Komplexní řešení inteligentního domu na bázi procesoru Intel 8051

Complex Smart House solution based on Intel 8051 processor

Bc. Martin Žeravík

Diplomová práce
2007

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
Ústav aplikované informatiky
akademický rok: 2006/2007

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Martin ŽERAVÍK**
Studijní program:	**N 3902 Inženýrská informatika**
Studijní obor:	**Informační technologie**
Téma práce:	**Komplexní řešení inteligentního domu na bázi procesoru Intel 8051**

Zásady pro vypracování:

Práce řeší formou projektu problematiku návrhu inteligentní budovy včetně implementace bezpečnostního řešení. Součástí práce je také propojení vnitřních autonomních informačních technologií s bezpečným řešením umožňujícím řízení a monitoring procesů po internetu.

1. Analyzujte informační zdroje řešící problematiku inteligentních budov
2. Analyzujte a stanovte možnosti zvoleného procesoru Intel pro řízení budovy.
3. Navrhněte projekt řešení inteligentního domu na bázi procesoru Intel 8051
4. Vyhodnoťte úspěšnost řešení a stanovte závěry z pohledu pozitiv a negativ zvoleného řešení.

Rozsah práce:

Rozsah příloh:

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

Seznam odborné literatury:

[1] VALEŠ, M. Inteligentní dům. Brno: ERA, 2006. 136 s. ISBN 80-7366-062-8.
[2] PECHAL, S. Monolitické mikropočítače. Praha: BEN technická literatura, 1998, 272 s. ISBN 80-86056-30-9
[3] SKALICKÝ, P. Mikroprocesory řady 8051. 2. rozšířené vydání. Praha: BEN technická literatura, 1998. 160 s. ISBN 80-86056-39-2
[4] FRONC, V. Mikrokontroléry ATMEL s jádrem 8051. Praha: BEN technická literatura, 2002, 200 s. ISBN 80-7300-008-3
[5] VLACH, J., VLACHOVÁ, V. Počítačová rozhraní. Praha: BEN technická literatura, 2002, 176 s. ISBN 80-7300-010-5
[6] MACDONALD, M., SZPUSZTA, M. ASP.NET 2.0 a C++. Brno: Zoner Press, 2006, ISBN 80-86815-38-2
[7] TZB-info: Portál pro technická zařízení budov [online] <http://www.tzb-info.cz>
[8] Vše z oboru elektroniky a automatizace -- HW server [online] <http://www.hw.cz>

Vedoucí diplomové práce: **Mgr. Roman Jašek, Ph.D.**
Ústav informatiky a statistiky

Datum zadání diplomové práce: **13. února 2007**

Termín odevzdání diplomové práce: **28. května 2007**

Ve Zlíně dne 13. února 2007

prof. Ing. Vladimír Vašek, CSc.
děkan

doc. Ing. Ivan Zelinka, Ph.D.
ředitel ústavu

ABSTRAKT

Tato diplomová práce je zaměřena na problematiku návrhu komplexního řešení inteligentního domu na bázi procesoru Intel 8051. Práce popisuje základní principy činnosti systému pro inteligentní dům a hardwarové řešení zajišťující jeho provoz. Dále je zde popsáno softwarové řešení, které slouží pro veškerou správu inteligentního domu a jeho komunikaci s vnějším okolím. Činnost tohoto systému je demonstrována na modelovém domu navrženém speciálně pro tento účel.

Klíčová slova: Inteligentní dům, Intel 8051

ABSTRACT

This graduation thesis is focused on problems of design of complex smart house solution based on Intel 8051 processor. Thesis describes fundamental principles of operation of system for smart house and hardware solution securing his operation. Also here is described software solution which performs all smart house management and his communication with outside environment. Operation of this system is demonstrated on model house designed specially for this purpose.

Keywords: Smart House, Intel 8051

Na tomto místě chci poděkovat vedoucímu mé diplomové práce Doc. Mgr. Romanu Jaškovi, Ph.D. za zájem, podporu při řešení problémů, připomínky a čas, který věnoval mé práci.

Prohlašuji, že jsem na diplomové práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků, je-li to uvolněno na základě licenční smlouvy, budu uveden jako spoluautor.

Ve Zlíně …….

Podpis diplomanta

OBSAH

ÚVOD **8**
I TEORETICKÁ ČÁST **10**
1 INTELIGENTNÍ DŮM **11**
1.1 STÁVAJÍCÍ SYSTÉMY PRO INTELIGENTNÍ BUDOVY 12
2 MOŽNOSTI PROCESORU INTEL 8051 ..... **13**
2.1 ARCHITEKTURA PROCESORU 8051 13
2.2 ORGANIZACE PAMĚTI 14
2.3 KOMUNIKAČNÍ SCHOPNOSTI 16
2.4 VHODNÉ VARIANTY PROCESORU 8051 17
3 NÁVRH ŘEŠENÍ INTELIGENTNÍHO DOMU **20**
3.1 NÁVRH TECHNICKÉHO ŘEŠENÍ 20
3.1.1 Řídicí jednotky 20
3.1.2 Komunikace 20
3.1.3 Počítač Master PC 21
3.2 ZABEZPEČENÍ SÍTĚ 22
3.2.1 Firewall 22
3.2.2 Typy firewallů 23
3.2.3 Zabezpečení bezdrátové sítě 24
3.2.4 Příklady sítí 25
4 SOFTWAROVÁ PLATFORMA **27**
4.1 .NET FRAMEWORK 27
4.2 PROGRAMOVACÍ JAZYK C# 29
4.3 ASP.NET 30
4.4 SQL 30
4.4.1 Microsoft SQL Server 32
II PRAKTICKÁ ČÁST **33**
5 NÁVRH MODELOVÉHO DOMU **34**
6 HARDWAROVÉ ŘEŠENÍ **35**
6.1 ŘÍDICÍ JEDNOTKY 35
6.2 ZAPOJENÍ ŘÍDICÍCH JEDNOTEK 36
6.3 KOMUNIKACE JEDNOTEK 38
6.4 POČÍTAČ MASTER PC 39
7 SOFTWAROVÉ ŘEŠENÍ **41**

7.1 LOGICKÁ STRUKTURA INFORMAČNÍHO SYSTÉMU 41
7.2 PROGRAM ŘÍDICÍCH JEDNOTEK 43
7.3 DATABÁZE 44
7.3.1 Struktura databáze 44
7.3.2 Uložené procedury 45
7.4 SDÍLENÉ KOMPONENTY 50
7.4.1 Třídy 51
7.4.2 DAO 53
7.4.3 Com 54
7.4.4 Hardware 55
7.5 ŘÍDICÍ APLIKACE 56
7.5.1 Struktura aplikace 56
7.5.2 Sériová komunikace 60
7.5.3 Síťová komunikace 61
7.6 INTERNETOVÁ APLIKACE 61
7.6.1 Design webové aplikace 63
7.6.2 Rozmístění souborů webové aplikace 64
7.6.3 Zabezpečení 64
7.7 KOMUNIKACE MEZI APLIKACEMI 66
ZÁVĚR 69
CONCLUSION 70
SEZNAM POUŽITÉ LITERATURY 71
SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK 72
SEZNAM OBRÁZKŮ 74
SEZNAM TABULEK 76
SEZNAM PŘÍLOH 77

ÚVOD

Pojem *inteligentní dům* je v současné době stále populárnější a bývá zmiňován v mnoha článcích a diskuzích. Co je vlastně podstatou takového inteligentního domu a co musí umět, aby jej bylo možné za inteligentní označit? Neexistuje žádná jednotná definice, co musí dům splňovat, aby jej bylo možné považovat za inteligentní. U inteligentního domu je ale podstatné, že všechny systémy spolupracují dohromady. V běžných domech máme zvlášť systém pro světla, druhý na topení, třetí na rolety a rovněž ovládání audio a videotechniky je oddělené. Důležité je propojit vše pomocí nějaké programovatelné logiky. Inteligence domu spočívá v tom, že se umí sám rozhodovat. Samozřejmě pouze v rutinních opakujících se činnostech, které se mu rozhodneme svěřit. Například v místnosti může být instalované čidlo detekující přítomnost osob. Když v místnosti delší dobu nikdo není, systém ztlumí topení, zhasne světla, vypne televizi. Smysl inteligentního domu je ulehčit lidem život a přitom se nemuset o techniku starat a myslet na ni. A především usnadnit její ovládání.

Jedním z mnoha způsobů řešení takového inteligentního domu se zabývá tato diplomová práce. Jak již se samotného názvu diplomové práce vyplývá, bude se jednat o systém řízení inteligentního domu založený na mikroprocesoru s jádrem 8051 firmy Intel. V teoretické části práce jsou popsány vlastnosti mikroprocesoru 8051, jeho architektura a jeho komunikační schopnosti. Dále jsou zde popsány konkrétní varianty mikroprocesorů s jádrem 8051 včetně jejich technických specifikací, které jsou vhodné pro účely tohoto projektu. Důležitou kapitolou teoretické části je pak samotný návrh řešení inteligentního domu, kde jsou popsány principy činnosti řízení domu založeného na speciálních řídicích jednotkách osazených mikroprocesorem s jádrem 8051, jejich komunikace a komunikace s nadřazeným počítačovým systémem zajišťujícím archivaci dat a přístup k domu přes internet. V souvislosti s tím jsou zde probrány možnosti zabezpečení počítačové sítě především proti útoku z internetu. Nakonec je zde popsána softwarová platforma Microsoft .NET, na které poběží celý informační systém určený pro komunikaci s uživatelem.

V praktické části této práce je nejprve popsán modelový dům, na kterém je celý systém inteligentního domu demonstrován. Poté je zde podrobně popsáno řešení řídicích jednotek, jejich schémata zapojení a způsob komunikace včetně popisu komunikačního protokolu navrženého speciálně pro tento účel. Rovněž je zde uveden popis počítače typu PC, který je vhodné použít jako nadřazený počítačový systém určený pro archivaci dat, ovládání

celého domu a komunikaci s uživatelem prostřednictvím místní počítačové sítě nebo sítě internet. Poslední kapitola je pak věnována samotnému softwarovému řešení. Je zde popsán nejen samotný program pro mikroprocesory řídicích jednotek, ale také struktura celého informačního systému běžícího na nadřazeném počítači. Jedná se o popis struktury databáze, která souží k definici závislostí mezi jednotlivými zařízeními domu a archivaci jejich měřených hodnot. Dále je zde popsána aplikace sloužící ke komunikace se samotnými řídicími jednotkami a ke správě celého systému. Poslední aplikace slouží k zobrazování a nastavování stavu jednotlivých zařízení prostřednictvím internetu nebo místní počítačové sítě. Nakonec je zde uveden způsob, jakým mezi sebou jednotlivé aplikace vzájemně komunikují.

Na médiu přiloženém k této diplomové práci (P VII) jsou uloženy veškeré zdrojové kódy zde popsaných aplikací včetně komentářů popisujících jejich činnost.

I. TEORETICKÁ ČÁST

1 INTELIGENTNÍ DŮM

Inteligentní dům v nejširším možném smyslu slova je budova vybavená počítačovou a komunikační technikou, která předvídá a reaguje na potřeby obyvatel s cílem zvýšit jejich komfort, pohodlí, snížit spotřebu energií, poskytnout bezpečí a zábavu pomocí řízení všech technologií v domě a jejich interakcí s vnějším světem. Takto široké pojetí je zatím ještě vzdálené nynější realitě, nicméně velmi rychlý růst technických možností v posledním desetiletí nás k této vizi neustále přibližuje. Pojem „inteligentní dům“ se v současnosti využívá velmi volně – od domu, který má např. pouze běžný bezpečnostní kamerový systém a strukturované kabelové rozvody pro počítačovou síť, až po ukázkové domy budoucnosti, které slouží jako výzkumné laboratoře pro vývoj a testování nejnovější techniky.

Myšlenka takového bydlení není nikterak nová: koncept automatizovaného domu, který sám řídí topení, má roboty na čištění podlah a audio/video systémy v každé místnosti, existuje již od padesátých let minulého století.

Inteligentní dům do budoucna slibuje naprostou změnu způsobu, jakým dnes žijí lidé s technikou a používají technologie v domácnosti.

Hlavním cílem inteligentního domu je usnadnit a zpříjemnit bydlení. Inteligentní dům dokáže všechnu techniku v domě propojit tak, aby dokázala navzájem spolupracovat. Zároveň sjednotí její ovládání, a to jak z hlediska vzhledu vypínačů a displejů na zdi, tak především poskytne jednotný způsob ovládání, přizpůsobený na míru pro konkrétní dům a jeho obyvatele.

Díky propojení všech systémů do jednoho společně řiditelného celku a možnosti libovolně programovat funkci každého vypínače lze oproti běžnému domu zcela změnit způsob ovládání. Na rozdíl od klasického manuálního ovládání jednotlivých světel, rolet, topení či hudby můžeme vytvořit tzv. scény neboli režimy – ať už pro celý dům, nebo pro samostatnou místnost. Scény mohou být definovány např. pro spánek, dovolenou, večeři, sledování televize nebo návštěvu. Scénu lze vyvolat stiskem jednoho tlačítka, a nastavit tak všechna světla a ostatní zařízení v domě automaticky do požadovaného stavu.

Samozřejmostí je sdružení všech dálkových ovládání do jediného, namísto dnes časté situace, kdy na stole leží několik ovladačů pro domácí kino a k nim navíc ovladač rolet a klimatizace.

Pomocí elektronické regulace topení a osvětlení obvykle uspoříme až jednu třetinu nákladů, a to zároveň se zvýšením pohodlí uživatelů. Požadované teploty se dají nastavit zvlášť pro různé místnosti v závislosti na čase a dni v týdnu. Po odchodu obyvatel z domu se teplota sama sníží a vypnou se zapomenutá světla. Otevření okna zastaví topení pod ním.

1.1 Stávající systémy pro inteligentní budovy

V současné době se pro inteligentní řízení budov používá několik různých standardů, jako je například LonWorks, BACnet nebo KNX. Mezi nejrozšířenější patrně patří LonWorks, který je svou rozsáhlou sofistikovanou strukturou vhodný spíše pro větší a složitější aplikace. Naproti tomu KNX sdružuje tři existující technologie (EIB, BatiBus, EHS) a umožňuje tak komunikaci mezi přístroji mnoha výrobců. KNX má uplatnění především v menších domech.

Sběrnice KNX umožňuje adresovat až 65 tisíc jednotek, přičemž za jednotku se považuje každé zařízení, jako je spínač, zásuvka, snímač pohybu, světlo a další. Díky možnosti komunikace peer-to-peer (každý s každým) jsou možnosti propojení jednotlivých zařízení takřka neomezené. Zařízení sběrnice KNX bývá rovněž připojeno k počítači PC, v němž se uchovávají veškerá měřená data, jako jsou teploty nebo průtoky a pomocí tohoto PC je možné také celý dům ovládat.

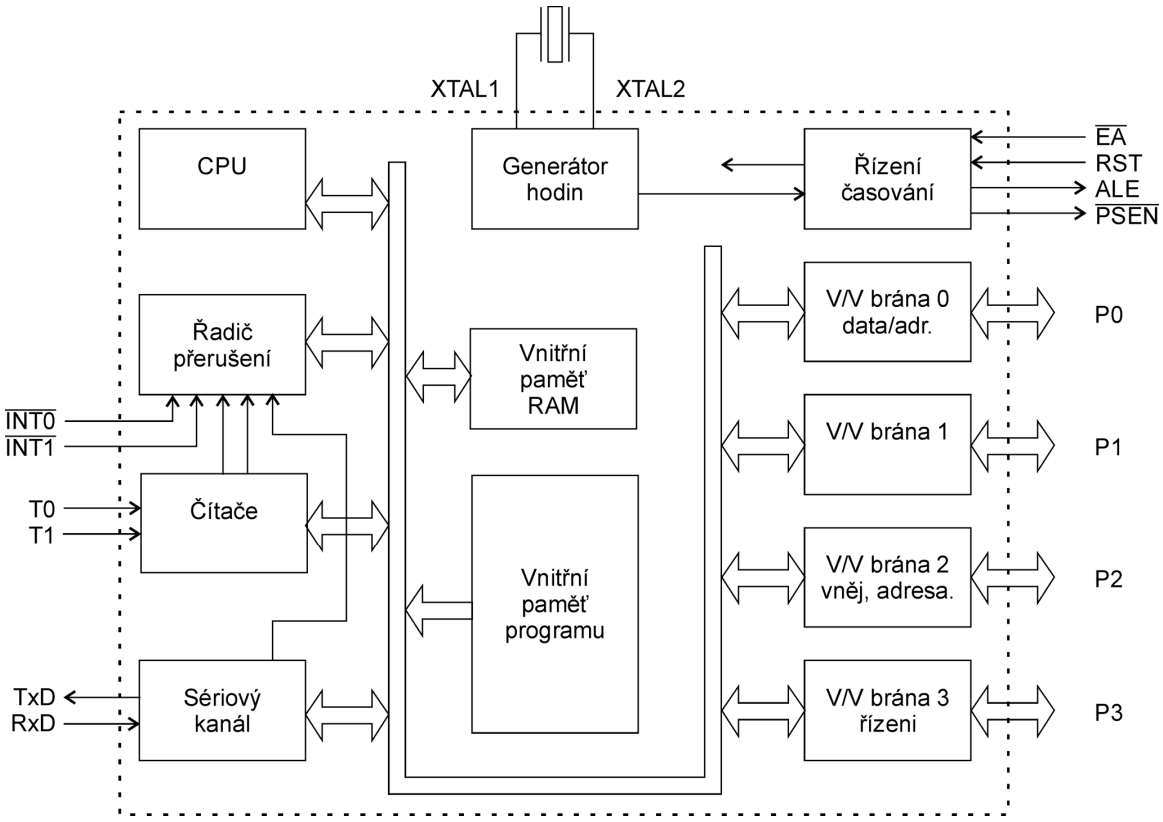
Sběrnice KNX přináší výhody jednotnosti jak komunikační sběrnice, tak i jejich instalace a díky tomu je možné do stávajícího systému inteligentního domu přidat mnoho dalších zařízení různých výrobců. Velkou nevýhodou tohoto řešení jsou však velké finanční náklady na pořízení zařízení pro tuto sběrnici. Další nevýhodou je poměrně složitá konfigurace všech zařízení, aby vykonávaly svou činnost dle požadavků uživatele. A protože každé zařízení obsahuje elektroniku nutnou k provozu a komunikaci s dalšími zařízeními, je další nevýhodou to, že každé zařízení je neustále napájeno a má tedy pořád jistý odběr proudu.

2 MOŽNOSTI PROCESORU INTEL 8051

Mikroprocesor 8051 pochází z roku 1980 a je vývojově procesorem relativné starým. U návrhářů však dosáhl takové obliby, že i v současné době se řada výrobců orientuje na výrobu procesorů s jádrem 8051, které je rozšířeno o další periferie. Procesory s jádrem 8051 jsou vyráběny v nejrůznějších provedeních, například s pamětí programu o velikosti 2kB – 32kB, kterou lze programovat jednou (OTP) nebo několikrát (EPROM), s pamětí EEPROM pro uchování konstant, s 8 nebo 10bitovým A/D převodníkem, s dvoudrátovou komunikační sběrnicí I^2C a podobně. Výrobci nabízejí procesory od základního hodinového kmitočtu 12MHz až po 33MHz ve standardním pouzdře DIL, přes provedení PLCC až po malá pouzdra PQFP.

2.1 Architektura procesoru 8051

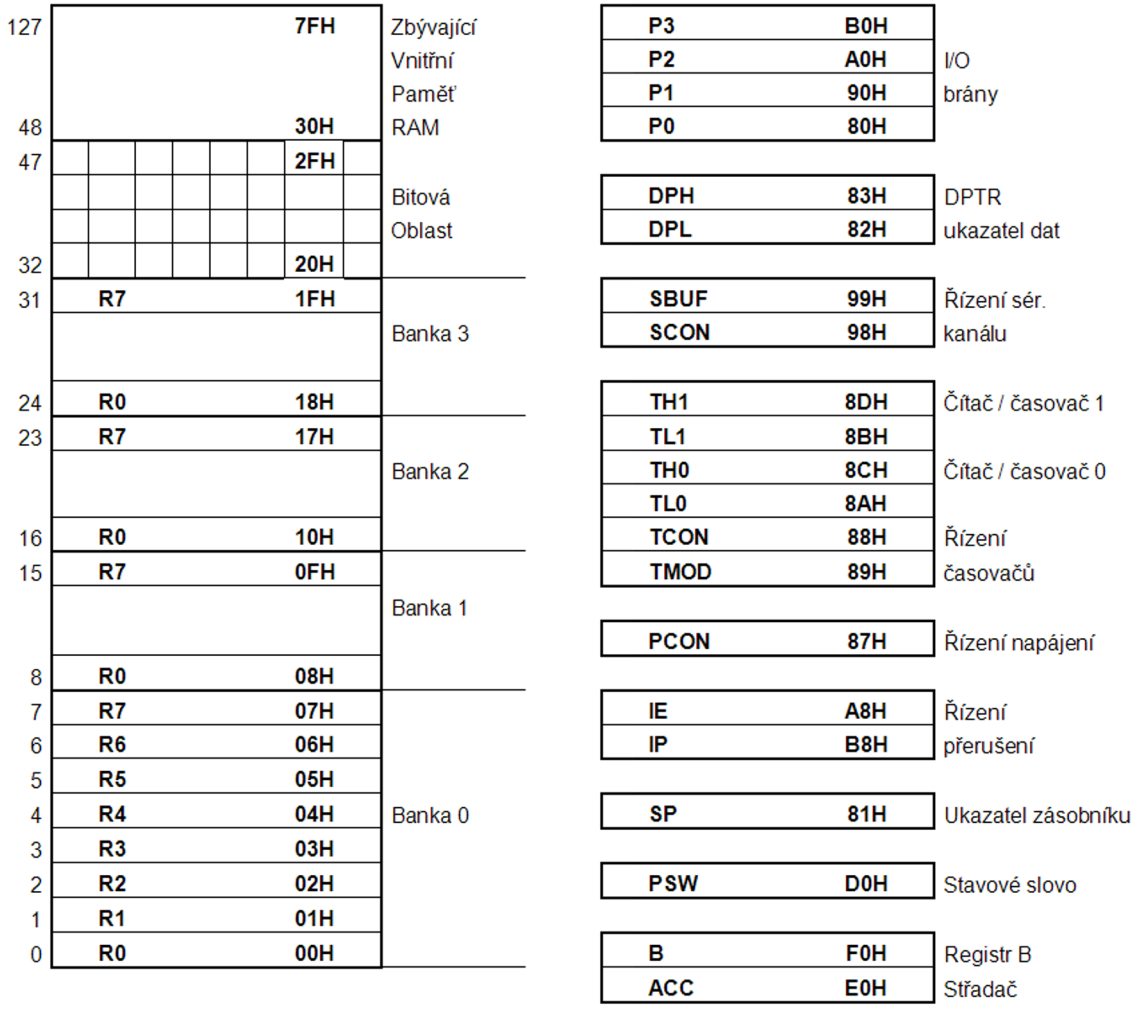
Mikroprocesor 8051 je 8bitový jednočipový mikroprocesor s harvardskou strukturou, u které je oddělena programová a datová paměť. Vnitřní struktura procesoru je znázorněna na obrázku (Obr. 1). Procesor je schopen samostatné činnosti po připojení vnějšího piezokeramického rezonátoru a jednoho napájecího napětí 5V. Na čipu je umístěna vlastní procesorová jednotka CPU, která je vnitřní společnou sběrnicí propojena s pamětí programu ROM nebo EPROM, s pamětí RAM a se čtyřmi vstupně/výstupními branami P0 až P3 zajišťujícími styk procesoru s vnějšími periferiemi. Pro snadnější styk s periferiemi je procesor vybaven řadičem přerušení, který zpracovává 5 zdrojů přerušení (2 externí, 2 od časovačů a 1 od sériového kanálu). Jednotlivá přerušení mají danou prioritu. Čítače, které usnadňují realizaci časování, jsou 16bitové s hodinovým signálem odvozeným z interního generátoru hodin nebo z vnějších vstupů T0 nebo T1. Pro snazší sériový styk s nadřazenými počítači nebo jinými spolupracujícími procesory je vybaven plně duplexním sériovým kanálem. Procesor je dále vybaven Booleovským procesorem, který umožňuje pracovat s jednotlivými bity vnitřní paměti RAM i interních periferií. Pro základní hodinový kmitočet 12MHz trvají instrukce 1μs nebo 2μs, nejdelší instrukce násobení a dělení pak trvají 4μs.



