či je zapnut elektrický proud, dojde k inicializaci. Po inicializaci jsou všechny úkoly (tasks) uvedeny do výchozího nastavení. Následují události podle jejich důležitosti. Provede se reakce na událost offline a wink. Následují prioritní události. Na závěr se provádí běžné události. Pokud nenastane žádná událost, zůstane v nekonečné smyčce než nastane událost, kterou obslouží a poté začne celý cyklus znovu. Podle toho je přizpůsoben i programovací jazyk resp. přístup k obsluze těchto událostí pomocí funkcí a direktiv. Bližší popis je uveden v kapitole 2.1.3.



Obrázek 5: Vnitřní struktura Neuron chipu

[7]MAC - (Medium Access Control) řízení přístupu k přenosovému médiu



Obrázek 6: Pracovní scan cyklus Neuron chipu

2.1.3 Programovací jazyk Neuron C

Samotný programovací jazyk Neuron C sje odvozen od jazyka ANSI C, ovšem je upraven pro funkce a fungování Neuron chipu. Programování pomocí Neuron C umožňuje rychlejší vývoj aplikací než programování u jednodušších procesorů, kde se používá převážně assambler. Lze využít téměř všech znalostí z jazyka ANSI C. Objekty Neuron C neobsahuje, ale obsahuje možnost vytváření tříd. Neuron C poskytuje další funkce nezbytné pro práci s LonWorks či výpočty. Nevýhodou Neuron C je omezenější repertoár datových typů než jaké známe z ANSI C a též výpočet s desetinou čárkou není jednoduchý. Jedná

se o využití funkcí pro převod do formátu desetinných čísel a následné dělení. Využívá se při tom třídy pro výpočty s desetinou tečkou. Bližší informace jsou v [14]. Též práce s událostmi je jiná. Vzhledem k práci Task Scheduleru je každá funkce spouštěna až událostí, kterou programátor vybere. Lze říci, že se jedná o událostmi řízený procesor. Následující příklad ukazuje spuštění funkce přepnutí led diody při události uplynutí nastavené doby časovače:

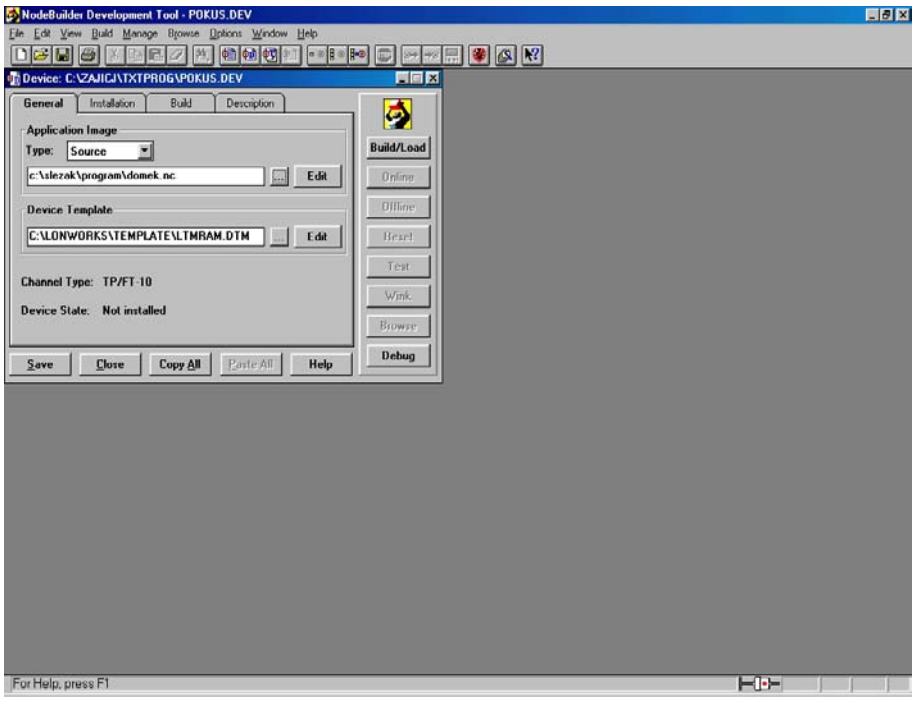
```
when ( timer_expires( timer_blink ) )
{
  switch_service_pin();
}


void switch_service_pin(void)
{
if(state_pin!=OFF)
  {
  state_pin=OFF;
  activate_service_led = FALSE;
  }
else
  {
  state_pin=ON;
  activate_service_led = TRUE;
  }
}
```

Když nastane událost uplynutí času nastaveného na časovači `timer_blink`, je odchycena pomocí `timer_expires`, která je v podmínce `when`. Nejčastěji se reaguje na události nové hodnoty proměnné pomocí `nv_update_occurs(jméno proměnné)`, na událost reset `when (reset)` a na událost wink `when (wink)`. Wink je událost, která slouží k indetifikaci zařízení. Například pokud je několik zařízení vedle sebe a nutnu zjistit jaké zařízení s daným Neuron ID je to správné. Vyvolá se nějakým programem událost wink pro zařízení s potřebným Neuron ID a dané zařízení se podle uvedené funkce u události wink projeví. Vytvořený nod v rámci diplomové práce rozsvítí nepřerušovaně červenou led diodu po dobu 5 vteřin po vyvolání události `wink`.

2.1.4 Vývojové prostředí Node Builder

Celé vývojové prostředí určené pro vývoj softwaru a test hardwaru pro nody v síti LonWorks se nazývá Node Builder. Jedná se o samotné programovací prostředí, testovací zařízení obsahující Neuron chip řady 3150, kartu do ISA slotu pro spojení s testovacím nodem a manuály. Samotný testovací nod je Neuron chip s procesorem 3150 a 16k RAM pro test programu. Výhodou toho typu procesoru je větší paměť RAM, která umožňuje testovat program bez nutnosti zápisu do paměti EEPROM. Výstup všech pinů Neuron chipu je vyveden na konektor. Na tento konektor je připojen plochý kabel. Na boku

boxu ve kterém je samotný nod umístěn jsou dvě LED diody pro různé funkce. Dále celé prostředí obsahuje kartu do ISA slotu v počítači. Karta primárně obsahuje transceiver FTT-10A na rozšiřující desce. Pro přepnutí komunikace Neuron chipu byla použita rozšiřující karta s diferenčním můstkem pro komunikaci v diferenčním módu. Závěrem celé obsahuje i samotné programovací prostředí, které umožňuje nahrát jednotlivé programy do procesoru, výběr transceiveru, rychlosti přenosu a mnoho dalších funkcí při překladu programu. Programovací prostředí neobsahuje debugovací nástroj a proto je potřeba všechny testy programového kódu provádět pomocí signalizace na uvedených LED diodách.



Obrázek 7: Ilustrační obrázek prostředí Node Builderu

2.2 Vhodné zařízení pro nod

2.2.1 Výběr technologie

Pro správnou funkci nodu je třeba vybrat i vhodné zařízení k ovládání ventilu radiátoru. Z dostupných technologií na trhu bylo nutné rozhodnout, zda zvolit ovládání typu OTEVŘENO/ZAVŘENO či postupné uzavírání umožňující nastavit ventil do pootevřeného stavu. Vhodnější bylo postupné uzavírání. Vybraný způsob ovládání nabízí motorový či termokeramický pohon. S ohledem na spotřebu proudu byl vybrán termokeramický pohon, který funguje na principu rozpínání keramické hmoty při jejím zahřívání pomocí tranzistoru umístěného na jejím povrchu.

Obrázek 8: Fotografie vytvořeného nodu s termohlavicí

2.2.2 Zvolená termohlavice

Celá termohlavice je vybavena velmi jednoduchou elektronikou umístěnou na desce malých rozměrů umístěné blízko její větrací mřížky. Je zde i LED dioda pro diagnostiku. Oranžová barva LED diody signalizuje, že je keramická hmota zahřívána a tím uzavírá ventil. Termohlavice je spojena s Neuron chipem a napájena pomocí telefonního kabelu. Ten je vybaven standardním telefonním konektorem RJ11. Začátek telefonního kabelu je zasunut do zástrčky v nodu a konec je zapojen v elektronice termohlavice. Napájecí napětí je 24V stejnosměrných a jako řídící napětí je použito 5V stejnosměrných. Poslední žíla telefonního kabelu je nezapojena.

Pro správnou funkci celého nodu a regulátoru byly naměřeny časy uzavření a otevírání ventilu. Dále byl měřen proud protékající hlavicí při jednotlivých řídících napětí. Podle naměřených dat byl vybrán vhodný způsob řízení termohlavice pomocí PWM[8].

2.3 Konstrukce LonWorks nodu

2.3.1 Napájecí část

Procesor je napájen napětím 5V stejnosměrných. Pro napájení byl zvolen DC/DC měnič 24V/5V s výstupním proudem 500 mA. Umožňuje velmi jednoduchou instalaci. Je dodávám v černé krabičce U-DI1. Pro stabilizaci DC/DC měniče byly použity doporučené keramicke kondenzátory uvedáděné výrobcem. Pomocí keramických kondenzátorů vypočtených parametrů byla zajištěna stabilizace napájení samotného procesoru. Pomocí radiálního kondenrátoru na přívodním vedení napájecího napětí je zajištěna

[8]Pulse Width Modulation - Pulzně Šířková Modulace

stabilizace napětí pro přívodní vedení. Kapacita radiálních kondenzátorů je dimenzována pro přívodní kabel délky až 2 metry.

2.3.2 Procesorová část

Procesor umožňuje být taktován v širokém rozmezí frekvencí od 0,625 MHz až po 20MHz. Při volbě krystalu pro frekveci procesoru byla vybrána frekvence 5MHz. Na této frekvenci operuje transceiver FTT-10A. Zapojení resistorů a kondenzátorů bylo voleno podle doporučených hodnot od výrobce Neuron chipu. Propojení procesoru resp. Neuron chipu s transceiverem je přes dva výstupní piny. Dále je zajištěna stabilizace napětí pro sběrnici pomocí 2 radiálních kondenzátorů. Podrobnější schéma v příloze [A.1.1]. Neuron chip je přes piny CP0 a CP1 spojen s transceiverem. Na straně transceiveru se jedná o piny T1 a T2. Jako napěťový hlídací obvod byl využit chip TL7705A.Celé napájení je chráněno diodou proti přepólování napětí. Indikace správně zapojeného napájecího napětí je LED dioda modré barvy. Diagnostický a indiakčním prostředkem je červená LED dioda připojená na servisním pinu procesoru. Spojení procesorové a výkonné části je umožněno pomocí dvouřadých zlacených lámací kolíků. Dodaný Neuron chip byl umístěn v pouzdře 32-pin SOP.

2.3.3 Výkoná část

Výkonou část lze pomyslně rozdělit do dvou částí. První část zajišťuje napájení termohlavice. Druhá část zajišťuje její ovládání. Vzhledem k tomu, že obě části využívají rozdílné napětí, bylo nutné použít galvanického oddělení pomocí optočlenu SFH610A-1. Malý prostor pro umístění ovládací části a po opravě návrhu plošného spoje umožňoval použít pouze napěťový dělič pro vytváření ovládacího napětí. Podrobnější schéma v příloze A.1.2.

2.3.4 Softwarová část

Celý program se skládá ze tří částí. V první části jsou definovány pragma, konstanty a definice funkcí. Pragma umožňuje potřebné nastvení překladu. Pro lepší orientaci a možnost pozdějších změn jsou definovány konstanty.

```
#pragma set_node_sd_string "@0,1,3.Test of config and SCPT"
#pragma enable_sd_nv_names

#define ON  0
#define OFF 1

#define OCCUP FALSE
#define NON_OCCUP TRUE

#define FULL_TIME 180U
#define LIMIT_TIME 65000
#define NIGHT_TEMP_LIMIT 20
```

```
void spocti_x_diferencne(void);
void spocti_x_interacne_lepsi(void);
void switch_service_pin(void);
```

Následuje definice vstupních a výstupních síťových proměnných. Je vhodné volit intuitivní názvy těchto proměnných. Při pozdějším zapojení do sítě LonWorks nebude docházet ke špatnému spojení SNVT proměnných. Každá SNVT proměnná umožňuje vložit do své definice i stručný popis. Není vhodné psát dlouhý popis. Každý znak se ukládá do paměti procesoru a zabírá paměťový prostor na úkor výkonného programu. Vstupní proměnná může nabývat předdefinované hodnotu zadané programátorem při jejím vytvoření.

```
network output sd_string("@1|1.") SNVT_temp_p nvoSpaceTemp;
network output sd_string("@1|2.") SNVT_hvac_status nvoUnitStatus;

network input sd_string("@2|1.") SNVT_temp_p nviSpaceTemp = 2000;
network input sd_string("@2|2.") SNVT_temp_p nviSetPoint = 2000;
```

Při výběru časovačů, je možné vybírat ze dvou typů časovačů. Samospoustěcí časovač či časovač bez opakovaného spouštění. Opakované spouštění časovače se umožňuje direktivou `repeating` umístěnou před jeho název. Po uplynutí času se časovač znovu sám nastaví na zadaný čas a spustí se znovu. Po uplynutí času vyšle o tom událost, na kterou lze reagovat programovým kódem. Bez této direktivy se časovač zastaví po uplynutí času.

```
stimer repeating sample_time;
stimer           timer_working = 10;
```

Adresace výstupních či vstupních pinů je velice jednoduchá, jak ukazuje následující příklad.

```
IO_0 output bit pulse_level = ON;
```

Při adresaci pinu 0, se definuje jako výstupní a jedná se o jeden bit. Při definici je přiřazena hodnota ON, která byla v definici konstant nastavena na hodnotu `TRUE`.

Po řadě definic v první části, je v druhé části je samotný program. Jedná se o 14 `when` funkcí, které pracují s časovači, ošetřují vstupní proměnné,volají funkce regulátoru a reagují na systémové události.

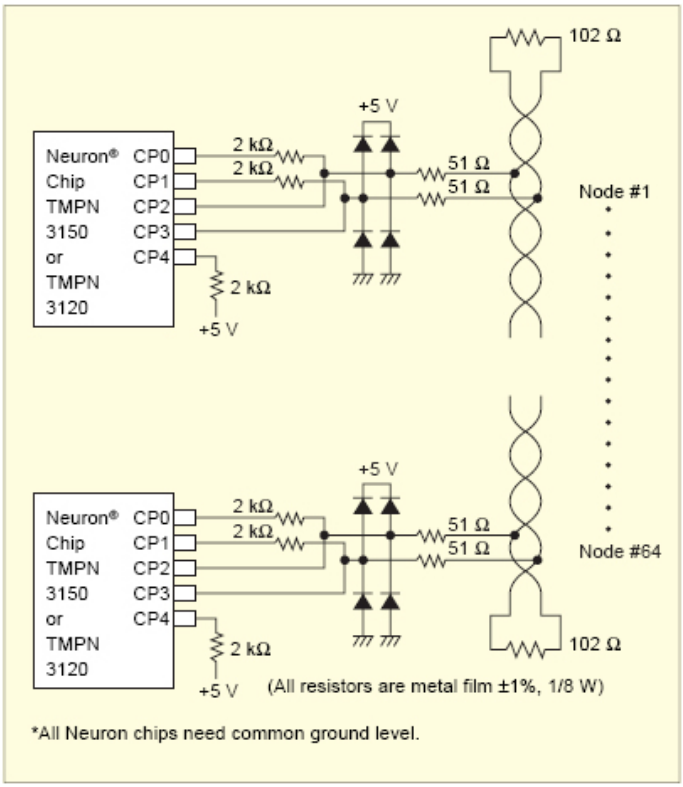
```
when ( nv_update_occurs( nviSetPoint ) )
{
if(tempNeededEeprom!=nviSetPoint)
  {
  tempNeededEeprom=nviSetPoint;
  }
}
```

Příklad ilustruje ochranu proti zbytečné aktualizaci proměnné. Využívá se funkce `nv_update_occurs`, která reaguje na vysílání nové hodnoty proměnné uvedené v jejím parametru. Vstupní proměnná může nabývat i stejné hodnoty. Pokud se hodnota ve vstupní proměnné liší od hodnoty uložené, dojde k její aktualizaci.

V třetí části je definice funkcí regulátoru. Byly naprogramovány dva druhy módy (způsoby) regulace. Diferenční mód (implicitní) a skokový mód. Diferenční mód vytváří velikost regulačního zásahu podle rozdílu teplot požadované a v místnosti. Vypočtená hodnota je násobena regulační konstantou určené k jemnějšímu nastavení regulátoru. Skokový mód upraví regulační zásah podle znaménka rozdílu teplot v místnosti a požadované. Následně je násobeno konstantou o hodnotě 30. Podobnější popis v kapitole 2.4. Při vývoji softwaru byl přibližně dodržen profil organizace Lonmark: VAV Controller (VAV).

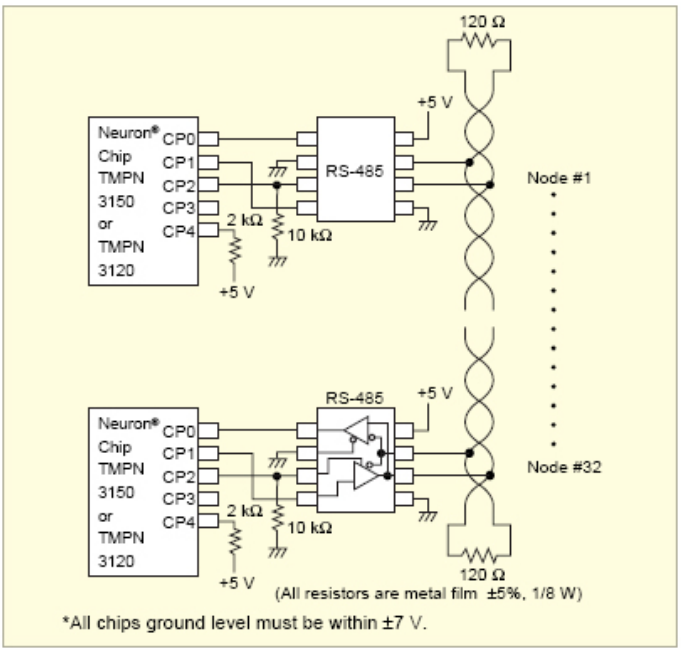
2.3.5 Přepnutí režimu komunikace

Neuron chipy mohu být v různých režimech komunikace. Diferenční mód využívá ke své komunikaci diferenčního můstku 9. Skládá se ze čtyř diod zapojených do můstku a rezistorů stanovených výrobce. Další módy jsou Sigle-Ended mód 10 a Special-Purpose mód 11. Použitý Neuron chip byl výrobcem nastaven do diferenčního módu. Pro komunikaci s transceiverem FTT-10A bylo nutné přepnout mód neron chipu z diferenčního módu do Single-Ended módu. Před přepnutí komunikace byl na nepájivém poli sestaven diferenční můstek podle referenčního manuálu Neuron chipu. Druhý diferenční můstek byl umístěn na rozšiřující kartě v počítači. K tomu byla využita rozšiřující karta do počítače vytvořená Martinem Linhartem v rámci jeho diplomové práce [10]. Diferenční můstky byly spojeny kroucenou dvoulinkou. Přes tuto sběrnici byl daný Neuron chip spojen s počítačem. Komunikace byla přepnuta pomocí programu `Nodeutil`, který je umístěn na přiloženém DVD. Pro funkčnost programu je potřeba operačním systémem MS-DOS. Na stránkách firmy Echelon je program `Nodeutil` novější varianty, který nepracoval správně. Pokud stažený program nebude korektně fungovat, je možné využít program `Nodeutil` na přiloženém DVD či na CD přiložené k diplomové práci Martina Linharta. Podrobný postup je popsán v [10]. Komunikaci na sběrnici je nutné monitorovat pomocí digitáního osciloskopu, aby bylo možno vidět způsob komunikace před přepnutím komunikace a po jeho přepnutí. Může nastat případ, kdy program potvrdí přepnutí komunikace, ovšem k jejímu fyzickému přepnutí nedošlo. Proces je nutné opakovat. Další indikací správného provedení je ztráta spojení. Neuron chip již pracuje v Sigle-Ended módu, proto nedokáže komunikovat s druhým Neuron chipem přes diferenční můstek. Pokud nelze detekovat Neuron chip při použití programu `Nodeutil`, lze použít reset masteru. Poté vždy došlo ke správné detekci Neuron chipu.



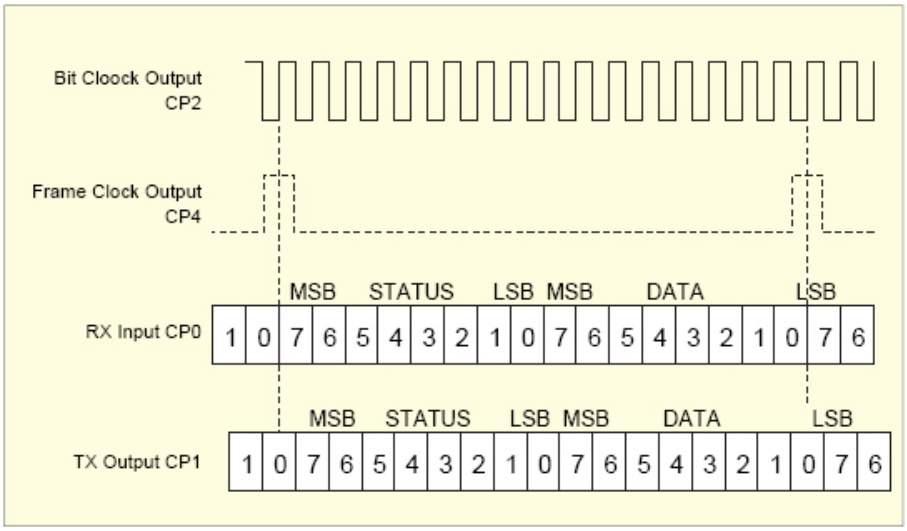
Obrázek 9: Zapojení Neuron chipu v diferenčním módu

Sigle-Ended mód umožňuje maximální délku sběrnice 1200 metrů.



Obrázek 10: Zapojení Neuron chipu v Sigle-Ended módu

Neuron chip může být nastaven v Special-Purpose módu. Tento mód umožňuje přenos většího množství dat mezi Neuron chipem a transceiverem. Často se používá při komunikaci po napájecí síťi.

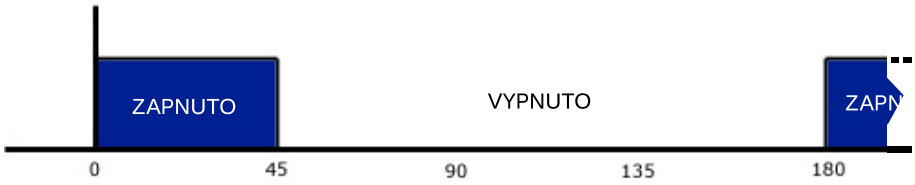


Obrázek 11: Komunikace Neuron chipu v Special-Purpose módu

2.4 Regulátor

2.4.1 Vhodné ovládání

Při připojení napětí se začne keramická hmota termohlavice maximální rychlostí ohřívat resp. roztahovat a po jejím odpojení začne pomalu chladnou resp. se smršťovat. Proto bylo zvoleno ovládání regulace termohlavice pomocí PWM. Pro šířku celého pulsu byl stanoven počet 180 dílků. Vypočtený zásah regulátoru je přičten k aktuálnímu počtu dílků šířky pulzu z maximálních 180. Pokud je regulace například 20 dílků, tak je puls široký 1/9 času trvání maximálních 180 dílků.



Obrázek 12: Pulsně šířková modulace v nodu termohlavice

Vypočtený zásah regulátoru je přičten k nynější šírce pulsu a jeho šířka udává délku čas zapnutí/vypnutí napětí na termohlavici a tím i velikost zavření/otevření ventilu. Podle naměřených časových konstant termohlavice byl volen přednastavený čas. Naměřený čas uzavírání termohlavice byl 3 minuty a čas jejího otevírání 15 minut, proto byl volen čas 180 vteřin resp. 3 minuty jako délka celého pulzu. Měření obou časů byla

několikrát opakována. Změřené časy udávají čas uzavření vetilu ze stavu plného otevření a z plného zavření na maximální otevření.

2.4.2 Softwarové ovládání

Neuron chip ovládá termohlavici pomocí pinu `IO 0`. Tento pin je spínán a vypínán podle času určovaného regulátorem. Možnost řízení chodu regulátoru, lze ovlivnit pomocí osmi vstupních SNVT proměnných. Důležitou vstupní proměnnou je `nviSpaceTemp`, která obsahuje teplotu v místnosti, kde je umístěn regulátor. Další důležitou vstupní proměnnou je požadovaná teplota v místnosti `nviSetPoint`. Následují nepovinné proměnné, které slouží ke komfortnější či přesnější funkci regulátoru. Jedná se o `nviSampleTime`, která udává čas vzorkování pro výpočet regulačního zásahu. Mód regulátoru je ovlivněn pomocí proměnné `nviType`. Čas než regulátor vyhodnotí poruchu v komunikaci mezi jím a teploměrem udává proměnná `nviAlarmTime`. Pokud je tato proměnná rovna nule, je tato funkce vypnuta. Další výpis funkcí včetně jejich funkcí a popisu je uveden v manuálu k termohlavici [13]. Výstupem regulátoru jsou tři SNVT proměnné. První udává teplotu, kterou regulátor přijímá. Druhá je objekt obsahující vnitřní stavové proměnné jako intenzita topení. Obsahující inforamce zda je regulátor v alarmu, v jakém módu topení se regulátor nachází a další.

2.4.3 Skokový regulátor

Skokový rugulátor upraví regulační zásah podle znaménka rozdílu teplot a je násoben konstantou o hodnotě 30. Matematický vzorec výpočtu:

$$y_{k+1} = y_k - sign(T_{mistnosti} - T_{pozadovana}) \cdot 30$$

2.4.4 Diferenční regulátor

Diferenční mód (implicitní) vytváří velikost regulačního zásahu podle rozdílu teplot v místnosti a teploty požadované. To vše je jěště násobeno regulační konstantou pro přesnější chování regulátoru. Matematický vzorec výpočtu:

$$y_{k+1} = y_k + \alpha \cdot (T_{mistnosti} - T_{pozadovana})$$

2.5 Zapojení nodu termohlavice a Wago do sítě LonWorks

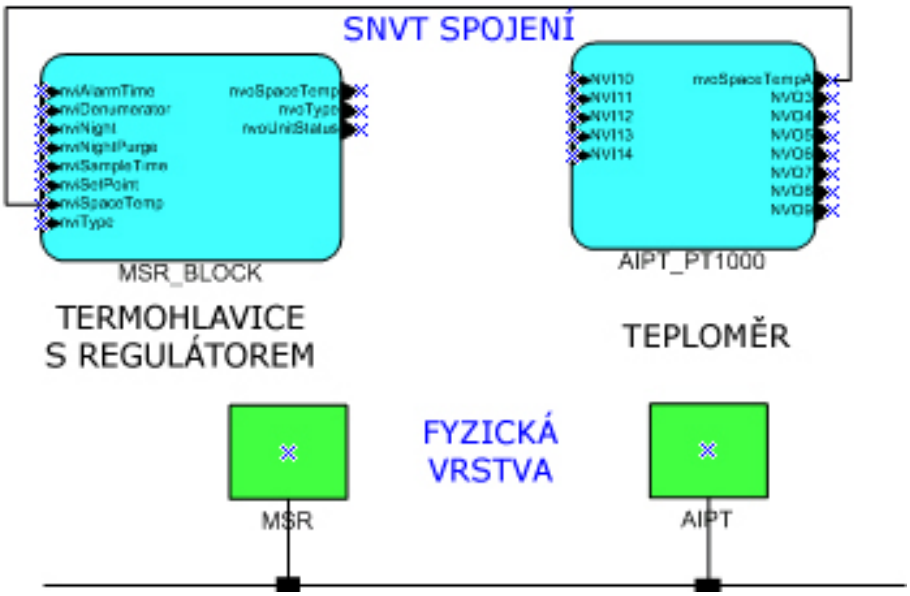
2.5.1 Node Builder

Při zakládání projektu v programu Node Builder je potřeba vybrat správný model procesoru, správný mód komunikace s procesorem, jeho frekvneci a nastavit další potřebné položky. Tímto nastavením uživatele provede velmi intuitivní průvodce. Pokud je již vytvořen program, je překladačem přeložen do souboru, který se následně nahraje do paměti procesoru. Samotný program se nahrává pomocí Node Builderu do paměti EEPROM procesoru, proměnné se nahrávají do paměti RAM, pokud nebylo použito direktivy *eeprom* pro zápis do paměti EEPROM. Konfigurační proměnné se též nahrávají do paměti

EEPROM. Podrobnější popis prostředí Node Builderu a návod na programování, včetně překladu a nahrání programu do procesoru naleznete v příloze D.4. Takto připravený nod s nahraným programem lze zapojit do sítě LonWorks. Postup připojení naleznete v příloze D.1

2.5.2 Termohlavice v síti LonWorks

Po nahrání programu do Neuron chipu a fyzickém připojení nodu do sítě LonWorks následuje konfigurace sítě. Následující obrázek [13] ukazuje zapojení teploměru v místnosti s nodem resp. s termohlavicí. Pomocí konfiguračního nástroje LonMaker je nutné spojit výstupní SNVT proměnnou udávající teplotu v místnosti se vstupní proměnnou `nviSpaceTemp`, aby nod správně komunikoval se sítí a mohl správně fungovat. Poté se tato konfigurace uloží do směrovací tabulky teploměru, tak i do nodu termohlavice. Distribuce konfigurace je proveden neprodleně pomocí programu LonMaker po provedení změn v konfiguraci, pokud je nastaven v módu *Onnet*. V módu *Offnet* je nutné uložit změny do konfigurace sítě uložením projektu v programu Microsoft Visio. Následně je vhodné provést resynchronizaci databáze LNS, aby nedocházelo k chybám při jejím používání. *LonMaker* → *Resynchronize*. Celý nod s regulátorem a termohlavicí je nyní kompletně zapojen do sítě LonWorks.



Obrázek 13: Zapojení nodu s termohlavicí do sítě LonWorks

Všechny potřebné postupy ke správné konfiguraci celé sítě LonWorks, zařízení a konfigurace PLC Wago jsou umístěny v příloze D.

