Dálkové řízení a monitoring vysílačů

Remote control and monitoring of transmitters

Bc. Tomáš Plachý

Diplomová práce
2009

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
Ústav aplikované informatiky
akademický rok: 2008/2009

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Tomáš PLACHÝ**
Studijní program:	**N 3902 Inženýrská informatika**
Studijní obor:	**Počítačové a komunikační systémy**
Téma práce:	**Dálkové řízení a monitoring vysílačů**

Zásady pro vypracování:

1. **Seznámení s technologií .NET a potřebným vybavením.**
2. **Seznámení se současným stavem řízení a monitorování vysílačů.**
3. **Návrh a analýza potřebných funkcí navrhovaného systému.**
4. **Návrhu struktury a architektury řídícího a monitorovacího systému.**
5. **Realizace navrženého systému, testování.**
6. **Zhodnocení přínosu systému, významu pro uživatele a směr možného dalšího rozvoje.**

Rozsah práce:

Rozsah příloh:

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

Seznam odborné literatury:

1. **ASP.NET 2.0 cookbook. Sebastopol, Calif. : O/'Reilly, c2006.1014 s. ISBN: 0596100647.**
2. **Programming ASP.NET AJAX. Beijing ; Cambridge : O/'Reilly, c2007. 454 s. ISBN: 0596514247.**
3. **ASP.NET 2.0 a C-sharp : tvorba dynamických stránek profesionálně. Brno : Zoner Press, 2006. 1376 s. ISBN: 8086815382.**
4. **MCTS self-paced training kit (exam 70-536) : Microsoft .NET Framework 2.0 - application development foundation. Redmond : Microsoft Press, c2006. 1039 s. + 1 CD-ROM + 1 DVD. ISBN: 9780735626195.**
5. **MCTS self-paced training kit (exam 70-528) : Microsoft .NET framework 2.0 web-based client development /. Redmond, Wash. : Microsoft Press, c2007. 902 s. + 1 CD-ROM + 1 DVD-ROM /. ISBN: 0735623341.**
6. **Adding Ajax. Sebastopol, CA : O/'Reilly & Associates, 2007. 382 s. ISBN: 0596529368.**
7. **Ajax : Profesionálně /. Brno : Zoner Press, 2007. 668 s. ISBN: 9788086815770.**
8. **C-sharp a .NET 2.0: Profesionálně /. Brno : Zoner Press, 2006. 1197s. ISBN: 8086815420.**

Vedoucí diplomové práce: **Ing. Petr Šilhavý**
Ústav aplikované informatiky

Datum zadání diplomové práce: **20. února 2009**

Termín odevzdání diplomové práce: **27. května 2009**

Ve Zlíně dne 13. února 2009

prof. Ing. Vladimír Vašek, CSc.
děkan

UNIVERZITA TOMÁŠE BATI VE ZLÍNĚ
Fakulta aplikované informatiky
L.S.

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

ABSTRAKT

Hlavním cílem této diplomové práce je vytvoření komplexního systému pro vzdálený monitoring a správu rozhlasových vysílačů, přičemž výsledným produktem bude kolekce aplikací vázaných na vhodný monitorovací hardware. V první části práce jsou uvedeny a popsány softwarové technologie, které byly použity během vývoje projektu. Prostor byl vyhrazen i představení vhodného zařízení pro měření a řízení. Práce dále uvádí popis stávající situace a vlastností současného monitorovacího systému. Praktická část se zaměřila především na samotný produkt ControlIT. Dle námětu inspirovaného původním systémem a novými požadavky proběhla vstupní analýza, následuje přiblížení průběhu projektu, popis finálních aplikací a zhodnocení rozdílů stávajícího a nového řešení. Práci uzavírá náhled do budoucího vývoje projektu a popis možností implementace v praxi.

Klíčová slova: .NET, WCF, ASP.NET, ADO.NET, MS SQL Server, ControlIT, monitoring, rádio

ABSTRACT

The main aim of this thesis is development of a complex monitoring and administration system for radio transceivers with a collection of applications interconnected with a suitable monitoring hardware as the final product. Some software technology was used during the development of the project. Some software technology was used during the development project is mentioned and circumscribed in forepart of thesis. The next chapter was stipulated for description of the hardware for control and instrumentation. The last chapter of the theoretic part contains some information about the parameters of the actual monitoring system. Practical part is concentrating on project ControlIT, which is inspired by actual system and new needs defined by me and our company. The next chapters give description of an input analysis, course of the project development, the description of the final applications collection and an estimation of differences between the current and the new solution. My thesis is concluded by a part about the future of the next project development.

Keywords: .NET, WCF, ASP.NET, ADO.NET, MS SQL Server, ControlIT, monitoring, radio

Rád bych poděkoval vedoucímu diplomové práce Ing. Petru Šilhavému za cenné rady a podnětné připomínky udílené během vypracovávání této práce.

Úměrně velký dík patří RNDr. Pavlovi Foretníkovi, jenž mi poskytl důležité a praktické informace, čímž se velkou měrou zasloužil o rozvoj myšlenky celého projektu.

Dále chci poděkovat svým nejbližším a přátelům, kteří mi byli během psaní práce neocenitelnou podporou.

Prohlašuji, že

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

Prohlašuji,

že jsem na diplomové práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.

Ve Zlíně, 22.5.2009

.....

Podpis diplomanta

OBSAH

ÚVOD

Potřeba člověka ovládat a mít moc nad čímkoliv v dnešní době pronikla do téměř jakékoliv lidské činnosti. Trh automatizace, monitoringu, řízení a měření se díky této potřebě dynamicky rozvíjí a nabízí vhodné produkty pro téměř každý více či méně důležitý projekt.

Možnosti měřící a řídící techniky lze úspěšně nasadit a provozovat i v rámci rozhlasového vysílání, neboť každá stanice vyjadřuje potřebu kontrolovat veškerou techniku své vysílací sítě. I jediná sekunda ticha způsobená nefunkčností některého z prvků přináší nepříjemné finanční ztráty a sankce úřadů.

Snad všechna česká rádia si z tohoto důvodu budují své vlastní technologické know - how, jež zajistí včasné předání informací o případných poruchách na vysílacích trasách. Rozhodně se tedy nelze spoléhat pouze jen na dobrou vůli posluchačů, kteří v případě ticha na jejich oblíbené frekvenci na případnou chybu upozorní.

Rozhlasová stanice Radio Haná disponující signálovým pokrytím značné části území Moravy provozuje své frekvence na síti vysílačů, které si týdně naladí průměrně 144 000 posluchačů. Pro technologii takového rozsahu vyvstala zásadní potřeba zavedení jednotného systému pro vzdálený monitoring a řízení. V současné době samozřejmě funguje jisté vlastní řešení kontroly stavu komponent vysílačů, ovšem možnosti informačních technologií mezitím značně pokročily kupředu a stávající technologie se postupně stala morálně zastaralou.

Projekt diplomové práce tedy přichází s novým, ze základu inovovaným řešením, jež by mělo úspěšně změnit současný stav monitorovacích technologií na stěžejních vysílačích a navíc se postupně rozšířit na celou síť lokálních vykrývačů.

V rámci teoretické části budou představeny jednotlivé softwarové technologie a hardwarová řešení umožňující realizaci projektu, praktická část se pak zabývá definicí funkcí včetně následného detailního popisu řešení nového systému. V závěru práce je uvedeno celkové zhodnocení projektu, jeho přínos a stav nasazení do reálného provozu.

I. TEORETICKÁ ČÁST

1 SOFTWAROVÉ TECHNOLOGIE POUŽITÉ PŘI VÝVOJI

Majoritní prostor teoretické části diplomové práce se zabývá popisem softwarového vybavení, které bylo použito během vývoje celého projektu. Jelikož pro běh všech aplikací byla zvolena platforma Microsoft Windows, pocházejí všechny popisované technologie právě z dílen výše jmenované softwarové korporace. Zdrojové kódy v praktické části odpovídají syntaxi jazyka Visual C#.

1.1 Visual Studio

Microsoft Visual Studio nabízí programátorovi jako kompletní vývojové prostředí (IDE) vysoký pracovní komfort a funkční flexibilitu. Vytvořením tohoto softwarového produktu si společnost Microsoft splnila záměr dodat programátorům prostředí umožňující jednotnou tvorbu aplikací napříč vlastními platformami:

- Microsoft Windows
- .NET
- Windows Mobile
- Windows CE
- .NET Compact Framework
- Microsoft Silverlight

Široké spektrum nástrojů umožňuje vývojářům tvorbu různých druhů aplikací dle následujícího rozdělení:

- Konzolové aplikace
- Aplikace s grafickým rozhraním realizovaným pomocí knihoven Windows Forms
- Služby
- Webové stránky, aplikace a služby

Rychlost tvorby programu citelně ovlivňuje editor kódu podporující IntelliSence, pomocí něhož dochází k automatickému dokončování kódu, nabízení vhodných možností a zamezování chyb.

Případné chyby programu kontroluje integrovaný debugger na úrovni kódu i na úrovni stroje .NET. Další vestavěné nástroje zahrnují designer formulářů pro tvorbu GUI aplikací, designer webu, tříd a databázových schémat. Vysoká modulárnost celého Visual Studia

umožňuje přidávat rozšíření, což vylepšuje funkčnost a užitnou hodnotu vývojářského software. [1]

1.1.1 Architektura

Samotné Visual Studio jako takové neobsahuje žádnou přímou podporu programovacího jazyka nebo nástroje. Veškerou funkčnost získává od služeb zabalených do balíčků VSPackage, které je nutné dle potřeby nainstalovat. Základní IDE poskytuje tři služby:

- *SVsSolution* – umožňuje očíslování projektů a sestav
- *SVsUIShell* – rozdělování na okna a UI funkce (panely, nástrojové lišty)
- *SVsShell* – registrace balíčků VSPackage

Krom výše jmenovaného se IDE stará navíc o veškerou koordinaci služeb a komunikaci mezi nimi. Podpora programovacích jazyků je přidána balíčkem *Language service*, jež definuje různá rozhraní, které lze v balíčcích VSPackage implementovat, a tím přidat u každého nainstalovaného jazyka různou funkčnost. Tímto způsobem se přidává podpora zvýraznění syntaxe, zvýrazňování párů závorek, doplňování příkazů, tipy parametrů informací nebo chybové značky pro kompilaci na pozadí. [2]

Jazykové služby mohou použít kód z překladače nebo editoru jazyka. Při implementaci ve strojovém kódu mohou být použity pomocí COM rozhraní, případně Babel Frameworku. Pro spravovaný kód obsahuje MPF obaly pro psaní spravovaných jazykových služeb.

1.1.2 Prvky IDE

Visual Studio, stejně jako každé jiné integrované prostředí, obsahuje několik základních, navzájem propojených prvků:

- **Editor kódu** používaný pro všechny podporované jazyky umožňuje zvýraznění syntaxe a s pomocí IntelliSence automatické dokončování kódu nejen pro proměnné, metody, funkce, ale též složitější konstrukce jako cykly a dotazy. Podporu IntelliSence zahrnují všechny základně nainstalované jazyky včetně XML, CSS či JavaScriptu.

  Další navigační pomůcky zahrnují podporu záložek v kódu pro rychlejší navigaci, sbalování bloků kódu. Za oceňovanou vlastnost editoru se považuje podpora snippetů, což jsou uložené šablony opakujících se bloků kódu, které lze dle potřeby rychle vyvolat a přizpůsobit aktuálnímu projektu.

Editor mimo jiné podporuje refaktorování, tedy změny ve zdrojovém kódu programu neměnící jeho funkčnost, ale jeho vnitřní strukturu tak, aby byl pro programátora lépe čitelný.

Pomocí tzv. přírůstkové kompilace, tedy postupné kompilace na pozadí Visual Studio průběžně hlídá a informuje o výskytu syntaktických a kompilačních chyb, které následně podtrhne červenou vlnovkou, případná varování pak zelenou. Pro kompilaci na pozadí se používá jiný kompilátor, takže se během psaní negeneruje spustitelný kód. [3]

- **Integrovaný debugger** umožňuje procházení spravovaného i strojového kódu, přičemž jej lze použít pro debugování aplikací jakéhokoliv jazyka podporovaného Visual Studiem. Debugger smí být nasazen i na kontrolu běžících procesů, navíc v případě dostupnosti zdrojového kódu lze tento procházet, v opačném případě se zobrazuje kód Assembleru.

  Důležitou vlastností debuggeru je podpora nastavení breakpointů (bodů zastavení na určité pozici chodu programu) a watch sledujících obsah proměnných v procesu. Zastavení na breakpointu může být též podmíněné, tedy aktivované jen v případě splnění určité podmínky kódu.

  Pomocí ovládacích nástrojů lze řídit běh procházení přes breakpointy, tzn. přecházení či vstupování do volaných funkcí. Při procházení přes proměnnou může být zobrazena a editována její hodnota bez nutnosti přerušení běhu programu, stejně tak debugger umožňuje pomocí funkce *Edit and Contiue* přímou editaci kódu za běhu, samozřejmě i s patřičným následným proběhnutím vytvořených změn. [3]
- **WinForms designer** se používá k rychlému a elegantnímu vývoji GUI aplikace za pomocí knihoven Windows Forms.

  Zdrojový kód každého formuláře obsahuje dva základní soubory:

  1. FormName.cs
  2. FormName.Designer.cs

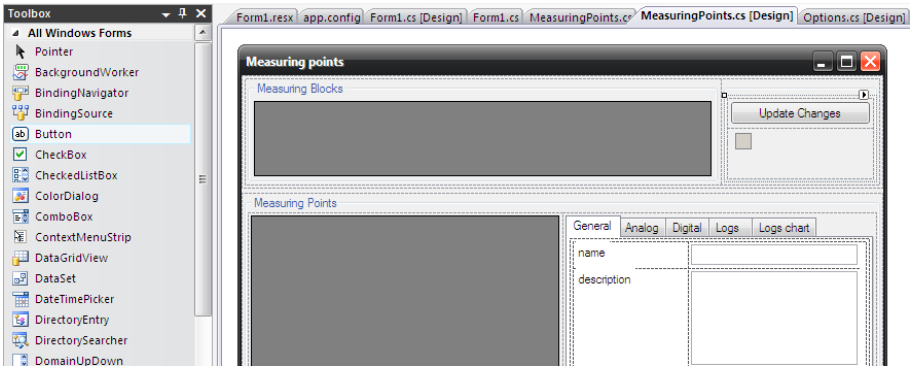
  Zatímco v prvním jmenovaném bývá umístěna aplikační logika formuláře, do druhého souboru designer na pozadí dynamicky vkládá generovaný kód popisující vzhled a rozmístění prvků formuláře. I tento kód má možnost vývojář měnit za účelem vlastních drobných úprav vzhledu.

  Nástroj designéru ve vývojovém prostředí obsahuje paletu ovládacích prvků, mezi nimiž najdeme krom základních tlačítek, popisek či comboboxů i různé

specializované moduly pro práci s daty, interním během programu, případně dodatečně doinstalované objekty (Obr. 1).

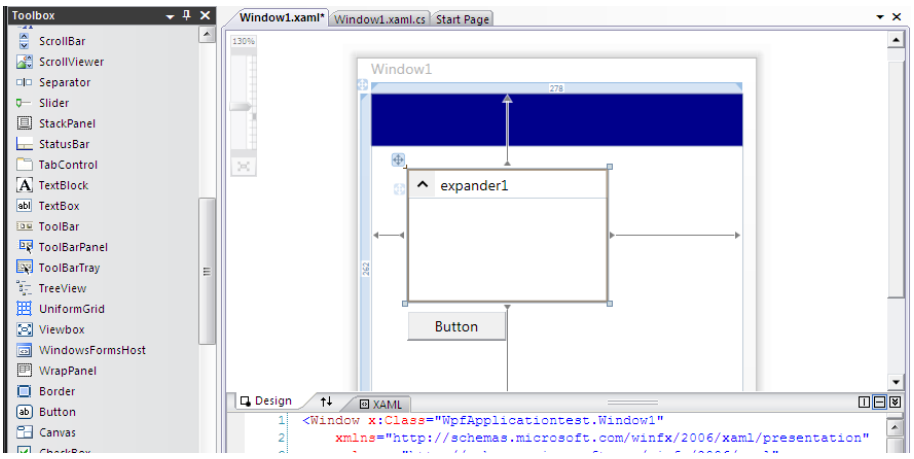
Všechny jmenované prvky mohou být uchopeny a umístěny na plochu formuláře. Práci zjednodušují a přehlednou strukturu udržují různé druhy kontejnerů, které do svého těla přesunuté prvky vloží a případně uzamknou.

Velká část ovládacích prvků zobrazujících data umožňuje jednoduché propojení přes vlastnosti databindings či datasource do různých datových zdrojů.



Obr. 1. WinForms designer s toolboxem a návrhem formuláře

- **WPF designer** (Obr. 2) byl představen jako novinka Visual Studia 2008. Pomocí tohoto designéru vývojář navrhuje uživatelské rozhraní pro Windows Presentation Foundation (WPF), jehož definice vzhledu je postavena na jazyku XAML. Design vytvořeného formuláře lze pomocí XAML kódu libovolně upravovat v nástrojích Microsoft Expression, se kterými se Visual Studio dobře provazuje. XAML kód se s kódem aplikace propojuje pomocí code-behind modelu. Celkově díky WPF došlo k maximálnímu odloučení návrhu designu aplikačního rozhraní a jeho funkční logiky.



Obr. 2. WPF designer s toolboxem, návrhem formuláře a kódem XAML

- **Web designer** umožňuje vývojáři v integrovaném prostředí jednoduše vytvářet webové stránky a aplikace na bázi ASP.NET, přičemž podporuje zvýraznění syntaxe a IntelliSense kódu HTML, CSS a JavaScript. Engine zobrazování vytvořeného webu Visual Studio sdílí s aplikací Expression Web.
- **Ostatní designéry** zjednodušují ve Visual Studiu práci s návrhem objektového modelu tříd, designem dotazů z SQL atd.
- **Průzkumníci** se řadí do speciální kategorie editačních nástrojů integrovaného vývojového prostředí. Obecně usnadňují práci a zpřístupňují vlastnosti v těchto kategoriích:
    1. **Průzkumník serveru** se používá pro přístup k databázi na přístupném lokálním nebo síťovém počítači. Mimo jiné s ním lze provádět procházení výpočtů výkonu, služeb Windows, zprávy, či jej použít jako zdroj dat.
    2. **Průzkumník dat** je užíván pro správu databází MS SQL Serveru. S pomocí tohoto průzkumníku lze přímo z Visual Studia provádět úpravy připojených tabulek databází, generovat za použití Transact-SQL nebo spravovaného kódu dotazy a uložené procedury.
    3. **Průzkumník řešení** zobrazuje hierarchii řešeného projektu. Umožňuje pracovat se všemi soubory kódů, spravovat reference či přidávat a odebírat do solution další projekty.
    4. **Průzkumník objektů** slouží pro procházení jmenných prostorů a knihoven tříd Microsoft .NET Frameworku. Zobrazuje strukturu a možnosti parametrů a metod vybraných tříd.
- **Editor vlastností** vypisuje a edituje dostupné vlastnosti v GUI panelu Visual Studia pro všechny vybrané objekty, včetně tříd, formulářů, webových stránek atd.

1.1.3 Dodané programovací jazyky

Přestože samotné IDE Visual Studia díky své modulárnosti nativně neobsahuje žádný programovací jazyk, Microsoft dodává společně s prostředím hned několik produktů, které zajišťují vývojáři možnost volby pro něj nejpohodlnějšího řešení:

- **Microsoft Visual C#** byl vyvinut speciálně pro použití s Visual Studiem, tedy jeho implementace je cílena na .NET Framework spolu s jazykovými službami, které pak Visual Studio IDE umožňují podporovat C# projekty.

Zatímco jsou jazykové služby součástí Visual Studia, překladač je k dispozici odděleně jako součást .NET Frameworku. Poslední veřejně publikovaný překladač ve Visual Studiu podporoval Visual C# 2008 verzi 3.5.

Dle slov výrobce lze Visual C# považovat za koncentrát nejlepších vlastností všech známých programovacích jazyků. Přebírá jednoduchost syntaxe Javy společně s komplexností a robustností C/C++, přičemž vývojáře zbavuje mnoha nepříjemností známých u C/C++, např. díky *garbage collectoru* starostí o život nepotřebných objektů a správu paměti. [3]

- **Microsoft Visual C++** je znám jako implementace překladače C a C++ od společnosti Microsoft, jehož specifické nástroje a služby se integrují do Visual Studia. Překladač umožňuje kompilaci v C nebo C++ módu, přičemž platí:
    - V případě kompilace C se dodržují platné normy ISO C s některými specifikacemi C99 společně s vlastními doplňky v knihovnách vytvořených Microsoftem.
    - Během kompilace C++ překladač pracuje se specifikací ANSI C++, přičemž podporuje i C++/CLI pro psaní spravovaného kódu, případně i kombinovaný mód (strojový a spravovaný kód zároveň).

  Jazyk Visual C++ obsahuje podporu modelu COM a knihovny MFC. Krom vytváření a přizpůsobování GUI aplikací přes MFC může využívat i designer formulářů Visual Studia. Dále lze pomocí Visual C++ pracovat s běžnými funkcemi a objekty Windows API.

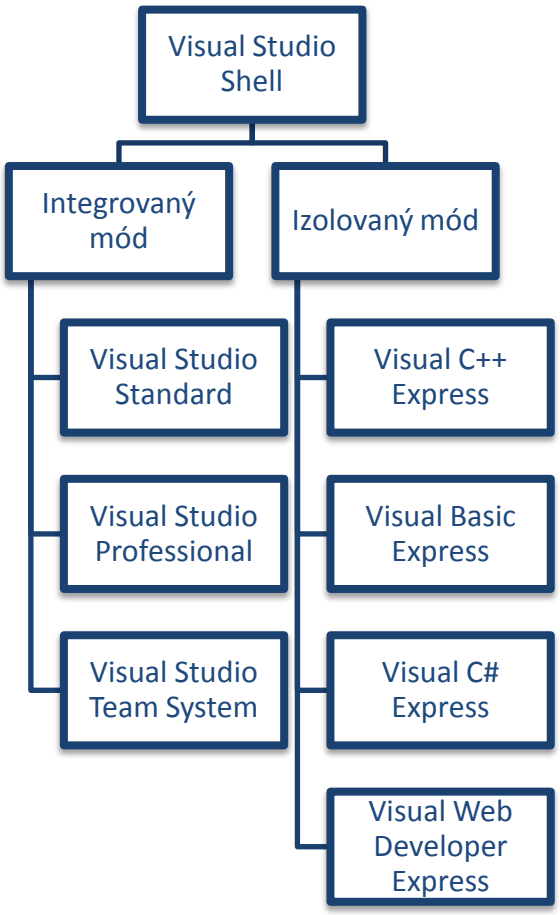
- **Microsoft Visual Basic .NET** byl představen ve Visual Studiu .NET 2002 jako nástupce oblíbeného Visual Basic 6.0, který do té doby znamenal jeden z nejrychlejších způsobů vývoje aplikací, tedy dle Microsoftu. [1] Oproti starší verzi přichází s několika zásadními změnami:
    - **Povinná deklarace proměnných** zakazuje paměťově náročný typ *Variant* používaný v případě opomenutí deklarace proměnné.
    - **Kompletní objektově orientovaný model** plně podporuje OOP stejně jako ostatní jazyky .NETu.
    - **Rychlejší kód** dosahující díky kompilaci do strojového kódu až těsně před spuštěním na klientské stanici mnohonásobně vyšší rychlosti oproti Visual Basicu 6.

- **Rozšíření možností jazyka**, které si vynutilo převedení pod platformu .NET

Za hlavní nevýhodu jazyka Visual Basic .NET bývá považována zpětná nekompatibilita kódu se staršími verzemi. Přestože Microsoft vydal několik nástrojů pro jednoduchou konverzi projektů, složitější programové celky si museli programátoři přizpůsobit sami. Dnes je již tento problém díky delšímu působení verze .NET na trhu téměř pasé. [4]

1.1.4 Edice

Microsoft vydává Visual Studio v několika různých řadách. Díky zajímavému cenovému a funkčnímu rozdělení si na své přijde programátor začátečník, stejně tak i tým profesionálů. Rozdělení a vztahy mezi různými edicemi ilustruje obrázek se schematickým diagramem (Obr. 3) a následující popis odlišností [3] včetně shrnující tabulky vlastností (Tab. 1).



Obr. 3. Vztahy edic Visual Studia

- **Visual Studio Express** poskytuje Microsoft zdarma jako jednotlivé edice odlehčených individuálních IDE, jež jsou rozděleny na základě určitých jazyků. Díky „své ceně“ obsahují tyto edice jen několik základních nástrojů, přičemž postrádají podporu designéru tříd, vzdálené databáze pro designer dat, stejně tak podporu rozšíření. Studentům a amatérům, kteří jsou hlavní cílovou skupinou, byl poskytnut jen omezený rozsah MSDN knihoven.
- **Visual Studio Standard** oproti Express edicím nabízí IDE pro všechny produkty a rozšíření, navíc může podporovat celou MSDN knihovnu. Bohužel neobsahuje integraci s MS SQL serverem, prvek Server Explorer tedy existuje až u vyšších verzí. Přestože Visual Studio 2005 umožňovalo tvorbu aplikací pro mobilní zařízení na bázi .NET compact frameworku, u novějšího vydání je tento směr vývoje zařazen až do verze Professional, stejně tak i podpora vývoje WCF aplikací.
- **Visual Studio Professional** obsahuje všechny nástroje předchozích edic včetně integrace s MS SQL serverem, která pak umožňuje jednoduchou tvorbu, správu a komunikaci s databází přímo z prostředí Visual Studia. Samozřejmostí je již podpora vývoje pro mobilní zařízení a WCF služby. Professional edice obsahuje kromě jiného i všechny potřebné designéry.
- **Visual Studio Team System** poskytuje kromě jiného sadu softwarového vývoje, spolupráce, metrik a záznamových nástrojů. Zajímavé jsou odlišnosti sad nástrojů na základě role vývoje software, pro kterou je použito. Tyto nástroje se dle rolí dělí do několika skupin:
    - Architecture Edition
    - Database Edition
    - Development Edition
    - Team Explorer
    - Test Edition

Tab. 1. Vlastnosti různých edic Visual Studia

Produkt	Express	Standard	Professional	Team System
Rozšíření	ne	ano	ano	ano
Externí nástroje	minimální	ano	ano	ano
Nastavení projektů	omezeně	omezeně	ano	ano
MSDN integrace	MSDN Express	ano	ano	ano
Designer tříd	ne	ano	ano	ano
Refaktorování	omezeně	ano	ano	ano
Debugging	omezeně	ano	ano	ano
64-bitové aplikace	ne	ano	ano	ano

Procesory Itanium	ne	ne	ne	ano
Tools for Office	ne	ne	ano	ano

1.1.5 Historie

Visual Studio společnost Microsoft postupně vyvíjí jako komplexní vývojářský software již od roku 1997. Od té doby bylo uvedeno několik verzí, přičemž produkt stále prochází rapidním vývojem:

- **Visual Studio 97** lze označit za prapůvodce, kdy nápad spojit několik v té době oddělených vývojářských nástrojů dohromady Microsoftu zcela zjevně vyšel. První verze Visual Studia obsahovala následující součásti:
  - Visual Basic 5.0 – vývoj především pro Windows
  - Visual C++ 5.0 – vývoj především pro Windows
  - Visual J++ 1.1 – vývoj pro Javu a Windows
  - Visual FoxPro 5.0 – tvorba databází, především xBase
  - Visual InterDev – vývoj ASP aplikací a webů
  - Off-line výňatek z MSDN knihoven

  Nutno podotknout, že zatímco Visual C++, Visual J++, InterDev a MSDN Library používalo jednotné vývojové prostředí - Developer Studio, produkty Visual Basic a Visual FoxPro byly na tomto prostředí nezávislé. Sjednocení přišlo až v dalších verzích.
- **Visual Studio 6.0** obsahuje poslední verzi Visual Basicu založeného na základě COM, další verze budou pracovat jen pod .NET frameworkem. Právě nelibost některých programátorů vůči .NETu činí ze starého Visual Basicu 6.0 stále oblíbený nástroj. Produkty Visual C++ , Visual Basic a Visual FoxPro pracovaly v odděleném IDE, zatímco Visual J++ a Visual InterDev sdílely společné IDE. Visual FoxPro se do dalších verzí Visual Studia již nedostala a byla nabízena zvlášť.
- **Visual Studio .NET (2002)** znamenal v rámci vývoje softwarového balíku revoluční průlom, neboť bylo představeno ucelené vývojové prostředí pro spravovaný kód za použití .NET Frameworku.

