



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA STROJNÍHO INŽENÝRSTVÍ
ÚSTAV AUTOMATIZACE A INFORMATIKY

FACULTY OF MECHANICAL ENGINEERING
INSTITUTE OF AUTOMATION AND COMPUTER SCIENCE

SOFTWARE PRO OVLÁDÁNÍ PROGRAMOVATELNÝCH REGULÁTORŮ S WEBOVÝM ROZHRANÍM

OPERATING SOFTWARE WITH WEB INTERFACE FOR PROGRAMMABLE CONTROLLERS

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. TOMÁŠ MATĚJKA

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. PAVEL HOUŠKA, Ph.D.

Vysoké učení technické v Brně, Fakulta strojního inženýrství

Ústav automatizace a informatiky

Akademický rok: 2009/2010

ZADÁNÍ DIPLOMOVÉ PRÁCE

student(ka): Bc. Tomáš Matějka

který/která studuje v **magisterském navazujícím studijním programu**

obor: **Aplikovaná informatika a řízení (3902T001)**

Ředitel ústavu Vám v souladu se zákonem č.111/1998 o vysokých školách a se Studijním a zkušebním řádem VUT v Brně určuje následující téma diplomové práce:

Software pro ovládání programovatelných regulátorů s webovým rozhraním

v anglickém jazyce:

Operating software with web interface for programmable controllers

Stručná charakteristika problematiky úkolu:

Cílem práce je vytvořit software, který umožní přístup a ovládání programovatelných regulátorů přes webové rozhraní. Pro řešení práce je doporučeno použití technologií ASP.NET, Remoting a SilverLight, obsažených v Microsoft .NET Framework.

Cíle diplomové práce:

1. Prostudujte problematiku uživatelského rozhraní programovatelných regulátorů, porovnejte výhody / nevýhody webového rozhraní a samostatné aplikace,
2. Navrhněte koncepci software pro ovládání programovatelných regulátoru,
3. Navrhněte způsob zabezpečení a řízení práv uživatelů pracujících s webovým klientem,
4. Navržený software realizujte.

Seznam odborné literatury:

- [1] Evjen B., Hanselman S., Rader D.: Professional ASP.NET 3.5 In C# and VB, Wiley Publishing, Indianapolis 2008, ISBN 978-0-470-18757-9
- [2] Microsoft: Visual Studio 2008 Developer Center [web], URL:<<http://msdn.microsoft.com/en-us/vs2008/default.aspx>>
- [3] Microsoft: Silverlight [web], URL:<<http://silverlight.net/>>

Vedoucí diplomové práce: Ing. Pavel Houška, Ph.D.

Termín odevzdání diplomové práce je stanoven časovým plánem akademického roku 2009/2010.

V Brně, dne

L.S.

Ing. Jan Roupec, Ph.D.
Ředitel ústavu

prof. RNDr. Miroslav Doupovec, CSc.
Děkan fakulty

ABSTRAKT

Tato diplomová práce se zabývá vytvořením software, který umožní přístup a ovládání programovatelných regulátorů přes webové rozhraní. Software je postaven na architektuře klient-server, přenos dat mezi serverem a klientem je realizován prostřednictvím technologie Windows Communication Foundation integrované v Microsoft .NET Framework od verze 3.0. K realizaci řešení byl použit programovací jazyk C# a knihovny Microsoft .NET Framework, vizualizace dat ve webové aplikaci byla realizována pomocí technologie Microsoft Silverlight.

ABSTRACT

This diploma thesis deals with developing of software, which allows access and control of programmable controllers through web interface. Software uses client-server model and for communication between client and server is used Windows Communication Foundation interface which is integrated in Microsoft .NET Framework from version 3.0. Applications were written in C# programming language using .NET Framework libraries. Data visualization in web application was created by Microsoft Silverlight.

KLÍČOVÁ SLOVA

Programovatelné regulátory, C#, .NET Framework, ASP.NET, Silverlight

KEYWORDS

Programmable controllers, C#, .NET Framework, ASP.NET, Silverlight

PODĚKOVÁNÍ

Tímto bych rád poděkoval panu Ing. Pavlovi Houškovi, Ph.D. za všechny cenné rady, připomínky a odborné vedení, které mi poskytl během vzniku této práce.

OBSAH

ZADÁNÍ DIPLOMOVÉ PRÁCE	3
ABSTRAKT	5
PODĚKOVÁNÍ.....	7
1 ÚVOD	13
2 ŘÍZENÍ TECHNOLOGICKÝCH PROCESŮ.....	15
2.1 Počítačové řídicí systémy	15
2.1.1 Centralizované systémy řízení.....	15
2.1.2 Distribuované systémy řízení	15
2.1.3 Distribuované hierarchické řízení.....	16
2.2 Využití osobních počítačů k řízení	17
2.2.1 Hlavní charakteristiky	18
2.2.2 Výhody a nevýhody webových aplikací.....	18
3 REGULÁTORY	21
3.1 Kompaktní regulátory	22
3.1.1 Modulární regulátory.....	22
3.2 Číslicové regulátory	22
3.3 Programovatelné regulátory.....	26
3.3.1 Struktura programu programovatelného regulátoru	26
3.3.2 Ovládání programovatelných regulátorů	27
3.4 Komunikační rozhraní	28
3.4.1 Diagnostika závad	29
4 POUŽITÉ TECHNOLOGIE	31
4.1 Microsoft .NET Framework.....	31
4.1.1 Struktura a základní prvky .NET	31

4.2	ASP.NET	33
4.2.1	Základní prvky ASP.NET	34
4.3	Windows Communication Foundation.....	34
4.3.1	Služby	35
4.3.2	Další klíčové vlastnosti infrastruktury WCF	37
4.4	Programovací jazyk C#	37
4.4.1	Charakteristika jazyku	37
4.4.2	Prvky jazyku	38
4.5	Microsoft Silverlight	39
4.5.1	Historie technologie	39
4.5.2	Základní charakteristika.....	40
4.5.3	Porovnání Microsoft Silverlight a Adobe Flex.....	40
4.6	Visual Studio	41
5	KONCEPCE REALIZOVANÉHO ŘEŠENÍ.....	43
5.1	Architektura systému.....	43
5.2	Přístup a ovládání regulátorů.....	44
6	DESKTOPOVÁ APLIKACE	47
6.1	Práce s regulátory	47
6.1.1	Připojení regulátoru	47
6.1.2	Stav regulace	48
6.1.3	Požadovaná hodnota	48
6.1.4	Nastavení regulačních veličin a konfigurace zařízení	49
6.2	Vzdálené ovládání regulátorů a jeho zabezpečení	49
6.2.1	Nastavení vzdáleného ovládání.....	50
6.2.2	Správa vzdálených uživatelů.....	51
6.2.3	Komunikace s klienty	53

7	WEBOVÁ APLIKACE	55
7.1	Přihlášení uživatele	55
7.2	Práce s regulátory.....	55
7.2.1	Monitoring.....	56
7.3	Propojení se serverem	57
7.3.1	Nastavení komunikace.....	57
8	INSTALACE	59
8.1	Certifikáty	59
8.1.1	Tvorba self-signed certifikátů.....	59
8.2	Instalace desktopové aplikace.....	62
8.3	Instalace webové aplikace.....	62
8.3.1	Serverový certifikát webové aplikace	62
8.3.2	Hostování webové aplikace na IIS	63
8.3.3	Nastavení udržování stavu relace	64
9	ZHODNOCENÍ VLASTNOSTÍ REALIZOVANÉHO ŘEŠENÍ	65
10	ZÁVĚR.....	67
	SEZNAM POUŽITÉ LITERATURY.....	69
	PŘÍLOHY.....	73

1 ÚVOD

V dnešní době, kdy je Internet součástí našeho každodenního života, roste i tlak ze strany uživatelů programovatelných regulátorů a řídicích jednotek na jejich ovládání prostřednictvím Internetu. To s sebou nese několik závažných problémů, především z pohledu spolehlivosti a bezpečnosti. Před připojením těchto zařízení do Internetu je nutné vyřešit, jak zabránit útokům ze strany hackerů, kteří se, z pro normálního člověka neznámých důvodů, snaží napadat jednotlivá zařízení na Internetu a převzít nad nimi kontrolu, nebo je alespoň vyřadit z provozu. Dalším problémem je, že dovybavení programovatelného regulátoru nebo řídicí jednotky o rozhraní pro připojení k Internetu v mnoha případech výrazně zvyšuje jejich cenu a spotřebu elektrické energie.

Zde je nutné si uvědomit, že cenu těchto zařízení tvoří pouze hardwarové náklady, ale převážně i cena softwaru, která v případě rozšíření o potřebnou „internetovou gramotnost“ může být několikanásobně vyšší než cena softwaru potřebného pro vlastní funkčnost regulátoru nebo řídicí jednotky. Výrobci u většiny dnešních zařízení implementují jednoduché komunikační rozhraní, které umožňuje připojení k PC nebo notebooku servisní technik pomocí rozhraní TIA/EIA 232, 485 nebo USB.

Cílem této práce je využít již existujícího řešení pro připojení k PC a prostřednictvím tohoto PC zpřístupnit připojená zařízení pod různými úrovněmi oprávnění na Internetu s využitím moderních technologií, které jsou implementované v moderních operačních systémech pro PC.

V práci byla zvolena softwarová architektura klient-server, kde server je desktopová aplikace, která se stará o komunikaci s programovatelnými regulátory a řídicími jednotkami a klient je webová aplikace, která komunikuje se serverem pomocí technologie Windows Communication Foundation (WCF). Tato technologie, určená pro tvorbu distribuovaných aplikací, byla zvolena zejména kvůli jednoduchosti implementace a zabezpečení komunikace s použitím certifikátů.

Technolog, který má na starosti proces řízení, má díky desktopové aplikaci plnou kontrolu nad připojenými programovatelnými regulátory a zároveň má možnost povolení jejich vzdáleného ovládání uživatelům, jejichž správa je také v aplikaci umožněna. Uživatelé jsou rozděleni do uživatelských skupin a podle úrovně oprávnění, která daná skupina má, může uživatel buďto pouze sledovat stav regulace nebo do ní

i zasahovat. V praxi to znamená, že ředitel společnosti může mít přístup ke všem programovatelným regulátorům, ale nemůže například ovlivňovat jejich nastavení, zatímco operátor může mít přístup pouze k regulátorům, které má na starosti, a u nich může nastavovat například pouze žádanou hodnotu.

Toto řešení tedy odděluje programovatelné regulátory a řídicí jednotky od přímého přístupu z Internetu a zprostředkovává pouze ty vlastnosti a činnosti, které provozovatel uzná za vhodné a komu uzná za vhodné.

2 ŘÍZENÍ TECHNOLOGICKÝCH PROCESŮ

Donedávna byly budovány zvlášť počítačové systémy určené pro řízení technologických procesů a zvlášť počítačové informační systémy zaměřené spíše do oblasti ekonomiky a zpracování dat. Pokud byly tyto dva systémy propojeny, dělo se tak v omezené míře a hranice mezi oběma systémy byla zřetelná. V důsledku rozvoje počítačových sítí došlo k průniku obou oblastí a dnes je situace taková, že snad pouze s výjimkou procesních počítačů napojených přímo na technologii, může každý počítač v síti plnit v podstatě jakékoliv funkce. Tato kapitola čerpá z [1], [2] a [3].

Hlavními důvody využívání počítačových řídicích a informačních systémů na úrovni řízení technologie jsou:

- zajištění co nejdokonalejšího dodržování technologických podmínek;
- maximální využití kapacity technologického zařízení a kvality produkce;
- eliminace nespolehlivého lidského faktoru z procesu řízení.

2.1 Počítačové řídicí systémy

Počítačové řídicí systémy jsou charakterizovány jejich částmi, jejich propojením a tím, jak se podílejí na plnění daných funkcí. Základní rozdělení struktur počítačových řídicích systémů je:

- centralizované řízení;
- distribuované řízení;
- distribuované hierarchické řízení.

2.1.1 Centralizované systémy řízení

Tento druh řízení používá jediného řídicího počítače pro řízení celé výroby. Ten je napojen na všechny části technologického procesu, přijímá z něj veškeré informace a vydává veškeré povely. Centralizovaný způsob řízení byl obvyklý v počátcích používání řídicích počítačů, kdy byl počítač náročný na umístění, velký a drahý. Dnes se používá jen při malých aplikacích, např. k samostatnému řízení jednotlivých aparátů.

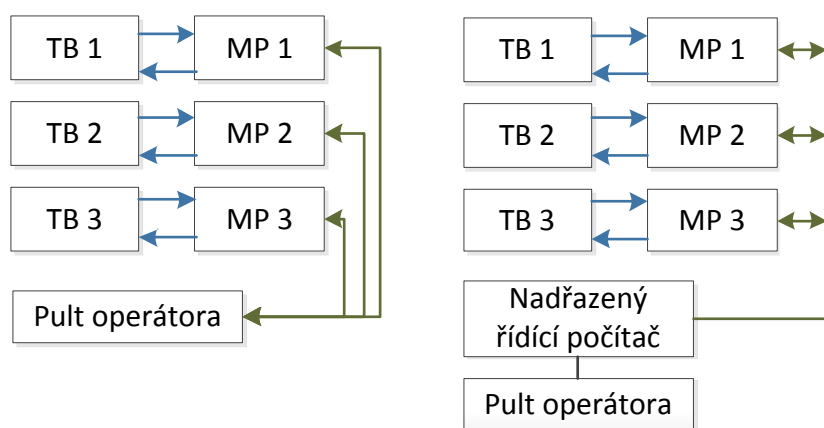
2.1.2 Distribuované systémy řízení

Princip distribuovaného řízení spočívá v tom, že řídicích (procesních) počítačů je více a jsou rozmístěny přímo u jednotlivých technologických bloků nebo

u jednotlivých aparátů. Jsou napojeny komunikačním systémem na pult operátora, kam se přenášejí informace ze všech počítačů a odkud se všem počítačům vydávají povely.

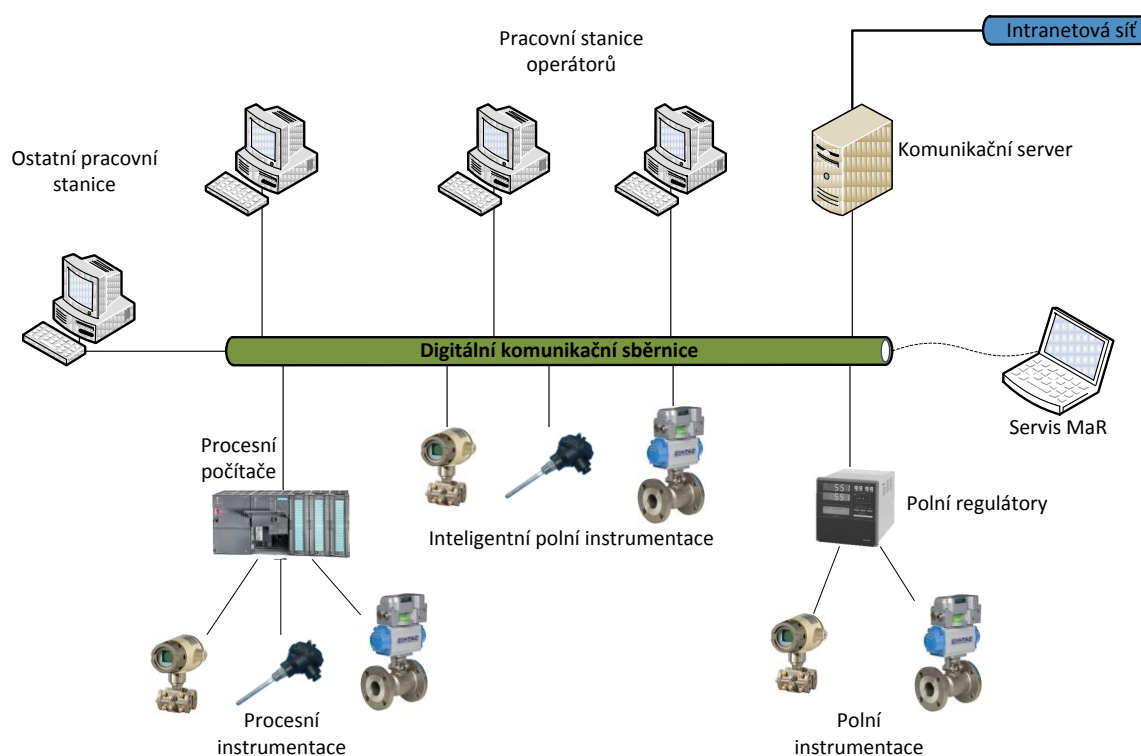
2.1.3 Distribuované hierarchické řízení

Distribuované hierarchické řízení se od pouhého distribuovaného řízení liší tím, že jednotlivé řídicí (procesní) počítače nejsou napojeny na pult operátora, ale na nadřazený řídicí počítač, na kterém jsou rovněž provozovány některé řídicí algoritmy. Rozdíl mezi distribuovaným a distribuovaným hierarchickým řízením je patrný z obrázku 1, kde TB jsou technologické bloky a MP místní řídicí (procesní) počítače.