Obr. 1. Vnitřní bloková struktura procesoru 8051

2.2 Organizace paměti

Mikroprocesor 8051 má oddělené adresové prostory programu a dat, které jsou přístupné různými instrukcemi. Paměťový prostor můžeme dále dělit na vnitřní, umístěný na čipu, a vnější, který lze v případě potřeby vytvořit s pomocí dalších součástek. Proti překrývání vnitřního (4kB) a vnějšího (64kB) programového adresového prostoru je procesor vybaven vstupem EA. Je-li vstup EA=0, potom programová paměť je tvořena celou vnější pamětí, je-li vstup EA=1, potom instrukce v adresovém prostoru 000H až FFFH se čtou z vnitřní paměti ROM nebo EPROM a mimo tento prostor (1000H až FFFFH) ze zbývajících 60kB vnější paměti programu. Překrývání vnitřního a vnějšího datového adresového prostoru je odstraněno tím, že přístup do každého prostoru je realizován pomocí rozdílných instrukcí. Na obrázku (Obr. 2) je základní rozdělení vnitřní datové paměti včetně umístění speciálních registrů.



Obr. 2. Rozložení vnitřní paměti RAM procesoru 8051

Vnitřní datová paměť RAM 128 bajtů je tvořena čtyřmi bankami (0, 1, 2, 3) po osmi registrech R1 až R7 (adresy 00H – 1FH), za kterými je vyhrazeno 16 bajtů (adresy 20H – 2FH) pro tzv. bitovou oblast. Jednotlivé bity paměťových míst počínaje nejnižší adresou (20H) a nejnižším bitem (b0 – adresa bitu 00H) jsou vzestupně přímo adresovatelné. Přímo adresovatelných bitů v této oblasti je 128 (posledním bitem s adresou 7FH je bit b7 na adrese 2FH). Další datová paměť RAM, počínaje adresou 30H a konče adresou 7FH, je uživateli volně přístupná pro přímé i nepřímé adresování. Zbývajících 128 adres (od 80H do FFH) se využívá k adresování některých významných bitů příslušejících speciálním registrům.

Je-li procesor vybaven rozšířenou vnitřní datovou pamětí v adresovém prostoru 80H až FFH, potom je tato paměť s ohledem na speciální registry přístupná pouze pomocí nepřímého adresování. Vnější datová paměť RAM s kapacitou až 64kB je přístupná přes 16-bitový pomocný ukazatel datové paměti DPTR (Data Pointer). Určitou nevýhodou je to,

že DPTR může být pouze naplněn konkrétní adresou paměťového místa nebo inkrementován (DPTR může být zmenšován pouze odečítáním hodnoty od jeho dílčích částí, to je DPH a DPL). Vnější datová paměť může být přístupná i s pomocí registrů R0 a R1 příslušné aktivní banky, které se využívají k 8bitovému nepřímému adresování. Zapsáním 8bitové hodnoty do výstupní brány P2, která představuje horní část adresy při adresování vnější paměti, vybereme jeden z 256 bloků paměti RAM o 256 bajtech. Nepřímým adresováním pomocí registrů R0 a R1 potom můžeme zapsat nebo přečíst vybraný bajt ve zvoleném bloku. Oblast speciálních funkcí (SFR) je tvořena 21 registry, které leží v adresovém prostoru 80H až FFH. To znamená za vnitřní pamětí RAM nebo ve stejném adresovém prostoru jako leží rozšířená datová paměť RAM u nástupců procesorů 8051. Z toho důvodu jsou speciální registry přístupné pouze pomocí přímého adresování bajtů nebo případně i bitů.

Více informací o architektuře procesoru 8051 naleznete v [3].

2.3 Komunikační schopnosti

Procesor 8051 disponuje plně duplexním sériovým kanálem integrovaným na čipu procesoru, který umožňuje komunikaci ve standardním 8bitovém a 9bitovém asynchronním režimu nebo 8bitovém synchronním režimu s pevnou přenosovou rychlostí. Tím je výrazně usnadněna komunikace s nadřazeným počítačem např. typu PC, k jejíž realizaci je v současně době zapotřebí jeden integrovaný obvod (MAX232, MAX233) zajišťující převod úrovní TTL sériového kanálu na úrovně V24 (PC). Plně duplexní sériový kanál umožňuje současně vysílat i přijímat hodnoty po tomto kanálu, který tvoří minimálně 3 vodiče (RxD, TxD, zem). Přijímací kanál je vybaven vyrovnávacím registrem, do kterého je uložena právě přijatá hodnota, čímž je umožněn okamžitý příjem další hodnoty. Procesor není vybaven příznaky, které indikují ztrátu přijaté hodnoty (chybu přeplnění), chybu rámce a parity neb indikaci přerušení, které jsou obvyklé u specializovaných obvodů.

Režim přenosu dat sériové linky se nastavuje pomocí konfiguračních bitů SM0 a SM1 v registru SCON. Každý z těchto režimů má svou specifickou délku jednoho bajtu a přenosovou rychlost. Přehled těchto režimů je uveden v tabulce (Tab. 1).

Tab. 1. Komunikační režimy procesoru 8051

SM0, SM1	Režim	Typ přenosu	Bitová rychlost
0, 0	0	Synchronní 8bit bez rámcové synchronizace	$f_{osc}/12$
0, 1	1	8bitový UART	časovač 1
1, 0	2	9bitový UART	$f_{osc}/32$, $f_{osc}/64$
1, 1	3	9bitový UART	časovač 1

Pro komunikaci mikroprocesoru s počítačem PC je vhodné sériovou linku nastavit do módu 1. Jedná se o 8bitový UART. Přenášené hodnoty se vysílají výstupem TxD a přijímají vstupem RxD a skládají se z deseti intervalů učených převrácenou hodnotou přenosové rychlosti v baudech. První bit je vždy logická 0, takzvaný start bit, po něm následuje 8 datových bitů počínaje bitem s nejnižší váhou a poslední je vždy logická 1 představující stop bit. Přenosová rychlost je volitelná a je určena periodou přetečení čítače/časovače 1 a hodnotou bitu SMOD v registru PCON. Přenosovou rychlost v baudech je možné vypočítat podle rovnice (1), kde *TH1* je obsah registru TH1 a f_{osc} je kmitočet oscilátoru.

$$rychlost = \frac{1}{T} = \frac{2^{SMOD}}{32} \cdot \frac{f_{osc}}{12 \cdot (256 - TH1)} \tag{1}$$

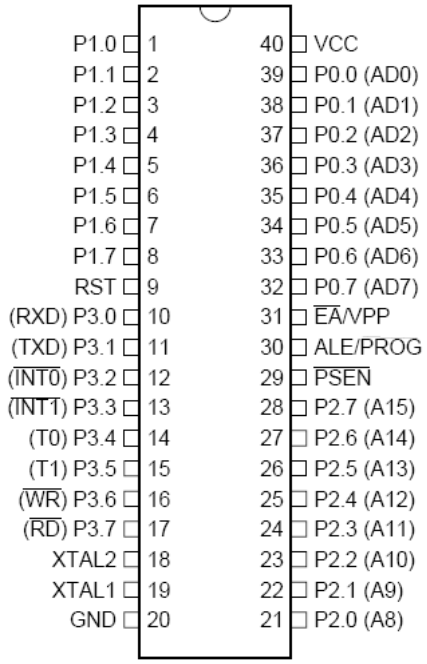
2.4 Vhodné varianty procesoru 8051

Pro účely řešení inteligentního domu je vhodný mikroprocesor například AT89C51 firmy Atmel, který se drží původní architektury 8051 a to jak po stránce hardwarové, tak i z hlediska rozložení paměti a instrukční sady. Existuje i varianta AT89C52 s programovou pamětí EPROM rozšířenou na 8kB a pamětí RAM 256kB. Dále pak existuje například AT89S51, který je možné programovat pomocí rozhraní ISP (In-System Programming) přímo v obvodu a není tedy nutné při změně programu procesor vyjímat z patice. Všechny tyto varianty se vyrábějí ve 40vodičovém pouzdře PDIP a ve 44vodičových pouzdrech PLCC a PQFP/TQFP.

Vlastnosti mikroprocesoru AT89C51 jsou následující:

- Kompatibilní s mikropočítači MCS-51
- 4kB reprogramovatelné Flash paměti

- Plně statický provoz: 0MHz – 24MHz
- Tříúrovňový zámek programové paměti
- 128x8 bitů vnitřní paměti RAM
- 32 programovatelných vstupně výstupních linek
- Dva 16bitové čítače/časovače
- Šest zdrojů přerušení
- Programovatelný sériový kanál
- Úsporné režimy Idle a Power-down

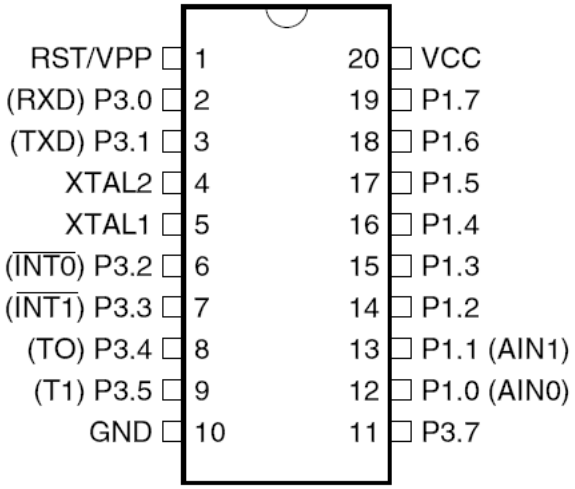


Obr. 3. Rozložení vývodů mikroprocesoru AT89C51

Pro méně náročnější řešení může být vhodným typem mikroprocesor AT89C2051 rovněž vyráběný firmou Atmel. Jedná se o menší variantu mikroprocesoru kompatibilního s architekturou 8051. Je však vyráběn ve 20vodičovém pouzdře s tím, že na vnější vodiče pouzdra nejsou vyvedeny vstupně-výstupní brány P0 a P2 pro adresování externí paměti. Programovou paměť má o velikosti 2kB, ale narozdíl od AT89C51, a jemu podobných, je tento typ vybaven analogovým komparátorem napětí, který lze použít jako jednoduchý A/D převodník.

Vlastnosti mikroprocesoru AT89C2051:

- Kompatibilní s mikropočítači MCS-51
- 2kB reprogramovatelné Flash paměti
- Plně statický provoz: 0MHz – 24MHz
- Dvouúrovňový zámek paměti
- 128x8 bitů vnitřní paměti RAM
- 15 programovatelných vstupně výstupních linek
- Dva 16bitové čítače/časovače
- Pět zdrojů přerušení
- Programovatelný sériový kanál
- Zabudovaný analogový komparátor
- Úsporné režimy Idle a Power-down



Obr. 4. Rozložení vývodů mikroprocesoru AT89C2051

3 NÁVRH ŘEŠENÍ INTELIGENTNÍHO DOMU

3.1 Návrh technického řešení

3.1.1 Řídicí jednotky

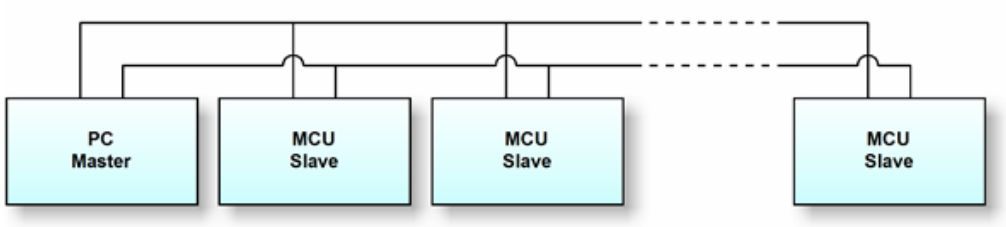
Řešení inteligentního domu spočívá ve vytvoření speciálně navržených řídicích jednotek na bázi mikroprocesoru s jádrem Intel 8051, kdy jedna jednotka bude obsluhovat jednu místnost domu. V případě použití u menších místností, jako je koupelna, WC nebo chodba, může být jedna jednotka použita pro více těchto místností. Řídicí jednotka bude obsahovat jak slaboproudou část zajišťující sériovou komunikaci, připojení spínačů pro osvětlení, senzorů otevření dveří a podobně, tak i silnoproudou část pro samotné spínání osvětlení nebo zapínání a vypínání elektrických zásuvek. Každá z těchto řídicích jednotek bude fungovat nezávisle na ostatních a všechny základní akce, například rozsvícení osvětlení při stisku spínače, budou prováděny dle vnitřního programu řídicí jednotky.

Připojení vstupních zařízení do řídicí jednotky, jako je spínač osvětlení, detektor pohybu a podobně, bude řešeno pomocí kontaktních spínačů. Lze tak použít běžně dostupné spínače osvětlení, které mají pouze kontaktní spínač bez jakékoliv elektroniky. Velkou výhodou tohoto řešení je pořizovací cena, protože klasické spínače lze pořídit řádově v cenách stokorun. Podobným způsobem je možné připojit detektor pohybu nebo požární hlásič, který má spínač s kontaktním relé. Osvětlení bude spínáno pomocí triaků uvnitř řídicí jednotky a zásuvkový okruh každé místnosti speciálním elektronickým relé určeným pro větší výkony. Některé zásuvky ale musí být k síti připojeny neustále. Jsou to zásuvky například pro lednici nebo mrazák.

3.1.2 Komunikace

Všechny řídicí jednotky v domě budou napojeny na komunikační sběrnici, na kterou bude rovněž připojen počítač typu PC. Řídicí jednotky spolu přímo nebudou komunikovat, ale komunikace bude probíhat způsobem Master – Slave, kde nadřazený počítač PC bude typu Master a řídicí jednotky typu Slave. V Master PC budou ukládána a zpracována veškerá data ze všech řídicích jednotek a na základě zadaných kriterií mohou být prováděny různé akce, jako změna výkonu topení podle teploty v místnostech, automatické rozsvícení světla v místnosti při setmění slunečního světla a podobně. Protože každá řídicí jednotka pracuje

nezávisle na ostatních, je zajištěna základní funkcionalita všech místností i v případě nečinnosti Master PC počítače, který může být odstaven například kvůli poruše nebo údržbě.



Obr. 5. Schéma komunikace jednotek a PC

3.1.3 Počítač Master PC

Vzhledem k tomu, že pro plný chod inteligentního domu je od Master PC počítače vyžadován nepřetržitý provoz 24 hodin denně, je vhodné použít systém s nízkým příkonem. Jako nejlepší volba pro sestavu počítače se nabízí základní deska VIA EPIA SP13000, znázorněna na obrázku (Obr. 6), která disponuje integrovaným procesorem o taktu 1,3GHz s příkonem maximálně 10W. Tato základní deska je vyráběna v miniaturním formátu Mini-ITX a patří mezi řešení typu „vše v jednom“. Obsahuje totiž veškerá zařízení potřebná pro chod počítače. Je to především integrovaná grafická karta, zvuková karta, dva IDE a dva SATA konektory pro připojení pevného disku, jeden PCI slot, jeden slot pro DDR paměti, ale také řadič LAN, FireWire, USB 2.0 nebo sériový port. K této základní desce stačí tedy připojit pevný disk a paměťový modul a počítač je připraven k provozu.

Obr. 6. Základní deska VIA EPIA SP13000

Kromě komunikace s řídicími jednotkami bude Master PC sloužit také jako internetový informační server, jehož prostřednictvím bude možné dům vzdáleně nejen sledovat, ale také ovládat. A to jak z lokální počítačové sítě, tak prostřednictvím sítě internet.

3.2 Zabezpečení sítě

Protože počítač Master PC slouží jako internetový server a je vystaven nejen lokální počítačové síti, ale i internetu, je třeba provést jeho zabezpečení proti útokům zvenčí, aby nedošlo k poškození nebo ztrátě jeho dat. Aby bylo takové zabezpečení zajištěno, je potřeba do sítě vložit bezpečnostní komponenty, jako například firewall, které slouží k zabránění případných útoků zvenčí a přitom zachovávají funkčnost celé sítě.

3.2.1 Firewall

Firewall je komponenta nebo skupina jednotlivých komponent vložena mezi dvě sítě. Tyto komponenty se skládají z počítačových systémů, routerů nebo skupiny routerů, fungující jako buffer mezi jakýmikoliv veřejnými sítěmi a privátní sítí. Moderní firewall má takové vlastnosti, aby zabránil neautorizovanému přístupu. Mezi tyto vlastnosti patří:

NAT

NAT (Network Address Translation) je technika překladu adres, která může schovat interní IP adresu hosta před externím hostem. Nezvaný host má mnohem větší problémy s napadením systému, který nemá registrovanou IP adresu.

Strong authentication

Technika silné autentizace se postará o povolení přístupu do sítě pouze autorizovaným osobám.

Rozšířené systémové logování

Rozšířené logování přístupů vede k zajištění zpětné kontroly. Je tak zpětně možné zjistit, kdy a kým byl vykonán přístup ke které části sítě.

Použití šifrovaného přenosu dat

Tato vlastnost mnohonásobně zvýší bezpečnost komunikace. Přenášená data jsou šifrována pomocí speciálního klíče a pro případného útočníka jsou bez toho klíče nečitelná.

Paketové filtry

Metoda filtrování paketů je založena na zkoumání, zda je paket autorizován k propuštění skrz sítě. IP filtrování paketů se obvykle provádí na routeru určenému k filtrování paketů. To může být založeno na těchto kritériích:

- Zdrojová IP adresa
- Cílová IP adresa
- TCP/UDP zdrojový port
- TCP/UDP cílový port

Správce sítě tak může předcházet některým typům útoků a může omezit přístup z internetu a na internet pro vybrané protokoly nebo počítače (podle jejich IP adresy).

Paketové filtry jsou schopny chránit i samy sebe. Kladně se u paketových filtrů hodnotí také jejich rychlost a transparentnost.

3.2.2 Typy firewallů

Hardwarový

Jde o speciální zařízení se speciálním operačním systémem. Například routery.

Výhody a nevýhody: Nízká flexibilita, vysoká cena, velká výkonnost, komplikovanější údržba a správa

Softwarový

Jde o standardní počítače jako PC provozující běžné operační systémy jako Linux nebo Windows.

Výhody a nevýhody: Vysoká flexibilita, nízká cena, nižší výkonnost, jednodušší správa

3.2.3 Zabezpečení bezdrátové sítě

Dosah Wi-Fi přístupového bodu bývá často i mimo budovu, a proto je potřeba bezdrátový přístup k síti zabezpečit. To lze provést několika způsoby, z nichž některé jsou více či méně bezpečné.

Filtrování MAC adres

U přístupového bodu je možné nastavit seznam MAC adres síťových karet, jimž je přístup povolen, nebo naopak zakázán. Tato ochrana lze však poměrně jednoduše obejít. Například v systému Windows lze změnou položky v registru MAC adresu nastavit na libovolnou hodnotu, takže se případný útočník může vydávat za pravého uživatele.

WEP

Všechna certifikovaná zařízení pro WLAN podle IEEE 802.11a/b/g mají zabudovaný základní mechanizmus zabezpečení – protokol WEP (Wired Equivalent Privacy). Jako nejstarší mechanizmus WEP nedostojí svému názvu, když zajišťuje pouze určitý stupeň utajení přenášených dat a poskytuje jen slabou autentizaci a integritu dat.

Základem WEP je tajný klíč (o délce 40 nebo 104 bitů), který sdílí všechny stanice v dané WLAN. Tento klíč se používá jak pro autentizaci, tak pro šifrování dat. Klíč je statický. Vzhledem ke slabinám WEP se doporučuje jej periodicky měnit.

WPA

WPA (Wi-Fi Protected Access) je novější bezpečnostní mechanizmus a původně měl opravit chyby WEPu. Sice nešťastně používá stejný šifrovací algoritmus RC4 kvůli jednoduchému upgradu firmwaru stávajících zařízení, ale přináší s sebou řadu vylepšení. Standardně používá 128 bitový dynamický klíč TKIP (Temporal Key Integrity Protocol), který se mění každých 10 000 paketů. Dalším zlepšením je MIC (Message Integrity Check), který je používán současně s CRC32 a tím řeší jeho nedostatky, díky kterým bylo možné změnit zprávu při zachování stejného kontrolního součtu.

RADIUS server

RADIUS (Remote Authentication Dial In User Service) server slouží ke vzdálenému ověřování klienta. Ověření klienta spočívá v nalezení jeho záznamu na RADIUS serveru.

Komunikace probíhá tak, že přístupový bod na základě nalezení přítomnosti klienta odešle výzvu k identifikaci. Klient odešle zprávu, která obsahuje identifikační údaje o klientovi. Přístupový bod zprávu přijme a pošle ji RADIUS serveru. Po prohledání databáze vyšle RADIUS server přístupovému bodu zprávu o tom, zda klienta přijmout či odmítnout a přístupový bod ji pošle ještě ke klientovi. Jestliže byl klient přijat do sítě, je pro něj otevřen příslušný port.

3.2.4 Příklady sítí

Server jako softwarový firewall

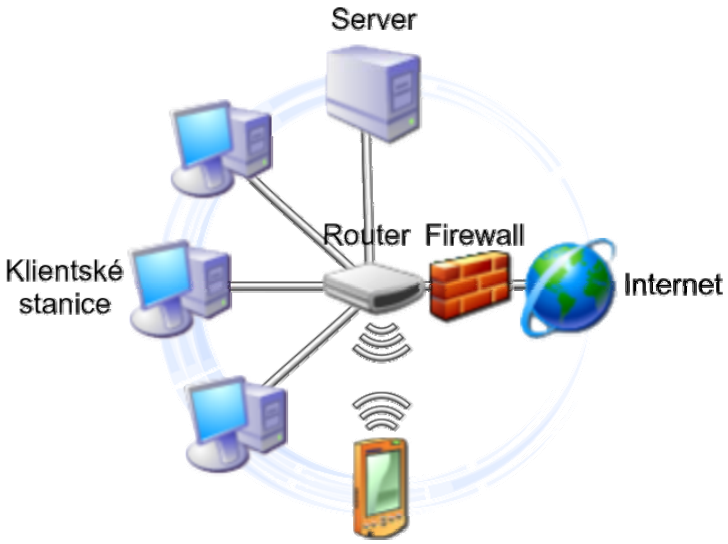
Obr. 7. Server jako firewall

Místní server slouží jako softwarový firewall a může tak zajišťovat ochranu internetového spojení kontrolou paketů i na aplikační vrstvě. Místní počítače jsou se serverem spojeny pomocí síťového mostu (bridge), který může sloužit i jako Wi-Fi přístupový bod pro připojení notebooku či PDA. Místní server může rovněž fungovat jako aplikační server místní sítě, sdílené úložiště dat, tiskový server, router, proxy brána a podobně.