  Oproti předchozím postupům, kdy byly programy ihned kompilovány do strojového kódu, se software vyvinutý za použití .NET ukládá do formátu Microsoft Intermediate Language (MSIL) nebo Common Intermediate Language (CIL). V případě MSIL aplikace se kompiluje až v okamžiku spuštění a navíc

přímo pro platformu, na které se spouští, což umožňuje dostupnost programu na několika různých platformách v nejoptimálnější možné podobě pro každou z nich. V rámci nového Visual Studia představil Microsoft C#, nový programovací jazyk cílený právě na platformu .NETu. Zatímco uživatelé téměř neregistrovali následníka J++ zvaného nyní J#, zcela nový od základu předělaný Visual Basic rozčaroval snad každého. Velké změny ovšem přinesly čistší rozhraní a větší soudržnost IDE prostředí. [1]

- **Visual Studio .NET 2003** představilo drobná zlepšení. Mimo aktualizaci .NET Frameworku na verzi 1.1 již umožňuje vývoj aplikací pro mobilní zařízení.
- **Visual Studio 2005** postrádá v názvu příponu .NET, která se nyní hlavně pojí s .NET Frameworkem, v té době aktualizovaným na verzi 2.0. Visual Studio se tak dočkalo vylepšení v podobě podpory všech prvků nové verze Frameworku. Verze 2005 přinesla podporu tvorby 64-bitových aplikací v podobě kompilátorů pro x86-64 (AMD64 a Intel 64 a IA-64 (Itanium). Samotné vývojové prostředí ovšem zůstalo jen 32-bitové.
- **Visual Studio 2008** bylo v době psaní práce nejnovější veřejně dostupnou verzí vývojářského balíku. Dle záměrů Microsoftu mělo být určeno pro vývoj aplikací pro Windows Vista, Office 2007 a webové aplikace. Verze 2008 tak přinesla nový designér pro tvorbu Windows Presentation Foundation či HTML editor vycházející z produktu Microsoft Expression Web. Společně s novým Visual Studiem byl uveden .NET Framework 3.5 rozšířený o technologie WCF, LINQ a jiné. Vývojář nyní může pracovat s knihovnou MFC 9.0 obsahující nové funkce vizuálních stylů představených ve Windows Vista
- **Visual Studio 2010** znamená prozatím jen hudbu budoucnosti. Přesto některé nové prvky této verze již byly zveřejněny. Lze očekávat vylepšenou podporu automatického dokončování kódu za pomocí IntelliSense, pro jejíž služby se plánuje využití SQL Server Compact databáze.

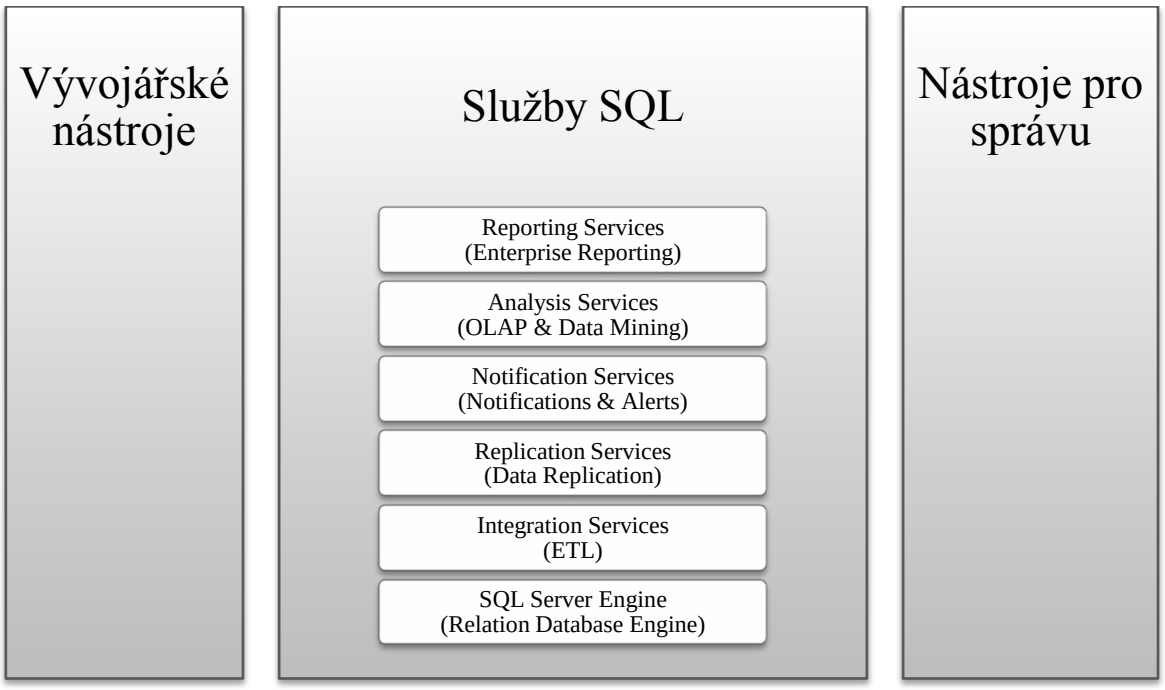
1.2 MS SQL Server

Microsoft SQL Server patří do skupiny relačních databázových systémů (RDBMS) vyvinutý společností Microsoft. Patří mezi jedny z nejvýkonnějších relačních databázových systémů, které dnes velmi často tvoří základní databázovou vrstvu podnikových informačních systémů.

Hlavními dotazovacími jazyky jsou ANSI SQL a T-SQL, přesto pro komunikaci s SQL Serverem není nutná detailní znalost těchto jazyků SQL, neboť existuje množství uživatelsky přijatelnějších grafických nástrojů velice dobře usnadňujících práci s tímto serverem.

1.2.1 Architektura databázových služeb

Databázové služby SQL serveru jsou rozděleny do několika vrstev, které popisuje následující obrázek (Obr. 4):



Obr. 4. Struktura databázových služeb SQL Serveru

- **Služba SQL Server Reporting Services** představuje ucelenou na server orientovanou platformu pro vytváření, správu a doručování tradičních papírových, případně interaktivních, webových sestav. Služba Reporting Services spojuje funkce pro správu dat serveru SQL Server a systému Microsoft Windows Server s aplikacemi sady Microsoft Office. Reporting Services podporuje celou řadu datových zdrojů: [5]
    - OLE DB
    - Open Database Connectivity (ODBC)
    - Množství výstupních formátů (pro webové prohlížeče, aplikace sady Microsoft Office, PDF, CSV,...)

- **SQL Server Analysis Services** opět patří mezi služby, jež jsou součástí MS SQL Serveru. Přesto je na SQL Serveru zcela nezávislý, neboť se jedná o samostatný OLAP server, který ke komunikaci s klienty využívá rozhraní OLE DB for OLAP. Úlohou technologie OLAP je poskytování FASMI (tj. Fast Analysis Shared Multidimensional Information).
- **SQL Server Notification Services** funguje jako framework vytvářející aplikace generující různá upozornění. Zároveň poskytuje platformu pro hostování těchto aplikací. Lze tedy vytvořit aplikaci, která generuje hlášení na základě nějaké události a následně ji umístit na Notification Services server.
- **SQL Server Integration Services** nahrazuje jako nová součást MS SQL Serveru 2005 službu transformace dat Data Transformation Services pro SQL Server 2000 s tím, že poskytuje funkce a výkon potřebný k vytváření aplikací pro integraci dat. Data za pomocí Server Integration Services přicházejí do datového skladu z různých zdrojů, transformují a ukládají se. Nabízí se široká paleta možných transformací dat:
  - Třídění dat
  - Spojování dat
  - Konverze dat
  - Podmíněné rozdělení dat
  - Odvozené sloupce
  - Agregace
  - Lookup
  - Fuzzy lookup, Fuzzy grouping - „Nepřesné“ vyhledávání nebo seskupování vhodné například k seskupování dat zadaných z formulářů na WWW.

1.2.2 Architektura databázového engine

Za základní komponentu každého SQL serveru se považuje databázový engine, jenž odpovídá za ukládání, správu a zabezpečení dat. Data mohou být v MS SQL Serveru ukládána ve formě relačních tabulek, případně také ve formě XML dokumentů. Nová verze SQL Serveru přinesla rozšířenou podporu pro práci s XML dokumenty, což znatelně rozšiřuje možnosti serveru. Mezi základní úkoly databázového engine patří: [5]

- **Spolehlivé datové úložiště**, přičemž základním požadavkem je samozřejmě kvalitní spolehlivý hardware. Ideální případ hovoří o umístění dat na pole RAID.

Přesto samotnému SQL Serveru na použitém hardware nezáleží, neboť si sám spravuje své datové struktury tak, aby všechna data byla uchována v konzistentní podobě. Každý řádek tabulky se ukládá v tzv. datové stránce o velikosti 8KB, což přináší omezení designu při návrhu tabulek.

- **Kontrola přístupu** k datům se provádí na několika úrovních:
  - Kontrola na úrovni serveru
  - Kontrola na úrovni databáze
  - Kontrola na úrovni objektu

  Novinkou od verze 2005 jsou schémata, poskytující možnost kontroly přístupu. Autentizace a přihlášení k SQL Serveru lze provádět těmito metodami:
  - Uživatelský učet Windows (Windows Autentizace)
  - Uživatelský účet na serveru
  - Uživatelský účet Active Directory (Integrated Authentication)
- **Integrita dat** znamená takový stav, při němž záznamy v celé databázi vyhovují soustavě určitých definovaných pravidel. Rozeznáváme tři úrovně datové integrity:
  - Doménová – jedná se o hodnoty, jež můžeme uložit do jednotlivých sloupečků
  - Entitní - neexistují duplikátní záznamy
  - Referenční – znamenající odkazy mezi jednotlivými tabulkami, popřípadě mezi sloupci z jedné tabulky pouze na existující hodnoty

  Pro zajištění integrity má SQL Server několik metod – constraints, triggery a schémata.
- **Transakční log** uchovává v SQL Serveru informace o probíhajících transakcích, přičemž využívá metodu write-ahead - při práci se serverem se nejprve uloží záznam do logu, poté jsou teprve data měněna v cache. Systém sám v určitých intervalech ukládá změny přímo na disk u těch transakcí, které byly dokončeny. Schopnosti transakčního logu se oceňují v případě havárie serveru, neboť je užíván při jeho obnově. [2]

1.2.3 Edice

S příchodem produktové řady SQL Server 2005 přišla společnost Microsoft s rozdělením do čtyř nových edic majících lépe pokrývat potřeby a finanční možnosti zákazníků. Jedná se o tyto produkty:

- Express – distribuovaná zdarma
- Workgroup – pro pracovní skupiny
- Standard – profesionální nasazení
- Enterprise – nejvyšší verze

Hlavní produktovou řadu doplňují edice pro specifické použití:

- Developer
- Mobile – používána ve Windows Mobile přístrojích
- Compact – malé embedded přístroje

Tab. 2. Srovnání možností a limitů všech edic MS SQL Server

Funkce	Express	Workgroup	Standard	Enterprise
Počet CPU	1	2	4	Bez limitu
64bit podpora	WOW	WOW	Ano	Ano
RAM	1	3	Bez limitu	Bez limitu
Velikost databáze	4GB	Bez limitu	Bez limitu	Bez limitu
Partitioning	Ne	Ne	Ne	Ano
Indexed views	Ne	Ne	Ne	Ano
Online Indexing	Ne	Ne	Ne	Ano
Online system changes	Ano	Ano	Ano	Ano
Online Page and File Restore	Ne	Ne	Ne	Ano
Database Mirroring	Ne	Ne	Ne	Ano
Backup Log-shipping	Ne	Ano	Ano	Ano
Failover Clustering	Ne	Ne	Ano (2 nody)	Ano

Edice Microsoft SQL Express distribuovaná zdarma navíc obsahuje některá další omezení a vlastnosti:

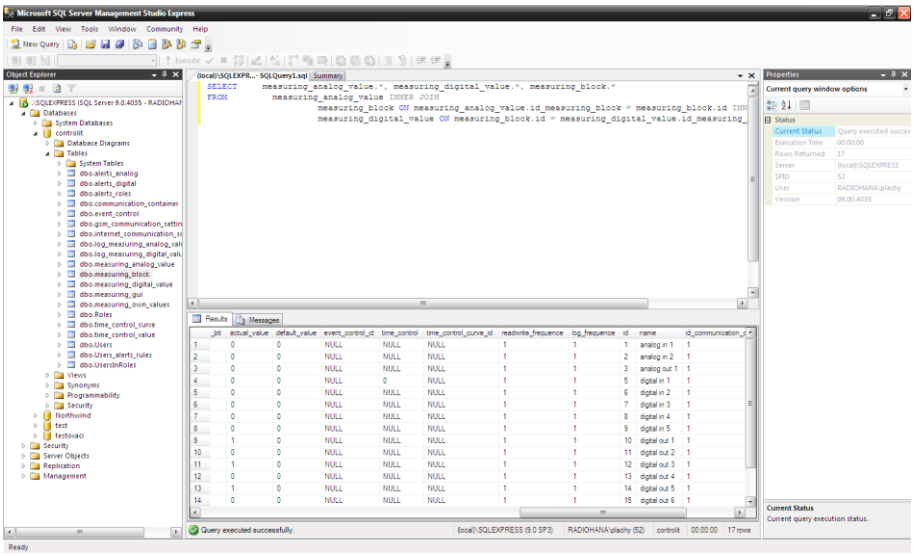
- Neobsahuje službu SQL Server Agent.
- Velikost DB omezena na 4GB (bez velikosti transakčního logu), není omezeno množství vytvořených databází.
- Neobsahuje OLAP, Integration Services (SSIS), Notification Services, Report Builder, Database Engine Tuning Advisor, SQL Profiler a další enterprise funkce
- Instalace bez SQL Server Management Studio, dá se doinstalovat zvlášť.
- Není podporován protokol http jako jedna z možností připojení k serveru.
- Není omezen počet příhlášených uživatelů, ani běžících strojů na jednom počítači.

1.2.4 Pracovní nástroj MS SQL Management Studio

Volně stažitelný nástroj z dílen Microsoftu se používá pro kompletní správu dat a struktury databází v MS SQL Serveru. Produkt v sobě spojuje aplikace Enterprise Manager a Query Analyzer známé z verze 2000. Vzhled aplikace se zpracovaným SQL dotazem ilustruje obrázek níže (Obr. 5).

Mezi základní úlohy proveditelné v SSMS patří:

- Správa databází
- Správa databázových objektů – tabulky, schémata, procedury, relace, funkce, triggery, indexy
- Správa uživatelských účtů, loginů a rolí
- Tvorba a správa automatizovaných úloh
- Zálohování a obnova struktury a dat
- Nastavení parametrů serveru – jazyk, využití CPU, paměti, výchozí umístění souborů
- Psaní a ladění SQL skriptů, interaktivní tvorba SQL dotazů, včetně jejich organizace do projektů



Obr. 5. Microsoft SQL Server Management Studio

1.3 ADO.NET

Značná část dnes používaných aplikací funguje na bázi zpracování dat, proto platforma .NET Framework definuje celou řadu jmenných prostorů pro komunikaci s místními

i vzdálenými úložišti dat. Dnes již historické ODBC, DAO, RDO, RDS a ADO nahradilo ADO.NET A LINQ.

ADO.NET sestává z řízených tříd, které umožňují aplikacím .NET připojení ke zdrojům dat, spravovat odpojená data a vykonávat příkazy.

1.3.1 Architektura ADO.NET

Zatímco předchozí databázové technologie používaly obecnou sadu objektů pro práci s daty bez ohledu na to, s jakým zdrojem dat pracují, ADO.NET používá model poskytovatelů dat. Jedná se o sadu tříd umožňující přístup ke konkrétní databázi. Představuje tedy pomyslný most mezi aplikací a zdrojem dat. Třídy tvořící poskytovatele dat obsahují objekty popsané tabulkou (Tab. 3).

Tab. 3. Objekty poskytovatele ADO.NET

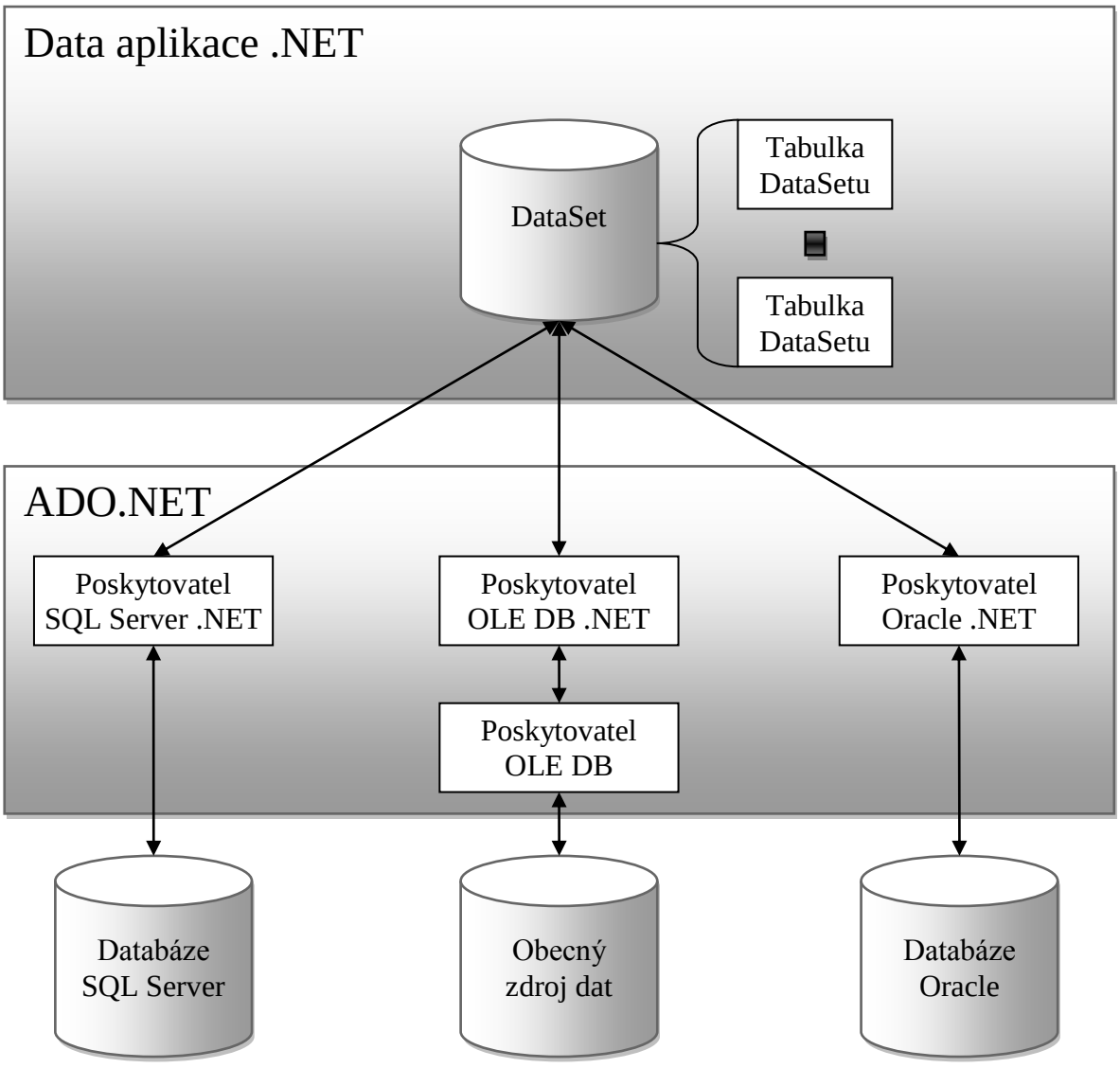
Objekt	Třida	Implementované rozhraní	Význam
Connection	Dbconnection	IDbConnection	Poskytuje funkce pro připojení a odpojení od nějakého úložiště dat
Command	DbCommand	IDbCommand	Reprezentuje dotaz SQL, případně uloženou proceduru.
DataReader	DbDataReader	IDataReader, IDataRecord	Poskytuje přístup k datům, která ovšem umí jen číst
DataAdapter	DbDataAdapter	IDataAdapter, IDbDataAdapter	Přenáší sady dat mezi volajícím a datovým úložištěm. Obsahuje 4 objekty příkazů pro select, insert, delete a update
Parameter	DbParameter	IDataParameter, IDbDataParameter	Reprezentuje pojmenovaný parametr v dotazu
Transaction	DbTransaction	IDbTransaction	Představuje databázovou transakci

.NET Framework integruje kolekci čtyř poskytovatelů dat, která může být samozřejmě rozšířena o další poskytovatele třetích stran. Mezi jmenované pak patří:

- **Poskytovatel SQL serveru** umožňující optimalizovaný přístup k databázi SQL Serveru.

- **Poskytovatel OLE DB** připojitelný k jakémukoliv zdroji dat, který obsahuje ovladač OLE DB.
- **Poskytovatel Oracle** s optimalizovaným přístupem do databází Oracle 8i a novějších
- **Poskytovatel ODBC** poskytující přístup do jakéhokoliv zdroje dat disponujícího ovladačem pro ODBC:

Přestože .NET Framework umožňuje použít více poskytovatelů dat, u kterých se může zdát, že disponují různou vnitřní strukturou, není tomu tak, neboť každý z poskytovatelů je založen na téže sadě rozhraní a základních tříd. Všechny objekty připojení např. obsahují společné klíčové metody *Open()* a *Close()*. Na pozadí ovšem pracují naprosto odlišná nízkoúrovňová API. Architekturu komunikace ADO.NET demonstruje obrázek (Obr. 6).



Obr. 6. Architektura ADO.NET s využitím DataSetu

1.3.2 Data adaptéry, datasety a práce s nimi

Práce s daty získanými skrze dataset probíhá podle následujícího schématu:

1. Deklarace proměnných pro datatypy SqlConnection, SqlDataAdapter, SqlCommandBuilder, DataSet.
2. Naplnění proměnných, přiřazení obsahu atributu *select* každému DataAdaptéru.
3. Vygenerování dotazů *insert*, *update*, *delete* pro DataAdaptéry. Toto automatické vygenerování není možné provést pro SQL *select* dotazy obsahující jakýkoliv druh JOIN propojení tabulek.
4. Naplnění tabulek DataSetu informacemi z DataAdaptérů.
5. Práce s daty v tabulkách DataSetu.
6. Uzavření *edit módu* v tabulkách DataSetu.
7. Aktualizace dat do databáze skrze DataAdaptéry.

Popsaný průběh práce s výše popsanými prvky ADO.NET demonstruje kód příkladu v obrázku (Obr. 7).

```
1   private SqlConnection SqlConnection1 = new SqlConnection();
2   private SqlDataAdapter daCommunication_container;
3   string strCommunication_container = "SELECT
    communication_container.* FROM communication_container";
4   dset = new DataSet();
5   daCommunication_container = new
    SqlDataAdapter(strCommunication_container, SqlConnection1);
6   SqlCommandBuilder cmdBldrDaCommunicationContainer = new
    SqlCommandBuilder(daCommunication_container);
7   daCommunication_container.Fill(dset,
    "communication_container");
8   // práce s obsahem tabulek DataSetu
9   foreach (DataTable tabulka in dset.Tables)
10  {foreach (DataRow dr in tabulka.Rows) {EndEdit();}}
11  daCommunication_container.Update(dset.Tables["communication_con
    tainer"]);
12  dset.AcceptChanges();
```

Obr. 7. Příklad práce s objekty ADO.NET

1.4 WCF

Windows Communication Foundation (WCF) byla představena jako jedna z novinek .NET Frameworku verze 3.5. Jedná se o komplexní nástroj pro vytváření servisně orientovaných aplikací, které začínají značně nabývat na oblibě a četnosti použití. [2]

Za hlavní přednost WCF se považuje sjednocení technologií Microsoftu používaných pro tvorbu distribuovaných aplikací. Před uvedením WCF museli vývojáři volit mezi několika různými možnostmi přístupu:

- Web Services Enhancements
- ASP.NET Web Services - interoperabilita
- .NET Remoting - výkon
- Enterprise Services - distribuované transakce
- Microsoft Message Queuing (MSMQ) – spolehlivé doručování dat

Každá z technologií se ovšem hodila pro zcela jiné využití, takže díky současnému různorodému síťovému prostředí byla nutnost často kombinovat mezi sebou minimálně dvě různá řešení, kdy například část systému komunikovala kvůli výkonnosti s pomocí .NET Remoting, zatímco jiná využívala Web Services. [6]

1.4.1 Způsob komunikace

Komunikační model WCF rozlišuje mezi klientem a službou. Zatímco klientská aplikace iniciuje komunikaci, služba čeká na zavolání požadavků klientem.