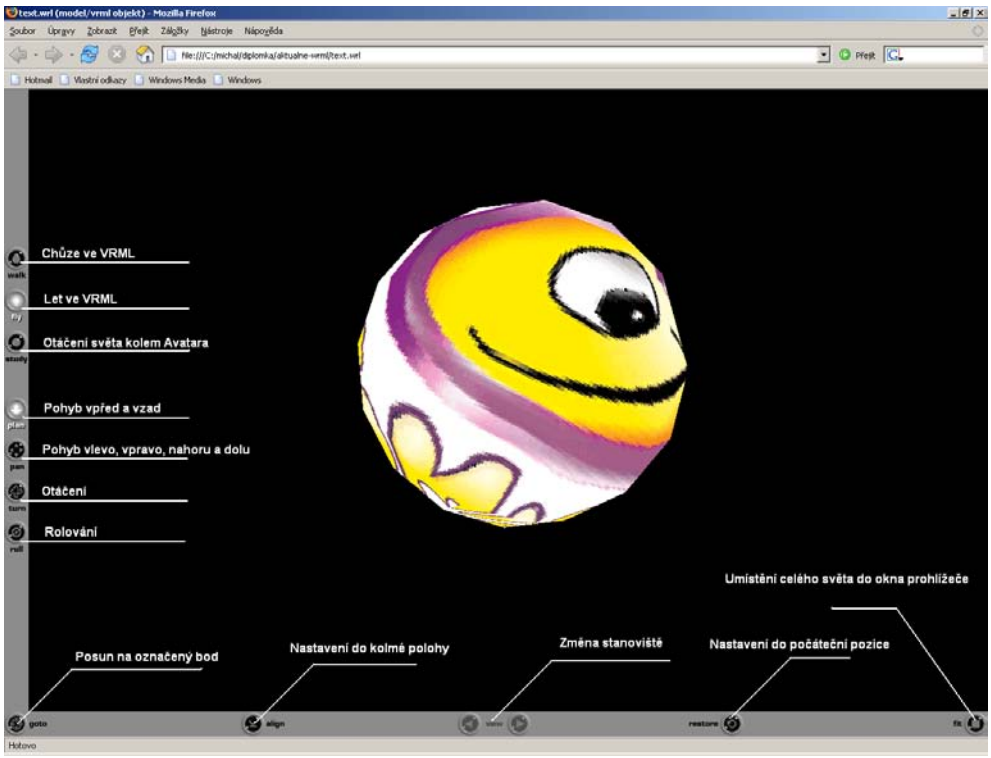
3 VRML a internetová prezentace

Kapitolu lze rozdělit do dvou sekcí. První a dominantní část je věnována jazyku VRML[9], vytvořenému virtuálnímu modelu budovy a spojení této virtuální reality skrze Javu s reálným modelem. Menší část je věnována webovému portálu, který představuje zdroj informací k projektu administrativní budovy a též složí k zakomponování virtuálního světa do webové prezentace.

3.1 Úvod do VRML

Jazyk VRML 97 je mezinárodní normou ISO pro popis statických a dynamických světů. Vytváří snadnou cestu, jak umístit 3D objekty do webových stránek, vytvořit interaktivní průchody světy či objekty. Mezi zakladatele ISO normy patří i autor knihy VRML 97, Doc. Ing. Jiří Žára, CSc. Jazyk VRML definuje způsob zápisu virtuálního světa do formy textového souboru, který má přesně stanovenou strukturu. Vychází z grafické knihovny OpenGL a následné knihovny OpenInventor, která se stala základem VRML. Vytvořený virtuální svět v textovém souboru je pak reprezentován pomocí prohlížeče virtuálního světa. Může se jednat o prohlížeč Cortona od firmy Parallelgraphics, Cosmo Player od firmy Platinum Technology, BS Contact od firmy Bitmanagement Software a mnoho dalších. Virtuální svět je pomocí zvoleného prohlížeče zobrazován v okně webového prohlížeče. Ten k vykreslování virtuálních světů používají vykreslovače (renderery) jako jsou DirectX, OpenGL či softwarové vykreslování. Vedle pouhého vykreslování, nabízí prohlížeče pohyb a ovládání ve virtuálním světě pomocí ovládacích prvků umístěných po jejich stranách. Další vývoj v jazyka VRML nabízí konzorcium Web3D, které se stará o rozvoj 3D grafiky na internetu a jejich norma X3D.

[9] Virtual Reality Modeling Language



Obrázek 14: Prohlížeč Cortona s ovládacími prvky

3.1.1 Technologie VRML

Celý virtuální svět je zapsán ve strukturovaném textovém souboru. První řádek je povinný a nelze ho měnit. Nelze jej ani posunout na druhý řádek. Obsahuje sdělení, že se jedná o VRML dokument a kódování UTF8. Další řádky již jsou volitelné a obsahují zápis virtuálního světa podle potřeb tvůrce. Celý svět je po načtením prohlížečem strukturován do DOM[10] modelu. Jedná se o stromovou strukturu s definovanými uzly, které mají své rodiče, sourozence a potomky. Tato struktura umožňuje flexibilnější vytváření a ovládání virtuálního světa. VRML umožňuje i import již vytvořených světů do vytvářeného světa. Samozřejmostí je možnost využití všech výhod, které poskytuje internetvá technologie, jako jsou odkazy, URL[11], obrázky či textury umístěné na jiných serverem atp.. VRML umožňuje vytvořit prakticky jakýkoliv předmět pomocí základních objektů a tento objekt animovat pomocí dynamický uzlů a událostí. Základní objekty jsou:

- Box - krychle
- Con - kónus
- Cylinder - válec
- Sphere - koule

[10] Document Object Model
[11] Uniform Resource Locator

Další možností vytváření objektů je použít ploch, které se dokáží tvarovat podle stanovených souřadnic, či lze přímo zadáváním souřadnic a normál vytvářet jakákoliv tělesa. Každý takový objekt má kromě svých rozměrů i vlastnosti, které ilustruje jednoduchý příklad koule potažené texturou.

```
#VRML V2.0 utf8
...
Shape { #defince objektu koule
  geometry Sphere {radius 1}    # koule o~polomeru 1 metr
  appearance Appearance {     # vlastnosti objektu koule
    material Material {       #material
      diffuseColor 0 0 0       #zakladni slozeni barvy

      #barva povrchu zesvetlovana jasem prostoru
      ambientIntensity 0
      emissiveColor 0 0 0       #svitiva barva povrchu
      specularColor 0 0 0       #jakou barvu povrch odrazi
      shininess 0.2           #ostrost odrazu
      transparency 0.2         #pruhlednost objektu
    }
    texture ImageTexture {     # objekt je potazen texturou
    repeatS TRUE           #opakovani textury ve vodorovnem sveru
    repeatT TRUE           #opakovani textury ve svislem sveru
    url "smile.jpg"         #adresa textury - muze byt i url odkaz
    }
  }
}
...
```

Obrázek 15: Příklad koule potažené texturou

Takto definovaný objekt neobsahuje transformace pro určení polohy či natočení. Proto se objekty uzavírají do uzlů *Transform*, které obsahují funkce na transformaci polohy, rotace a zvětšování. Všechny tyto transformace se distribuují do potomků a do potomků těchto potomků. Z tohoto zápisu je patrná výhoda stromové struktury VRML zápisu. Jednoduchý příklad uzlu *Transform* je ilustrován níže.

```
#VRML V2.0 utf8
Transform {
  translation 0 0 0     # pozice od stredu sveta VRML
  rotation 0 1 0 1.571  # rotace kolem osy Y o PI/2
  scale 1 2 1       # dvojnasobne zvetseni v ose Y
  children [
  Shape {
  ...
  #defince objektu koule
  ...
      }
    }
  }
  ]
}
```

Takto vytvořené světy nevyvolají dojem reálného světa. Proto se do virtuálního světa zavádí světlo, zvuky či pozadí. Světla lze vytvořit jako bodová, šířící se z jednoho bodu do všech stran (*Pointlight*). Dalším zdrojem je bodové světlo s možností směrování světelných paprsků (*Spotlight*) a v poslední řadě směrové světlo v podobě rovnoběžných paprsků (*Directionallight*). Světlo se nešíří ve všech místech stejně, ale jeho útlum je dán zabudovanými rovnicemi v prohlížečích VRML světů a normou. Významnou vlastností je pozadí, které může navodit pocit většího prostoru než je skutečnost. Zvuk významně zvyšuje dojem reálného světa. Jeho šíření též podléhá výpočtu podle rovnic. Další vlastností je možnost vytvoření iluze mlhy (*Fog*) resp. pocitu mizejích předmětů, které jsou již daleko z dohledu. Bližší informace o dalších uzlech a tvorbě VRML jsou uvedeny v [12]. Vzhledem k tomu, že se mnoho objektů opakuje, lze si vytvářet knihovny obsahující již vytvořené objekty. Poté stačí vložit tyto objekty z knihovny od vytvářeného světa a nastavit příslušné parametry. Takto lze stavět knihovny z objektů z jiných knihoven a vytvářet tak bohatší objekty. Následující příklad ukazuje použití objektu okno z knihovny *komponenty.wrl*.

```
==== definice okna ====
PROTO UplneOkno
  [
    #vstupni posun v prostoru
     field SFVec3f     celek_posunuti 0 0 0
     #vstupni natoceni v prostoru
     field SFRotation  celek_natoceni 0 0 0 0
     #vstupni zvetseni
     field SFVec3f  celek_zvetseni 1 1 1
  ]
  {
    EXTERNPROTO Futro [ # pripojeni definice ramu okna
     field SFVec3f        s_posunuti
     field SFRotation  s_natoceni
     field SFVec3f      s_zvetseni]
     "komponenty.wrl#Futro"
```

```
     #ukazatel na objekt ramu dveri v knihovne komponenty.wrl

    EXTERNPROTO Okno [  # pripojeni definice samotneho okna
     field SFVec3f        s_posunuti
     field SFRotation  s_natoceni
     field SFVec3f      s_zvetseni
     field SFFloat        s_pruhlednost]
     #ukazatel na okno v knihovne komponenty.wrl
     "komponenty.wrl#Okno"
    Group {
        children [
          Transform {
          #spojeni posunu s~promennou posunu
          translation IS celek_posunuti
          #spojeni rotace s~promennou rotace
          rotation IS celek_natoceni
          #spojeni zvetseni s~promennou zvetseni
          scale IS celek_zvetseni
          children [
            #umisteni objektu ramu dveri v prostoru bez parametru
            Futro {}
            #umisteni okna
            Okno {s_posunuti 0 .05 0 s_pruhlednost .6}
          ]
        }
      ]
    }
  }
====== použití okna v~jiném objektu
PROTO StenaSOknem
[
  field SFVec3f    s_celek_posunuti 0 0 0
  field SFRotation s_celek_natoceni 0 0 0 0
  field SFVec3f    s_celek_zvetseni 1 1 1
]
{
  EXTERNPROTO UplneOkno
  [
     field SFVec3f     celek_posunuti
     field SFRotation  celek_natoceni
     field SFVec3f  celek_zvetseni
  ]
  "celky.wrl#UplneOkno" #UplneOkno
  ...
  Group {
     children [
      Transform {
      ...
```

```
      children [
        UplneOkno {celek_posunuti 0 1 0 celek_zvetseni 1.85 1.36 1}
        StenaProOkno {}
      ]
      }
    ]
  }
}
```

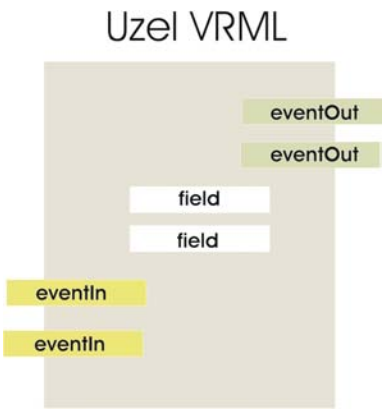
Pokud procházíme již vytvořeným světem, je nutné si uvědomit, kdo zastupuje uživatele ve virtuální světě. Postava zastupující uživatele se nazývá AVATAR. Avatar je definován poloměrem kružnice opisující šířku jeho těla, výšku celého těla a výšku překážek, které dokáže avatar překročit či na ně nakročit. Další vlastností Avatara je umístění svítilny resp. zdroje světla na čele hlavy, která sloužící k osvětlení objektů umístěných před ním. Pokud jsou již ve světě obsaženy světelné zdroje, je vhodné tuto lampu vypnout, jinak negativně ovlivňuje světelné efekty na objektech. Lze určit na jakou vzdálenost Avatar vidí nebo přesněji řečeno na jakou vzdálenost se mají jěště objekty vykreslovat. Nastavitelná je rychlost avatarova pohybu a je možné omezit možnosti pohybu ve virtuálním světě.

```
NavigationInfo{
avatarSize [0.10, 1.35, 0.45] #rozmery Avatara
headlight TRUE # rosviceni svitilny Avatara
visibilityLimit 50 #viditelnost na 50m
speed 0.5 # rychlost pohybu 0.5m/s
type ["WALK"] # pohyb pouze pomoci chuze
}
```

Pokud je svět rozlehlý nebo je v něm několik zajímavých míst pohledu, lze nadefinovat tzv. *ViewPoints* fungujících jako teleporty. Ve virtuálním světě může být mnoho takových bodů. Každý bod má své jméno,pozici a orientaci Avatara v něm.

```
Viewpoint {
description "Pozice v~prezentacni mistnosti"
position -4.5 4.55 2
orientation 0 1 0 3.141
}
Viewpoint {
description "Pozice v~horni kancelari"
position 4.5 7.55 2
orientation 0 1 0 3.141
}
```

Takto vytvořený svět je již možné procházet. Interakce mezi objekty a Avatarem zatím není žádná. Proto je definována skupina objektů, které jsou schopny reagovat na akce od Avatara a poskytovat je jako události. Pro příjem události je v každém těle objektu volitelné pole s názvem *EventIn* a pro poskytování události je to pole *EventOut*.



Obrázek 16: Parametry uzlu VRML

Celý proces vyvolání události, zpracování události a směrování na cíl ukazuje diagram na obrázku [17]. Je na něm pět uzlů, které si postupně předávají události. Ty mohou být různého typu, při jejich cestě dynamickým řetězcem dokonce často dochází k několika změnám typu předáváné události. Na počátku je prvek detekující dynamickou událost. Jedná se o čidla, která reagují na různá chování Avatara. Dále následuje prvek, který rozhodne o tom, zda byly splněny všechny podmínky pro spustění dynamické akce. K tomu slouží uzel *Script*. Následuje časovač, který je zodpovědný za časový průběh akce. Ovšem může vhodně měnit i dynamiku děje. Následuje prvek, který podle předem definovaných počátečních a koncových hodnot průběžně vypočítávat nové hodnoty. Na konci řetězce je cíl, na němž je dynamická akce viditelná. Výsledné hodnoty předchozího uzlu se do toho cíle zapíší. Například se zapíše změněná barva povrchu objektu.

Obrázek 17: Logické schéma obecné dynamické akce

Uvnitř těla skriptu lze pracovat s jakýmkoliv uzlem rodiče skriptu. Skript je také uzel, proto může vytvářet události a dále je směřovat do jiných uzlů. V tělě skriptu se mohou použít funkce JavaScriptu či volat třídy (class) programovacího jazyka Java. Příklad objasňuje použití skriptu na otevírání dveří pomocí JavaScriptu.

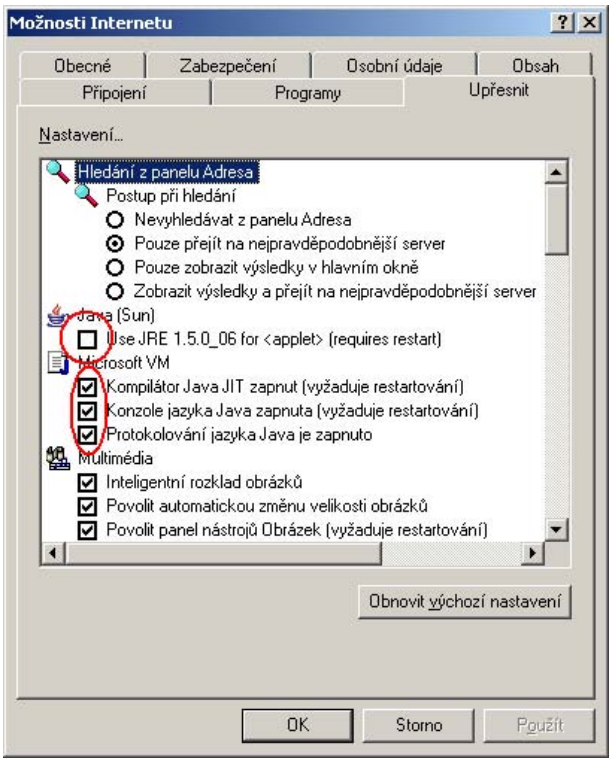
```
PROTO DverePrezentacka
...
EXTERNPROTO DeskaDveri
...
EXTERNPROTO Klika
...
   Group {
    children [
```

```
      Transform {
      translation  0 .1 0
      children [
        DEF PREZENTDOOR Transform { #definice dveri
          children [
            DEF DOORTOUCH TouchSensor {} #senzor dotyku
            DeskaDveri {}
            Klika {posunuti .347 1 0}
        ]
        }
        DEF SWITCHDOOR Script { #definice skriptu
        field SFBool stav FALSE #vnitrni stav
        field SFNode dvere USE PREZENTDOOR #ukazatel na dvere
        eventIn SFBool switchdoor #udalost spoustejici funknci
        eventOut SFRotation rotace #vystupni hodnota
        eventOut SFVec3f posun #vystupni hodnota
        url "javascript:
        function switchdoor(hodnota) #funkce otevirajici dvere
        {
          if(hodnota)
          {
            if(stav)
            {
              stav=!stav; #zmena stavu dveri a~nastaveni pozice
              rotace = new SFRotation(0,1,0,-1.571);
              posun = new SFVec3f(-0.43,0,0.45);
            }
            else
            {
              stav=!stav;#zmena stavu dveri a~nastaveni pozice
              rotace = new SFRotation(0,1,0,0);
              posun = new SFVec3f(0,0,0);
            }
          }
        }
        "
        }
      ]
      #distribuce udalosti ze senzoru do skriptu
      ROUTE DOORTOUCH.isActive TO SWITCHDOOR.switchdoor

      #distribuce vyslednych hodnot ze skriptu do dveri
      ROUTE SWITCHDOOR.rotace TO PREZENTDOOR.set_rotation
      ROUTE SWITCHDOOR.posun TO PREZENTDOOR.set_translation
      }
    ]
  }
}
```

Pokud se používájí třídy javy místo funkce JavaSciptu, volá se příslušná třída, ale zápis se liší. Definice umístění třídy se zapisuje do hranatých závorek. V parametru *url* může být deklarováno několik adres, kde je třída umístěna. Pokud se nezdaří nalézt třídu na první adrese v seznamu, pokračuje na další.

```
...
url ["MyJava.class","http://myclass.cz/myjava.class"]
...
```

Významnou a velmi důležitou podmínkou použití tříd javy ve webovém prohlížeči a prohlížeči VRML, je překlad zdrojového souboru pro verzi javy 1.1. Pro vyšší verzi není možné použít třídy. Další nutnou podmínkou je použit při prohlížení Microsoft Java Virtual Machine. A poslední podmínkou je při prohlížení takového virtuálního světa použít webový prohlíže Internet Explorer s nainstalovaným VRML prohlížečem. Bez splnění všech těchto podmínek nebudou třídy fungovat. Obrázek [18] ukazuje nastavení Internet Exploreru, *Možnosti internetu* tak, aby byl schopen prohlížet světy spojené s třídami javy.



Obrázek 18: Nastavení Microsoft Java Virtual Machine

Pokud třída obsahuje ve svém kódu připojení na server specifikovaný IP adresou a portem, nedojde k jeho vykonání z bezpečnostních důvodů implementovaných v javě. Pro připojení na server je nutné, aby na stejném serveru byla umístěna i volající třída. Bližší informace o tvorbě vituálních světů pomocí VRML jsou uvedeny v [12].

3.1.2 Koncept VRML modelu

Fyzický model administrativní budovy neodpovídá přesně reálným proporcím budov. Proto byl virtuální dům postaven odšlišně s přihlédnutím na reálné proporce a běžné rozměry budov. Všechny místnosti, ovládací prvky a podstatná zařízení z reálného modelu byly vytvořeny i ve virtuálním modelu.

Obrázek 19: Celý virtuální model budovy včetně okolí

Výstavba celého domu je postavena na využívání knihoven objektů, které se následně sdružují do větších celků až po místnosti. Byl navržen model domu jak je uvedeno na obrázku [19] výše. Dům lze rozdělit do dvou části spojených chodbou. Rozložení všech místností respektuje rozložení místností v reálném modelu. V levé části je umístěna garáž. Nad garáží se nachází prezentační místnost. V pravé části je technická místnost, která ve virtuálním světě neslouží žádnému účelu a proto do ní nevedou žádné dveře. Nad technickou místností je kancelář číslo jedna a nad ní je kancelář číslo dva. Obě mají stejné technické vybavení. Jen se liší rozmístěním sortimentem a rozmístěním nábytku. Chodby jsou rozděleny do třech pater včetně přízemí. Každá chodba je spojena s výtahem. Neobsahuje žádné světlo, jako reálný model. Jednotlivé místnosti jsou vybaveny takto. Garáž je vybavena pouze vraty, dvěmi zářivkovými světly s jejich vypínači. Tyto dveře, vrata a světla nemají v realném modelu ekvivalent. V prezentační místnosti je umístěno stahovací pláno ovládané dvěma vypínači. Všechny ovládací vypínače jsou umístěny na pravé zdi vedle dveří. Po stranách místnosti jsou barevná světla, sloužící k vyvolání barevné atmosféry. Na stropě uprostřed místnosti jsou umístěna dvě zářivková světla. Na jejím kraji blízko plátna jsou umístěna halogenová světla. Oproti realitě, kde je jen jedno okno, je v prezentační místnosti šest oken. Celá místnost je vybavena pěti řadami židlí, několika obrazy na stěnácha a stupínkem. Na něm je stůl s židlí. Podél stěny je radiátor a na stěně jsou hodiny.

Obrázek 20: Prezentační místnost

V dolní kanceláři jsou umístěna dvě zářivková světla se svými vypínači. Dále dvě skříne, tři obrazy, kancelářský stůl s židlí a hodiny na stěně. Na podlaze je umístěn koberec. V místnosti je též radiátor.

Obrázek 21: Dolní kancelář

Horní kancelář obsahuje též dvě zářivková světla se svými vypínači. Čtyři skříně, dva obrazy a zbylé vybavení je jako v dolní kanceláři.

Obrázek 22: Horní kancelář

Všechny tři chodby v budově jsou vybaveny až na drobné detaily stejně. Obsahují vchodová dveře či místnoti, dveře od výtahu a okno.

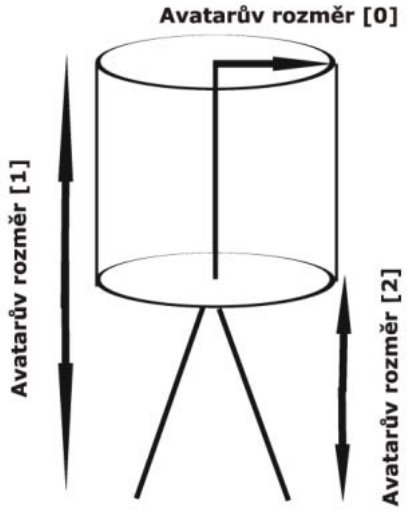
Obrázek 23: Chodba v přízemí

Dynamika virtuálního modelu je vytvořena pomocí senzorů dotyku na vypínačích, dveřích či oknech spojených s výkonnou jednotkou. Výkonná zařízení mohou být světla, objekty reprezentujícími světelné zařízení jako zářivky, halogeny nebo jiné objekty jako plátno, dveře či okna. Po stisku vypínače, se událost přes funkci *ROUTE* předá do vstupní proměnné skriptu. Ten nastaví intenzitu a barvu světla, nastaví svítící barvu a intenzitu barvy na objektu reprezentující daný svítící objekt. Dveře jsou samy čidlem dotyku, tak ovládanou jednotkou. Po dotyku dveří se událost předá skriptu a ten nastaví natočení a posun příslušných dveří. Stejný proces je aplikován na otevření garážových vrat i na otevírání oken v kancelářích. V případě výtahu je situace komplikovanější o přesun Avatara v prostou. Hlavní ovládací panel je uvnitř kabiny výtahu, kde lze přivolat výtah

do přízemí, prvního a druhého patra. Dotykové senzory v tlačítkách jsou napojeny na skripty, které přesouvají celý objekt kabiny výtahu do zvoleného patra a následně mění hodnotu pozice Avatara v ose Y. Tento proces vyvolá dojem posunu výtahu i s Avatarem do požadovaného patra.

Obrázek 24: Vnitřní prostor výtahové kabinky s ovládacími tlačítky

Pro správne pochopení jak se virtuální svět staví a jak je vnímán, je důležité pochopit i metriku, která v něm platí. Základní jednotkou je metr. Tedy pokud v uzlu *Transform* bude napsána hodnota 1 v ose Y, jedná se o posun o jeden metr v dané ose. Dalším příkladem je definice rozměru objektu *Box*. Pokud se zadají rozměry `geometry Box {size 1 1 1}`, znamená to vytvoření krychle o velikosti 1x1x1 metr. Krychle o obsahu $1m^3$. Další a velmi důležitou vlastností je, že se vše začíná počítát od středu virtuálního světa. Pokud se jedná o potomky uzlů, tak ty mají střed ve středu jejich rodičů. Není-li zadána transformace posunu, tak se objekt umíst svým středem do středu virtuálního světa či do středu svého rodičovského uzlu. Z těchto metrik se odvijí i rozměr Avatara. V tomto případě byl Avatar zvolen o velikostech uvedených v položce *NavigationInfo* resp. v parametru *avatarSize [0.25, 1.5, 0.6]*. Tedy 25cm jako poloměr kružnice opisující svislý rozměr těla. Výška Avatara byla stanovena na 1,5m. Výška, kterou bude možné překročit či vyjít je stanovena na hodnotu 60cm.



Obrázek 25: Geometrický význam parametrů rozměru Avatara

3.1.3 Stručné využití a potenciál

VRML má za účel přiblížit prezentaci prostorových dat bez použití drahých CAD programů. Veliká výhoda VRML je orientace na svět Internetu. Po instalaci zásuvných modulů, je možné prohlížet virtuální světy z jakéhokoliv místa světa. Značné uplatnění může VRML naléz v architektuře, stavebnictví či při prodeji výrobků, kdy si zákazním může prohlédnout výrobek aniž by musel navštěvovat kamenný obchod daného výrobce. Mezi další výhody je možnost propojení s aplikacemi spojených s databází či skriptovacími jazyky jako PHP, JSP či ASP. Poněvač se jedná o textové soubory, lze vytvářet světy z databáze a následně data získaná ve virtuálním světě od návštěvníka ukládat do databází. Jednu z takových aplikací prezentuje průchod virtuální Prahou [7], která je vytvořena Computer Graphics Group na katedře počítačů, ČVUT v Praze. Dalším využitím je spojení s programoavcím jazykem Java a s tím spojených všech možností, které tento programovací jazyk nabízí. V posledních několika letech byl konsorciem WEB3D vyvinut ISO standard X3D[12], který je následníkem a zastřešujícím standardem nad VRML. Tedy VRMLje podtřída X3D. Přináší strukturovanost zápisu a požnost flexibilnějšího rošíření možností. Mezi významná rozšíření je GeoVRML, které umožňuje modelovat celý svět podle měření geodetů. Dále standard X3D umožňuje po překladu do binárního zápisu import do videa ve formátu MPEG-4. Poté může být klasický video proud rozšířen o virtuální realitu. Konsorcium je seskupení významných softwarových firem, ale i menších firem včetně jednotlivců. Existuje celkem 15 skupin vývojářů pracujících na vývoji a rozvoji 3D grafiky na internetu. Bližší informace o X3D a 3D grafice na webových stránkách jsou na [6].

[12] Extensible 3D

3.2 Abstrakce modelu

Při vytváření virtuálního modelu byl kladen důraz na převod stejného množství ovládacích i ovládaných prvků z reálného modelu do virtuální reality. Jelikož nebylo možné dodržet všechny rozměry a proporce, byl vybudován model se stejným počtem prvků, stejným účelem použítí, ale s jejich rozdílným umístěním v modelu. Nemožnost převodu reálného modelu do virtuálního světa je především v disproporci prezentační místnosti v porovnání s reálnými rozměry dvěří a oken. Dalším faktem je, že reálný model postrádá přední stěnu resp. je nahrazena průhledným plexisklem. Neposledním aspektem je i estetická stránka, kdy ve virtuálním světe je možné celý dům vybavit okolím v podobě stromů, trávníku či plotu. Uvnitř modelu jsou umístěny obrazy, hodiny a kancelářský nábytek, který v reálném modelu není obsažen.

3.3 Realizace

3.3.1 Konstrukce z knihoven

Virtuální model budovy využívá 17 knihoven. Zálkadní knihovnou je *dily.wrl*, která obsahuje úplně nejzákladnější díly a tvary nutné pro další konstrukce. Obsahuje například definice plaňky k plotu, desky dveří, desky podlah, stěn, dřevěné kvádry pro okna, obrazy, atp. Nadstavbovou knihovnou této základní je knihovna *komponenty.wrl* obsahující již atomická spojení základních dílů, které vytváří vnitřní vybavení modelu. Stavební prvky a vybrané prvky vybavení jsou umístěny v knihovně *celky.wrl*. Další nadstavbou nad knihovnou *celky.wrl* je *celky stavba.wrl*, která zajišťuje všechny stavební prvky se zdmi a v nich umístěné díly jako okna či dveře. Následuje množina knihoven nazvaných stejně jako jejich obsah. Tedy knihovna s názvem *chodba.wrl* obsahuje definici celé chodby, *prezentacka.wrl* obsahuje definici celé prezentační místnosti atp. Nad všema knihovnama definující jednotlivé místnosti je zastřešující knihovna *dum.wrl*. Hlavní knihovna je *domek.wrl* a obsahuje definici celého virtálního modelu včetně okolí a všech světel, plotu a stromů.