Obr. 1: Distribuované řízení (vlevo) a hierarchické distribuované řízení (vpravo) [2].

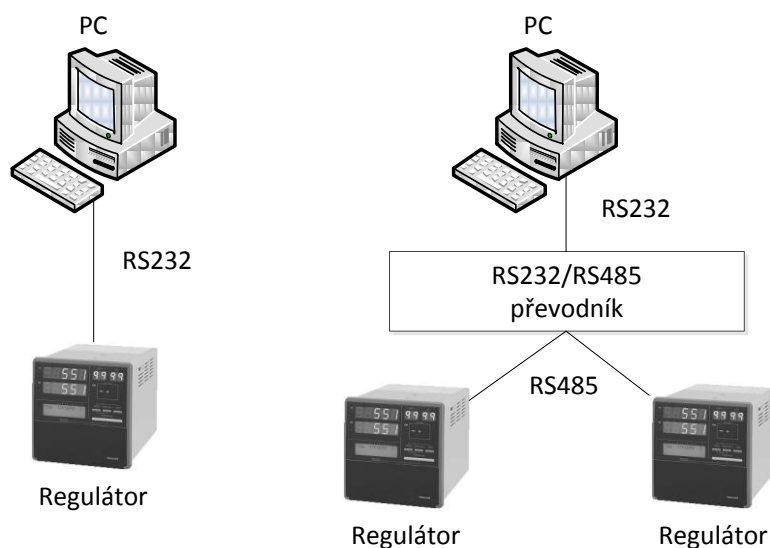
V dnešní době se používá prakticky výhradně hierarchické struktury. Základní úroveň bývá osazena programovatelnými automaty a procesními počítači a jako nadřazené počítače se používají podle potřeby pracovní stanice na bázi PC, technologické verze PC, nebo větší a výkonnější počítače. Hierarchických hladin může být několik. Co se týče topologie prvků distribuovaného hierarchického řízení, tak v současné době je zcela volná, protože jako komunikační systém slouží podniková počítačová síť, popřípadě síť Internet. Rozhraní mezi jednotlivými hierarchickými úrovněmi tu nejsou patrná, umístění prvku do určité hladiny není dáno propojením, ale jeho programovým vybavením a funkcí v celém systému. Příklad moderního řídicího a informačního systému je na obrázku 2.



Obr. 2: *Struktura moderních řídicích a informačních systémů [3].*

2.2 Využití osobních počítačů k řízení

Pro menší aplikace a pro výzkumné a vývojové účely se často používají řídicí systémy tvořené jedním osobním počítačem doplněným kartami analogových a binárních vstupů a výstupů. Tyto karty pracují se standardními signály a lze k nim tedy připojovat běžná čidla, akční členy nebo regulátory.



Obr. 3: Přímé připojení regulátoru do počítače (vlevo) a připojení více regulátorů přes převodník (vpravo).

2.2.1 Hlavní charakteristiky

Na počítači běží lokální aplikace pro monitorování, případně i ovládání regulované soustavy. V případě požadavku dostupnosti ovládání z Intranetu nebo Internetu může tento počítač fungovat i jako webový server [4].

Mezi hlavní výhody patří:

- řádově nižší cena oproti rozsáhlým počítačovým systémům řízení;
- rychlost implementace do výrobního procesu;
- izolovanost jednotlivých stanic v případě počítačového útoku.

Hlavní nevýhody pak jsou:

- decentralizace systému řízení, zálohování a správy práv;
- osobní počítač není odolný vůči vlivům prostředí.

2.2.2 Výhody a nevýhody webových aplikací

Hlavní výhodou webových aplikací oproti klasickým aplikacím je jejich dostupnost z jakéhokoliv počítače připojeného k síti Internet. Mezi další výhody patří:

- nejsou nutná žádná nastavení ani žádné aplikace (kromě webového prohlížeče) na klientském počítači;
- jednoduchá aktualizovatelnost (každý uživatel vždy používá nejnovější verzi aplikace);

- centralizace zdrojů dat, jejich údržba a zálohování.

Nevýhody jsou:

- nutnost zajištění autorizace a autentizace uživatelů a zabránění možnosti odposlechu při přenášení citlivých dat;
- rychlost chodu aplikace závisí na rychlosti připojení klientského počítače k síti Internet (Intranet);
- nutnost zajištění kompatibility s různými druhy a verzemi webových prohlížečů;
- omezení plynoucí z možností protokolu HTTP při komunikaci.

3 REGULÁTORY

Ze základů teorie řízení je známo, že analogové regulátory, jejichž chování je popsáno pomocí diferenciálních rovnic a realizováno prostřednictvím prvků analogové elektroniky popř. pneumaticky, je možné aproximovat pomocí diskrétních regulátorů, které jsou popsány diferenčními rovnicemi a realizovány pomocí číslicových počítačů. Číslicové počítače, obvykle jednočipové, jsou tak základním stavebním prvkem číslicových regulátorů, podobně jako jím v případě analogových elektronických regulátorů byly operační zesilovače. Celá tato kapitola čerpá z [5], [6] a [7].

Moderní průmyslové regulátory se sice mohou navenek chovat i jako regulátory analogové, mají však oproti nim řadu výhod. Především jsou flexibilnější, protože regulační funkce popsané příslušným algoritmem se realizují softwarově, a při změně procesních podmínek se dají velmi pružně měnit. Jsou přesnější, protože digitálně realizované parametry jsou bez posunu (drift). Dále mohou obsahovat zavedení jakýchkoliv matematických operací a korekcí. Nabízejí velkou škálu přídatných funkcí, které jsou realizovatelné programově. Mezi tyto funkce patří například:

- výpočet regulačních algoritmů s adaptivním nastavením (optimalizací) parametrů regulátoru;
- ošetření plynulého (beznárazového) přechodu z ručního režimu do automatického;
- zabránění nežádoucímu nasycení integrační složky (wind up) při dosažení omezení akční veličiny;
- efektivní strukturalizace složitých regulačních parametrů (kaskádní regulace, fuzzy logika, neuronové sítě, integrace některých funkcí PLC aj.);
- náběhové funkce (rampy) s volitelnými parametry;
- účinná digitální filtrace rušených signálů;
- řízení samokontroly a autodiagnostiky;
- tvorba a správa časových programů v reálném čase.

Uvedené funkce ve svém souhrnu zvyšují inteligenci a užitnou hodnotu regulátorů.

Z hlediska konstrukčního provedení regulátorů je možné rozlišit tyto dvě základní varianty:

- kompaktní regulátory;
- modulární regulátory.

3.1 Kompaktní regulátory

Kompaktní regulátory se vyznačují tím, že v jediném pouzdře soustředí všechny hardwarové a softwarové komponenty, které jsou potřebné pro řešení běžných regulačních úloh v různých oblastech průmyslové automatizace. Moderní průmyslové kompaktní regulátory mají vedle napájecího zdroje zpravidla pět základních částí – vstupní, ústřední, výstupní, ovládací a komunikační a dodávají se komplexně připravené k připojení k regulovanému procesu.

Možnost uživatelsky změnit konfiguraci jednou dodaného přístroje je však omezená. Většinou ji lze modifikovat v určitém rozsahu, například pro použití jiného typu termočlánku, případně jiného teplotního čidla. Samozřejmě lze vždy užít standardní napěťové a proudové signály, avšak rozsáhlejší změna konfigurace obvykle není možná.

3.1.1 Modulární regulátory

Modulární regulátory rozšiřují možnosti kompaktních regulátorů o změnu počtu a provedení vstupů a výstupů. Jejich změnou či doplněním lze měnit konfiguraci regulátoru ve větším rozsahu, než je tomu u regulátorů kompaktních.

3.2 Číslicové regulátory

Zjednodušenou strukturu číslicového regulátoru je možné blokově vyjádřit uspořádáním znázorněným na obrázku 4. Jeho centrem je mikroprocesor a potřebné paměti:

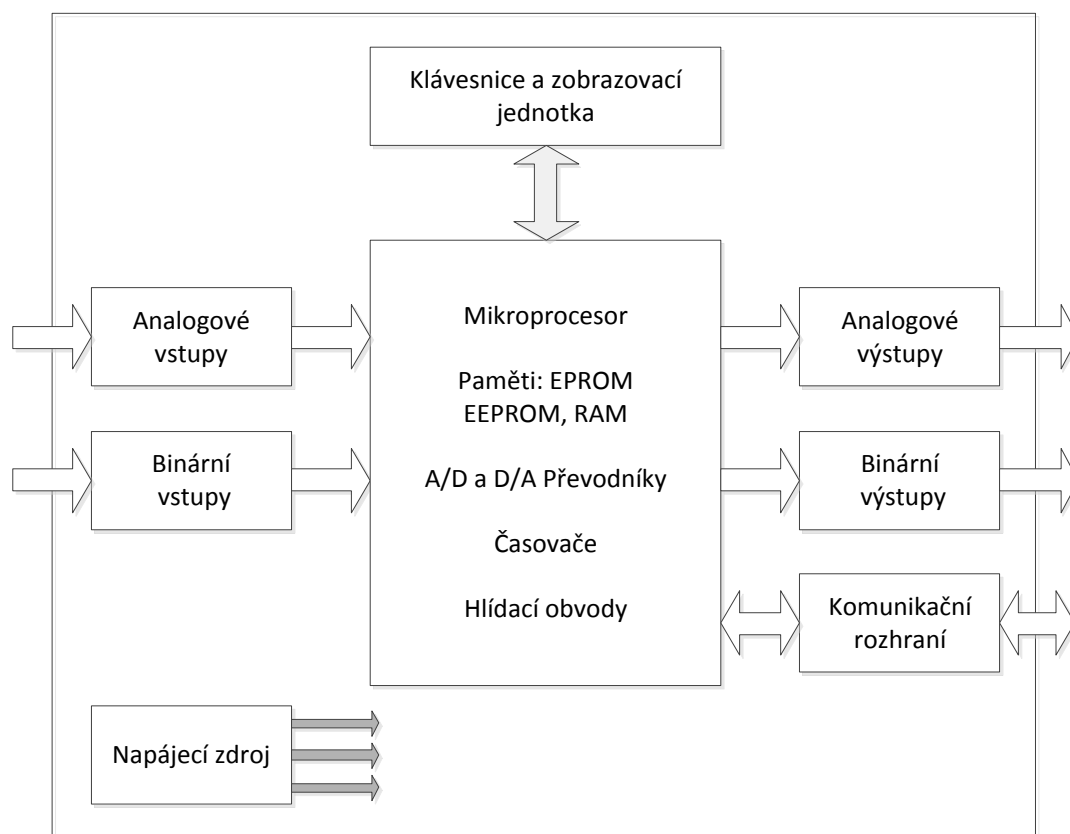
- ROM/EPROM na uložení programu;
- EEPROM resp. Flash EEPROM na uložení parametrů, které je třeba měnit, jejichž hodnota však zároveň musí být zachována i při odpojení napájecího napětí;
- operační paměť RAM.

Jelikož regulátor řídí procesy, které mají spojitou povahu, jsou dále zapotřebí A/D a D/A převodníky.

Všechny algoritmy číslicového řízení vyžadují ke své činnosti přesnou a stabilní časovou základnu zabezpečující, že perioda vzorkování bude konstantní, popř. se bude měnit podle požadavku regulačního algoritmu. Nezbytnou součástí regulátoru jsou proto časovací obvody, které jsou obvykle realizované pomocí přesného krystalem řízeného oscilátoru a proměnného děliče kmitočtu.

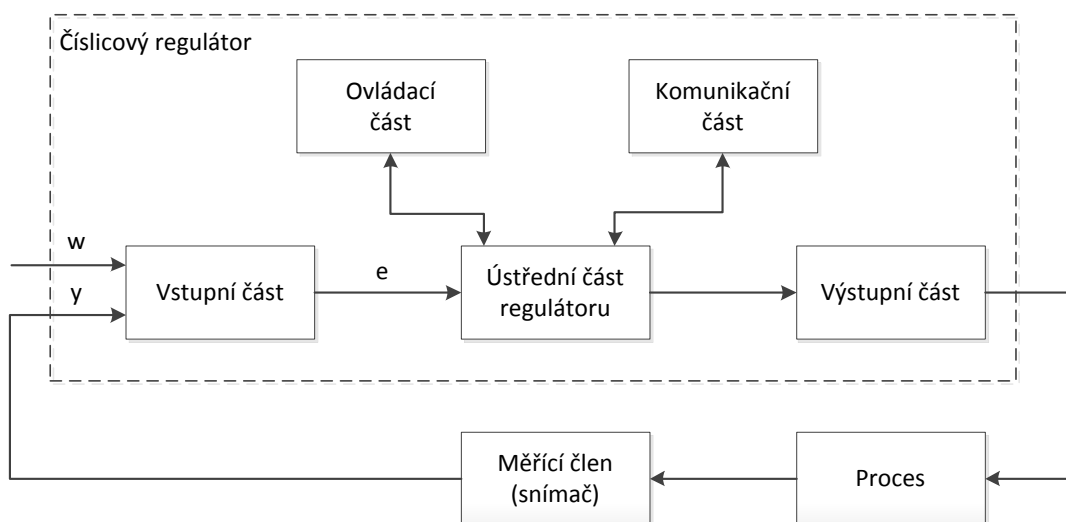
Dále jsou obvykle zapotřebí různé pomocné hlídací obvody zabezpečující definované chování regulátoru v případě vybočení napájecího napětí z přípustných mezí, zacyklení či jiného nesprávného chování programu v důsledku skryté chyby nebo rušení pronikajícího k mikroprocesoru po napájecím napětí či jinými cestami.

V současnosti lze všechny tyto funkce realizovat jedním obvodem, tzv. jednočipovým mikropočítačem, který na jednom čipu integruje vedle vlastního mikroprocesoru a paměti i další funkční bloky včetně A/D a někdy i D/A převodníků. V jednoduchých regulátorech určených např. k jednosmyčkové regulaci teploty či jiných pomalých procesů se stále běžně používají osmibitové mikroprocesory nebo jednočipové mikropočítače, ve složitějších zařízeních pak spíše 16-ti nebo 32-ti a výjimečně i 64 bitové mikroprocesory.



Obr. 4: Blokové schéma číslicového regulátoru [6]

Do vstupní části regulátoru se přivádějí informace o skutečné hodnotě regulované veličiny (y) z procesu ve formě výstupního signálu příslušného měřicího členu. Zde se vstupní hodnota převádí do číslicové podoby a po úpravě (zesílení, linearizaci, filtraci aj.) se porovnává se žádanou hodnotou regulované veličiny (w). Výsledkem porovnání je regulační odchylka (e) v digitálním tvaru. Standardně má regulátor jeden i více analogových vstupů, které lze bez úprav hardwaru volně naprogramovat pro příjem unifikovaných elektrických signálů i pro všechny běžné linearizace připojených snímačů (teploty, tlaku aj.). Dražší modely dovolují nakonfigurovat vlastní uživatelskou převodní charakteristiku měřicího snímače. Schéma mikroprocesorového regulátoru je na obrázku 5.



Obr. 5: Obecné blokové schéma mikroprocesorového regulátoru [5].