Služby jsou chápány jako základní stavební kámen WCF, díky kterým komunikujeme se vzdáleným systémem pomocí tzv. endpointů. Tyto slouží jako brána, kde dochází k příjmu a vysílání zpráv. WCF služba je tedy okolím vnímána jako kolekce endpointů, přičemž musí vždy existovat minimálně jeden endpoint. Službu WCF tvoří tři základní části:

1. Třída služby implementující poskytované metody služby
2. Hostovací aplikace
3. Jeden nebo více endpointů

Endpoint ve službě znamená, jak již bylo uvedeno, místo sloužící pro příjem a odesílání zpráv. Tvoří ho kombinace tři parametrů:

- **Address** uvádějící, kde se běžící služba nachází a kam mají být zasílána data

- **Binding** v konfiguraci uvádí, jakým způsobem má daná služba a klient komunikovat se svým okolím, tedy jaký má být zvolen komunikační protokol, zda se má použít nějaká metoda bezpečnosti, v jakém kódování bude probíhat komunikace atd.
- **Contract** popisuje rozhraní, jakým služba komunikuje, tzn., uvádí, jaké metody a datatypy služba nabízí. Důležitou vlastností pro použitelnost celé WCF je nezávislost kontraktu na nastavení adresy a protokolu. Podrobnějším popisem druhů kontraktů se zabývá další kapitola.

Definovat nastavení endpointu lze dvěma způsoby. Pomocí GUI nástroje lze deklarativně vygenerovat XML nastavení v App.config aplikace (ilustruje obrázek s kódem (Obr. 8), případně vytvořit nastavení imperativně přímo v kódu aplikace dle příkladu v obrázku (Obr. 9).

```
1   <endpoint address="http://localhost:8080/ControlIT"
2   binding="wsHttpBinding"
3   bindingConfiguration="WSHttpBinding_IControlIT_Service"
4   contract="ControlIT.IControlIT_Service" />
```

Obr. 8. Příklad založení endpointu WCF deklarativně v konfiguraci aplikace

```
1 Uri baseAddress = new Uri("http://localhost:8080/ControlIT");
2 Type serviceType = typeof(ControlIT.IControlIT_Service);
3  Type contractType = typeof(ControlIT.IControlIT_Service);
4 BasicHttpBinding binding = new BasicHttpBinding();
5 ServiceHost host = new ServiceHost(serviceType, baseAddress));
6 host.AddServiceEndpoint(contractType, binding, baseAddress);
```

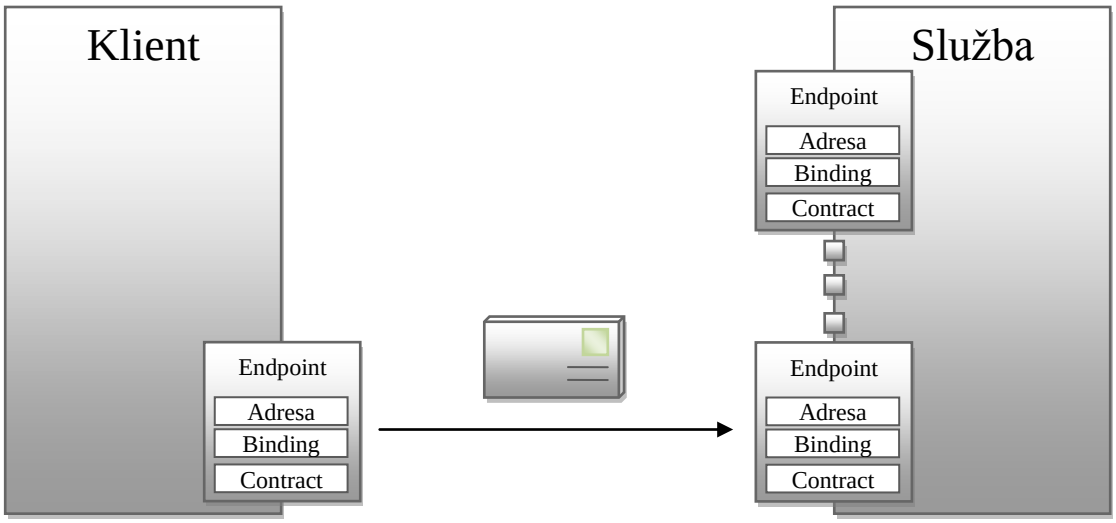
Obr. 9. Příklad založení endpointu WCF imperativně v kódu aplikace

Samotná komunikace klienta a služby probíhá na bázi zasílání **SOAP zpráv**, kdy je tato chápána jako skupina dat obsahující záhlaví a tělo zprávy. Přenos zpráv se vyznačuje nezávislostí na použitém přenosovém protokolu. Ve službách WCF existují tři možné modely zasílání zpráv:

- **One way** – klient odesílá zprávu službě a neočekává odpověď.
- **Request-response** – klient pošle požadavek a čeká na odpověď.

- **Duplex** – zajišťuje obousměrnou komunikaci mezi klientem a službou, přičemž probíhá asynchronně (služba díky duplexní komunikaci může vynutit spuštění metody na straně klienta).

Komunikaci klienta a služby za pomocí zasílání zpráv (metodou one way) popisuje následující obrázek (Obr. 10).



Obr. 10. WCF komunikace klienta a služby

Konfigurace binding ve WCF popisuje způsob přenosu dat mezi klientem a službou. Každý endpoint musí mít pro zajištění své funkčnosti přiřazenu konfiguraci binding. Tato konfigurace se skládá z několika částí:

- Transport (HTTP, TCP, Named Pipes, MSMQ, ...)
- Encoding (binary, text, ...)
- Transakce
- Reliable session
- Security

Při nastavování binding musí vývojář použít vždy povinné položky transport a encoding, zbývající jsou volitelné.

V rámci WCF existuje skupina bindings označovaná jako *system bindings* umožňující rychlé nastavení s vlastnostmi popsanými v tabulce níže (Tab. 4), možnosti použití deklaruje další tabulka (Tab. 5).

Tab. 4. Druhy Bindings a jejich vlastnosti

Binding	Interoperabilita	Bezpečnost	Session	Transakce	Duplex	Encoding
BasicHttpBinding	Basic Profile 1.1	Transport Message Mixed	Ne	Ne		Text
WSHttpBinding	WS	Transport Message Mixed	Transport Reliable Session	Ano		Text
WSDualHttpBinding	WS	Message	Reliable Session	Ano	Ano	Text
WSFederationHttpBinding	WS - Federation	Message Mixed	Reliable Session	Ano	Ne	Text
NetTcpBinding	.NET	Transport Message Mixed	Transport Reliable Session	Ano	Ano	Binary
NetNamedPipeBinding	.NET	Transport	Transport	Ano	Ano	Binary
NetMsmqBinding	.NET	Transport Message	Ne	Ano	No	
NetPeerTcpBinding	Peer	Transport	Ne	Ano	Ano	
MsmqIntegrationBinding	MSMQ	Transport	Ne	Ano		

Tab. 5. Možnosti použití jednotlivých bindings

Binding	Použití
BasicHttpBinding	Komunikace s webovými službami splňujícími WS-Basic Profile
WSHttpBinding	Zabezpečené a interoperabilní propojení bez podpory duplexních kontaktů
WSDualHttpBinding	Zabezpečené a interoperabilní propojení s podporou duplexních kontaktů
WSFederationHttpBinding	Podpora protokolu WS-Federation
NetTcpBinding	Zabezpečené a optimalizované binding pro komunikaci WCF aplikací po síti přes TCP protokol
NetNamedPipeBinding	Komunikace WCF aplikací na úrovni jediného počítače – lokální uzavřená komunikace
NetMsmqBinding	Komunikace zprostředkovaná přes MSMQ
NetPeerTcpBinding	Síťová komunikace mezi více počítači na úrovni rovný k rovnému
MsmqIntegrationBinding	Komunikace mezi aplikací WCF a existující aplikací MSMQ

1.4.3 WCF Contracts

Zatímco WCF bindings uvozují způsob, jak přenášet data mezi klienty a službou, WCF contrats mají na starosti definování rozhraní oslovované služby a popis datové struktury, která se mezi účastníky komunikace přenáší. Ve WCF pak existují tři druhy kontraktů:

- **Service contracts** – kontrakty služeb

- **Data contracts** – kontrakty dat
- **Message contracts** – kontrakty zpráv

Service contract popisuje operace vykonávané službou. Uvození se provádí přidáním atributu *[ServiceContract]* k deklaraci třídy či rozhraní, které se mají stát přístupné pro potřeby vzdáleného volání služby. Samotné operace třídy (metody) definuje atribut *[OperationContract]* přidělený k definici metody třídy nebo rozhraní. Příklad definice kontraktu služby demonstruje zdrojový kód v obrázku (Obr. 11). [6]

```
1 [ServiceContract]
2 public interface IControlIT_Service
3 {
4     [OperationContract]
5     string DoWork();
6     [OperationContract]
7     ControlIT_Service_MeasuringBlock[] getMeasuringBlocks(int
      idCommunicationContainer);
8 }
```

Obr. 11. Příklad deklarace Service contract

Data contract definuje strukturu přenášených dat. Jestliže se snažíme přenést z nebo do služby jiný datový typ, než jsou ty základní (int, float, string,…), musíme službě uvést definici data contract. Třída definující datovou strukturu je používaná častěji než message contracts, navíc ji často využívá více služeb.

Data contract se označuje atributem *[DataContract]* přidaným před deklaraci třídy. Jednotliví členové datové struktury se následně definují pomocí položky *[DataMember]*. Příkladnou definici Data contract představuje kód uvedený v obrázku níže (Obr. 12). [6]

```
1 [DataContract]
2  public class ControlIT_Service_Digital
3 {
9     int id = 0;
10    string name = "none";
11    [DataMember]
12    public int Id
13    {
14       get { return id; } set { id = value; }
15    }
```

```
16     [DataMember]
17     public int Id_measuring_block
18     public string Name
19     {
20        get { return name; } set { name = value; }
21     }
22 }
```

Obr. 12. Příklad deklarace Data contract

Message contract se moc neodlišuje od předchozího Data contract, neboť popisuje též strukturu – zprávy. Tato třída reprezentující zprávu se dělí na části s požadavkem a odpovědí. Zatímco Data contract jsou určeny pro obecný přenos dat v datových strukturách, Message contract byly určeny pro specifickou operaci služby, není je možno znovupoužít.

Message contract uvozuje třídu pomocí atributu *[MessageContract]*, přičemž jednotlivé části zprávy definujeme pomocí *[MessageHeader]* a *[MessageBodyMember]*. Obrázek s kódem (Obr. 13) představuje příklad použití tohoto kontraktu. [6]

```
 1 [MessageContract]
 2 public sealed class MeasuringAlert
 3 {
 4     private string alertName;
 5     private string alertBody;
 6     [MessageHeader]
 7     public string AlertName
 8     {
 9        get { return alertName; } set { alertName = value; }
10     }
11     [MessageBodyMember]
12     public string AlertBody
13     {
14        get { return alertBody; } set { alertBody = value; }
15     }
16 }
```

Obr. 13. Příklad deklarace Message contract

1.4.4 Autentizace a bezpečnost ve WCF

WCF disponuje širokým spektrem mechanismů zajišťujících ochranu dat při přenosu a přístup do služeb. Tyto mechanismy lze rozdělit do tří základních kategorií:

- **Auditing** – logování důležitých událostí
- **Transfer security** – zabezpečení přenosu
- **Access control** – zajištění kontroly přístupu

Zabezpečení přenosu mezi službami a klienty WCF lze provést dvěma následujícími způsoby:

- **Transport security mode** – bezpečnost se implementuje na úrovni přenosového protokolu, který většinou zastává HTTPS.
- **Message security mode** – zabezpečení proběhne na úrovni SOAP zpráv, například pomocí WS-Security.

WCF standardně podporuje několik druhů ověřování, mezi něž patří tyto metody:

- **UserName tokens** - B2C aplikace
- **Windows autentizace** - intranetové aplikace
- **Certifikáty** - B2B aplikace
- **CardSpace** - B2C aplikace
- **Credentials (Security Token Service)** - B2B aplikace

Příklad využití Windows Autentizace demonstruje zápis obrázku (Obr. 14), který deklaruje nastavení úrovně zabezpečení na vrstvě přenosu a zprávy:

```
1 <security mode="Message">
2    <transport clientCredentialType="Windows"
     proxyCredentialType="None" realm="" />
3    <message clientCredentialType="Windows"
     negotiateServiceCredential="true" algorithmSuite="Default"
     establishSecurityContext="true" />
4 </security>
```

Obr. 14. Příklad Windows autentizace pro zabezpečení intranetové aplikace

1.5 ASP.NET

Vývojáři Microsoftu popsali ASP.NET jako svou šanci „odeslat příkaz pro zformátování systému“ a začít se zcela novým modernějším vývojovým modelem. Tradiční pojmy

týkající se vytváření webových aplikací však i s příchodem ASP.NET stále platí. Webové aplikace se stále skládají z webových stránek, kdy můžeme zpracovávat bohatě vybavený HTML, používat JavaScript, vytvářet komponenty a jiné. V pozadí však ASP.NET pracuje odlišně od stávajících tradičních skriptovacích technologií, mezi něž patří klasické ASP nebo PHP. [3]

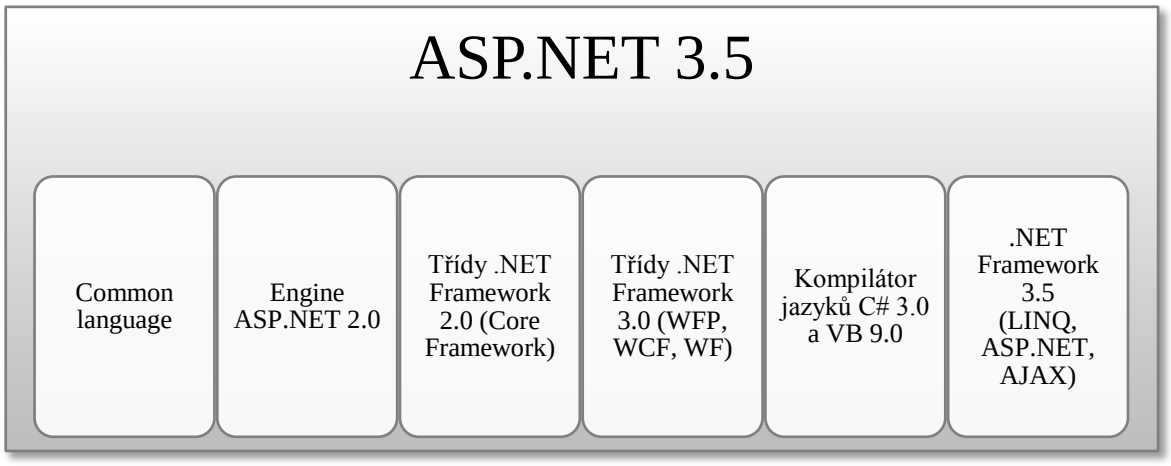
ASP.NET se od ostatních platforem odlišuje v několika základních skutečnostech:

- ASP.NET přináší nový, zcela objektově orientovaný programovací model obsahující architekturu řízenou událostmi, jež je založena na ovládacích prvcích, což umožňuje zapouzdření kódu a jeho následné opětovné využívání.
- Kód pro ASP.NET lze psát v jakémkoliv podporovaném jazyku .NET, což ulehčuje programátorům práci s přechodem na vývoj webových aplikací.
- Díky kompilaci na požádání stránky ASP.NET a komponenty dosahují vysokého výkonu zpracování. ASP.NET umožňuje flexibilní ukládání dat do cache a obsahuje vyladěný model pro přístup k datům.
- ASP.NET lze provozovat na různých operačních systémech i webových serverech, např. IIS (Windows), Apache (Windows, Linux s open source implementací .NETu Monem)

Aplikace ASP.NET se vždy kompilují, přičemž procházejí dvěma kompilačními etapami. Nejprve se kód některého z programovacích jazyků zkompiluje do přechodového jazyka MSIL. Tento první krok může nastat automaticky při prvním vyvolání webové stránky, případně pomocí předběžné kompilace.

Druhý krok kompilace nastává těsně před skutečným vyvoláním webové stránky. Tím se kód MSIL zkompiluje do nativního nízkoúrovňového strojového kódu. [7]

Během psaní diplomové práce nejnovější verze ASP.NET 3.5 v podstatě neobsahovala žádnou novou verzi ASP.NET. V podstatě se jedná o rozšířenou sadu funkcí přidanou k ASP.NET 2.0, přičemž části, které se v novější verzi nezměnily jsou označován jako *red bits*, zatímco změněné dostaly terminologii *green bits*. Strukturu komponent ASP.NET popisuje následující obrázek (Obr. 15).



Obr. 15. Struktura komponent ASP.NET 3.5

1.5.1 Struktura ASP.NET aplikace

Kompletní strukturu webového projektu zobrazuje a umožňuje s ní pracovat panel Solution explorer v prostředí Visual Studio. Každý z projektů obsahuje nebo může obsahovat následující součásti: [7]

- **App_Data** funguje jako složka pro uložení souborů s daty, které nejsou přístupná z vnějšího prostředí. Do této složky smí tedy přistupovat jen samotná aplikace, a tak ji použít např. pro uložení databází, souborů s hesly a další činnosti s podobným účelem.
- **App_Themes** obsahuje podsložky s jednotlivými tématy webové aplikace. Krom definic kaskádových stylů CSS může obsahovat složku s obrázky užitými v tématu, případně soubor *Skin.skin* definující jednotný vzhled pro všechny ovládací prvky ASP.NETu.
- **App_Code** slouží pro uložení všech sdílených tříd aplikace.. Kdykoliv dojde ke změně v adresáři App_Code, ASP.NET dynamicky překompiluje všechny uložené třídy a automaticky poskytne k dispozici pro všechny stránky webové aplikace.
- **Bin** zastává funkci úložiště pro již zkompilované soubory (dll, exe), které mají být využity v ASP.NET aplikaci.
- **Soubor Web.config** zastává v každém ASP.NET projektu velmi důležitou funkci. Stejně jako App_Data není z vnějšího prostředí přístupný, protože obsahuje konfiguraci celé webové aplikace. Pomocí *web.config* lze provádět následující nastavení:
    - Nastavení uživatelských práv

- Povolení přístupů uživatelů k jednotlivým částem webu
- Nastavení údajů pro připojení k databázi
- Konfigurace providerů
- Inicializace knihoven a zdrojů

Díky struktuře konfiguračního souboru založeného na formátu XML lze k jednotlivým parametrům jednoduše přistupovat a měnit je.

- **Soubory *.aspx** představují jednotlivé webové stránky aplikace. Mohou obsahovat kompletní kód včetně aplikační logiky, ovšem vhodnější je tuto aplikační logiku umístit do souboru typu ***.aspx.cs**. Příklad hlavičky zdrojového kódu webové stránky aspx demonstruje jednoduchý příklad kódu (Obr. 16).

```
1 <%@ Page Title="" Language="C#"
  MasterPageFile="~/MasterPage.master" AutoEventWireup="true"
  CodeFile="add_measuring_block.aspx.cs"
  Inherits="add_measuring_block" %>
```

Obr. 16. Hlavička kódu souboru aspx

- **MasterPage.master** patří svým obsahem mezi běžné aspx stránky s tím rozdílem, že obsahuje specifické tagy pro vnoření dynamického obsahu – jiných aspx stránek. Statické prvky (logo, menu, patička) mohou být tak editovány v jediném souboru, ovšem změny se projeví ve všech stránkách, které se do masterpage vnořují.

1.5.2 Práce s MasterPage

Téměř všechny webové aplikace disponují jednotným uživatelským rozhraním. Webový layout lze rozdělit na bloky loga, menu, patičky a především hlavní část s dynamicky vkládaným obsahem vyžádaným návštěvníkem webu. Tohoto rozdělení sestavení stránky se docílí několika způsoby:

- Využití rámců se zcela autonomními webovými stránkami, jež se mohou navzájem ovlivňovat.
- SSI (Server Side Includes) umožňuje bezrámcové propojení na straně serveru.
- Vkládání dynamického obsahu do standardizovaného layoutu pomocí jakékoliv skriptovací technologie na straně serveru (PHP, ASP.NET).

ASP.NET přistupuje k problému rozložení layoutu pomocí MasterPages. Jedná se o speciální stránku začínající obvyklými značkami *<!DOCTYPE...* a *<html>*, končí tagem *</html>*. Do layoutu definovaného v MasterPage lze pomocí komponenty

ContentPlaceHolder dosadit dynamický obsah, který se k ní bude vázat. Obsah MasterPage může vypadat podobně, jako v kódu na obrázku (Obr. 17).

```
1  <%@ Master Language="C#" AutoEventWireup="true"
   CodeFile="MasterPage.master.cs" Inherits="MasterPage" %>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
3  <script runat="server">
4  </script>
5  <html xmlns="http://www.w3.org/1999/xhtml" >
6  <head runat="server">
7      <title>Untitled Page</title>
8  </head>
9  <body>
10     <form id="form1" runat="server">
11     <div>
12        <asp:contentplaceholder id="ContentPlaceHolder1"
          runat="server">
13        </asp:contentplaceholder>
14     </div>
15     </form>
16 </body>
17 </html>
```

Obr. 17. Obsah MasterPage

Kód webové stránky, jež chceme do MasterPage vložit musí samozřejmě splňovat jistá pravidla, především obsahovat komponentu *Content* obsahující atribut *ContentPlaceHolderID* s názvem place holderu uvedeného v MasterPage na místě, kam chceme dynamický obsah vložit. Přehlednější představu o propojení si lze udělat díky příkladu v obrázku s kódem (Obr. 18).

```
1  <%@ Page Title="" Language="C#"
   MasterPageFile="~/MasterPage.master" AutoEventWireup="true"
   CodeFile="add_measuring_block.aspx.cs"
   Inherits="add_measuring_block" %>
2  <asp:Content ID="Content1"
   ContentPlaceHolderID="ContentPlaceHolder1" Runat="Server">
3     <!— kód vložený na místo komponenty ContentPlaceHolder1 -->
4  </asp:Content>
```

Obr. 18. Webové stránka vložitelná do MasterPage

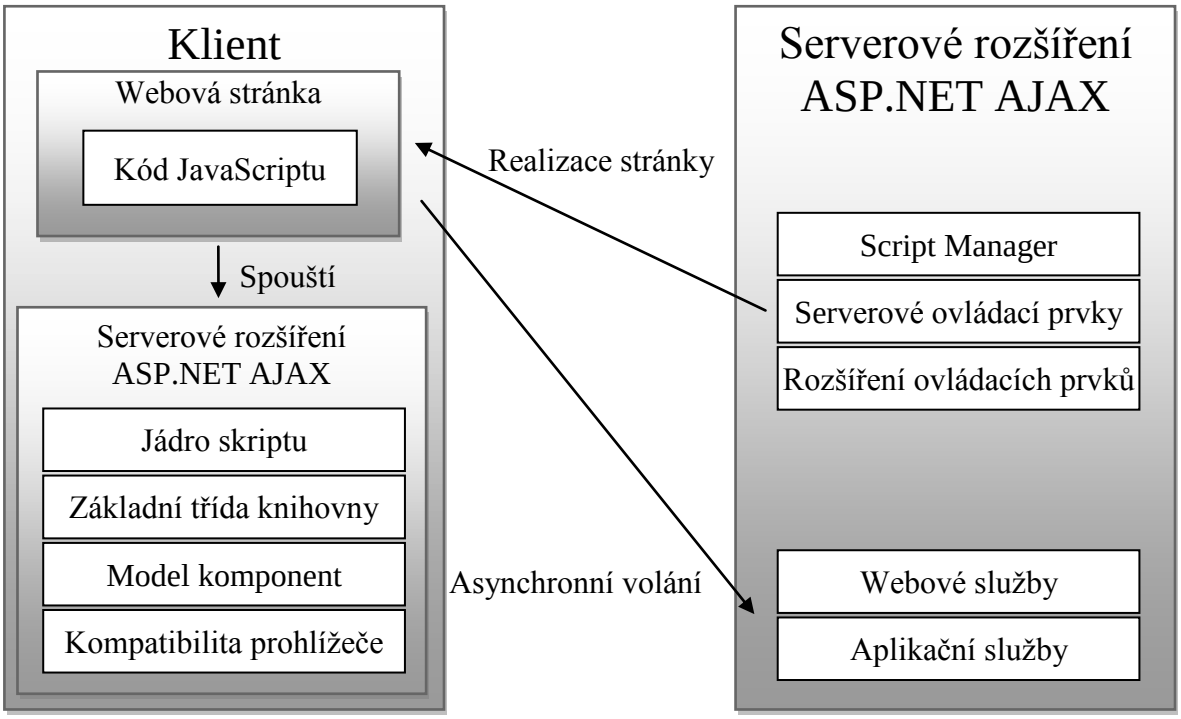
1.5.3 AJAX v ASP.NET 3.5

AJAX (Asynchronous JavaScript and XML) zná internetová veřejnost jako technologii pro vývoj interaktivních webových aplikací měnících svůj obsah bez nutnosti jejich opětovného načtení. Pomocí technologie AJAX se tedy dociluje uživatelsky příjemnějšího prostředí a více interaktivní webové aplikace. Podmínkou bezchybného běhu AJAXu je ovšem použití moderního webového prohlížeče. [8]

AJAX aplikace lze vyvíjet pomocí následující skupiny technologií:

- **HTML** (nebo **XHTML**) a **CSS** prezentující informace
- **DOM** a **JavaScript** pro zobrazování a dynamické změny obsahu informací
- **XMLHttpRequest** pro asynchronní výměnu dat s webovým serverem

Základním stavebním blokem ASP.NET AJAX jsou knihovny JavaScript na straně prohlížeče klienta. Obrázek (Obr. 19) popisuje funkci knihoven a komunikace AJAXu:



Obr. 19. Architektura ASP.NET AJAX

S příchodem ASP.NET 3.5 se pro webové vývojáře na této platformě rapidně zjednodušil vývoj AJAX aplikací, neboť do palety nástrojů přibylo několik komponent, původně obsažených v ASP.NET AJAX Extensions, jež umožňují využívat výhody AJAXu na pár kliknutí.

Jedná se o tyto komponenty:

- **ScriptManager** – engine serverového modelu ASP.NET AJAX realizující odkazy na knihovny ASP.NET AJAX – generuje a odkazuje na URL adresy, které jsou přiřazeny skriptovým zdrojům.
- **ScriptManagerProxy** – zastává funkci ScriptManageru na vložené stránce. Zatímco ScriptManager lze vložit např. do MasterPage a jednotně nakonfigurovat, některé vložené stránky mohou mít potřebu rozdílného nastavení manažeru oproti globálnímu z MasterPage, to zajišťuje práve ScriptManagerProxy.
- **UpdatePanel** – poskytuje jako serverový ovládací prvek důležitou funkci. Umožňuje převzít část běžného obsahu stránky i serverovou logiku a zajistit jejich asynchronní obnovení ve stylu AJAXu.
- **UpdateProgress** – funguje jako indikátor probíhající aktualizace dat. V případě právě probíhající aktualizace změní svůj atribut visibility na viditelný a zobrazí svůj obsah, kterým může být prakticky cokoliv od textu po animaci.
- **Timer** – obnovuje části AJAX bloků bez nutnosti vyvolání akce uživatelem. Atribut *Interval* určuje čas v milisekundách pro prodlevu mezi aktualizacemi. Během vyvolání události tiku časovače lze provést navíc jakoukoliv programovou akci.