3.3.2 Spojení jednotlivých komponent

Dynamické akce nelze spojit skrze knihovny. Proto jsou objekty, které mají být interaktivní, definovány v jednom rodičovském uzlu, aby bylo možné předávat události ze senzorů na skripty a provést příslušnou viditelnou reakci. Proto zářivková světla mají ve svém uzlu definovány i jejich ovládací vypínače. Při vytváření interaktivního světa je možné definovat objekty se kterýmy bude později manipulováno. Pomocí několika skriptů je možné řídit jeden objekt, když ukazatel na něj obsahují všechny příšlušné skripty. Při řízení mohou nastat dva případy jak využít skripty. V prvním případě skript přímo ovlivňujeme definovaný objekt.

```
DEF SWITCHHALOGENDOWN Script {
        eventIn SFBool switchlight
        field SFNode levyhalogen USE LEVYHALOGEN
        field SFNode pravyhalogen USE PRAVYHALOGEN
        ...
```

```
        url "javascript:
        function switchlight(hodnota)
        {
          if(hodnota)
          {
            levyhalogen.appearance.material.emissiveColor.r = 0;
            ...
            pravyhalogen.appearance.material.emissiveColor.r = 0;
            ...
          }
        }
   "
}
```

V druhém případě je uvnitř skiptu vytvořena požadovaná hodnota a následně se distribuuje na cílový objekt, jak ilustruje následující příklad.

```
...
DEF SWITCHGATEVRATA Script {
    field SFBool stav FALSE
    field SFNode dvere USE VRATA
    eventIn SFBool switchdoor
    eventOut SFRotation rotace
    eventOut SFVec3f posun
    url "javascript:
    function switchdoor(hodnota)
    {
        if(hodnota)
        {
            if(stav)
            {
            stav=!stav;
            rotace = new SFRotation(0,1,0,1.571);
            posun = new SFVec3f(-1.5,0,-1.5);
            }
            else
            {
            stav=!stav;
            rotace = new SFRotation(0,1,0,0);
            posun = new SFVec3f(0,0,0);
            }
        }
    }
    "
    }
    ]
ROUTE TOUCHVRATA.isActive TO SWITCHGATEVRATA.switchdoor
ROUTE SWITCHGATEVRATA.rotace TO VRATA.set_rotation
```

```
ROUTE SWITCHGATEVRATA.posun TO VRATA.set_translation
...
```

Poslední možností jak vytvořit dynamickou akci, je použití tříd javy. K vytvoření tříd je nutné připojit příslušné knihovny pro manipulaci s DOM[13] modelem virtuálního světa. Pro řízení událostí a akcí ve virtuálním světě složí vytvořené funkce v připojených knihovnách.

```
import vrml.eai.event.*;
import vrml.eai.field.*;
import vrml.eai.*;

public class myApplet extends Script
{
private SFTime startTime; // eventIn
private MFString ChangedText; // eventOut
private SFVec3f  ChangedPosition; // eventOut
...
public void processEvent (Event touch)
  {
  ...
  SFVec3f position = new SFVec3f ( 0, -1, 0 );
ChangedPosition.setValue ( position );
...
  }
}
```

3.3.3 Dynamické VRML

Pro správné pochopení práce skriptu, který udává dynamiku ve virtálním světe, je nutné pochopit jeho chování a smysl prováděných operací. Skript je uzel se speciální vlastností. Může obsahovat libovolný kód definovatelný v jazyce javascript nebo v jazyce java. Pro pozdější směrování události do toho skriptu je nutné použít direktivu *DEF*, která jej definuje jako jedinečný objekt, na který se lze odkazovat jeho jménem. Následující příklad ilustruje, jak událost dotyku senzoru na vypínači halogenových světel je předána do vstupní proměnné *switchlight* v tělě uzlu Script.

```
DEF SWITCHHALOGENDOWN Script {
eventIn SFBool switchlight
...
}
...
ROUTE HALOGENTOUCHUP.isActive TO SWITCHHALOGENUP.switchlight
```

Pro spuštění potřebné funkce je nutné, aby se funkce jmenovala stejně jako vstupní proměnná. Poté co se zapíše hodnota do vstupní proměnné, zavolá se funkce stejného jména.

[13] Document Object Model

```
function switchlight(hodnota)
{
if(hodnota)
  {
  ...
  }
}
```

Vstupní proměnná se jmenuje *switchlight* a má hodnotu uloženou v proměnné *hodnota*. V tomto případě se očekává binární hodnota z definice vstupní proměnné *switchlight*. Následně je možné podle hodnoty vstupní proměnné provést různé příkazy. V těle skriptu je možné nadefinovat výstupní hodnoty proměnných pro další distribuci do jiných skriptů či pro řízení objektů.

```
...
#napojeni na drive definovany uzel LEVYHALOGEN
field SFNode levyhalogen USE LEVYHALOGEN

#vystupni promenna pro rotaci objektu
eventOut SFRotation rotace

#vystupni promenna pro posun objektu
eventOut SFVec3f posun
...
#distribuce vypoctene rotace na objekt
ROUTE SWITCHGATE.rotace TO GATEKLIKA.set_rotation

#distribuce vypocteneho posunu na objekt
ROUTE SWITCHGATE.posun TO GATEKLIKA.set_translation
```

Je možné vytvářet lokální proměnné sloužící jen pro potřeby skriptu.

```
...
field SFBool stav FALSE #definice stavu objektu
...
if(stav)
  {
  stav=!stav;
  ...
  }
...
```

Pokud jsou využity třídy javy, je nutné, aby byl název volané třídy zapsán v hranatých závorkách.

```
DEF SWITCHLEVAZARIVKAUP Script {
  eventIn SFBool switchlight
```

```
  field SFNode levazarivka USE LEVAZARIVKA
  field SFNode levesvetlo USE LEVAZARIVKAPOINTLIGHT
  url ["PrezentLeftLightUP.class"]
}
```

Po připojení příslušných knihoven pro práci s VRML do třídy, je možné jak přijímat události z virtálního světa, tak i na ně reagovat. Jsou dvě možnosti jak lze propojit virtuální svět s jazykem java. První je vytvořit java applet a následně pomocí ukazatele na prohlížeč pracovat s virtuálním světem. Podmínkou je zaregistrování třídy jako posluchače událostí ve virtuální světě. Bližší informace včetně mnoha příkladů je na [9] a [8]. Druhou možností je vytvořit jednotlivé třídy, které se následně volají v těle uzlu. V diplomové práci byl zvolen druhý přístup.

Při definici třídy má hlavička přesně stanovené rozšíření typu Script.

```
public class MyScript extends Script
{
...
}
```

Deklarují se VRML proměnné, které bude nutné řídit či od nich přijímat události.

```
...
// deklarace vstupnich a~vystupnich promennych
private SFTime startTime; // eventIn
private MFString ChangedText; // eventOut
private SFVec3f  ChangedPosition; // eventOut
...
```

Následuje funkce, která se volá automaticky při spuštění vituálního světa. Proto je vhodné do jejího těla umístit příkazy k nastavení skriptu na počáteční hodnotu.

```
public void initialize ()
{
...
#inicializace tridy
...
}
```

Důležitou funkcí nutnou pro reakce na události z VRML je *processEvent*. Její vstupní hodnota je událost, která je pak zpracována v těle funkce resp. dalšími funkcemi třídy. Dále v třídě mohou být deklarovány další funkce podle potřeby.

```
public void processEvent (Event touch)
{
...
#reakce na udalost v~promenne touch
...
}
```

Následně stačí do každého uzlu, který má vykonat nějakou dynamickou akci, vytvořit a připojit příslušnou třídu. Následující příklad ukazuje souvislost mezi řízením proměnné v třídě a ve virtuálním světě.

```
==== trida v~jave ====
public void initialize ()
{
...
//nastaveni na eventOut
ChangedPosition = (SFVec3f) getEventOut ("ChangedPosition");
...
}

public void processEvent (Event touch)
{
...
private SFTime startTime; // eventIn
private SFVec3f  ChangedPosition; // eventOut
...
SFVec3f position = new SFVec3f ( 0, -1, 0 );
ChangedPosition.setValue ( position );
...
}
==== rizeny uzel ve VRML svete ====
DEF ClickTextToTest TouchSensor { }

DEF MyScript Script {
...
eventIn SFTime startTime
eventOut SFVec3f  ChangedPosition
url [ "MyScript.class"]
...
DEF TextPosition Transform {
   translation 0 0 0
   ...
}
ROUTE ClickTextToTest.touchTime TO MyScript.startTime
ROUTE MyScript.ChangedPosition TO TextPosition.set_translation
```

3.4 Java ve VRML

3.4.1 Zakomponování Javy do VRML

Pro ovládání reálného světa resp. prezentační místnosti byly vytvořeny třídy, které se pomocí dvou komunikačních tříd připojují na LNS server a řídí reálný model. Použíté komunikační třídy jsou popsány v kapitole 3.4.3. Všechny řídící třídy jsou stejné, pouze se liší v cíli jejich řízení a proto je popsána pouze jedna vzorová třída.

Pro vytvoření třídy se připojí příslušné knihovny pomocí direktiv *import*.

```
import vrml.*;
import vrml.field.*;
import vrml.node.*;
```

Následuje samotný název třídy, který musí souhlasit se jménem volaného skriptu ve virtuálním světě. Následuje direktiva *extends* pro určení, že se jedná o skript pro virtální svět.

```
public class PrezentLeftLightUP extends Script
{
...
}
```

Poté se deklarují potřebné proměnné, které jsou ve virtuálním světě a též se deklarují třídy pro spojení s LNS serverem.

```
public class PrezentLeftLightUP extends Script
{
private SFBool      switchlight;  // eventIn
private SFVec3f     svitivabarva  // eventOut
private SFBool      spustenisvetla  // eventOut
private LNSconnection lnsspojeni; //komunikacni trida
private LNScommander lnscommands; //ridici trida
...
}
```

Po definici proměnných, je nutné přihlásit výstupní proměnné jako zdroje událostí, aby bylo možné je dále ve virtuálním světě distribuovat. Na to je vhodná funkce *initialize*, která se volá po načtení virtuálního světa.

```
public void initialize ()
{
//zavedeni vystupni udalosti
  svitivabarva = (SFVec3f) getEventOut ("svitivabarva");
  //zavedeni vystupni udalosti
  spustenisvetla = (SFBool) getEventOut ("spustenisvetla");
  return;
}
```

Zbývá definovat reakce na požadovanou událost. Na vyvolané události se reaguje v těle funkce *processEvent*. V této funkci se též definuje spojení na LNS server a vytvoření třídy pro řízení. Následně se vykoná příkaz v reálném modelu a ve virtálním modelu. Navázané spojení zůstane aktivní a bude vázat otevřený port. Pokud bude zavřeno okno s VRML, uzavřou se i všechny takto otevřené porty.

```
public void processEvent (Event touch)
{
  lnsspojeni = new LNSconnection ("147.32.87.242", 2540, "domecek_01");
  lnscommands = new LNScommander(lnsspojeni);
  lnscommands.LeftLight("ON");
  SFVec3f barva = new SFVec3f ( 0.7, 0.7, 0.7);
svitivabarva.setValue ( barva );
SFBool stav = new SFBool (TRUE);
spustenisvetla.setValue ( stav );
return;
}
```

Nyní je připravena celá třída a je ji nutné přeložit do příslušného tvaru. Překladu a všemu s ním spojeným se věnuje kapitola 3.4.2.

3.4.2 Konstrukce java tříd pro virtální svět

Pro správnou funkci java tříd je nutné splnit podmínky a nastavit prohlížeč jak je uvedeno v kapitole 3.1.1. Dále je nutné mít na počítači nainstalovanou javu od firmy Sun resp. její *Java Developement Kit*, který nainstaluje i příslušný *Java Runtime Environment*. Jak je zmíněno v kapitole 3.1.1, je nutné tento *Java Runtime Environment* vypnout v nastavení *Možnosti internetu* a zapnout používání *Microsoft Java Virtual Machine*. Pro úspěšné použití vytvořených tříd, je nutné v překladači javy *javac*, nastavit překlad na verzi javy 1.1 a připojit příslušné knihovny. Následující příklad ukazuje překlad skriptu vytvořeného v kapitole 3.4.1.

```
#prekladac vytvori tridu pro javu 1.1
#pouzije knihovnu libclasses.zip
#prelozi script PrezentLeftLightUP

javac -O -target 1.1 -source 1.2
         -classpath libclasses.zip; PrezentLeftLightUP.java
```

3.4.3 Ovládání modelu pomocí Javy

Spojení s LNS serverem a vykonávání příkazů je vytvořeno pomocí dvou tříd. První třída zprostředkovává spojení se samotným LNS serverem, vybírá podsystémy a příslušná zařízení. Druhá třída vykonává sled jednoduchých příkazů, které ve svém důsledků vedou k požadovanému příkazu.

První třída zajišťující spojení s LNS serverem se nazývá LNSconnection. Obsahuje připojené knihovny od firmy Echelon pro práci se serverem LNS. Definuje proměnné nutné pro obsluhu celého systému výběru podsystémů a zařízení. V parametru konstruktoru této třídy se udává IP adresa serveru, na který se má připojit. Port, který je určen ke komunikaci s LNS serverem. Standardně se jedná o port 2540. Poslední parametr je LonWorks síť do jaké se má připojit. Následuje vytvoření kontruktoru a připojení do nastavené sítě, výběr prvního podsystému a prvního zařízení v tomto podsystému.

```
LNSconnection (String IPaddress, int Port, String Network)
  {
     ...
     objectServer = new LNSObjectServer ();
     networks = objectServer.getNetworks();
     network = networks.getItem(ServerIP, ServerPort, SelectNetwork);
     network.open();
     systems = network.getSystems();
     system = systems.getItem(1);
     system.open();
     mainsubsystems = system.getSubsystems();
     // vyber rozhrani do LonWorks
     mainsubsystem = mainsubsystems.getItem (2);
     //====== INICIALIZACNI VYBER SUBSYSTEMU =======
     subsystems = mainsubsystem.getSubsystems();
     subsystem = subsystems.getItem (0);// vyber mistnosti
     //====== INICIALIZACNI VYBER ZARIZENI =======
     appdevices = subsystem.getAppDevices();
     appdevice = appdevices.getItem(0);
  }
```

Dále následují deklarace atomických funkcí, které slouží k jednoduchým operacím s LNS serverem resp. s LNS databází vybrané sítě.

- **ChoseSubsystem(int Subsystem)** - výběr podsystému podle zadaného čísla
- **ChoseDevice(int Device)** - výběr zařízení podle zadaného čísla v aktuálně zvoleném podsystému
- **getValue(String NameOfVariable)** - vrácí hodnotu proměnné zadanou jejím jménem jako vstupní hodnotu
- **setValue(String NameOfVariable,String Value)** - zapíše zadanou hodnotu do proměnné vložené jejím jménem jako vstupní hodnotu
- **getNameOfChosenSubsystem()** - vrácí název aktuálně zvoleného podsystému
- **getNameOfChosenDevice()** - vrácí název aktuálně zvoleného zařízení v aktuálně zvoleném podsystému
- **printAllNamesOfSubsystems()** - vypíše na obrazovku statistiku o všech podsystémech. Jména, pořadí v jakém jsou v LNS databázi, počet zařízení v jednotlivých podsystémech.
- **printAllNamesOfDevicesOfChosenSubsystem()** - vypíše na obrazovku statistiku o všech zařízeních v aktuálním podsystému. Jméno, pořadí v jakém jsou podsystému, počet proměnných v zařízení a neuron ID daného zařízení.
- **PushUp()** - vyvolá stejnou událost jako rychlé stisknutí a uvolnění horního tlačítka na bezdrátovém ovladači

- **HoldUp()** - vyvolá stejnou událost jako trvalé stisknutí horního tlačítka na bezdrátovém ovladači
- **ReleaseUp()** - vyvolá stejnou událost jako trvalé uvolnění horního tlačítka na bezdrátovém ovladači
- **PushDown()** - vyvolá stejnou událost jako rychlé stisknutí a uvolnění dolního tlačítka na bezdrátovém ovladači
- **HoldDown()** - vyvolá stejnou událost jako trvalé stisknutí dolního tlačítka na bezdrátovém ovladači
- **ReleaseDown()** - vyvolá stejnou událost jako trvalé uvolnění dolního tlačítka na bezdrátovém ovladači
- **LogOut()** - odpojí se od databáze a LNS serveru
- **LogIn()** - připojí se k dříve zvolené databázi a LNS serveru

Nadstavbovou třídou nad LNSconnection je LNScommander. Jedinou vstupní hodnotu konstruktoru třídy je LNSconnection se spojením na LNS server a příslušnou LNS databázi. Následují funkce, které obsahují sledy jednoduchých příkazů z funkcí třídy LNSconnection, které ve svém důsledky vykonají příslušný příkaz. Každá funkce má jako vstupní parametr požadovaný příkaz. Tento popis ilustruje příklad funkce k ovládání plátna v prezentační místnosti.

```
public String Screen(String Command)
{
String Direction="";
String State="";
if(Command.equals("UP")){ //reakce na prikaz UP
  //vyber vhodneho zarizeni
listDevices.setValue("nviActiveDevice","3,0 0");
//zadani prislusne hodnoty
listDevices.setValue("nviScreenSwitch","0,0 0");
//vyvolani sepnuti tlacitka ovladani
listDevices.PushUp();
}
if(Command.equals("DOWN")){//reakce na prikaz DOWN
  //vyber vhodneho zarizeni
listDevices.setValue("nviActiveDevice","3,0 0");
//zadani prislusne hodnoty
listDevices.setValue("nviScreenSwitch","100,0 0");
//vyvolani sepnuti tlacitka ovladani
listDevices.PushDown();
}
if(Command.equals("STATE")){//reakce na prikaz STATE
  //vyber vhodneho zarizeni
listDevices.setValue("nviActiveDevice","3,0 0");
//prevede vnitrni reprezentaci stavu
```

```
//na jeji slovni reprerentaci
int Vyskyt = listDevices.getValue("nvoScreen").indexOf(",");
State = listDevices.getValue("nvoScreen").substring(0,Vyskyt);
if(State.equals("100"))
{
Direction = "UP";
}
if(State.equals("0"))
{
Direction = "DOWN";
}
}
return Direction;
}
```

Následuje jednoduchý popis vytvořených funkcí třídy LNScommander,které ovládají zařízení umístěné v prezentační místnosti.

- **Screen(String Command)** - povolené vstupní hodnoty příkazů: UP, DOWN, STATE. Vytáhne, spustí plátno a vypíše slovní reprezentaci stavu plátna.
- **Halogens(String Command)** - povolené vstupní hodnoty příkazů: ON, OFF, STATE. Vypne, zapne halogenová světla a vypíše slovní reprezentaci stavu.
- **LeftLight(String Command)** - povolené vstupní hodnoty příkazů: ON, OFF, HOLDUP, HOLDDOWN,RELEASEUP, RELEASEDOWN, STATE. Vypne, zapne levé zářivkové světlo. Postupně vypíná či rozsvěcí světlo a vypíše slovní reprezentaci stavu.
- **RightLight(String Command)** - povolené vstupní hodnoty příkazů: ON, OFF, HOLDUP, HOLDDOWN,RELEASEUP, RELEASEDOWN, STATE. Vypne, zapne pravé zářivkové světlo. Postupně vypíná či rozsvěcí světlo a vypíše slovní reprezentaci stavu.
- **RedLedLight(String Command)** - povolené vstupní hodnoty příkazů: ON, OFF, HOLDUP, HOLDDOWN,RELEASEUP, RELEASEDOWN, STATE. Vypne, zapne červené podkresové světlo. Postupně vypíná či rozsvěcí světlo a vypíše slovní reprezentaci stavu.
- **GreenLedLight(String Command)** - povolené vstupní hodnoty příkazů: ON, OFF, HOLDUP, HOLDDOWN,RELEASEUP, RELEASEDOWN, STATE. Vypne, zapne zelené podkresové světlo. Postupně vypíná či rozsvěcí světlo a vypíše slovní reprezentaci stavu.
- **BlueLedLight(String Command)** - povolené vstupní hodnoty příkazů: ON, OFF, HOLDUP, HOLDDOWN,RELEASEUP, RELEASEDOWN, STATE. Vypne, zapne modré podkresové světlo. Postupně vypíná či rozsvěcí světlo a vypíše slovní reprezentaci stavu.
- **Roller(String Command)** - povolené vstupní hodnoty příkazů: UP, DOWN, STATE. Vytáhne, spustí roletu a vypíše slovní reprezentaci stavu rolety.

Výše popsané třídy dokáží plně simulovat akce vyvolané uživatelem a ovládat všechna zařízení v prezentační místnosti. V závěru diplomové práce je představen možný postup rozšíření třídy LNScommander o další místnosti a zařízení.

3.5 VRML a realný model administrativní budovy

3.5.1 Java komponenty firmy Echelon

Pro ovládání reálného modelu, bylo vytvořeno 16 tříd, které byly postaveny tak, jak bylo popsáno v kapitole 3.4.1. Všechny popsané a vytvořené třídy ve svém základu využívají knihovny firmy Echelon. Pokud se připojuje třída na LNS server, využívá objektu LNSObjectServer, který následně obsahuje spojení s LNS serverem a vybranou sítí. Po vytvoření tohoto objektu, je nutné volat funkci *getNetworks()*, která jako návratovou hodnotu vrací objekt *LNSNetworks*, který již bude poskytovat funkci pro spojení s příslušným LNS serverem. Příklad ukazuje navázání spojení s LNS serverem.

```
LNSNetworks networks;
...
objectServer = new LNSObjectServer ();
networks = objectServer.getNetworks();
network = networks.getItem(ServerIP, ServerPort, SelectNetwork);
...
```

Poté je nutné samotnou síť otevřít, vybrat rozhraní, které je spojením do LonWorks. Po provedení těchto operací je třída LNSconnetion plně připojena do sítě LonWorks skrze příslušný LNS server.

```
network.open();
systems = network.getSystems();
system = systems.getItem(1);
system.open();

mainsubsystems = system.getSubsystems();
// vyber rozhrani do LonWorks umístěné
// umistene v~realnem modelu budovy
mainsubsystem = mainsubsystems.getItem (2);
```

3.5.2 Proces komunikace mezi virtuálním a reálným světem

Celý proces od vzniku události ve virtálním světe až po vykonání příkazu v reálném světe je řetězcem mnoha kroků. Ve virtuálním světě je prvním krokem vyvolání události v dotykovovém senzoru. Událost dotyku se přenese přes direktivu *ROUTE* na příslušný uzel *Script*. Ten zavolá zadanou třídu. Tato třída vytvoří spojení s definovaným LNS serverem, vykoná příkazy uvedené v těle třídy. Tím se zapíší hodnoty do proměnných v LNS databázi, které ve svém důsledku znamenají příkaz pro fyziká zařízení k nastavení svým výstupů na zadanou hodnotu. Výsledkem je provedení požadované akce. Tento řetězec je velice krátký a rychlý. Proto reakce zařízení od vyvolání události dotyku je

kratší než při použití vizualizace, která využívá služeb OPC serveru. OPC server je prostředník a dále zapisuje nebo získává data z LNS serveru. V závěru diplomové práce je uveden možný rozvoj aplikace s ohledem na možnou vizualizaci procesů v reálném modelu budovy.

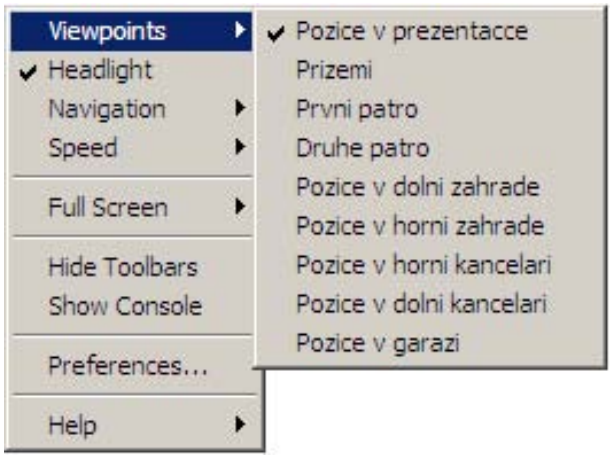
3.6 Cortona VRML prohlížeč

3.6.1 Popis prohlížeče

K prohlížení virtuálního světa VRML byl vybrán prohlížeč firmy Parallelgraphics s názvem Cortona. Na webových stránkách Parallelgraphics, lze vybrat z instalace prohlížeče VRML přímo do webového prohlížeče nebo je nabízena možnost stažení celého prohlížeče a následné instalace. Vhodnější je stážení kompletního balíčku prohlížeče a následné instalace do webového prohlížeče. Samotný prohlížeč Cortona se nainstaluje jako zásüvný modul do všech prohlížečů, které jsou podporovány. Jedná se o prohlížeč Internet Explorer, Mozilla Firefox a Opera. Prohlížeč Cortona byl vybrán pro svou jednoduchost a přehlednost svých ovládacích prvků. Na obrázku 14 je prohlížeč Cortona vyobrazen včetně všech svých ovládacích prvků.

3.6.2 Ovládání prohlížeče

Prohlížeč Cortona umožňuje vedle ovládání pomocí myši, ovládání pomocí kláves. Pokud je stisknuta klávesa *ALT*, bude způsob pohybu přepnut z pohybu standardního, přirozeného lidské chůzi, na pohyb pouze v kolmých směrech. Tedy vlevo, vpravo, vzad a vpřed. Nelze provádět otáčení. Při stisku klávesy *MEZERNÍK* je Avatar ukotven v aktuálním místě a klávesami šipek se lze otáčet kolek středu Avatara. V levé části prohlížeče jsou tři ovládací prvky k řízení způsobu pohybu ve virtuálním světě. *WALK* je určen k pohybu klasickou chůzí včetně všech fyzikálních důsledků. Pokud tedy Avatar půjde po plošině, ve které bude otvor, propadne se pokud nad něj vstoupí. *FLY* umožňuje pohybovat se zcela volně v prostoru, jako by mohl Avatar létal. V tomto případě by avatar do otvoru při průchodu nad ním nespadl. Posledním způsobem je *STUDY*. Jedná se o pohyb celého virtuálního světa kolem jeho středu. Další možnosti ovládání nabízí lokální nabídka.



Obrázek 26: Lokální nabídka prohlížeče Cortona

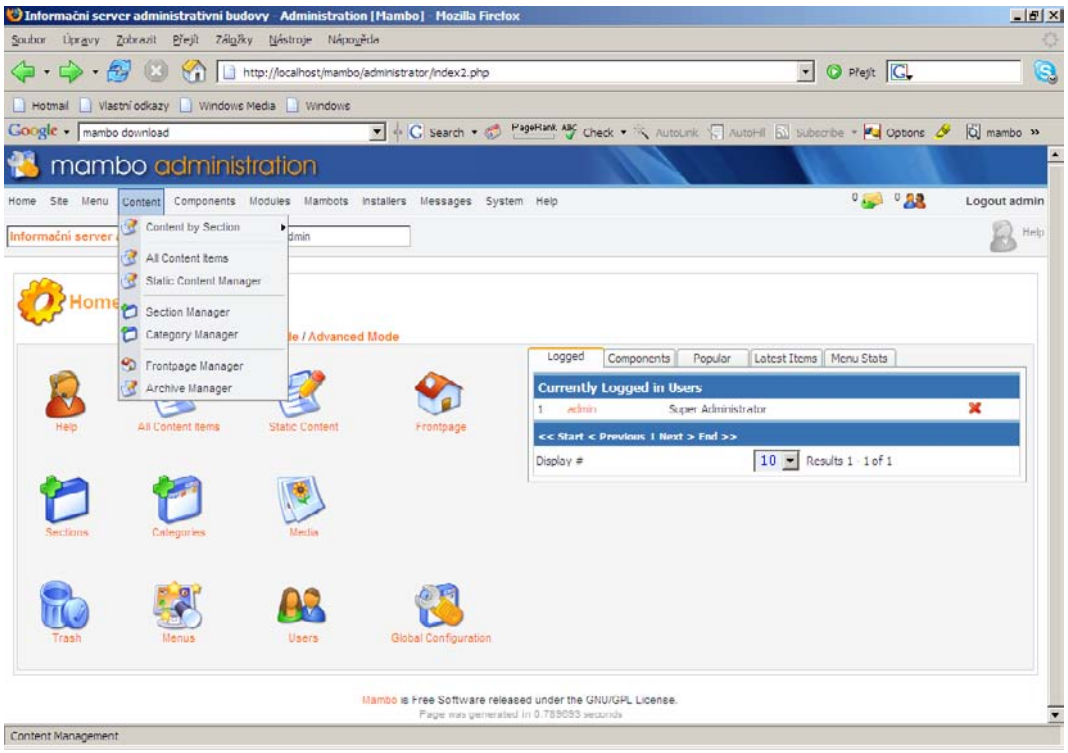
3.6.3 Další alternativní prohlížeče VRML světů

Vedle prohlížeče Cortona, existují i další VRML prohlížeče. Jedná se o Cosmo Player od firmy Cosmo Software, BS Contact od konzorcia Web3D a mnoho dalších.

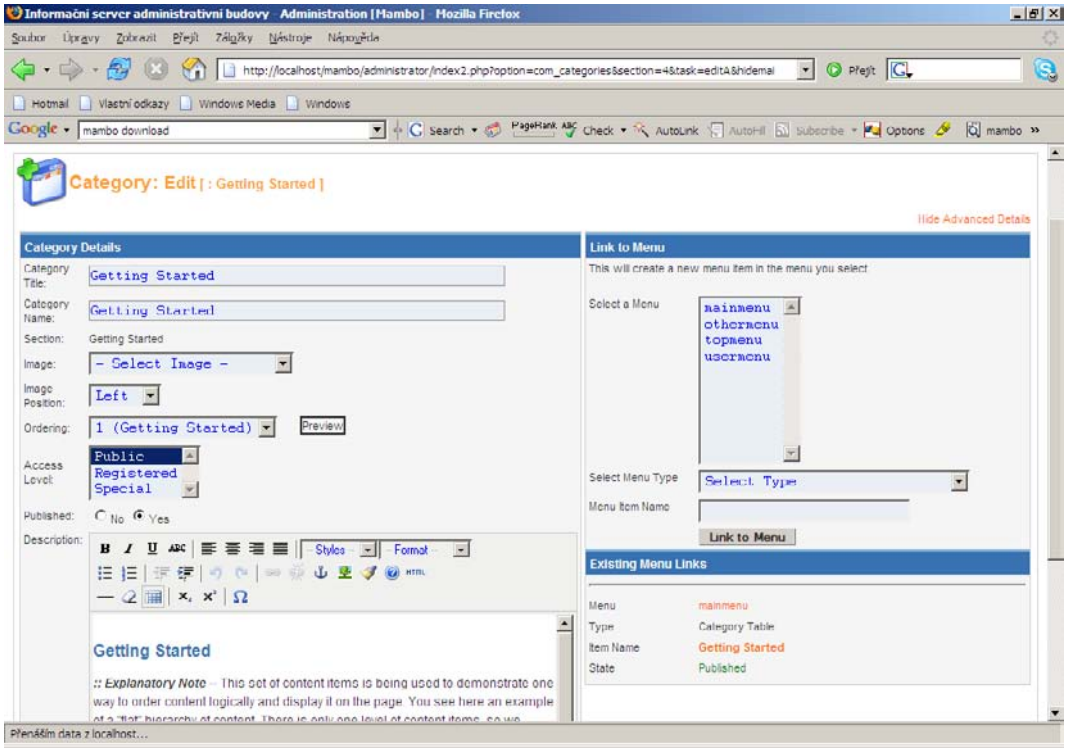
3.7 Webový portál

3.7.1 Redakční systém

Pro prezentaci a zdroj informací slouží redakční systém umístěný v serveru v modelu budovy. Pro tento účel byl vybrán redakční systém Mambo. Jedná se o *Open source* projekt, který je volně stažitelný na stránkách www.mamboserver.com. Celý projekt je podporován širokou komunitou vývojářů. Mambo poskytuje administrační prostředí umožňující veškeré akce, které jsou nezbytné ke kvalitnímu vedení informačního serveru. Nabízí sadu přístupových práv, instalaci vzhledu, jazykové mutace, vkládání článků, registraci uživatelů, vytváření menu a též obsahuje grafický nástroj pro psaní příspěvků.