Výhody/nevýhody:

- Klientské počítače v místní síti jsou chráněny softwarovým firewallem, který poskytuje vysokou flexibilitu a dobrou úroveň zabezpečení.
- Při selhání nebo výpadku serveru je celá místní síť odpojena od internetu. Je však nadále zachována komunikace mezi jednotlivými místními počítači.

Hardwarový firewall



Obr. 8. Hardwarový firewall

Připojení místní sítě k internetu je zabezpečeno hardwarovým firewallem, který funguje na principu paketového firewallu. Firewall může být někdy přímou součástí routeru. PC server může sloužit jako aplikační server, sdílené úložiště dat, nebo tiskový server. Není jej ale možné použít jako firewall nebo router. V tomto případě však výpadek serveru neohrozí připojení místní sítě k internetu.

Výhody/nevýhody:

- Připojení klientských stanic k internetu není závislé na stavu serveru.
- Síť je chráněna pouze paketovým firewallem, který je součástí routeru.

4 SOFTWAROVÁ PLATFORMA

4.1 .NET Framework

Microsoft .NET Framework je softwarová komponenta, kterou lze doplnit do operačního systému Windows (podporovány jsou Windows 98 a novější). Nabízí velké množství předem naprogramovaných řešení pro nejčastější požadavky a spravuje spouštění programů napsaných pro tuto platformu. .NET Framework je klíčová komponenta firmy Microsoft a očekává se od ní, že bude použita většinou aplikací vytvořených pro platformu Windows.

Předprogramovaným řešením systému .NET Framework je knihovna tříd, která pokrývá velký rozsah požadavků v oblastech, jako je uživatelské rozhraní, přístup k datům, databázová konektivita, kryptografie, vývoj webových aplikací, výpočetní algoritmy nebo síťová komunikace. Funkce knihoven tříd využívají programátoři, kteří je kombinují do svých vlastních kódů za účelem vytvoření vlastních aplikací.

Programy napsané pro .NET Framework jsou spouštěny v prostředí, které spravuje běhové požadavky programu. Toto prostředí, které je rovněž součástí .NET Frameworku, se nazývá Common Language Runtime (CLR). CLR poskytuje aplikaci virtuální běhové prostředí, takže programátor nemusí zvažovat schopnosti konkrétního procesoru, na kterém program běží. CLR rovněž nabízí další významné služby, jako bezpečnostní mechanismy, správu paměti a obsluhu výjimek. Knihovna tříd a CLR společně tvoří .NET Framework, který se zaměřuje na zjednodušení vývoje aplikací a na snížení zranitelnosti aplikací a počítače proti bezpečnostním hrozbám.

.NET Framework 1.0

První verze .NET Frameworku byla vydaná 13. února 2002 jako balíček k opětovné distribuci a jako SDK (Software Development Kit - nástroje pro vývoj). Tato verze je součástí prostředí Microsoft Visual Studio .NET známého jako Visual Studio 2002.

.NET Framework 1.1

Toto je první významný upgrade .NET Frameworku. Rovněž je k dispozici jako balíček k opětovné distribuci a jako SDK. Je součástí Visual Studio .NET verze 2003. Je to první verze, která je součástí operačního systému Windows, konkrétně Windows Server 2003.

Verze 1.1 přináší vylepšení v oblasti zabezpečení, nově přináší .NET Compact Framework (prostředí pro malá zařízení, jako kapesní počítače nebo chytré telefony), podporu internetového protokolu IPv6 a další množství změn v programovém API (Application Programming Interface).

.NET Framework 2.0

Verze 2.0 je vydána současně s prostředím Visual Studio 2005, Microsoft SQL Server 2005 a BizTalk 2006. Verze 2.0 je poslední verzí, která je podporována na systému Windows 2000. Rovněž je tato verze součástí systému Windows Server 2003 R2. Přináší mnohé změny v API, která dávají možnost větší kontroly nad během programu. Jedná se například o multithreading, alokaci paměti a další. Dále plně podporuje 64bitové systémy, a to jak hardwarovou platformu x64, tak IA64. Dalším přínosem této verze jsou významná vylepšení v oblasti ASP.NET a nově také .NET Micro Framework určený k vývoji aplikací pro malá zařízení v elektronice pro domácnost.

.NET Framework 3.0

Verze 3.0, nazývána také jako WinFX, obsahuje novou sadu kódu, která je součástí operačního systému Windows Vista a Windows Server Longhorn. Rovněž je k dispozici pro Windows XP SP2 a Windows Server 2003. Verze 3.0 je složena ze čtyř hlavních komponent:

- Windows Presentation Foundation (WPF), nazýváno též jako Avalon, je nové uživatelské rozhraní založené na XML a vektorové grafice.
- Windows Communication Foundation (WCF), též Indigo, je systém předávání zpráv umožňující komunikaci mezi aplikacemi jak místně, tak vzdáleně.
- Windows Workflow Foundation (WF) umožňuje vytvoření automatizovaných úloh.
- Windows CardSpace (WCS) je softwarová komponenta sloužící k bezpečnému ukládání osobních digitálních identit a poskytuje jednotné rozhraní pro volbu identity pro jednotlivé úkony, jako například přihlášení uživatele na webovou stránku.

.NET Framework 3.5

Tato verze bude obsahovat nový překladač s podporou nových vlastností, jako Language Integrated Query (LINQ) stejně jako nové vlastnosti jazyka C# nebo VB.NET. Tato verze je plánovaná jako součást nové verze prostředí Visual Studio následujícího po verzi 2005.

4.2 Programovací jazyk C#

C#, vyslovované anglicky jako C sharp (doslova to označuje notu cis), je vysokoúrovňový objektově orientovaný programovací jazyk vyvinutý firmou Microsoft zároveň s platformou .NET, později schválen standardizačními komisemi ECMA a ISO. Microsoft založil C# na jazycích C++ a Java a je tedy nepřímým potomkem jazyka C, ze kterého čerpá syntaxi. C# se využívá hlavně k tvorbě databázových programů, webových aplikací, webových služeb, formulářových aplikací ve Windows a podobně.

C# je jednoduchý, moderní, obecně účelový a objektově orientovaný programovací jazyk. Jazyk a jeho implementace poskytuje podporu pro principy softwarového inženýrství jako hlídání indexů polí, detekce nepoužití vytvořené proměnné a automatický garbage collector. Důležité jsou také jeho vlastnosti jako robustnost, trvanlivost a programátorská produktivita. Je vhodný pro vývoj softwarových komponent distribuovaných v různých prostředích. Přenositelnost zdrojového kódu je velmi důležitá, obzvláště pro ty programátory, kteří jsou obeznámeni s C a C++. Přestože C# aplikace byly zamýšleny jako úsporné, je zde citelný nárůst spotřeby RAM a výkonu procesoru.

Program „Hello World!“

Následující jednoduchá aplikace vypíše „Ahoj světe!“ na standardní výstup (konzolová aplikace).

```
1  using System;
2
3  namespace ConsoleApplication1
4  {
5      class Class1
6      {
7          [STAThread]
8          static void Main(string[] args)
9          {
10             Console.WriteLine("Ahoj, světe!");
```

```
11            }
12        }
13  }
```

4.3 ASP.NET

ASP.NET je nadstavba .NET Frameworku firmy Microsoft pro tvorbu webových aplikací a služeb. Je nástupcem technologie ASP (Active Server Pages) a přímým konkurentem JSP (Java Server Pages).

Ačkoliv název ASP.NET je odvozen od starší technologie pro vývoj webů ASP, obě technologie jsou velmi odlišné. ASP.NET je založen na CLR (Common Language Runtime), který je sdílen všemi aplikacemi postavenými na .NET Frameworku. Programátoři tak mohou realizovat své projekty v jakémkoliv jazyce podporujícím CLR, např. Visual Basic .NET, J#, C#, Managed C++, ale i mutace Perlu, Pythonu a další. Aplikace založené na ASP.NET jsou také rychlejší, neboť jsou předkompilovány do jednoho či několika málo DLL souborů, narozdíl od ryze skriptovacích jazyků, kde jsou stránky při každém přístupu znovu a znovu parsovány.

ASP.NET ulehčuje programátorům přechod od programování klasických aplikací pro Windows do prostředí webu. Stránky jsou poskládány z objektů, ovládacích prvků (Controls), které jsou obdobou ovládacích prvků ve Windows. Při tvorbě webových stránek je tedy možné používat ovládací prvky jako tlačítko (Button), nápis (Label) a další. Těmto prvkům lze přiřazovat určité vlastnosti, zachytávat na nich události, atd. Tak, jako se ovládací prvky pro Windows samy kreslí do formulářů na obrazovku, webové ovládací prvky produkují HTML kód, který tvoří část výsledné stránky poslané do klientova prohlížeče.

4.4 SQL

SQL je standardizovaný dotazovací jazyk používaný pro práci s daty v relačních databázích. SQL je zkratka anglických slov Structured Query Language (strukturovaný dotazovací jazyk). V 70. letech 20. století probíhal ve firmě IBM výzkum relačních databází. Bylo nutné vytvořit sadu příkazů pro ovládání těchto databází. Vznikl tak jazyk SEQUEL (Structured English Query Language). Cílem bylo vytvořit jazyk, ve kterém by se příkazy tvořily syntakticky co nejblíže přirozenému jazyku (angličtině). Relační databáze byly stále významnější a bylo nutné jejich jazyk standardizovat. Tento standard

bývá označován jako SQL-86 podle roku, kdy byl přijat. V dalších letech se ukázalo, že SQL-86 obsahuje některé nedostatky a naopak v něm nejsou obsaženy některé důležité prvky týkající se hlavně integrity databáze. V roce 1992 byl proto přijat nový standard SQL-92 (někdy se uvádí jen SQL2). Zatím nejnovějším standardem je SQL3 (SQL-99), který reaguje na potřeby nejmodernějších databází s objektovými prvky. Standardy podporuje prakticky každá relační databáze, ale obvykle nejsou implementovány vždy všechny požadavky normy. A naopak, každá z nich obsahuje prvky a konstrukce, které nejsou ve standardech obsaženy. Přenositelnost SQL dotazů mezi jednotlivými databázemi je proto omezená.

SQL příkazy se dělí do čtyř základních skupin.

Příkazy pro manipulaci s daty

Jsou to příkazy pro získání dat z databáze a pro jejich úpravy. Označují se zkráceně DML – Data Manipulation Language (jazyk pro manipulaci s daty).

SELECT – vybírá data z databáze, umožňuje výběr podmnožiny a řazení dat.

INSERT – vkládá do databáze nová data.

UPDATE – mění data v databázi (editace).

DELETE – odstraňuje data (záznamy) z databáze.

EXPLAIN PLAN FOR – speciální příkaz, který zobrazuje postup zpracování SQL příkazu. Pomáhá uživateli optimalizovat příkazy tak, aby byly rychlejší.

Příkazy pro definici dat

Těmito příkazy se vytvářejí struktury databáze – tabulky, indexy, pohledy a další objekty. Vytvořené struktury lze také upravovat, doplňovat a mazat. Tato skupina příkazů se nazývá zkráceně DDL – Data Definition Language (jazyk pro definici dat).

CREATE – vytváření nových objektů.

ALTER – změny existujících objektů.

DROP – odstraňování objektů.

Příkazy pro řízení dat

Do této skupiny patří příkazy pro nastavování přístupových práv a řízení transakcí. Označují se jako DCL – Data Control Language (jazyk pro ovládání dat), někdy také TCC – Transaction Control Commands (jazyk pro ovládání transakcí).

GRANT – příkaz pro přidělení oprávnění uživateli k určitým objektům.

REVOKE – příkaz pro odnětí práv uživateli.

BEGIN – zahájení transakce.

COMMIT – potvrzení transakce.

ROLLBACK – zrušení transakce, návrat do původního stavu.

Ostatní příkazy

Do této skupiny patří příkazy pro správu databáze. Pomocí nich lze přidávat uživatele, nastavovat systémové parametry (kódování znaků, způsob řazení, formáty data a času a podobně). Tato skupina není standardizována a konkrétní syntaxe příkazů je závislá na databázovém systému. V některých dialektech jazyka SQL jsou přidány i příkazy pro kontrolu běhu, takže lze tyto dialekty zařadit i mezi programovací jazyky.

4.4.1 Microsoft SQL Server

Microsoft SQL Server je systém pro správu relační databáze vytvořený firmou Microsoft. Jeho primárním dotazovacím jazykem je Transact-SQL, implementace ANSI/ISO standardu pro SQL. SQL Server je běžně používaný pro systémy s malými nebo středně velkými databázemi, ale posledních pět let zaujal významné postavení v produktech pro větší databáze. Kromě komerčních verzí Microsoft vydal také SQL Server Express, který je volně ke stažení a je bezplatnou verzí databázového systému. Tato verze má různá technická omezení, která ji činí nepoužitelnou pro velké aplikace. Primárně je však určena pro studijní účely.

II. PRAKTICKÁ ČÁST

5 NÁVRH MODELOVÉHO DOMU

Pro demonstraci řešení inteligentního domu jsem navrhl modelový rodinný dům vyobrazený na obrázku v příloze (P I). Cílem je pro tento dům navrhnout inteligentní řízení založené na bázi procesoru s jádrem Intel 8051. Dalším cílem je provést zabezpečení domu proti neoprávněnému vniknutí a proti požáru. Jedná se o zapojení nejrůznějších snímačů otevření dveří a oken, snímačů pohybu a podobně. Celý systém bude rovněž obsahovat možnost dálkového řízení domu jak pomocí domácí počítačové sítě, tak prostřednictvím internetu.

Mnou navržený dům je jednopodlažní stavba obsahující běžné místnosti jako je ložnice, kuchyně, obývací pokoj, dětské pokoje, ale také pracovna a technická místnost. Zvláště technická místnost je pro tento projekt důležitá. Do ní bude totiž umístěno hlavní vybavení inteligentního řízení.

Na obrázku v příloze (P II) je jedno z možných umístění senzorů. Pro zjednodušení je jako senzor uvažováno každé zařízení připojitelné k řídicí jednotce, ať už je to vstupní nebo výstupní zařízení. Jejich význam je uveden v tabulce (Tab. 2).

Tab. 2. Význam symbolů senzorů

Obrázek	Význam	Obrázek	Význam
	Světlo		Detektor pohybu
	Zářivka		Snímač otevření dveří/okna
	Spínač		Snímač rozbití okna
	Zásuvka		Požární hlásič

Protože jsou různé místnosti řízeny různými jednotkami, je třeba určit, která místnost bude řízena kterou jednotkou. Dále je vhodné některé malé místnosti sloučit tak, aby byly ovládány ne samostatně, ale jednou společnou řídicí jednotkou. Toto rozdělení místností podle řídicích jednotek ze znázorněno v příloze (P III), přičemž každá oblast ovládaná jednou řídicí jednotkou je tučně orámovaná.

6 HARDWAROVÉ ŘEŠENÍ

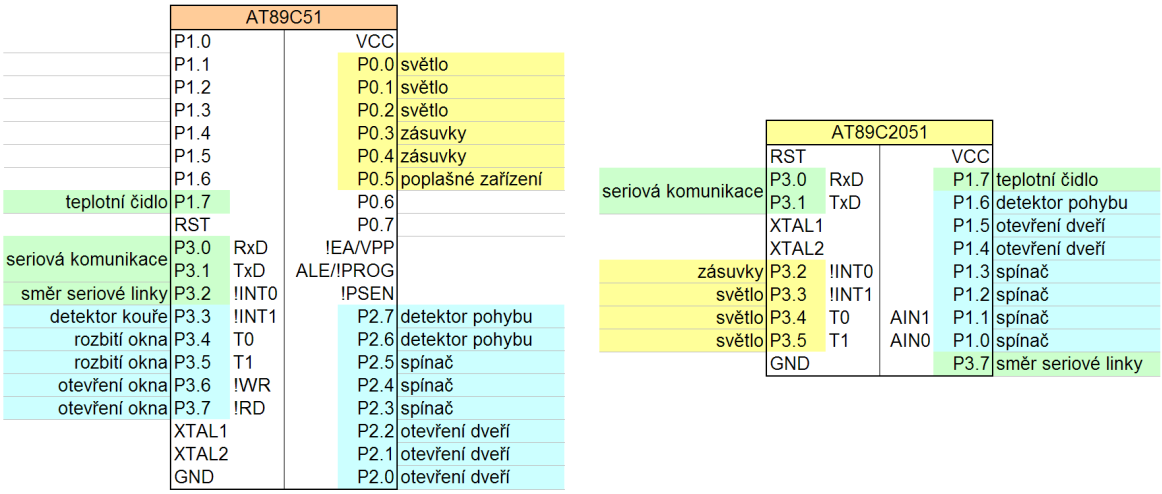
6.1 Řídicí jednotky

Při návrhu řídicích jednotek jsem vycházel z mnou navrženého modelového domu. Řídicí jednotka bude existovat ve dvou variantách. Jedna varianta s mikroprocesorem AT89C51 pro použití v běžných místnostech a druhá s mikroprocesorem AT89C2051 pro menší místnosti, jako je například koupelna. Na obrázku v příloze (P II) je možné umístění všech senzorů uvnitř modelového domu a na jejich základě jsem rozhodl o počtech slaboproudých a silnoproudých vstupů a výstupů pro obě varianty řídicí jednotky. Tyto údaje jsou uvedeny v tabulce (Tab. 3).

Tab. 3. Počet a typ vstupů a výstupů řídicích jednotek

Položka	**AT89C51**	**AT89C2051**
Sériová komunikace	3	3
Světlo	3	3
Zásuvka (okruh)	2	1
Spínač světel	3	4
Snímač otevřených dveří	3	2
Snímač otevřených oken	2	-
Detektor pohybu	1	1
Detektor kouře	1	-
Detektor rozbití okna	2	-
Poplašné zařízení	1	-
Teplotní čidlo	1	1

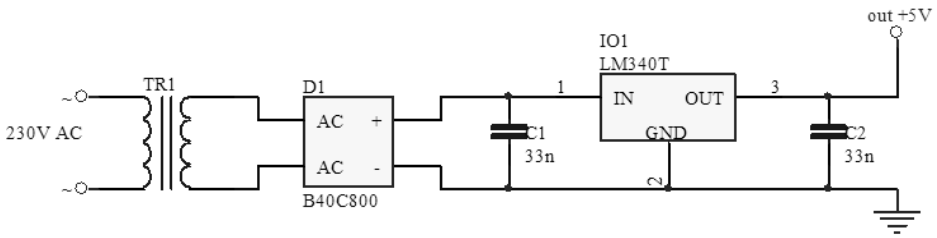
Na obrázku (Obr. 9) je znázorněno, který typ senzoru je připojen na který vývod mikroprocesoru u obou variant řídicích jednotek. Modrou barvou je znázorněna slaboproudá část a žlutě silnoproudá. Kromě samotných senzorů jsou zde označeny i vývody související s komunikací. Ty jsou označeny zelenou barvou. Jedná se nejen o samotnou sériovou komunikaci s nadřazeným Master PC jako takovou, ale také přepínání směru komunikace nebo komunikace s inteligentními teplotními čidly.



Obr. 9. Využití vstupů a výstupů obou variant mikroprocesorů

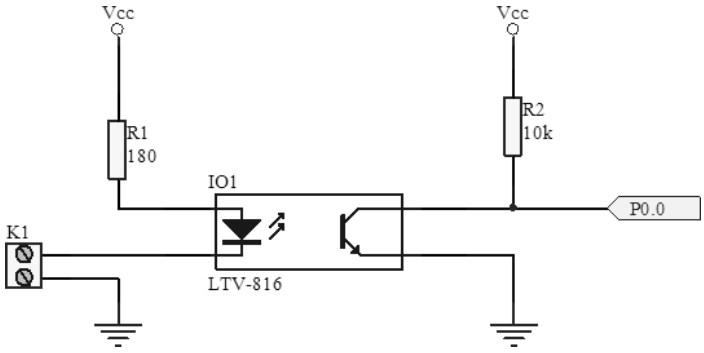
6.2 Zapojení řídicích jednotek

Napájení řídicí jednotky je řešeno jednoduchým zapojením s využitím transformátoru, usměrňovacího můstku a stabilizátoru napětí. Díky transformátoru je zde zajištěno galvanické oddělení od sítě, což je důležité z hlediska bezpečnosti. Schéma zapojení napájecího obvodu je znázorněno na obrázku (Obr. 10).



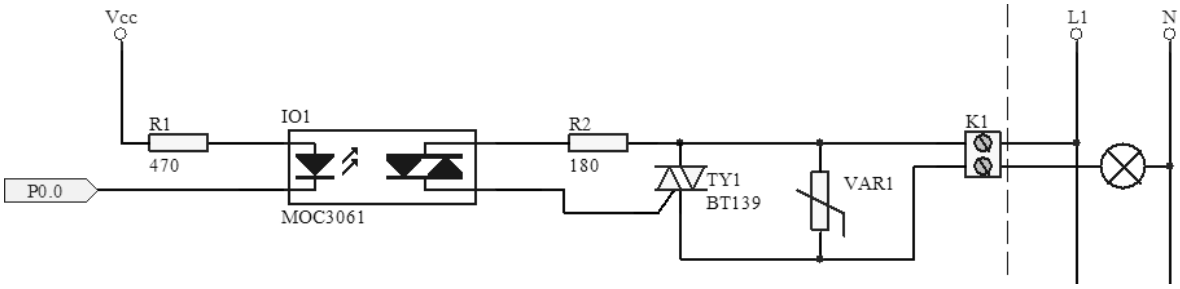
Obr. 10. Napájení řídicí jednotky

Převážná většina vstupních zařízení připojených k řídicí jednotce je založena na kontaktních spínačích. Jsou to nejen spínače pro osvětlení, ale také detektory pohybu nebo detektory kouře s reléovým výstupem. Podobně jako u napájení je i zde vhodné vstup galvanicky oddělit od samotného obvodu řídicí jednotky. To je zajištěno optronem s tranzistorovým výstupem, na obrázku (Obr. 11) označen jako IO1. Při sepnutí kontaktu na vstupu začne procházet proud diodou uvnitř optronu, čímž dojde k sepnutí tranzistoru na jeho výstupu. Tím dojde k poklesu napětí na vstupu mikroprocesoru na nulovou hodnotu, což je programově detekováno jako sepnutí vstupu. Vstup mikroprocesoru je zde označen jako P0.0. Stejným způsobem lze ale připojit vstupní zařízení na jiný libovolný vstup.



Obr. 11. Optické oddělení vstupu

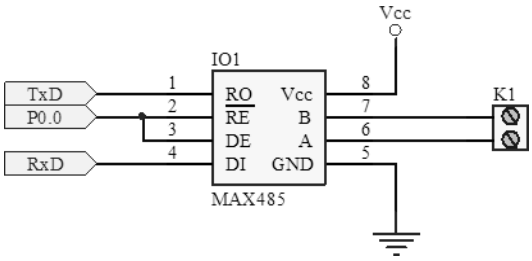
Výstupy řídicí jednotky jsou realizovány dvěma způsoby. Spínání výstupů s menším výkonem, například pro osvětlení, je řešeno pomocí triaku. Triak je spínán pomocí integrovaného obvodu MOC3061, který funguje jako optický oddělovač a zároveň zajišťuje sepnutí výstupního zařízení pří průchodu napětí nulovou hodnotou. Tím je odstraněna jakákoliv možnost rušení v síti při spínání výstupního zařízení. Tento způsob spínání výstupu je znázorněn schématem na obrázku (Obr. 12). Výstup z mikroprocesoru je zde označen jako P0.0, výstupní zařízení je připojeno do svorky K1.



Obr. 12. Spínaní výstupu pomocí triaku

Druhý způsob spínaní výstupů je pro zařízení s většími výkony. Jedná se o elektrické zásuvky v místnosti. Ty jsou spínány pomocí tzv. elektronického relé (Solid State Relay), například KSD210AC8. Tento obvod umožňuje spínat zařízení s odběrem proudu do 10A při napětí 250V.