Příkladné využití komponent AJAX, tedy prvky ScriptManager, Timer a UpdatePanel demonstruje následující kód (Obr. 20):

```
 1  <asp:ScriptManager ID="ScriptManager1" runat="server">
 2 </asp:ScriptManager>
 5 <asp:UpdatePanel ID="UpdatePanelForm" runat="server">
 6 <ContentTemplate>
 7    <asp:Timer ID="Timer1" runat="server" Interval="1000"
      ontick="Timer1_Tick">
 8    </asp:Timer>
 9    <!-- obsah aktualizovaný AJAXem -->
10 </ContentTemplate>
11 </asp:UpdatePanel>
```

Obr. 20. Využití AJAXu

Širokou paletu užitečných komponent využívajících všechny přednosti AJAXu přináší projekt ASP.NET AJAX Control Toolkit[1], jehož prvky lze jednoduše implementovat do toolboxu Visual Studia. Rozšíří se tím tak znatelně možnosti programátora, který následně může používat množství zajímavých AJAX objektů:

- **Accordion** – animované rozbalování a zabalování div bloků
- **Animation** – animovaná vyskakovací okna
- **AutoComplete** – automatické dokončování a nabízení možných variant textového vstupu
- **Calendar** – skvěle vyřešený dynamický kalendář plně nahrazující stávající ASP.NET calendar
- **DragPanel** – okno posouvatelné kursorem myši po ploše webové stránky
- **Slider** – objekt nastavující číselnou hodnotu za pomocí šoupátka známého z WinForms
- **Tabs** – objekt záložek formuláře podobný objektu ve WinForms
- **ValidatorCallout** – validační prvek s možností výpisu chybového hlášení formou „bubliny“
- Desítka dalších zajímavých objektů

[1] Domovské stránky projektu ASP.NET AJAX Control Toolkit na http:/ajax.asp.net/ajaxtoolkit

2 VHODNÝ HARDWARE PRO REALIZACI PROJEKTU

Před započetím tvorby celého projektu se uvažovaly různé varianty měřícího hardware. Zapojení vyžadovalo krom konkrétních technických požadavků podrobněji popsaných v další části práce především jednoduchost, flexibilitu a finanční náklady ve vyrovnaném poměru cena / výkon. Po průzkumu nabídky trhu byl pro použití zvolen monitorovací kit Velleman K8055.

2.1 Měřící deska Velleman K8055

Distribuce měřící desky Velleman K8055[2] probíhá většinou ve formě nezkompletované elektrotechnické stavebnice, jejíž součástí jsou následující komponenty:

- Neosazená deska plošných spojů
- Diskrétní součástky vlepené do pásku dle postupu pájení
- Integrované obvody včetně naprogramovaného mikrokontroléru
- Propojovací USB kabel
- Manuál a CD s drivery

2.1.1 Popis I/O desky

Přestože dle slov výrobce najde měřící deska (Obr. 21) využití pouze „v domácnosti nebo kanceláři“, lze ji použít pro široké spektrum více či méně profesionálních aplikací. Verze K8055 disponuje následujícími parametry:

- 5 digitálních vstupů (0 = zem, 1 = rozpojeno). Vybaveno zkušebními tlačítky na desce.
- Analogové vstupy s volbou útlumu nebo zesílení.
- Vybaveno vnitřním zkušebním napětím +5V.
- 8 digitálních výstupních spínačů s otevřeným kolektorem (max. 50V / 100mA). Indikace LED na desce.
- 2 analogové výstupy:
    - 0 ~ 5V, výstupní odpor 1,5kΩ

[2] Webová prezentace produktu: http://www.vellemanusa.com/us/enu/product/view/?id=500349# , katalog českého prodejce: http://www.gme.cz/cz/index.php?page=product&detail=764-367

  - 0 ~ 100% výstup PWM s otevřeným kolektorem max. 100mA / 40V
  - Indikace úrovně analogového výstupu pomocí LED na desce
- Průměrný čas konverze: 20ms / příkaz
- Možnost napájení přes USB cca 70mA
- Kompatibilní se standardem USB 2.0 a 1.1
- Přiložený diagnostický software a komunikační DLL

Obr. 21. Měřící deska K8055

Schéma zapojení, vyobrazení a osazení desky plošného spoje se nachází v příloze PI – PIII.

2.1.2 Popis knihoven pro řízení karty

K zařízení výrobce dodává kolekci DLL knihoven, díky kterým lze jednoduše volat funkce programu v následujících programovacích jazycích:

- Jazyky podporované pro Visual Studio
- Borland Delphi
- Borland C++ Builder

Knihovny (*K8055D.dll* a *K8055D_C.dll*) došly v průběhu vypracování projektu k revizi 3.0, přičemž oproti předcházejícím verzím plně podporují systém Windows Vista. Běh jakékoliv aplikace využívající funkce zařízení vyžaduje přítomnost uvedených knihoven ve stejném adresáři, jako aplikace samotná, případně v Jednotka:\WINDOWS\system32.

DLL knihovny pak obsahují funkce uvedené v tabulce (Tab. 6).

Tab. 6. Funkce v DLL knihovnách pro měřící desku K8055

Funkce	Popis funkce
Hlavní funkce zařízení	
OpenDevice(int CardAddress)	Otevře komunikační linku do zařízení K8055 na určité adrese (nastavené pomocí jumperů přímo na zařízení)
CloseDevice()	Zavře všechny otevřené linky
SearchDevices()	Vrací adresu v int nalezeného měřícího zařízení
Funkce pro čtení analogových vstupů	
ReadAnalogChannel(int ChannelNo)	Přečte měřenou úroveň analogovém vstupu na vybraném kanále. Výstup je int v intervalu <0;255>
ReadAllAnalog(int Data1, int Data2)	Přečte měřené úrovně na obou analogových vstupech a uloží do int proměnných v intervalu <0;255>
Funkce pro zápis do analogových výstupů	
OutputAnalogChannel(int ChannelNo, int Data)	Zapíše hodnotu z intervalu <0;255> na zvolený analogový výstup
OutputAllAnalog(int Data1, int Data2)	Zapíše hodnoty z intervalu <0;255> na oba analogové výstupy
ClearAnalogChannel(int ChannelNo)	Nastaví zvolený analogový výstup na minimum
ClearAllAnalog()	Nastaví oba analogové výstupy na minimum
SetAnalogChannel(int ChannelNo)	Nastaví zvolený analogový výstup na maximum
SetAllAnalog()	Nastaví oba analogové výstupy na maximum
Funkce pro čtení digitálních vstupů	
ReadDigitalChannel(int ChannelNo)	Přečte stav digitálního vstupu na zvoleném kanále, vrací datatyp bool
ReadAllDigital(int Buffer)	Vrací celočíselně stav všech digitálních vstupů
Funkce pro zápis digitálních výstupů	
WriteAllDigital(int Data)	Převede z int v intervalu <0;7> na jednotlivé bity, které zapíše na digitální výstupy. Vyšší bity = vyšší kanály výstupů
ClearDigitalChannel(int ChannelNo)	Nastaví zvolený digitální výstup na 0
ClearAllDigital()	Nastaví všechny digitální výstupy na 0
SetDigitalChannel(int ChannelNo)	Nastaví zvolený digitální výstup na 1
SetAllDigital()	Nastaví všechny digitální výstupy na 1
Funkce čítače	
ResetCounter(int CounterNo)	Vyresetuje čítač dle vstupu
ReadCounter(int CounterNo)	Načte aktuální stav 16-bitového čítače
SetCounterDebounceTime(int CounterNo, int DebounceTime)	Nastaví prodlevu na čítači do dalšího tiku v intervalu <0;5000>

2.2 Měřící deska Velleman K8061

V případě opravdu profesionálního využití výrobce doporučuje použít měřící zařízení K8061[3] s následujícími parametry:

[3] Webová prezentace produktu: http://www.vellemanusa.com/us/enu/product/view/?id=522974#

- 8 digitálních vstupů (0 = zem, 1 = rozpojeno)
- 8 analogových vstupů s 10bitovým rozlišením A/D převodníku v rozsahu 0 ~ 5V, případně 10V, impedance 20kΩ
- 8 analogových výstupů s 8bitovým rozlišením D/A převodníku v rozsahu 0 ~ 5V, případně 10V, impedance 47kΩ
- 8 digitálních výstupních spínačů s otevřeným kolektorem (max. 50V / 100mA). Indikace LED na desce.
- PWM module s rozlišením 10bitů, s otevřeným kolektorem (max. 100mA / 40V). Indikace LED na desce.
- Průměrný čas konverze: 4ms / příkaz
- Galvanicko-optickému oddělení datové komunikace zajištující ochranu počítače při případné poruše měřící karty.
- Kompatibilní se standardem USB 2.0 a 1.1
- Možnost napájení přes USB cca 60mA
- Přiložený diagnostický software a komunikační DLL
- Možnost připojení až osmi měřících zařízení k jednomu počítači
- Možnost dodání zkompletovaného zařízení pod označením VM140

Obr. 22. Měřící deska K8061

Obsah a funkce jednotlivých metod obslužných knihoven zařízení K8061 se podobají driverům pro zařízení K8055.

3 SOUČASNÝ STAV ŘÍZENÍ A MONITOROVÁNÍ VYSÍLAČŮ

Stávající řešení bylo do praxe nasazeno již před několika lety, přičemž jej vedení rozhlasové stanice stále považuje za funkční a spolehlivé, přesto ovšem díky prudkému vývoji v informačních technologiích přestává v jistých ohledech vyhovovat.

3.1 Současné možnosti měření a řízení

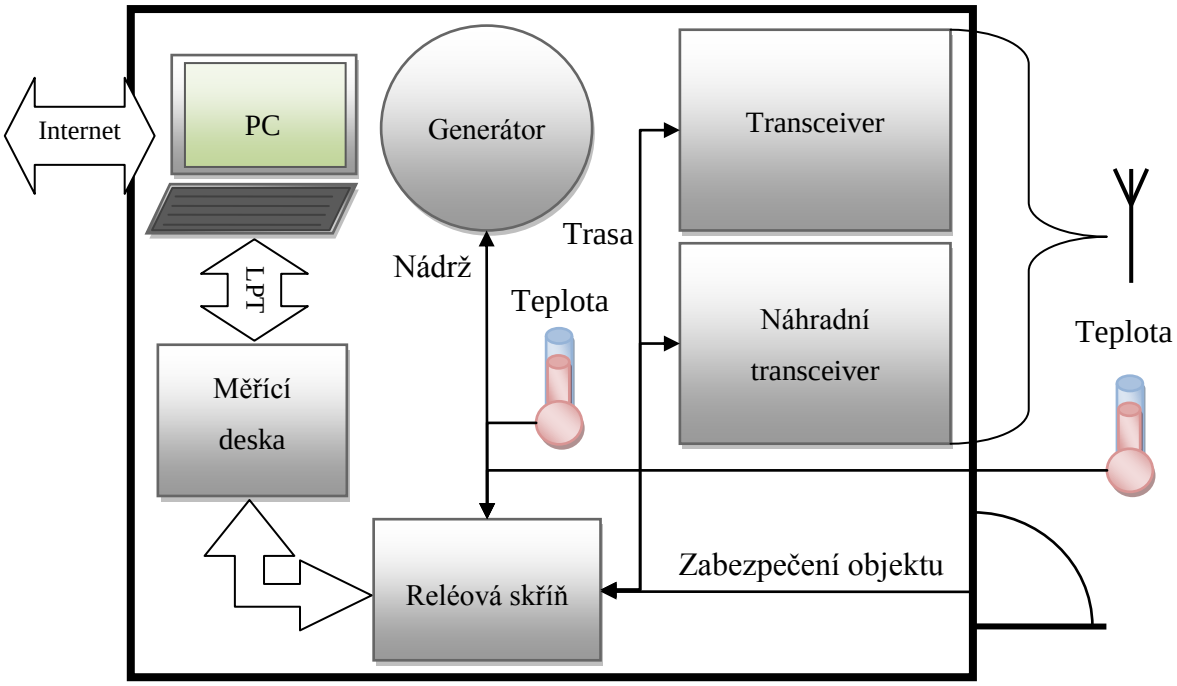
Měřící systém použitý pro správu a měření rozhlasového vysílače zastává několik stěžejních funkcí, které popisuje tabulka níže (Tab. 7).

Tab. 7. Využití současného měřícího systému

Funkce	Typ veličiny	Popis
Stav síťového napájení	Digitální vstup	Kontroluje stav zapnutí síťového napájení
Stav generátoru	Digitální vstup	Kontroluje stav chodu generátoru elektrického proudu
Stav vysílačů	Digitální vstup	Kontroluje stav zapnutí vysílacích prvků
Překročení teploty	Digitální vstup	Snímá stav relé na výstupu teplotního čidla nastaveného na určitou mezní teplotu
Poplach	Digitální vstup	Spínací prvek na vstupu dveří – zajištění objektu
Doba běhu generátoru	Digitální vstup	Čítá dobu běhu elektro generátoru. Dle této doby a velikosti nádrže s naftou se určuje objem zbývajícího paliva. Jedná se o tzv. nepřímé měření
Zapnutí trasy	Digitální výstup	Zapne či vypne vysílací prvky
Zapnutí náhradního vysílače	Digitální výstup	Zapne či vypne náhradní vysílací prvky
Venkovní teplota	Analogový vstup	Vstup z čidla měřícího teplotu vně budovy
Vnitřní teplota	Analogový vstup	Vstup z čidla měřícího teplotu uvnitř budovy
Výkon vysílače	Analogový vstup	Vstup informací o výkonu vysílače. Tyto informace dodává přímo transceiver, výkon je nastaven napevno, měření se nevyužívá
Skoková regulace výkonu	Analogový výstup	Řízení výkonu transceiveru. Podle nařízení musí být výkon neměnný, proto se tohoto prvku nevyužívá

3.2 Popis stávajícího měřícího zařízení

Současný systém pracuje na principu zpracování dat počítačem z připojené měřící desky. Schematicky celý proces měření a řízení popisuje obrázek (Obr. 23). Měřící deska byla ve své době nabízena firmou Velleman jako stavebnice. Oproti současným podobným produktům komunikuje s počítačem pomocí dnes již nepoužívaného LPT portu. Společně se stavebnicí výrobce dodával kolekci knihoven umožňující volání funkcí na kartě.



Obr. 23. Zapojení stávajícího měřícího zařízení

3.3 Popis stávajícího měřícího software a přístup k němu

Stávající software byl vytvořen v prostředí Microsoft Visual Basic 6.0. Představuje ho jednoduché GUI prostředí, jehož hlavní formulář disponuje přesným počtem ovládacích prvků přiřazených k určitému měřícímu portu. Funkčně program disponuje následujícími vlastnostmi:

- Nastavení portů (přepínač IN / OUT, VYP / ZAP, popiska)
- Ovládání stavů na portech
- Čtení situace na portech
- Logování situace a chyb
- Hlášení o závadách a předdefinovaných stavech určitému počtu osob skrze e-mail

3.4 Nedostatky stávajícího řešení

Stávající softwarové a hardwarové řešení nevyhovuje v několika základních aspektech, které ovšem přispěly k myšlence znovu vytvořit nový měřící systém zcela od základu. Za hlavní nedostatky se považuje:

- Minimální univerzálnost řešení, kdy jedna aplikace musí být vytvořena přesně na míru konkrétnímu měřícímu zařízení. Možnost jednoduchého rozšíření o další hardware v rámci jednoho řídícího programu tedy neexistuje.
- Absence webového rozhraní stávajícího řešení přináší nutnost vzdáleného připojení k aplikaci skrze vzdálenou plochu. Tento stav komunikace ovšem zvyšuje náročnost na šířku komunikačního kanálu, což přehledně demonstruje obrázek (Obr. 24).
- Aplikace používaná v současné době neumožňuje logování stavů na portech. Z dat tedy nelze např. vykreslovat grafy ani provádět jiné možnosti jejich zpracování.
- Současné know – how je hardwarově založeno na komunikaci s počítačem skrze port LPT, což popisuje i kapitola 3.2. Toto řešení, přestože stále plně funguje, působí v dnešní době již poněkud archaicky. Struktura kabeláže a konektoru LPT navíc nepůsobí tak odolně jako rychlé a jednoduché rozhraní USB. Výrobci desek i vývojáři od využití tohoto rozhraní postupně upustili, navíc společnost, stojící za výrobou stávajícího typu monitorovacího zařízení, již tento model delší dobu nedodává na trh, tudíž by v případě poruchy bylo zcela jistě nutné přeprogramovat řídící aplikaci pro použití s jiným modernějším zařízením.
- Aplikace pro řízení hardware si svým GUI koncepčně zakládá na vlastnostech Windows API, vytvořena pak byla v programovacím jazyku Microsoft Visual Basic 6.0. I tato softwarová technologie se dnes bohužel považuje za překonanou, přičemž zdaleka nenabízí možnosti současných programovacích jazyků a nástrojů. Další vývoj aplikace by se tedy zakládal na použití zastaralých nástrojů a metod, popřípadě by muselo dojít k ne vždy dokonalé konverzi zdrojového kódu.
- Další rozdíly a nedostatky probírá v rámci srovnání s novým řešením kapitola 7.1 na straně 87.

II. PRAKTICKÁ ČÁST

4 NÁVRH A ANALÝZA POTŘEBNÝCH VLASTNOSTÍ SYSTÉMU

Za prvopočátek každého projektu se vždy považuje důkladná analýza požadavků, funkcí a parametrů, kterými má výsledný produkt disponovat. Podrobná analýza vstupních požadavků proběhla i v případě diplomového projektu - měřícího systému ControlIT.

4.1 Definice pojmů

V následujících kapitolách se vyskytuje několik zcela specifických pojmů, které si pro přesnější představu čtenáře zaslouží podrobnější vysvětlení, jež definuje tabulka níže (Tab. 8):

Tab. 8. Definice pojmů pro měřící systém

Pojem	Definice
Měřící zařízení	Hardware obsahující kolekci měřících bodů. Skupina měřících bodů
Měřící blok	Obdobný název pro měřící zařízení
Měřící bod	Prezentuje vstup či výstup z měřícího zařízení určitého typu
Měřící port	Obdobný název pro měřící bod

4.2 Základní požadavky na systém

Ještě před započetím veškeré práce byla položena základní otázka, zda je potřeba tvořit zcela nový měřící systém a jak moc bude oproti stávajícímu řešení (popsanému v kapitole 3) přínosný. Během dlouhé diskuse tak byla stanovena následující základní kritéria a parametry:

- Možnost univerzálního měření a ovlivňování stavu jakékoliv veličiny dodaným monitorovacím hardware.
- Nezávislost systému na použitém měřícím hardware, maximální separace ovládacího hardware a software.
- Možnost přístupu uživatelské obsluhy skrze webové prostředí.
- Modulová přehlednost a univerzálnost řešení.
- Zajištění logování změn měřených veličin do grafu, možnost zpracování dat o měření.
- Relativně rychlá odezva systému, nikoliv však realtime.

4.3 Přístup uživatele k systému

Stávající systém umožňoval přístup pouze přes rozhraní jednoduché GUI aplikace, jež byla přístupná jen na desktopu počítače s měřící kartou. Komunikace s danou aplikací v rámci sítě se tak stala možnou pouze díky spojení pomocí software pro zobrazení a práci se vzdálenou plochou daného stroje, což přinášelo řadu omezení a nepohodlí. Navíc běžící software pro vzdálenou plochu většinou citelně vytěžuje přenosovou linku, jejíž kapacita by se jistě dala využít daleko efektivněji. V některých případech (např. spojení přes mobilní sít GPRS či EDGE) by nemuselo navíc být spojení vůbec možné.

Nový systém tedy musí zcela logicky přenést veškerou vizualizaci stavu měřených veličin a administrační prostředí na jeden centrální prvek s dostatečnou komunikační kapacitou, díky čemuž zajistí znatelné odlehčení komunikační lince mezi tímto prvkem a jednotlivými měřícími zařízeními.

Stavy měřících bodů u nového systému by měly být editovatelné minimálně pomocí dvou vizualizačních rozhraní:

- **Desktopová aplikace** s maximálním možným využitím všech vhodných prvků GUI implementovaného v .NET Framework knihovnách (Windows Forms). Uživateli by mělo toto rozhraní poskytnout především rychlé administrační prostředí pro následující funkce:
  - Prvotní vytvoření a následná správa měřících zařízení a bodů systému
  - Spouštění a zastavování jednotlivých služeb měřícího systému
  - Vizualizace naměřených dat
  - Správa uživatelů a pravidel varovných hlášení
  - Nastavení možností a parametrů aplikace
- **Webová aplikace** přináší především výhodu možnosti vstupovat do systému v podstatě z jakéhokoliv místa připojeného do sítě Internet. Krom využití všech možností prvků XHTML by měla webová aplikace díky technologii AJAX znatelně zvýšit v jistých ohledech uživatelský komfort - ideálně na úroveň možností použití desktopové aplikace. Webové rozhraní musí umožňovat minimálně tyto úkony:
  - Vytvoření a správa měřících zařízení a bodů systému
  - Vizualizace naměřených dat
  - Správa uživatelů a pravidel varovných hlášení

V rámci zajištění bezpečnosti přístupu k administračnímu rozhraní byla pro jednotlivé typy stanovena různá bezpečností pravidla:

- Autentizaci přístupu k **desktopové aplikaci** zajišťuje logování uživatele přes Windows Autentization, kdy každý uživatel, který chce s desktopovou aplikací pracovat, musí mít umožněn přístup do operačního systému počítače, kde aplikace běží. Jiný druh zabezpečení (např. přímé přihlášení v rámci aplikace) již není teoreticky potřeba.
- Přístup do **webové aplikace** na druhou stranu musí zajišťovat vstupní logovací formulář propojený se správou uživatelů. Ideálně pak každý případný uživatel bude disponovat vlastním loginem a heslem, přičemž dle přidělených práv se každému jednotlivci přidělí různá funkčnost webové aplikace. Práce s veřejnou webovou aplikací samozřejmě přináší ve srovnání s desktopovou mnohem vyšší bezpečnostní rizika.

4.4 Požadavky na dobu odezvy systému

Během diskuze zabývající se stanovením potřebné doby minimální doby odezvy systému byly definovány a vzaty v potaz následující skutečnosti:

- Internet, jako přenosové médium, které bude ve většině případů použito, nezajišťuje 100% jistotu realtime přenosu dat.
- Odezva provedení příkazu měřícího zařízení se dle typu (viz kapitola 2) může pohybovat v časových intervalech 4 – 20ms.
- Zavolání služeb, zápis do databáze, načtení a zpracování dat přidává další dopravní zpoždění systému.
- Prvotní účel měřícího systému, tedy vzdálená kontrola vysílačů rozhlasové stanice, nevyžaduje během sběru dat okamžitou reakci na změnu měřených hodnot.

Díky těmto skutečnostem bylo tedy rozhodnuto, že navrhovaný měřící systém **nebude realtimový**, přičemž bude disponovat těmito parametry:

- Minimální doba dopravního zpoždění 2s daná maximální rychlostí timeru ovládajícího čtení a zápis dat na stranách klienta a vizualizačního rozhraní.
- Možnost nastavení různé měřící frekvence čtení a zápisu u všech měřících portů dle zamýšlené potřeby, což eliminuje nadbytečný přenos dat. Minimální interval měření byl stanoven na jednu sekundu.

4.5 Měřitelné veličiny dle typu

V prvních verzích měřícího systému se předpokládá možnost zpracování dvou typů dat definovaných dle druhu měřeného či řízeného signálu.

4.5.1 Analogový signál

Analogový signál měřící zařízení po jeho diskretizaci prezentuje jako celočíselnou hodnotu v rozsahu bitového rozlišení A/D či D/A převodníku. Naměřený 8bitový signál tedy může být prezentován stavem v intervalu <0;255>. Tato hodnota je ovšem pro potřeby měření nic neříkající, proto se musí ke každému měřícímu bodu uvést minimální a maximální měřitelná hodnota dané veličiny. Díky těmto informacím může následně dojít k přepočtu dle hranic intervalu a následnou správnou interpretaci. Vypočtená hodnota se bude zaokrouhlovat maximálně na dvě desetinná místa. Za další důležitou informaci, jež náleží k analogovému signálu, je jistě název jednotky veličiny, které se měří. Tuto informaci smí uživatel editovat dle svého uvážení u každého z měřících portů.

4.5.2 Digitální signál

Oproti analogovému signálu není u tohoto datové typu zapotřebí uvozovat žádný interval značící hranice měřené hodnoty. Pro záznam dat tedy stačí vyhradit datový typ boolean, ovšem z důvodu zvýšení přehlednosti není od věci přidat informaci uvádějící popis obou možných stavů, který se následně bude uživateli vizualizovat. Místo *true* a *false* tak jistě dle dané situace více vynikne např. *zapnuto, vypnuto* nebo *svítí, nesvítí*. Tyto popisky si samozřejmě může uživatel definovat sám.

4.6 Možnosti zpracování dat

Uživatelské administrační rozhraní musí zcela jistě obsahovat funkci vizualizace naměřených dat včetně možnosti tato data dále zpracovávat. Základní požadavky lze rozdělit do tří kategorií:

- **Výpis hodnot** v rámci administračního rozhraní zajišťuje přehled aktuálního stavu všech portů na zařízení, přičemž hodnoty a stavy řízených portů smí uživatel editovat a ovlivňovat tak chování objektů připojených k měřícímu zařízení.
- **Logování** by mělo být zajištěno jako jedna ze služeb měřícího systému, neboť aktuální zobrazená hodnota veličiny na portech má nulovou vypovídací hodnotu v rámci dlouhodobého sledování. Průběžné ukládání stavu musí ideálně probíhat

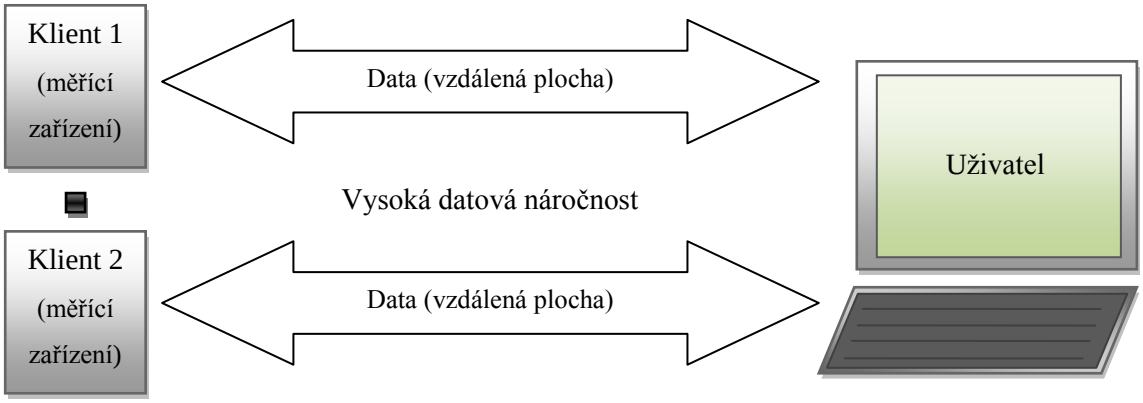
v různé, uživatelem nastavitelné frekvenci, a to v rámci databáze měřícího systému, protože bude jistě obsahovat tisíce záznamů. Zpracování dat (filtrování, řazení, editace, mazání, …) tak navíc bude rychlejší. Služby pro logování musí zajistit též export měřených dat pro každý port zvlášť tak, aby výstup byl dále zpracovatelný i jinými aplikacemi pro práci s daty (Microsoft Excel atd.).