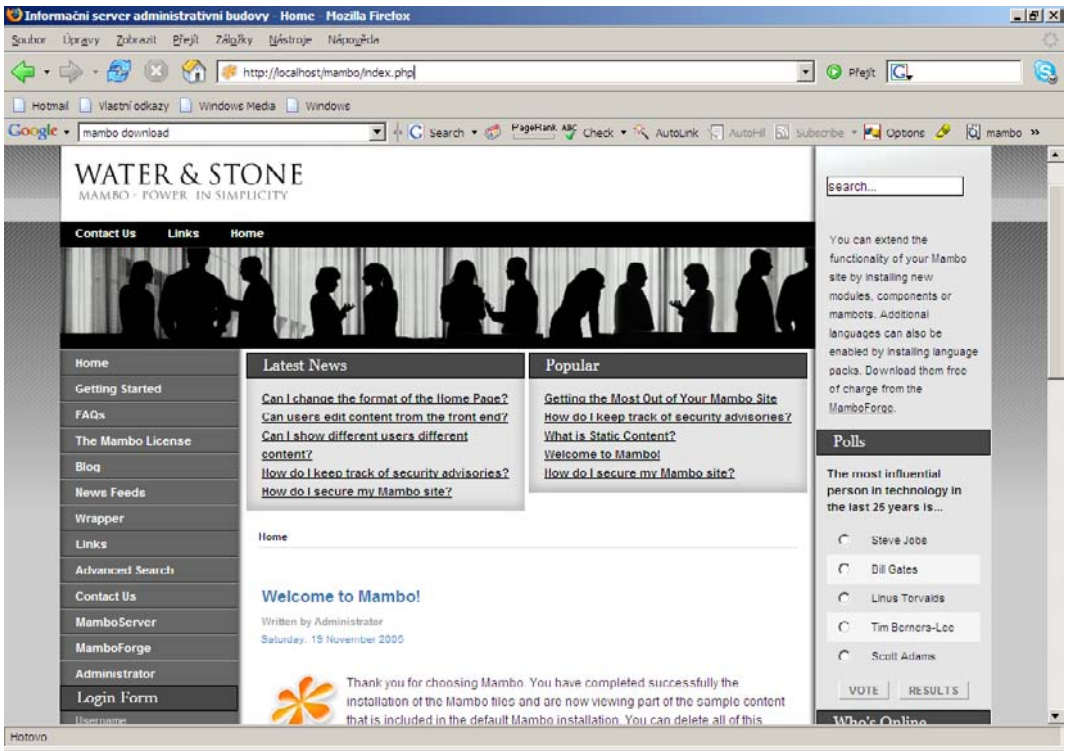
Obrázek 27: Administrační prostředí redakčního systému Mambo



Obrázek 28: Editor příspěvků

3.7.2 Rozvoj webového portálu

Hlavním účelem celého portálu je informovat návštěvníky o projektu řízení inteligentní administrativní budovy. Měl by obsahovat základní informace o projektu, požadované popisy jednotlivých částí modelu a stručná schémata elektroinstalace. Pro registrované uživatele by se měl otevřít širší repertoár informací. Od přesných popisů, přes elektrická schémata zařízení po návody a manuály k zařízením umístěných v modelu. V konečné fázi by měl mít informační server dvě úlohy. První je prezentační a druhá je informační. Prezenční část komplexně představuje projekt administrativní budovy. Druhá část je zdrojem všech potřebných informací pro studenty pracující s modelem a urychluje jim seznámení se s funkcemi celé budovy a všech zařízení.



Obrázek 29: Standardní vzhled úvodní stránky redakčního systému

4 Otopný systém

4.1 Úvod do problematiky vytápění objektů

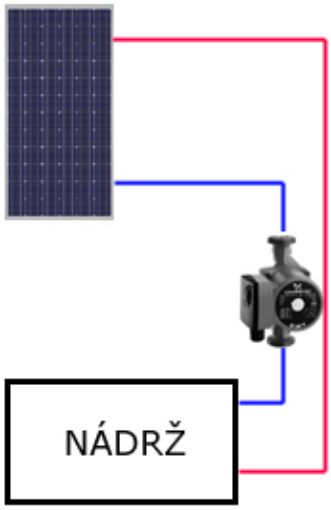
V dnešní době, kdy stoupá cena energií, je potřeba s energií spotřebovanou na vytápění uvážlivě šetřit. Z tohoto důvodu se vedle klasických zdrojů energie pro vytápení jako je uhlí, plyn a elektřina zkouší i alternativní zdroje energie a tepla jako tepelná čerpadla, větrné elektrárny, spalování biomasy, solární kolektory a fotovoltalické články. K dalším úsporám energie dojde při důkladné izolaci vytápěných objektů. Proto se starší domy při renovaci zateplují. Nové domy se již bez důkladné izolace ani nestaví. Starší panelové domy čerpají dotace či z vlastních prostředků pořizují izolaci a maximálně eleminují únik tepla. Též domky v soukromém vlastnictví investují do izolace. Nejedná se o lacinou investici, ovšem v dlouhodobém horizontu se taková investice vyplatí. Jako ukázka efektivity vytápění při použití izolace byly v administrativní budově vybudovány dvě rozdílné kanceláře, kde první kancelář nebyla vybavena dostatečnou izolací, ale obsahuje výkonné otopné těleso. Druhá místnost má menší a tím i na provoz levnější otopné zařízení a je lépe tepelně izolována pro menší odvod tepla do okolí.

4.1.1 Vytápění budov

Problematika vytápění budov či obecně staveb je rozsáhlá oblast. Zaměříme-li se pouze na vytápění budov, lze identifikovat kroky vedoucí k efektivnímu vytápění. První je výběr vhodného zdroje energie pro vytápění. Dále je nutné se soustředit na maximální úsporu energie. Vytápění budov je procesem od vhodného výběru energie pro vytápění (uhlí, plyn, biomasa aj.), přes výběr tepelné izolace, kalkulace tepelných ztrát objektu a na to navazující výpočet výkonu topných zařízení (radiátory, přímotopy aj.). Volba jejich vhodného umístění a nosného média pro teplo (voda, solární tekutina aj.). Následuje návrh vedení pro topné médium a konstrukce celého otopného systému.

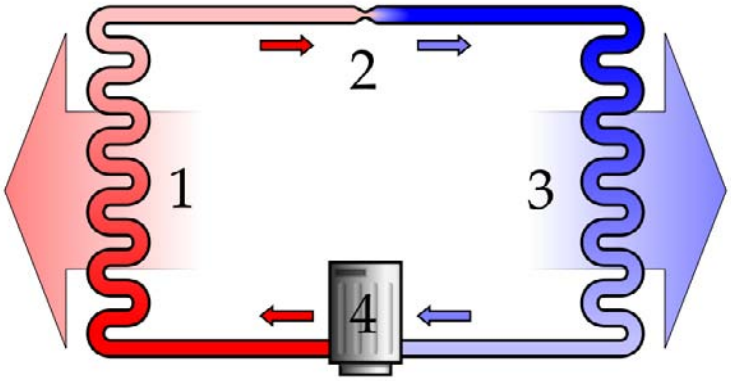
4.1.2 Tepelné zdroje

Zdrojem tepla může být klasické spalování uhlí, plynu či přeměna elektrické energie na teplo. Dalším zdrojem tepelné energie jsou alternativní druhy jako jsou solární kolektory, které přijímají sluneční záření a přenáší jej na ohřev kapaliny a ten následně na ohřev otopné vody pro vytápění či výrobu elektrické energie. Zde je podmínkou efektivity výroby elektrické energie velký počet dní slunečnečního svitu. Proto se staví výkonné elektrárny na této bázi především v pouštích. Nevýhoda tohoto zdroje je v jeho závislosti na silném slunčním svitu. Nejvíce je energie potřeba v zimním období, kdy je právě silného sluněčního svitu velmi málo.



Obrázek 30: Tepelný solární okruh

Proto jsou ještě další možnosti získávální alternativních zdrojů tepla. V posledních letech se začíná využívat jako zdroj tepelná čerpadla. Pomocí stejného principu jako v chladicích zařízení je odebíráno teplo z okolí (vzduch, voda, hlubinné vrty, ..) a získané teplo je předáváno přes kompresor do vytápěného objektu. Bližší informace na [5].



Obrázek 31: Diagram tepelného čerpadla: 1) kondenzátor, 2) expanzní ventil, 3) výparník, 4) kompresor

4.1.3 Tepelné ztráty

Možností jak ušetřit na vytápění, je zaměřit se na omezení úniku tepla z vytápěné budovy. Existuje mnoho způsobů jak zabránit úniku energie. Nejvíce se rozšířilo umístění tepelné izolace na obvodové stěny budovy ve spojení s použitím tepelně výhodnějších stavebních

materiálů jako je porotherm[14]. Obvodové zdi se izolují použitím polystyrénových desek, které slouží k odstranění tepelných mostů[15]. Používají se desky od 5cm tloušťky od 15 cm. Dále je použito několik dalších vrtev tloušťky 1mm pro zlepšení tepelné izolace a montáž na obvodové zdivo. Významná část tepla uniká střechou, proto se izolují i podkroví a ve vzniklém prostoru se budují obytné místnosti. Pomocí termokamery můžeme získat termogram viz obrázek [32]. Zdroj dat [3].

Obrázek 32: Prostup tepla z budovy rodinného domu a demonstrace tepelných mostů

4.2 Vybrané otopné systémy

Mnoho zdrojů energie nelze v našem případě pro realizaci v modelu administrativní budovy použít. Proto byly vybrány druhy energie, které byly dostupné a realizovatelné. Pro vytápění je použit jako zdroj energie elektrický proud z veřejné elektrické sítě a solární kolektor.

4.2.1 Elektrokotel

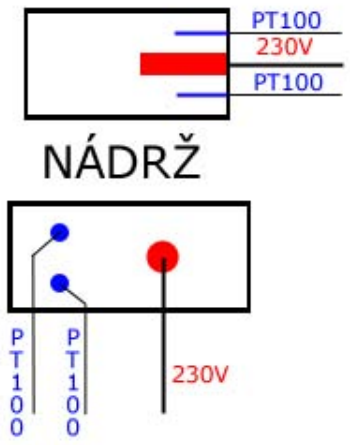
Elektrický kotel ohřívá pomocí topného tělesa o výkonu 1000W otopnou vodu, která dále dodává teplo do radiátorů. Je napájen napětím 230V/50Hz. Celý kotel má objem 40l (bez objemu potrubí a radiátorů). V kotli jsou umístěna dvě čidla teploty[16]. Jsou zasunuta do jímek, které zasahují 10cm do kotle, což zajištuje dostatečně přesné měření teploty v kotli. Měří se teploty odchozí a příchozí vody. Vzhledem k tomu, že nedochází k extrémním výkyvům teplot v okolí modelu, přistoupilo se k následujícímu kompromisu: solární kolektor není oddělen od vody pro otopná tělesa. Pokud by byl solární kolektor vystaven velkým výkyvům teplot, byl by v kotli umístěn výměník, který by předával získanou energii otopné vodě. Jako kapalina se v tomto případě používá např. Kolekton[17]

[14] Bližší informace na www.porotherm.cz

[15] Tepelný most je místo, kde uniká více tepelné energie a má v interiéru studenější povrch a naopak v exteriéru teplejší povrch než okolní konstrukce

[16] Čidla jsou realizována pomocí odporových čidel Pt100

[17] Teplonosná kapalina do solárního kolektoru. Provozní teplota: -30°C až +140°C



Obrázek 33: Schéma rozmístnění čidel teploty a topného tělesa

4.2.2 Solární kolektor

První způsob vytápění modelu budovy je pomocí solárního kolektoru. Ten je umístěn na střeše modelu. Pro simulaci sluneční energie byl vybrán infrazářič o výkonu 2000W. Jedná se o klasický infrazářič s rubínovou trubicí vyzařující červené světlo ve viditelném spektru vyzařovaného vlnění. Je umístěn přibližně 80 cm od plochy kolektoru. Menší vzdálenost není možná z důvodu vysokých teplot vytvářených infrazářičem na povrchu solárního kolektoru. Větší vzdálenost je omezena prostorem v kterém je model umístěn. Tento infrazářič a solární kolektor je ovládán programovatelným automatem Wago[18].

4.2.3 Vytápění radiátory

Druhým způsobem vytápění modelu budovy je vytápění pomocí elektrokotle. Ohřátá voda je distribuována čerpadly do radiátorového okruhu. Menší radiátor o rozměrech 50x30x2 centimetrů byl umístěn do izolované místnosti a větší výkonější radiátor o rozměrech 80x50x2 centimetrů byl umístěn do méně izolované místnosti. Tyto radiátory jsou napojeny na systém trubek určených k rozvodu otopné vody po budově. Radiátory jsou opatřeny ventily pro regulaci přívodu otopné vody.

4.3 Tepelné ztráty modelu

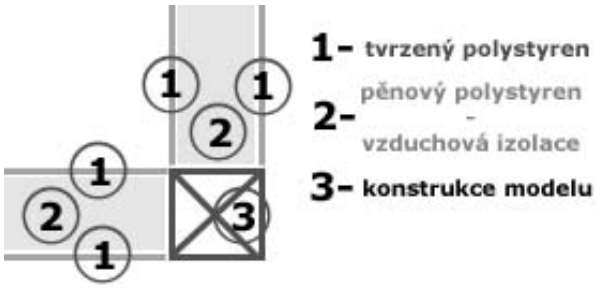
Výpočet tepelných ztrát je budovy je velmi náročná úloha pokud chceme dosáhnu přesného výsledku. K jejímu výpočtu je potřeba vědět jednotlivé rozměry budovy, strukturu zdiva, rozložení prostoru a vypočtený objem prostor. Dále je potřeba vědět maximální a minimální teploty požadované ve vytápěném prostoru, maximální a minimální teploty okolí vytápěného prostoru. Vzhledem k této těžké a komplikované úloze bylo rozhodnuto o kooperaci s faktutou stavební. Pomocí speciálního programu vypočetli ztráty

[18] Wago typ 750-819 s rozhraním pro připojení k LonWorks

pro jednotlivé místnosti. Každý materiál ve struktuře zdi má jinou tepelnou vodivost. Proto byl dodán řez stěnou s popisem jednotlivých vrstev.

4.3.1 Materiály pro výpočet tepelných ztrát modelu

Pro výpočet ztrát bylo potřeba změřit rozměry kanceláří. Dále byl nakreslen profil zdiva resp. pláště modelu budovy.



Obrázek 34: Průřez strukturou pláště stěny modelu

Byly popsány jednotlivé části pláště a z jakého materiálu jsou jednotilvé vrstvy. Vypracovaný podklad z těchto měření pro výpočet byl ještě doplněn o teploty okolí a požadované teploty v místnostech.

4.3.2 Výpočet tepelných ztrát

Samotný výpočet tepelných ztrát se řídí podle několika norem ČSN. Zdroj dat [4]. Normy pro výpočet součinitele prostupu tepla:

- ČSN 06 0210: květen 1994
- ČSN EN ISO 14683:2000
- ČSN EN ISO 13370:1999
- ČSN EN 12831: březen 2005
- ČSN 73 0540-4:červen 2005
- ČSN 73 0540-3:listopad 2005

Z výše uvedených norem se získá metodika výpočtu součinitele prostupu tepla. Podle tohoto součinitele prostupu tepla a dalších parametrů se vypočte tepelná ztráta budovy a podle požadované teploty se navrhnou výkony tepelných těles. Vypočtené ztráty jsou 168W pro horní izolovanou místnost a 520W pro dolní neizolovanou místnost. Podle vypočtených ztrát a podle volených radiátorů byl dimenzován i objem kotle. Topenářskou firmou byl navrhnut, po předložení požadavků a plánů, na objem 40l bez obsahu potrubí a radiátorů.

4.4 Realizace

Celá konstrukce byla navržena a konzultována s topenářskou firmou. Následně byla touto firmou provedena celá konstrukce včetně dodávky potřebných zařízení a komponent do modelu. Stejná firma dodala i solární kolektor a zkonstruovala elektrokotel včetně jímek pro čidla.

4.4.1 Konstrukce

Elektrokotel je na míru svařená nádoba z železných plechů tloušťky 5mm. V horní části jsou dva otvory na přívod otopné vody ze solárního kolektoru a z topného okruhu. Ve střední části je zabudováno topné těleso o výkonu 1000W. V horní a dolní části boční stěny jsou umístěny jímky pro čidla teploty[19]. Ve spodní části kotle jsou vývody pro solární kolektor a topný okruh. Pro odečet tlaků v kotli je na boční stěně umístěn analogový tlakoměr včetně analogového snímače teploty. Dále je zde použita možnost míšení vody z elektrokotle a vody v potrubí vracející se z topného okruhu pomocí směšovacího ventilu ovládaného hlavicí Siemens. Jako potrubí bylo použito běžně využívaných měděných trubek o průměru 15mm. Spojeny jsou cínovou pastou. Všechna šroubení jsou zaizolována silikonovými páskami. Ve střešní části modelu je umístěn solární kolektor. Ten je k topnému okruhu připojen pomocí dvou vlnovců o délkách 50cm a 1m. Celý přívod k solárnímu kolektoru je možné odpojit pomocí dvou ventilů. Pro eliminaci nárazových tlaků a pro udržení tlaků je zde expanzní nádoba o objemu 1,5 l a o maximálním tlaku 5 barů. K odvzdušnění topného okruhu je v nejvyšší části budovy upouštěcí ventil. Solární kolektor má rozměry 80x80x7 cm. Je postaven na výkon 800 W. Pro zajištění bezpečnosti tepelného okruhu je na horní části kotle teplotní čidlo. Má dvě pojistné části. První část se nastaví na požadovanou teplotu, kdy má čidlo vypnout přívod proudu do topného tělesa. Nastavení se provádí otočným kolečkem s naznačenými teplotami. Pokud dojde k překroční nastavené teploty a následnému vypnutí proudu, stačí počkat na vychladnutí vody v kotli a stlačit tlačítko k obnově přívodu proudu. Druhá část pracuje na stejném principu. Jen má několik odlišností. Pevně je stanovena teplota 90 stupňů celsia. Pokud dojde k vypnutí proudu v důsledku překročení této teploty, je nutné toho čidlo rozebrat a uvnitř znova zapnout pojistku. Jako poslední ochrana je zde tlakový ventil. Při překročení tlaku 1,8 bar začne upouštět vodu do okolí budovy. Běžný provozní tlak tepelného okruhu je kolem 1,2 baru.

4.4.2 Otopná tělesa

Pro vytápění objektu byly použity běžně používané deskové radiátory[20] podle vypočtených výkonů. Jednotlivé radiátory jsou opatřeny ventily pro regulaci přívodu topné vody. V horní místnosti, více izolované, je na ventilu umístěna termohlavice vytvořená během této diplomové práce. V dolní místnosti je na ventil od radiátoru umístěna hlavice firmy Siemens STA71. Oba radiátory jsou vybaveny samoodvzdušňovacími ventily. Též je možné jednotlivé radiátory odpojit od topného okruhu po uzavření ventilů na přívodním i na odtokovém potrubí radiátoru. Uchycení na stěnu je provedeno pomocí čtyř železných plátů o tloušťce 3 mm a šířce 50 mm ve tvaru J, které jsou šroubky

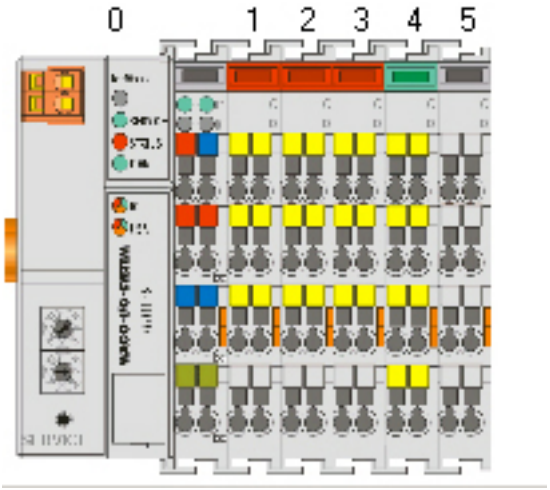
[19] Odporová čidla Pt100

[20] Ocelový deskový radiátor typ Kermi

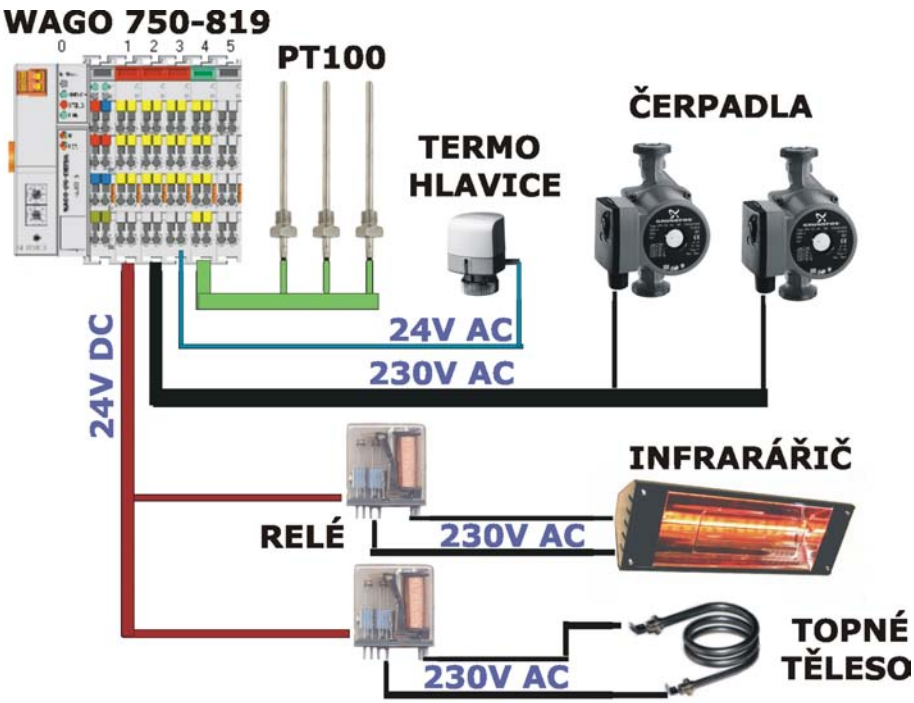
přichyceny ke stěně.

4.4.3 Elektrická konstrukce

Elektrická konstrukce je složena z napájení 24V střídavých, 24V stejnosměrných pro řízení elektroniky a 230V střídavých pro napájení topné části. Následující obrázek [35] ukazuje konfguraci Wago PLC 750-819 s rozšiřujícími moduly. V prvním slotu jsou zapojeny dvě relé na spínání topného tělesa o výkonu 1000W v kotli a na spínání infrazářiče o výkonu 2000W. V druhém slotu je zapojena hlavice Siemens STA71, která ovládá směšovací ventil. Třetí slot ovládá obě čerpadla pohánějící vodu v solárním kolektoru a v radiátorech. Poslední čtvrtý slot je spojen s čidly teploty PT100. Schéma [36] ukazuje elektrické zapojení.



Obrázek 35: Zapojení přídavných slotů v PLC Wago



Obrázek 36: Elektrické zapojení celého otopného systému

4.4.4 Použité ovládací prvky

Pro ovládání topného tělěsa a infrazářiče bylo zvoleno relé od firmy FINGER, typ FINGER 4031. Maximální spínací proud 20A, trvalý proud 10A, max. spínaný výkon 2500 kVA. Spínané maximální napětí 380V. Mechanická životnost je udávána 10^7 cyklů. Jako čidla teploty byly zvoleny tyčová odporová čidla PT100. Čerpadla byla použita Grundfos, typ UPS 25-40 180. Jsou nastaveny v nejnižším výkonovém stupni. Vzhledem k jejich předimenzovanosti je jejich výkon na tomto stupni dostačující.

4.5 Regulátor

Regulátor pro celý otopný systém se skládá ze tří částí. První část je regulace v kanceláři č.1. Regualci v této místnosti zajišťuje PLC Siemens a bylo nastaveno v rámci diplomové práce Václava Vozára. Druhý regulátor je v kaceláři č.2. Jedná se o regulátor vytvořený v rámci této diplomové práce. a třetí regulátor reguluje teplotu v nádrži, která dodává teplo do jednotlivých radiátorů. Tento regulátor je zabudován do PLC Wago. Tedy jedná se o tři na sobě přímo nezávislé regulátory. Pro další popis byl vybrán posledně zmiňovaný regulátor.

4.5.1 Hardware pro regulátor

Regulátor se skládá z modulárního I/O systému Wago řady 750. Přesné označení typu je 750-819. Obsahuje rozhraní LonWorks, možnost nastavení adresy. V dolní části mohulu je konfigurační rozhraní např. pro bluetooth nebo seriový kabel. Na druhé straně od

seriového rozhraní je přepínač chodu PLC mezi RUN a STOP. Vlevo je důležitý spínač SERVICE PIN sloužící k identifikaci Waga v síťi LonWorks. Následují diagnostické LED diody a modul pro napájení. V této části je velice důležité vhodně zapojit napájecí a výkonné napětí. Předejde se tím případnému zničení PLC. Nesmí se propojit horní řada napájecích otvorů s dolní řadou. Mohlo by dojít k připojení vyššího napětí na napájení a tím dojde ke zničení celé elektroniky PLC. Proto byly v modelu vybudovány dva okruhy 24V stejnosměrných. Jeden okruh je určen pro elektroniku a druhý pro výkonnou část. Toto opatření je zavedeno kvůli možnému přehlédnutí kontaktních plíšků v těle modulu, které mohou spojit napětí zapojené do modulu na napětí v hlavním modulu. K hlavnímu modulu je připojeno pět přídavných modulů. První tři jsou výstupní digitální přepínací releové bezpotenciálové moduly, typ 750-517. Následující modul má analogový vstup a slouží k připojení odporových čidel PT100. Typ je 750-460. Modul se nedá nastavovat, proto se musí použít pouze tento typ čidla. Pokud použijeme jiný či není umístěn žádný, je to indikováno rozsvícenou červenou LED diodou na příslušném místě modulu. Poslední modul je zakončovací tzv. terminátor sběrnice. Při špatném připojení ukončovacího modulu nebude správně fungovat sběrnice v PLC Wago. Poslední modul je typ 750-600. Je nutné ho připojit na konec všech modulů vždy, pokud pracujeme se sběrnicí v PLC Wago.

4.5.2 Software regulátoru

Regulátor v PLC je typu PID. Konstanty do tohoto regulátoru je možné vkládat ze sítě LonWorks. Následující schéma [37] ukazuje vstupní a výstupní proměnné.

Obrázek 37: Vstupní a výstupní proměnné Wago regulátoru

Celý program je strukturován do sedmi celků. Každý celek má nastaroti svoji specifickou činnost, která je popsána níže.

```
(* Probehne inizializace *)
IF(NOT(INIT)) THEN
INICIALIZACE(); (* inicializace promennych *)
END_IF;

(* vlozi hodnoty sitovych promennych*)
(* do prislusnych programovych promennych *)
(* uprava teploty podle simulvaneho pocasi *)
LINKERS();
```

```
(* overeni spravnosti rozsahu promennych *)
CHECKERS();

(* definovane casovace *)
CASOVANI();

(* definovane citace *)
CITACE();

(* samotny progrma regulace *)
REGULACE();

(* ochrana pred prekrocenim kritickych hodnot *)
WATCHDOGS();
```

Regulátor vypočítává čas, po který bude sepnuto topné těleso. Podle zadaných PID konstant se vypočítá regulační zásah a ten se převede na délku pulzu. Maximální délka pulzu je dána podle vstupní hodnoty času vzorkování.

```
IF(VystupCitacSample=TRUE) THEN
    Pomoc1:=(LocTempInTanker/10);
    Pomoc2:=(TempOutTanker/10);
    Pomoc3:=(Pomoc1-Pomoc2);
    P:=Pomoc3*PC;
    I:=B*PI;
    Pomoc4:=(TempLast/10);
    Pomoc5:=(Pomoc2-Pomoc4);
    D:=Pomoc5*PD;
    B:=P+I+D;
    IF(B<0) THEN
        B:=0; (* anti wind-up*)
    END_IF;
    IF(B>CAS_SAMPLE) THEN
        B:=CAS_SAMPLE; (* anti wind-up*)
    END_IF;
    VyslednyZasah:=REAL_TO_INT(B);
    Cas_ON:=VyslednyZasah;
    Cas_OFF:=CAS_SAMPLE-Cas_ON;
    TempLast:=TempOutTanker;
END_IF;
```

Změna teplot resp. simulace počasí je zajišťována v části *LINKERS()*. Zadaný offset je odečten od aktuální teploty. Tím se vytváří simulace změn teploty resp. počasí.

4.5.3 Tepelné okruhy

V modelu jsou vytvořeny dva tepelné okruhy, které mají společné místo v topné nádrži. První okruh je okruh solární. Vedení vody je vytvořeno měděným vedením a spojuje nádrž, hnací pumpu a solární článek. V těsné blízkosti solárního kolektoru je umístěn odvzdušňovací ventil. Umístění v této části je z důvodu nejvyššího položení a tím i lepší funkce odvzdušňování. Vedle solárního kolektoru je umístěna expanzní nádrž, která slouží k vyrovnávání rázových tlaků v potrubí a též k udržování tlaku v potrubí. Druhý okruh je složen z potrubí přivádějícího ohřátou vodu k radiátorům. Jedná se o dva radiátory. Každý radiátor je v jiné místnosti. V dolní kanceláři č.1 je umístěn větší radiátor z důvodu menší izolace. Ovládání a regulace je zajištěno ventilem Siemens a termohlavicí Siemens STA71. Regulátor je v PLC Siemens DESIGO RXC31.1 nahrán a jeho činnost se ovlivňuje vybráním aplikace. Tato část byl vytvořená v rámci diplomové práce Václava Vozára. V horní kanceláři č.2 je umístěn menší radiátor. Je zde silná izolace. Ventil je od firmy Siemens, termohlavice je od firmy Minařík typ ETP - 23 - 2. Samotný regulátor je vytvořen v rámci této diplomové práce. Každý radiátor je opatřen odvzdušňovacím ventilem.