V ústřední části regulátoru probíhá vlastní zpracování regulační odchylky. Klíčovou součástí je zde mikroprocesor s jeho podpůrnými obvody, který z aktuální hodnoty regulační odchylky podle zvoleného typu regulačního algoritmu určuje velikost akčního zásahu do regulovaného procesu. Možnosti konfigurace mikroprocesorových regulátorů jsou:

- dvou nebo třípolohový regulátor;
- krokový nebo spojitý regulátor;
- pro poměrovou, vlečnou, kaskádní, popř. volně programovatelnou regulaci;
- pro použití matematického či logického modulu;
- možnost zadání několika interních žádaných hodnot;
- možnost volby z několika sad zpětnovazebních parametrů;
- hlídání nastavených mezí;
- volba fuzzy logiky pro zlepšení dynamických vlastností procesu;
- volba optimalizační funkce.

Obslužné programy (firmware), nastavitelné parametry, data a veškeré proměnné jsou ukládány do energeticky nezávislých pamětí typu EEPROM apod., kde zůstávají uchovány i po vypnutí regulátoru nebo v případě výpadku napájení po dobu delší než 100 let.

Ve výstupní části regulátoru se upravuje výsledek číslicového zpracování regulační odchylky pro ovládání použitého typu akčního členu. Nejčastěji se digitální výstup ústřední části převede převodníkem D/A na unifikovanou analogovou výstupní

veličinu (0 až 20 mA, 0 až 10 V aj.), popř. se využije v binárním tvaru jako reléový či tranzistorový spínaný výstup regulátoru.

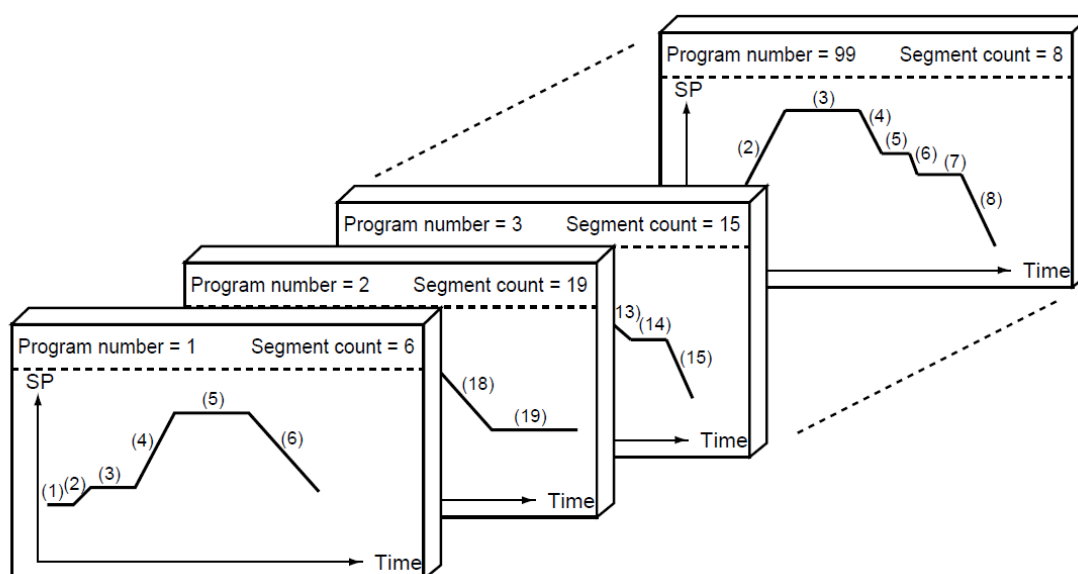
3.3 Programovatelné regulátory

Programovatelné regulátory rozšiřují možnosti číslicových regulátorů o možnost vytváření složitých regulačních struktur (v některých případech i vlastních regulačních algoritmů) na základě uživatelem definovaných programů (pravidel).

3.3.1 Struktura programu programovatelného regulátoru

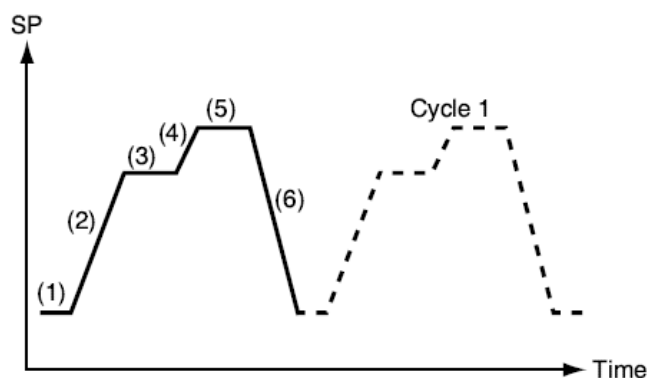
Dříve striktní rozdíl mezi programovatelnými automaty, jakožto prvky pro logické řízení, a číslicovými regulátory, jakožto zařízeními určenými k realizaci funkcí dříve vykonávaných spojitými analogovými regulátory, za použití programovatelných regulátorů, díky jejich rozsáhlým možnostem, částečně mizí. S programovatelností se lze setkat častěji u modulárních systémů, existují však i programovatelné kompaktní regulátory.

Každý regulátor může obsahovat několik na sobě nezávislých programů, které mohou obsahovat různý počet segmentů, přičemž každý program musí obsahovat alespoň jeden segment (obr. 6).



Obr. 6: Schéma programů a jejich segmentů u programovatelných regulátorů [7].

Regulátory umožňují nastavení opakování programů v tzv. cyklech, což je znázorněno na obrázku 7.



Obr. 7: Znázornění opakování programu v cyklu [7].

Nejčastěji se u programovatelných regulátorů využívá takzvané časové osy, kdy se žádaná hodnota mění v čase. Mezi další závislosti žádané hodnoty patří např.:

- závislost na otáčkách motoru;
- závislost na teplotě;
- závislost na tlaku.

3.3.2 Ovládání programovatelných regulátorů

Hlavním rysem uživatelského rozhraní programovatelných regulátorů je jeho jednoduchost. Většinou obsahuje pouze několik ovládacích tlačítek a jednoduchý displej pro zobrazení řídicí veličiny. Dražší modely mohou obsahovat více displejů (případně jeden velký LCD displej), na kterých se mohou zobrazovat:

- hodnoty uživatelem definovaných proměnných programu;
- číslo aktuálního programu a segmentu;
- aktuální stav programu (READY, RUN, HOLD, END);
- systémová hlášení a chyby;
- grafy (u modelů vybavených LCD displejem);
- další informace související s aktuálním stavem řízené soustavy.



Obr. 8: Programovatelný regulátor Honeywell DCP100.

3.4 Komunikační rozhraní

Důležitou součástí regulátorů je také komunikační rozhraní, které je umožňuje provozovat nikoliv jen jako izolovaná zařízení, ale také jako součást rozsáhlejšího distribuovaného řídicího systému. Konkrétní podoba a použitelnost tohoto rozhraní bývá ovšem velmi různá.

Často vybavení regulátoru komunikačním rozhraním znamená, že je na něm k dispozici pouze konektor pro připojení rozhraní TIA/EIA 232C, 422 nebo 485, přičemž však protokol, podle něhož komunikace probíhá, není popsán. V tomto případě je komunikace omezena pouze na možnost spojení s jinými zařízeními popř. programovým vybavením téhož výrobce a její praktická použitelnost je proto jen omezená. Regulátory však mohou být vybaveny i podstatně lépe a umožňovat připojení k nějaké standardizované průmyslové sběrnici (CAN, Profibus DP, Foundation Fieldbus apod.) nebo být konfigurovatelné i pro připojení k několika různým typům těchto sběrnic.

Některé modely mají i rozhraní pro připojení k síti Ethernet (např. 10Base-T podle normy IEEE 802.3) a mohou data přenášet s podporou komunikačního protokolu

TCP/IP, což jim zajišťuje možnost přístupu na síť Internet (popř. Intranet). Každý regulátor je jednoznačně identifikován svou IP adresou a může fungovat jako webový server umožňující monitorovat, regulovat nebo řídit technologické procesy pomocí webového prohlížeče.

3.4.1 Diagnostika závad

Diagnostika provozních závad regulátorů se postupně změnila od jednoduché indikace provozních a poruchových stavů pomocí barevných LED na ovládacím panelu až k důmyslným softwarovým rutinám, které cyklicky kontrolují interní toky dat v regulátoru a při zjištění chyby automaticky zobrazí na displeji chybové hlášení často i s kódem pravděpodobné příčiny. Komunikační schopnosti regulátorů umožňují zasílat výsledky diagnostiky na dálku přes telefonní modem, Internet nebo síť GSM. Servisní pracovník tak může obdržet e-mail nebo krátkou textovou zprávu o aktuálním stavu regulátoru přímo na svůj počítač nebo mobilní telefon a po jejím posouzení předat uživateli stejnou cestou nebo telefonicky pokyny pro úpravu konfigurace, nové seřízení nebo odstranění případné závady regulátoru, což snižuje dobu odstranění závady.

4 POUŽITÉ TECHNOLOGIE

4.1 Microsoft .NET Framework

Platforma .NET byla oficiálně představena firmou Microsoft v roce 2000 jako klíčový produkt, jehož rozvoj a propagace je součástí dlouhodobé strategie firmy. První verze pak byla vydána v únoru roku 2002. Microsoft .NET znamená novou generaci systému vývoje aplikací, který je založený na řízeném běhovém prostředí (CLR – Common Language Runtime), obohaceném o sadu základních tříd, nesoucích jméno .NET Framework. Tato kapitola čerpá z [8], [9], [10], [11] a [12].

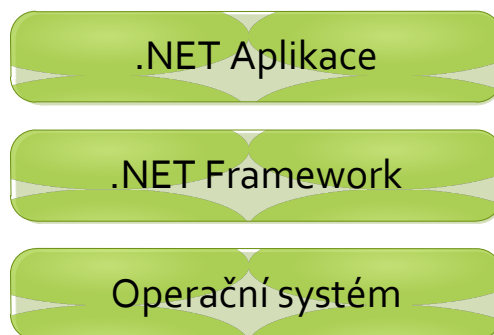
Hlavní důvody pro vývoj platformy byly:

- usnadnění a urychlení vývoje aplikací;
- zjednodušení instalace a údržby aplikací;
- zavedení kompatibility mezi programovými nástroji;
- snížení vysoké chybovosti aplikací;
- odstranění nekompatibility překladačů různých programovacích jazyků;
- zastaralý a nepřehledný způsob vývoje webových aplikací (ASP).

Všechny tyto problémy efektivně řeší platforma .NET, a to použitím již zmíněného řízeného běhového prostředí, systémem assemblies, což jsou základní stavební prvky aplikací, a novou technologií ASP .NET pro vývoj webových aplikací.

4.1.1 Struktura a základní prvky .NET

Srdcem platformy .NET je jádro nazvané .NET Framework, které je nadstavbou nad operačním systémem. V současné době existuje v těchto verzích: 1.0, 1.1, 2.0, 3.0, 3.5 a 4.0. Nejnovější verze 4.0 byla oficiálně uvedena na trh 12. 2. 2010.



Obr. 9: Vztah .NET Frameworku k operačnímu systému a aplikaci postavené na NET.

Na úplně nejnižší úrovni ve struktuře .NET Framework se nachází společné běhové prostředí realizující základní infrastrukturu, nad kterou je celý .NET Framework vybudován. Nad CLR se nachází několik sad hierarchicky umístěných knihoven. Základem je knihovna základních tříd (BCL – Base Class Library), nad ní následuje podpora pro přístup k datům, práci s XML, reflexi, atd. Obě tyto knihovny vytvářejí objektově orientovanou podporu pro efektivní a jednotný vývoj aplikací.

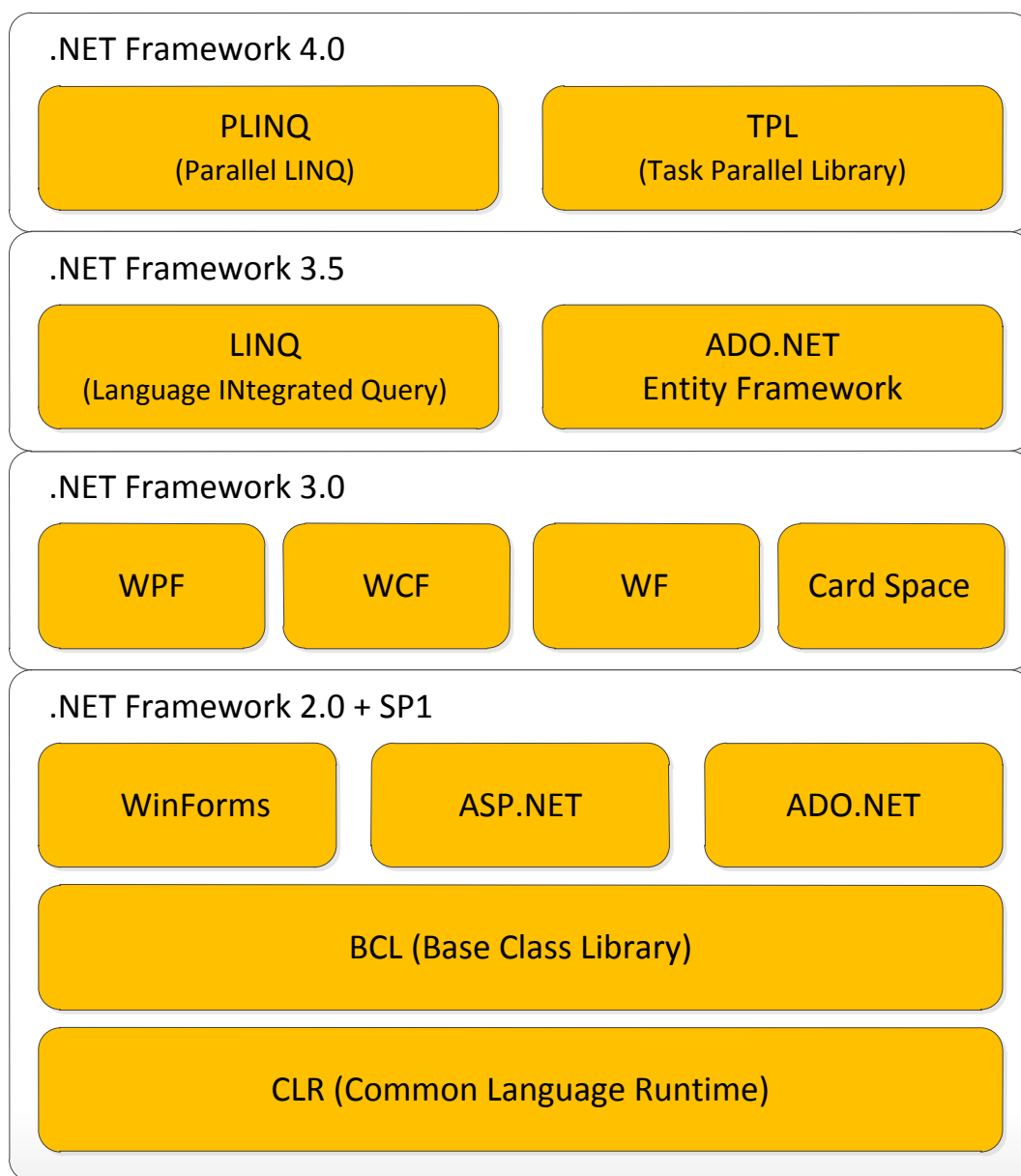
Další vrstvu tvoří neomezená množina programovacích jazyků. Ve spodní části se nachází Common Language Specification (CLS) definující základní vlastnosti, jež jsou očekávány v každém programovacím jazyce na platformě .NET. Nad ní následují jednotlivé programovací jazyky. Tato množina není uzavřená a libovolný výrobce nebo vývojář ji může rozšířit o novou položku.

S příchodem verzí 3.0 a 3.5 byl .NET Framework rozšířen o tyto knihovny:

- WPF (Windows Presentation Foundation) – k tvorbě „bohatého“ uživatelského rozhraní;
- WCF (Windows Communication Foundation) – sjednocuje většinu dřívějších technologií určených pro komunikaci včetně webových služeb;
- WF (Windows Workflow Foundation) – framework pro modelování, vývoj a správu aplikací založených na workflow;
- Windows CardSpace – systém určený k vytváření vztahů mezi weby a online službami;
- LINQ (Language INtegrated Query) – podpora dotazování nad daty přímo z .NET programovacího jazyku – až ve verzi 3.5;
- ADO.NET Entity Framework – plnohodnotný O-R mapper pro .NET – až ve verzi 3.5.