- **Vykreslení do grafu** zajistí uživateli v administračním rozhraní přehled nad situací měření každého portu. Graf by měl obsahovat funkce pro přiblížení detailů, pohyb po křivce nebo export obrázku.

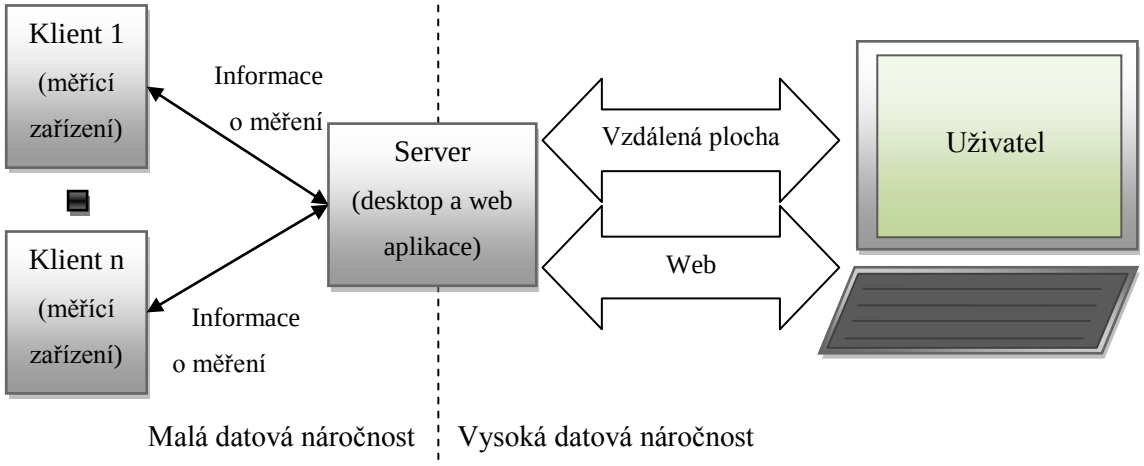
4.7 Způsob komunikace s měřícím zařízením

Jak již bylo naznačeno v předchozím textu, stávající řešení umožňuje práci uživatele s administračním prostředím pouze na desktopu počítače, ke kterému je připojeno měřící zařízení. Musí být tedy vytvořen komunikační kanál mezi tímto PC a uživatelem, většinou realizovaný pomocí software typu vzdálená plocha.

Nový systém si dává za úkol přenést veškerou datově náročnou komunikaci z jednotlivých měřících bodů do centrálního uzlu s dostatečnou výpočetní a především komunikační kapacitou. Mezi tímto centrálním uzlem – serverem a jednotlivými klienty pak bude probíhat jen komunikace přes určitý, datově nenáročný protokol – přenášeny budou jen stěžejní informace o stavu měření a řízení portů. Přenos ostatních dat, především vizualizace administračního prostředí bude k uživateli putovat již po kanále, který může zajistit vyšší datovou kapacitu. Diagramy na obrázcích (Obr. 24, Obr. 25) ilustrují přenos dat u stávajícího a navrhovaného řešení.



Obr. 24. Komunikace dle stávajícího řešení



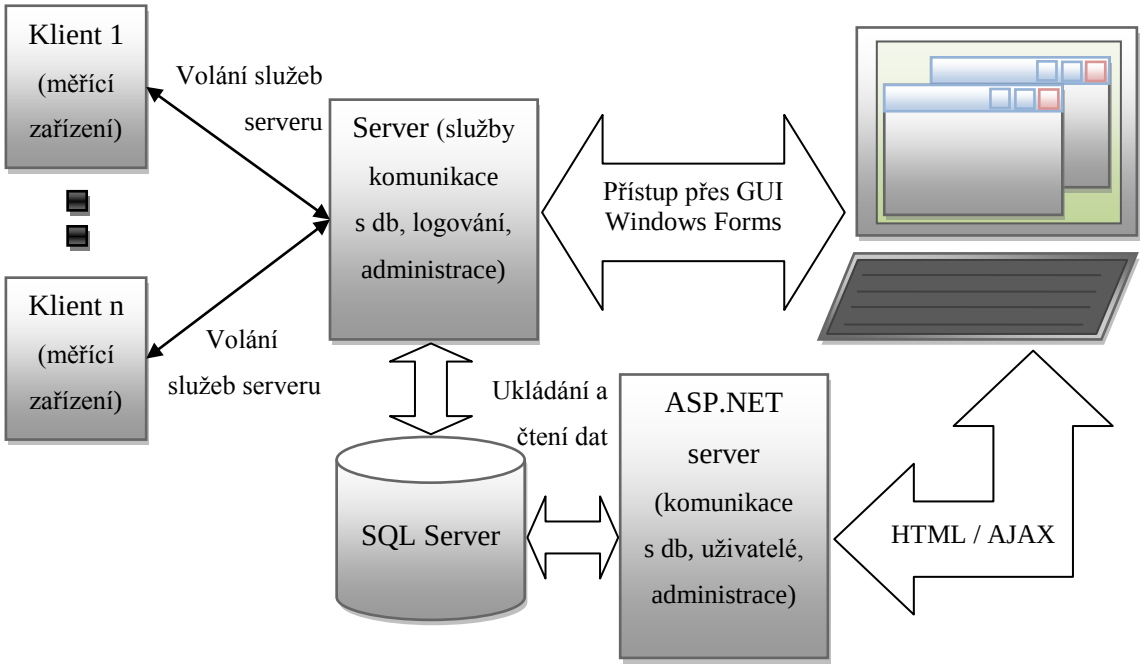
Obr. 25. Komunikace dle navrhovaného řešení

5 NÁVRH ARCHITEKTURY SYSTÉMU

Po základní analýze problému, zhodnocení současného řešení a diskusí stanovených cílech přichází na řadu návrh řešení nového systému.

5.1 Struktura softwarového vybavení systému

Dle závěru všech stanovených požadavků a návrhů byla zvolena struktura měřícího systému na bázi klient – server s centrálním úložištěm dat a několika možnostmi jejich vizualizace. Celkovou strukturu navrhovaného systému ilustruje obrázek (Obr. 26). Navrhovaný systém lze tedy chápat jako kolekci tří aplikací se specifickými vlastnostmi definovanými v následujících kapitolách.



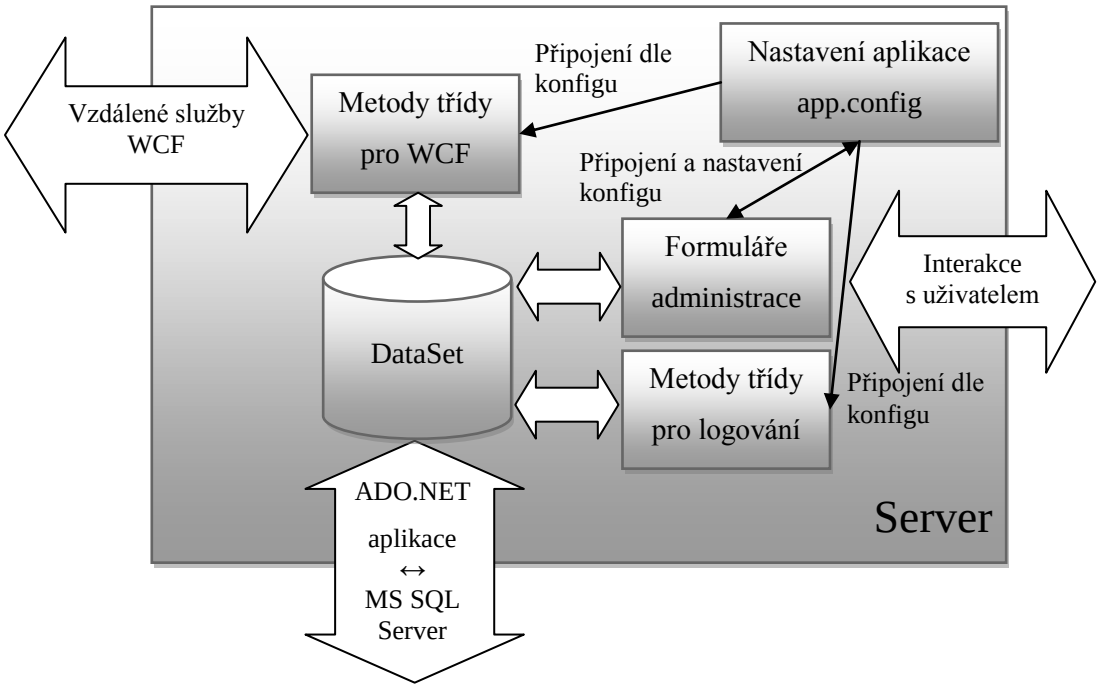
Obr. 26. Struktura aplikací navrhovaného systému

5.1.1 Serverová aplikace

„Centrální mozek“ celého systému musí zajišťovat především:

- Komunikaci s databází
- Logování
- Poskytování vzdálených služeb klientům
- Administrační rozhraní pro uživatele

Činnost všech navrhovaných součástí serverové aplikace demonstruje obrázek (Obr. 26).

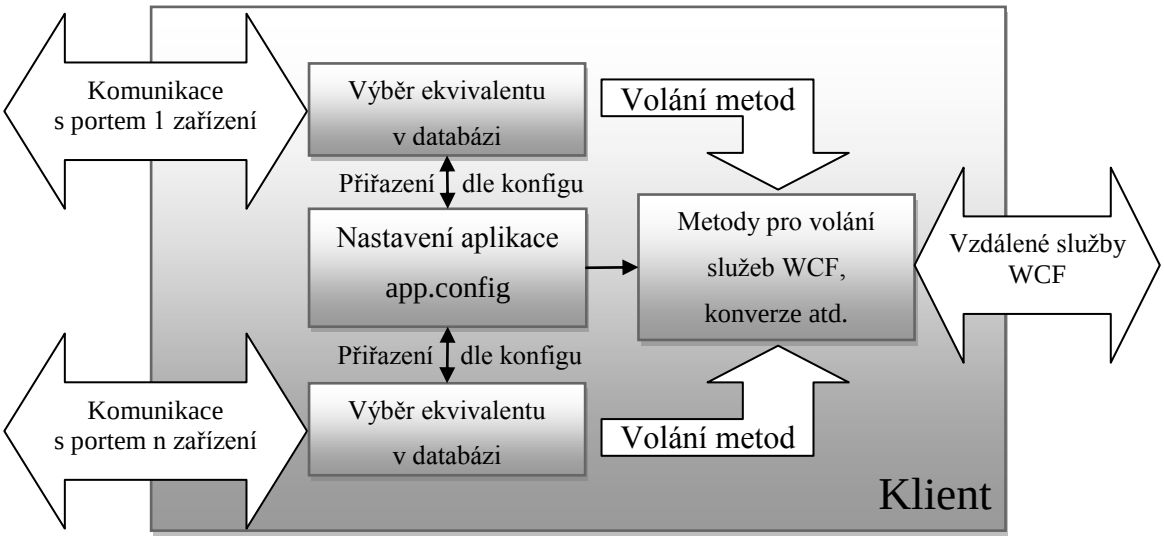


Obr. 27. Struktura serverové aplikace

5.1.2 Klientská aplikace

Klientská aplikace tvoří most mezi konkrétním hardware a obecnými službami serverové aplikace, k nimž se připojí. Je tedy nutné navrhnout dostatečně pružnou a především robustní formu, kterou lze následně lehce uzpůsobovat pro různé druhy měřících zařízení.

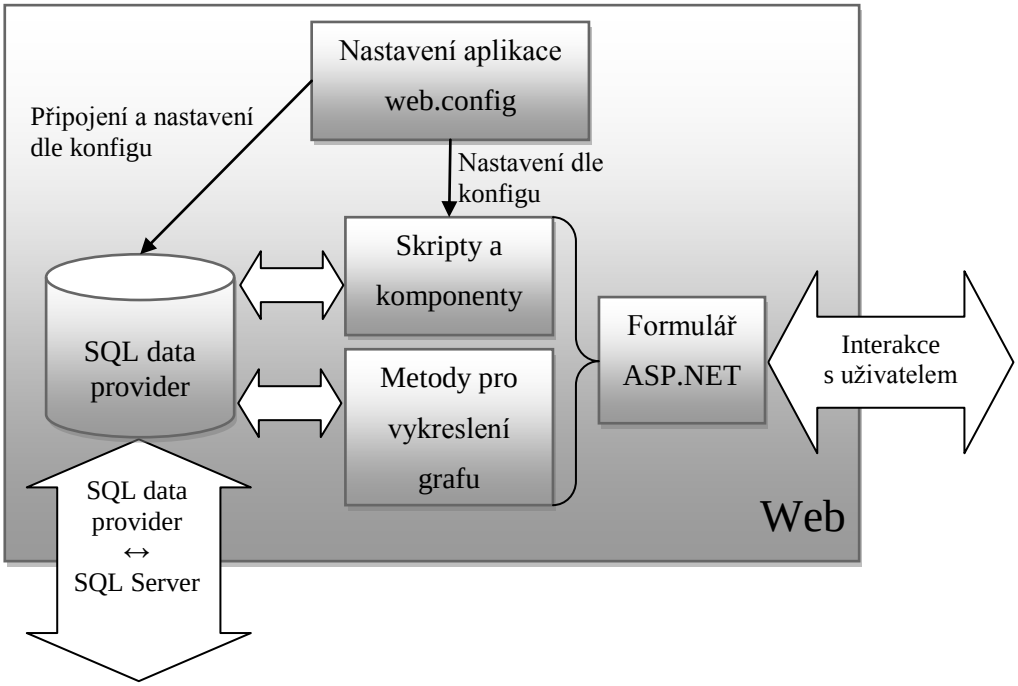
V rámci GUI aplikace lze skrze zmíněné služby přiřadit každému reálnému portu zařízení patřičný ekvivalentní záznam v databázi. Obrázek (Obr. 28) ilustruje činnost klientské aplikace.



Obr. 28. Struktura klientské aplikace

5.1.3 Webová aplikace

Webové prostředí umožňuje uživatelům připojení do systému z jakéhokoliv webového prohlížeče na zařízení připojeném k Internetu či společné lokální síti. Prioritně slouží především jako administrační rozhraní a vizualizace činnosti jednotlivých měřících portů. Některé z funkcí serverové aplikace nejsou pro webovou část logicky dostupné. Činnost webové aplikace popisuje obrázek (Obr. 29).

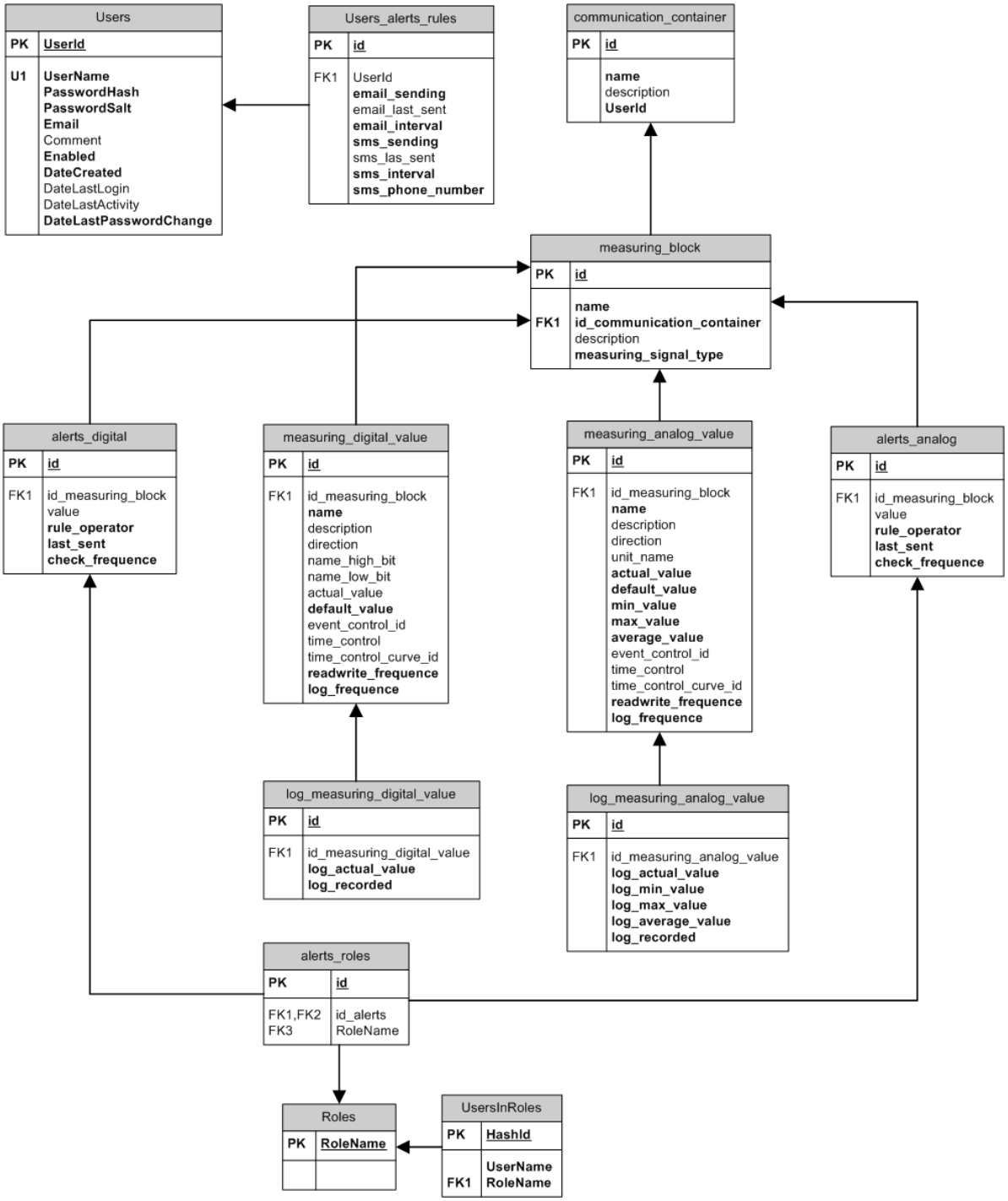


Obr. 29. Struktura webové aplikace

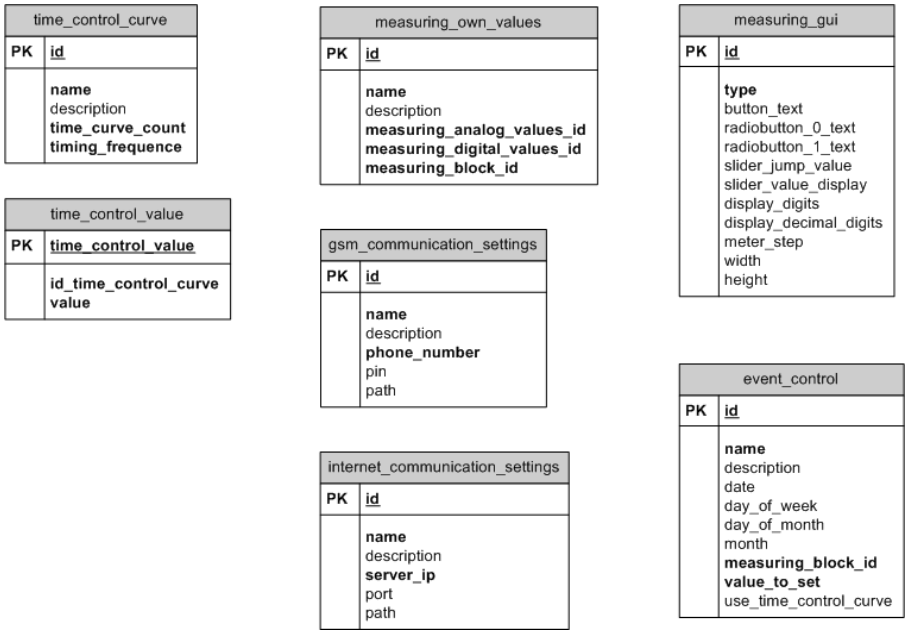
5.2 Struktura databáze systému

Správný návrh databázové struktury projektu znamená zajištění poloviny úspěchu, neboť každé pozdější zásahy a úpravy do tabulek navržené databáze mohou způsobit značné problémy s nenadálou nekompatibilitou již vytvořeného obslužného kódu aplikace. Proto byla návrhu databáze věnována maximální pečlivost.

Celou strukturu databáze ilustrují ERA diagramy (Obr. 30, Obr. 31) obsahující krom entit i jejich atributy. Propojení mezi jednotlivými tabulkami řeší SQL příkazy pro spojení *INNER JOIN* nebo *LEFT JOIN*. Struktura tabulek využívá relační vztahy mezi jednotlivými entitami, což zajišťuje vyšší integritu dat – žádné mrtvé záznamy po smazání položky provázané entity.



Obr. 30. Struktura databáze ControlIT – tabulky s relacemi



Obr. 31. Struktura databáze ControlIT – tabulky bez relací

Každá z entit v databázi zastává některou z důležitých funkcí systémů. Jejich popis včetně relačního propojení mezi entitami uvádí tabulka níže (Tab. 9).

Tab. 9. Popis jednotlivých entit databáze ControlIT

Název entity	Relační propojení	Popis entity
alerts_analog	measuring_block alerts_roles	Obsahuje definici varování a jeho chování pro analogové porty
alerts_digital	measuring_block alerts_roles	Obsahuje definici varování a jeho chování pro digitální porty
alerts_roles	alerts_analog alerts_digital Roles	Propojovací tabulka určující jednotlivá varování skupinám uživatelů
communication_container	measuring_block	Definuje vlastnosti měřícího zařízení, pod jeho ID se připojují jednotlivé měřící body
event_control	-	Zatím se nevyužívá
gsm_communication_settings	-	Zatím se nevyužívá
internet_communication_settings	-	Zatím se nevyužívá
log_measuring_analog_value	measuring_analog_value	Ukládá naměřené hodnoty analogových portů
log_measuring_digital_value	measuring_digital_value	Ukládá naměřené hodnoty digitálních portů
measuring_analog_value	log_measuring_analog_value measuring_block	Ukládá vlastnosti analogového měření pro každý měřící bod. Zahrnuje hodnoty měření a směr (měření / řízení)
measuring_block	alerts_analog	Měřící bod / port. Uvozuje

	alerts_digital measuring_analog_value measuring_digital_value communication_container	jeho vlastnosti, především datový typ veličiny
measuring_digital_value	log_measuring_digital_value measuring_block	Ukládá vlastnosti digitálního měření pro každý měřící bod. Zahrnuje hodnoty měření a směr (měření / řízení)
measuring_gui	-	Zatím se nevyužívá
measuring_own_values	-	Zatím se nevyužívá
Roles	alerts_roles UsersInRoles	Definuje uživatelské role
time_control_curve	-	Zatím se nevyužívá
time_control_value	-	Zatím se nevyužívá
Users	Users_alerts_rules	Ukládá soupisku uživatelů
Users_alerts_rules	Users	Propojovací tabulky mezi uživatelskými rolemi – pravidly a varováními
UsersInRoles	Roles	Propojovací tabulka uživatelů a rolí

5.3 Popis datatypů měřitelných a řiditelných objektů

První verze měřícího systému obsahuje zatím pouze dva datatypy použitelných objektů:

- Datatyp pro zpracování **analogového signálu**
- Datatyp pro zpracování **digitálního signálu**

V dalších verzích systému se počítá s přidáním následujících datatypů:

- Datatyp pro řízení pomocí **PWM modulace**
- Datatyp pro **ovládání interního časovače** desky K8055

5.3.1 Definice datatypu analogového signálu

Tento datový typ lze použít pro přenos a uložení informací o stavu analogových vstupů a výstupů v měřících zařízeních. Právě jeden záznam analogového datatypu se vždy váže na jeden měřící port. Disponuje několika více či méně zásadními atributy, které popisuje tabulka (Tab. 10). Některé položky datatypu se zatím v současné verzi nevyužívají, ovšem s jejich nasazením se počítá do budoucna v rámci přidávání dalších funkcí.

Tab. 10. Datatyp analogvého signálu

Název atributu	Datatyp	Popis
id	int	Primární klíč, autoinkrementační
id_measuring_block	int	Cizí klíč na měřící blok, ke

		kterému je datatyp přiřazen
name	nvarchar(255)	Definuje jméno analogového portu
description	ntext	Vložení popisky portu (místa, kde se nachází, co ovládá, …)
direction	bit	Určuje směr toku dat: log 0 pro výstup z měřícího zařízení; log 1 pro vstup do měřícího zařízení
unit_name	nvarchar(50)	Název jednotky měřené veličiny
actual_value	float	Aktuální uložená hodnota
default_value	float	Základní hodnota
min_value	float	Minimální měřitelná hodnota veličiny (nutná pro přepočet)
max_value	float	Maximální hodnota veličiny (nutná pro přepočet)
average_value	float	Průměrná hodnota veličiny
event_control_id	int	Nepoužívá se
time_control	bit	Nepoužívá se
time_control_curve_id	int	Nepoužívá se
readwrite_frequence	int	Udává frekvenci čtení a zápisu (v sekundách)
log_frequence	int	Udává frekvenci logování (v sekundách)

5.3.2 Definice datatypu digitálního signálu

Datový typ pro přenos a uložení stavu digitálních portů působí na první pohled oproti analogovému datatypu mnohem jednodušeji, neboť neobsahuje atributy pro minimální, maximální a průměrnou hodnotu. Přidává ovšem možnost zadání popisků pro *high* a *low* stavy. Stejně jako u entity pro analogový signál i tento datatyp obsahuje položky, jejichž využití je otázkou budoucích verzí systému. Tabulka (Tab. 11) podrobně ilustruje strukturu tohoto datatypu.