Protože pro komunikaci nadřazeného Master PC a řídicích jednotek je použita sériová linka RS485, je nutné zajistit převod sériové linky mikroprocesoru na tuto linku. Převod je zajištěn integrovaným obvodem MAX485, jehož zapojení je nakresleno ve schématu na obrázku (Obr. 13).



Obr. 13. Převod sériové linky mikroprocesoru na linku RS485

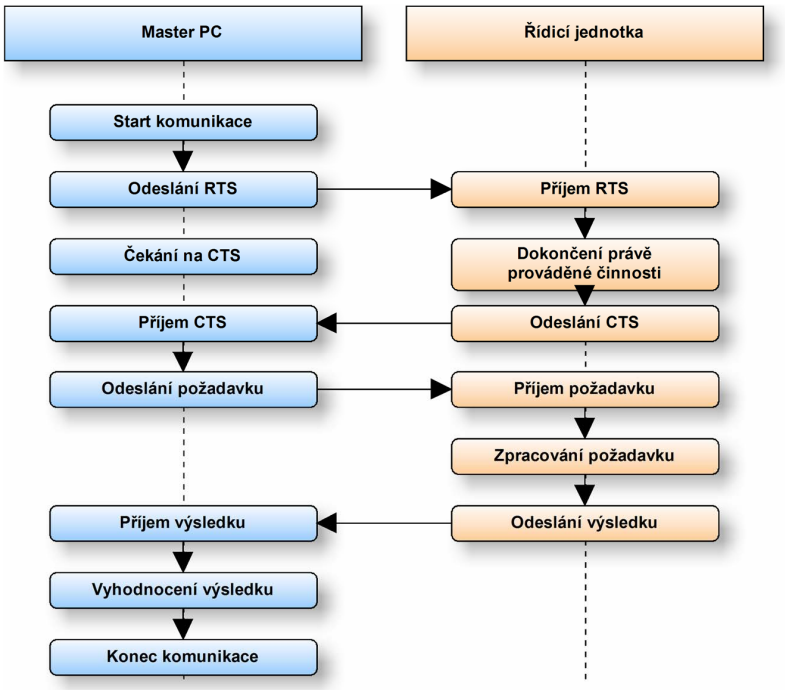
Protože v jednom okamžiku je možné na sériovou linku buď vysílat nebo přijímat, je potřeba přepínat směr přenosu dat. To je zajištěno přepínáním logických úrovní na vstupech RE a DE obvodu MAX485. Protože řídicí jednotka je typu Slave, její klidový stav je nastaven na příjem dat, kdy čeká na povel nadřazeného Master PC. Při odesílání dat nadřazenému počítači se přepne směr komunikace na vysílání a po skončení přenosu zpět na příjem dat, aby byla jednotka připravena přijmou další povely od Master PC. Samotný princip komunikace je popsán v kapitole 6.3.

Kompletní schémata zapojení obou variant řídicích jednotek jsou v přílohách (P IV) a (P V).

6.3 Komunikace jednotek

Komunikace jednotek je založena na sériovém přenosu dat přes linku RS485. Komunikace probíhá způsobem Master – Slave, kdy jedna Master jednotka posílá dotazy libovolnému počtu Slave jednotek a ty jednotce Master odpovídají podle konkrétního požadavku. Jednotkou Master je v tomto případě řídicí počítač Master PC a Slave jednotky jsou samotné řídicí jednotky uvnitř domu.

Komunikace probíhá pomocí jednoduchého komunikačního protokolu, který jsem navrhl pro tyto účely. Každá řídicí jednotka má svou hardwarovou adresu, která může nabývat hodnot 0 – 15. Z této adresy je logickým součtem s číslem 0x80 vytvořen požadavek RTS (z anglického Ready To Send), který je vyslán po sběrnici jako požadavek o navázání spojení s danou jednotkou. Jednotka nejdříve dokončí právě prováděnou operaci a po příjmu tohoto požadavku odešle kód CTS (Clear To Send), čímž dá najevo svou připravenost ke komunikaci. Poté Master vyšle svůj požadavek, který je jednotkou zpracován a následně jednotka odešle výsledek této operace zpět na Master. Princip této komunikace je znázorněn na obrázku (Obr. 14).



Obr. 14. Komunikace mezi Master PC a řídicí jednotkou

Celý komunikační protokol je uveden v příloze (P VI). Před každým dotazem je nutné vyslat požadavek RTS a počkat na odpověď CTS. Výjimku tvoří příkaz Present, kterým se zjišťuje, jaké jednotky jsou na komunikační sběrnici přítomny. Tento příkaz je vyslán bez RTS. Po jeho obdržení jednotky dokončí právě vykonávanou operaci a čekají na opětovné přijetí Present požadavku. Tím se jednotky synchronizují, aby nedošlo k situaci, kdy začnou vysílat dvě nebo více jednotek současně. Po přijetí druhého Present požadavku jednotky jedna za druhou odesílají své adresy, aby Master věděl, která jednotka je fyzicky přítomna.

6.4 Počítač Master PC

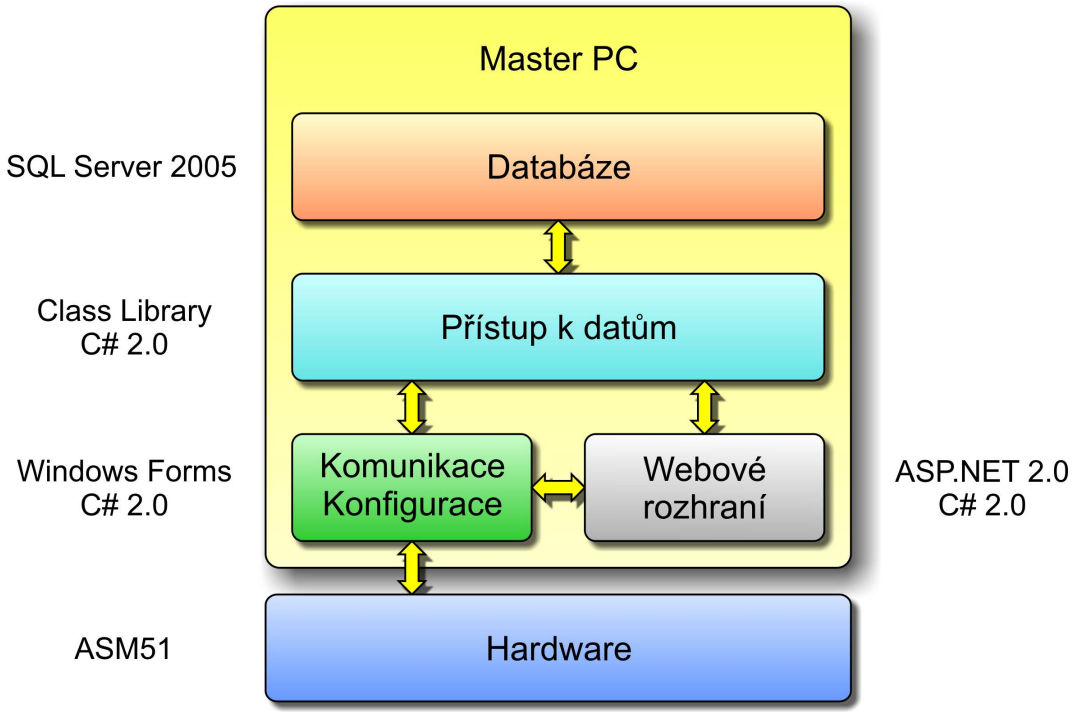
Jako počítač Master PC může sloužit libovolný osobní počítač typu PC. Protože je systém inteligentního domu navržen tak, že Master PC neustále komunikuje s řídicími jednotkami a je tedy nepřetržitě zapnutý, je vhodné použít počítač s nízkým příkonem. Jako základní deska je tedy vhodná například VIA EPIA SP13000 popsaná v kapitole 3.1.3. Tato základní deska je osazena procesorem s nízkým příkonem, který dosahuje maximálně 10W. K této základní desce stačí doplnit pouze paměťový modul a pevný disk a počítač je kompletní.

Protože Master PC bude sloužit zároveň jako internetový server, je vhodným operačním systémem Windows Server 2003 R2. Tento systém nabízí patřičné nástroje pro konfiguraci a zabezpečení sítě. Rovněž je na něm zajištěna správná činnost veškerého software, jako je Microsoft SQL Server 200 Express nebo .NET Framework.

Vzhledem k tomu, že řídicí jednotky komunikují po sériové sběrnici RS485, kterou většina osobních počítačů není vybavena, je třeba zajistit převod této linky buďto na RS232 nebo na USB. Jako převodník může sloužit produkt OM USB RS vyráběný firmou Orbit Merret. Jedná se o převodník z RS232 a RS485 na USB, který je z USB přímo napájen. Do OS Windows stačí nainstalovat potřebné ovladače a komunikace může začít.

7 SOFTWAROVÉ ŘEŠENÍ

Softwarové řešení je rozdělenou na dva základní celky. Je to software běžící na řídicích jednotkách a software běžící na Master PC. Struktura tohoto softwarového řešení je znázorněna na obrázku (Obr. 15).

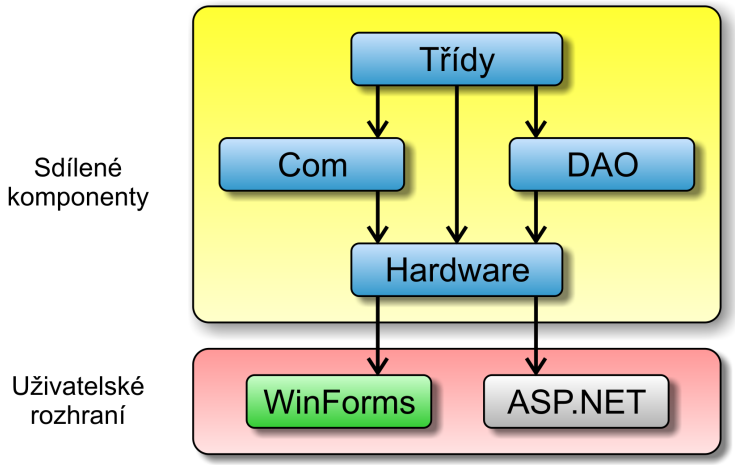


Obr. 15. Struktura softwarového řešení

Celý informační systém, který zajišťuje sledování, vyhodnocování a ovládání stavu inteligentního domu, jsem vytvořil s využitím technologií společnosti Microsoft. Jedná se o technologii .NET ve verzi 2.0 a SQL Server 2005. K vývoji tohoto systému jsem použil volně dostupné vývojové nástroje Visual C# 2005 Express Edition, Visual Web Developer 2005 Express Edition, SQL Server 2005 Express Edition a SQL Server Management Studio Express. Pro vytvoření aplikací pro mikroprocesor řídicích jednotek jsem použil obyčejný textový editor a překladač 8051 Cross Assembler společnosti MetaLink.

7.1 Logická struktura informačního systému

Informační systém je rozdělen do dvou logických celků. Jedná se o sdílené komponenty zajišťující přístup k datům a komunikaci a dále je to samotné uživatelské rozhraní. Znázornění tohoto rozdělení je na obrázku (Obr. 16).



Obr. 16. Logická struktura informačního systému

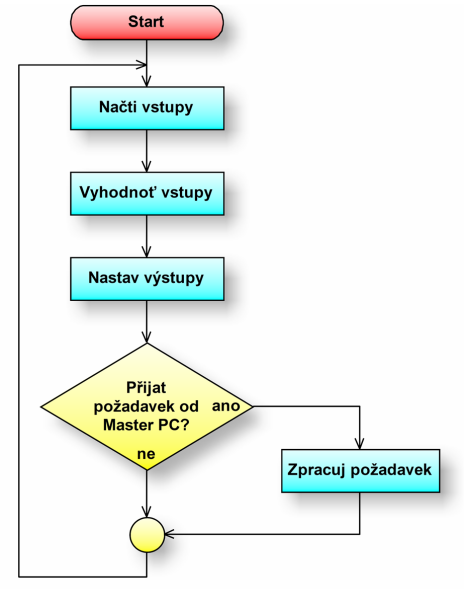
Sdílené komponenty jsou jednotlivé části programu zkompilované do DLL knihoven, kde každá komponenta zastává svou specifickou činnost:

- Třídy – Zde jsou definovány veškeré třídy reprezentující každý objekt v systému inteligentního domu. Jedná se nejen o samotné objekty místností, senzorů nebo jednotek, ale jsou ze rovněž třídy reprezentující grafické zobrazení uživatelem definovaných pohledů. Je to například samotný pohled zobrazení, ikona, zvýraznění ikony a podobně.
- Com – Tento objekt zastává činnost veškeré komunikace nejen se samotným hardwarem inteligentního domu, ale rovněž síťovou komunikaci na místním počítači nebo lokální sítí za účelem výměny dat mezi řídicí a internetovou aplikací.
- DAO – Název DAO vznikl z anglického Data Access Object. Jedná se o komponentu zajišťující přístup k datům v databázi. Každá část programu, která požaduje čtení nebo ukládání hodnot databáze, nekomunikuje přímo s databází, ale volá metody této komponenty. Výhoda tohoto způsobu řešení je ta, že v případě změny databáze nebo změny způsobu ukládání dat jako takových není třeba program nijak upravovat, ale pouze se přizpůsobí metody této komponenty.
- Hardware – V této komponentě je zapouzdřena veškerá funkcionalita systému inteligentního domu. Jsou zde logicky uspořádány třídy reprezentující objekty inteligentního domu a také veškeré komunikační funkce.

7.2 Program řídicích jednotek

Program pro řídicí jednotky jsem vytvořil v programovacím jazyce assembler pro architekturu x51 (zkráceně ASM51). Uvnitř každé jednotky je stejný program, pouze mírně upravený podle počtu a druhu senzorů uvnitř místnosti. Tento program zajišťuje základní funkcionalitu, aby každá z místností byla funkční i při odpojení Master PC. To znamená, aby například byla zajištěna funkce osvětlení při stisku příslušného spínače nebo při detekci pohybu v případě vnějšího osvětlení. Ta důležitější část programu, která je ve všech jednotkách stejná, je komunikace s Master PC popsaná v kapitole 6.3

Program je tvořen nekonečnou smyčkou, v níž se neustále opakují stejné činnosti. Program načte vstupy, vyhodnotí je a podle nastavených pravidel nastaví hodnoty výstupů. Poté zkontroluje sériovou linku, zda nepřišel z Master jednotky nějaký požadavek. Pokud ano, požadavek se zpracuje a jednotka pokračuje dál ve své činnosti. Tento princip činnosti je zjednodušeně znázorněn na obrázku (Obr. 17).



Obr. 17. Obecný princip programu řídicí jednotky

Celý tento program, včetně řádných komentářů popisujících jeho činnost, je uložen na médiu přiloženém k diplomové práci (P VII).

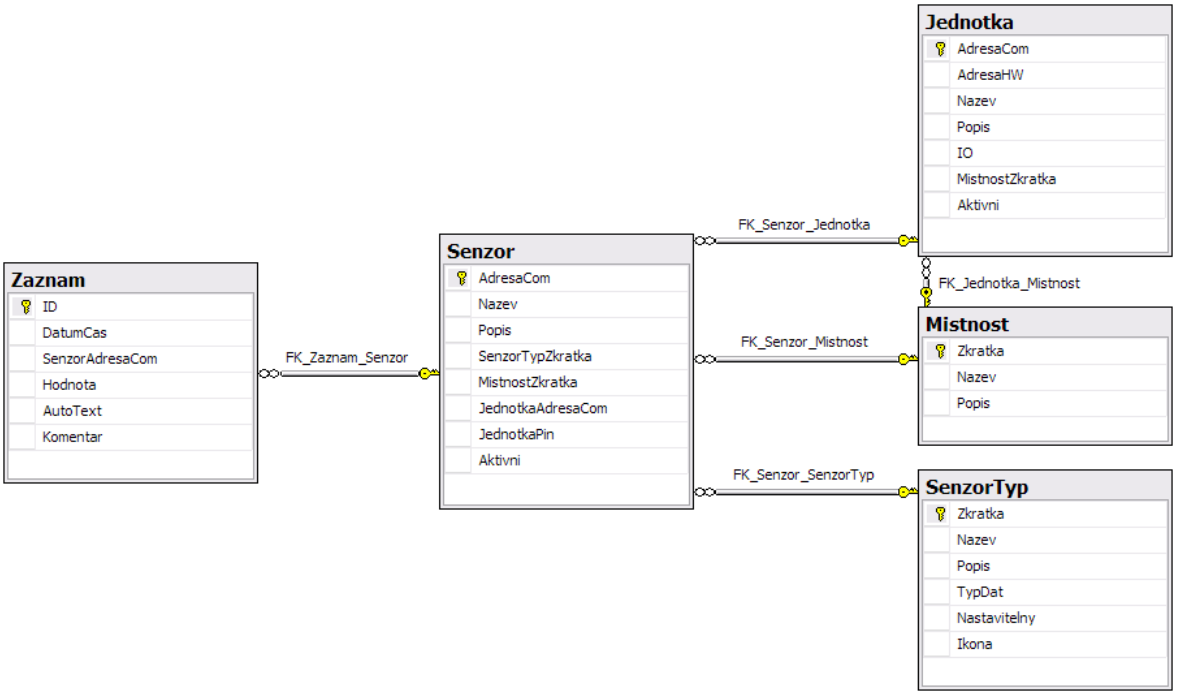
7.3 Databáze

Databáze pro systém inteligentního domu běží na Microsoft SQL Serveru 2005 Express Edition, který je s jistými omezeními dostupný zdarma. Omezení jsou v tomto případě nepodstatná a jedná se například o maximální velikost jedné databáze. Ta může dosahovat nejvýš 4GB, což je pro potřeby tohoto systému víc než dostačující. Databázi jsem vytvořil a spravoval pomocí nástroje Microsoft SQL Server Management Studio Express, kde je velice snadným způsobem možné vytvořit potřebné tabulky, uložené procedury, nastavit přístupová práva a podobně.

Databáze slouží nejen pro uchovávání veškerých měřených hodnot v domě, ale také je v ní uložena samotná definice struktury celého systému. To znamená, jak se která místnost jmenuje, kde leží která jednotka, jaký senzor je k ní připojen a podobně.

7.3.1 Struktura databáze

Na obrázku (Obr. 18) je znázorněna základní struktura databáze pro systém inteligentního domu. Jedná se o pět hlavních tabulek mezi sebou logicky propojených. Tyto tabulky slouží k definici závislostí mezi jednotlivými objekty inteligentního domu.



Obr. 18. Základní struktura databáze

V tabulce `SenzorTyp` jsou definovány typy senzorů použitých v inteligentním domě (světlo, teplotní čidlo, detektor pohybu…) a jejich základní vlastnosti.

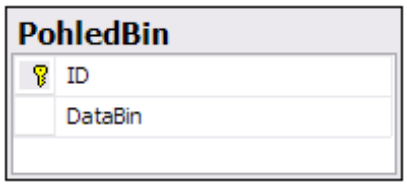
Tabulka `Mistnost` definuje veškeré místnosti v domě. Tato tabulka má význam především pro třídění a filtrování zobrazovaných záznamů. Například lze s její pomocí zobrazit všechny senzory ležící v kuchyni nebo zobrazit historii měřených teplot pro obývací pokoj.

Tabulka `Jednotka` reprezentuje samotnou řídicí jednotku. V této tabulce jsou uloženy informace o každé řídicí jednotce včetně její hardwarové adresy a informace o počtu vstupů a výstupů jednotky.

V tabulce `Senzor` jsou pak údaje o konkrétních senzorech. Pro každý senzor je zde vložena informace o tom, jakého je senzor typu, v jaké místnosti leží a k jaké řídicí jednotce je připojen. Rovněž je zde uvedeno, k jakému vstupu nebo výstupu jednotky je senzor připojen. Tento údaj může nabývat hodnoty 0 – 31 a ve skutečnosti znamená, na jaký pin mikroprocesoru je připojen.

Do tabulky `Zaznam` jsou ukládány hodnoty senzorů podle nastavených pravidel. Z této tabulky jsou data načítána například za účelem zobrazení grafu historie měřených hodnot pro daný časový úsek. Ukládána jsou do ní data buďto periodicky v uživatelem nastaveném intervalu, nebo při změně hodnoty senzoru.

Na obrázku (Obr. 19) je tabulka `PohledBin`, která slouží k uchování dat pohledů zobrazení pro rozhraní internetových stránek. Přípona `Bin` v názvu tabulky znamená, že veškeré údaje jsou zde uloženy v podobě binárních dat. Datovou strukturu pohledů zobrazení by samozřejmě bylo možné vyjádřit pomocí propojených tabulek podobně, jako v případě jednotek a senzorů, ale uložením těchto dat jako jednoho binárního celku je podstatně jednodušší a databáze zároveň není zatěžována velkým množstvím dotazů při načítání nebo ukládání změn pohledů zobrazení.



Obr. 19. Tabulka pro uložení pohledů zobrazení

7.3.2 Uložené procedury

K datům uloženým v databázi je možné přistupovat několika způsoby. Za chodu programu lze přímo sestavovat různé výběrové, vkládací nebo mazací dotazy (SELECT, INSERT,

UPDATE, DELETE…) a tyto dotazy nad databází pak spouštět. Tento způsob by bezesporu fungoval, avšak přináší jedno velké riziko. V případě nesprávně sestaveného dotazu může dojít k poškození dat uvnitř databáze, nebo i k poškození celé její struktury. Této slabiny využívají například útoky označované jako injektáž SQL. Jedná se vlastně o proces, kdy se do aplikace předá nepředpokládaný, záměrně poškozující, SQL kód. Představme si například jednoduchou situaci, kdy nějaký systém vypisuje vlastnosti uživatele podle zadaného uživatelského jména. Výběrový dotaz může být sestaven takto:

```
dotaz = "SELECT * FROM uzivatele WHERE jmeno='" + jmeno + "';";
```

Za normálních okolností, kdy uživatel zadá například jméno *Honza*, se spustí dotaz

```
SELECT * FROM uzivatele WHERE jmeno='Honza';
```

A zobrazí se vlastnosti tohoto uživatele. Pokud ale uživatel zadá jako jméno řetězec

```
a';DROP TABLE uzivatele
```

Dojde ke spuštění dvou dotazů oddělených středníkem.

```
SELECT * FROM uzivatele WHERE jmeno='a';DROP TABLE uzivatele;
```

První dotaz pouze zobrazí vlastnosti uživatele se jménem „a“, který pravděpodobně ani není vytvořen a vzápětí poté druhý dotaz z databáze vymaže tabulku s názvem `uzivatele` a dojde tak k poškození celého systému. Proti tomuto druhu útoku se dá snadno zabránit použitím uložených procedur databáze, kdy se předem připraví sada dotazů manipulujících s daty a ty se pak z hlavní aplikace pouze volají s předem definovanými parametry. Parametry uložené procedury jsou brány jako celek bez ohledu na to, jaké znaky jsou v nich použity a i když se uživatel pokusí provést útok zadáním jména `a';DROP TABLE uzivatele`, uložená procedura celý tento řetězec vyhodnotí jako jediný parametr a vyhledá tak uživatele se jménem `a';DROP TABLE uzivatele`, který s nejvyšší pravděpodobností neexistuje. Částečná nevýhoda použití uložených procedur je, že je potřeba předem znát veškeré operace, které budou s databází prováděny a pro tyto operace připravit uložené procedury. Tato drobná nevýhoda je však nepatrná v porovnání s výhodou zabezpečení aplikace proti útokům injektáží SQL. Dalším krokem, jak vylepšit obranu proti narušení databáze, je povolit aplikaci přístup pouze k těmto uloženým procedurám a přístup k samotným tabulkám databáze zakázat. Zabrání se tak, třeba i nechtěnému, spuštění poškozujícího SQL kódu.

