JIHOČESKÁ UNIVERZITA V ČESKÝCH BUDĚJOVICÍCH

PEDAGOGICKÁ FAKULTA

KATEDRA FYZIKY

Vzdálené ovládání a řízení pomocí protokolu TCP/IP

DIPLOMOVÁ PRÁCE

Autor práce : Bc. Jan Šítal

Vedoucí práce: Ing. Michal Šerý

Datum odevzdání: 30. dubna 2009

Anotace

Návrh aplikace klient – server pro vzdálené ovládání. Softwarová realizace aplikace klient a aplikace serveru v jazyce C#. Využití protokolu TCP/IP a jeho vlastností pro navázání spojení. Doporučený hardware pro server. Popis spojení mezi klientem a serverem. Serverem ovládaný prvek s využitím jednočipového mikroprocesoru Atmel AT89C2051 pro vzdálené ovládání. Návrh desky plošného spoje. Program pro mikroprocesor Atmel AT89C2051 v jazyce Assembler.

Annotation

Proposal of the client - server software for remote control. Software implementation of client - server applications in C # language. Use TCP/IP protocol and its properties for the connection. Recommended hardware for the server. Description of the connection between the client and server. Server controlled element using mikroprocesor Atmel AT89C2051 microprocessor for remote control. Draft printed circuit boards. Develop program for the Atmel AT89C2051 microprocessor in Assembler language.

Prohlášení:

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně, pouze za odborného dohledu vedoucího diplomové práce Ing. Michala Šerého a že jsem se držel všech uvedených zásad pro vypracování. Dále prohlašuji, že veškeré podklady ze kterých jsem čerpal jsou uvedeny v seznamu použitých zdrojů. a v souladu s § 47b zákona č. 111/1998 Sb. v platném znění souhlasím se zveřejněním své diplomové práce, a to v nezkrácené podobě, fakultou elektronickou cestou ve veřejně přístupné části databáze STAG provozované Jihočeskou univerzitou v Českých Budějovicích na jejích internetových stránkách.

V Dobré Vodě u Českých Budějovic Dne..... Podpis.....

Poděkování:

Tímto bych chtěl poděkovat vedoucímu diplomové práce za poskytnuté rady, materiály, náměty a obětovaný čas při vedení této diplomové práce. Dále bych tímto chtěl poděkovat všem, kteří mi jakkoli pomohli při studiu a velký dík patří zejména mým rodičům za podporu během studia.

Obsah:

ÚVOD

Výběr tématu diplomové práce není podle mého názoru jednoduché rozhodnutí. Když jsem se v této situaci ocitl já, měl jsem jen přibližnou představu o zaměření.Témata bylo z čeho vybírat, ale bez konkrétní představy je rozhodování vždy velice složité. Vzhledem ke skutečnosti, že mezi mé zájmy patří datové sítě, výpočetní technika, programování a elektronika, přikláněl jsem se k výběru tématu spíše z této oblasti. Jelikož i studovaný obor je mimo jiné zaměřen do okruhu mých zájmů, zvolil jsem nakonec téma, ve kterém se všechny tyto oblasti prolínají.

Základem pro vypracování této diplomové práce jsou pokročilé znalosti ze všech zmíněných odvětví, což mi práci zrovna nezlehčuje, ale věřím, že to pro mne nebude problémem. V této práci se setkáte s programovacím jazykem C# a prostředím .NET, protokolem TCP/IP (Transmission Control Protocol / Internet Protocol), hardwarem pro klientskou stanici a serverovou stanici, jejich obslužným softwarem a ovládacím prvkem s jednočipovým mikroprocesorem.

Pevně věřím, že tato práce nebude přínosem jen pro mne, ale že v ní najdou inspiraci i ostatní zájemci o podobnou problematiku vzdáleného ovládání a řízení pomocí protokolu TCP/IP. Možností, které dnešní oblast programování, elektroniky a datových sítí nabízí je tak rozmanitá, že stejný problém lze řešit několika zcela odlišnými způsoby.

1 Proč použít protokol TCP/IP

Tento protokol se používá po celém světě ať už se jedná o domácí nebo privátní sítě PAN,metropolitní ať veřejné nebo neveřejné sítě MAN či dokonce sítě překračující hranici států a kontinentů WAN. Protokol zasahuje snad do všech odvětví co se datových sítí týče a jeho použití a komptabilitu lze považovat téměř za bezproblémovou. Použití je tedy velice rozsáhlé s v současné době by bylo chybou se nezmínit o síti internet, která je na tomto protokolu postavená. Protokol TCP/IP lze právem považovat za fenomén, díky kterému mohl internet tak jak ho známe dnes vůbec vzniknout. Málokoho by však napadlo, že koncepce jako taková je stará téměř 30 let.

Přestože se stal standardním souborem protokolů teprve v nedávné době, je starý již více jak dvacet let. V počátku byl použit pro spojení vládních počítačů (síť ARPANET - předchůdce dnešního Internetu), nyní je jeho největší využití právě v síti Internetu, jenž se stala největší celosvětovou sítí. Jako TCP/IP standard se tento síťový protokol začal prosazovat v době, kdy byl implementován do systému UNIX a jemu podobných, zhruba někdy kolem 80 let. Díky této podpoře a zároveň díky jeho vyplývající historické kompatibilitě vůči velkému množství hardwarových
a softwarových systémů se dnes těší velké rozšíření. Není také bez zajímavosti jeho využití u většiny dnešních her (podporující síťové funkce), komunikačních programů a jednoduchých sítích peer-to-peer, což si většina lidí ani neuvědomuje. TCP/IP jakožto standart je definován v TCP je definován v RFC 793.