Nejnovější verze 4.0 pak přidává především:

- TPL (Task Parallel Library) – paralelní vykonávání úkolů aplikace;
- PLINQ (Parallel LINQ) – nadstavba pro paralelizaci LINQ dotazů.



Obr. 10: Struktura .NET Framework [12].

4.2 ASP.NET

ASP.NET je technologie určená k tvorbě webových stránek, aplikací a služeb. Poprvé byla představena v lednu roku 2002 (po čtyřech letech vývoje) jako součást .NET Frameworku verze 1.0. Ačkoliv název ASP.NET je odvozen od starší technologie pro vývoj webů ASP (Active Server Pages), obě tyto technologie jsou velmi odlišné. Hlavní výhody ASP.NET oproti ASP jsou:

- zvýšení výkonu aplikací díky kompilovanému kódu;
- podpora různých programovacích jazyků;

- schopnost ukládat celou stránku nebo pouze její části do vyrovnávací paměti (caching), což podstatně snižuje zátěž serveru;
- programovatelné ovládací prvky;
- podpora událostmi řízeného programování;
- komponenty založené na XML;
- podpora uživatelských účtů a rolí;
- podobný přístup jako k aplikacím pro Windows zjednodušuje přechod od jednoho prostředí k druhému;
- jednodušší rozšiřitelnost;
- jednodušší nastavitelnost aplikace.

Kapitola čerpá z [13] a [14].

4.2.1 Základní prvky ASP.NET

Technologie ASP.NET poskytuje, hlavně díky konceptu WebForms a událostmi řízeným programováním, programátorům klasických desktopových aplikací snadnější přechod k tvorbě webových aplikací. Mezi její hlavní rysy patří:

- stránky (oficiálně nazývané WebForms) – jsou základním prvkem při tvorbě webových aplikací. Skládají se ze statického ((X)HTML) obsahu, Web Controls a User Controls, které mohou obsahovat statickou i dynamickou část;
- code-behind model – odděluje prezentační a obchodní logiku. Code-behind soubory mají obvykle stejný název souboru jako ASPX stránka doplněný o příponu značící použitý programovací jazyk (např.: *.cs v případě programovacího jazyku C#);
- obsluha stavu aplikace – ačkoli je webový protokol HTTP bezstavový, technologie ASP.NET podporuje zachování stavu aplikace (např. pomocí technologie ViewState).

4.3 Windows Communication Foundation

V minulosti bylo při vytváření aplikací využívajících servisně orientovanou architekturu (SOA – Service Oriented Architecture) na výběr několik technologií, pomocí kterých aplikace komunikovala s okolím. Zpravidla platilo, že programátor

napsal program podle toho, jaké bylo nejlepší využití dané technologie pro účel, který potřeboval. Příkladem takových technologií jsou:

- ASP.NET Web Services,
- Web Services Enhancements (WSE),
- .NET Remoting,
- Enterprise Services,
- Microsoft Message Queuing (MSMQ) a další.

Velkou nevýhodou těchto technologií je, že vývoj aplikace závisí na použité technologii. Další nevýhodou je to, že tyto technologie nejsou vzájemně kompatibilní, což výrazně znesnadňuje práci s nimi.

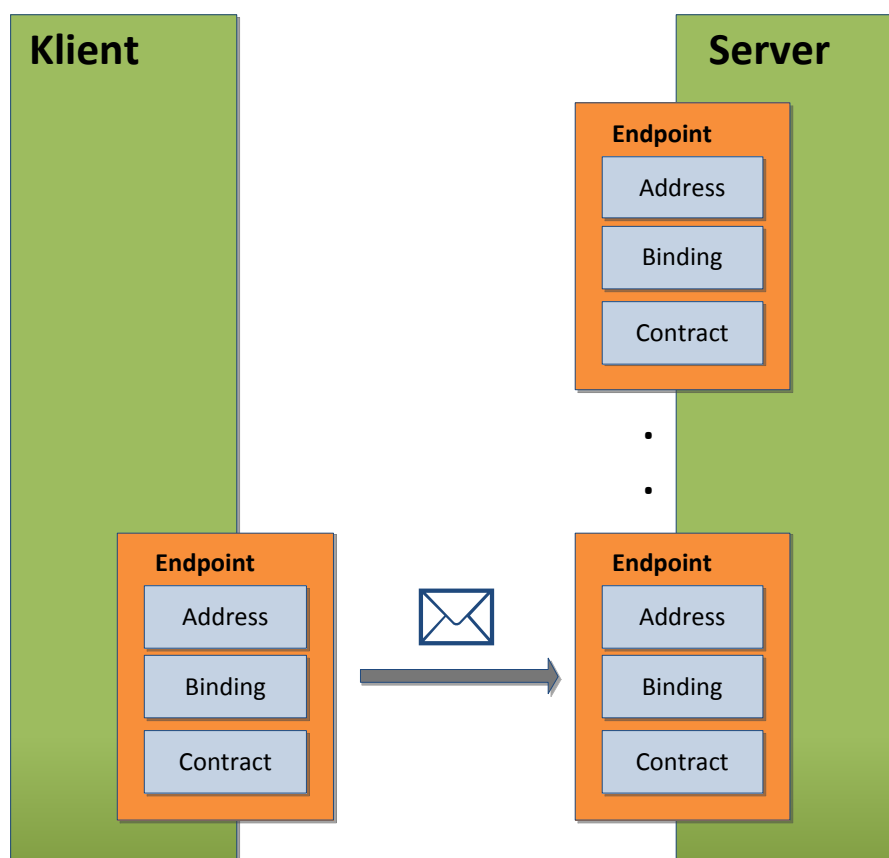
Windows Communication Foundation (WCF) je infrastruktura, která sjednocuje všechny tyto dřívější technologie od firmy Microsoft. Z toho vyplývá, že programátor nemusí vůbec znát konkrétní programový model dané technologie, ale stačí mu znát koncepci WCF. WCF v tomto ohledu přináší velké zjednodušení pro programátory. WCF má rychlost komunikace stejnou nebo vyšší, než dřívější technologie a navíc nabízí možnost jednoduchého nastavení bezpečnosti a dalších vlastností služby.

WCF je založené na komunikaci pomocí zpráv. Zpráva (message) je skupina dat obsahující záhlaví a tělo zprávy. Příkladem zprávy je například HTTP požadavek nebo MSMQ zpráva. Model WCF rozděluje aplikace na klienty a na služby. Klient je aplikace, která iniciuje komunikaci, služba je aplikace, která čeká na požadavky klientů. Kapitola čerpá z [15] a [16].

4.3.1 Služby

Základním stavebním prvkem WCF aplikací jsou služby (services). Služba je systém, který poskytuje jeden nebo více koncových bodů (endpoints), které slouží pro příjem a odesílání SOAP zpráv (messages). Tvoří ji tři základní části:

- třída služby, což je její samotná implementace;
- hostovací prostředí neboli místo, kde služba poběží;
- jeden nebo více koncových bodů.



Obr. 11: Komunikace klienta se službou [16].

Koncový bod

Koncový bod (endpoint) je ve službě místo, které slouží k přijímání a odesílání zpráv. Tvoří jej tři části:

- adresa (address) – říká, kde služba běží, tedy kam budou zasílány zprávy;
- binding – definuje, jakým způsobem bude služba komunikovat, tedy jaký komunikační protokol je zvolen, použité kódování, ale také druh zabezpečení a další;
- kontrakt (contract) – specifikuje rozhraní, které služba poskytuje, metody a další. Kontrakt je nezávislý na volbě adresy a bindingu.

SOAP zprávy

SOAP (Simple Object Access Protocol) zprávy jsou zprávy zasílané mezi klientem a serverem (službou) a jsou nezávislé na přenosovém protokolu. Je to požadavek a odpověď po nějakém komunikačním kanále. Přesnější model posílání zpráv (messaging patterns), které WCF podporuje, je následující:

- one way – klient odešle zprávu službě a neočekává odpověď;
- request-response – klient pošle požadavek a čeká na odpověď;
- duplex – obousměrná komunikace mezi klientem i službou probíhající asynchronně (služba může vynutit spuštění metody na straně klienta).

4.3.2 Další klíčové vlastnosti infrastruktury WCF

Hostování (hosting) – služba musí být hostovaná v nějakém procesu. Host je aplikace, která kontroluje životní cyklus služby.

Kanál (channel) – je prostředí, ve kterém se přenášejí zprávy. Předtím, než si dvě aplikace začnou posílat zprávy, musí se mezi nimi vytvořit kanál směrem ke koncovému bodu.

Metadata – popisují službu. Tento popis používají externí systémy, aby dokázali komunikovat se službou. Metadata mohou být zpracována pomocí nástroje Service Model Metadata Utility Tool (*Svcutil.exe*) a použita k vytvoření WCF klienta.

4.4 Programovací jazyk C#

C# (C Sharp) je vysokoúrovňový objektově orientovaný programovací jazyk vyvinutý firmou Microsoft zároveň s platformou .NET Framework, později schválený standardizačními komisemi ECMA (ECMA-334) a ISO (ISO/IEC 23270). Microsoft založil C# na jazycích C++ a Java (a je tedy nepřímým potomkem jazyku C, ze kterého čerpá syntaxi). Tato kapitola čerpá z [17], [18] a [19].

4.4.1 Charakteristika jazyku

V jazyce C# neexistuje vícenásobná dědičnost – to znamená, že každá třída může být potomkem pouze jedné třídy. Toto rozhodnutí bylo přijato, aby se předešlo komplikacím a přílišné složitosti, která je spojena s vícenásobnou dědičností. Třída však může implementovat libovolný počet rozhraní. Dále neexistují žádné globální proměnné a metody. Všechny funkce a metody musí být deklarovány uvnitř tříd. Náhradou za ně jsou statické metody a proměnné veřejných tříd.

V objektově orientovaném programování (OOP) se z důvodu dodržení principu zapouzdření často používá vzor, kdy k datovým atributům třídy lze zvenčí přistupovat pouze nepřímo a to pomocí metod *get* (accessor) a *set* (mutator). V C# lze místo toho definovat tzv. vlastnosti (properties), které zvenčí stále fungují jako datový atribut,

ale uvnitř vlastnosti se mohou definovat get a set metody. Výhodou je jednodušší práce s datovým atributem při zachování principu zapouzdření.

C# je typově bezpečnější než C++. Jediné výchozí implicitní konverze jsou takové, které jsou považovány za bezpečné jako rozšiřování Integerů (např. z 32 bitového na 64 bitový) nebo konverze z odvozeného typu na typ rodičovský. Neexistuje implicitní konverze z typu Integer na Boolean, ani mezi výčtovým typem Enum a typem Integer.

C# neobsahuje a ani nepotřebuje dopřednou deklaraci – není důležité pořadí deklarace metod. Další důležitou vlastností je, že jazyk C# je case sensitive – to znamená, že rozlišuje mezi velkými a malými písmeny. Identifikátory "hodnota" a "Hodnota" tedy nejsou na rozdíl např. od VB.NET ekvivalentní.

4.4.2 Prvky jazyku

V jazyce C# je realizováno přibližně 80% základních knihoven .NET Framework. I přesto, že je koncipován hlavně pro psaní řízeného kódu, na jehož užití je platforma .NET postavena, lze jej v případě potřeby využít i pro tvorbu kódu neřízeného (bloky unsafe). Použití neřízeného kódu znamená, že běhové prostředí CLR neověřuje, zda je napsaný kód bezpečný (například se neověřuje jinak vyžadovaná typová bezpečnost).

Mezi základní prvky jazyku C# patří:

- třídy (classes) – základní stavební prvek při tvorbě objektově orientovaných aplikací obsahující metody a atributy;
- struktury (structs) – lze je chápat jako zjednodušené třídy, jejich užitím jsou nejčastěji popisovány vlastní datové struktury;
- výčtové typy;
- vlastnosti (properties) – někdy označované jako chytré proměnné;
- pole (arrays) a jejich „chytrá“ verze nazývaná indexery;
- delegáti (delegates) – typově bezpečné ukazatele na funkce;
- události (events) – druh zástupců sloužící ke zpracování asynchronních operací.

S příchodem verze 2.0 vydané koncem roku 2005 pak přibýly především [20]:

- generické typy (generics) – slouží pro vytváření abstraktních (znovupoužitelných) typů;

- částečné třídy (partial classes) – možnost rozdělení třídy do několika souborů;
- statické třídy (static classes) – třídy, které nemohou být inicializovány;
- iterátory (iterators) – nová forma iterátoru poskytující funkčnost generátoru;
- nullovatelné typy (nullable types) – přidávají hodnotu null do množiny povolených hodnot pro jakýkoliv datový typ.

Verze 3.0 vydaná koncem roku 2007 pak přinesla hlavně:

- LINQ (Language INtegrated Query) – integrovaný dotazovací jazyk pro dotazování nad jakýmkoliv daty (např.: LINQ to SQL);
- lambda výrazy (lambda expressions) – umožňují tvořit anonymní metody a použít je v situaci, kdy je očekávána instance delegáta;
- rozšiřující metody (expression methods) – statické metody, které se dají volat jako metody instance.

Nejvýznamnější novinky v aktuálně nejnovější verzi 4.0 jsou [21]:

- dynamicky typované objekty – typ proměnné je určen až za běhu aplikace;
- volitelné parametry a parametry určené jménem;
- vylepšená COM interoperabilita.

4.5 Microsoft Silverlight

Ačkoli je web nejpopulárnějším prostředím pro firemní software, jsou požadavky, které nelze pomocí kombinace HTML, CSS a JavaScriptu jednoduše splnit. Jedná se zejména o animace nebo trojrozměrné grafiky, což je u desktopových aplikací považováno za standard. Překonání těchto omezení je možné pouze použitím zásuvných modulů pro webový prohlížeč. Těmito moduly byly před vydáním technologie Silverlight pouze Java applety nebo Adobe Flash. Kapitola čerpá z [22] a [23].

V dubnu roku 2007 vydala firma Microsoft svou vlastní technologii zásuvného modulu nazvanou Windows Presentation Foundation/Everywhere (později Silverlight) jako přímou konkurenci technologie Adobe Flash (Flex).

4.5.1 Historie technologie

Verze 1.0 zahrnovala pouze funkcionalitu pro 2D kreslení a přehrávání médií. Veškerý kód však ještě bylo nutné psát v JavaScriptu. Druhá verze 1.1 (později

změněná na 2.0) přinesla radikální změnu v podobě zahrnutí CLR, podporu podmnožiny tříd .NET Frameworku a model uživatelského rozhraní, který je založen na WPF. Třetí verze vydaná v červenci 2009 pak přinesla především podporu 3D grafiky, offline Silverlight aplikací a nové ovládací prvky.

4.5.2 Základní charakteristika

Silverlight je cross-platformní. Přímou Microsoft podporuje platformy Windows a Mac OS a nejrozšířenější prohlížeče. Verzi pro Linux potom zajišťuje společnost Novel, jejichž implementace se jmenuje Moonlight.

Vzhledem k tomu, že Silverlight neobsahuje celý .NET Framework, ale pouze jeho část, je velikost instalačního balíčku pouze několik megabajtů (v současnosti necelých 5 MB), na rozdíl od několika desítek v případě celého .NET Framework. Instalace je snadná a rychlá, v principu podobná instalaci Adobe Flash Playeru.

Silverlight je kompatibilní s plným .NET Frameworkem, takže kód napsaný pro Silverlight lze použít v nezměněné podobě například pro aplikaci napsanou s použitím WPF.