Pro přístup k datům v databázi informačního systému pro inteligentní dům jsem použil výhradně uložené procedury, které lze rozdělit do čtyř základních kategorií:

Vkládání a úprava dat

Tab. 4. Uložené procedury pro vkládání dat

Název procedury	Argumenty	Význam procedury
UpravSenzorTyp	Zkratka Nazev Popis TypDat Nastavitelny Ikona	Procedura přidá nový typ senzoru. Pokud typ senzoru se zadanou zkratkou již existuje, pouze změní jeho hodnoty.
UpravMistnost	Zkratka Nazev Popis	Procedura přidá novou místnost. Pokud místnost se zadanou zkratkou již existuje, pouze změní její hodnoty.
UpravJednotku	AdresaCom AdresaHW Nazev Popis IO MistnostZkratka	Procedura přidá novou jednotku. Pokud jednotka se zadanou komunikační adresou (AdresaCom) již existuje, pouze změní její hodnoty.
UpravSenzor	AdresaCom Nazev Popis SenzorTypZkratka MistnostZkratka JednotkaAdresaCom JednotkaPin Aktivni	Procedura přidá nový senzor. Pokud senzor se zadanou komunikační adresou (AdresaCom) již existuje, pouze změní jeho hodnoty.
UpravPohledBin	ID DataBin	Procedura vloží nový pohled zobrazení. Pokud již existuje pohled se zadaným ID, pouze se změní jeho data.
VlozZaznam	SenzorAdresaCom Hodnota Komentar	Procedura vloží do historie záznamů hodnotu senzoru identifikovaného jeho komunikační adresu. Uložená procedura automaticky k záznamu doplní datum a čas, kdy byl záznam do databáze vložen.

Mazání dat

Tab. 5. Uložené procedury pro mazání dat

Název procedury	Argumenty	Význam procedury
SmazSenzorTyp	Zkratka	Procedura vymaže typ senzoru se zadanou zkratkou.
SmazMistnost	Zkratka	Procedura vymaže místnost se zadanou zkratkou.
SmazJednotku	AdresaCom	Procedura vymaže jednotku se zadanou komunikační adresou.
SmazSenzor	AdresaCom	Procedura vymaže senzor se zadanou komunikační adresou.
SmazPohledBin	ID	Procedura vymaže pohled zobrazení se zadaným ID.

Načtení dat

Tab. 6. Uložené procedury pro načtení dat

Název procedury	Argumenty	Význam procedury
NactiSenzorTyp	Zkratka	Procedura načte konkrétní typ senzoru podle zadané zkratky.
NactiSenzorTypy	-	Procedura načte všechny typy senzorů.
NactiMistnosti	-	Procedura načte všechny místnosti.
NactiJednotky	MistnostZkratka	Procedura načte místnost se zadanou zkratkou. Pokud zkratka není zadaná, načte procedura všechny místnosti.
NactiSenzory	SenzorTypZkratka MistnostZkratka JednotkaAdresaCom	Procedura načte senzory, které jsou zadaného typu, leží v zadané místnosti a jsou připojeny k zadané jednotce. Pokud některý z těchto argumentů není zadán, nebere se v úvahu.
NactiPohledyBin	ID	Procedura načte pohled zobrazení se zadaným ID. Pokud je ID=0, načte procedura všechny pohledy zobrazení.
NactiZaznamy	SenzorAdresaCom DatumOd DatumDo	Procedura načte uloženou historii záznamů pro zvolený senzor ve vybraném časovém úseku.

Pomocné

Tab. 7. Pomocné uložené procedury

Název procedury	Argumenty	Význam procedury
`MistnostZavisle`	`Zkratka`	Procedura načte všechny jednotky a senzory, které leží v místnosti zadané její zkratkou. Procedura je použita při mazání místnosti pro kontrolu, zda je místnost prázdná a zda ji lze tedy smazat.
`SenzorTypZavisle`	`Zkratka`	Procedura načte všechny senzory, které jsou zadaného typu. Procedura je použita při mazání typu senzoru pro kontrolu, zda existuje senzor zadaného typu a zda jej lze tedy smazat.

Následující ukázka demonstruje SQL skript, který vytvoří jednu z uložených procedur, konkrétně `UpravSenzorTyp`. Ta slouží k upravení typu senzoru. V případě, že požadovaný typ senzoru v tabulce neexistuje, vytvoří se nový záznam s danými hodnotami parametrů.

```
1  CREATE PROCEDURE UpravSenzorTyp
2   @Zkratka NVARCHAR(10),
3   @Nazev NVARCHAR(50),
4   @Popis NVARCHAR(50),
5   @TypDat INT,
6   @Nastavitelny INT,
7   @Ikona VARBINARY(MAX)
```

V uložené proceduře jsou nejprve definovány parametry, se kterými se pracuje. Každý parametr má přesně daný svůj datový typ a velikost. Datový typ `INT` má automaticky velikost 4B, u ostatních je velikost zadána hodnotou v závorce. V případ datového typu `NVARCHAR` číslo v závorce udává maximální počet znaků, který lze parametru přiřadit.

```
8  AS
9  DECLARE @Pocet INT
10 SET @Pocet=0
```

Následuje deklarace pomocné proměnné `Pocet`, která se dále využije při rozhodování, zda bude vložen nový záznam nebo upraven již existující.

```
11 BEGIN
12  SELECT @Pocet = COUNT(Zkratka)
13        FROM SenzorTyp
```

```
14          WHERE Zkratka=@Zkratka
```

Proměnná `Pocet` se naplní výsledkem dotazu na zjištění počtu záznamů s odpovídající hodnotou primárního klíče `Zkratka`.

```
15    IF (@Pocet=0)
16    BEGIN
17          INSERT SenzorTyp (Zkratka, Nazev, Popis, TypDat,
    Nastavitelny, Ikona)
18          VALUES (@Zkratka, @Nazev, @Popis, @TypDat, @Nastavitelny,
    @Ikona)
19          RETURN @@IDENTITY
20    END
```

Pokud není žádný takový prvek v tabulce nalezen, vloží se příkazem `INSERT` nový.

```
21    IF (@Pocet=1)
22    BEGIN
23          UPDATE SenzorTyp SET
24          Nazev=@Nazev, Popis=@Popis, TypDat=@TypDat,
    Nastavitelny=@Nastavitelny, Ikona=@Ikona
25          WHERE Zkratka=@Zkratka
26    END
```

Naopak pokud je záznam nalezen, provede se příkazem `UPDATE` jeho úprava.

```
27 END
```

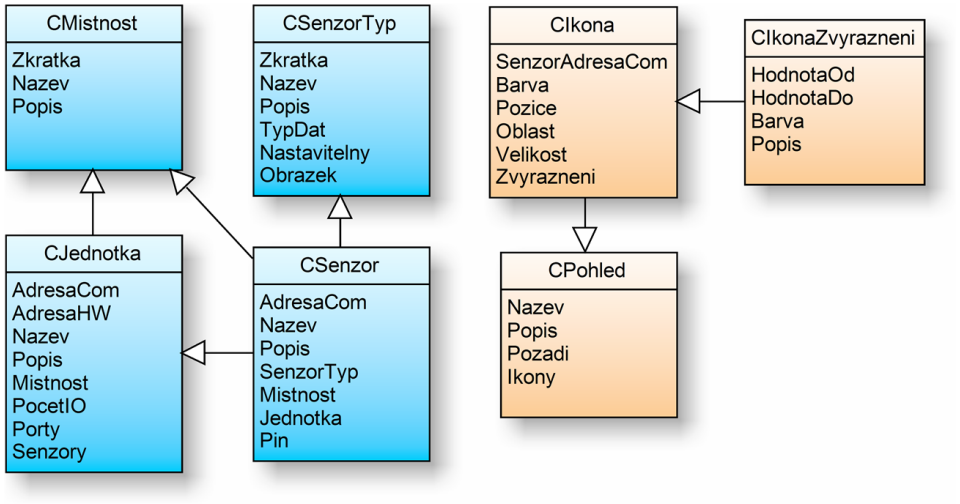
Za zmínku zde stojí testovací podmínka `IF (@Pocet=1)`. Mohlo by se zdát, že nastane situace, kdy je v tabulce `SenzorTyp` více záznamů se stejnou hodnotou položky `Zkratka` a že je tedy potřeba podmínku nastavit na `IF (@Pocet>1)`. Taková situace ale v praktickém provozu nikdy nastat nemůže, protože záznamy jsou do této tabulky vkládány pouze uloženou procedurou `UpravSenzor` a žádnou jinou. Takže v případě, že žádný záznam s danou hodnotou `Zkratka` neexistuje, vytvoří se právě jeden nový. Pokud již takový existuje, nový se nevloží, ale existující se přepíše. Nikdy tedy nenastane situace, kdy by se do tabulky `SenzorTyp` vložilo více záznamů se stejnou hodnotou pole `Zkratka`.

7.4 Sdílené komponenty

Sdílené komponenty jsou sada zdrojových kódů zkompilovaných do DLL knihoven, aby je bylo možné využít více programy současně. Jsou znázorněny na obrázku (Obr. 16) a využívá je jak řídicí aplikace, tak aplikace internetová.

7.4.1 Třídy

Sdílená komponenta Třídy tvoří základní vnitřní strukturu informačního systému. Jsou zde definovány veškeré třídy reprezentující každý objekt v systému inteligentního domu. Nejdůležitější třídy, které se podílejí na činnosti inteligentního domu, jsou společně s jejich nejdůležitějšími vlastnostmi zakresleny na obrázku (Obr. 20).



Obr. 20. Klíčové třídy sdílené komponenty Třídy

Tyto třídy přímo kopírují strukturu databáze z obrázku (Obr. 18) a (Obr. 19). Jsou do nich načítána veškerá data z databáze a poté je s nimi dále manipulováno dle potřeby. Kromě těchto tříd komponenta obsahuje pomocné třídy například pro převody mezi různými datovými typy nebo třídy pro logování nejrůznějších stavových a chybových hlášení.

Z každé třídy, popisující strukturu systému, je vytvořena kolekce zděděním od třídy `List`, aby do ní bylo možné ukládat více objektů stejného typu. U třídy CMistnost je taková třída deklarovaná jako

```
1 public class CMistnosti : List<CMistnost>
2 {
3     …
4 }
```

Stejným způsobem jsou vytvořeny kolekce i z ostatních tříd. Do takto vytvořené kolekce pak lze snadno přidávat nové objekty, odstraňovat nebo měnit jejich parametry. A to buď prostým zadáním indexu položky, například `mistnosti[0]`, nebo pomocí vlastního indexeru. Ten umožňuje vyhledat položku nejen podle číselného indexu, ale i například pomocí řetězcové hodnoty. Indexer pro třídu `CMistnosti` je vytvořený takto:

```
1 public CMistnost this[string Zkratka]
```

Položka kolekce se bude vyhledávat podle hodnoty řetězce `Zkratka`.

```
2 {
3     get
```

Sekce `get` je určena pro případy, kde je položka z kolekce načítána, a to například pomocí příkazu `mistnost = mistnosti["OBYV"]`.

```
4     {
5         foreach (CMistnost m in this)
```

Cyklem `foreach` se projdou všechny prvky v této kolekci.

```
6         {
7             if (m.Zkratka == Zkratka)
8             {
9                 return m;
```

Pokud se hodnota indexu `Zkratka` shoduje s položkou `Zkratka` aktuálně prohledávané položky, je tato položka předána jako návratová hodnota.

```
10            }
11        }
12        return null;
```

Pokud se při průchodu položkami kolekce nepodařilo najít položku s vyhovujícím parametrem `Zkratka`, je jako výsledek předána hodnota `null`.

```
13    }
14    set
```

Sekce `set` je určena pro případ, kdy se položka kolekce nastavuje, například pomocí příkazu `mistnosti["OBYV"] = novaMistnost`.

```
15    {
16        bool nasel = false;
17        for (int i = 0; i < this.Count; i++)
```

V cyklu se projdou všechny položky kolekce.

```
18        {
19            if (this[i].Zkratka == Zkratka)
20            {
21                this[i] = value;
22                nasel = true;
23                break;
24            }
```

Pokud je nalezena položka s odpovídající hodnotou pole `Zkratka`, je této položce nastavena nová hodnota a cyklus vyhledávání je ukončen.

```
25          }
26          if (!nasel)
27          {
28              value.Zkratka = Zkratka;
29              this.Add(value);
```

Pokud nebyla hledaná položka nalezena, přidá se do kolekce nová.

```
30          }
```

Podobným způsobem jsou vytvořeny indexery u všech ostatních kolekcí tříd.

7.4.2 DAO

Komponenta CDAO slouží k přístupu k databázi a tvoří tak vlastně komunikační vrstvu mezi databází a samotnou aplikací. V komponentě CDAO jsou vytvořené funkce pro obsluhu uložených procedur databáze, ale jsou zde i funkce, které kontrolují, zda je vůbec možné s databází navázat spojení.

Následující ukázka zdrojového kódu provede načtení seznamu místností z databáze, které předává jako návratovou hodnotu.

```
1 public CMistnosti NactiMistnosti()
2 {
3     CMistnosti mistnosti = new CMistnosti();
```

Nejdřív se vytvoří objekt `mistnosti`, který slouží jako dočasné úložiště pro místnosti načtené z databáze.

```
4     using (SqlConnection con = new SqlConnection(connectionString))
5     {
6         try
7         {
8             con.Open();
9
10            SqlCommand cmd = new SqlCommand("NactiMistnosti", con);
11            cmd.CommandType = CommandType.StoredProcedure;
```

Vytvoří se nový SQL příkaz a jeho ty je nastaven na uloženou proceduru.

```
12            SqlDataReader reader = cmd.ExecuteReader();
```

Příkaz `ExecuteReader` spustí daný SQL příkaz.

```
13            while (reader.Read())
```

Cyklem `while` se projdou všechny záznamy výsledku dotazu.

```
14            {
```

```
15              CMistnost mistnost = new CMistnost(reader.GetString(0),
   reader.GetString(1), reader.GetString(2));
16              mistnosti.Add(mistnost);
```

Uvnitř cyklu se vytvoří nová instance třídy `CMistnost`, která je vzápětí přidána do pomocné kolekce `mistnosti`.

```
17          }
18      }
19      catch (Exception ex)
20      {
21          ChybovaHlaska(ex);
```

V případě, že během vykonávání příkazů uvnitř bloku `try { }` nastane výjimka, vykonávání programu se v daném místě přeruší a dále se pokračuje blokem `catch { }`. Zde se pouze zavolá funkce `ChybovaHlaska`, která zobrazí výpis chyby a uloží ji do logu aplikace.

```
22      }
23      finally
24      {
25          con.Close();
```

Po skončení práce s databází se uzavře spojení.

```
26      }
27  }
28  return mistnosti;
```

Na závěr je výsledná pomocná kolekce místností předána návratovou hodnotou jako výsledek funkce.

```
29 }
```

7.4.3 Com

Komponenta Com zastává veškeré činnosti komunikace a je rozdělena do dvou hlavních částí. Jedná se o komunikaci po sériové lince s řídicími jednotkami a komunikaci mezi aplikacemi.

Sériová komunikace

Pro komunikaci po sériové lince je možné zvolit libovolný sériový port dostupný v počítači a nastavit jeho přenosovou rychlost. Ta musí být shodná s přenosovou rychlostí nastavenou v řídicích jednotkách. Kromě možnosti nastavit tyto hodnoty jsem rovněž

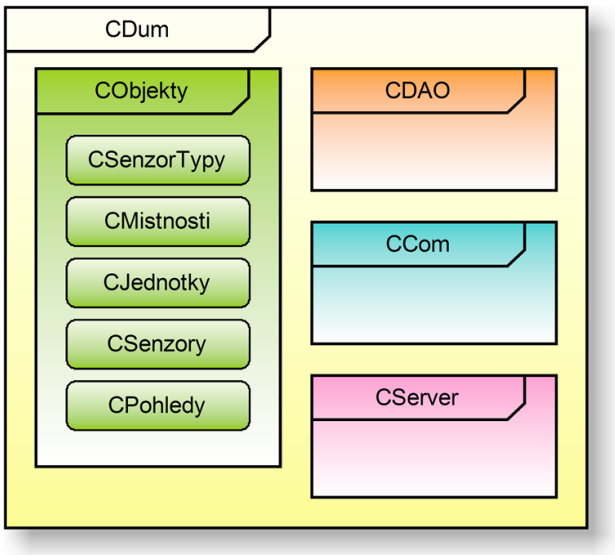
implementoval samotný komunikační protokol popsaný v kapitole 6.3, aby byla zajištěna správná komunikace s jednotkami.

Komunikace mezi aplikacemi

Komunikace mezi aplikacemi probíhá pomocí spojení přes sockety. Socket je koncový bod komunikace přes internetový protokol a je složen ze samotného protokolu (TCP, UDP…), místní IP adresy, místního portu, vzdálené IP adresy a vzdáleného portu. Tato komunikace je nutná pro přenos dat mezi webovou aplikací a samotnou řídicí aplikací, která přímo komunikuje s řídicími jednotkami. Je zde definována funkce serverové části, stejně jako klientské. Serverová část je určena pro běh uvnitř řídicí aplikace, klientská pak pro webovou aplikaci, která v případě potřeby odesílá na serverovou část dotazy s žádostí o různá data.

7.4.4 Hardware

Jak je znázorněno na obrázku (Obr. 16), komponenta Hardware zapouzdřuje všechny tři předchozí komponenty a vytváří z nich jeden samostatně fungující celek. Komponenty jsou do sebe zapouzdřeny podle obrázku (Obr. 21). Jsou zde uvedeny názvy pouze těch nejdůležitějších tříd.



Obr. 21. Zapouzdření komponent do jednoho funkčního celku

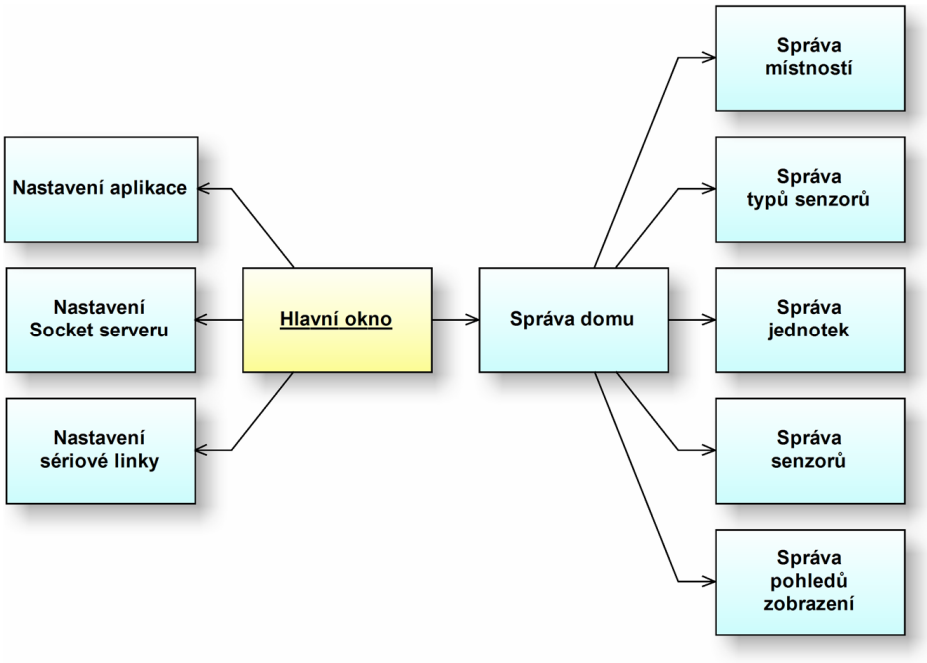
V komponentě Hardware jsem vytvořil třídu `CDum`, která obsahuje nejen jednotlivé třídy pro uchování veškeré struktury dat načítaných z databáze, ale také třídy pro přístup k datům a komunikaci.

7.5 Řídicí aplikace

Řídicí aplikaci jsem vytvořil v prostředí Microsoft Visual C# 2005 Express Edition. Řídicí aplikace je standardní Windows aplikace s grafickým rozhraním. Tato aplikace slouží jak k definici veškerých objektů v inteligentním domě, tak ke komunikaci, sběru dat a jejich archivaci. Neslouží ale k zobrazování aktuálních hodnot ani k jejich nastavování. Tuto činnost zastává aplikace pracující přes webové rozhraní, která je popsaná v kapitole 7.6.

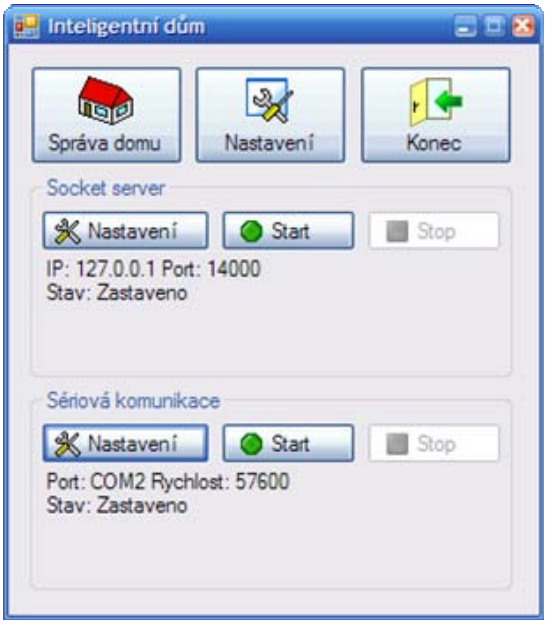
7.5.1 Struktura aplikace

Řídicí aplikace je z pohledu uživatele rozdělena do několika dialogových oken, pomocí nichž lze nastavovat samotnou aplikaci a spravovat systém inteligentního domu. Schematicky je struktura těchto dialogů znázorněna na obrázku (Obr. 22).



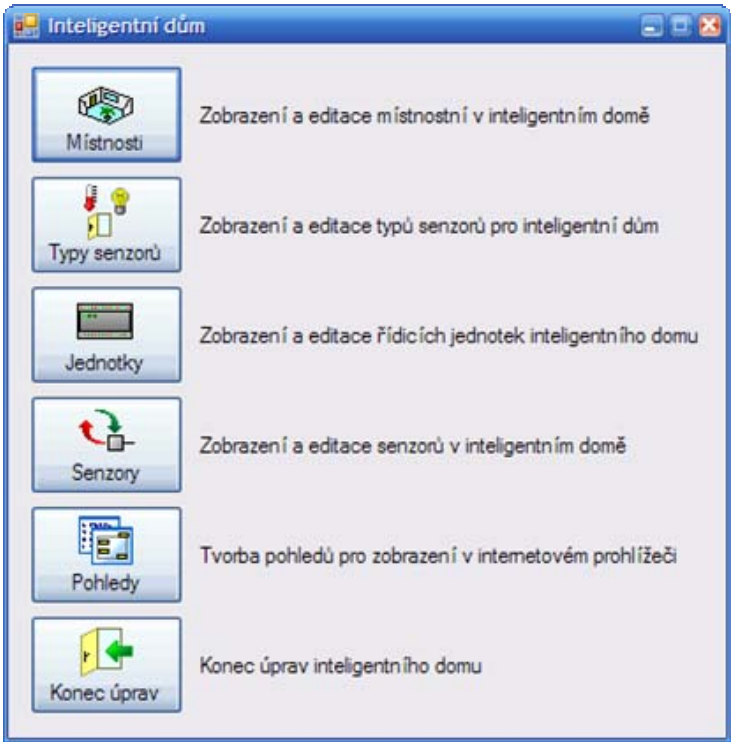
Obr. 22. Schematická struktura dialogů řídicí aplikace

Hlavní okno aplikace, které se zobrazí při spuštění programu, je znázorněno na obrázku (Obr. 23). V tomto okně není mnoho ovládacích prvků, protože k činnosti řídicí aplikace není třeba téměř žádný zásah ze strany uživatele.