Tab. 11. Datatyp digitálního signálu

Název atributu	Datatyp	Popis
id	int	Primární klíč, autoinkrementační
id_measuring_block	int	Cizí klíč na měřící blok, ke kterému je datatyp přiřazen
name	nvarchar(255)	Definuje jméno analogového portu
description	ntext	Vložení popisky portu (místa, kde se nachází, co ovládá, …)
direction	bit	Určuje směr toku dat: log 0 pro výstup z měřícího zařízení; log 1 pro vstup do měřícího zařízení
name_high_bit	nvarchar(50)	Název k měřenému datatypu
name_low_bit	nvarchar(50)	Popis k měřenému datatypu

actual_value	bit	Aktuální uložená hodnota
default_value	bit	Základní hodnota
event_control_id	int	Nepoužívá se
time_control	bit	Nepoužívá se
time_control_curve_id	int	Nepoužívá se
readwrite_frequence	int	Udává frekvenci čtení a zápisu (v sekundách)
log_frequence	int	Udává frekvenci logování (v sekundách)

6 REALIZACE NAVRŽENÉHO SYSTÉMU

Podle všech navržených stanovisek a kritérií započal postupný vývoj kolekce aplikací ControlIT. Kapitola popisuje jednotlivé programy kolekce, jejich vlastnosti, možnosti použití a přibližuje funkce jednotlivých stěžejních metod a objektů.

6.1 Průběh vývoje projektu

Během vývoje celého projektu bylo kromě samotného převedení návrhu do reality potřeba nastudovat značné množství informací o technologiích, jež se musely k realizaci použít. Tabulka (Tab. 12) nepopisuje tedy jen samotný průběh vývoje, ale zahrnuje i dobu potřebnou k osvojení všech prostředků. Jednotlivá stádia se samozřejmě postupně prolínala, což ovšem tabulka neilustruje.

Tab. 12. Průběh vývoje projektu

<table>
<tr><th>Měsíc a rok</th><th colspan="2">ControlIT Server</th><th>ControlIT Klient</th><th>ControlIT Web</th></tr>
<tr><td>Říjen 2008</td><td colspan="4">Analýza</td></tr>
<tr><td>Listopad 2008</td><td>Komunikace s databází

Návrh GUI hlavních formulářů

Knihovny pro práci s konektivitou k databázi</td><td colspan="3">Návrh základní struktury databáze</td></tr>
<tr><td>Prosinec 2008</td><td rowspan="3">Formulář a aplikační logika správy měřících bodů</td><td colspan="2" rowspan="3">WCF – vytvoření služby a jejich metod</td><td>-</td></tr>
<tr><td>Leden 2009</td><td>-</td></tr>
<tr><td>Únor 2009</td><td rowspan="2">Vývoj webu v ASP.NET</td></tr>
<tr><td>Březen 2009</td><td colspan="2">-</td><td>-</td></tr>
<tr><td>Duben 2009</td><td colspan="2">Logování stavů
Grafy generované z logovaných hodnot</td><td>-</td><td>Grafy generované z logovaných hodnot</td></tr>
<tr><td>Květen 2009</td><td colspan="4">Testování</td></tr>
</table>

6.2 Popis serverové aplikace ControlIT

První z kolekce aplikací zastává stěžejní funkci serveru, tedy jádra celého systému. Přestože prioritně poskytuje především služby pro komunikaci, obsahuje široké spektrum dalších funkcí:

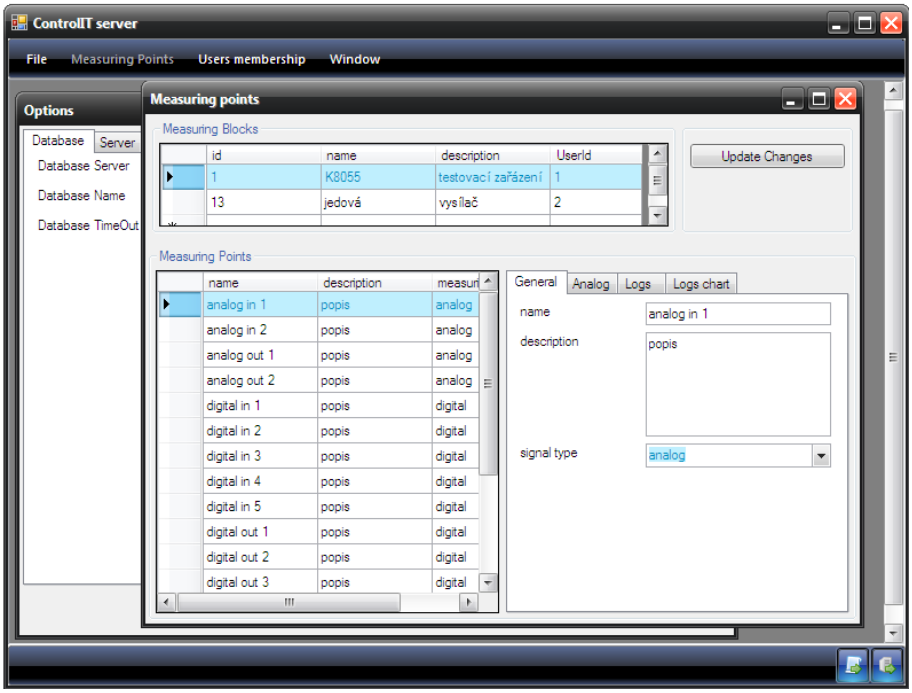
- Administrační prostředí pro správu měřících zařízení a portů, včetně správy logů a generování grafů
- Windows Communication Foundation služba pro komunikaci s měřícími klienty
- Logování změn veličin na měřených a řízených portech.
- Formulář pro nastavení aplikace

Aplikace disponuje GUI rozhraním na bázi knihoven *Windows Forms*, přičemž provoz serveru jako běžné služby Windows zatím není možný.

6.2.1 Hlavní okno aplikace

Serverová aplikace využívá možností vnořených formulářů MDI. Hlavní formulář (Obr. 32) lze popsat jako ekvivalent pracovní plochy, na které se zobrazují všechny ostatní vyvolané formuláře. Okno formuláře lze rozdělit na tři základní části:

1. Menu
2. Pracovní plocha
3. Pruh s indikátory běžících služeb



Obr. 32. Hlavní formulář serverové aplikace ControlIT

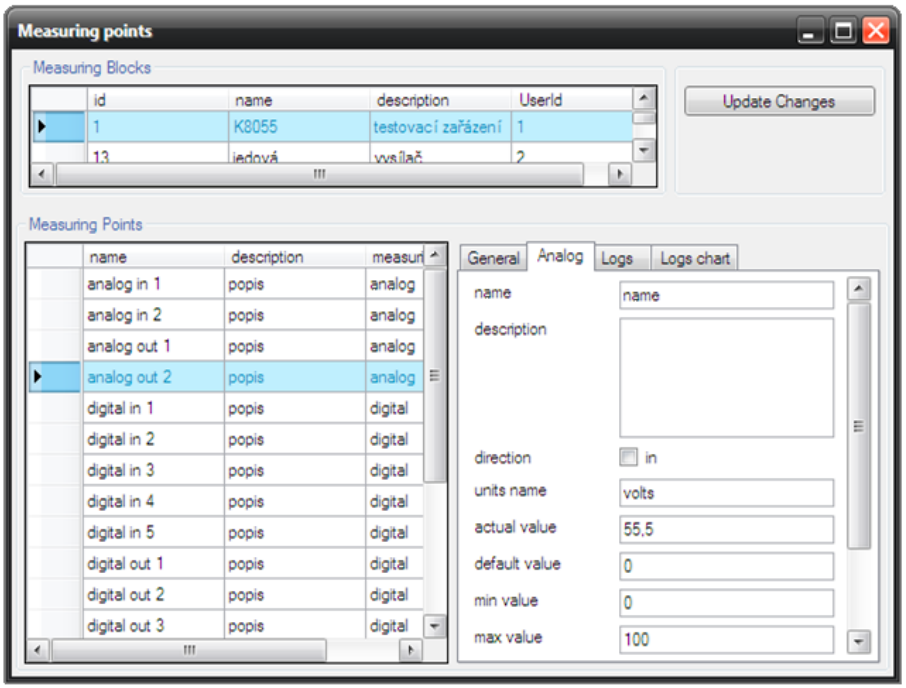
Důležitý prvek hlavního formuláře – pruh indikátorů běžících služeb obsahuje ikony se stavy jednotlivých součástí systému. Popis ikon služeb a možnosti jejich stavů definuje tabulka níže (Tab. 13).

Tab. 13. Popis jednotlivých stavových ikon služeb ControlIT Serveru

Služba	Ikona	Popis
Server WCF služeb		Univerzální ikona indikující zatím neběžící server
		Běžící WCF server
		Vypnutý WCF server (po pádu, případně akcí uživatele)
		Chyba WCF serveru
Logování		Univerzální ikona indikující zatím neběžící službu logování
		Běžící služba logování
		Vypnuté logování (po pádu, případně akcí uživatele)
		Chyba logování

6.2.2 Administrace měřících bodů

Kompletní administrační rozhraní zastupuje formulář *Measuring points* vyvolatelný z menu hlavního okna (Obr. 33).

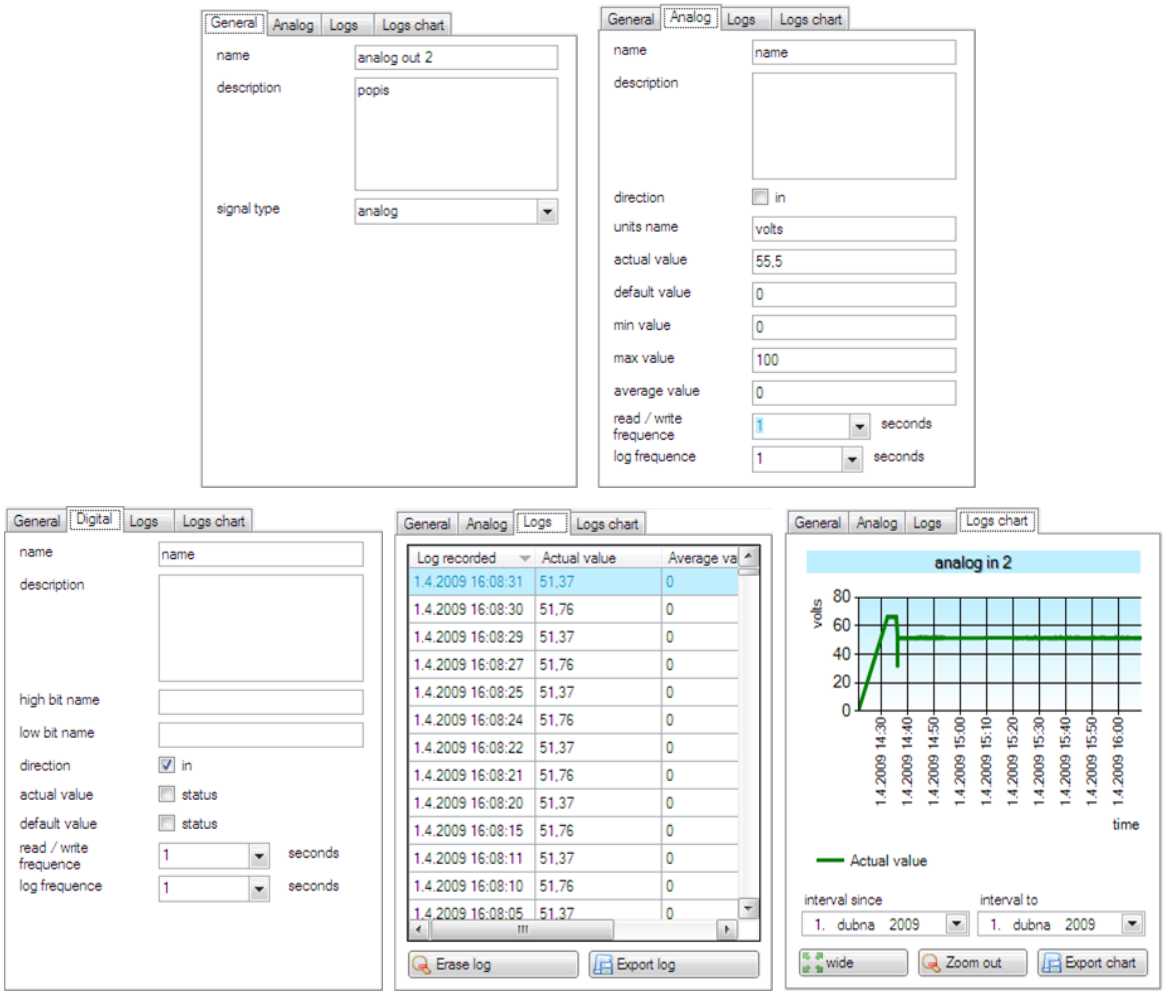


Obr. 33. Formulář Measuring Points - editace měřících bodů

Rozložení prvků v tomto formuláři lze rozdělit do čtyř funkčních celků:

1. Tabulka typu *DataGridView* v bloku *Measuring Blocks* s výběrem měřících zařízení.
2. Aktualizační tlačítko *Update Changes* zajišťující synchronizaci upravených dat aplikace s těmi ve vzdálené databázi.

3. Tabulka typu *DataGridView* v bloku *Measuring Points* s výpisem jednotlivých portů na označeném zařízení.
4. Systém záložek pro vizualizaci detailů a informací o stavu jednotlivých portů. Obsah všech záložek ilustruje obrázek (Obr. 34).



Obr. 34. Záložky formuláře s detaily měřících bodů

Formulář *Measuring points* umožňuje následující paletu činností:

- Přidání a editace měřícího zařízení
- Přidání a editace měřícího portu k danému zařízení
- Výběr typu měřené veličiny (analog / digital) a následná editace potřebných parametrů.
- Zobrazení logu pro vybraný port a jeho typ veličiny, log lze smazat nebo exportovat.
- Zobrazení grafu pro daný port a typ veličiny, v grafu lze táhnutím myší vybrat oblast pro zvětšení a tím znázornění detailů.

Vložení nového měřícího bodu a přiřazení portu může probíhat následujícím způsobem:

1. Zadání názvu, popisky a id vlastníka měřícího bodu do tabulky *Measuring Blocks*. Id vlastníka by mělo být v současné verzi systému automaticky nastaveno na 0.
2. Stisknutím tlačítka *Update* zajištění synchronizace dat s databází.
3. Zadání názvu, popisky a především datového typu (slova *analog* nebo *digital*) do tabulky *Measuring Points*.
4. Stisknutím tlačítka *Update* zajištění synchronizace dat s databází.
5. K novému měřícímu portu se přiřadí datatypy *analog* a *digital*, přičemž se vybere k editaci typ vybraný v definici portu. Je potřeba vyplnit především minimální a maximální hodnoty (u analogového datatypu), dále pak frekvence zápisu a logování.
6. Stisknutím tlačítka *Update* zajištění synchronizace dat s databází.

Časté použití tlačítka *Update* je vyžadováno z důvodu aktuální synchronizace dat. V případě nekorektních změn ve formuláři administrace se tyto změny navíc bez výslovného požadavku uživatele neuloží a znovunačtením formuláře se tedy dá pracovat od místa poslední aktualizace změn. Popis stěžejních funkcí použitých v popisovaném formuláři shrnuje tabulka (Tab. 14).

Tab. 14. Stěžejní funkce formuláře administrace měřících bodů

Název funkce	Popis funkce
datafill_form()	Funkce inicializuje objekty ADO.NET, vygeneruje dotazy, připojí se k databázi, naplní DataSet daty
datafill_logChart(DataView dataView, string signalType)	Podobná funkcionalita jako u datafill_form() ovšem filtrovaná pouze na jednu hodnotu, naplní graf daty
formBindings()	Provede propojení (binding) mezi tabulkami DataSetu a prvky formuláře
getParentInstance(parentForm instance)	Dědí do formuláře prvky rodičovského okna – umožňuje vypnutí / zapnutí položky menu pro otevření formuláře
logsViewRefresh()	Provede refresh dat v *dataGridView* tabulce pro logování hodnot
measuring_point_filter(string currentRow_ID)	Filtr měřících bodů podle vybraného měřícího zařízení
measuring_values_filter(string currentRow_ID)	Filtr datatypů hodnot přiřazených k vybranému portu
tabPagesRefesh()	Překreslení záložek podle vybraného datatypu – zobrazí jen *analog* nebo jen *digital*
updateData()	Projde všechny tabulky DataSetu a uzavře jejich stavy editace, následně aktualizuje přes DataAdaptery všechna data

6.2.3 WCF služby

Veškerá vzdálená volání WCF služeb a jejich následné vykonávání zajišťují třída *ControlIT_Service : IControlIT_Service.cs*, jež je nedílnou součásti projektu. Aplikace ControlIT Server pak funguje jako hostitel těchto služeb, umožňuje tedy jejich spuštění a zastavení. Tabulka (Tab. 15) popisuje všechny vykonatelné kontrakty služby.

Tab. 15. WCF služby aplikace server

Název kontraktu	Návratový datatyp	Popis
DoWork()	string	Testovací metoda, která zjistí, zda komunikace klienta a služby opravdu funguje
getCommunicationContainers()	ControlIT_Service_ CommunicationContainer[]	Metoda vrací objekty typu list s položkami typu ControlIT_Service_ CommunicationContainer. Zajišťuje klientovi příjem výpisu všech měřících zařízení
getMeasuringBlocks(int idCommunicationContainer)	ControlIT_Service_ MeasuringBlock[]	Metoda vrací objekty typu list s položkami typu ControlIT_Service_ MeasuringBlock. Zajišťuje klientovi příjem výpisu všech měřících portů dle vstupního id zařízení
setDigitalToServer(ControlIT_ Service_Digital measuringDigitalPort)	Bool	Umožňuje klientovi vložit do databáze systému objekt typu ControlIT_Service_Digital – aktuální hodnoty vybraného digitálního portu
getDigitalFromServer(int idMeasuringDigitalPort)	ControlIT_Service_Digital	Umožňuje klientovi získat aktuální informace o vybraném digitálním portu skrze objekt ControlIT_Service_Digital
setAnalogToServer (ControlIT_Service_Analog measuringAnalogPort)	Bool	Umožňuje klientovi vložit do databáze systému objekt typu ControlIT_Service_Analog – aktuální hodnoty vybraného analogového portu
getAnalogFromServer (int idMeasuringAnalogPort)	ControlIT_Service_Analog	Umožňuje klientovi získat aktuální informace o vybraném analogovém portu skrze objekt ControlIT_Service_ Analog

Jednotlivé metody služby využívají funkce (Tab. 14) totožné s administračním formulářem. Neváží se ovšem na žádné formulářové prvky, ani interakci od uživatele.

6.2.4 Logování

Všechny funkce pro logování hodnot zastává třída *Logging.cs*. Obecně lze popsat funkci logování následujícím postupem:

1. Připojí se k databázi a stáhne si do DataSetu obsah logovací tabulky
2. Vyfiltruje datasety logování podle id hodnoty, která je na řadě
3. Porovná poslední měřenou hodnotu s poslední uloženou hodnotou logu
4. Jsou-li rozdílné, společně s aktuálním časem uloží nový záznam logu, jinak poslední měřenou hodnotu zahodí – tato metoda sníží nadbytečnost dat

Zalogované hodnoty se následně využívají ke konstrukci grafů, případně exportu pro další možné zpracování v jiném software.

Obsah tabulky (Tab. 16) představuje stěžejní funkce třídy logování.

Tab. 16. Funkce třídy logování

Název funkce	Popis funkce
countersSet()	Univerzální čítač časovače pro intervaly tiku 10, 30, 60, 1800 a 3600 sekund
Datafill()	Funkce inicializuje objekty ADO.NET, vygeneruje dotazy, připojí se k databázi, naplní DataSet daty
enabled()	Veřejná funkce pro zapnutí a či vypnutí logování. Po spuštění provede test konektivity a konzistence databáze, následně zapne časovač
logAnalogValues()	Funkce pro zalogování analogových hodnot, projde všechny položky datasetu, ověří, v jakém cyklu se mají logovat, udělá kontrolu posledního záznamu a případně zaloguje
logDigitalValues()	Funkce pro zalogování digitálních hodnot, projde všechny položky datasetu, ověří, v jakém cyklu se mají logovat, udělá kontrolu posledního záznamu a případně zaloguje
OnTimedEvent(object source, ElapsedEventArgs e)	Event přiřazení přerušení časovače, vyvolá navýšení čítače časovače pro různé intervaly, následně funkce pro zápis logů analogového a digitálního signálu

Výsledky logování lze prezentovat dvojím způsobem:

1. Výpis logu dle vybraného měřícího portu může uživatel procházet pomocí objektu *dataGridView* formuláře *Measuring Points*, viz obrázek (Obr. 34) v předchozí kapitole. Tabulka je doplněna tlačítky s možností smazání celého logu pro zadaný

port, případně existuje možnost exportu logu ve formátu XML (včetně schématu) pro další použití.

2. Vykreslení grafu z logovaných dat – podrobněji popisuje následující kapitola.

6.2.5 Vykreslování grafů

Důležitou funkcí pro zpracování měřených dat je vizualizace jejich předchozího stavu, tedy vykreslení do grafu. Zobrazení dat v objektu grafu včetně veškeré aplikační logiky zajišťuje výborně zpracovaná knihovna Microsoft Chart Control[4], kterou musí administrátor před použitím aplikace ControlIT Server doinstalovat jako doplněk .NET Frameworku 3.5. Pro vývojáře byl pak uvolněn plug-in pro Visual Studio 2008, jenž rozšíří nabídku toolboxu u Windows Forms a ASP.NET aplikací o objekt *Chart*.

Každý z grafů obsahuje pět kolekcí objektů:

- **Anotace**
- **ChartAreas** – nezávislé plochy s vlastními parametry os
- **Legends** – popisky grafů, os
- **Series** – kolekce křivek, koláčů a jiných typů grafu, které se vykreslí do dané oblasti
- **Titles** – titulky grafu

V projektu bylo použito pro objekt *Chart* dvou položek *ChartAreas*, pro každý datatyp veličiny zvlášť, přičemž tak mohou vykreslené grafy disponovat různým nastavením rozsahů os, jejich popisek a možnostmi zoomu. Aplikace pak ošetřuje zobrazení správného grafu dle požadovaného datatypu.

Zatímco základní vlastnosti grafu umožňuje jednoduše nastavit nástroj designéru, připojení dat se řeší imperativně v kódu (Obr. 35) formuláře *Measuring Points*. Na objekt *Chart* lze namapovat data uložená v v datových nosičích DataView, DataTable, DataSet atd.

```
1 // Nastavit datasource pro chart, umí DataView, DataTable atd.
2 logChart.DataSource = dataView;
3 // Nastavit zdroje dat pro jednotlivé osy
4 logChart.Series["SeriesAnalogActual"].XValueMember =
```

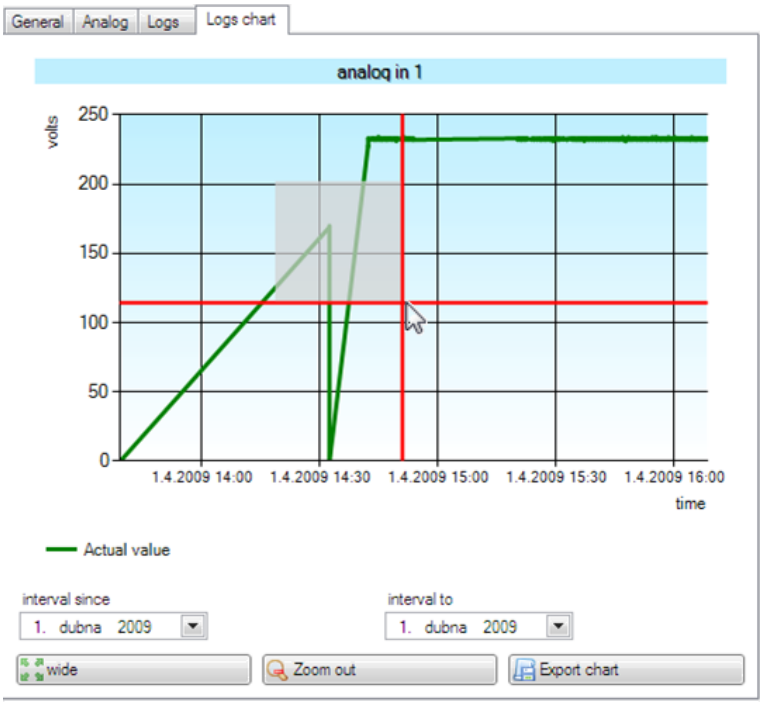
[4] Webová prezentace produktu Microsoft Chart Control: http://code.msdn.microsoft.com/mschart

```
    "log_recorded";
5   logChart.Series["SeriesAnalogActual"].YValueMembers =
    "log_actual_value";
6   // Určit, jaký datový typ bude přiřazen pro X osu (čas)
7   logChart.Series["SeriesDigitalActual"].XValueType =
    ChartValueType.DateTime;
8   logChart.DataBind();
```

Obr. 35. Napojení dat na objekt Chart

Pro pohodlnější procházení dat ve vykresleném grafu umožňuje objekt *Chart* metody změny maximálního či minimálního intervalu a především možnost přiblížení nad vybranou oblastí – zoom. V případě zobrazení analogového datového typu je možné tahem myši (Obr. 36) vybrat konkrétní oblast v osách X i Y, u digitálního datového typu lze zobrazit přiblížení osy X. Oddálení výběru může uživatel provést klepnutím na ikonku *mínus* „⊖“ pro každou osu zvlášť, případně celkově tlačítkem *Zoom out*.

Pro případ potřeby omezení intervalu zobrazené křivky grafu formulář disponuje kalendářovými prvky pro výběr dolní a horní hranice. Tlačítko *Wide* rozšíří plochu grafu na téměř celou plochu formuláře. Funkce *Export chart* umožňuje uložit zobrazená data do běžných grafických rastrových formátů.

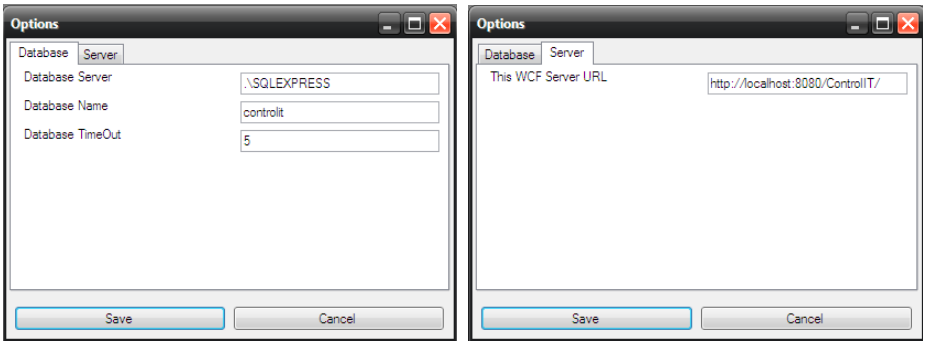


Obr. 36. Zoom v grafu analogového signálu

6.2.6 Možnosti konfigurace

Aplikace ukládá veškerou svoji konfiguraci do datové struktury vytvořené pomocí integrovaného nástroje Visual Studia určeného pro správu projektů (menu *Project – Properties* – záložka *Settings).*

Pro vizualizaci a editaci hodnot konfigurace slouží formulář *Options*, který nabízí tematicky dělené záložky s nastavením (Obr. 37). Metodu načtení a uložení hodnot demonstruje kód v dalším obrázku (Obr. 38).