4.5.3 Porovnání Microsoft Silverlight a Adobe Flex

Základní princip Silverlightu je v podstatě stejný jako u technologie Adobe Flex – uživatelské rozhraní je definováno ve značkovacím jazyku, o výkonnou část se stará objektově orientovaný programovací jazyk a k běhu je vyžadován plugin v prohlížeči. Aby měl Microsoft šanci uspět v konkurenci Flash Playeru s více než čtyřmi miliardami instalací a tržním podílem větším než 90 %, musel nabídnout nějakou klíčovou výhodu, a tou je již výše zmiňovaná podpora .NET Frameworku, čímž automaticky vzniká několik předností oproti technologii Flex:

- je na výběr z několika jazyků. Vedle očekávaného C# a Visual Basicu lze používat též jazyky dynamické, jako jsou Ruby, Python nebo PHP;
- podpora OOP konceptů je v .NET jazycích na vyšší úrovni než v ActionScriptu (programovací jazyk v prostředích Flex), což je výhodné především u větších projektů;
- serverový i klientský vývoj může probíhat se znalostí jednotného programovacího modelu a jazyka, což je výhodné z pohledu nároků na znalosti vývojářů a potažmo tedy i nákladů. O podobnou integraci

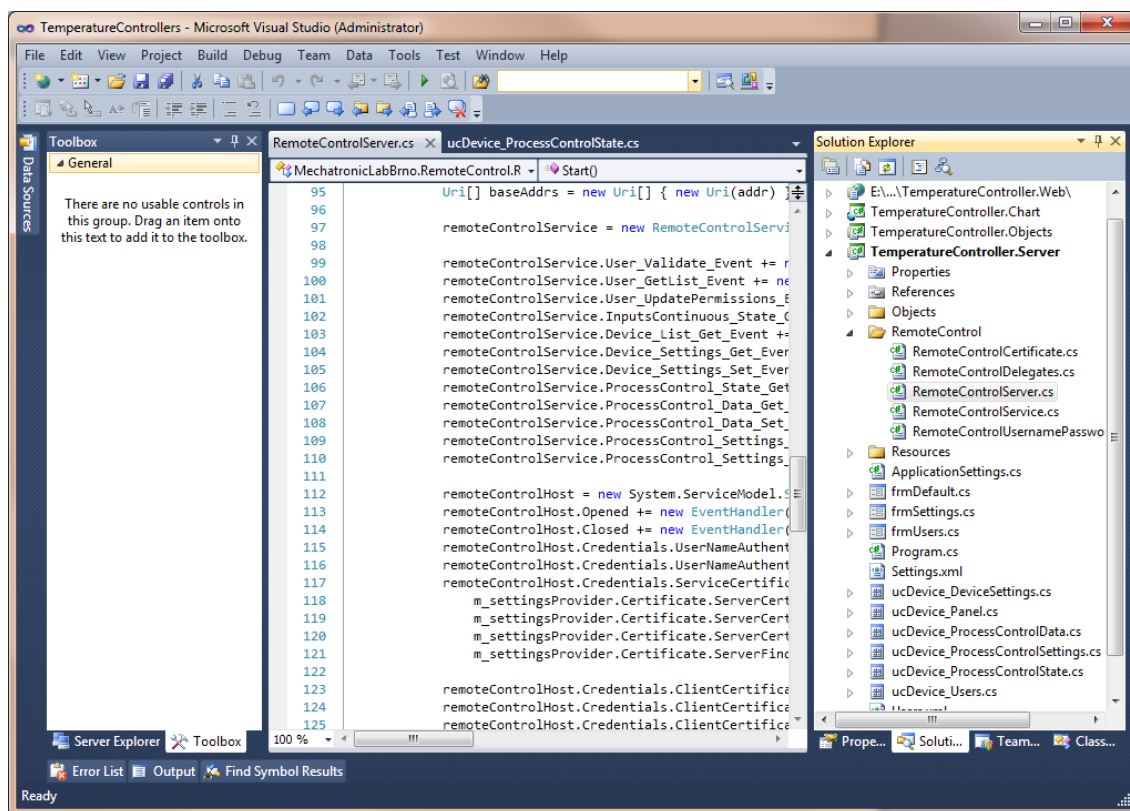
se postupně snaží všechny platformy, .NET je však v tomto ohledu v současnosti nejdále;

- vývojářů .NET je na světě řádově více než Flex vývojářů, což je skutečnost zajímavá především pro manažery, kteří musí pro projekt zajistit lidské zdroje.

4.6 Visual Studio

Microsoft Visual Studio je vývojové prostředí (IDE – Integrated Development Environment) od firmy Microsoft. Je určeno pro vývoj konzolových aplikací, aplikací s grafickým rozhraním spolu s Windows Forms aplikacemi, webovými stránkami, webovými aplikacemi a webovými službami jak ve strojovém kódu, tak ve spravovaném kódu na platformách Microsoft Windows, Windows Mobile, Windows CE, .NET, .NET Compact Framework a Microsoft Silverlight [24].

Visual Studio obsahuje editor kódu podporující IntelliSense (doplňování kódu při jeho psaní) a refaktorování. Integrovaný debugger pracuje jak na úrovni kódu, tak na úrovni stroje. Další vestavěné nástroje zahrnují designer formulářů pro tvorbu GUI aplikací, designer webu, tříd a databázových schémat. Je možné přidávat rozšíření, což vylepšuje funkčnost na téměř každé úrovni – od přidání podpory pro verzovací systémy (jako Subversion a Visual SourceSafe) až po přidání nových nástrojů jako jsou editory a vizuální designery pro jazyky specifické pro obor, nebo nástroje pro další aspekty návrhu programu.



Obr. 12: Vývojové prostředí Microsoft Visual Studio 2010 Professional Edition.

5 KONCEPCE REALIZOVANÉHO ŘEŠENÍ

Cílem práce je vytvořit aplikaci typu klient-server, ve které bude server poskytovat rozhraní pro komunikaci s programovatelnými regulátory k němu připojenými a webový klient bude pomocí rozhraní serveru k těmto regulátorům přistupovat.

Tato koncepce je zvolena z následujících důvodů:

- jednotky a regulátory z cenových důvodů převážně disponují rozhraním TIA/EIA232 (RS232) nebo TIA/EIA 485 (RS485), které neumožňuje komplexní zasílání;
- maximální vzdálenost propojených zařízení je pouze 15m (v případě rozhraní RS232) nebo 1000m (v případě RS485 bez použití opakovačů), takže nelze snadno řešit vzdálené ovládání regulátorů;
- není možno centralizovat správu uživatelů a uživatelských práv při připojení více regulátorů;
- regulátory nemají potřebná zabezpečení pro přímý přístup ze sítě Internet;
- v případě použití modulu pro připojení regulátoru do sítě LAN neúměrně rostou pořizovací náklady.

Cílem práce nebylo vytvoření obecného uživatelského rozhraní umožňujícího připojení libovolného typu regulátoru, ale zprovoznění a zabezpečení vzdáleného ovládání regulátorů. Proto byla implementována pouze podpora ovládání programovatelných regulátorů teploty.

5.1 Architektura systému

Systém používá architekturu klient-server, kde server je reprezentován desktopovou aplikací, běžící na počítači, ke kterému jsou připojeny regulátory, a klient je webová aplikace.

Desktopová aplikace je neustále spuštěna a zajišťuje:

- připojení a komunikaci s regulátory;
- udržování historie stavů regulátorů;
- správu uživatelských účtů a skupin;
- rozhraní pro komunikaci s klienty.

Webová aplikace následně zajišťuje:

- připojení k serveru a komunikaci přes poskytnuté rozhraní;
- přihlašování a komunikaci s uživateli.

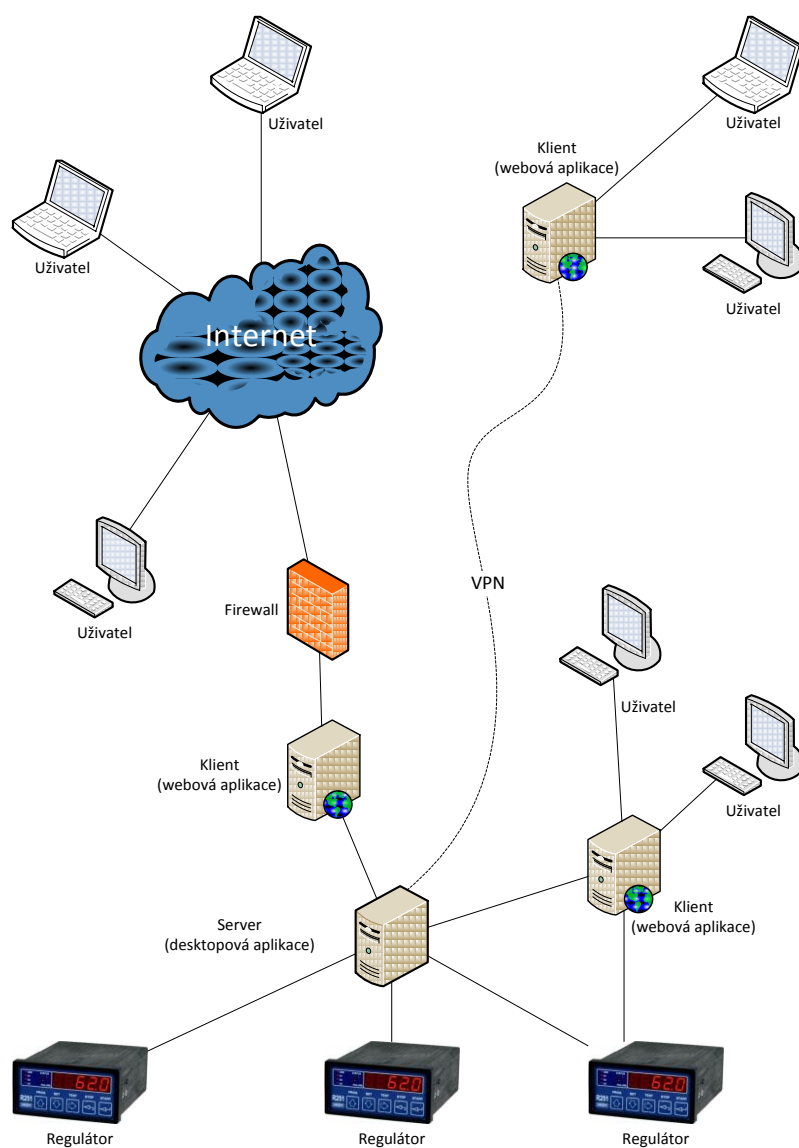
5.2 Přístup a ovládání regulátorů

Uživatel má následující možnosti pro přístup, případně ovládání regulátorů:

- přímo pomocí uživatelského rozhraní desktopové aplikace;
- vzdáleně přes webové rozhraní.

V případě vzdáleného ovládání regulátorů probíhá komunikace následujícím způsobem:

- uživatel odešle požadavek na zobrazení XHTML stránky přes protokol HTTPS webové aplikaci;
- webová aplikace (klient) naváže spojení a odešle požadavek desktopové aplikaci (server) přes HTTPS pomocí technologie WCF (Windows Communication Foundation), která je popsána v kapitole 4.3;
- desktopová aplikace zpracuje, případně zamítne požadavek, a v případě potřeby ho odešle přes sériové rozhraní do regulátoru;
- výsledek zpracování odešle desktopová aplikace zpět (opět pomocí WCF) webové aplikaci, která ho zpracuje a pošle přes HTTPS jako XHTML stránku uživateli.



Obr. 13: Schéma použité architektury.

6 DESKTOPOVÁ APLIKACE

Desktopová aplikace je napsána v programovacím jazyku C# s využitím knihoven platformy Microsoft .NET. Tvoří ji hlavní menu a záložky, jejichž obsah sestává z ovládacích panelů připojených regulátorů, případně ze záznamů komunikace s klienty (záložka *Remote control log*). Jednotlivé ovládací prvky aplikace se dají rozdělit na dva logické celky:

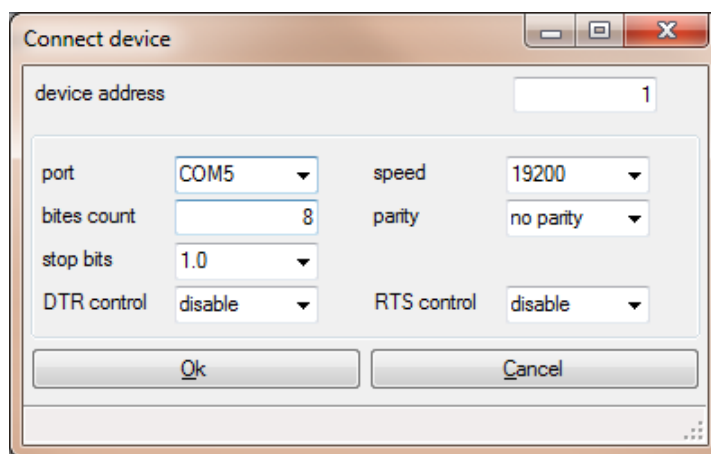
- prvky pro práci s regulátory;
- prvky pro vzdálený přístup.

6.1 Práce s regulátory

Tyto prvky aplikace zajišťují načítání hodnot, jejich nastavování a nastavování připojených regulátorů. Pro komunikaci (obousměrnou) mezi regulátory a aplikací přes sériové rozhraní (porty mohou být i virtuální) jsou použity knihovny *deviceInterface.dll* a *mechLabControls.dll*, jejichž vývoj nebyl součástí této diplomové práce.

6.1.1 Připojení regulátoru

Po připojení k regulátoru (obr. 14) aplikace vypočítá jeho jedinečný identifikátor (hash) z názvu portu a adresy, na kterou je připojen a přidá ho do seznamu aktuálně připojených regulátorů. Identifikátor regulátoru, k jehož výpočtu je použit algoritmus MD5, slouží k jeho jednoznačnému určení při požadavku na vzdálené ovládání.

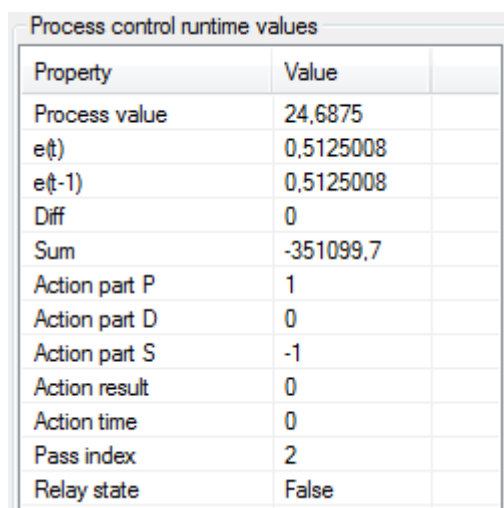
The image shows a Windows-style dialog box titled "Connect device". It has a standard title bar with minimize, maximize, and close buttons. The dialog contains several input fields and dropdown menus. At the top, there is a "device address" field with the value "1". Below this, there are two columns of settings. The left column includes "port" (set to "COM5"), "bites count" (set to "8"), "stop bits" (set to "1.0"), and "DTR control" (set to "disable"). The right column includes "speed" (set to "19200"), "parity" (set to "no parity"), and "RTS control" (set to "disable"). All dropdown menus have a small downward arrow. At the bottom of the dialog, there are two buttons: "Ok" and "Cancel".

Obr. 14: Formulář pro připojení k regulátoru.

Dále pak načte konfiguraci zařízení, stav a nastavení regulace a zobrazí je v ovládacím panelu regulátoru.

6.1.2 Stav regulace

Aplikace periodicky načítá stav regulace, jednotlivé veličiny zobrazuje v tabulce (obr. 15) a ukládá je do paměti. Díky tomuto mechanismu uchovávání historie stavů nemusí aplikace přistupovat přímo k regulátoru při požadavku z klienta, což je výhodné zejména kvůli nízké přenosové rychlosti sériového rozhraní mezi aplikací a regulátorem. Vzdáleným uživatelům se tedy vždy zobrazí hodnoty, které má desktopová aplikace v paměti.



Property	Value
Process value	24,6875
$e(t)$	0,5125008
$e(t-1)$	0,5125008
Diff	0
Sum	-351099,7
Action part P	1
Action part D	0
Action part S	-1
Action result	0
Action time	0
Pass index	2
Relay state	False

Obr. 15: Zobrazení stavu regulátoru v desktopové aplikaci.

6.1.3 Požadovaná hodnota

Obdobně jako hodnoty stavu regulátoru, uchovává desktopová aplikace i historii předchozích hodnot požadované hodnoty. Její nastavení je dostupné prostřednictvím ovládacího prvku zobrazeného na obrázku 16.

Obr. 16: Nastavení požadované hodnoty.

6.1.4 Nastavení regulačních veličin a konfigurace zařízení

Uživatelské rozhraní ovládacího panelu připojeného regulátoru nabízí i možnost pro nastavení regulačních veličin a konfiguraci zařízení (obr. 17).