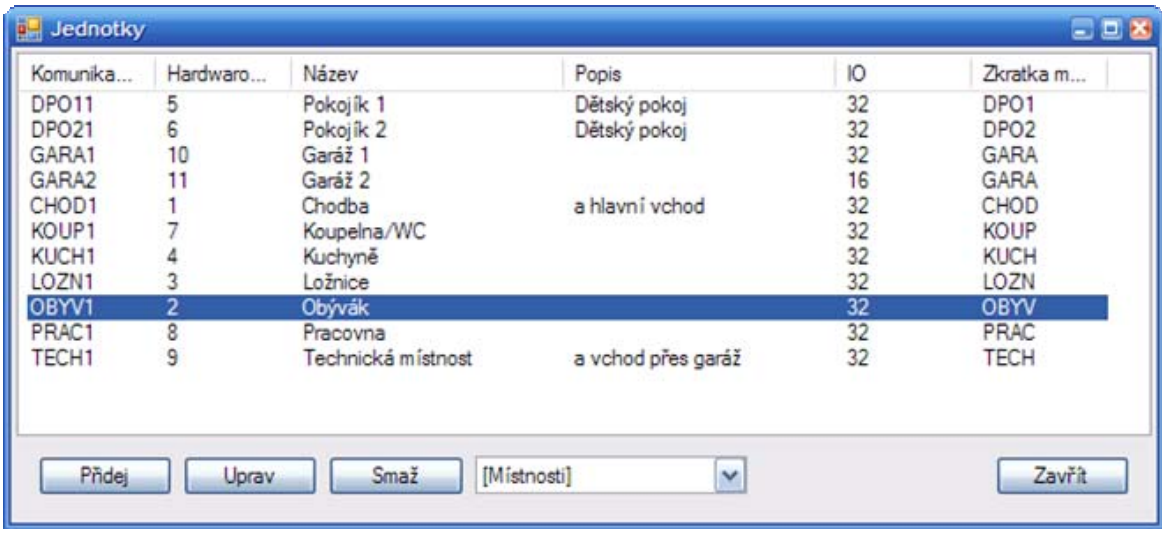
Obr. 23. Hlavní okno řídicí aplikace

Důležitým ovládacím prvkem je zde tlačítko „Správa domu“. Kliknutím na toto tlačítko se uživatel dostane do okna, které slouží v podstatě jako rozcestník při správě objektů popisujících strukturu celého domu. Toto okno je na obrázku (Obr. 24).



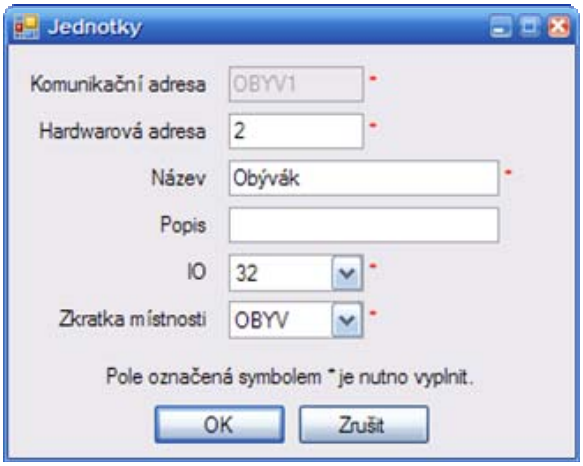
Obr. 24. Hlavní okno správy inteligentního domu

Zde si uživatel může zvolit, zda bude editovat místnosti, typy senzorů, řídicí jednotky, samotné senzory nebo pohledy zobrazení. Pokud si uživatel zvolí například editaci řídicích jednotek, zobrazí se dialog znázorněný na obrázku (Obr. 25).



Obr. 25. Seznam řídicích jednotek

Je zde zobrazen seznam všech definovaných řídicích jednotek včetně jejich nastavených parametrů. Řídicí jednotky je zde možné přidávat, mazat, ale také upravovat. Dialog pro vytvoření nové jednotky je stejný, jako dialog pro její úpravu. Znázorněn je na obrázku (Obr. 26).

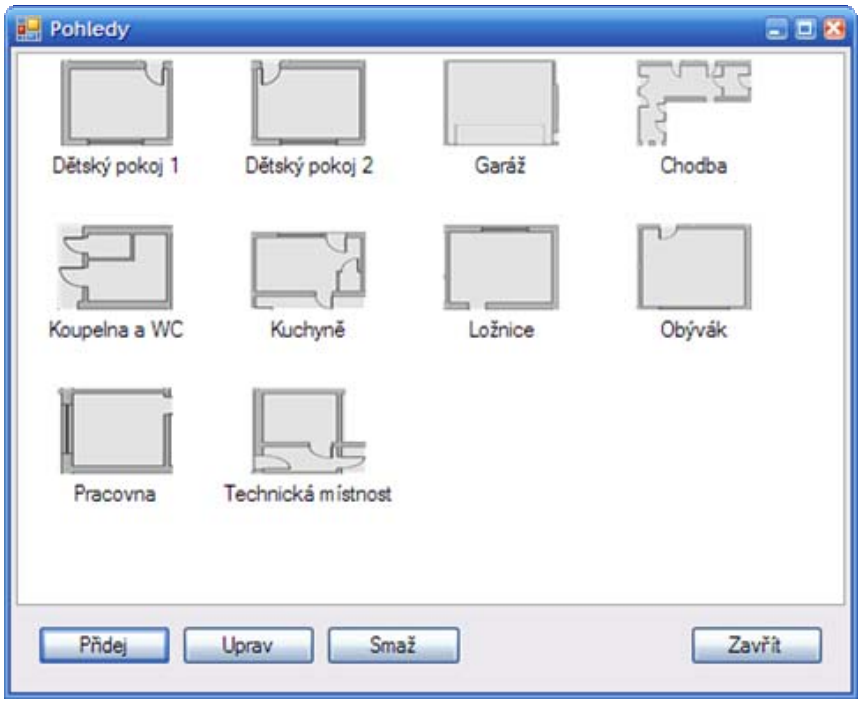


Obr. 26. Dialog pro editaci řídicí jednotky

V případě úpravy jednotky je pole „Komunikační adresa“ zakázané (zašednuté), aby jej nebylo možné upravovat. Je to z toho důvodu, že komunikační adresa je brána jako identifikátor jednotky a v případě, kdy jsou na této jednotce závislé nějaké senzory, by mohla změna komunikační adresy jednotky způsobit přerušení vazeb mezi jednotkou a

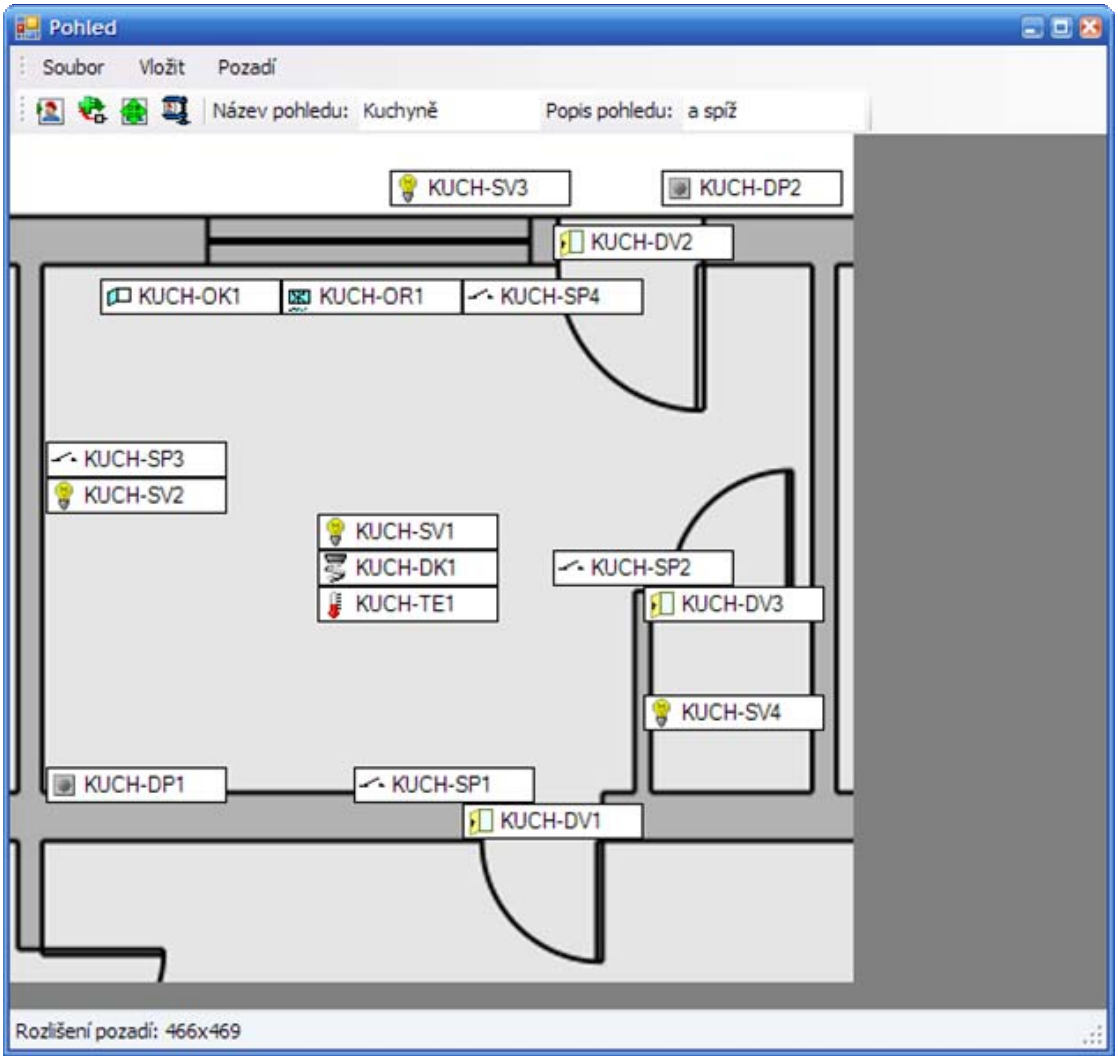
senzory. Komunikační adresu je tedy možné nastavit pouze při vytvoření nové řídicí jednotky.

Seznam pohledů zobrazení, kam se lze rovněž dostat z hlavního okna správy inteligentního domu, je na rozdíl od všech ostatních řešen grafickou formou. Jedná se totiž o grafické pohledy zobrazení pro webovou aplikaci a v této formě je jejich seznam přehlednější. Okno se seznamem pohledů zobrazení je na obrázku (Obr. 27).



Obr. 27. Seznam pohledů zobrazení

Samozřejmě i zde je možnost přidat nový pohled zobrazení, smazat nebo upravit již existující. Jak vypadá dialog pro tvorbu a editaci pohledů zobrazení, je vidět na obrázku (Obr. 28). Nejprve se zvolí obrázek na pozadí, který umožňuje zpříjemnit vzhled pohledu zobrazení a zároveň definuje jeho rozměry. Poté je do obrázku možné vkládat libovolné senzory a posunovat je po obrázku podle potřeby. V jednom pohledu zobrazení lze mít vložené libovolné senzory z jakékoliv místnosti. Nemá ani vliv, na kterou řídicí jednotku je senzor fyzicky připojen. Je tedy možné vytvořit nejen pohled reprezentující konkrétní místnost, ale například přehledovou mapku se zabezpečovacími prvky, kde lze sledovat pouze stav otevření oken, dveří, pohybových čidel nebo požárních hlásičů.



Obr. 28. Editace pohledu zobrazení

Kromě samotného vkládání objektů, reprezentujících jednotlivé senzory, je možné těmto objektům nastavit různé barevné zvýraznění podle toho, jaké hodnoty nabývá daný senzor. Uživatel si tak může nastavit, že při detekci požáru bude požární hlásič zobrazen červeně a v klidu naopak zeleně. Podobně lze nastavit i například teplotní čidlo, kde se mohou zvolit různé barvy pro různé rozsahy teplot. Pro teploty pod nulou to může být modrá barva, pro teploty do 25°C žlutá a pro vyšší teploty oranžová. Vše záleží pouze na vkusu uživatele.

7.5.2 Sériová komunikace

Ke komunikaci s řídicími jednotkami přes sériovou linku program využívá sdílenou komponentu Com popsanou v kapitole 7.4.3.

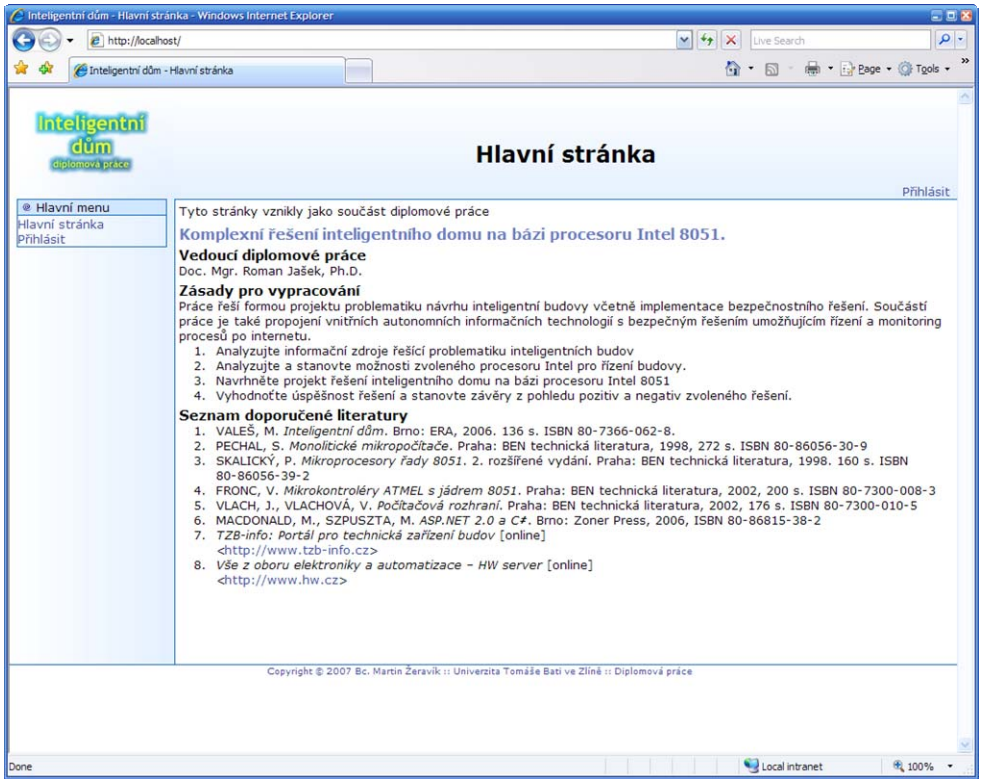
7.5.3 Síťová komunikace

Síťová komunikace, stejně jako sériová, je rovněž zajištěna sdílenou komponentou Com. Komunikace probíhá pomocí spojení přes sockety a může probíhat nejen přes síť internet nebo po místní síti, ale i v rámci jednoho počítače, čehož můj informační systém využívá.

7.6 Internetová aplikace

Internetovou aplikaci jsem vytvořil s využitím technologie ASP.NET 2.0 a programovacího jazyka C# 2.0 ve volně dostupném vývojovém prostředí Microsoft Visual Web Developer 2005 Express Edition.

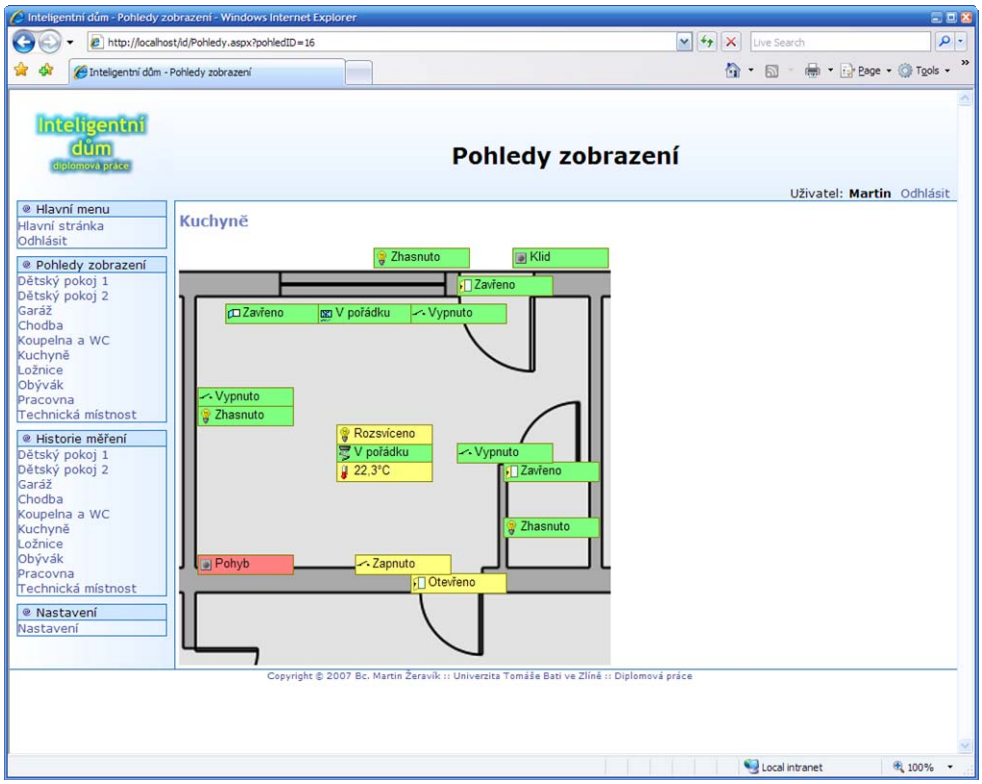
Aplikace je určena k zobrazování aktuálních hodnot senzorů domu, nastavování jejich hodnot a zobrazování historie uložených záznamů v podobě grafu. Hlavní stránka, která se zobrazí při navštívení této webové aplikace, je na obrázku (Obr. 29).



Obr. 29. Hlavní strana webové aplikace informačního systému

V této podobě se stránka zobrazí každému anonymnímu uživateli, který ji navštíví a to včetně menu na levé straně stránky, které je omezeno na minimum položek. Po přihlášení uživatele se zobrazí další volby v menu, kterými je možné přecházet mezi nejrůznějšími částmi aplikace, jako je sledování aktuálních hodnot senzorů, zobrazení jejich historie, ale

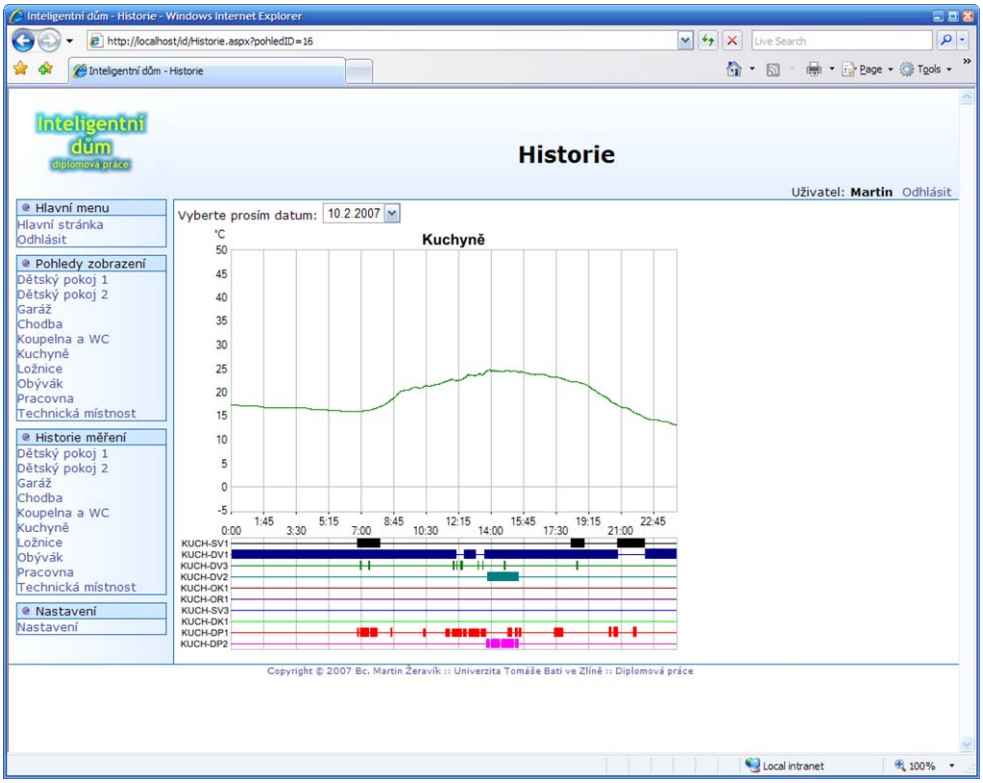
také nastavení samotné aplikace. Ukázka, jak vypadá stránka pro přihlášeného uživatele, je na obrázku (Obr. 30).



Obr. 30. Webová aplikace z pohledu přihlášeného uživatele

Na stránce, která se zobrazí přihlášenému uživateli, jsou kromě základních položek v menu zobrazeny i seznamy vytvořených pohledů zobrazení a uživatel tak má možnost zobrazit si aktuální data pro vybranou místnost.

Kromě aktuálních dat je možné zobrazit i historii měřených dat v podobě grafu, který je vykreslován podle požadavků uživatele. Stačí vybrat datum, pro které se má graf vykreslit a uživatel následně uvidí graf, který může vypadat tak jako na obrázku (Obr. 31).



Obr. 31. Historie měřených dat inteligentního domu

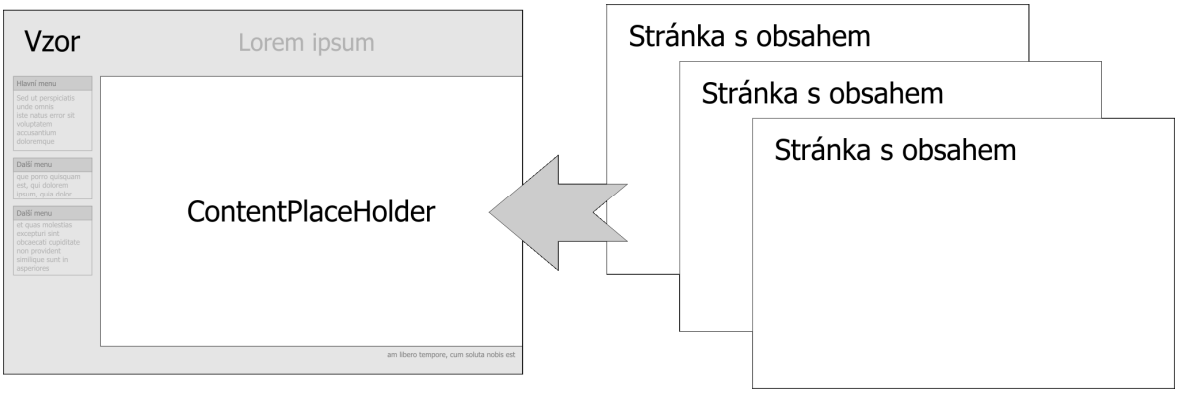
7.6.1 Design webové aplikace

Design webové aplikace jsem vytvořil pomocí takzvaných vzorů stránek. Aplikaci v podstatě tvoří dva typy stránek, a to stránka se vzorem a stránky s obsahem.