.

5 Závěr

V diplomové práci jsem se seznámil s technologií LonWorks. Realizoval jsem pro model administrativní budovy zařízení pro regulaci teploty radiátoru ústředního topení. Vytvořil jsem pro toto zařízení návrh desky plošných spojů, včetně výběru vhodných elektronických dílů s využitím Neuron chipu a transceiveru FTT-10A. Úspěšně jsem přepnul komunikaci v Neuron chipu, což mimo jiné vyžadovalo několikrát opakovat celý postup, než se konfiguraci podařilo zapsat do EEPROM procesoru. Seznámil jsem se s programovacím prostředím NodeBuilder a s programovacím jazykem Neuron C, ve kterém jsem vytvořil software regulátoru teploty. Při jeho tvorbě jsem dodržel standardizovaný profil organizace Lonmark pro zařízení VAV Controller. Vytvořený nod je plně zakomponován do sítě LonWorks kde komunikuje s čidlem teploty. Toto zařízení bylo umístěno v kanceláři č.2 v modelu automatizované budovy v laboratoři katedry řídicí techniky na Karlově nám.

V modelu budouvy jsem dále navrhnl a posléze zrealizoval celý otopný systém z běžně dostupných topenářských výrobků. Skládá se ze dvou radiátorů, solárního kolektoru ohřívaného infrazářičem (simulace slunečního záření) a elektrokotle. Voda v soustavě je poháněna dvěmi čerpadly. Samotná nádoba elektrokotle byla vyrobena na zakázku topenářskou firmou. Do PLC Wago umístěného v technické místnosti jsem navrhl a implementoval algoritmy regulátoru pro řízení výstupní teploty z elektrokotle a celé tepelné soustavy.

V druhé části této diplomové práce jsem se zabýval vizualizací domu a možností jeho ovládání přes internet pomocí běžného webového prohlížeče. Za tímto účelem jsem se seznámil s jazykem VRML, jenž byl vyvinut pro modelování 3D objektů v běžném internetovém prohlížeči. V jazyce VRML jsem následně vytvořil kompletní virtuální model budovy včetně jejího okolí a vybavení interiérů.

Takto vytvořený model je však statický. Aby bylo možné ovládat jeho vybavení, jako jsou světla, plátno v prezentační místnosti, výtah a další, bylo nutné implementoval v jazyce Java příslušné objekty. S jejich pomocí jsem provázal chování jednotlivých aktivních prvků s akčnímy členy (okno, dveře, světlo atp.) virtuálního domu.

Jedním z požadavků uvedených v zadání této diplomové práce bylo svázání vytvořeného virtuálního domu a skutečného modelu. Toho jsem úspěšně dosáhl za pomoci knihovny v jazyce Java od společnosti Echelon a LNS serveru, jenž je nainstalován na počítači v reálném modelu. Při implementaci tohoto propojení se vyskytly problémy s nekompatibilitami mezi verzemi Java 1.1 a vyšší. Kroky, které řeší tyto potíže jsou podrobně popsány v kapitole 3.1.1.

Model administrativní budovy je velice zajímavý projekt. Během prací na tomto projektu mne napadly další kroky vedoucí k jeho rozšíření a vylepšení. Vzhledem k tomu, že je plánováno vystavovat tento model na výstavách, bylo by vhodné celý model opatřit nátěrem, jenž by připomínal omítku. Do interiéru umístit umělé květiny, obrázky a koberce do kanceláří. Obrazy a květiny mohou opticky srovnat disproporce v rozměrech modelu. Dále by bylo zajímavé opatřit model reproduktory a vytvořit umělou inteligenci spojenou s indentifikačním systémem. Inteligence by pak mohla komunikovat

s uživatelem hlasem. Spíše komentovat události, které by vykonávala v modelu. Poté co bude umístěn v modelu robot, bylo by zajímavé vytvořit virtuální model reálného domu, ovšem s přesnými rozměry. Poté pomocí appletu v Javě navigovat robota tak, jak se pohybuje Avatar ve virtuálním světě. Již vytvořený virtuální svět je nutné po dokončení reálných zařízení rozšířit o jejich virtuální kopie. Tedy doplnit třídu LNScommander o další příkazy ovládající nové zařízení.

Dále by bylo vhodné umístit na otopný systém průtokoměr, aby se lépe řídil směšovací ventil. Pro řízení tepelné rohože je pouze implementována vstupní proměnná v PLC Wago v technické místnosti modelu. Bylo by však zajímavé vytvořit regulátor například v softwarovém PLC. Vedle spojení s virtuální realitou by bylo možné využít i samotného grafického prostředí Javy pro vytvoření jednoduššího řídícího rozhraní, které by dle očekávání bylo rychlejší, než stávající implementace komunikující přes OPC server.

Reference

[1] INFOENERGIE.CZ; *Úspory energie v domácnostech* [online]. Poslední revize 2006-02-28 [cit. 2004-12-13], ⟨http://www.infoenergie.cz⟩.

[2] LONMARK.ORG; *LonMark International* [online]. Poslední revize 2006-02-28 , ⟨http://www.lonmark.org/⟩.

[3] TZB-INFO.CZ; *TZB-info: Tepelné mosty ve stavebních konstrukcích* [online]. Poslední revize 2006-03-06 , ⟨http://www.tzb-info.cz/⟩.

[4] TZB-INFO.CZ; *Porovnání výpočtů tepelných ztrát podle ČSN 06 0210 a ČSN EN 12831* [online]. Poslední revize 2006-03-06 , ⟨http://www.tzb-info.cz/⟩.

[5] WIKIPEDIA.ORG; *Tepelná čerpadla* [online]. Poslední revize 2006-04-17 , ⟨http://en.wikipedia.org/⟩.

[6] WEB3D.ORG; *Web3D Consortium* [online]. Poslední revize 2006-05-16 , ⟨http://www.web3d.org/⟩.

[7] WWW.CGG.CVUT.CZ; *Virtual Old Prague Project* [online]. Poslední revize 2006-05-16 , ⟨http://www.cgg.cvut.cz/vsp/⟩.

[8] WWW.CGG.CVUT.CZ/VYUKA/36MUS; *CGG Prague - Education* [online]. Poslední revize 2006-05-18 , ⟨http://www.cgg.cvut.cz/vyuka/36MUS⟩.

[9] SERVICE.FELK.CVUT.CZ/COURSES/36MUS; *CGG Prague - Education* [online]. Poslední revize 2006-05-18 , ⟨http://service.felk.cvut.cz/courses/36MUS/⟩.

[10] MARTIN LINHART *Diplomová práce* Komunikační sběrnice LON (Local Operating Network), Praha 2004

[11] TOSHIBA *Datasheet* Neuron Chip Local Operating Network LSIs, 1999

[12] JIŘÍ ŽÁRA *VRML 97* Laskavý průvodce virtuálními světy, 1999

[13] MICHAL SLEZÁK *Aplikační manuál* LonWorks termohlavice ETP-23-3, 2005

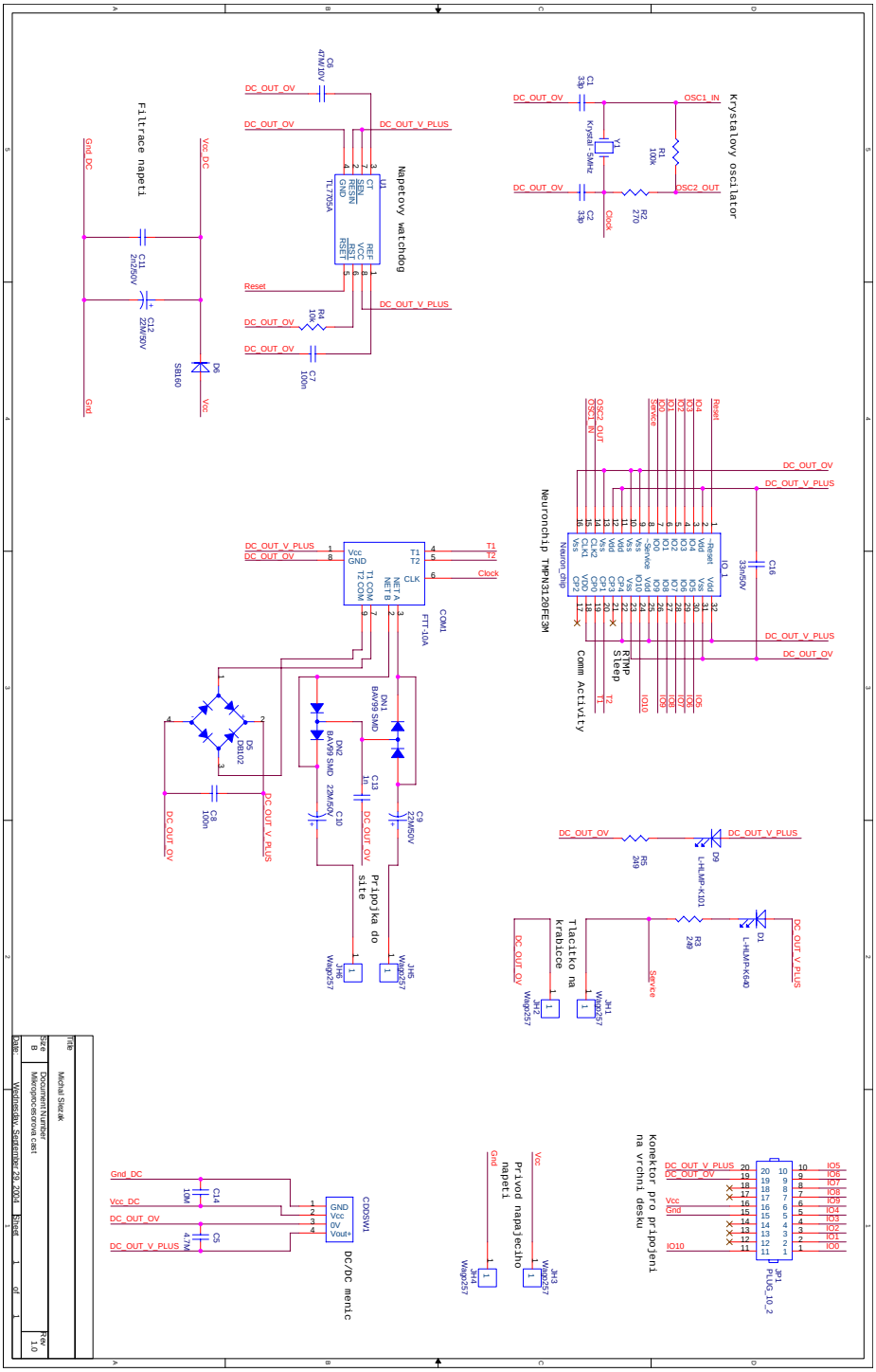
[14] ECHELON *Referenční manuál Neuron C* Neuron C Programmers Guide, Revize 5, 2001

[15] WAGO *Wago modular I/O system 750-319,750-819* Technická specifikace,instalace a konfigurace, Verze 2.0.0

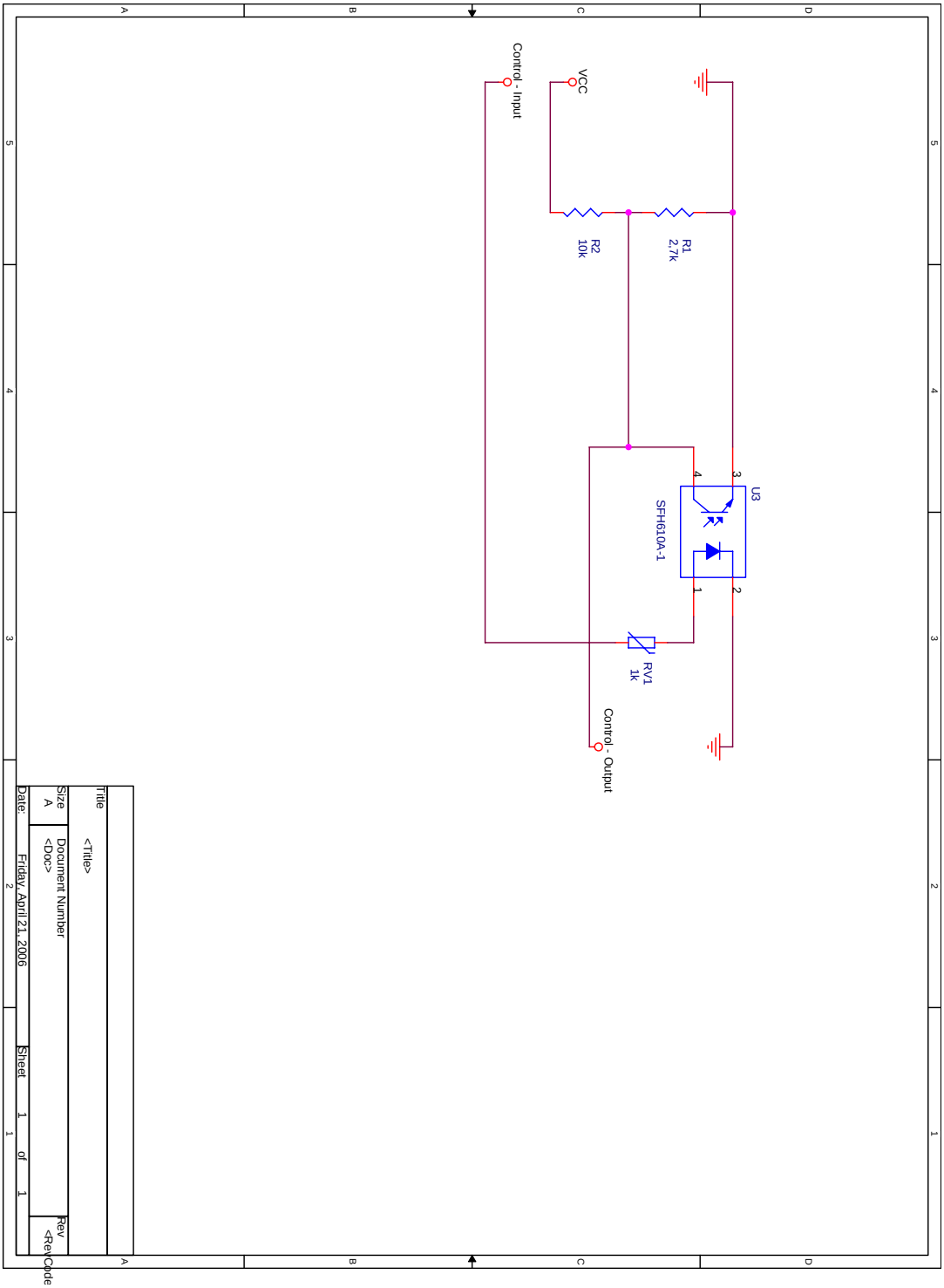
A Dokumentace LonWorks

A.1 Elektrická zapojení

A.1.1 Procesorová část



A.1.2 Ovládací část



A.2 Seznam součástek

A.2.1 Procesorová část

Pořadí	Počet	Název	Hodnota
1	1	COM1	FTT-10A
2	2	C2,C1	33p
3	1	C5	4.7M
4	1	C6	47M/10V
5	2	C7,C8	100n
6	3	C9,C10,C12	22M/50V
7	1	C11	2n2/50V
8	1	C13	1n
9	1	C14	10M
10	1	C16	33n/50V
11	1	DC_DC1	CDDSW1
12	2	DN1,DN2	BAV99 SMD
13	1	D1	L-HLMP-K640
14	1	D5	DB102
15	1	D6	SB160
16	1	D9	L-HLMP-K101
17	1	IO_1	Neuron_chip
18	6	JH1,JH2,JH3,JH4,JH5,JH6	Wago257
19	1	JP1	PLUG_10_2
20	1	R1	100k
21	1	R2	270
22	2	R3,R5	249
23	1	R4	10k
24	1	U1	TL7705A
25	1	Y1	Krystal - 5MHz

A.2.2 Ovládací část

Pořadí	Počet	Název	Hodnota
1	1	R1	10k
2	4	JP1,JP3,JP4,JP5	ARK550_2
3	1	JP2	PLUG 10x2
4	1	R2	2,7k
5	1	TRIMR	1k
6	1	OPTOČLEN	SFH610-1

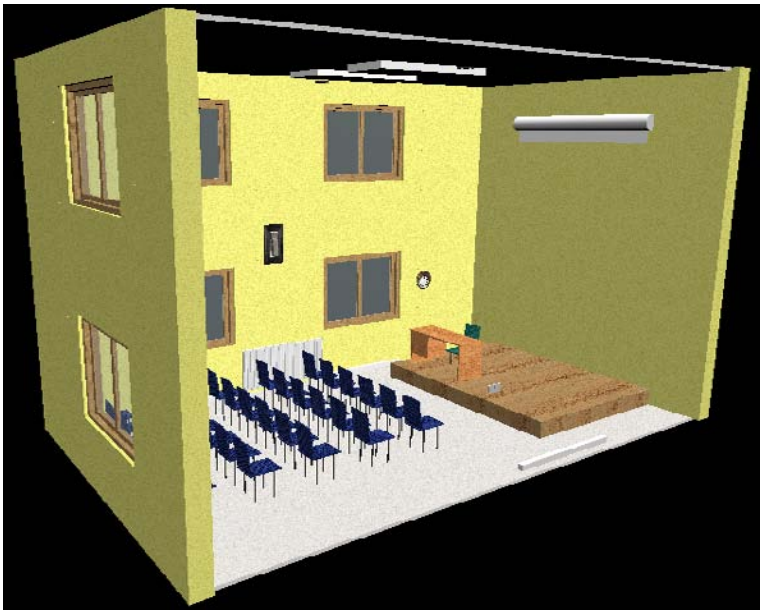
B Použité zkratky

ČSN ..... Česká státní norma
DOM ..... Document Object Model
EEPROM ..... Electrically Erasable Read Only Memory
ISA ..... Industrial Standard Architecture
ISO ..... International Organisation for Standardization
LD ..... Ladder diagram
LNS ..... LonWorks Network Services
LON ..... Local Operating Network
MAC ..... Media Access Control
OPC ..... Object Linking and Embedding for Process Control
OSI ..... Open System Interconnection
PLC ..... Programable Logical Controler
PWM ..... Pulse Width Modulation
VRML ..... Virtual Reality Modeling Language
RAM ..... Random Access Memory
ROM ..... Read Only Memory
SNVT ..... Standart Network Variable Type
ST ..... Structured text
URL ..... Uniform Resource Locator
X3D ..... Extensible 3D

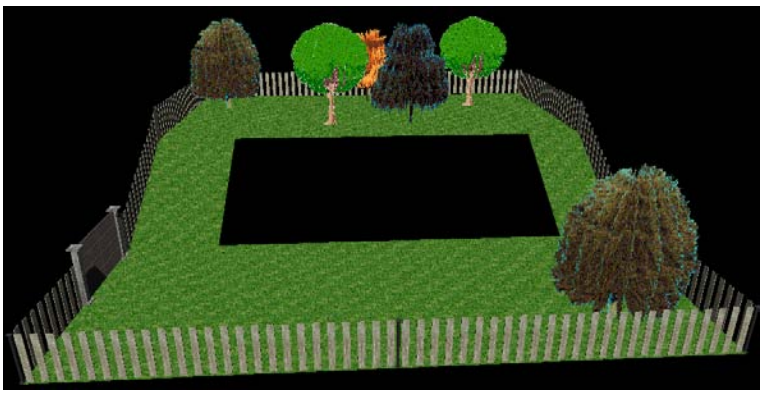
C Schémata

C.1 VRML model

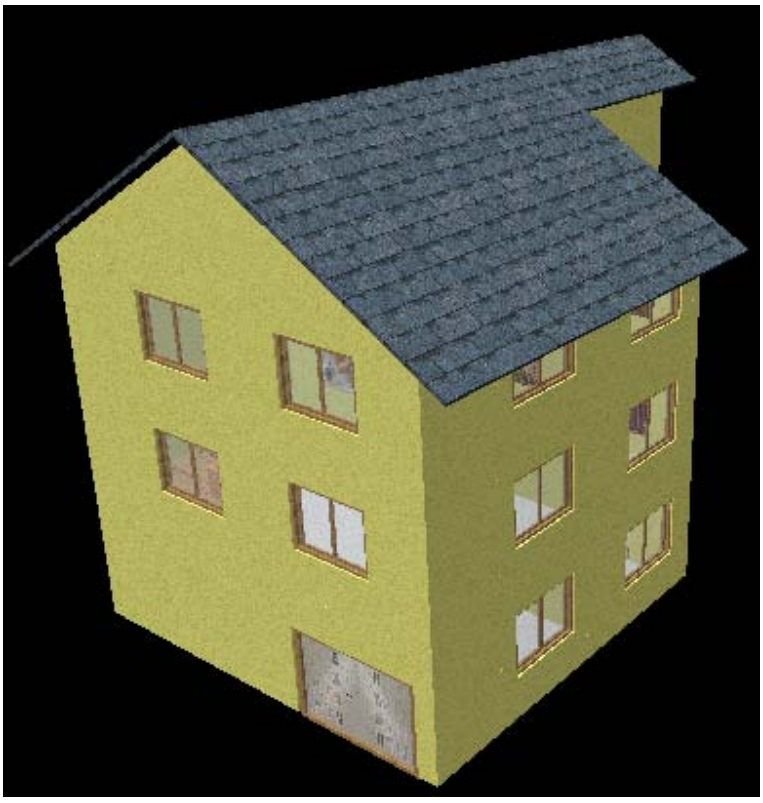
Obrázek 38: Celkový pohled na virtuální model



Obrázek 39: Průřez prezentační místností



Obrázek 40: Pohled na okolí modelu

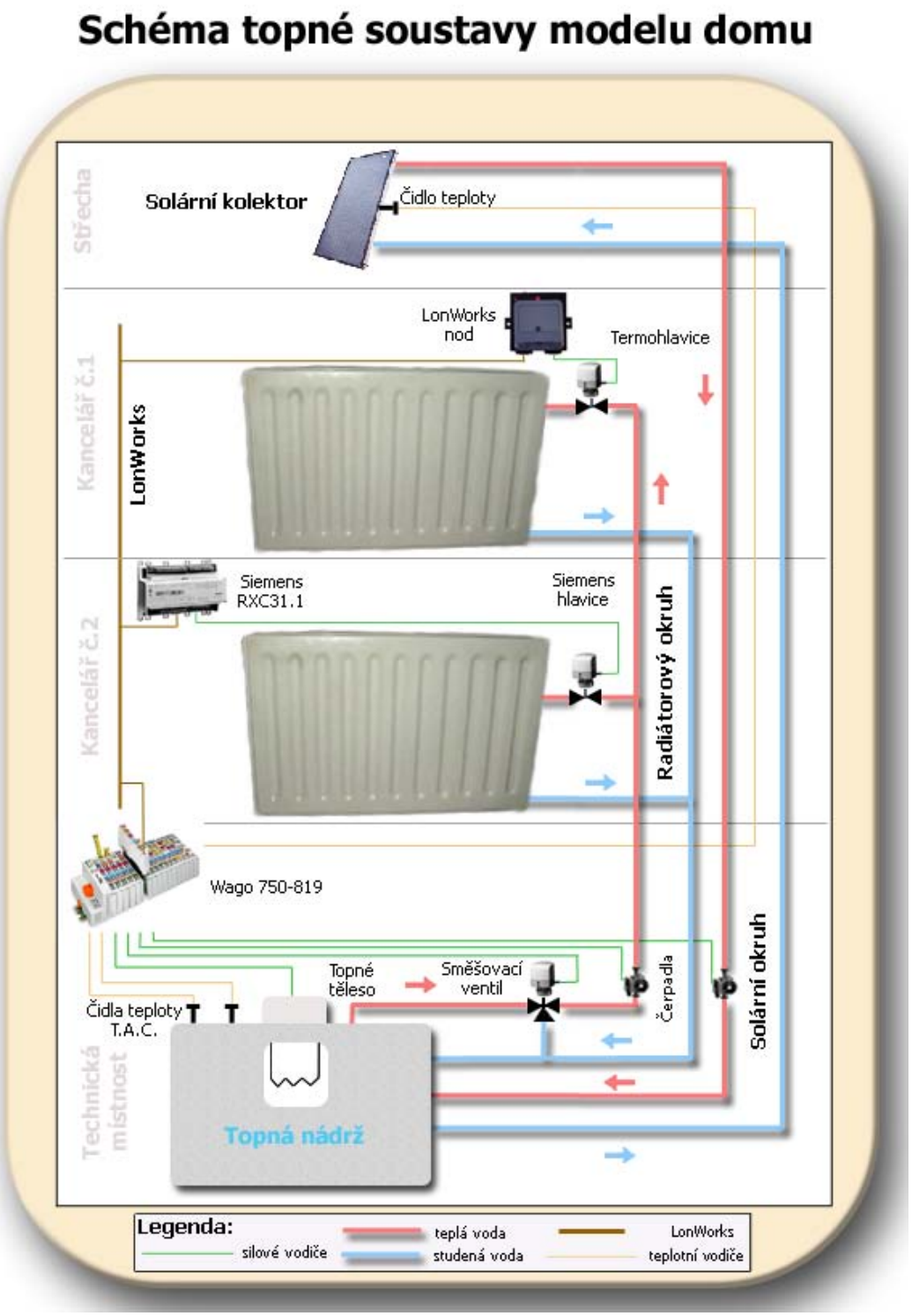


Obrázek 41: Samotný virtuální model

Obrázek 42: Pohled na kanceláře

Obrázek 43: Pohled na chodby ve všech patrech

C.2 Tepelná část

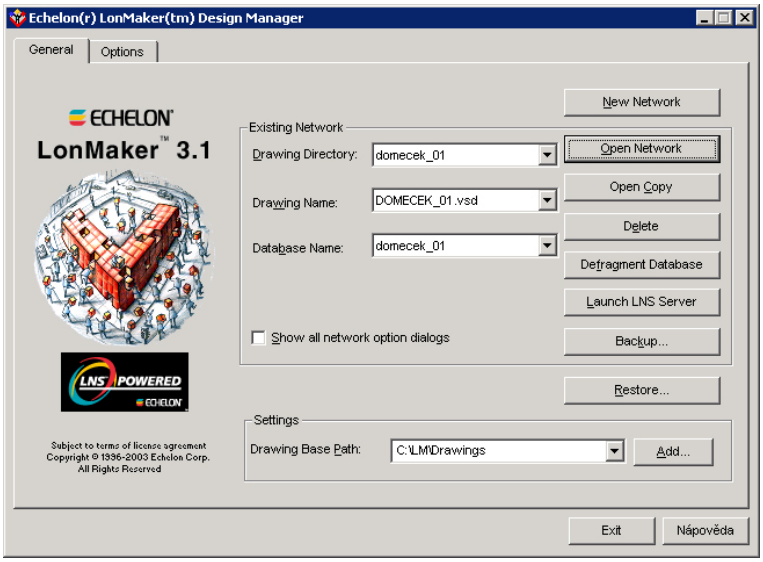


D Průvodci

D.1 LonMaker

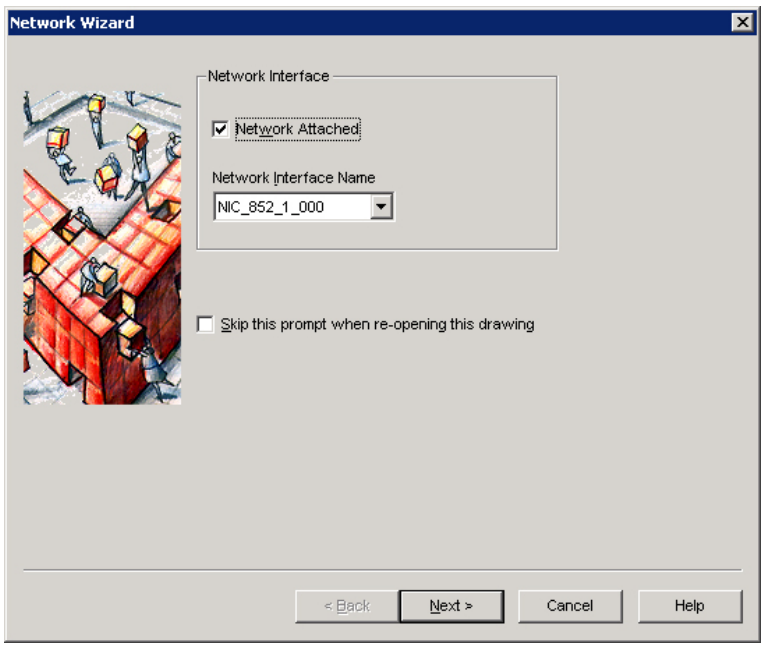
Vytvořený nod bylo potřeba zapojit do sítě LonWorks a provedena její konfigurace. K tomu byl použit konfigurační program LonMaker. Jedná se o program složený ze skupiny zásuvných modulů a maker v programu Microsoft Visio 2003. Důležité je při spouštění Microsoft Visio dodržet správný sled spouštění programu Iplongate a poté samotného Microsoft Visio. Pokud se nedorží tento sled, nedojde ke správnému napojení LonMakeru na LNS databázi. LonMaker vypíše chybové hlášení, ale pokračuje dál ve své činnosti. Takto spuštěným nástrojem nelze nic správně nakonfigurovat či načíst hodnoty proměnných. Po spuštění programu LonMaker a vybrání databáze proměnných pro administrativní budovu se spustí malý průvodce, ve kterém se zvolí přes jaké zařízení se bude přistupovat do sítě LonWorks. Zda se všechny změny konfigurace budou okamžitě zapisovat do sítě resp. do směrovacích tabulek zařízení. Podobnější kroky ke konfugiraci a zakomponování zařízení do sítě jsou následující:

- Pro správnou funkci LonMakeru je nutné spustit Iplongate
- Základní menu LonMakeru nabízí možnost vytvoření nové sítě a tím i databáze, načtení sítě, spuštění LNS serveru. Má ještě několik nabídek, ovšem ty nejsou nutné pro konfiguraci připojeného zařízení. Důležité je mít nastavenu cestu k plánu sítě v položce *Drawing base path*, vybrat správný adresář a plán sítě s koncovkou *.vsd*



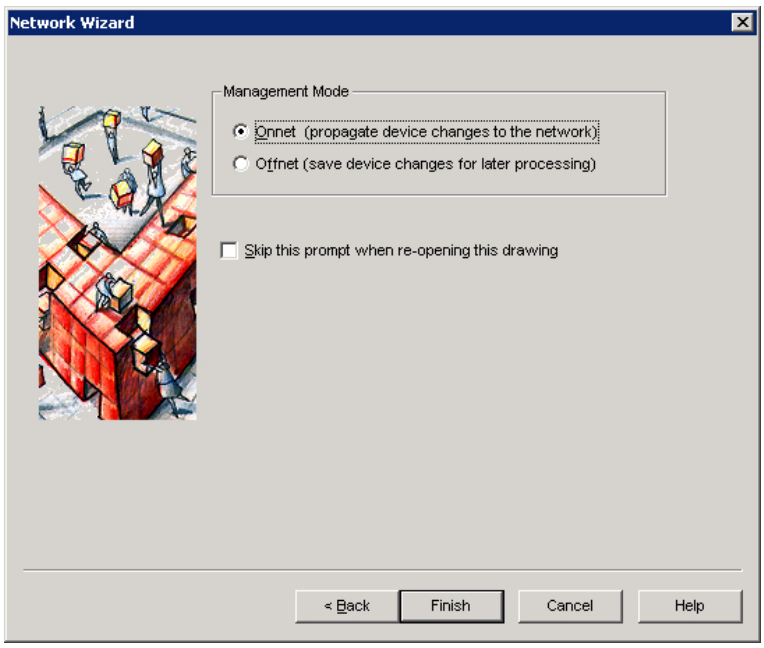
Obrázek 44: Základní menu LonMakeru

- Po výběru výkresu sítě a jejím otevření následuje výběr rozhraní přes které se bude přistupovat do LonWorks. Vybere se L-switch či X LON Dongle pokud je nainstalován. Pokud nechceme, aby se LonMaker na tuto volbu později ptal, zaškrtneme dolní políčko.