Obr. 37. Formulář pro nastavení aplikace ControlIT Server

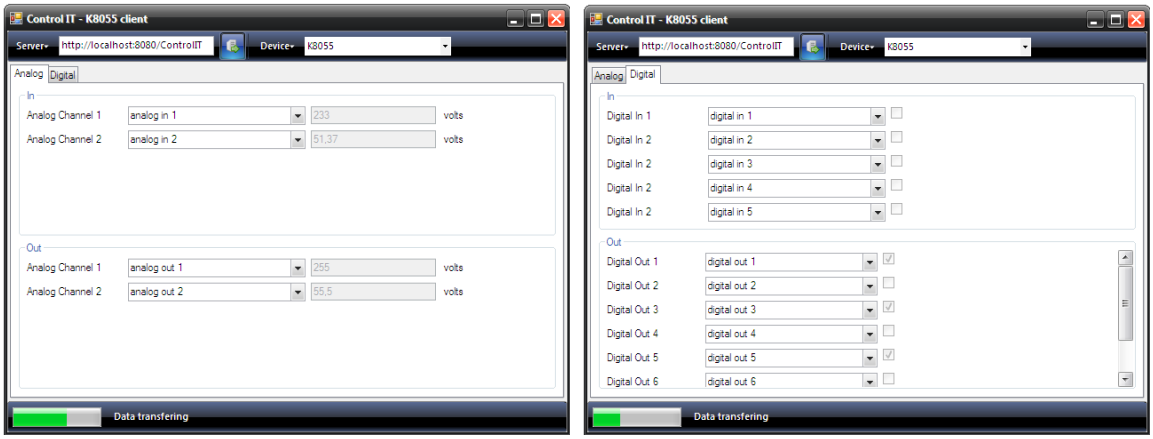
```
1  // načtení hodnoty do objektu formuláře
2  textBox_database_server.Text =
3  Properties.Settings.Default.database_server.ToString();
4  // reset hodnot v nastavení na původní
5  Properties.Settings.Default.Reset();
6  // uložení z objektu formuláře do nastavení
7  Properties.Settings.Default.database_server =
8  textBox_database_server.Text.ToString();
9  // nutno zadat příkaz k zápisu
10 Properties.Settings.Default.Save();
```

Obr. 38. Příklad kódu pro načtení a uložení hodnot konfigurace

6.3 Popis klientské aplikace ControlIT

Klientská aplikace ControlIT Client představuje most mezi měřícím hardware a WCF službami serveru. Pro každý druh zařízení je potřeba vytvořit specifického klienta, neboť se tato zařízení mohou lišit různým počtem portů a především jinými metodami přístupu (API) na daný hardware.

GUI aplikace (Obr. 39) působí na uživatele zcela jednoduše a přehledně. Komunikační porty jsou rozděleny podle jednotlivých typů do záložek a odděleny dle vstupů a výstupů. Při správném běhu programu lze sledovat aktuální stavy na jednotlivých portech, ovšem bez možnosti zásahu, tedy ovlivnění hodnot uživatelem.



Obr. 39. Formulář klientské aplikace pro zařízení K8055

Klient vyžaduje minimum nastavení, ke správnému chodu musí uživatel provést následující konfiguraci:

1. Připojení měřícího zařízení do portu USB počítače, na kterém bude běžet klient.
2. Po spuštění klienta zadat správnou URL adresu WCF serveru. V případě neúspěšného připojení provést kontrolu, zda na počítači se serverem existuje Windows uživatelský účet stejný, pod jakým pracuje klientská aplikace. Tento účet musí mít práva spouštět serverovou aplikaci. Zajistí se tak ověření přístupu legálních uživatelů do měřícího systému.
3. Vybrat ve výběrovém seznamu název zařízení, na které se budou mapovat porty.
4. Namapovat každému měřícímu portu pomocí výběrového seznamu jeho ekvivalent v databázi.
5. Klepnutím na položku *Device* připojit zařízení.
6. Po provedení všech kroků správně by měl klient načíst a zapsat hodnoty měřícího zařízení a komunikovat se serverem. Tuto činnost včetně výpisu chyb vizualizuje ve statusbaru aplikace.
7. Přes položky *Server* a *Device* lze nastavit následující vlastnosti:
    - **Server – Autostart** – pokusí se připojit k serveru již po spuštění aplikace
    - **Server – Manual** – připojí se až na žádost uživatele

- **Device – Autostart** – v označeném módu se pokusí najít a připojit zařízení již během startu aplikace
- **Device – Reset I/O when closed** – resetuje všechny stavy na zařízení těsně před jeho odpojením
- **Device – Open / Close** – připojí či odpojí zařízení

Stěžejní funkce ilustruje tabulka (Tab. 17), bližším popisem obsahu a využití některých z nich se zabývají následující kapitoly.

Tab. 17. Funkce klientské aplikace

Název funkce	Popis funkce
dataUpdate()	Ověří, zda je klient připojen k serveru a postupně aktualizuje data všech portů
getDataAfterConnect()	Použije se po prvním připojení k serveru, ověří platnost posledně nastavených údajů a provede propojení mezi porty a ekvivalenty získané přes WCF
inicialize()	Provede kontrolu připojení k serveru a monitorovacímu zařízení podle nastavení aplikace, co je stanoveno, připojí a následně spustí časovač pro čtení / zápis
saveSelectedValues()	Funkce uloží do konfigu aplikace nastavené propojení hardware a ekvivalentů získaných přes WCF
setCurrentAnalogToActualValue(…)	Pomocná funkce pro převod naměřené hodnoty z analogového vstupu na hodnotu přepočtenou z rozdílu minimální a maximální možné hodnoty veličiny
getCurrentActualValueToActualValue(…)	Pomocná funkce pro převod vstupní hodnoty do analogového výstupu podle rozdílu minimální a maximální možné hodnoty veličiny a nastavované hodnoty
timerDataUpdate_Tick(…)	Zpracování přerušení časovače, navýší hodnoty pro jednotlivé časové intervaly
usbeDeviceConnection(…)	Provede vyhledání připojeného USB zařízení a následně i jeho připojení k aplikaci
WcfServerConnection()	Funkce pro připojení klientské aplikace k WCF službám. Přebírá parametry z nastavení aplikace

6.3.1 Připojení měřícího zařízení

Připojení měřícího zařízení kompletně řeší dodané API funkce, které nejprve provedou hledání na portech USB a v případě úspěchu jej připojí. Každá deska K8055 disponuje dvojicí jumperů umožňujících nastavení ID zařízení. V současné verzi ovšem klient ignoruje možnost připojení více měřících desek k jednomu počítači, a tedy vybere tu, která

je nalezena nejdříve. Obrázek (Obr. 40) demonstruje kód zajišťující připojení a odpojení desky.

```
1 // Nalezení zařízení
2 device_id = usbe.SearchDevices();
3 if (usbe.OpenDevice(device_id) != 0 && device_id != 0)
4 {
5    // zařízení připojeno
6 }
7 // Odpojení zařízení
8 usbe.CloseDevice();
```

Obr. 40. Příklad připojení zařízení K8055

6.3.2 Připojení k serveru

Připojení k službám serveru pomocí Windows Communication Foundation realizuje klient dle nastavení automaticky nebo manuálně. V každém případě je ovšem nutné zadat správnou URL adresu s portem, na které WCF server naslouchá a ujistit se, že uživatel Windows na klientské stanici existuje i na straně serveru. Status připojení indikuje stav ikony tlačítka v horní liště totožný s popisem v tabulce (Tab. 13) kapitoly 6.2.1.

Vytvoření nové instance WCF klienta a načtení dat demonstruje kód na následujícím obrázku (Obr. 41).

```
 1 // Přebere z nastavení aplikace aktuální adresu, kam se připojit
 2 EndpointAddress address = new
 3 EndpointAddress(Properties.Settings.Default.service_wcf_url);
 4 // Nastavení nové instance bindings
 5 WSHttpBinding binding = new
 6 WSHttpBinding("WSHttpBinding_IControlIT_Service");
 7 // Založení nové instance WCF klienta
 8 wcfClient = new
 9 ControlIT_Service.ControlIT_ServiceClient(binding, address);
10 wcfClient.Open();
11 // Využívání vzdálených služeb
12 wcfClient.Close();
```

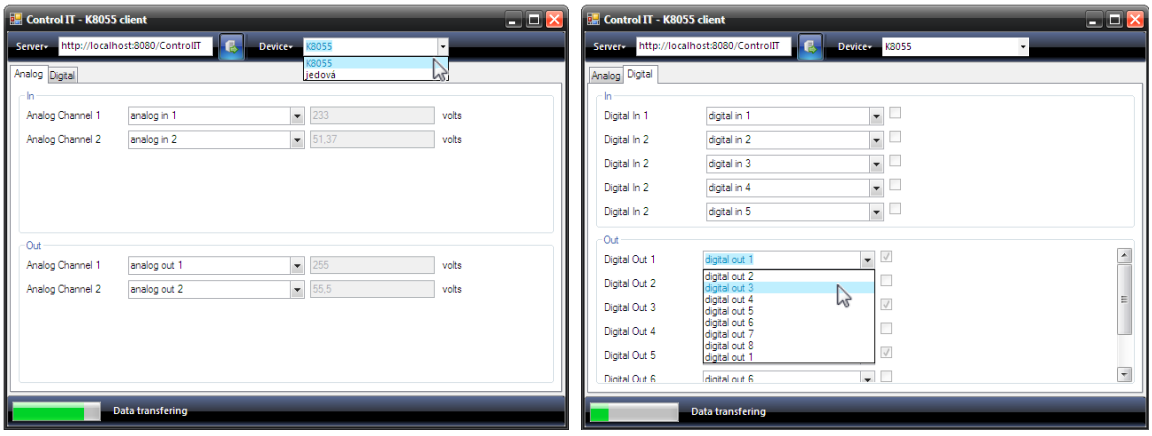
Obr. 41. Příklad založení instance klienta a komunikace

Z kódu je patrné, že zatímco definice adresy endpointu se získává z objektu *Properties.Settings*, kam jej smí skrze formulář vložit uživatel, konfigurace binding typu *WSHttpBinding* zapsaná přímo v *app.config* souboru se v rámci aplikace měnit nedá. Standardně je tedy použito zabezpečení komunikace s ověřováním pomocí Windows Autentizace. Aplikace tak alespoň se 100% jistotou zjistí, zda se připojila ke správnému hostiteli a průběžně ověřuje, jestli toto spojení stále existuje. V módu vypnutého zabezpečení se klientská aplikace „připojila" na jakoukoliv zadanou (i smyšlenou) adresu, což samozřejmě není reálné, natož žádoucí.

6.3.3 Přiřazení záznamů k portům

Za neměnnou součást každého zařízení můžeme považovat počet analogových a digitálních portů. Podle něj lze pak relativně jednoduše vytvořil klientskou aplikaci „na míru". Jiná situace ovšem nastává u zakládání údajů o portech v databázi, kde uživatel může ke každému zařízení zvolit jakýkoliv počet a typ portů. Nastává tedy problém – nutnost propojení patřičných reálných vstupů a výstupů s jejich ekvivalenty v databázi.

Program ControlIT Client obsahuje několik výběrových prvků, které volají skrze WCF vzdálené služby serveru a načítají do své kolekce položek objekty z databáze s informacemi o dostupných měřících portech. Uživatel tak musí nejprve vybrat název zařízení, které chce „namapovat" a následně pak pomocí výběrových seznamů přiřazuje jednotlivým typům vstupů a výstupů patřičné záznamy. Položky si aplikace po jejich vybrání uloží do své konfigurace, takže při příštím spuštění a pozitivním ověření existence záznamů v databázi vše opět správně přiřadí. Průběh výběru ilustruje obrázek (Obr. 42).



Obr. 42. Mapování databázového záznamu k reálnému portu

Průběh naplnění dat formulářového prvku pro „mapování“ analogových vstupů pomocí vzdáleného volání služeb WCF serveru ilustruje zdrojový kód v obrázku (Obr. 43). U ostatních datových typů funguje kód obdobným způsobem.

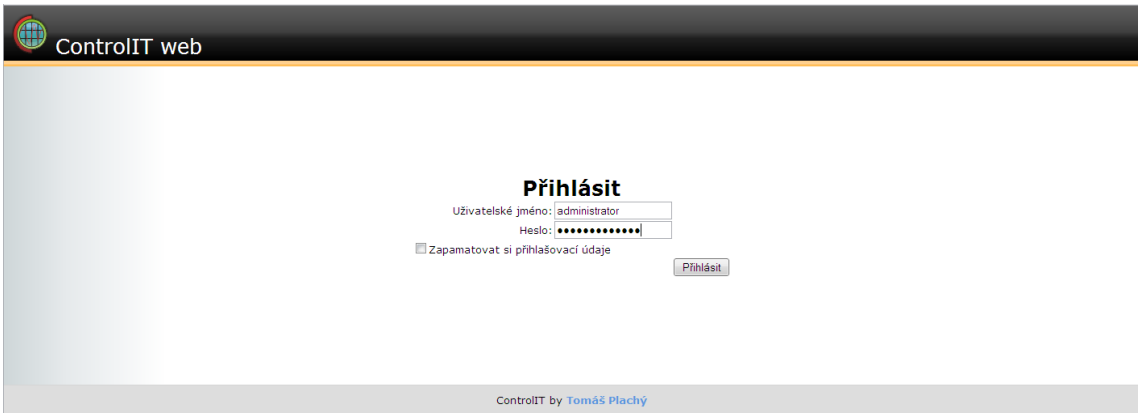
```
1  // postupný průchod kolekcí comboBoxů pro analogové vstupy
2  foreach (ComboBox comboBox in
   collectionControlIT_ClientFormAnalogInComboBox)
3  {
4     Naplnění instance listem objektů měřících portů
5     measuringBlocksList =
6     wcfClient.getMeasuringBlocks((int)Properties.Settings.
      Default.idCommunicationContainer);
7     // Vymazání stávajících položek
8     comboBox.Items.Clear();
9     // Naplnění instance portu analog hodnotou přiřazenou
      k danému měřícímu portu
10    analogPort =
      wcfClient.getAnalogFromServer(measuringBlock.Id);
11    // Filtr – jen vstupní hodnoty
12    if(analogPort.Direction == true)
13    {
14       // Přidání objektu do seznamu comboBoxu
15       comboBox.Items.Add(measuringBlock);
16    }
17 }
```

Obr. 43. Příklad kódu pro mapování databázových záznamů k prvku formuláře

6.4 Popis webové aplikace ControlIT web

Přístup do webové aplikace by měl být umožněn v rámci celé sítě Internet, v tomto případě se pak rapidně zvyšuje využitelnost celého měřícího systému, neboť je tak administrátorovi umožněn zásah z jakéhokoliv zařízení připojeného k sítí.

Přístupnost webového administračního rozhraní v Internetu ovšem přináší jistá bezpečností úskalí, proto každý z uživatelů musí disponovat svým loginem a heslem, přičemž před započetím práce s prostředím systém vyžaduje autorizované zalogování ve formuláři, který obsahuje obrázek (Obr. 44).



Obr. 44. Přihlašovací stránka prostředí ControlIT web

Po nainstalování webového rozhraní se automaticky vytvoří administrátorský účet s uživatelským jménem *administrator* a heslem *administrator*. Toto heslo je nutné po přihlášení co nejdříve změnit.

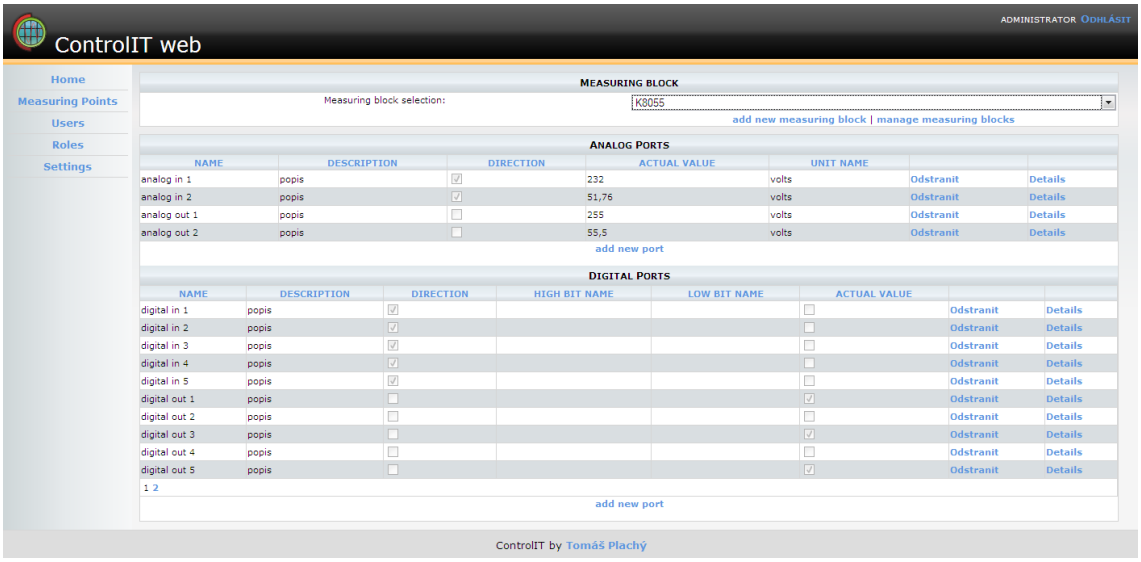
Uživatelské rozhraní webové administrace lze považovat za zcela intuitivní. Stěžejní prvek – hlavní menu nabízí uživatelům zobrazení všech hlavních modulů:

- **Home** – hlavní stránka webu s rychlými hlášeními
- **Measuring points** – soubor skriptů pro kompletní administraci měřících bodů
- **Users** – administrace uživatelů a jejich oprávnění
- **Roles** – spravuje jednotlivá uživatelská oprávnění a skupiny, v současné verzi webu relativně nepodstatný modul
- **Settings** – bude obsahovat možnosti nastavení webové aplikace, prozatím lze konfigurovat jen pomocí souboru *web.config*

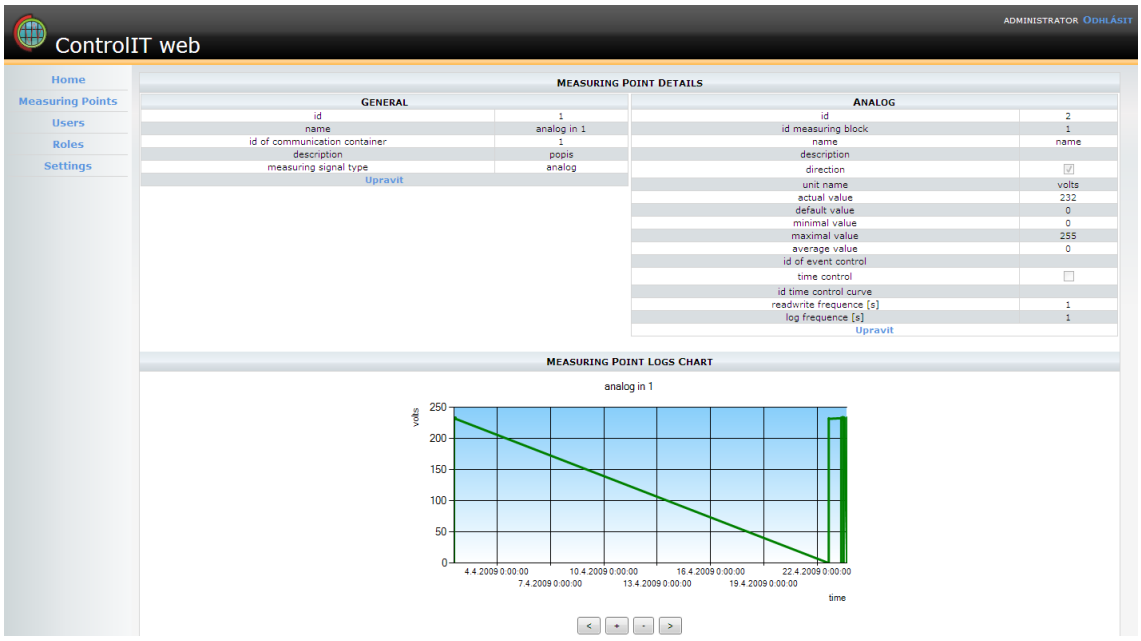
6.4.1 Administrace měřících bodů

Hlavní formulář pro správu měřících prvků (Obr. 45) obsahuje tabulku s rychlým přehledem stavů jednotlivých analogových a digitálních portů. Vyobrazené hodnoty nelze přímo měnit, k jejich ovlivnění se používá rozšířeného formuláře s detaily (Obr. 46), na který lze přejít klepnutím na URL odkaz *Details* u každého záznamu portu.

Přímo v hlavním formuláři *Measuring Points* smí uživatel vybírat z nabídky *comboBoxu* jednotlivá dostupná zařízení, přidávat nové záznamy a skrze podformuláře spravovat jejich stav. V rámci výpisu analogových a digitálních portů existuje odkaz na formulář přidávající další záznamy portů.



Obr. 45. Administrační formulář měřících bodů



Obr. 46. Formulář s detailem analogového portu

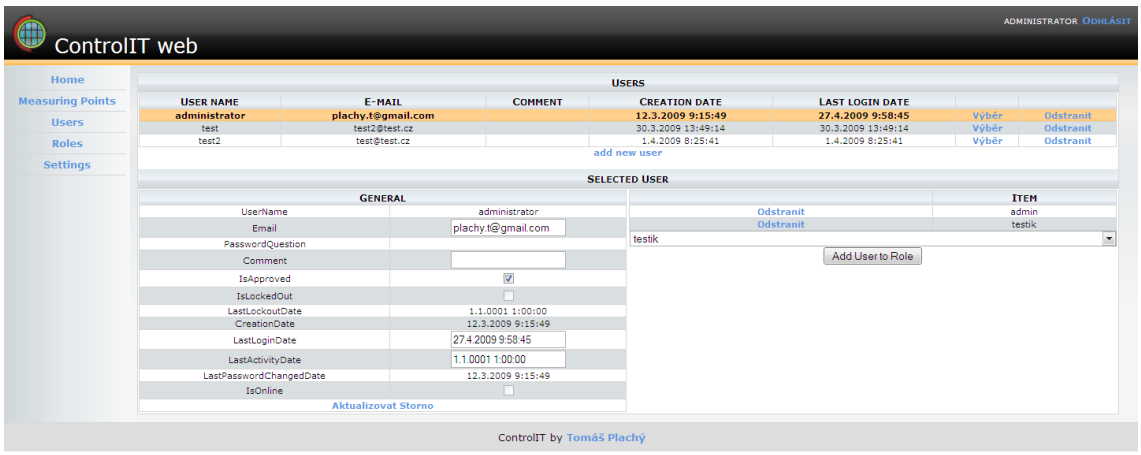
Většina administračních formulářů ve své podstatě vizualizuje a zpracovává data získaná z MS SQL Server databáze. Skripty formulářů tedy většinou využívají stejné ASP.NET komponenty, jejichž struktura a výběr je následující:

- **ScriptManager** – inicializuje prvky AJAXu.
- **UpdatePanel** – kontejner zajišťující asynchronní aktualizaci obsahu, který je v něm umístěn.
- **Timer** – vyvolává automatickou obnovu informací v určitém intervalu.

- **GridView** – vygeneruje tabulku podle kolekce dat získané z databáze, umožňuje seřazení, výpis detailů atd.
- **DetailsView** – vypíše konkrétní detaily o jednom záznamu databáze.
- **Chart** – vykresluje obrázek s grafem zaznamenaných hodnot. Vychází z dodatečné komponenty Microsoft Chart Control.
- **SqlDataSource** – vytváří most mezi datovým zdrojem a vizualizovanými daty. Disponuje jednodušší strukturou než ADO.NET.

6.4.2 Správa uživatelů

Správa uživatelů představuje kolekci skriptů pro vytvoření, úpravu a smazání jednotlivých účtů. Hlavní formulář (Obr. 47) vypisuje všechny zaregistrované uživatele, po rozklepnutí tlačítkem *Select* u některého ze záznamů se zobrazí tabulka se všemi detaily, především pak výpis skupin a rolí, kterými uživatel disponuje. V tomto formuláři lze příslušnost k rolím plně spravovat.



Obr. 47. Formulář správy uživatelů

Zatímco u ostatních formulářů se pro získávání a zadávání dat využívá především *SqlDataSource* komponenta, správa uživatelů je postavena na *MembershipUser*, tedy providerovi, který svými funkcemi zajišťuje kompletní správu uživatelů.

6.4.3 Grafy

Jak již uvádí předchozí kapitola, vykreslení grafů zajišťuje plug-in Microsoft Chart Control zastoupený ASP.NET komponentou *Chart*. Díky umístění v AJAXovém *UpdatePanelu* webová stránka poskytuje uživateli prostředí aktualizované informace. Ke grafu se navíc

váže několik ovládacích prvků vymezujících interval zobrazení průběhu zaznamenaných hodnot. Tyto prvky zastávají následující funkce:

- „<“ - posouvá pohled na křivku grafu doleva dynamicky podle velikosti zoomu.
- „+“ - přiblíží, tedy zmenší interval pohledu na křivku, a to dynamicky podle zobrazovaného rozsahu (jiné přiblížení pro dny a týdny, jiné pro sekundy).
- „-„ - oddálí, tedy zvětší interval pohledu na křivku, stejně dynamicky, jako v případě přiblížení.
- „>“ - posouvá pohled na křivku grafu doprava dynamicky podle velikosti zoomu.
- *Interval since* – komponenta kalendáře určující počátek zobrazené křivky grafu.
- *Interval to* – komponenta kalendáře určující konec zobrazené křivky grafu.

Během zoomování grafu probíhá přepočet počátečního a koncového data zobrazení. Pomocí vlastní metody *changeDateTimeInterval*, která využívá funkci *ToOADate* ze struktury *System.DateTime,* se navíc provede převod na jednočíselný OLE formát, kde dojde k přepočítání rozdílu horní a dolní meze, podle nějž se určí úroveň odstupu jednotlivých kroků zoomu a formát data u popisek os.

6.4.4 Využití technologie AJAX

Technologie AJAX, tedy asynchronní volání webového serveru (popsaná v kapitole 1.5.3), zvyšuje použitelnost celé aplikace. Uživatel může sledovat okamžité změny stavů v tabulkách portů a zároveň do těchto změn zasahovat. Princip činnosti formulářů využívajících aktualizaci dat skrze komponenty AJAXu je následující:

1. Načtení dat do tabulky formuláře umístěné v prvku *UpdatePanel.*
2. Po načtení stránky následuje zapnutí časovače. Standardně je tik nastaven na 2 sekundy, nižší doba obnovení by způsobovala zahlcení serveru dotazy.
3. Vyvolané přerušení zajistí znovunačtení dat v *UpdatePanelu.*
4. V případě editace údajů ve formulářích se z důvodu udržení editovaných dat automatický časovač vypíná. Po dokončení úprav probíhá pravidelná aktualizace dat dále.