Stránka se vzorem, v tomto případě soubor `Zakladni.Master`, je šablona stránky, v níž jsou definovány oblasti, kam má být umístěn například konkrétní text, navigační prvky nebo samotný obsah stránky. Opakující se část, která se zobrazuje ve všech stránkách, je v mém případě záhlaví stránky včetně loga a navigační menu na levé straně. Obsah stránky je ve vzorové stránce reprezentován objektem ContentPlaceHolder.

Ve stránce s obsahem je pak uvedeno, kterou vzorovou stránku přebírá a jaký obsah bude zobrazen uvnitř konkrétních objektů typu ContentPlaceHolder.

Jakým způsobem fungují vzorové stránky je naznačeno na obrázku (Obr. 32).



Obr. 32. Princip vzorových stránek

Výsledkem činnosti vzorových stránek je to, že ať se uživatel pohybuje po jakékoliv stránce, mění se pouze obsah v objektech ContentPlaceHolder definovaných ve vzorové stránce a ostatní věci zůstávají stejné.

7.6.2 Rozmístění souborů webové aplikace

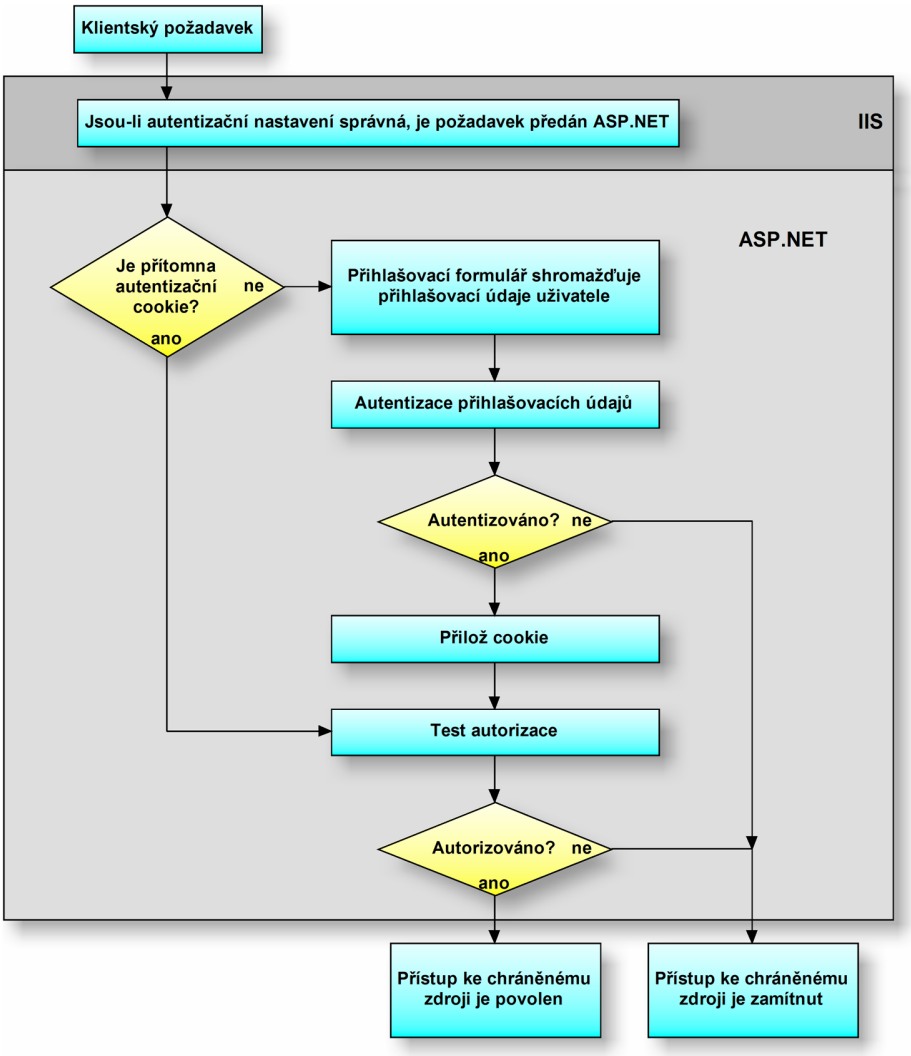
Všechny soubory, náležící webové aplikaci, které jsou přístupné všem uživatelům, se nacházejí v kořenovém adresáři aplikace, který je označován jako „~“. Jsou to nejen samotné zdrojové soubory aplikace, ale také soubory s kaskádovými styly nebo motivy zobrazení.

V podadresáři „~\ID“ se pak nacházejí soubory určené k samotnému sledování a ovládání inteligentního domu. Tento adresář je zabezpečen proti přístupu anonymních uživatelů a je přístupný pouze pro přihlášené uživatele.

7.6.3 Zabezpečení

Veškerá data, týkající se informací o inteligentním domě, jsou přístupná pouze po přihlášení uživatele zadáním uživatelského jména a hesla. Anonymní uživatel může vidět pouze obecné informace, které přímo nemusí souviset se samotným inteligentním domem. K ověřování uživatele využívám formulářovou autentizaci, která pracuje na principu lístků (tickets). Uživatel při přihlášení dostane lístek se základními uživatelskými informacemi. Informace jsou uloženy v šifrované cookie, která je připojována k odpovědi a která je automaticky nabídnuta při každém následujícím požadavku. Pokud uživatel požaduje chráněnou stránku, která je pro anonymního uživatele nepřístupná, runtime ASP.NET ověřuje, zda je k dispozici lístek formulářové autentizace. Pokud není, ASP.NET uživatele

automaticky přesměruje na přihlašovací stránku. Celý tento proces je znázorněn na obrázku (Obr. 33).



Obr. 33. Proces formulářové autentizace

Aby ASP.NET aplikace věděla, že má pro ověřování uživatele používat formulářovou autentizaci, musí být v konfiguračním souboru web.config uvedeny následující řádky:

```
1 <authentication mode="Forms">
2     <forms name="MyCookieName" loginUrl="Login.aspx"
  defaultUrl="Default.aspx" timeout="20" cookieless="AutoDetect">
3 </forms></authentication>
```

V nastavení `forms` jsou upřesňující nastavení pro formulářovou autentizaci. Parametr `loginUrl` určuje stránku, na kterou je uživatel přesměrován při žádosti o přihlášení. Parametr `defaultUrl` naopak určuje stránku, kam je přihlášený uživatel přesměrován při odhlášení.

Dalším důležitým nastavením je omezení přístupu uživatelů. V sekci `authorization` je povolen přístup všem uživatelům pomocí volby `allow users="*"`.

```
1 <authorization>
2     <allow users="*" />
3 </authorization>
```

Tím je zajištěno, že všechny soubory v aktuálním adresáři jsou přístupné nejen přihlášeným uživatelům, ale i anonymním. Tito uživatelé ale mají také přístup do všech podadresářů aplikace, které nejsou svým vlastním souborem `web.config` nastaveny jinak.

Jak již bylo dříve zmíněno, podadresář „~\ID“ má nastaveno oprávnění tak, aby byl přístupný pouze pro přihlášené uživatele. Toto nastavení je provedeno tak, že uvnitř adresáře je samostatný soubor `web.config`, v němž je v sekci `authorization` následující:

```
1 <authorization>
2     <deny users="?"/>
3 </authorization>
```

Volba `deny users="?"` zakáže přístup neznámým, tedy nepřihlášeným uživatelům. Při pokusu anonymního uživatele o přístup k chráněnému souboru je automaticky zjištěno nastavení formulářové autentizace a uživatel je přesměrován na přihlašovací stránku, v tomto případě `Login.aspx`.

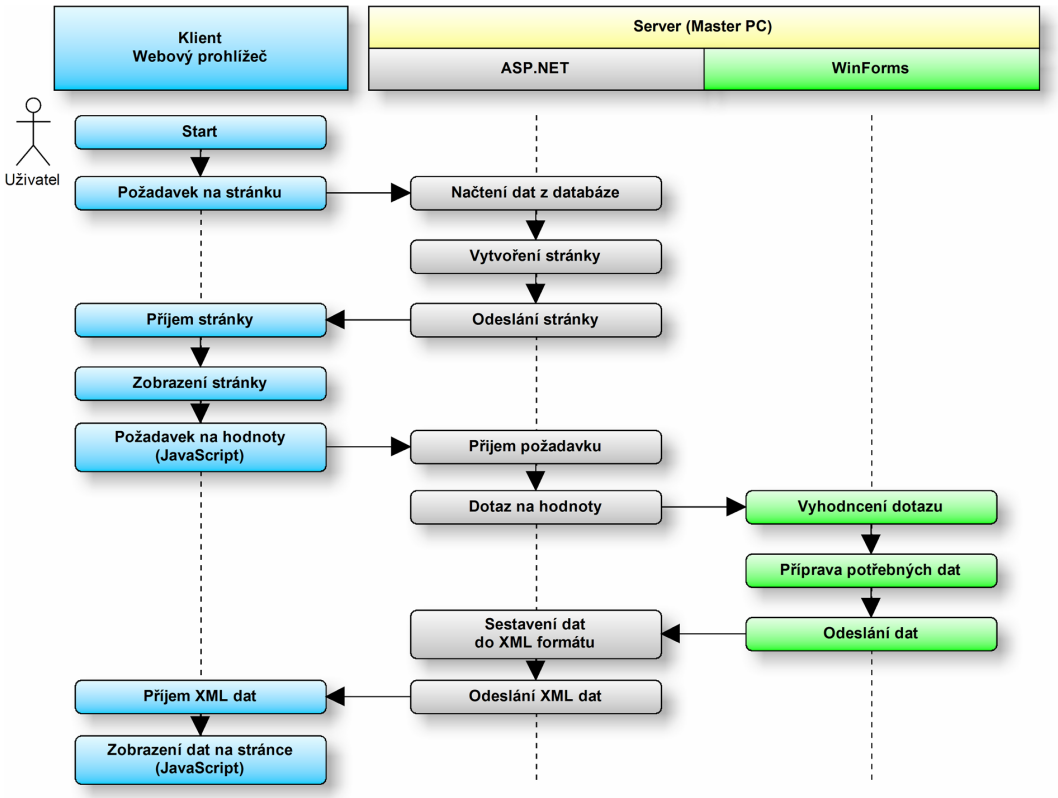
Vytváření uživatelských účtů

Při volbě způsobu vytváření uživatelských účtů jsem vycházel z toho, že přístup k webové aplikaci bude umožněn pouze pro vybrané obyvatele domu. Počet uživatelů i jejich jména jsou tedy předem známé. Z tohoto důvodu jsem k vytvoření uživatelských účtů využil nástroj integrovaný v ASP.NET, jímž je možné účty spravovat. Tuto správu lze provádět pouze na samotném serveru, na němž běží webová aplikace. Případný záškodník, který tuto aplikaci navštíví přes internet, nemá možnost si založit svůj vlastní účet a jeho prostřednictvím páchat jakékoliv škody.

7.7 Komunikace mezi aplikacemi

Aby bylo možné ve webové aplikaci zobrazovat aktuální hodnoty právě přečtené řídicí aplikací, je potřeba zajistit mezi těmito aplikacemi přenos dat. Ten probíhá pomocí spojení přes sockety na TCP/IP protokolu a s využitím technologie AJAX. Zkratka AJAX doslova znamená Asynchronous JavaScript And XML. Pomocí této technologie lze měnit obsah

vybraných částí internetové stránky bez nutnosti znovunačtení celé stránky, čímž se sníží objem přenášených dat a zvýší se rychlost odezvy webové aplikace. Princip této technologie je takový, že po načtení a zobrazení stránky se u klienta spustí JavaScript, který si vyžádá ze serveru konkrétní informace. Server tyto informace pošle v podobě XML dat. JavaScript na klientské stránce tyto data načte a zobrazí je do patřičných částí stránky. Celý tento proces je znázorněn na obrázku (Obr. 34).



Obr. 34. Přenos hodnot senzorů pomocí technologie AJAX

V XML datech přenášených mezi aplikacemi jsou pouze údaje nezbytné pro jejich správné zobrazení. Důvod je ten, aby zbytečně nedocházelo k přenosům velkých objemů dat. V těchto datech je pro každý senzor uvedena jeho komunikační adresa, podle které je na webové stránce daný senzor identifikován, název senzoru, jeho aktuální hodnota a barva pozadí. Následující příklad demonstruje část přenášených dat:

```
1 <data>
2     <Senzor>
3         <AdresaCom>KUCH-DV1</AdresaCom>
4         <Nazev>Dveře do kuchyně</Nazev>
5         <Hodnota>Otevřeno</Hodnota>
6         <Pozadi>#FFFF80</Pozadi>
```

```
 7      </Senzor>
 8      <Senzor>
 9          <AdresaCom>KUCH-SP1</AdresaCom>
10          <Nazev>Spínač světla</Nazev>
11          <Hodnota>Zapnuto</Hodnota>
12          <Pozadi>#FFFF80</Pozadi>
13      </Senzor>
14      ...
15  </data>
```

Data v tomto formátu jsou přenášena při každém požadavku klientova prohlížeče na zobrazení aktuálních hodnot. Po načtení stránky u klienta se spustí JavaScript, v němž je pomocí časovače tato činnost naprogramována automaticky. Uživatel se tak nemusí o nic starat a v pravidelných intervalech, v řádech sekund, dostává aktuální měřená data z Master PC uvnitř inteligentního domu. Tento proces funguje naprosto totožně pro uživatele zobrazujícího webovou aplikaci přes internet i pro uživatele na lokální síti uvnitř domu.

ZÁVĚR

Tato práce je zaměřena na vytvoření komplexního řešení inteligentního domu založeného na mikroprocesoru s jádrem Intel 8051. Je zde obecně popsána podstata inteligentního domu jako takového a jsou zde nastíněny možnosti existujících systémů pro inteligentní domy. Hlavní náplní této práce je ale vytvoření vlastního systému pro inteligentní dům založeného na zmíněném mikroprocesoru.

V práci je popsáno mnou navržené řešení, které je založeno na samostatně fungujících řídicích jednotkách osazených mikroprocesorem 8051, které dále komunikují s nadřazeným počítačem typu PC. V tomto počítači probíhá sběr dat všech zařízení uvnitř domu, veškeré rozhodovací operace týkající se činnosti domu, ale také archivace měřených dat a přístup k těmto datům z místní počítačové sítě a ze sítě internet. V souvislosti s tím jsou zde popsány možnosti zabezpečení počítačové sítě především proti útoku z internetu. Důležitým krokem je zde také volba softwarové platformy, na které je postaven celý informační systém běžící na hlavním PC. Zvolil jsem platformu .NET od společnosti Microsoft, protože tato platforma nabízí nejen vysokou úroveň zabezpečení, ale také řadu kvalitních vývojových nástrojů, díky nimž lze snadno vytvořit požadovanou aplikaci.

V praktické části práce je popsáno technické řešení mého návrhu. Nejprve je zde popsán modelový dům, který jsem navrhl pro demonstraci mého řešení. Pak jsou zde elektronická schémata obvodů řídicích jednotek včetně obecného popisu jejich činnosti. Poslední kapitola je věnována samotnému softwarovému řešení, které je nezbytné pro činnost celého systému. Je zde uveden popis programu, který obsluhuje činnosti řídicích jednotek, ale také celý informační systém běžící na hlavním počítači. Ten je rozdělen do tří velkých celků. Prvním je databáze, která slouží k definici závislostí a vazeb mezi jednotlivými zařízeními v domě a také k archivaci jejich měřených hodnot. Druhou částí je řídicí aplikace, pomocí níž lze tyto závislosti a vazby upravovat a zároveň tato aplikace komunikuje se samotnými řídicími jednotkami. Komunikace probíhá po sériové lince za pomoci protokolu, který jsem speciálně pro tento účel navrhl. Poslední, třetí částí informačního systému, je internetová aplikace sloužící ke komunikace systému s uživatelem. Komunikace probíhá přes internetový prohlížeč nejen ze sítě internet, ale také v rámci místní počítačové sítě přímo z domu.

K diplomové práci jsou přiloženy kompletní zdrojové kódy všech zde popsaných aplikací.

CONCLUSION

This thesis is focused on complex smart house solution based on microprocessor with Intel 8051 core. There is described principle of smart house in itself and there're outlined possibilities of existing systems for smart houses. Main content of this thesis is to create my own system for smart house based on mentioned microprocessor.

In this thesis there is described my designed of solution, which is based on independently functional control units with 8051 microprocessor, which communicate with superior PC computer. This computer performs data collection from all devices inside house, all deciding operations concerning to house activities, archiving of measured data and access to this data from local computer network and from the internet. In context of that there're described security possibilities of computer network firstly against attack from the internet. Important step is software platform choice, on which is built whole informational system running on master PC. I choose the .NET platform from Microsoft because this platform offers not only high level of security, but also number of quality development tools, which allow creating required application easily.

In practical parts of thesis there is described technical solution of my project. At first there is described model house which I designed for demonstration of my solution. Next there're electronic schemes of control unit circuits including common description of their function. Last chapter is dedicated to software solution which is necessary for function of whole system. There is described control unit program, which provides their function, but also whole information system running on a master computer. It is divided into three big parts. First part is database serves for definition of dependences and links among individual devices in house and also for archiving of their measured values. Second part is control application, which manages these dependencies and links and also this application communicates with control units. Communication between computer and control units is serial communication based on protocol specially designed for this purposes. Last, third part of information system, is internet application for communication between information system and user. Communication is realized through internet browser not only from the internet, but also from local computer network directly in house.

The diploma thesis appendix contains complete source codes of all applications.

SEZNAM POUŽITÉ LITERATURY

[1] VALEŠ, M. *Inteligentní dům*. Brno: ERA, 2006. 136 s. ISBN 80-7366-062-8.

[2] PECHAL, S. *Monolitické mikropočítače*. Praha: BEN technická literatura, 1998, 272 s. ISBN 80-86056-30-9

[3] SKALICKÝ, P. *Mikroprocesory řady 8051*. 2. rozšířené vydání. Praha: BEN technická literatura, 1998. 160 s. ISBN 80-86056-39-2

[4] FRONC, V. *Mikrokontroléry ATMEL s jádrem 8051*. Praha: BEN technická literatura, 2002, 200 s. ISBN 80-7300-008-3

[5] VLACH, J., VLACHOVÁ, V. *Počítačová rozhraní*. Praha: BEN technická literatura, 2002, 176 s. ISBN 80-7300-010-5

[6] MACDONALD, M., SZPUSZTA, M. *ASP.NET 2.0 a C#*. Brno: Zoner Press, 2006, ISBN 80-86815-38-2

[7] *TZB-info: Portál pro technická zařízení budov* [online]

<http://www.tzb-info.cz>

[8] *Vše z oboru elektroniky a automatizace – HW server* [online]

<http://www.hw.cz>

[9] Czech Computer [online]

<http://www.czechcomputer.cz>

[10] Wikipedie, otevřená encyklopedie [online]

<http://cs.wikipedia.org>

[11] Wikipedia, the free encyclopedia [online]

<http://en.wikipedia.org>

[12] IQdum.cz: Inteligentní dům [online]

<http://www.iqdum.cz>

[13] Orbit Merret [online]

<http://www.merret.cz>

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

AJAX Asynchronous JavaScript And XML – technologie pro vývoj interaktivních webových aplikací

ANSI American National Standards Institute – americká standardizační organizace

API Application Programming Interface – sada procedur, funkcí či tříd knihovny programu nebo jádra operačního systému, které může programátor využít.

ASP Aktive Server Pages – technologie pro tvorbu internetových stránek

CLR Common Language Runtime – běhové prostředí pro .NET aplikace

DAO Data Access Object – objekt (programová knihovna) pro přístup k datům

DCL Data Control Language – jazyk pro ovládání dat

DDL Data Definition Language – jazyk pro definici dat

DIL Dual Inline – patice pro integrované obvody, kde jsou vývody ve dvou řadách

DLL Dynamic Linking Library – knihovna (sbírka procedur, funkcí a datových typů) poskytující služby pro programy, kterou lze spojit s programem za chodu programu

DML Data Manipulation Language – jazyk pro manipulaci s daty

ECMA European Computer Manufacturers Association – mezinárodní organizace pro normalizaci

EEPROM Electrically Erasable Programmable Random Access Memory - Elektricky mazatelná programovatelná paměť

EPROM Electrically Programmable Random Access Memory - Elektricky programovatelná paměť

HTML Hypertext Markup Language – značkovací jazyk pro tvorbu internetových stránek

IA64 Intel Architecture-64 - 64bitová architektura procesoru

ISO International Organization for Standardization – mezinárodní organizace pro normalizaci

MCU	Microcontroller Unit – mikroprocesorová jednotka
OS	Operating Systém - Operační systém
OTP	One-Time Programmable – paměť programovatelná pouze jednou
PLCC	Plastic leaded chip carrier – patice integrovaných obvodů se specifickými rozměry
PQFP	Plastic quad flat pack – patice integrovaných obvodů se specifickými rozměry ve tvaru čtverce
RAM	Random Access Memory – paměť s náhodným přístupem
SDK	Software Development Kit – nástroje pro vývoj aplikací
SEQUEL	Structured English Query Language – předchůdce dnešního SQL
SQL	Structured Query Language – dotazovací jazyk pro práci s daty v relačních databázích
TCC	Transaction Control Commands – jazyk pro ovládání transakcí
TQFP	Thin quad flat pack – patice integrovaných obvodů se specifickými rozměry ve tvaru čtverce
x64	64bitové rozšíření 32bitové architektury procesoru x86
XML	eXtensible Markup Language – obecný značkovací jazyk pro snadné vytváření značkovacích jazyků a široké spektrum různých typů dat.