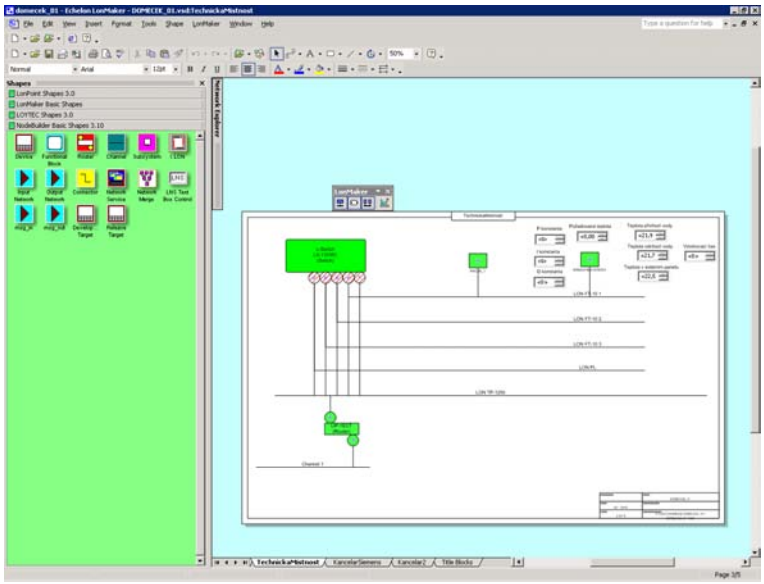
Obrázek 45: Výběr rozhraní pro přístup do LonWorks

- Vybere se způsob ditribuce konfigurace. Zda se mají změny v síti okamžitě zapisovat do konfigurace sítě volbou *Onnet* či se všechny změny provedou jen v plánu sítě a poté se nahrají do konfigurované sítě, pak se zvolí volba *Offnet*. Výhoda druhé volby je v tom, že síť je během konfigurace bezezměn, nedochází k nekonzistencím a všechny změny se nahrají až vjeden okamžik. V tomto případě, kdy není potřeba zajisit stálý chod stačí a je výhodnější volba *Onnet*.



Obrázek 46: Způsob propagace změn do konfigurace LonWorks

- Otevře se plán sítě, kde je zakreslena celá struktura sítě včetně routeru, L-switche a všech zařízení. Z důvodu přehlednosti jsou jednotlivé celky rozděleny na jednotlivé výkresy podle místností. Spojeny jsou pomocí komunikačních kanálů, které mají funkci sběrnice v daném celku. Je důležité vybrat správný komunikační kanál, který přísluší dané místnosti, kde je zařízení připojováno.



Obrázek 47: Základní výkres sítě

Nod pro řízení termohlavice byl fyzicky umístěn do technické místnosti v modelu

a proto byl vybrán výkres Technická místnost. Při konfiguraci v LonMakeru je v levé části nabídka, kde se nachází všechny předdefinované objekty, které lze vkládat do sítě. Důležité objekty a často používané jsou *DEVICE*, *FUNCTIONAL BLOCK*, *CONNECTOR* a *LNS TEXT BOX CONTROL*. Tahem levého tlačítka myši se přetáhne objekt *DEVICE* do výkresu. Následuje malý průvodce přidáním zařízení. Přesný postup je uveden v D.2.

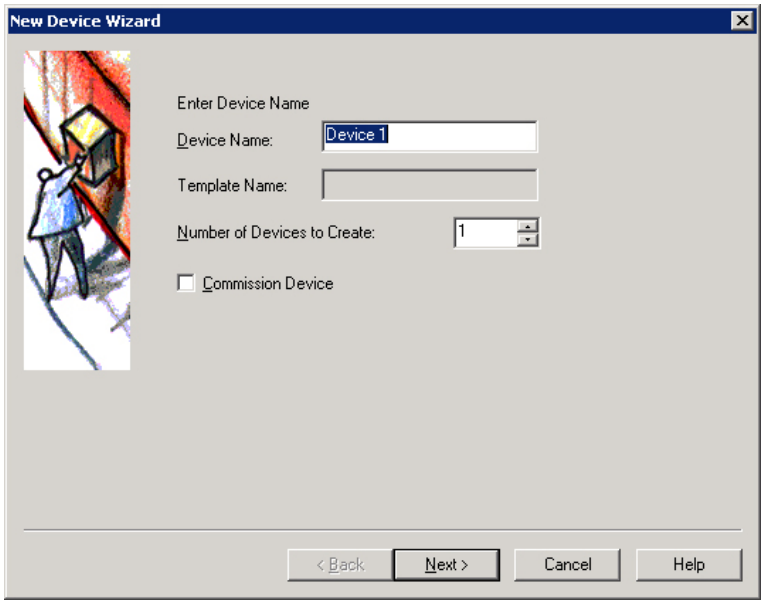
- Dalším krokem je propojení funkčních bloků resp. vstupních a výstupních proměnných v těchto blocích. Z levé nabídky se přetáhne objekt *FUNCTIONAL BLOCK* do výkresu. Následuje malý průvodce přidáním funkčního bloku. Přesný postup je uveden v D.3. Poté se již zobrazí vytvořený funkční blok včetně vyobrazení vstupních a výstupních proměnných. Vstupní jsou označeny šipkou směřující dovnitř bloku, výstupní směřují ven. Na spojení příslušných proměnných se použije nástroj *CONNECTOR*. Kliknutím na zvolenou výstupní proměnnou a tahem myší se připojí k požadované vstupní proměnné. Tímto se mezi proměnnými vytvoří spoj. Takto se propojí všechny vstupní a výstupní proměnné přidávaného zařízení. Konkrétně vstupní teplotu propojit s výstupní teploty čidla v regulované místnosti atp. Po skojení všech proměnných, je celé zařízení plně nakonfigurováno a je již připraveno k provozu.

- Celý nákres a konfigurace se uloží a provede se resynchronizace. Pokud Microsoft Visio 2003 vypíše chybu při ukládání, stačí znovu uložit projekt. Výkres bude již korektně uložen.

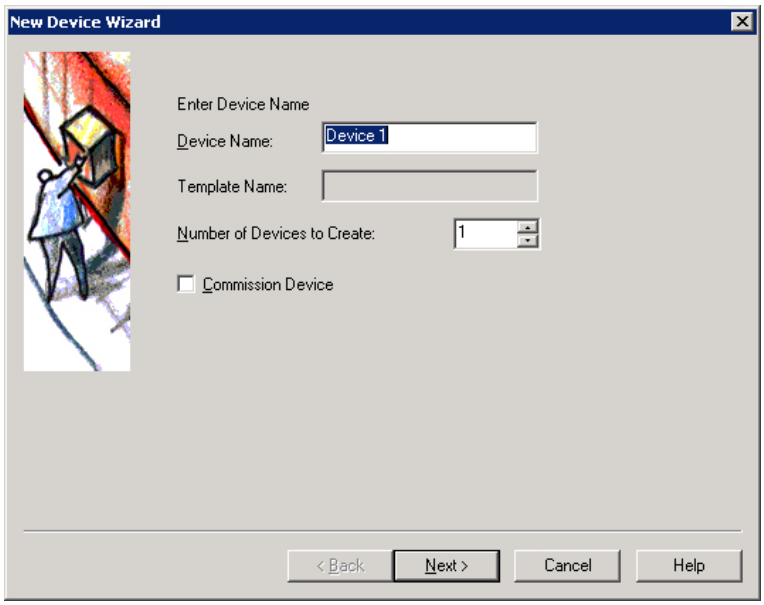
- Může se stát, že nákres nesouhlasí s fyzickým stavem sítě či se síť a další programy nechovají korektně. Může to být způsobeno nekonzistencí LNS databáze se stavem sítě. Lze to napravit použitím příkazu z nabídky v LonMakeru resp. Microsoft Visio. V nabídce se vybere *Lonmaker* → *Resynchronize..* . Spustí se synchronizace vybraných položek, které je nutné zvolit před spuštěním synchronizace.

Jestliže je nutné již v prostředí LonMakeru ovládat či číst proměnné, lze k tomu využít objektu *LNS TEXT BOX CONTROL*. Přetažením tohoto objektu na výkres se zobrazí velice jednoduchý a intuitivní průvodce, ve kterém se vybere na jakou proměnnou je směrovat tento objekt. Po výběr se již zobrazí obsah proměnné. Výběrem z lokální nabídky zvolit *Get value* pro načtení proměnné nebo *Set value* pro zápis zadané hodnoty do proměnné.

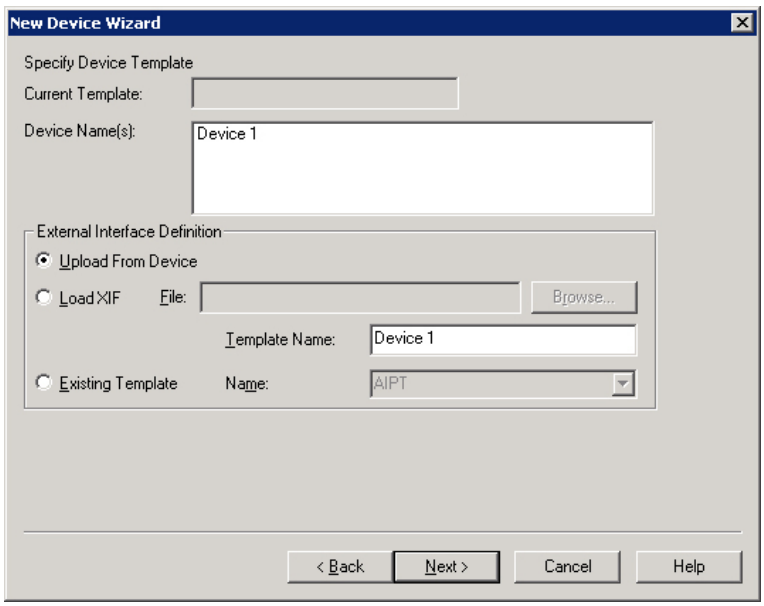
D.2 Přidání zařízení v LonMakeru



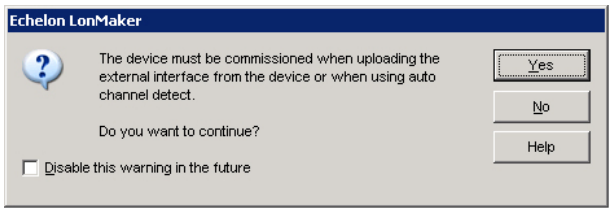
Obrázek 48: Zadání jména zařízení



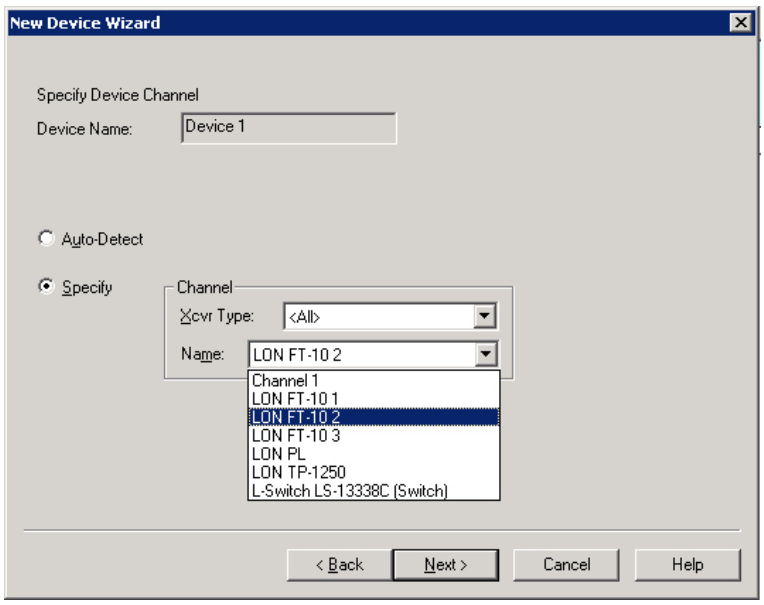
Obrázek 49: Výběr rozhraní - vybere se nahrát ze zařízení nebo z obrazu pokud je k dispozici



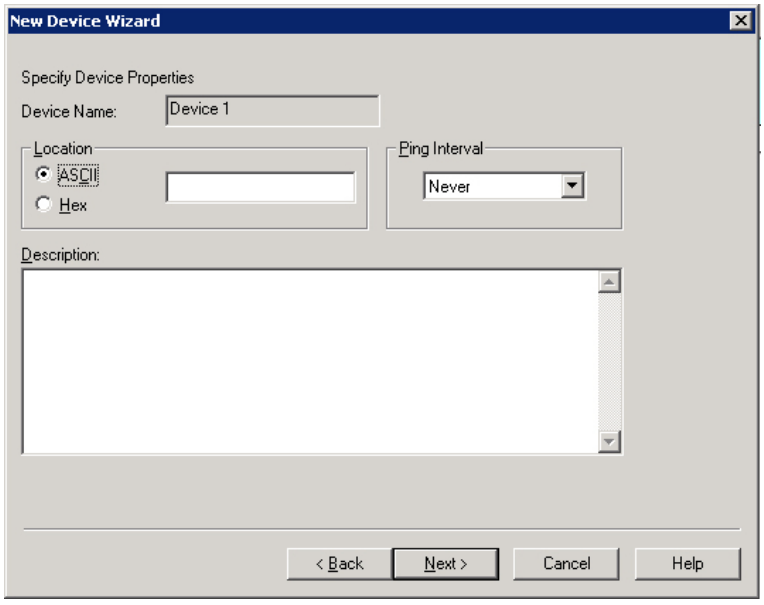
Obrázek 50: Výběr rozhraní - vybere se nahrát ze zařízení nebo z obrazu pokud je k dispozici



Obrázek 51: Z nabízeného vybereme ANO pro výběr kanálu. Autodetekce není spolehlivá

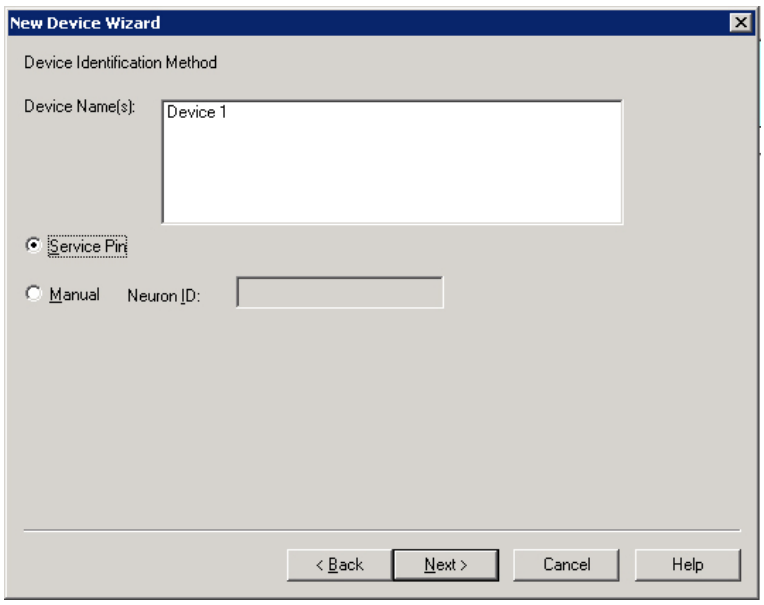


Obrázek 52: Vybere se kanál, ke kterému má být zařízení připojeno



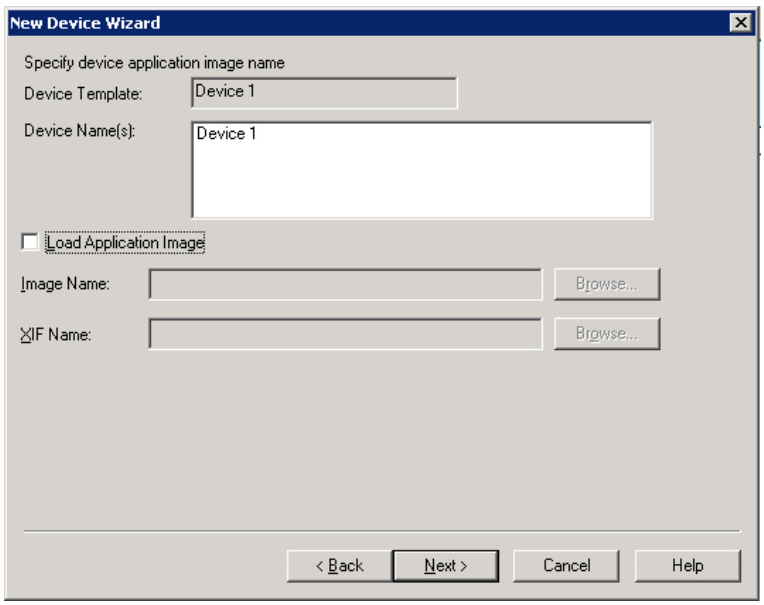
Obrázek 53: Vybereme jak často má být zařízení ověřováno na jeho přítomnost

Pokud vybíráme jakou metodou bude rozpoznáno připojené zařízení, je vhodné zvolit pomocí *service pinu*.Pokud zařízení neni korektně připojeno je to zjištěno hned tím, že LonMaker nedostane zprávu od zařízení.



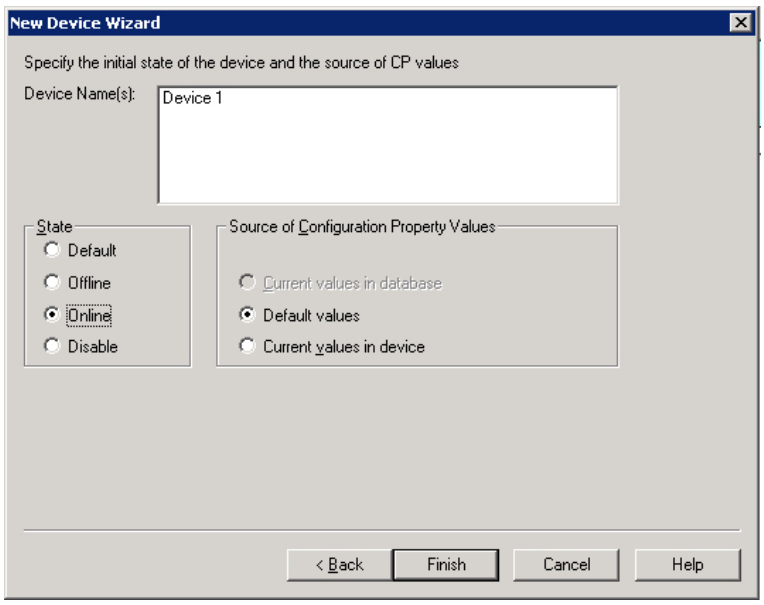
Obrázek 54: Výběr jakou metodou má být rozpoznáno připojené zařízení

Pokud již máme přeložený obraz programu, lze ho nahrát do zařízení.

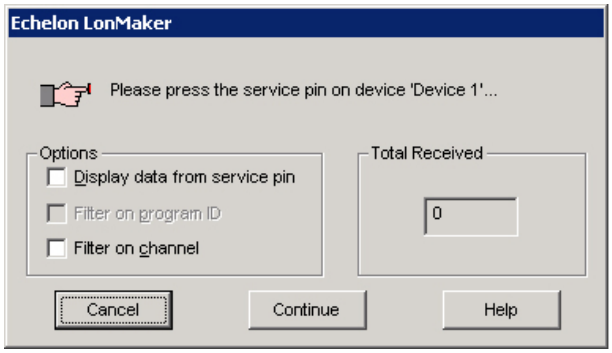


Obrázek 55: Nahrání aplikačního obrazu

Pokud je zařízení zapojeno do sítě, tak lze zvolit *Online* . Je vhodné vybrat *Default values* .

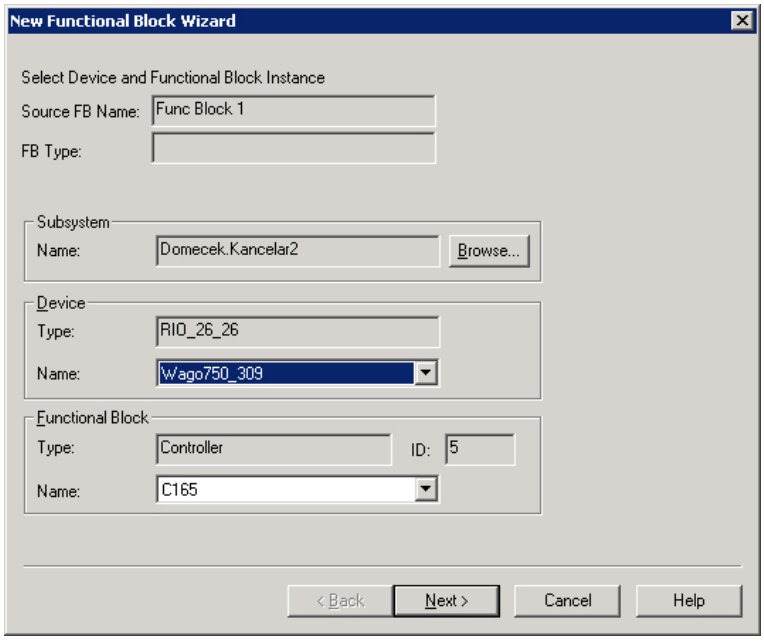


Obrázek 56: Výběr stavu zařízení a hodnot proměnných



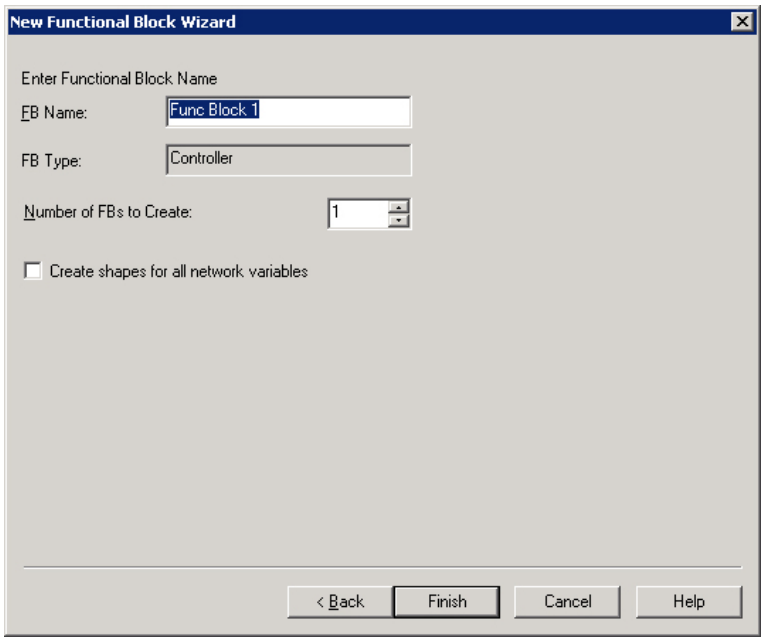
Obrázek 57: Výzva ke stisknutí servisního tlačítka

D.3 Přidání funkčního bloku v LonMakeru



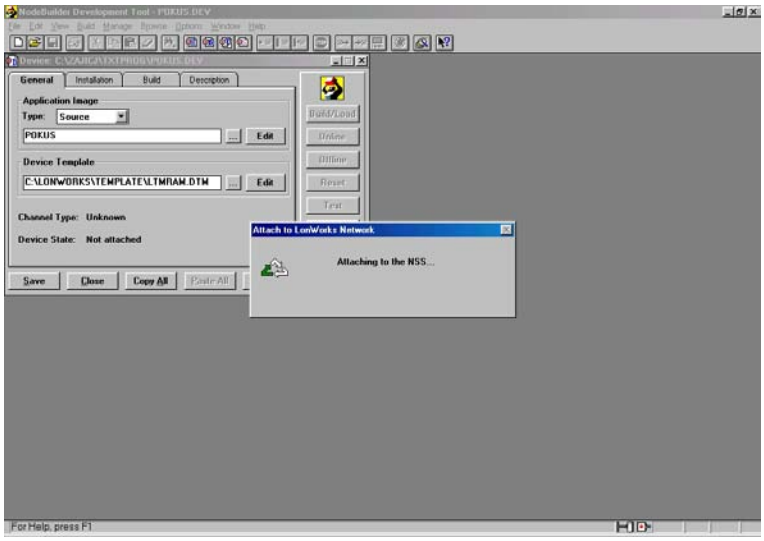
Obrázek 58: Výběr zařízení, které obsahuje požadovaný funkční blok a výběr toho funkčního bloku

Pokud je potřeba vytvořit funkční blok ze všech síťových proměnných, stačí zaškrtnout rámeček *Create shapes for all network variables.*



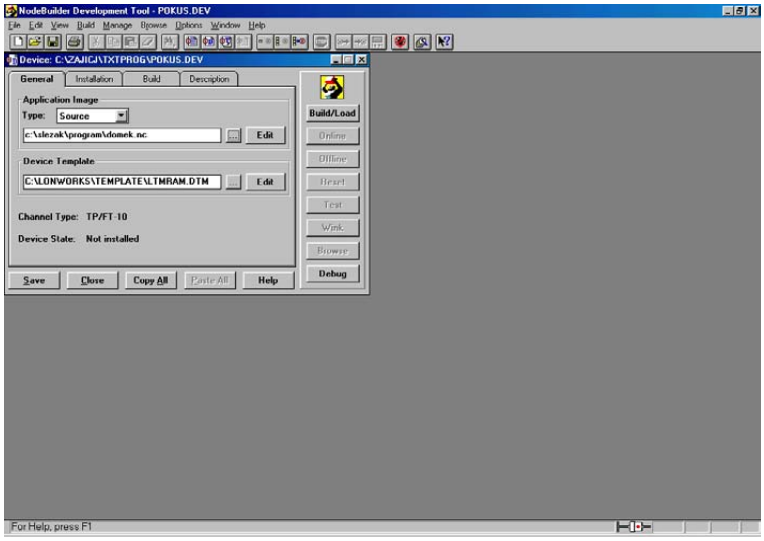
Obrázek 59: Pojmenování funkčního bloku

D.4 Implementace programu do Neuron chipu



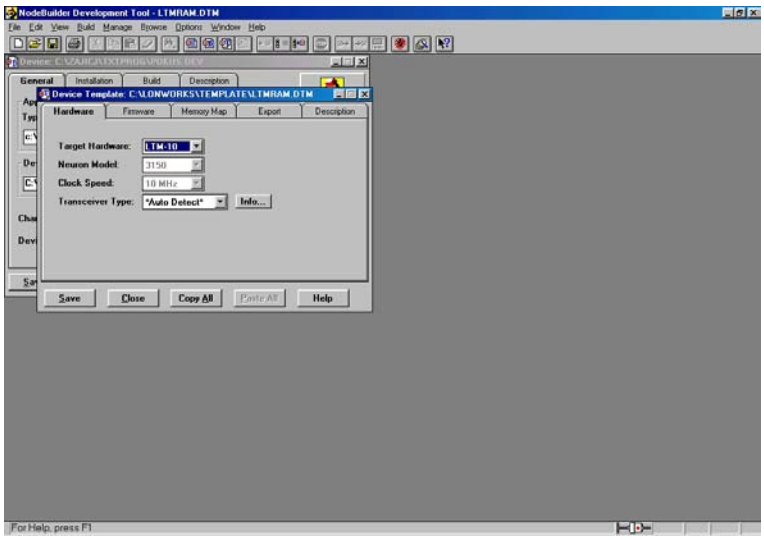
Obrázek 60: Spuštění Node Builderu a připojení do LonWorks

Po spuštění node builderu se zobrazí základní nabídka a prostředí. V nabídce se vybere příslušný program s koncovkou *.nc*. Dále se vybere příslušný typ procesoru a transceiveru v položce *Device template*.

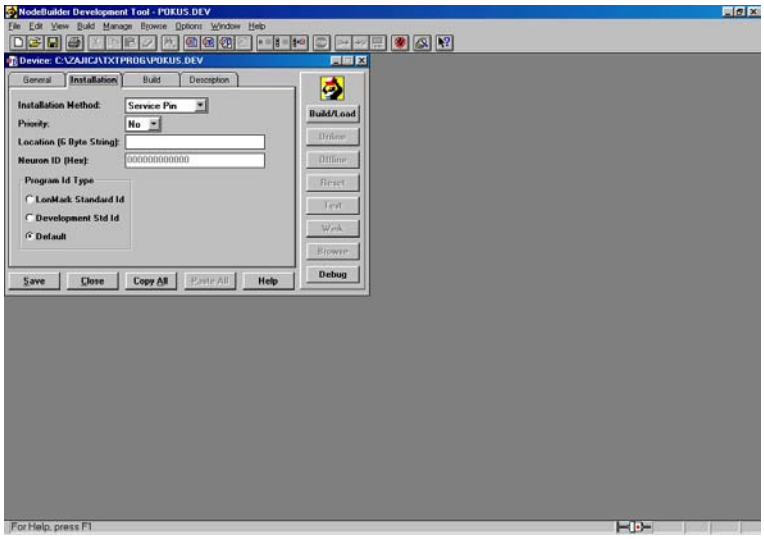


Obrázek 61: Node Builder a menu pro správu szařízení

V příšlušné šabloně *Device template* se nastaví pro jaký hardware se programu bude překládat, o jaký typ Neuron chipu se jedná, na jaké frekvenci pracuje a jakým transceiverem je připojen do sítě.