Obr. 17: Nastavení regulačních veličin (vlevo) a konfigurace zařízení (vpravo) v desktopové aplikaci.

I tato nastavení drží desktopová aplikace v paměti a mění je pouze v případě požadavku na změnu a úspěšném nastavení v regulátoru.

6.2 Vzdálené ovládání regulátorů a jeho zabezpečení

Po zapnutí vzdáleného ovládání regulátorů si aplikace načte nastavený serverový certifikát ze skladiště certifikátů systému Windows (Windows Certificate

Store) (nastavení viz. 6.2.1), spustí hostování WCF služby na všech nastavených URI (Uniform Resource Identifier) a čeká na požadavky od klientů (webových aplikací).

6.2.1 Nastavení vzdáleného ovládání

Nastavení serverové části vzdáleného ovládání regulátorů jsou dostupná přes formulář zobrazený na obrázku 18. Aplikace všechna tato nastavení ukládá do souboru Settings.xml, který je v kořenovém adresáři aplikace.

Remote control settings

Endpoints

☒	https://127.0.0.1:8080/
☒	https://controllers.vpn.mycompany.com/
☐	
☐	
☐	

Add Delete

Server certificate

Store location: LocalMachine Store name: My Find by type: FindByThumbprint

Find by value: ad ae 40 80 06 bf 88 78 9a 89 67 3d 06 46 02 80 58 a5 5f 99

Test

Clients certificates

Store location: CurrentUser Validation mode: PeerTrust

Validation mode help: PeerTrust * The certificate is valid if it is in the trusted people store. ChainTrust

☐ Start remote control on start-up

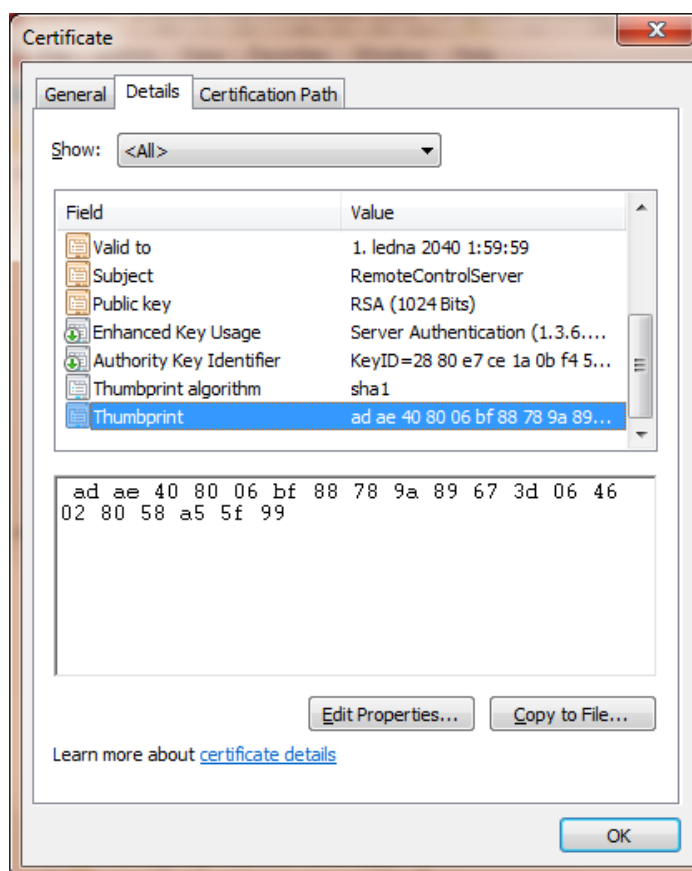
OK Cancel

Obr. 18: Nastavení vzdáleného ovládání v desktopové aplikaci.

V horní části formuláře (sekce *Endpoints*) se nastavují koncové body, na kterých bude hostována WCF služba, pomocí které komunikuje server s klienty. Aplikace požaduje použití protokolu HTTPS, což je dáno druhem použitého způsobu komunikace (Binding) a zabezpečení WCF služby.

V sekci *Server certificate* se nastavuje umístění a druh vyhledání certifikátu, který aplikace použije pro zabezpečení zpráv s klienty. Návod na vytvoření a instalaci tohoto certifikátu je v kapitole 8.1.1. Je doporučeno použít druh vyhledávání podle otisku certifikátu (*FindByThumbprint*), protože tato hodnota je vždy pro každý certifikát

jedinečná. Otisk certifikátu lze zjistit v nástroji MMC jako jednu z položek v jeho detailu (obr. 19). Nastavení a dostupnost certifikátu je možno otestovat tlačítkem *Test*.



Obr. 19: Zjištění otisku certifikátu v nástroji MMC.

V sekci *Client certificates* se určuje, kde musí být uložen klientský certifikát klienta (webového serveru), aby ho aplikace autentizovala. Je nutné nastavit úložiště a mód validace certifikátu. Módy validace certifikátu jsou:

- *PeerTrust* – veřejná část klientského certifikátu klienta musí být v seznamu důvěryhodných osob (Trusted People);
- *ChainTrust* – certifikát je validní, pokud je certifikační autorita, která ho vydala v seznamu důvěryhodných kořenových certifikačních autorit (Trusted Root Certification Authorities);
- *PeerOrChainTrust* – součet předchozích.

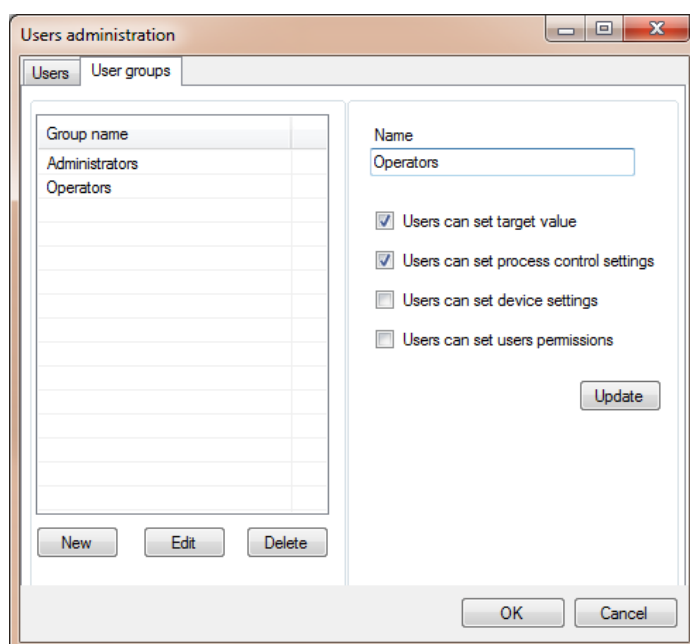
6.2.2 Správa vzdálených uživatelů

Aplikace ukládá všechna data o vzdálených uživateli a uživatelských skupinách do souboru Users.xml, který je stejně jako soubor s nastaveními vzdáleného

ovládání umístěn v kořenovém adresáři aplikace. Samotná správa uživatelů a skupin je dostupná přes položku hlavního menu *Remote control* → *Users*.

Na první záložce (*Users*) je seznam všech uživatelů s možnostmi jejich editace a mazání. Na druhé záložce (*User roles*), která je zobrazena na obrázku 20 je seznam uživatelských skupin. Při jejich úpravě (vytváření) je možno nastavit uživatelům, kteří do dané skupiny patří, následující práva:

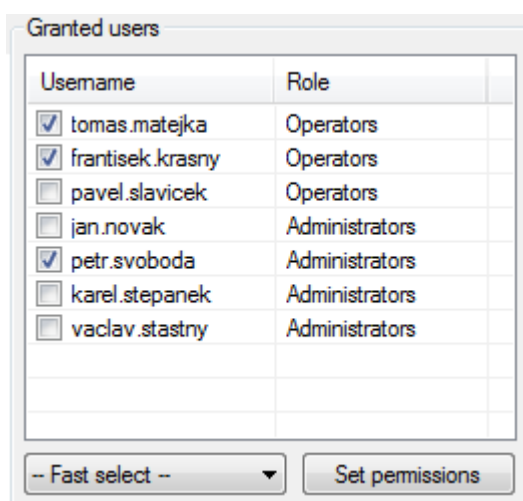
- *Users can set target value* – uživatelé mohou měnit žádanou hodnotu;
- *Users can set process control settings* – uživatelé mohou měnit regulační veličiny;
- *Users can set device settings* – uživatelé mohou měnit nastavení zařízení;
- *Users can set users permissions* – uživatelé mohou nastavovat, kteří uživatelé mají oprávnění pracovat s daným regulátorem.



Obr. 20: Nastavení uživatelských skupin v desktopové aplikaci.

Oprávnění práce s regulátorem

V ovládacím panelu každého připojeného regulátoru je sekce *Granted users* (obr. 21), ve které se provádí nastavení oprávnění práce s regulátorem jednotlivým uživatelům. Takže pouze vybraní uživatelé mají oprávnění vzdáleně pracovat s regulátorem. To, jaké činnosti mohou provádět, je závislé na uživatelské skupině, do které patří.



Obr. 21: Nastavení oprávnění práce s regulátorem.

6.2.3 Komunikace s klienty

Pro komunikaci s klienty je použita WCF služba, jejíž instance je vytvořena při prvním požadavku od klienta a je využívána pro všechny následující požadavky. Chová se tedy podle návrhového vzoru Singleton.

Při příchozím požadavku je klient nejprve autentizován (pomocí předaného klientského certifikátu). Pokud je autentizace úspěšná, tak server přechází k autorizaci uživatele. V opačném případě ukončí spojení.

Autorizace uživatele k požadované činnosti se provádí na základě těchto hodnot:

- uživatelské jméno a kontrolní součet hesla – jsou posílány v hlavičce SOAP zprávy od klienta;
- identifikátor regulátoru – pokud požadovaná činnost není požadavek na přihlášení nebo na vrácení seznamu připojených regulátorů;
- uživatelská skupina, do které uživatel patří – každá uživatelská skupina má různá práva pro práci s připojenými regulátory.

V případě úspěšné autorizace je vykonána požadovaná činnost (například nastavení požadované hodnoty) a je vrácen její výsledek klientovi a dále pak uživateli jako XHTML kód. V opačném případě je vyvolána výjimka, kterou klient odchytne a zobrazí chybovou hlášku uživateli.

7 WEBOVÁ APLIKACE

Webová aplikace je, stejně jako desktopová, napsána v programovacím jazyku C#. Pro její tvorbu bylo využito knihoven ASP.NET, které jsou součástí Microsoft .NET Frameworku.

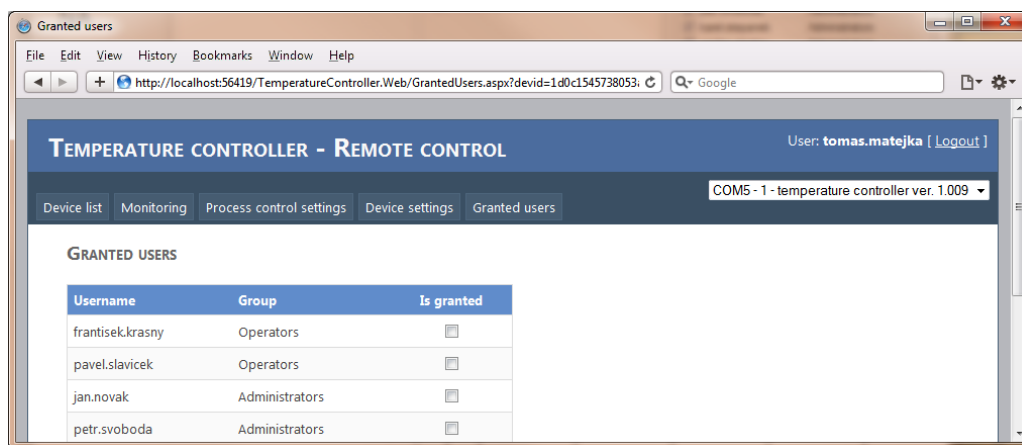
7.1 Přihlášení uživatele

Všechny stránky webové aplikace (kromě přihlašovací stránky) jsou dostupné pouze přihlášeným uživatelům. Pokud tedy uživatel napíše URL adresu do prohlížeče přímo a není přihlášen, je přesměrován na stránku s přihlašovacím formulářem. Po úspěšné autorizaci uživatele (postup autorizace je popsán v kapitole 6.2.3), je uživatel přesměrován na stránku se seznamem všech připojených regulátorů. Zobrazují se mu pouze ty regulátory, s nimiž má oprávnění pracovat.

Pro udržování stavu přihlášeného uživatele je použit mechanismus udržování stavu relace (sessions). Jako úložiště pro stav relace byl zvolen State Server. Výhodou tohoto řešení oproti ukládání stavu relace do ASP.NET procesu (InProc) je, že nedojde k jeho ztrátě ani v případě restartu hostujícího procesu.

7.2 Práce s regulátory

V menu webové aplikace jsou vždy dostupné pouze činnosti, které může uživatel s daným regulátorem provádět, což souvisí s uživatelskou skupinou, do které uživatel patří. Na obrázku 22 je zobrazeno menu přihlášeného uživatele s plným oprávněním pro práci s regulátorem.



Obr. 22: Ukázka menu webové aplikace.

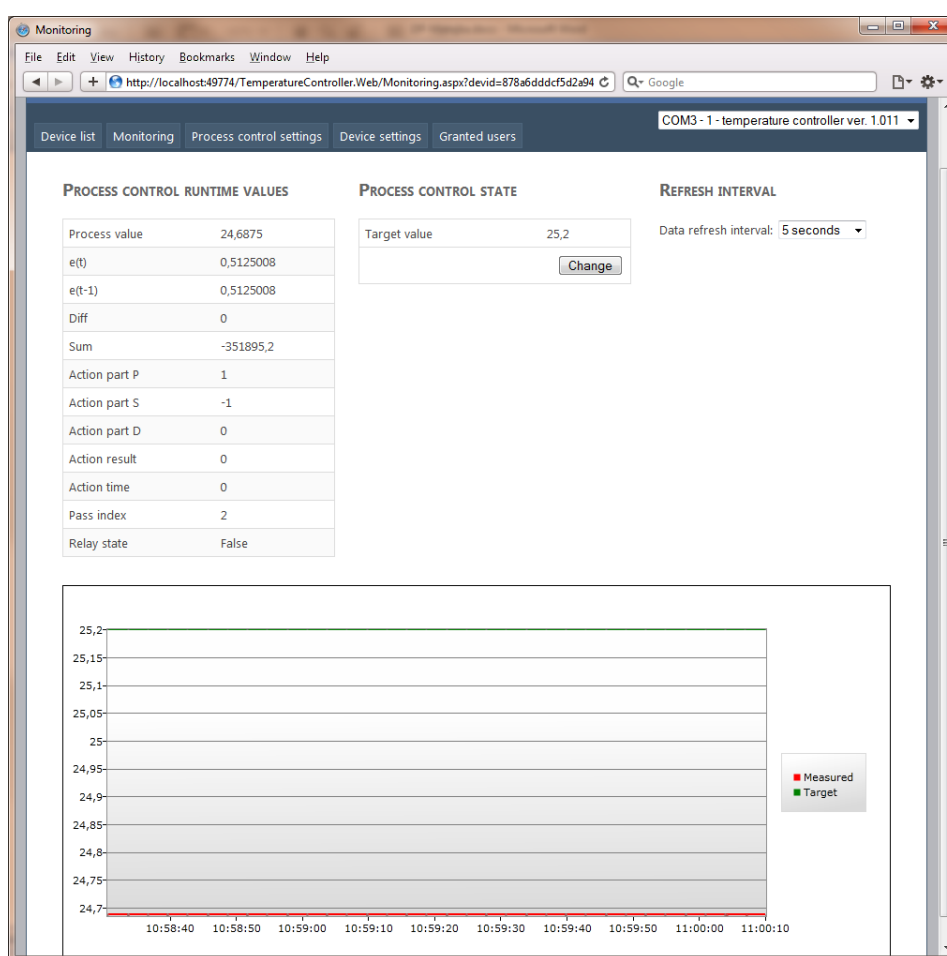
Uživatelské rozhraní webové aplikace nabízí stejné možnosti pro práci s regulátory jako desktopová aplikace (popsáno v kapitole 6.1).