Během vyvolání přerušení časovačem proběhne v kódu několik metod zajišťujících obnovení dat:

- **ajaxDataBinding** – metoda obnovující u jednotlivých *SqlDataSource* zdrojů pomocí funkce *DataBind* všechny záznamy.

- **dateTimeAjaxSet** – nastavení správného data a času dle hodnot v sessions, nastaví rozsahy kalendářů a grafu

Kapitola 1.5.3 přiblížila kolekci komponent ASP.NET AJAX Control Toolkit, jejíž některé prvky naleznou uplatnění v následujících verzích webového prostředí ControlIT Web. Konkrétně se plánuje použití následujících typů objektů:

- **SliderExtender** – vytvoří nestandardní objekt podobný slideru – šoupátku z *Windows Forms*. Tento prvek umožňuje plynulé nastavení číselné hodnoty pomocí myši, využití je plánováno pro nastavení a vizualizaci hodnot analogových portů.
- **Calendar** – rozbalovací kalendář s možností procházení měsíců a roků ve stylu Windows Vista. Umožňuje plné nastylování pomocí CSS, navíc oproti klasickému ASP.NET kalendáři zabírá mnohem méně místa na stránce.
- **DragPanel** – umožňuje pomocí myši pohyb prvků umístěných v komponentě *Panel* kdekoliv ve formuláři.
- **MultiHandleSlider** – pracuje podobně jako *SliderExtender*, ovšem na dráze slideru zobrazuje rovnou dvě šoupátka, takže tento prvek najde skvělé využití např. v nastavení horní a dolní hranice intervalu měření.
- **Tabs** – disponuje podobným vzhledem a chováním jako objekty záložek – karet ve *Windows Forms*. Umožní tedy přehlednější zobrazení dat ve formuláři a ušetří místo na ploše, které lze následně využít pro vložení grafu. Webové prostředí se navíc bude díky záložkám více podobat desktopové administraci.

6.5 Instalace aplikací ControlIT

Přiložené CD kromě samotných zdrojových kódů všech programů obsahuje též sadu instalátorů, které znatelně ulehčí zavedení a spuštění jednotlivých aplikací.

Instalační programy Microsoft Installer lze jednoduše vytvořit v rámci projektu Visual Studia. Vývojáři je poskytnuta rychlá cesta pomocí několika kroků průvodce, případně se nabízí několik obsáhlých designérů pro vytvoření adresářové struktury budoucí aplikace, možnost zápisu do registrů, případně editor obsahu samotného instalátoru.

Pro projekt ControlIT byly vytvořeny tři instalační aplikace:

- **ControlIT setup** obsahuje serverovou i klientskou aplikaci ControlIT. Toto řešení „vše v jednom“ se hodí především pro účely testování obou aplikací na jedné

stanici. V nabídce Start uživatel nalezne složku *ControlIT* se zástupci obou programů. Celou instalaci, především v systémech Windows Vista a Windows 2008 Server musí uživatel provést jako administrátor. Program pro simulaci měřící karty Velleman K8055 pracuje prakticky stejně jako klientská aplikace, ovšem neumožňuje připojení reálného hardware, neboť všechny porty nahrazuje formulářovými prvky.

- **ControlIT setup Server** provede instalaci pouze serverové aplikace. Složka krom samotného exe souboru navíc obsahuje dodatečné knihovny pro vykreslení grafů a práci s uživateli. V systémech Windows Vista se doporučuje nastavit oprávnění zápisu do složky s programem všem potřebným uživatelům. V nabídce Start spustí uživatel server ze složky *ControlIT Server*. I v tomto případě musí v systémech Windows Vista a Windows 2008 Server uživatel provést instalaci jako administrátor.
- **ControlIT setup Client** vytvoří v adresářové struktuře podobné serveru exe samotné aplikace včetně nezbytných knihoven nutných pro provoz programu. Spustit jej pak lze v nabídce Start ve složce *ControlIT Client*.

Před instalací všech programů se požaduje existence následujících systémových komponent:

- .NET Framework 3.5 SP1
- Microsoft SQL Server Express SP2
- V případě použití webové aplikace Internet Information Services s povoleným rozšířením ASP.NET

V případě nepřítomnosti některé ze jmenovaných součástí se je program pokusí stáhnout a nainstalovat do systému. Instalaci v systémech Microsoft Windows 2008 a Windows Vista musí uživatel provádět jako administrátor.

Zavedení webové aplikace proběhne nakopírováním obsahu zazipovaného adresáře *ControlIT_web.zip* do kořenové složky pro webové prezentace serveru IIS. Po následné konfiguraci ISS musí uživatel ještě správně nastavit parametry *ConnectionString* v souboru *web.config*.

7 PŘÍNOS SYSTÉMU, VÝZNAM PRO UŽIVATELE A SMĚR DALŠÍHO ROZVOJE

Kolekce programů ControlIT svou filozofií přináší univerzální možnost měření, logování a řízení pro široké spektrum projektů. Celý projekt se snažil převzít ty nejlepší vlastnosti konkrétního stávajícího měřícího systému pro monitoring vysílačů a zároveň změnit celkový přístup uživatele k ovládacímu prostředí.

7.1 Rozdíly oproti stávajícímu řešení

Za nejdůležitější rozdíl stávajícího systému a projektu ControlIT lze považovat především způsob zpracování dat a přístup do administračního prostředí. Zatímco nynější řešení tvoří komplexní na míru vytvořený desktopový software založený na Visual Basic 6.0, nové řešení přináší možnost umístění jednoduchého malého klienta, kterého není problém jednoduše přizpůsobit na daný hardware. O komunikaci s uživatelem a kompletní zpracování dat se stará jednotná serverová aplikace. Konkrétní rozdíly lze popsat v několika bodech:

- Jednu komplexní aplikaci nahrazuje soubor programů, které je možné jednodušeji vyvíjet zvlášť podle aktuální potřeby
- Jako úložiště dat slouží robustní Microsoft SQL Server, u stávajícího řešení se používá Microsoft Access databáze.
- Současná aplikace vystavěná na míru konkrétnímu zařízení neumožňuje stejně vysokou úroveň personifikace měřených portů.
- Hardware použitý v současném řešení lze považovat za morálně zastaralý, neboť komunikace probíhá skrze LPT port, jenž svými parametry a nedostatky pokulhává za moderním USB.
- Datová náročnost komunikace nového a starého řešení dosáhla značného rozdílu. Oproti použití vzdálené plochy volání WCF služeb minimálně vytěžuje komunikační linku.
- Přístup uživatele ke starému řešení na vzdáleném počítači byl možný pouze pomocí software pro práci se vzdálenou plochou. Nové řešení disponuje administračním rozhraním v serverovém software ControlIT Server, případně lze použít jednoduše přístupné webové prostředí.

- Klientská aplikace ControlIT Klient je připravena pro jednoduché rozšíření dle požadavků konkrétního hardware.
- Projekt ControlIT prozatím umožňuje měření a uživatelské řízení analogových a digitálních portů. Existuje ovšem již příprava na využití pulzní řízené modulace PWM.

7.2 Uvažovaný vývoj

Další vývoj měřícího a řídicího systému lze považovat díky bezpočtu hodin strávených nad dosavadním kódem za samozřejmý. Směr vývoje se bude s vysokou pravděpodobností ubírat následujícími fázemi:

- Odladění chyb první verze v ostrém provozu
- Využití komponent AJAX Control Toolkitu ve formulářích webové administrace
- Dořešení správy varovných hlášení – viz následující podkapitola 7.2.2
- Optimalizace logování
- Vytvoření modulu pro automatické řízení podle časové křivky

7.2.1 Vylepšení komunikace se serverem

Služby WCF v prvotní verzi projektu ControlIT může volat pouze regulérně autorizovaný uživatel, tedy klient běžící na platformě Windows s účtem stejným, jaký existuje zároveň na straně serveru. Tato vlastnost zajišťující ochranu před zneužitím je sice žádoucí, ovšem v případě využití klienta běžícího na zcela jiné platformě nastává zásadní problém – nemožnost přihlášení a využívání služeb. Z tohoto důvodu musí být vytvořena kolekce služeb založená na zcela jiném typu autorizace. Windows Communication Foundation sice umožňuje poskytovat služby bez nutnosti autorizace, ovšem během vývoje se projevily problémy s indikací stavu připojení na straně klienta. Celá záležitost tedy musí projít sérií testování.

7.2.2 Správa varovných hlášení

Systém varovných hlášení stojí a padá na propojení správy uživatelů a skupin se správou měřících portů. Každému portu tak může být přiřazeno n pravidel, která určují, co se bude v dané situaci dít. Tato pravidla jsou z druhé strany vázána na skupiny a konkrétní uživatele.

Varovná hlášení budou disponovat minimálně stejnými funkcemi, jaké nabízí stávající systém, tzn. odesílání e-mailů na konkrétní adresy vybraných uživatelů. V rozšířené verzi se uvažuje nad možností přímého zasílání SMS přes GSM bránu pomocí AT příkazů.

7.3 Nasazení systému do praxe

Projekt ControlIT během psaní diplomové práce procházel procesem ladění a testování. Jeho plnému nasazení na celé síti vysílačů rozhlasové stanice bude předcházet činnost v roli měřícího systému vykrývacího vysílače v Prostějově. Zařízení bude nejprve monitorovat teplotu prostředí (testování měření a vyhodnocení dat u analogových veličin) a napájení (testování měření a vyhodnocení dat u digitálních veličin), tedy chod některých komponent vysílače.

Tímto způsobem práce v reálném provozu se odkryje značná část potenciálních chyb a jistě dojde k doladění některých funkcí systému. Po dostatečném odladění celého systému může být nahrazeno stávající měřící zařízení na hlavním vysílači Jedová (Pohořany severovýchodně od Olomouce).

ZÁVĚR

Obsah diplomové práce probral tématiku vývoje specifického sledovacího software určeného pro vzdálené měření a ovládání rozhlasových vysílačů.

Jednotlivé dílčí technologie popsané v teoretické části představují jeden z dnešních možných směrů vývoje desktopových a internetových aplikací. Čtenář si tak může vytvořit hrubý přehled o možnostech integrovaného vývojového prostředí Microsoft Visual Studio, produktu Microsoft SQL Server, technologiích ASP.NET, WCF a dalších. Teoretická část dále popisuje vlastnosti zvoleného hardware pro řízení a měření včetně případných alternativ. Poslední kapitola pak čtenáři přiblíží aktuální stav možností vzdálené správy komponent reálného rozhlasového vysílače.

V rámci praktické části se práce zabývá definicí a popisem projektu ControlIT, který byl vytvořen jako nástupce současného řešení. Obsah kapitol postupně nastínil definici a analýzu potřebných funkcí nového systému. Z prvotní analýzy následující kapitola rozvíjí návrh architektury nového odbavovacího systému jak z pohledu software, tak i hardware. Tato část práce čtenáři představuje podrobný návrh struktury tabulek databáze a popis vlastností datových typů pro přenos a uložení informací sledovaných objektů.

Nejdůležitější úsek praktické části seznamuje se samotnou realizací projektu. Několik kapitol vyčerpávajícím způsobem popisuje jednotlivé dílčí aplikace projektu ControlIT, přičemž představeno není jen samotné uživatelské rozhraní všech programů, ale díky několika tabulkám a obrázkům se zdrojovými kódy se čtenářům dostává přehledu o činnosti důležitých funkcí kódu.

Praktickou část zakončuje kapitola shrnující popis rozdílů řešení, kde jsou probírány kladné i záporné vlastnosti obou systémů. Zájemcům o pohled do budoucnosti projektu se nabízí popis možné cesty vývoje, definice nových vlastností měřícího systému a návrh vylepšení uživatelského rozhraní. Celou praktickou část zakončuje krátký popis průběhu a stavu nasazení měřícího systému do praxe.

Z pohledu vývoje celého projektu se dopsáním diplomové práce nejedná o uzavřenou záležitost. Jednotlivé aplikace čeká dostatečné odladění díky nasazení do ostrého provozu. Po vychytání všech případných chyb a dodání dalších potřebných funkcí lze očekávat plný běh na některém z vykrývacích vysílačů, jehož funkčnost není tak stěžejní jako u hlavního celoplošného vysílače. Po dodání a vyzkoušení dalších kusů měřících desek se s nejvyšší

pravděpodobností začne uvažovat o nahrazení stávajícího hardware na hlavním vysílači. V případě, že se celé řešení osvědčí, naskýtá se možnost pro nabídku poupravené verze projektu ostatním rozhlasovým stanicím.

Filozofie a koncept programového vybavení umožňuje použít jednu hlavní aplikaci a jednotné uživatelské rozhraní pro několik různých měřících zařízení a lokací. Vysoká univerzálnost použití a možnost přístupu k ovládání skrze webové rozhraní, které je díky rozšířenosti připojení Internetu přístupné téměř kdekoliv, činí z kolekce aplikací nástroj s širokou možností uplatnění.

CONCLUSION

This thesis describes technology for development of software applications based on the .NET Framework and the production of monitoring system which is an example of use of this technology.

The theoretic part describes some individual items of technology .NET Framework. This involves possibilities of Microsoft Visual Studio, sub-technology ADO.NET, WCF and ASP.NET. The next chapter deals with the database product Microsoft SQL Server. Some readers may be interested in a description of the hardware for remote measuring and control, this represents the contents of the last chapter of the theoretic part. The main aim of theoretic part is demonstration of .NET development possibilities. The content is focused also on the comparison of individual products of Microsoft Company so that the reader can create some overview about possibilities of all offered products.

Within the frame of the practical part I dealt with solving and description of my present biggest project. According to the contract letting of the base analysis I was ordered to create a flexible and universal monitoring system, which should replace current morally outdated solution. This part of thesis describes an analysis of the original state and needs of our company for a new measuring and control system. Further description of the process of development ControlIT measuring system and his server, client and web modular application follow.

During implementation of this system I used latest technologies and processes, so its result is modular and robustness collection of applications built up on technology .NET Framework with possibility of universal usage also in quite different spheres of monitoring or control than the administration of transmitters only.

I do not consider this project as finished. I and the leadership of company, where was the project developed, plan the new modules and services. We will want to put this project into the practice after debugging of all application mistakes. After finish of all tests will be a system used on local area transmitter in Prostějov city. In the case of good functionality on this transmitter our company plans to purchase some additional measuring boards for the creating of a complete monitoring network for all transmitters.

SEZNAM POUŽITÉ LITERATURY

[1] **Troelsen, Andrew.** *C# a .NET 2.0 Profesionálně.* [editor] RNDr. Jan Pokorný. 1. vydání. Brno : Zoner Press, 2006. str. 1197. ISBN: 80-86815-42-0.

[2] **Troelsen, Andrew.** *Pro C# and the .NET 3.5 Platform.* 4. vydání. New York : Apress, 2008. str. 1370. ISBN: 1-59059-884-9.

[3] Microsoft Visual Studio. *Wikipedie - Otevřená encyklopedie.* [Online] [Citace: 06. Duben 2009.] http://cs.wikipedia.org/wiki/Visual_Studio.

[4] Visual Basic .NET. *Wikipedie - Otevřená encyklopedie.* [Online] [Citace: 06. Duben 2009.] http://cs.wikipedia.org/wiki/Visual_Basic_.NET.

[5] **Chmel, Marek.** Microsoft SQL Server 2005 – základní přehled. *Moderní správce.* [Online] 30. Říjen 2008. [Citace: 07. Duben 2009.] http://www.modernivyuka.cz/Serverov%C3%A9OSatechnologie/SQLServer/tabid/434/ctl/Details/mid/1339/ItemID/208/language/cs-CZ/Default.aspx.

[6] **Košťál, Marián.** Úvod do WCF. *Vyvojar.cz.* [Online] 1. únor 2007. [Citace: 6. duben 2009.] http://www.vyvojar.cz/Series/2-uvod-do-wcf.aspx.

[7] **MacDonald, Matthew a Szpuszta, Mario.** *ASP.NET 3.5 a C# 2008 - tvorba dynamických stránek PROFESIONÁLNĚ.* [překl.] Jan Gregor a RNDr. Jan Pokorný. 1. vydání. Brno : Zoner Press, 2008. str. 1584. ISBN: 978-80-7413-008-3.

[8] **Zakas, Nicholas C., McPeak, Jeremy a Fawcett, Joe.** *AJAX Profesionálně.* [překl.] Jiří Koutný. 1. vydání. Brno : Compute Press, 2007. str. 668. ISBN: 978-80-86815-77-0.

[9] **Libor Paláček.** Elektronický učební text pro MS SQL Server 2000. [Online] 06. Duben 2009. http://www.fs.vsb.cz/books/MSSQLServer/Index.htm.

[10] **Šimunek, Milan.** *SQL - kompletní kapesní průvodce.* 1. dotisk. Praha : Grada Publishing, 1999. str. 247. ISBN: 80-7169-692-7.

[11] **Fox, Dan.** *Naučte se ADO.NET za 21 dní.* [překl.] Bogdan Kiszka. 1. vydání. Brno : Computer Press, 2002. str. 508. ISBN: 80-7226-772-8.

[12] **Wenz, Christian.** *Programming ASP.NET AJAX.* 1. vydání. místo neznámé : O'Reilly, 2007. str. 473. ISBN: 0-596-51424-7.

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

AJAX Asynchronous JavaScript and XML – technologie umožňující asynchronní přenos dat mezi webovou stránkou u klienta a serverem¨.

ADO.NET ActiveX Data Objects – technologie pro práci s daty vyvinutá společností Microsoft.

ASP.NET Active Server Page pro .NET Framework.

B2B Business-to-business – značení pro obchodní vztahy mezi obchodními společnostmi, které neobsluhují konečné spotřebitele v masovém měřítku.

B2C Business-to-customer – aplikace, kde dochází k obchodních vztahům mezi obchodními společnostmi a koncovými zákazníky, realizované webovými aplikacemi - virtuálními obchody na Internetu.

CSS Cascading Style Sheets – Kaskádové styly.

GUI Graphical User Interface – Uživatelské grafické rozhraní.

HTML Hyper Text Markup Language – Hypertextový značkovací jazyk.

MSIL Microsoft Intermediate Language – přechodový jazyk mezi některým z jazyků podporovaných Visual Studiem (VB.NET, C#, ...) a strojovým kódem.

.NET Aplikační plaforma vyvinutá společností Microsoft.

WCF Windows Communication Foundation.- technologie zajištující vzdálené využívání služeb.

WPF Windows Presentation Foundation – technologie zajišťující velmi flexibilní GUI aplikací.

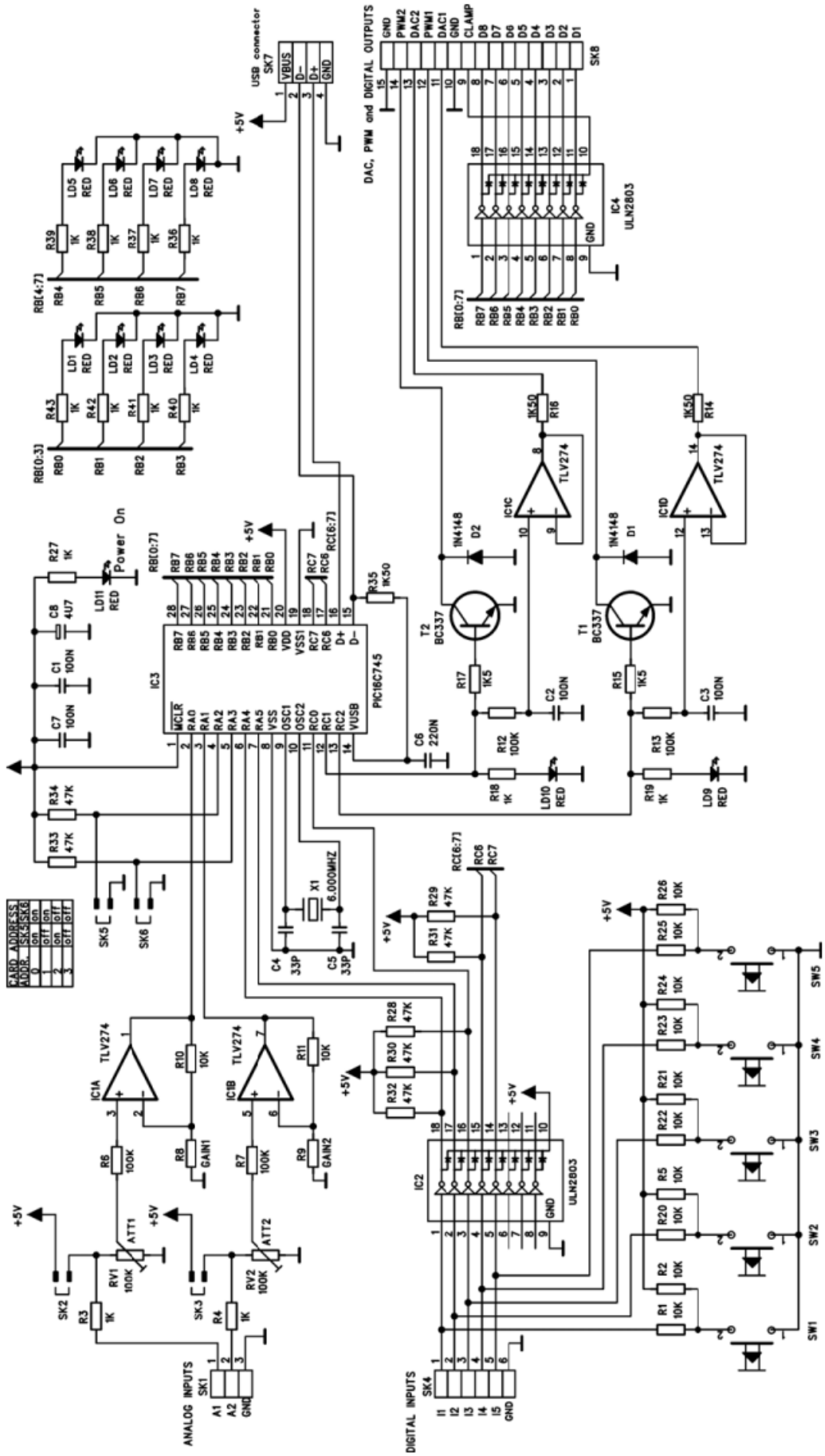
XHTML eXtensible HyperText Markup Language – Hypertextový jazyk založený na XML.

SEZNAM OBRÁZKŮ

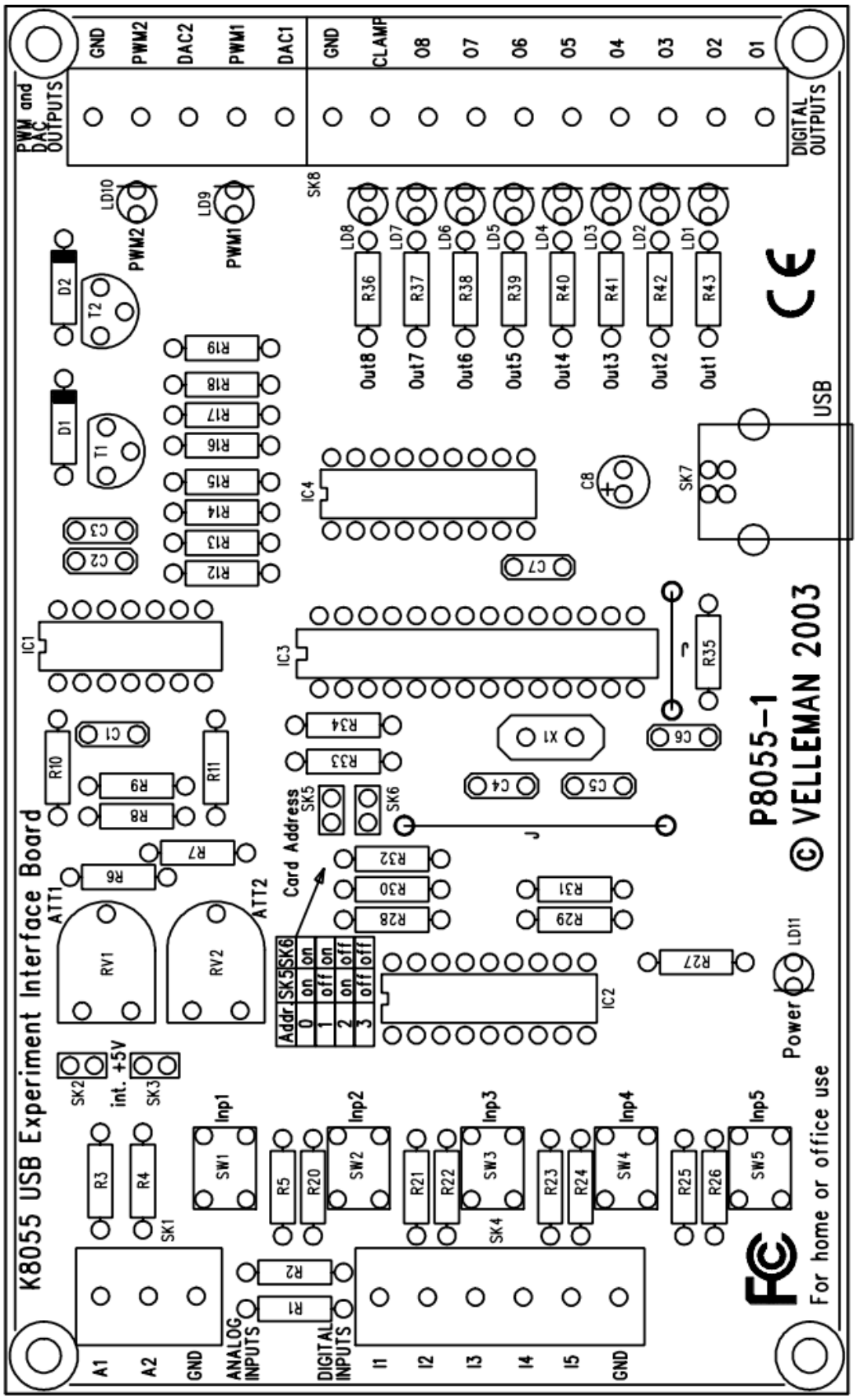
SEZNAM TABULEK

SEZNAM PŘÍLOH

PŘÍLOHA P I: SCHÉMA MĚŘÍCÍ DESKY VELLEMAN K8055



PŘÍLOHA P II: PCB MĚŘÍCÍ DESKY VELLEMAN K8055



PŘÍLOHA P III: ZAPOJENÍ MĚŘÍCÍ DESKY VELLEMAN K8055