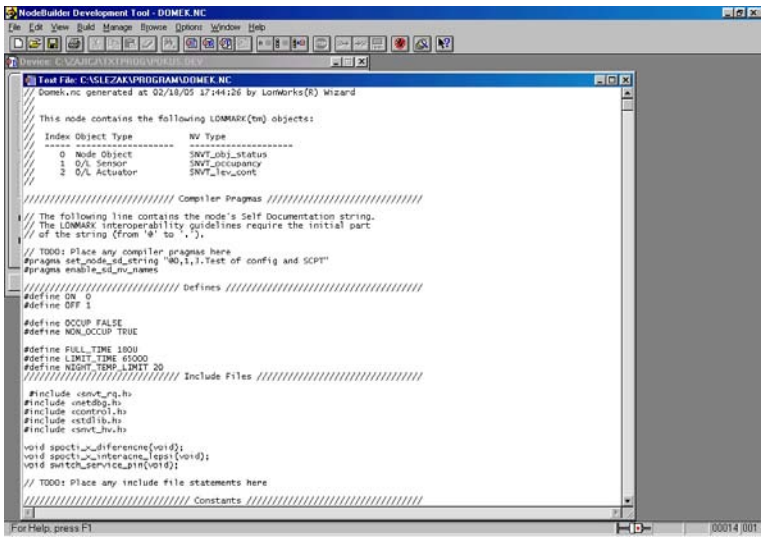


Obrázek 62: Nastavení vhodného template pro Neuron chip



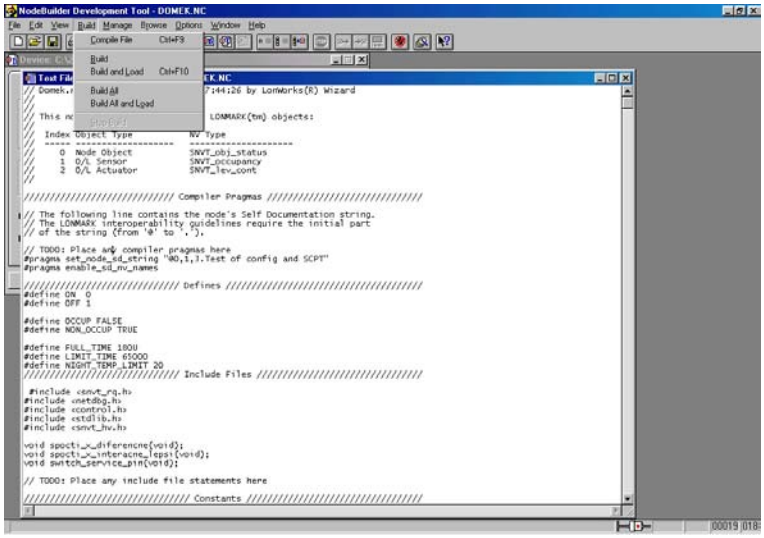
Obrázek 63: Výběr vhodné detekce zařízení

Po nastavení typu Neuron chipu, je nutné spustit *Edit* u položky aplikace. Zobrazí se samotné programovací prostředí. V tomto prostředí jsou již předpřipravené a komentované položky, kde je možné nadefinovat vstupní a výstupní proměnné, definici konstant atp. Je to jen vodíto a není povinností řídit se připraveným komentářem. Vygenerované komentáře pomáhají orientaci v kódu a struktuře programu.

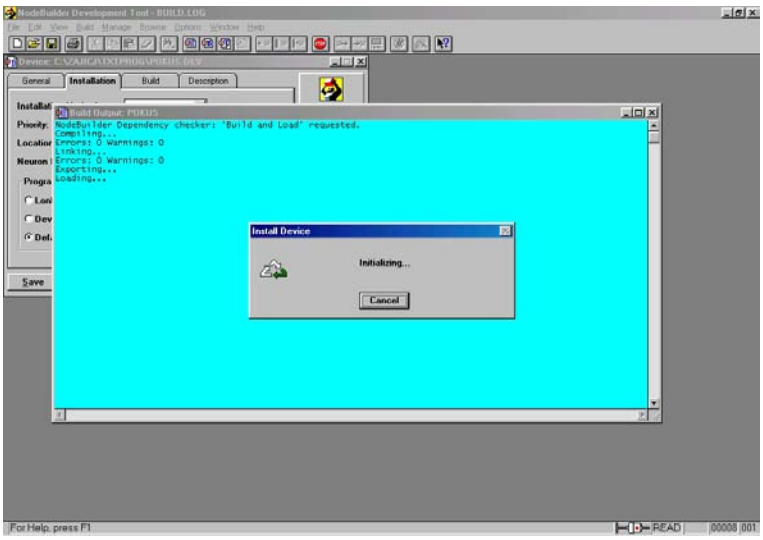


Obrázek 64: Programovací prostředí node builderu

Pokud je již program napsán, je možné v roletové nabídce zvolit *Build* → *Build All and Load*. Překladač přeloží program do příslušného obrazu pro vybraný Neuron chip a nahraje jej do jeho paměti.



Obrázek 65: Překlad programu

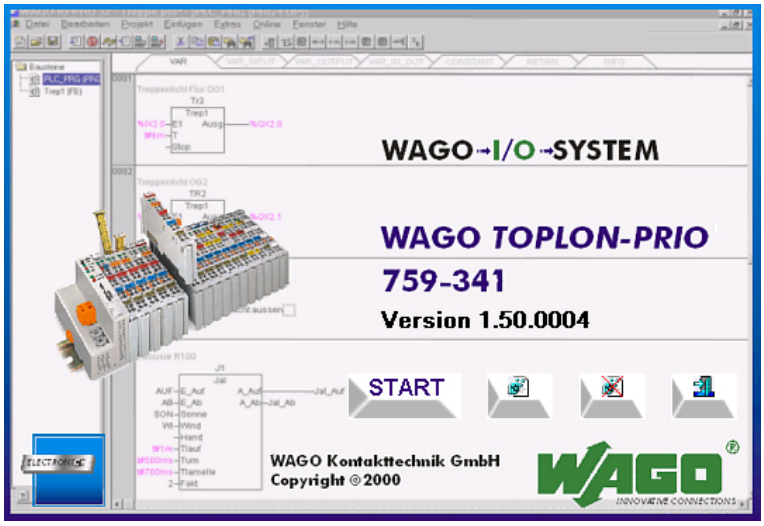


Obrázek 66: Překlad a nahrávání programu do Neuronu

Pokud nenastane chyba, je program úspěšně nahrán v Neuron chipu a začne se vykonávat jeho funkce.

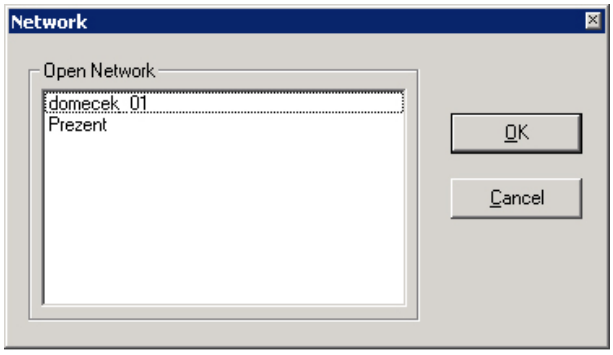
D.5 Wago Toplon-prio

Jak již název napovídá, slouží k celkové konfiguraci PLC Wago pro síť LonWorks. Konfigurační program Wago Toplon-prio je nainstalován jako zásuvný modul v programu Microsof Visio 2003 nebo jej lze spustit samostatně.

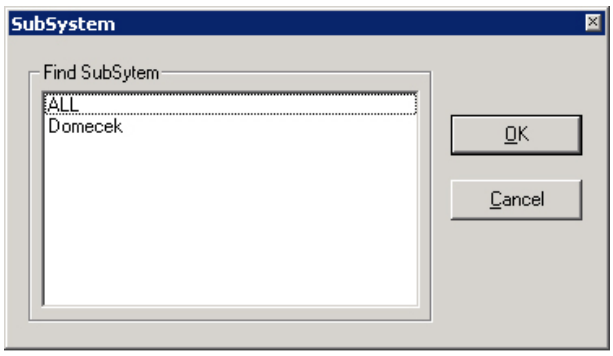


Obrázek 67: Aplikace Wago Toplon-prio

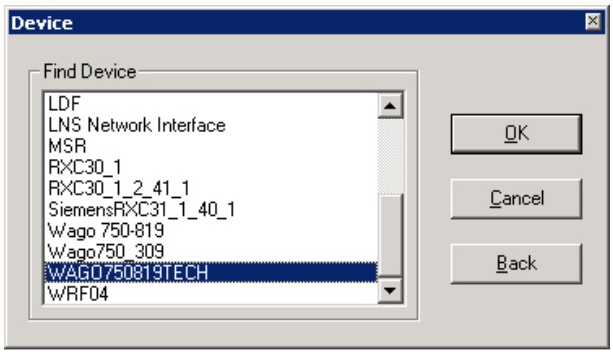
V prostředí Microsof Visio se spustí Toplon-prio pomocí *Configure* z lokální nabídky na konfigurovaném zařízení. Následující konfiguraci PLC Wago, lze rozdělit do dvou částí. Konfigurace hardwarové a softwarové části. Hardwarová část, je definice přídavných modulů umístěných na sběrnici PLC WAgo. Po spoštění Toplon-prio se vybere konfigurovaná síť, subsystém v této síti. Pokud není jisté v jakém subsystému je zařízení zakomponováno, vybere se *ALL*.



Obrázek 68: Výběr sítě pro konfiguraci Waga



Obrázek 69: Výběr subsystému v síti LonWorks

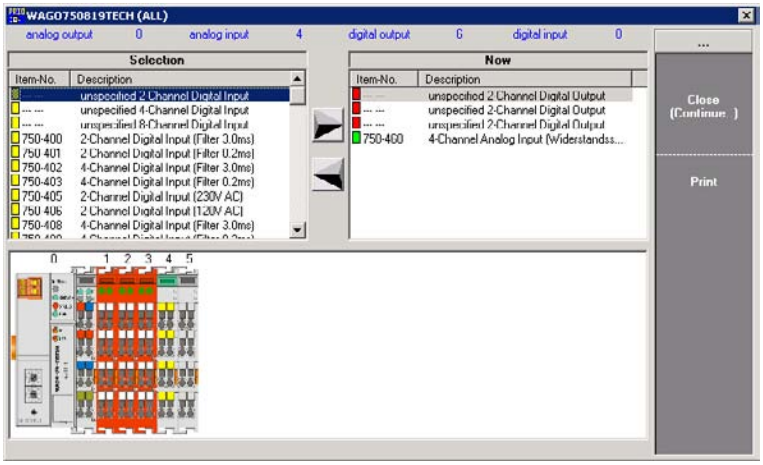


Obrázek 70: Výběr konfugurovaného Waga

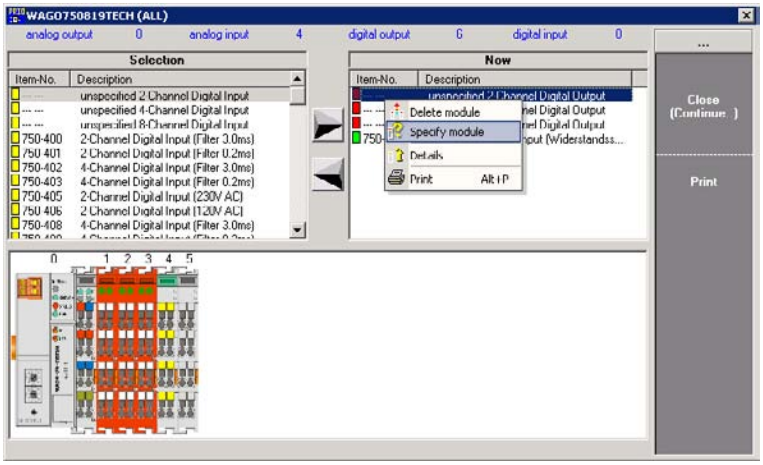
Pokud se jedná o první konfiguraci či došlo ke ztrátě informace o hardwarové konfiguraci PLC Wago, program vyzve ke specifikaci modulů.

Obrázek 71: Výzva ke specifikaci přídavných modulů PLC Wago

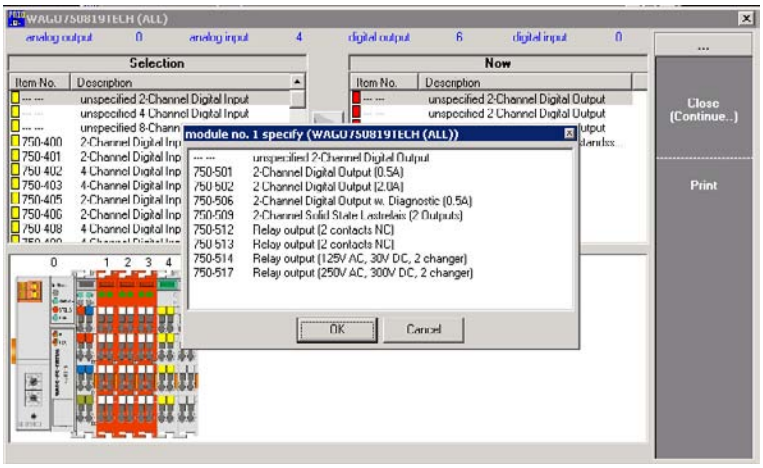
V konfiguračním okně lokální nabídky se určí, jaký modul je správný z nabízených modulů. Hardwarová konfigurace musí odpovídat fyzickému rozmístění modulů na sběrnici. Wago samo rozpozná o jaký typ modulu se jedná. Zda se jedná o vstupní či výstupní modul a zda jde o analogový či digitání modul.



Obrázek 72: Specifikace modulů

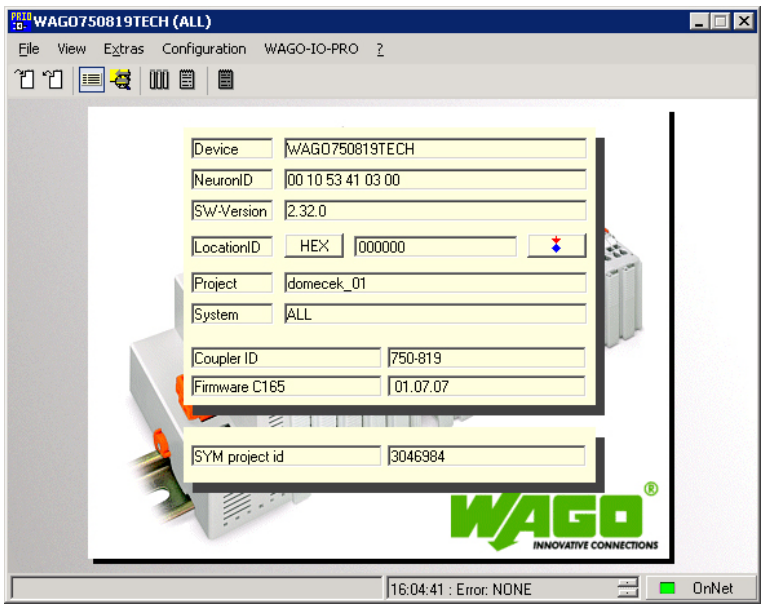


Obrázek 73: Výběr modulů



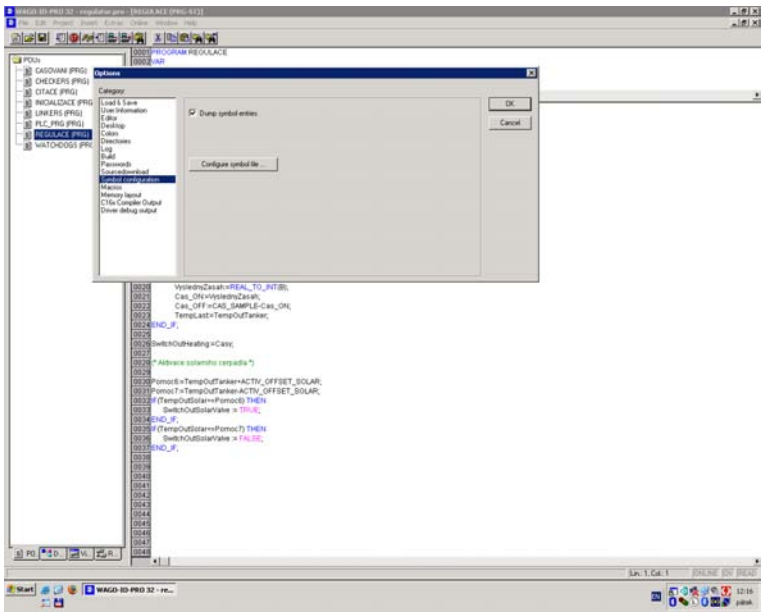
Obrázek 74: Nabídka možných modulů

Po specifikaci všech příslušných modulů se konfigurace potvrdí pomocí *Close* v pravé části aktuálného okna. Pokud bylo vše korektně nakonfigurováno, je hardwarová konfigurace ukončena a následuje softwarová konfigurace.



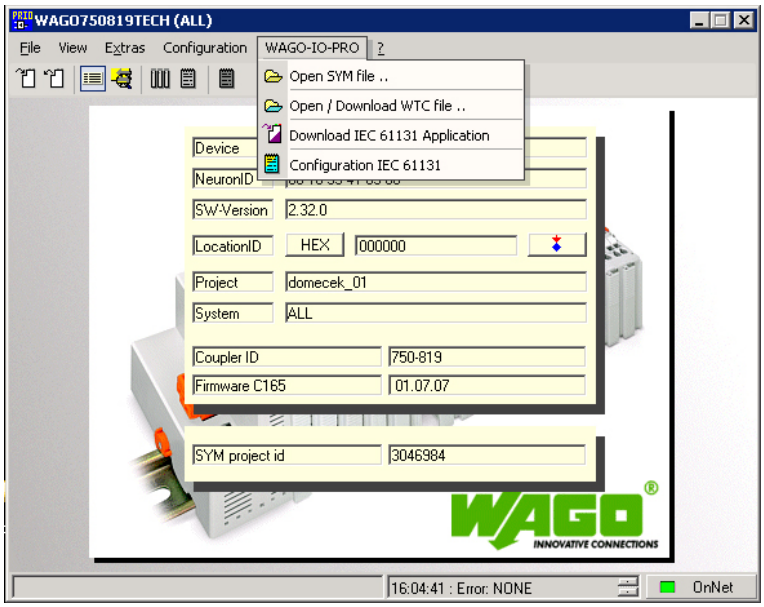
Obrázek 75: Konfigurační prostředí Wago Toplon-prio

Pro softwarovou konfiguraci je nutné vložení souboru s definicí proměnných z programu Waga a vstupně výstupních SNVT proměnných jak je uvedeno na obrázku [77]. Tento soubor má koncovkou *.SYM*. Pokud není tento soubor po překladu programu pro Wago vytvořen, je nutné nastavit v programu Wago-IO-Pro 32 či Codesys vytváření tohoto souboru. Pomocí nabídky *Project* → *Option..*, zaškrtnutím volby *Dump symbol entries* a následném překladu celého projektu se tento soubor vytvoří.



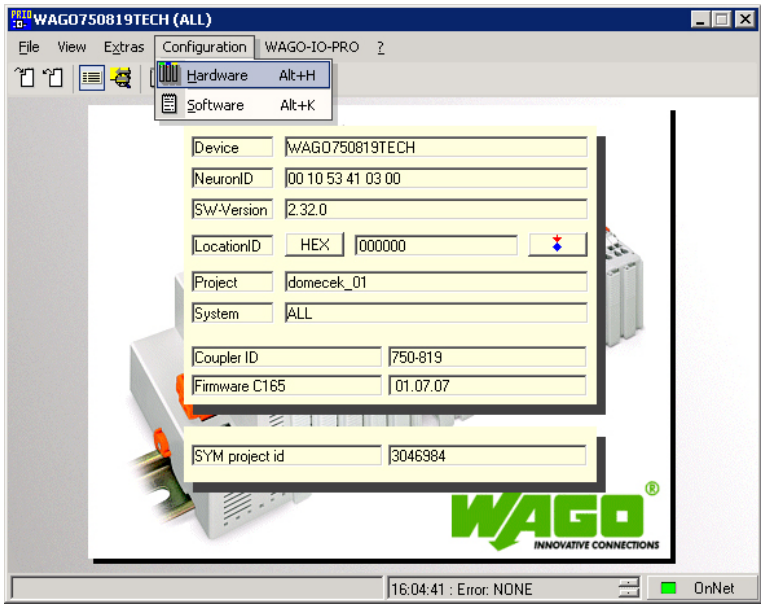
Obrázek 76: Vytvoření SYM souboru

Již vytvořený soubor se v Toplon-prio vloží pomocí nabídky *Open SYM file ...*



Obrázek 77: Import SYM souboru do Toplon-pria

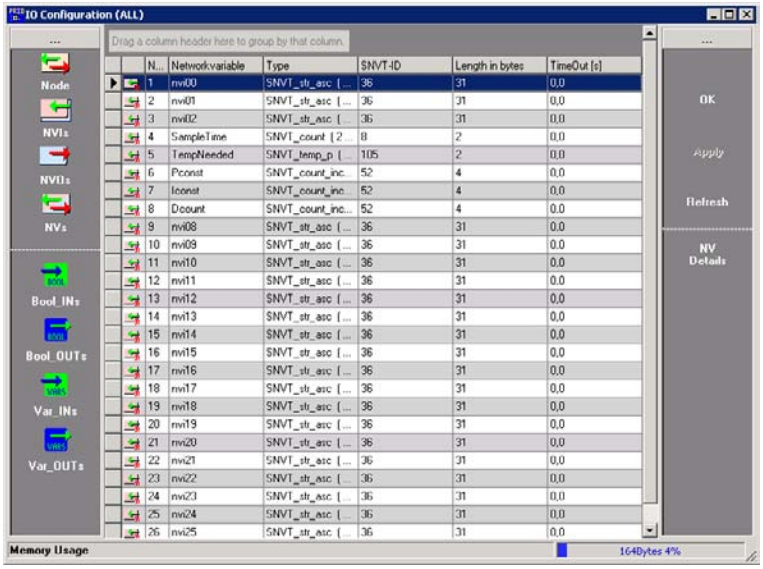
Po vložení souboru s definicí proměnných, následuje definice a spárování SNVT proměnných v PLC Wago. V nabídce *Configuration* → *Software* se vyvolá softwarové konfigurační prostředí.



Obrázek 78: Sofwarová konfigurace

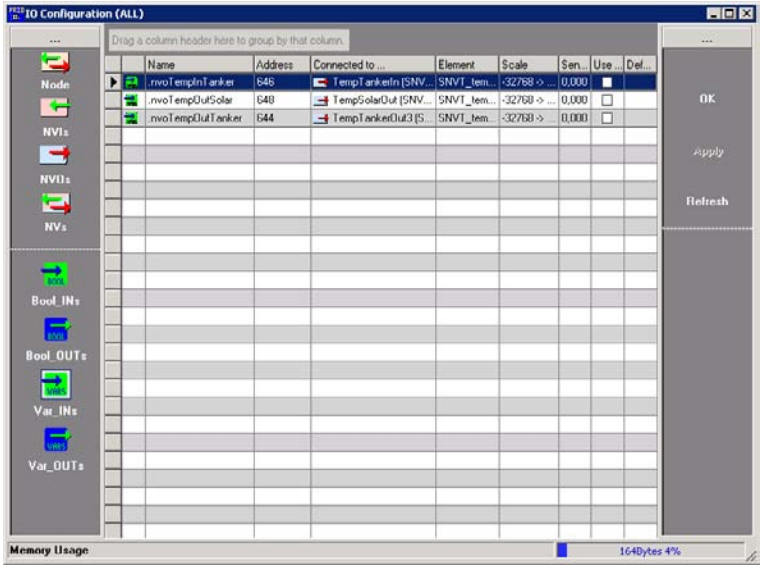
V levém horním oddílu NVIs a NVOs se pojmenují proměnné nvi a nvo tak, aby byla

v LonMakeru jednouchá a intuitivní jejich identifikace. Též se nastaví jakého typu je příslušná SNVT proměnná. Standardně jsou proměnné nastaveny na SNVT typ *SNVT str asc*.



Obrázek 79: Definování LonWorks proměnných

Následuje spárování nyní definovaných SNVT proměnných s proměnnými v programu. To se provede v levém dolním oddílu. Pro každou v programu nadefinovanou SNVT proměnnou se ve vysunovací nabídce přidělí definovaná proměnná v předešlém kroku. Dále se musí určit rozsahy proměnných. Pro typ *BOOL* je nejmenší datový typ byte, proto se určí jaká hodnota přísluší pro stav *OFF* a jaká pro hodnotu *ON*.



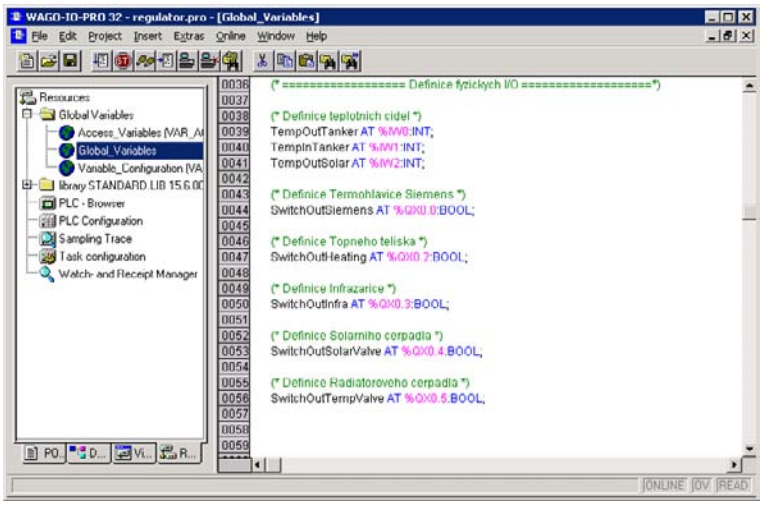
Obrázek 80: Spárování LonWorks proměnných s Wago proměnnými

Poté co se takto spárují všechny požadované proměnné, uloží se celá konfigurace opuštěním programu pomocí *File* → *Exit*. V dolní části okna je možné vidět proces ukládání konfigurace do PLC WAgo. Tímto je celá konfigurace hotova.

D.6 Wago-IO-Pro 32

Pro vytvoření programu pro Wago, je vhodné použít Wago-IO-Pro 32 s plnou licencí. Demo verze programu neobsahuje možnost vytváření *SYM* souboru, který je nutný pro konfiguraci. Další možností je použití programu Codesys. V tomto případě je nezbytné použít nejnovější verzi, která volbu tvorby genreování *SYM* souboru obsahuje. Dále bude popsán vývoj progamu v rámci Wago-IO-Pro 32.

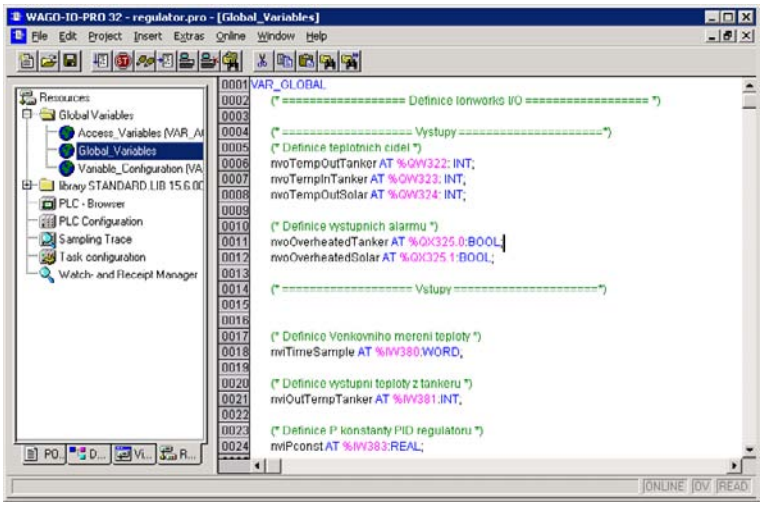
Pro tvorbu programu je nutné se rozhodnout jakým jazykem bude celý program psán. Mezi hlavní jazyky, z nabízených, patří LD[21] a ST[22]. Pro podobnost s klasickými programovacími jazyky byl vybrán Structured text. Celý program lze rozdělit do několika podprogramů uvnitř jednoho projektu. Výhoda toho konceptu je jeho přehlednost. Vhodné je zavést program na inicializace proměnných a nastavení programu či PLC do výchozího stavu. Důležitou části tvorby programu je správná adresace fyzický vstupů a výstupů. Též adresaci vstupů a výstupů pro LonWorks je potřeba správně nastavit. Při spuštění PLC Wago se adresují vstupy analogové, digitání, pak výstupy jak analogové tak digitální. Nezávisle na fyzické konfiguraci se adresují za sebou. Z toho plyne i adresace v programu. Pro předcházení kolizím je nutná znalost velikosti proměnné, kterou modul Waga poskytuje jako výstupní hodnotu či požaduje jako vstupní. Například sepnutí relé je hodnota jednoho bitu, ale teplota z čidla PT100 má velikost 1 byte. Záleží též na datovém typu proměnné. Zda se jedná o BIT, BYTE, WORD či DOUBLEWORD. Nerespektuje-li se velikost proměnné, může dojít k nechtěnému přepisu proměnných při jejich překryvu adres a tím i nefunkčnosti programu či nepředvídatelnému chování. Adresy od 0 do 255 typu word jsou určeny pro adresaci fyzických vstupů/vstupů. Jak vstupy tak výstupy mají svůj blok paměti začínající adresou 0 a končící na adrese 255. Pro síťové proměnné je určen blok adres od 256 do 511. Bližší a podrobný popis adresace je v dokementaci od PLC Wago [15] na straně 43 a 45. Obrázek [81] názorně ukazuje adresaci fyzických vstupů teplot ze třech čidel PT100 a pěti výstupů bitovového typu.



Obrázek 81: Adresace fyzických vstupů a výstupů

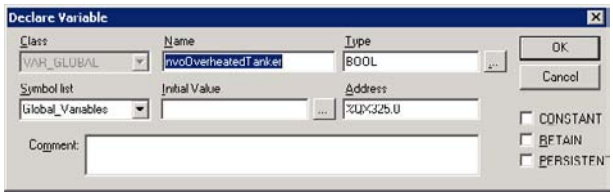
[21] Ladder diagram
[22] Structured text

Při adresaci síťových proměnných je volba adresy zcela na vůli programátora. Jen musí být v rozsahu 256 až 511. Obrázek [82] ukazuje adresaci několika síťových vstupních a výstupních proměnných.