SEZNAM OBRÁZKŮ

Obr. 1. Vnitřní bloková struktura procesoru 8051 14
Obr. 2. Rozložení vnitřní paměti RAM procesoru 8051 15
Obr. 3. Rozložení vývodů mikroprocesoru AT89C51 18
Obr. 4. Rozložení vývodů mikroprocesoru AT89C2051 19
Obr. 5. Schéma komunikace jednotek a PC 21
Obr. 6. Základní deska VIA EPIA SP13000 21
Obr. 7. Server jako firewall 25
Obr. 8. Hardwarový firewall 26
Obr. 9. Využití vstupů a výstupů obou variant mikroprocesorů 36
Obr. 10. Napájení řídicí jednotky 36
Obr. 11. Optické oddělení vstupu 37
Obr. 12. Spínaní výstupu pomocí triaku 37
Obr. 13. Převod sériové linky mikroprocesoru na linku RS485 38
Obr. 14. Komunikace mezi Master PC a řídicí jednotkou 39
Obr. 15. Struktura softwarového řešení 41
Obr. 16. Logická struktura informačního systému 42
Obr. 17. Obecný princip programu řídicí jednotky 43
Obr. 18. Základní struktura databáze 44
Obr. 19. Tabulka pro uložení pohledů zobrazení 45
Obr. 20. Klíčové třídy sdílené komponenty Třídy 51
Obr. 21. Zapouzdření komponent do jednoho funkčního celku 55
Obr. 22. Schematická struktura dialogů řídicí aplikace 56
Obr. 23. Hlavní okno řídicí aplikace 57
Obr. 24. Hlavní okno správy inteligentního domu 57
Obr. 25. Seznam řídicích jednotek 58
Obr. 26. Dialog pro editaci řídicí jednotky 58
Obr. 27. Seznam pohledů zobrazení 59
Obr. 28. Editace pohledu zobrazení 60
Obr. 29. Hlavní strana webové aplikace informačního systému 61
Obr. 30. Webová aplikace z pohledu přihlášeného uživatele 62
Obr. 31. Historie měřených dat inteligentního domu 63
Obr. 32. Princip vzorových stránek 64

Obr. 33. Proces formulářové autentizace 65
Obr. 34. Přenos hodnot senzorů pomocí technologie AJAX 67

SEZNAM TABULEK

Tab. 1. Komunikační režimy procesoru 8051 17
Tab. 2. Význam symbolů senzorů 34
Tab. 3. Počet a typ vstupů a výstupů řídicích jednotek 35
Tab. 4. Uložené procedury pro vkládání dat 47
Tab. 5. Uložené procedury pro mazání dat 48
Tab. 6. Uložené procedury pro načtení dat 48
Tab. 7. Pomocné uložené procedury 49

SEZNAM PŘÍLOH

P I Modelový rodinný dům

P II Umístění senzorů v domě

P III Rozdělení místností pro řídicí jednotky

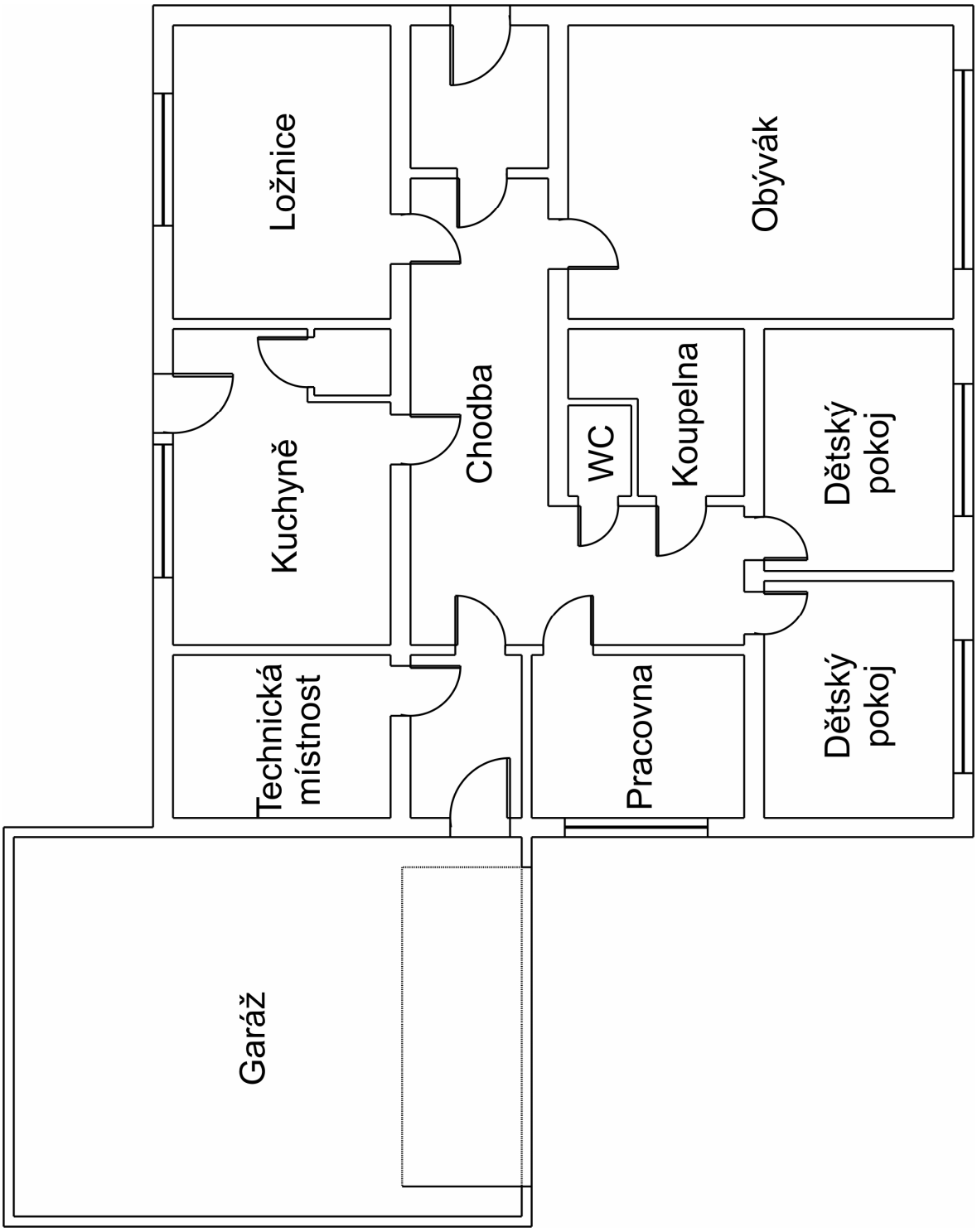
P IV Řídicí jednotka s procesorem AT89C51

P V Řídicí jednotka s procesorem AT89C2051

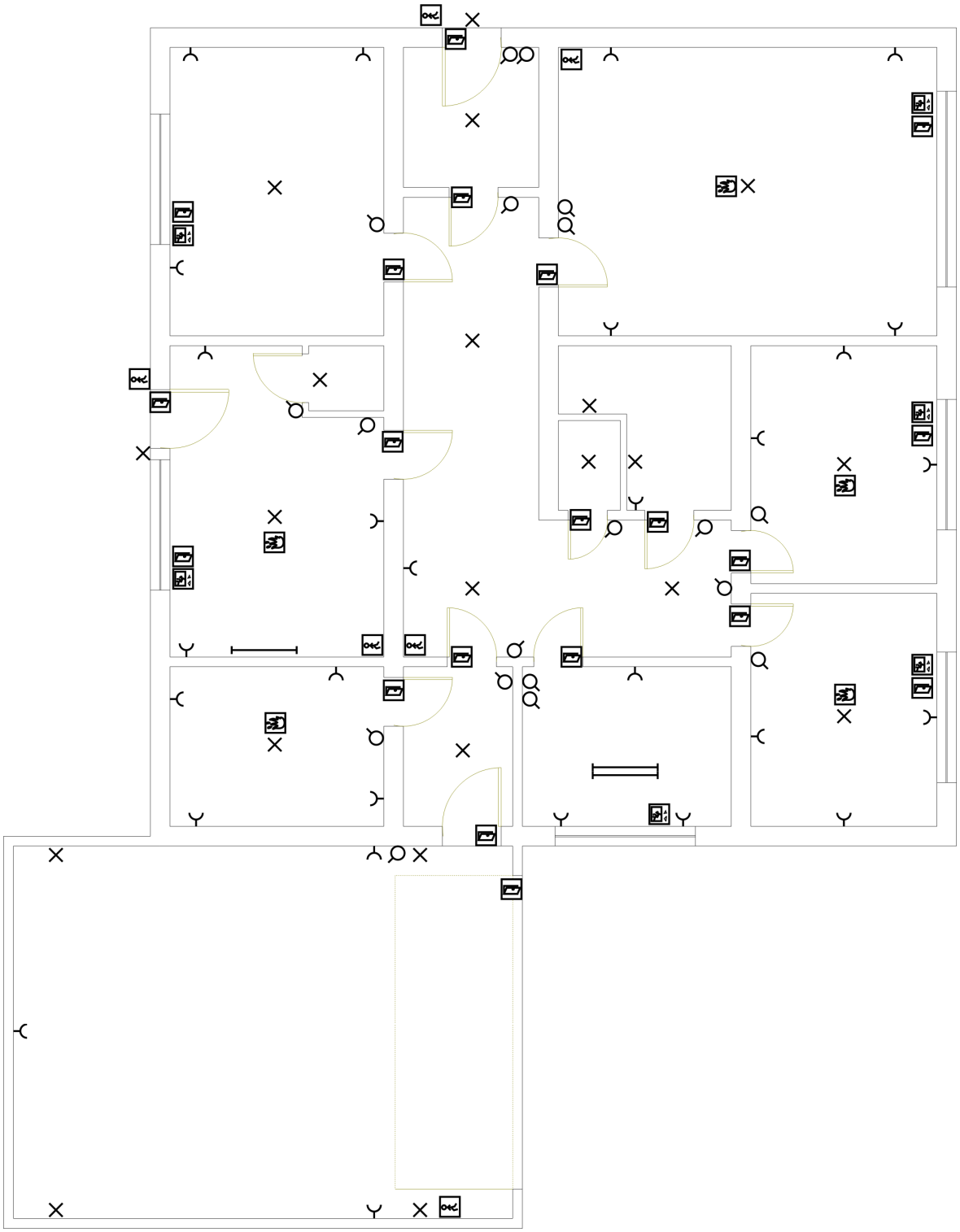
P VI Komunikační protokol

P VII CD-ROM

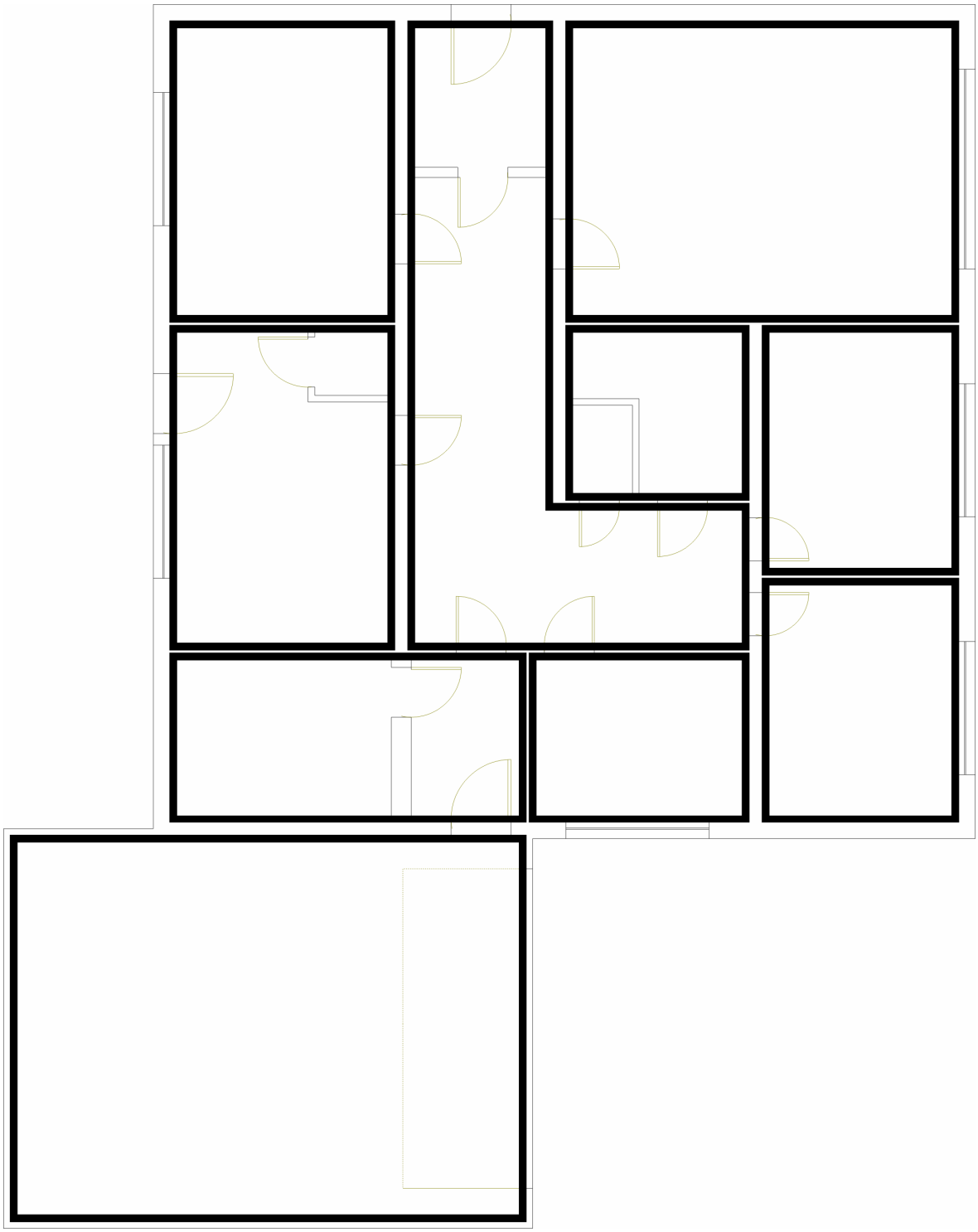
PŘÍLOHA P I: MODELOVÝ RODINNÝ DŮM



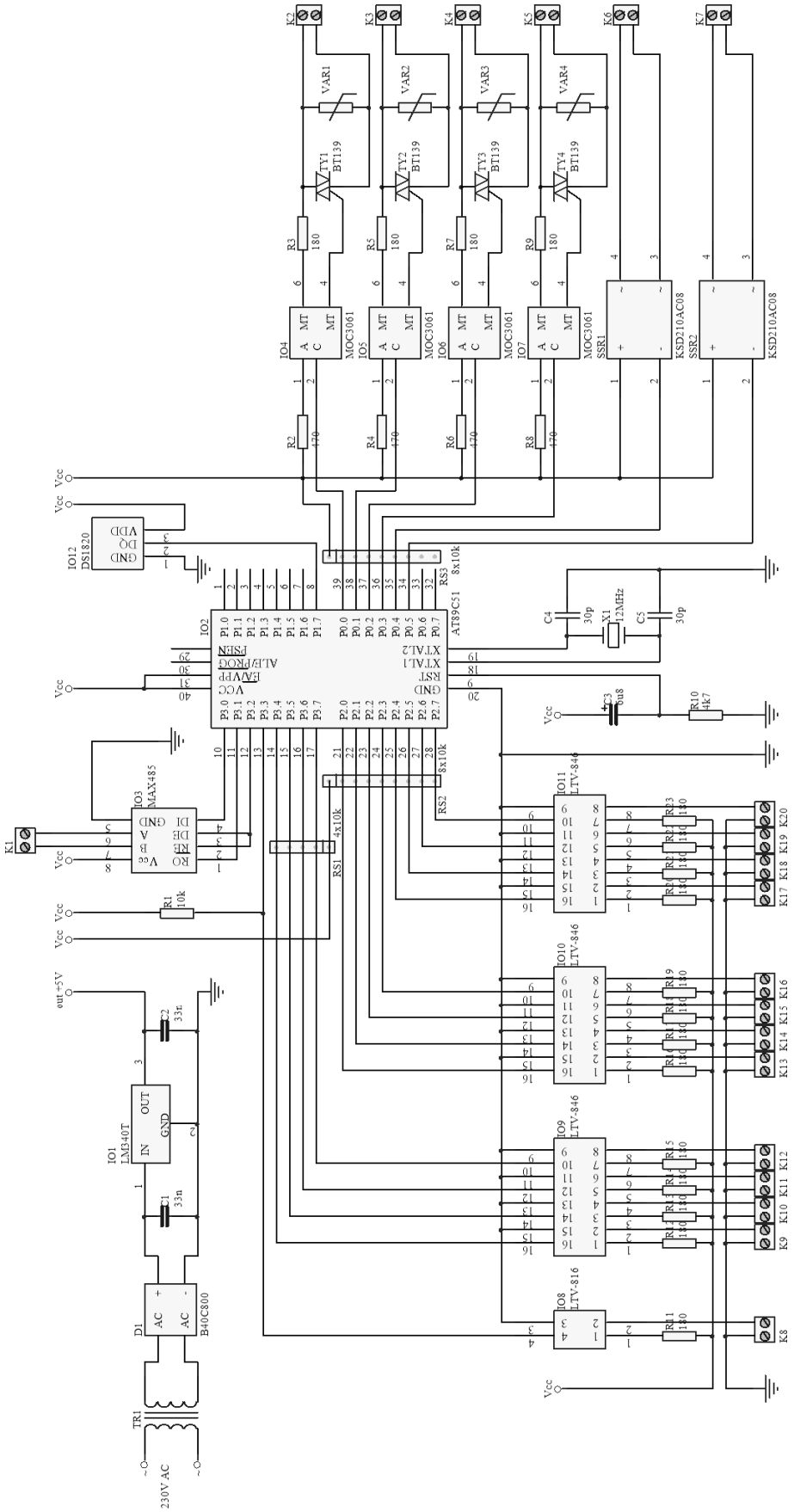
PŘÍLOHA P II: UMÍSTĚNÍ SENZORŮ V DOMĚ



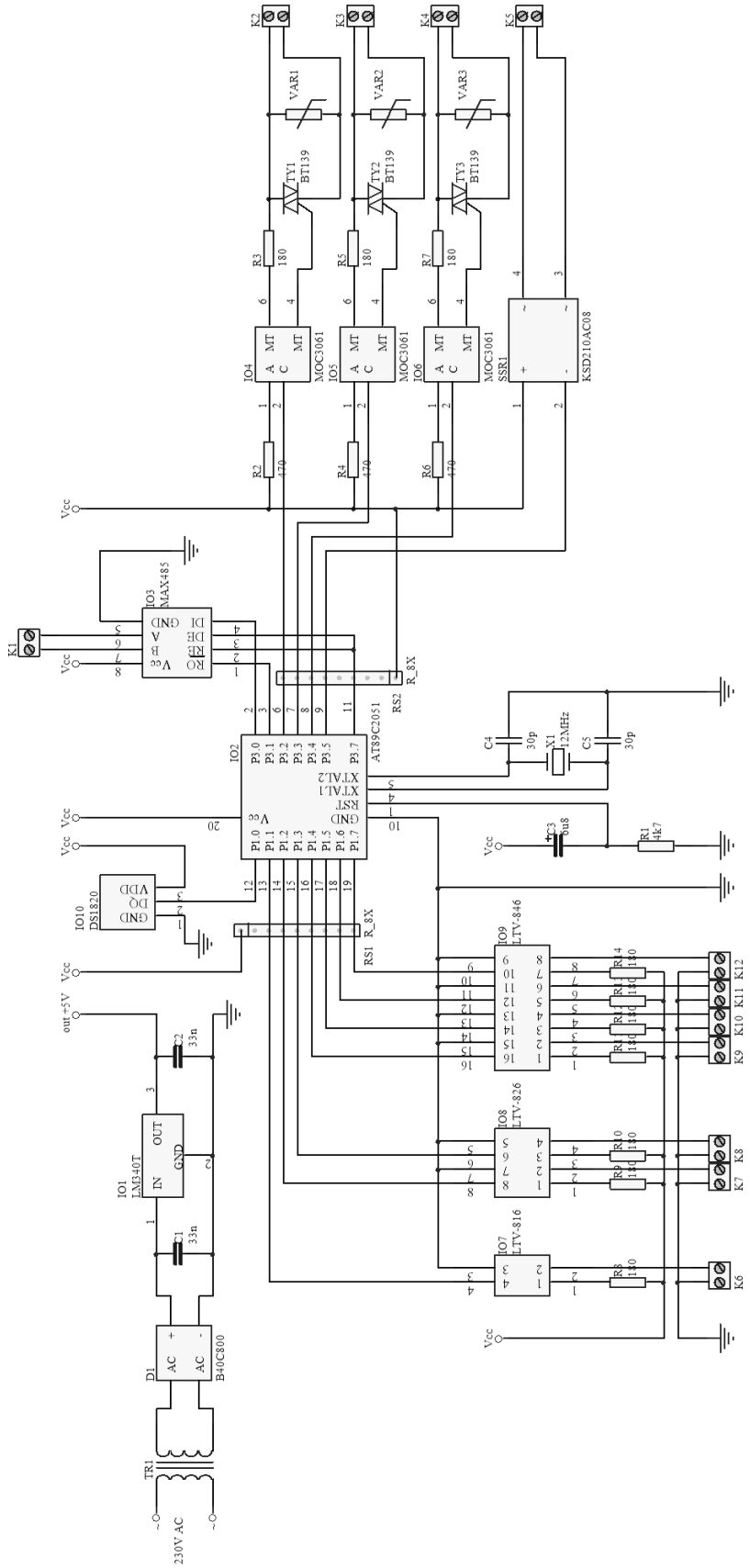
PŘÍLOHA P III: ROZDĚLENÍ MÍSTNOSTÍ PRO ŘÍDICÍ JEDNOTKY



PŘÍLOHA P IV: ŘÍDICÍ JEDNOTKA S PROCESOREM AT89C51



PŘÍLOHA P V: ŘÍDICÍ JEDNOTKA S PROCESOREM AT89C2051



PŘÍLOHA P VI: KOMUNIKAČNÍ PROTOKOL

Barevné značení

Striktně 0
Striktně 1
volitelné

			1B komunikace								HEX
			7	6	5	4	3	2	1	0	
Master	**RTS**		**1**	0	0	0	***adresa***				8x
Slave	CTS		**1**	0	0	0	***adresa***				8x
Master	**Dotaz na stav jednoho IO**		0	0	0	0	0	0	**1**	0	02
	stav (s) = 0, 1		0	*x*	***brána***		***pin***				xx
	brána = 0, 1, 2, 3										
	pin = 0, 1,… 7										
Slave	Stav zvoleného IO		0	***s***	***brána***		***pin***				xx
Master	**Dotaz na stav všech IO**		0	0	0	0	0	0	**1**	**1**	03
Slave	Postupně brány 0 - 3		*x*	*x*	*x*	*x*	*x*	*x*	*x*	*x*	xx
			x	*x*	*x*	*x*	*x*	*x*	*x*	*x*	xx
			x	*x*	*x*	*x*	*x*	*x*	*x*	*x*	xx
			x	*x*	*x*	*x*	*x*	*x*	*x*	*x*	xx
Master	**Nastav jeden IO**		0	0	0	0	0	**1**	0	0	04
	stav (s) = 0, 1		0	***s***	***brána***		***pin***				xx
	brána = 0, 1, 2, 3										
	pin = 0, 1,… 7										
Slave	Odpověď jako na 0x02		0	***s***	***brána***		***pin***				xx
Master	**Nastav všechny IO**		0	0	0	0	0	**1**	0	**1**	05
	Brána 0	Maska, které IO	0	*x*	*x*	*x*	*x*	*x*	*x*	*x*	xx
		Stavy IO	0	*x*	*x*	*x*	*x*	*x*	*x*	*x*	xx
	Brána 1	Maska, které IO	0	*x*	*x*	*x*	*x*	*x*	*x*	*x*	xx
		Stavy IO	0	*x*	*x*	*x*	*x*	*x*	*x*	*x*	xx
	Brána 2	Maska, které IO	0	*x*	*x*	*x*	*x*	*x*	*x*	*x*	xx
		Stavy IO	0	*x*	*x*	*x*	*x*	*x*	*x*	*x*	xx
	Brána 3	Maska, které IO	0	*x*	*x*	*x*	*x*	*x*	*x*	*x*	xx
		Stavy IO	0	*x*	*x*	*x*	*x*	*x*	*x*	*x*	xx
Slave	Po skončení slave odešle stav všech IO		/	/	/	/	/	/	/	/	
	(stejné, jako odpověd na 0x01 - stav IO)										
Master	**Dotaz na teplotu**		0	0	0	0	0	0	0	**1**	01
Slave	Teplota	Celá část	*x*	*x*	*x*	*x*	*x*	*x*	*x*	*x*	xx
		Desetinná část	*x*	*x*	*x*	*x*	*x*	*x*	*x*	*x*	xx
Master	**Kdo je připojen?**		**1**	**1**	0	0	0	0	0	0	C0
	Pauza, aby každý slave byl ready										
	Znovu poslat a čekat na stavy		**1**	**1**	0	0	0	0	0	0	C0
Slave	Čekací pauza podle adresy										
	Pak jeden slave za druhým odešle CTS		**1**	0	0	0	***adresa***				8x
			/	/	/	/	/	/	/	/	

PŘÍLOHA P VII: CD-ROM

Přiložený disk CD-ROM obsahuje tuto práci ve formátu PDF a veškeré zdrojové kódy všech zde popsaných programů.