7.2.1 Monitoring

Položka *Monitoring* v menu vede na stránku, na které jsou zobrazeny hodnoty stavu regulace a žádaná hodnota. Nachází se zde i graf zobrazující závislost aktuální a žádané hodnoty na čase (obr. 23). Tento graf byl vytvořen za použití technologie Microsoft Silverlight.

Celá stránka využívá asynchronního stahování dat (AJAX – Asynchronous Javascript And Xml) na pozadí. Tato technologie byla použita z těchto důvodů:

- snížení počtu přenášených dat mezi uživatelem a webovým serverem, protože se nemusí vždy stahovat celý obsah stránky, ale pouze její části;
- snížení zátěže desktopové aplikace (serveru). Při prvním načtení stránky se načtou hodnoty z historie předchozích stavů (pro zobrazení v grafu), při každém dalším (interval načítání se dá nastavit – položka *Data refresh interval*) se načtou už jen hodnoty od posledního načtení;
- stránka se nemusí v prohlížeči obnovovat při každé aktualizaci hodnot.



Obr. 23: Monitoring regulace ve webové aplikaci.

7.3 Propojení se serverem

Pro připojení a komunikaci se serverem používá webová aplikace WCF proxy, které bylo vytvořeno pomocí konzolového nástroje *svcutil.exe*. Tento nástroj vytvoří, podle metadat WCF služby, třídu, která umožňuje komunikaci s WCF hostem (serverem).

7.3.1 Nastavení komunikace

Každá webová aplikace napsaná v ASP.NET má svůj konfigurační soubor, který se jmenuje *Web.config* a je umístěn v kořenovém adresáři aplikace. V tomto souboru, který je ve formátu XML, se nachází i nastavení připojení k serveru (sekce *system.serviceModel*).

Nastavení koncového bodu

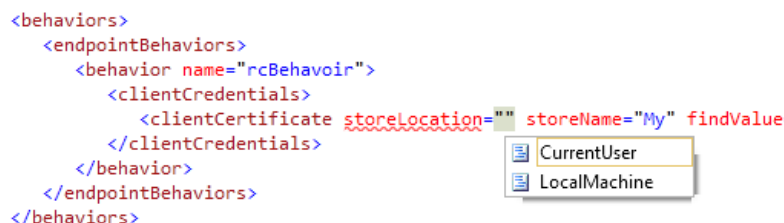
Koncový bod nastavený v souboru Web.config musí odpovídat některému z koncových bodů nastavených v serverové aplikaci (postup nastavení je popsán v kapitole 6.2.1). Nastavení se provádí v sekci system.serviceModel/client/endpoint – atribut address.

Nastavení certifikátu

Pro nastavení klientského certifikátu webové aplikace se využívá sekce system.serviceModel/behaviors/endpointBehaviors/behavior/clientCertificate, která obsahuje tyto atributy:

- storeLocation – umístění skladiště certifikátů systému Windows;
- storeName – jméno skladiště certifikátů systému Windows;
- x509FindType – druh vyhledání certifikátu;
- findValue – hodnota pro vyhledání certifikátu.

V případě použití vývojového prostředí Microsoft Visual Studio pro nastavení těchto atributů, jsou nástrojem IntelliSense nabízeny jejich možné hodnoty (obr. 24).



```

<behaviors>
  <endpointBehaviors>
    <behavior name="rcBehavoir">
      <clientCredentials>
        <clientCertificate storeLocation="" storeName="My" findValue="" />
      </clientCredentials>
    </behavior>
  </endpointBehaviors>
</behaviors>

```

The image shows a code editor with XML code for configuring a client certificate. The code is as follows:

```

<behaviors>
  <endpointBehaviors>
    <behavior name="rcBehavoir">
      <clientCredentials>
        <clientCertificate storeLocation="" storeName="My" findValue="" />
      </clientCredentials>
    </behavior>
  </endpointBehaviors>
</behaviors>

```

A dropdown menu is visible next to the empty string value for the storeLocation attribute, showing two options: "CurrentUser" and "LocalMachine".

Obr. 24: Podpora nástroje IntelliSense při úpravě souboru Web.config.

8 INSTALACE

Desktopová i webová aplikace byly navrženy tak, aby se eliminovalo odposlouchávání předávaných dat třetí stranou, a to jak při komunikaci mezi klientem a serverem tak i mezi uživatelem a klientem. Je tedy nutné použít zabezpečenou komunikaci, která je realizována pomocí protokolu HTTPS, který je nadstavbou standardního HTTP a využívá asymetrického šifrování přenášených dat.

8.1 Certifikáty

K tomu, aby komunikace mezi klientem a serverem fungovala, je nutné mít nainstalované tyto certifikáty:

- serverový certifikát s privátním klíčem na počítači, kde běží desktopová aplikace;
- klientský certifikát s privátním klíčem na počítači, který hostuje webovou aplikaci;
- veřejnou část klientského certifikátu na počítači, kde běží desktopová aplikace;

Je možné použít i certifikáty podepsané vlastní certifikační autoritou (tzv. self-signed). Tyto se však doporučuje používat pouze při testování aplikace. Místo nich je doporučeno používat certifikáty vydané Windows Server 2008(2003) certifikačním serverem nebo důvěryhodnými certifikačními autoritami, jako je např.: „První certifikační autorita, a.s.“.

8.1.1 Tvorba self-signed certifikátů

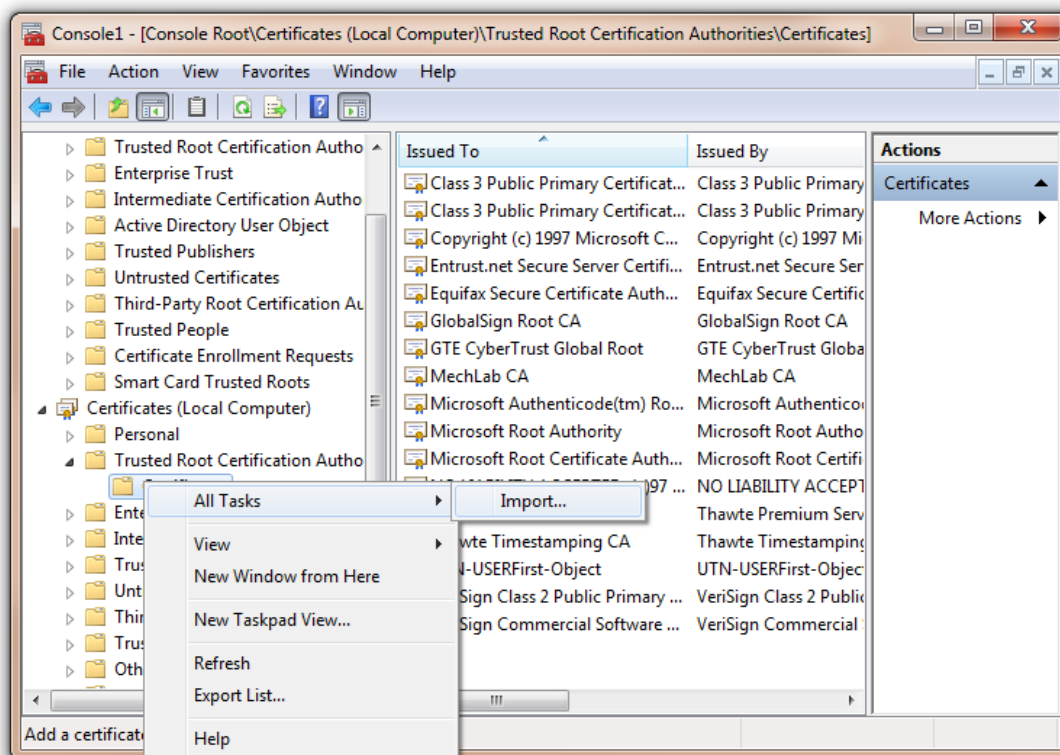
Následující body popisují vytvoření výše popsaných certifikátů pomocí příkazové řádky a nástroje Microsoft Management Console.

- Spuštění příkazové řádky v prostředí Visual Studia:

```
%comspec% /k "%ProgramFiles(x86)%\Microsoft Visual Studio
10.0\VC\vcvarsall.bat" x86
```
- Vytvoření vlastní certifikační autority:

```
makecert -n "CN=MechLab CA" -r -sv c:\MechLabCA.pvk
c:\MechLabCA.cer
```

- Import certifikátu certifikační autority do důvěrhodných kořenových certifikačních autorit (Trusted Root Certification Authorities) pomocí nástroje MMC (Microsoft Management Console) (obr. 25).



Obr. 25: Import certifikátu certifikační autority v nástroji MMC.

- Vytvoření a podepsání serverového certifikátu vlastní certifikační autoritou:

```
makecert -eku "1.3.6.1.5.5.7.3.1" -sk RemoteControlServer -iv c:\MechLabCA.pvk -n "CN=RemoteControlServer" -ic c:\MechLabCA.cer -sr localmachine -ss my -sky exchange -pe
```

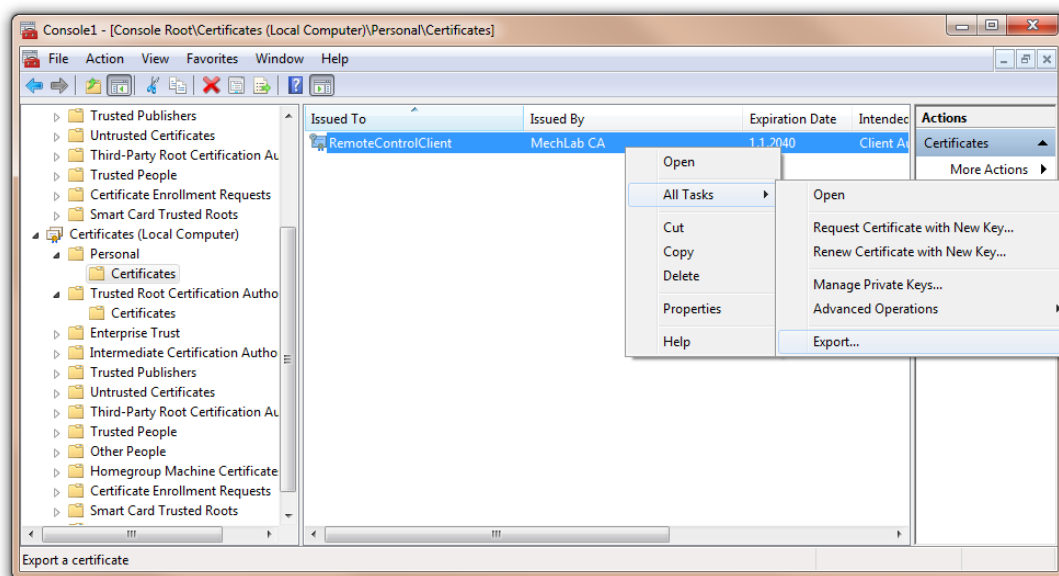
- Povolení certifikátu pro použití na určitém portu:

```
netsh http add sslcert ipport=127.0.0.1:6666  
certhash=94cd770beab056b2cbb99508e96fxxxxxxxxx appid={00112233-4455-6677-8899-AABBCCDDEEFF}
```

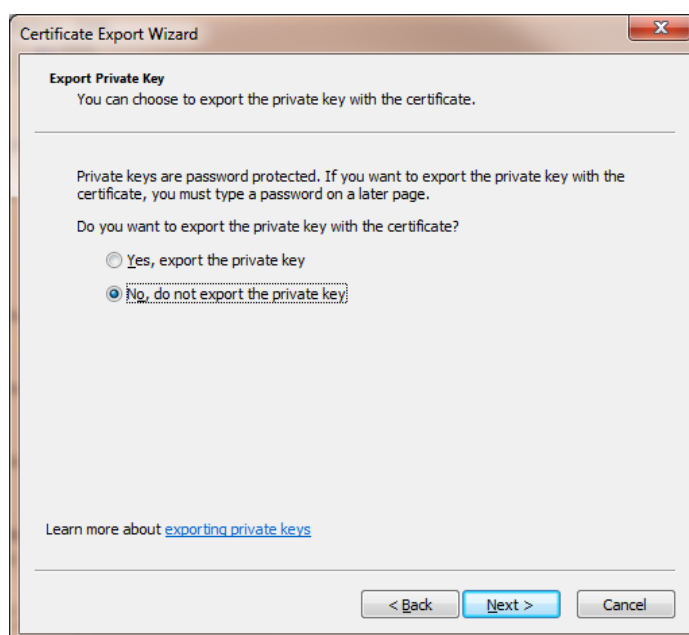
Parametr certhash musí odpovídat otisku certifikátu (zjištění otisku certifikátu v nástroji MMC je zobrazeno na obrázku 19).

- Vytvoření a podepsání klientského certifikátu certifikační autoritou:

```
makecert -eku "1.3.6.1.5.5.7.3.2" -sk RemoteControlClient -iv c:\MechLabCA.pvk -n "CN=RemoteControlClient" -ic c:\MechLabCA.cer -sr localmachine -ss my -sky exchange -pe
```
- Export veřejné části klientského certifikátu (obr. 26 a 27):

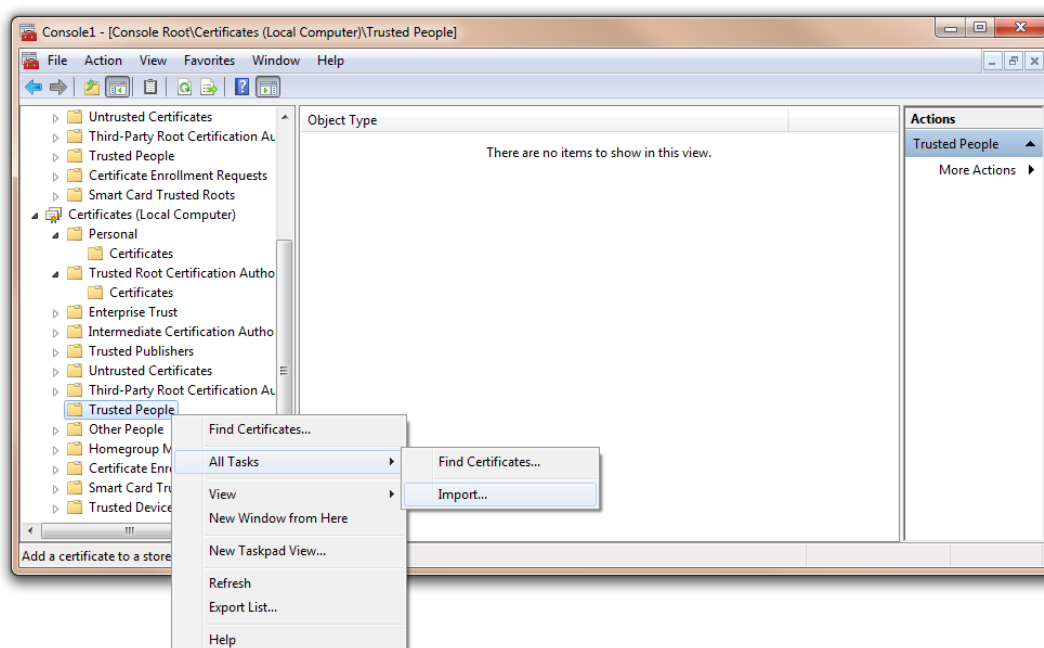


Obr. 26: Export certifikátu v nástroji MMC – krok 1



Obr. 27: Export certifikátu v nástroji MMC – krok 2

- Import veřejné části klientského certifikátu na počítači s desktopovou aplikací pomocí nástroje MMC mezi důvěryhodné osoby je zobrazen na obrázku 28.



Obr. 28: Import certifikátu v nástroji MMC.

8.2 Instalace desktopové aplikace

Je nutné, aby na počítači, na kterém poběží desktopová aplikace, byl nainstalován Microsoft .NET Framework verze 4.0 (viz. příložené CD). Samotná aplikace nevyžaduje žádnou speciální instalaci, stačí ji pouze zkopírovat.

8.3 Instalace webové aplikace

Jak již bylo popsáno výše, komunikace mezi uživatelem a webovým serverem probíhá pomocí protokolu HTTPS, který vyžaduje nainstalovaný serverový certifikát s privátním klíčem na počítači, na kterém je webová aplikace hostována. Hostování zajišťuje Internetová informační služba (IIS – Internet Information Services) systému Windows.