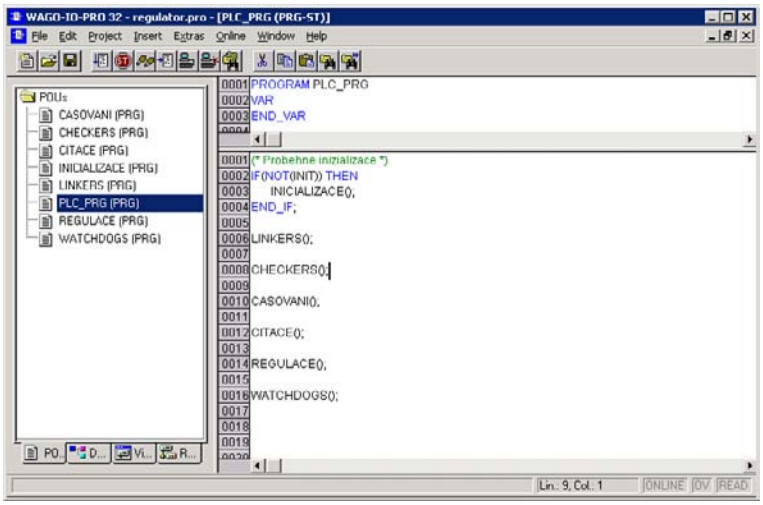
Obrázek 82: Adresace síťových vstupů a výstupů

Pokud jsou naadresovány všechny fyzické a síťové vstupy/výstupy, je možné definovat další proměnné. *Wago IO Pro 32* umožňuje velmi jednoduché zakládání dalších proměnných. Pokud během zápisu program proměnná ještě neexistuje, *Wago IO Pro 32* zobrazí dialogové okno pro její založení viz obrázek [83]. Je-li nutná její přístupnost v celém programu, musí se umístit do globálních proměnných.



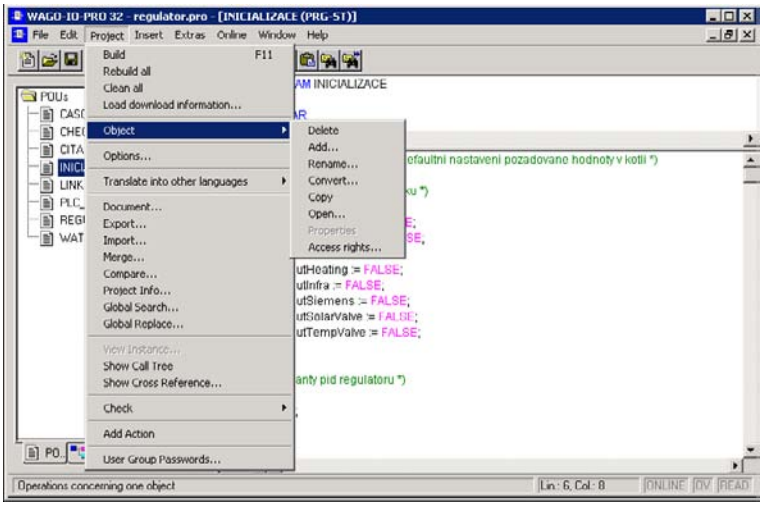
Obrázek 83: Založení nové proměnné

Vytvořený program je rozdělen na podprogramy, které umožňují vytvářet strukturu programu a tím zpřehlednit funkci jednotlivých podprogramů.



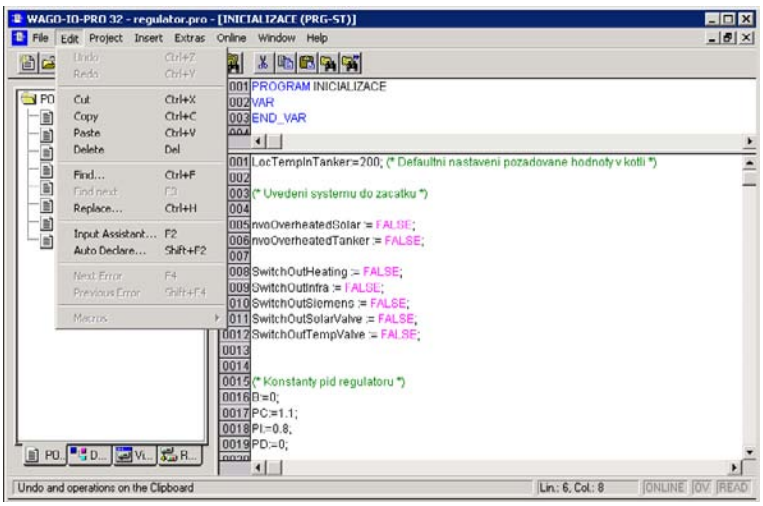
Obrázek 84: Hlavní program

Pro vložení nového podprogramu lze využít nabídky *Project* → *Object* → *Add...*. V této nabídce je umístěna kompilace celého projektu či *Option* pro nastavení prostředí Wago-IO-Pro 32.

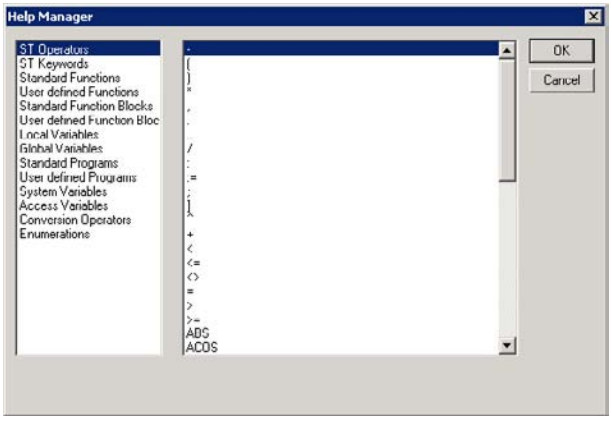


Obrázek 85: Vložení nového programu

Pro vložení proměnné do kódu programu, lze využít *Input Assistant* pro vkládání proměnných, deklarací či funkcí viz obrázek [86]. Docílí se rychlejšího psaní programového kódu.

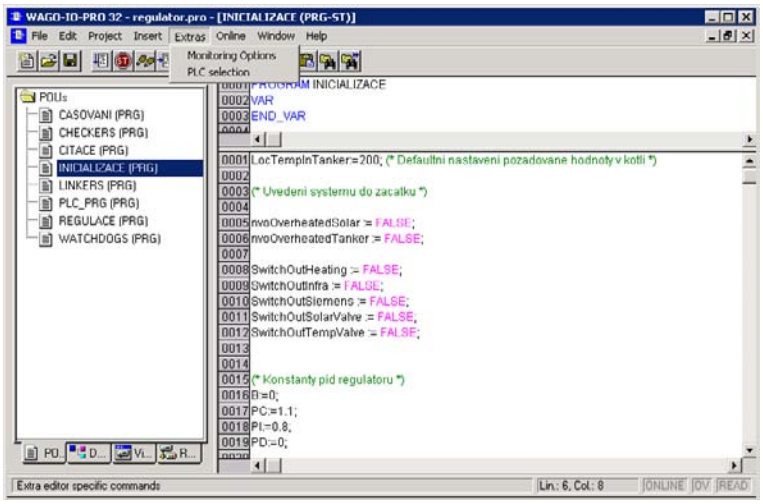


Obrázek 86: Input assistant

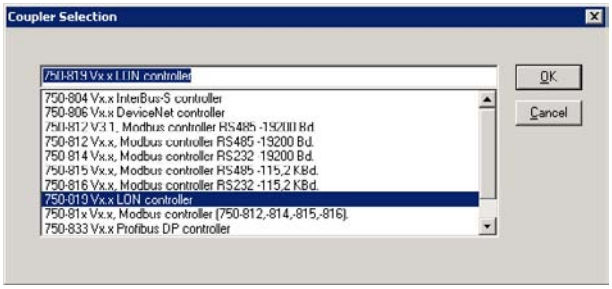


Obrázek 87: Vložení proměné pomocí asistenta

Pokud je již program vytvořen, je nutné ho přeložit. Celý překlad se provádí pro konkrétní typ PLC Wago. Pokud byl při zakládání projektu zvolen nesprávný typ, je možné ho v nabídce změnit *Extras* → *PLC selection*.

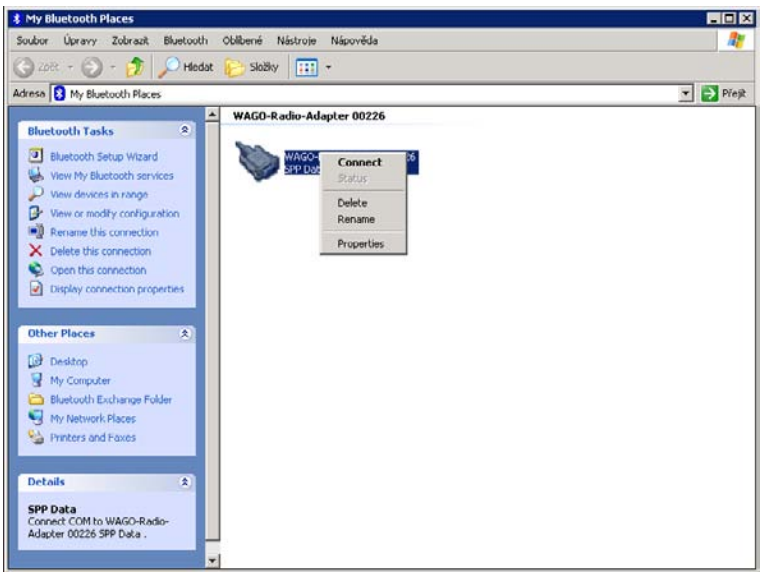


Obrázek 88: Změna PLC v projektu

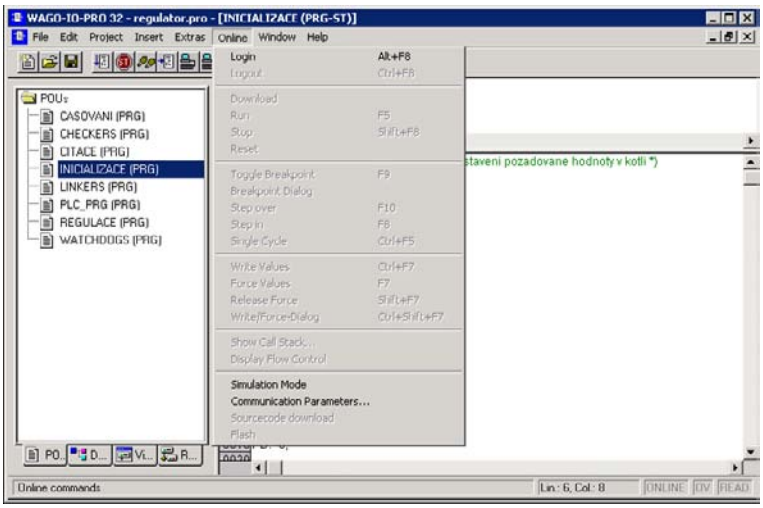


Obrázek 89: Výběr vhodného PLC

Po správném nastavení typu PLC Wago a přeložení programu, je možné se připojit k PLC Wago a nahrát program do jeho paměti RAM. Pro přenos dat se používá seriová linka. Lze využít seriového kabelu či bluetooth. Pokud se používá bluetooth pro vytvoření seriového spojení, je nutné toto prvně spojení navázat. Pokud bluetooth nedetekuje kratší dobu komunikaci, dojde k jeho odpojení. Proto je nutné velmi rychlé přepnoutí po navázání spojení s PLC Wago do *Wago IO Pro 32* a přihlásit se pomocí nabídky *Online* → *Login*.

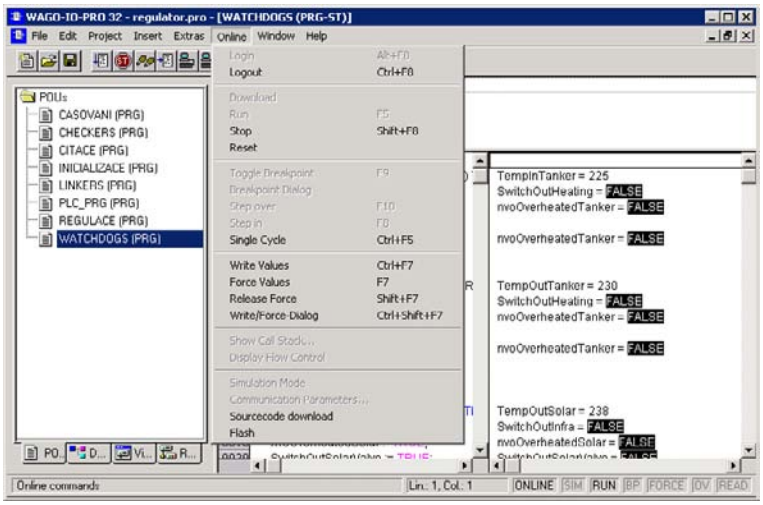


Obrázek 90: Navázání seriového spojení pomocí bluetooth s PLC Wago



Obrázek 91: Přihlášení do PLC

Při úspěšném přihlášení a nahrání programu, je potřebné PLC přepnout do režimu *RUN*. Posléze dojde k samotnému výkonu programu. Při běhu programu je možné v pravé části programovacího prostředí vidět stavy a hodnoty proměnných. Pokud je již program plně funkční, je nezbytné jej zapsat do paměti *FLASH*. Wago po resetu či připojení do napájecí sítě načte program z paměti *FLASH* a dojde k automatickému spuštění programu bez nutnosti opětovného nahrávání programu do PLC. Při zápisuje program do paměti *FLASH*, je nutné být přihlášen v PLC Wago. Poté se v nabídce vybere příkaz *Online* → *Flash*, který je umístěn poslední z nabídky. Nyní je program trvale nahrán v paměti PLC.



Obrázek 92: Umístění programu do paměti flash

Následně se PLC Wago nakonfiguruje podle kapitoly D.5 a zapojí do sítě LonWorks podle přílohy D.1.

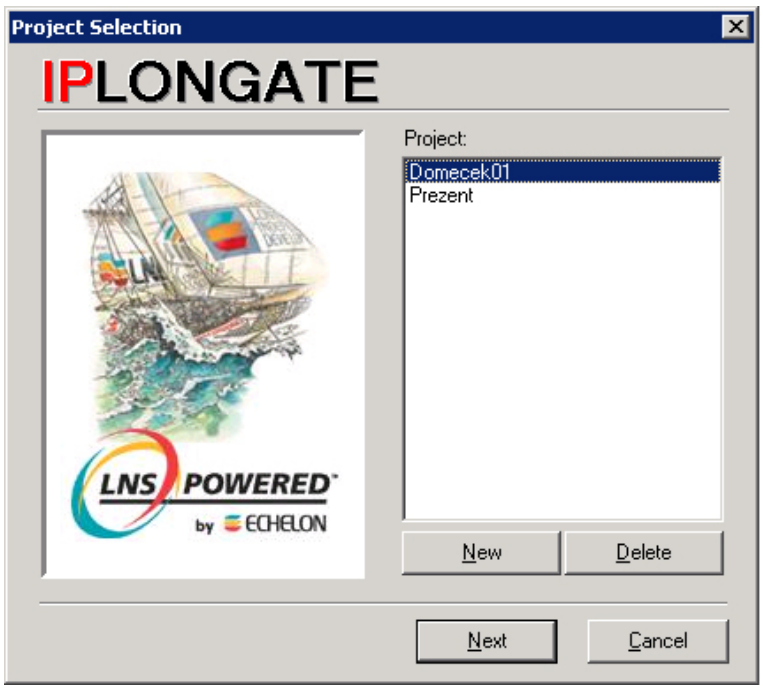
D.7 IPlongate

IPlongate je OPC[23] server sloužící k propojení různých aplikací s LNS databází. Je velmi důležitý pro funkčnost konfiguračního programu LonMaker. Při konfiguraci sítě Lonworks, je nutné prvně spustit Iplongate a následně LonMaker. Důležité je dodržení tohoto sledu spouštění aplikací. Při spouštění IPlongatu, se v nabídce vybere položka *Local*. Značíto to načítání z lokální LNS databáze.

Obrázek 93: Spouštění Iplongate

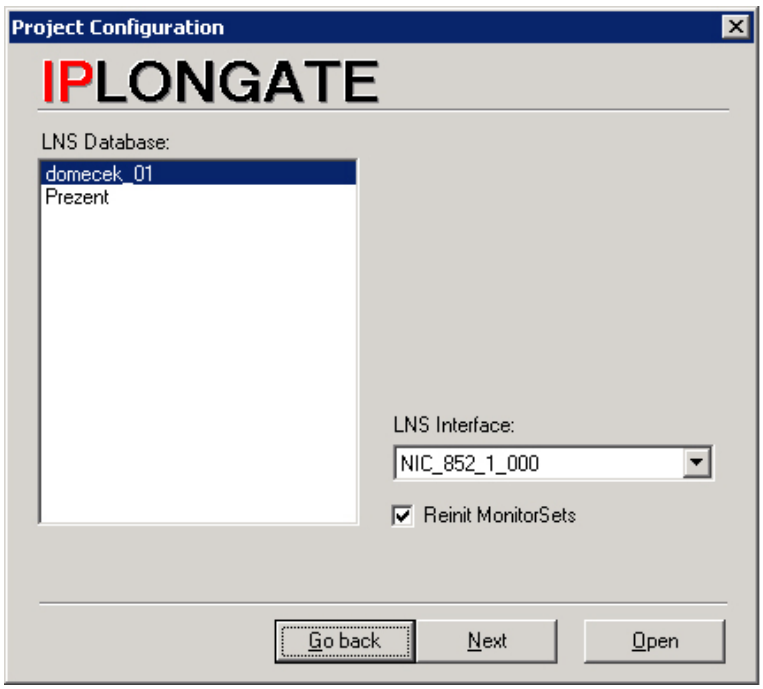
Následuje výběr projektu.

[23] Object Linking and Embedding for Process Control je soubor specifikací definujících způsob předávání informací mezi jednotlivými OPC aplikacemi



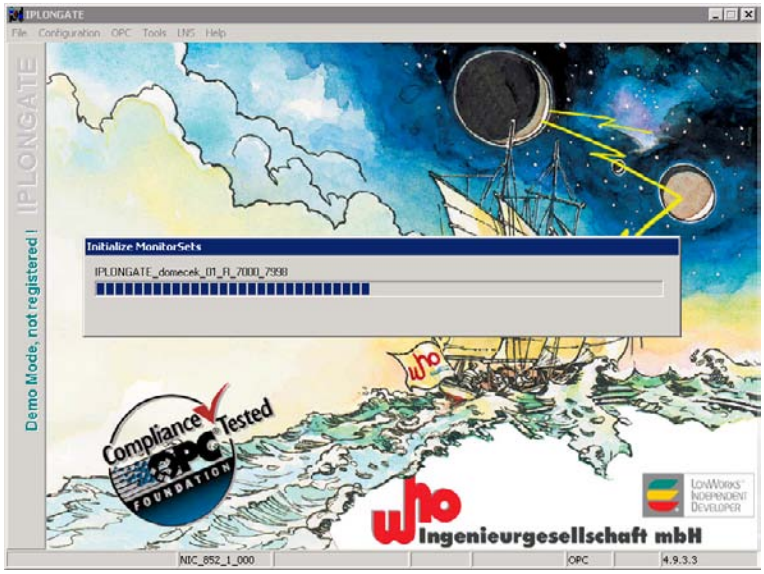
Obrázek 94: Výběr projektu

Další dialogové okno nabízí výběr LNS databáze, skrze jaké rozhraní se bude přistupovat k síti LonWorks. V tomto případě se jednalo o rozhraní L-switch firmy Loytec. Standardně se vybírá se toto rozhraní. Pokud je nainstalován driver pro XLON Dongle, lze přistupovat do sítě i přes toto rozhraní.



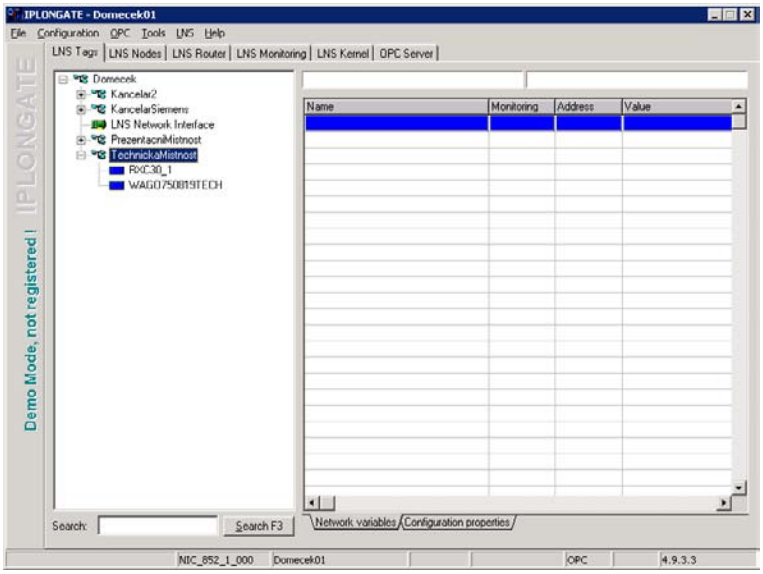
Obrázek 95: Výběr databáze a rozhraní

IPlongate začne načítat LNS databázi včetně všech proměnných a vytvoří z nich strukturovaný strom sítě.



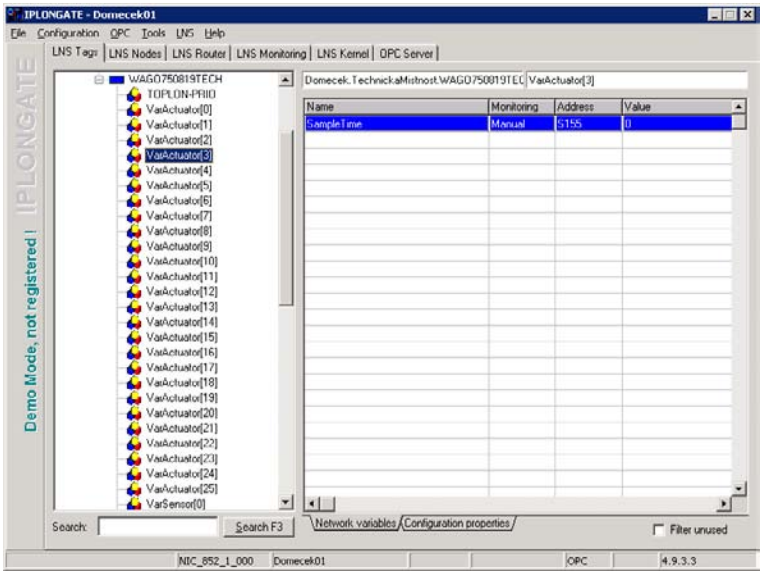
Obrázek 96: Načítání LNS databaze

Po načtení celé databáze, je v levé části umístěn vytvořený strukturovaný strom a lze v něm číst a zapisovat do proměnných.



Obrázek 97: Vytvořený strukturovaný strom LNS databáze

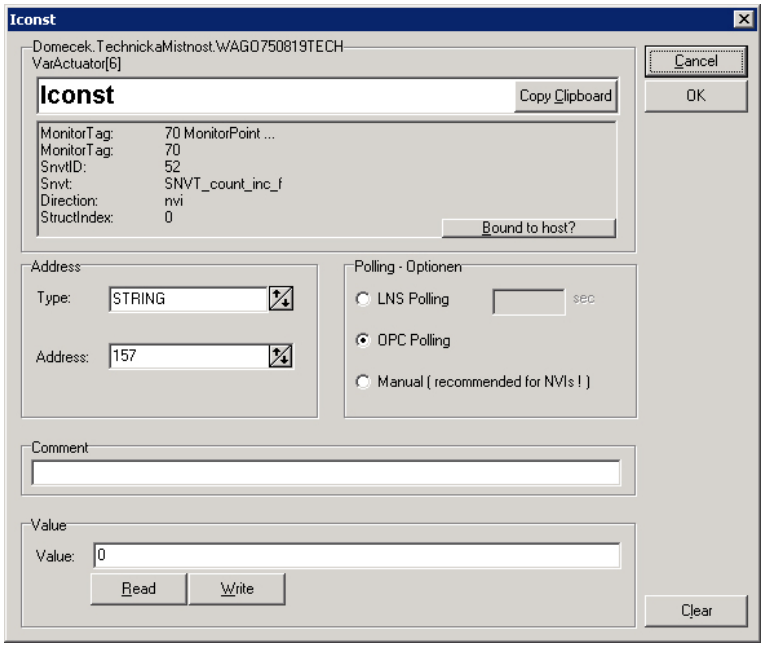
Vlevo na obrázku 98 je vidět rozvinutý strom pro všechny prvky v síti a též PLC Wago umístěné v technické místnosti. Pro Wago platí pravidlo, že výstupní proměnné mají název VarSensor[xx]. Vstupní proměnné jsou nazvány VarActuator[xx]. Po kliknutí na proměnnou v PLC Wago, lze zjistit její název, adresu a její aktuální hodnotu.



Obrázek 98: Výběr proměnné ve Wagu

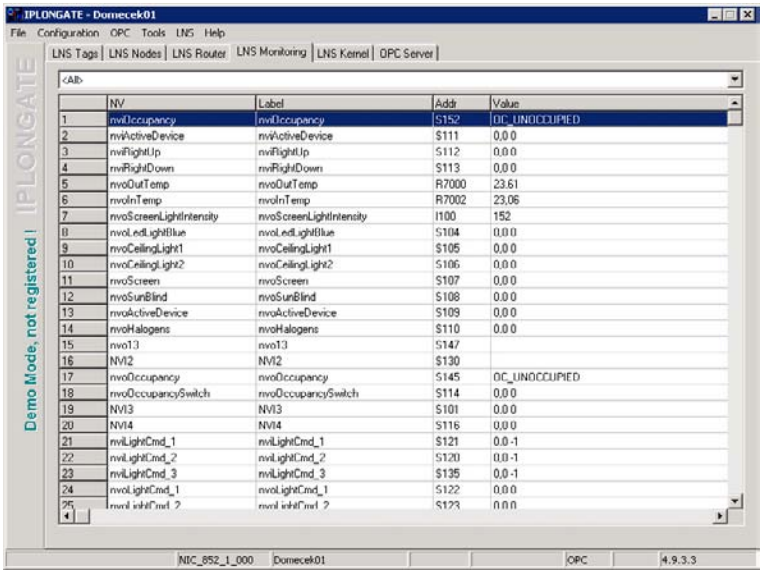
Pokud není uvedena adresa, druh nahrávání proměnné (polling) a hodnota, lze na proměnnou vpravo kliknout. Zobrazí se konfigurační okno dané proměnné. Lze si načíst či vložit hodnotu do proměnné. Je zde možnost nastavení opakovaného načítání neboli polling. Nabízí se nastavení nahrávání pomocí LNS serveru, až po žádosti OPC serveru či

manuální. Pokud se zvolí LNS nahrávání, musí se nastavit čas jak často se má proměnná nahráva z LNS serveru resp databáze. Pokud zvolíme OPC polling, tak se proměnná nahraje pouze, když je žádána od OPC serveru. Pokud chceme s danou proměnnou pracovat v programu Axeda Wizcon, musí se především u vstupních proměnných resp. VarActuators nastavit nahrávání na hodnotu *OPC Polling*. Jinak je vizualizační program nenalezne v databázi tagů. Manuální nahrání je jen při manuálním vyžádání, které se hodí u vstupních proměnných. Další využití Iplongate nachází v indetifikaci typu SNVT a adresy proměnné.



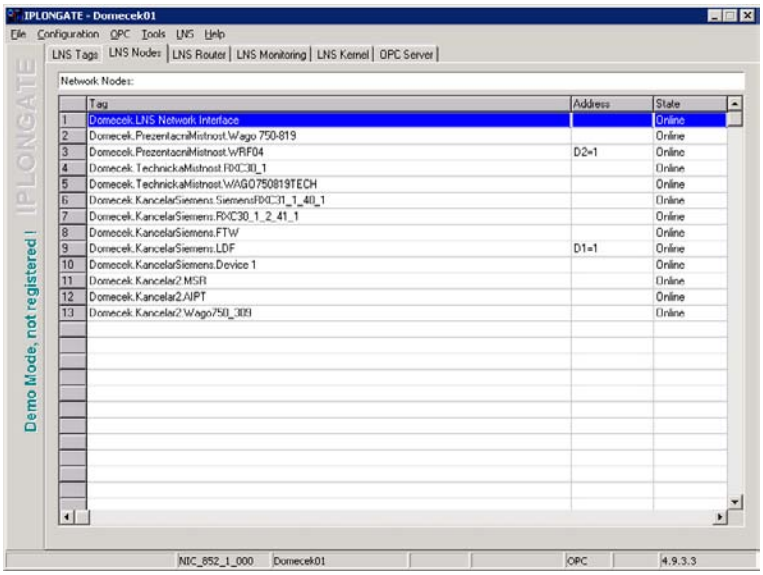
Obrázek 99: Konfigurace proměnné

Užitečnou vlastností k čemu využívat Iplongate je k identifikaci proměnných ve Wagu či jiném nodu v síti LonWorks. Pokud programátor nepojmenuje SNVT proměnné intuitivně, neni jednoznačné jak se potřebná proměnná jmenuje. Bohužel, někdy je dodaná dokumentace špatná a nelze z ní jednoznačně určit jaká proměnná je žádaná. Velmi často se proměnné jmenují nvi0, .. nvi25, z toho nelze přesně říci o jakou proměnnou jde. Pomocí pozorování změn hodnot proměnných v Iplongatu, lze zjisti o jakou proměnnou se jedná z jejího rozsahu a fyzikálního typu. Další vlastností je monitorování proměnných, kdy server poskytuje celý seznam proměnných s jejich aktuální hodnotou.



Obrázek 100: Seznam monitorovaných proměnných

Iplongate poskytuje seznam všech zařízení v databází resp. v dané síti LonWorks včetně jejich aktuálního stavu.



Obrázek 101: Seznam monitorovaných zařízení

E Obsah přiloženého DVD

Přiložené DVD obsahuje tyto adresáře a soubory:

- **/dp_2006_michal_slezak.pdf** - tento dokument ve verzi .pdf

- /**datasheets** - adresář obsahující všechny použité manuály
- /**pdf** - adresář se zdrojovými kódy této diplomové práce
- /**pictures** - adresář s obrázky použité v diplomové práci
- /**plans** - adresář s veškerými plány
- /**prezentation** - adresář obsahující všechny vytvořené prezentace
- /**software** - adresář obsahuje užitečný software potřebný k vývoji aplikací