8.3.1 Serverový certifikát webové aplikace

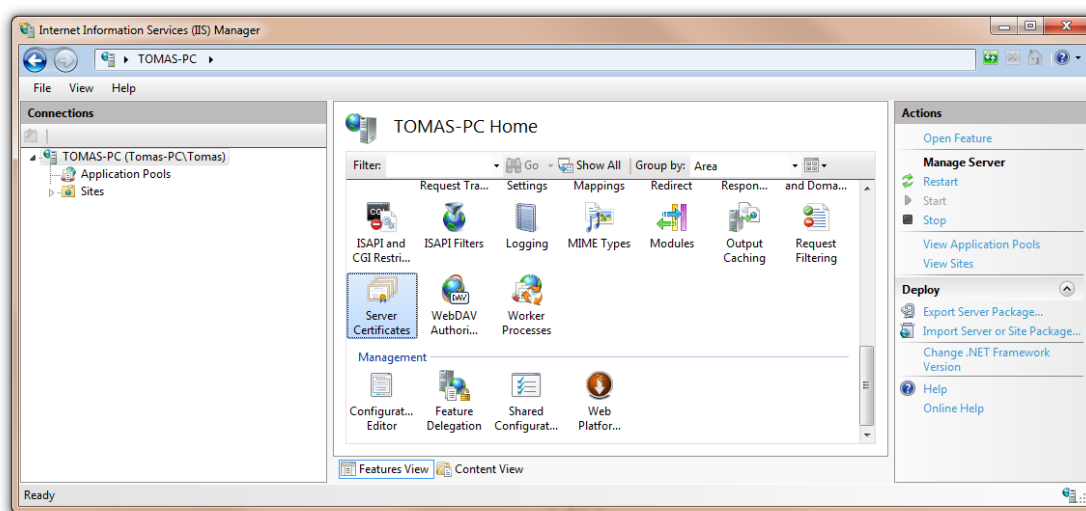
Tento certifikát lze vytvořit obdobným způsobem, jako v případě desktopové aplikace, tj. pomocí příkazu:

```
makecert -eku "1.3.6.1.5.5.7.3.1" -sk RemoteControlWebServer -iv
c:\MechLabCA.pvk -n "CN=RemoteControlWebServer " -ic c:\MechLabCA.cer
-sr localmachine -ss my -sky exchange -pe
```

Dále se musí certifikát vyexportovat (obr. 26), avšak tentokrát se musí exportovat včetně privátního klíče (na rozdíl od obr. 27). Vytvořený certifikát je pak nutné importovat do Internetové informační služby.

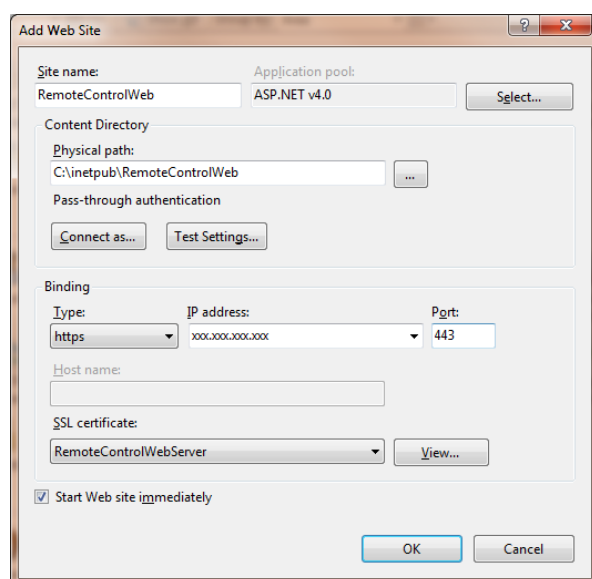
8.3.2 Hostování webové aplikace na IIS

Prvním krokem je nainstalování výše popsaného serverového certifikátu, což je zobrazeno na obrázku 29.



Obr. 29: Instalace serverového certifikátu v IIS Manageru.

Druhým krokem je přidání nové webové stránky. V sekci *Binding* musí být vybrán protokol HTTPS, IP adresa se kterou bude stránka svázána a port (standardní port pro HTTPS protokol je 443). Dále pak musí být vybrán dříve nainstalovaný certifikát. Všechna tato nastavení jsou vidět na obr. 30.

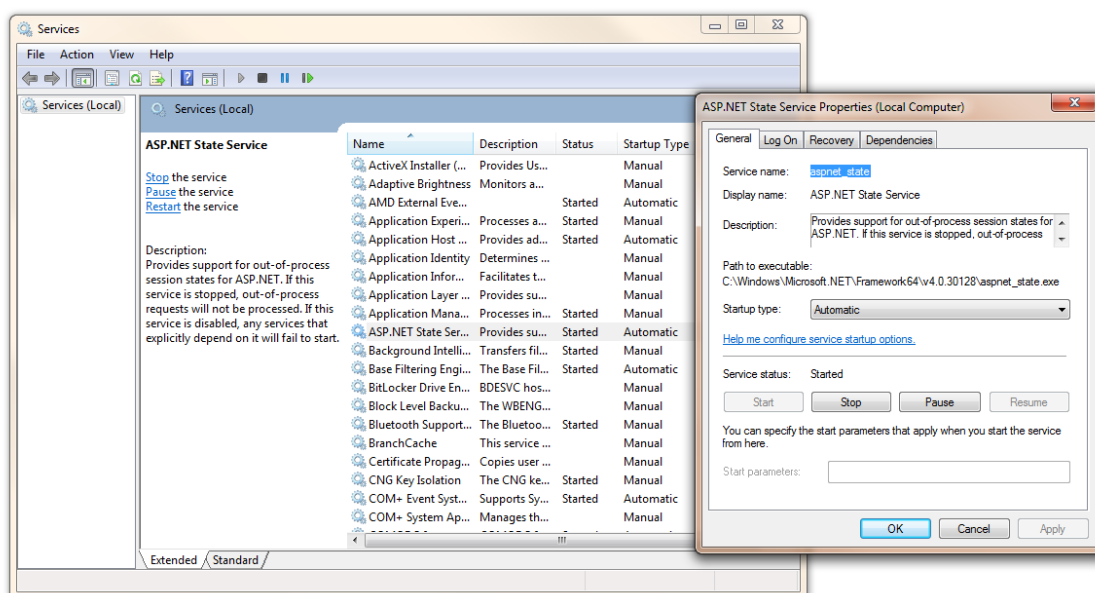


Obr. 30: Přidání webové aplikace v IIS Manageru.

8.3.3 Nastavení udržování stavu relace

Použitá konfigurace udržování stavu relace (sessions) vyžaduje následující nastavení na webovém serveru:

- záznam registru musí být nastaven na hodnotu 1:
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\aspnet_state\Parameters\AllowRemoteConnection
- musí běžet služba ASP.NET State Service (obr. 31), která relace spravuje.



Obr. 31: Spuštění služby ASP.NET State Service.

9 ZHODNOCENÍ VLASTNOSTÍ REALIZOVANÉHO ŘEŠENÍ

Hlavní výhodou realizovaného řešení je vysoká míra zabezpečení vzdáleného ovládání regulátorů, čehož je dosaženo použitím certifikátů k šifrování posílaných dat mezi uživatelem a klientem i mezi klientem a serverem.

Dalšími výhodami použitého řešení jsou:

- odstranění omezení plynoucích z použití sériového rozhraní pro komunikaci s regulátory. Jedná se zejména o maximální možnou vzdálenost propojených zařízení přes toto rozhraní;
- centralizování správy uživatelů a jejich oprávnění;
- k serveru může být připojeno více klientů (webových aplikací), což umožňuje použití pokročilých síťových architektur, jako například virtuální privátní sítě (VPN – Virtual Private Network);
- počítač, ke kterému jsou připojeny regulátory, nemusí být zároveň webový server a nemusí tedy být ani v případě požadavku ovládání regulátorů přes síť Internet, k této síti přímo připojen, což je výhodné zejména kvůli bezpečnosti.

Nevýhodami pak jsou:

- relativně složitá instalace z důvodu použití certifikátů;
- možnost připojení webové aplikace pouze k jednomu serveru. Toto omezení se dá obejít vytvořením více webových stránek s různým nastavením koncových bodů na hostující Internetové informační službě (IIS – Internet Information Services).

10 ZÁVĚR

Diplomová práce se zabývá tvorbou softwaru pro zpřístupnění vzdáleného ovládání programovatelných regulátorů přes webové rozhraní.

V rešeršní části práce byly popsány struktury systémů řízení technologických procesů. Dále pak jednotlivé typy regulátorů včetně programovatelných regulátorů, softwarové prostředky a technologie, které byly použity při realizaci. Praktická část se zabývá popisem použité architektury, popisem vytvořeného software (desktopové a webové aplikace) a vytvořením a nastavením certifikátů, jejichž použití je nutné z důvodu zachování vysoké míry zabezpečení komunikace.

Díky diplomové práci jsem se seznámil zejména s vývojem aplikací využívajících servisně orientovanou architekturu (SOA – Service Oriented Architecture) pomocí technologie Microsoft WPF (Windows Presentation Foundation), která umožňuje výměnu informací mezi aplikacemi a není závislá na použité platformě ani operačním systému. Dále jsem pak rozšířil svoje znalosti o tvorbu webových aplikací za použití technologie ASP.NET a základy tvorby interaktivních webových aplikací s použitím technologie Microsoft Silverlight.

Vytvořené aplikace implementují podporu jednoduchých programovatelných regulátorů teploty. Protože realizovaná aplikace splňuje požadované parametry jak z pohledu bezpečnosti, tak z pohledu rychlosti a komfortu ovládání, je možné přistoupit k jejímu rozšíření o další typy programovatelných regulátorů a řídicích jednotek. Dále by bylo vhodné se zabývat efektivitou tvorby logů o činnosti regulátorů i o činnosti jednotlivých uživatelů, a to jak z pohledu spolehlivosti, tak i z pohledu bezpečnosti.

SEZNAM POUŽITÉ LITERATURY

- [1] KMÍNEK, Miloš. *Účel a funkce počítačových řídicích systémů* [online]. 2005, [cit. 2010-02-13]. Dostupné z:
<<http://uprt.vscht.cz/kminekm/mrt/F6/F6k61-ucfu.htm>>.
- [2] KMÍNEK, Miloš. *Struktura počítačových řídicích systémů* [online]. 2005, [cit. 2010-02-13]. Dostupné z:
<<http://uprt.vscht.cz/kminekm/mrt/F6/F6k63-stru.htm>>.
- [3] KMÍNEK, Miloš. *Technické vybavení počítačových řídicích systémů* [online]. 2005, [cit. 2010-02-13]. Dostupné z:
<<http://uprt.vscht.cz/kminekm/mrt/F6/F6k64-tech.htm>>.
- [4] STANLEY, Paul. *Advantages of Web Applications* [online]. 2010, [cit. 2010-03-15]. Dostupné z: <<http://www.pssuk.com/AdvantagesWebApplications.htm>>
- [5] KABEŠ, Karel. Průmyslové kompaktní regulátory – přehled trhu. *Automatizace* [online]. 2006, únor [cit. 2010-04-18]. Dostupný na WWW:
<http://www.automatizace.cz/article.php?a=1079>.
- [6] HLAVA, Jaroslav. *Prostředky automatického řízení II: analogové a číslicové regulátory, elektické pohony, průmyslové komunikační systémy*. 1. vyd. Praha: ČVUT, 2000. 160 s. ISBN 80-01-02221-8.
- [7] HONEYWELL. *DCP551 Mark II – Digital Control Programmer – User's Manual*. 2009, červenec [cit. 2010-04-29]. Dostupné z:
<<http://hpsweb.honeywell.com/NR/rdonlyres/AE1A88F4-1371-44AE-8FD1-CA7925B8C80C/83724/HNLDCP551CPUM5024E13.pdf>>.
- [8] PUŠ, Petr. *Poznáváme C# a Microsoft .NET – 1.díl* [online]. 2004, listopad [cit. 2010-04-30]. Dostupné na WWW: <http://www.zive.cz/clanky/poznavame-c-a-microsoft-net--1dil/sc-3-a-120978/default.aspx>. ISSN 1212-8554.
- [9] SHARP, John; JAGGER, Jon. *Visual C# .NET krok za krokem*. GREGOR, Jan – překlad. 1. vydání. Praha: Mobil Media. 2002.
- [10] TROELSEN, Andrew. *C# a .NET 2.0 profesionálně*. KRISTIÁN, Pavel Jan – překlad. 3. edice. Brno: ZONER software s.r.o., 2006. ISBN 80-86815-42-0.
- [11] KAČMÁŘ, Dalibor. *Programujeme .NET aplikace ve Visual Studiu .NET*. 1. vydání. Praha: Computer Press. 2001. 366 s. ISBN 80-7226-569-5.

- [12] Wikipedie. *.NET* [online]. 2010-04-21 [cit. 2010-05-03]. Dostupné z:
<<http://cs.wikipedia.org/wiki/.NET>>.
- [13] Wikipedie. *ASP.NET* [online]. 2010-04-22 [cit. 2010-05-05]. Dostupné z:
<<http://cs.wikipedia.org/wiki/ASP.NET>>.
- [14] w3school.com, *ASP.NET vs. ASP* [online]. [cit. 2010-04-13]. Dostupné z:
<http://www.w3schools.com/aspnet/aspnet_vsasp.asp>.
- [15] LARYŠ, Kryštof. *WCF pro začátečníky – 1. díl: teorie, základní pojmy* [online]. 03.02.2009 [cit. 2010-04-19]. Dostupné z:
<<http://www.netstudent.cz/Articles/wcf-pro-zacatecniky-1-dil-teorie-zakladni-pojmy>>.
- [16] KOŠŤÁL, Marián. *WCF - Základné pojmy* [online]. 2007-02-08 [cit. 2010-03-24]. Dostupné z: <<http://www.vyvojar.cz/Articles/454-wcf-zakladne-pojmy.aspx>>.
- [17] Wikipedie. *C#* [online]. 2010-04-02 [cit. 2010-04-11]. Dostupné z:
<http://cs.wikipedia.org/wiki/C_Sharp>.
- [18] PUŠ, Petr. *Poznáváme C# a Microsoft .NET – 2.díl* [online]. 2004, prosinec [cit. 2010-04-20]. Dostupné na WWW:
<http://www.zive.cz/clanky/poznavame-c-a-microsoft-net--2-dil/sc-3-a-121246/default.aspx>. ISSN 1212-8554.
- [19] Standard ECMA-334 [online]. Dostupné z:
<<http://www.ecma-international.org/publications/files/ECMA-ST/Ecma-334.pdf>>.
- [20] Microsoft Corporation. *What's New in the C# 2.0 Language and Compiler* [online]. [cit. 2010-03-29]. Dostupné z:
<<http://msdn.microsoft.com/en-us/library/7cz8t42e%28VS.80%29.aspx>>.
- [21] AUGUSTÝN, Michal. *Novinky v C# 4.0 podle CTP* [online]. 2008-10-30 [cit. 2010-04-01]. Dostupné z:
<<http://www.augi.cz/programovani/novinky-v-c-40-podle-ctp/>>.
- [22] BERNARD, Borek. *Rich Internet Applications v roce 2008* [online]. 2008-04-25 [cit. 2010-03-26]. Dostupné z:
<<http://interval.cz/clanky/rich-internet-applications-v-roce-2008/>>.
ISSN 1212-0901.

- [23] MACDONALD, Matthew; SZPUSZTA, Mario. *ASP.NET 3.5 a C# 2008*. POKORNÝ, Jan; GREGOR, Jan – překlad. 1. vydání. Brno: ZONER software s.r.o. 2008. 1584 s. ISBN 978-80-7413-008-3.
- [24] Wikipedie. *Microsoft Visual Studio* [online]. 2010-03-22 [cit. 2010-03-24]. Dostupné z: <http://cs.wikipedia.org/wiki/Microsoft_Visual_Studio>.

PŘÍLOHY

Přiložené CD obsahuje:

- elektronickou podobu této diplomové práce (Text\DP_Matejka.pdf);
- zdrojové kódy aplikací (Applications\Sources);
- zkompilevané aplikace (Applications\Compiled);
- instalátor Microsoft .NET 4.0 (Install\dotNetFx40_Full_setup.exe).