1.1 Historie protokolu TCP/IP

Úplné počátky vzniku používání síťového protokolu TCP/IP se nacházejí v době, kdy žádostí ministerstva obrany v USA byl vznesen požadavek, na vytvoření komunikační sítě, jenž by zabezpečovala spojení i v případě jaderné války, přesněji řečeno ještě o několik let zpět. Tato síť tedy měla poskytovat jednoduchou a bezproblémovou komunikaci v případě jaderného ohrožení a dokonce i po jaderné válce[13].

Předchůdce TCP/IP původem ARPA-protokolem pro komunikaci byl **NCP** (Network Control Protocol), ale potom byl nahrazen propracovanější technologií TCP/IP. Nejedná se o jeden protokol, ale o celou sadu protokolů (v současnosti se jedná zhruba o několik desítek protokolů a další vznikají). Množství protokolů, které se dnes vyskytují u různých aplikací je velké množství a další , s vývojem IT přibývají. V anglické terminologii se proto o těchto protokolech hovoří jako o " TCP/IP protokol

suit", tedy jak již bylo zmíněno o rodině protokolů TCP/IP. Příklad některých aplikačních protokolů TCP/IP:

- HTTP (HyperText Transfer Protocol) - protokol pro komunikaci mezi WWW servery a jejich klienty
- SMTP (Simple Mail Transfer Protocol) - poštovní protokol pro vzájemnou komunikaci mezi poštovními servery, prostřednictvím kterého si jednotlivé servery předávají mezi sebou konkrétní zprávy
- FTP (File Transfer Protocol) - protokol pro přenos souborů mezi uzlovými počítači sítě.
- NTP (Network Time Protocol) - protokol použitý pro synchronizaci času
- SNMP (Simple Network Management Protocol) protokol sloužící potřebám správy sítí.

1.1.1 Dvě základní skupiny TCP/IP

I přes velkou rozmanitost rodiny TCP/IP je a vždy bude protokol rozdělen pouze na dvě základní skupiny [1]:

1 TCP (Transmission Control Protocol)
Je protokol transportní vrstvy z modelu ISO/OSI. Hlavním účelem protokolu TCP je získávat elektronické zprávy libovolné délky a převádět je do sekvence paketů, zpravidla o velikosti 64kb (poslední může být samozřejmě menší), na zdrojovém uzlu a pak je znovu sestavuje do původních zpráv na cílovém uzlu sítě.
Díky tomu může software řídící síťovou komunikaci zasílat zprávy po částech a kontrolovat každou z těchto částí samostatně. V případě, že se nepodaří daný paket přenést, tak se přenos opakuje. Efektivita přenosu je právě dána paketovým přenosem. Při chybě v přenosu se nemusí posílat celý "balík" dat, ale jen chybný paket.

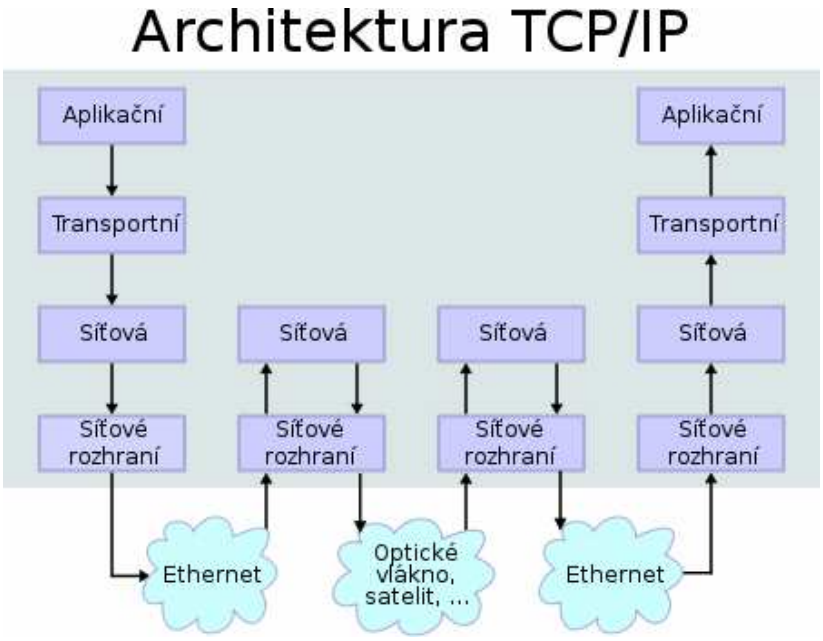
2 IP (Internet Protocol)
Je protokol síťové vrstvy a u každého paketu ověřuje jeho korektnost a obhospodařuje adresování, a to tak, aby pakety mohly být směrovány nejen přes řadu uzlů, ale dokonce i přes řadu sítí pracujících s různými komunikačními protokoly - nejen s původním ARPANETovským NCP standardem, ale i s jinými protokoly, jako jsou např. Ethernet, FDDI nebo X.25. Dále zajišťuje, aby byly pakety posílány ve správném pořadí a co možná nejvhodněji, co se týče cesty přenosu.

1.2 Problematika přenosu pomocí TCP/IP

TCP je spojově orientovaný protokol používaný pro spolehlivou přepravu dat přes síť. Protokol se nachází ve 4té vrstvě modelu OSI, ve které jsou protokoly dva – TCP a UDP . Spolehlivá a efektivní přeprava dat po síti pomocí TCP je zajišťována proudovým přenosem. Není potvrzován každý paket, ale skupina nazývaná „window“ kde je každý byte identifikován sekvenčním číslem. Pokud TCP dostane od vyšších vrstev balík dat, rozdělí je do segmentů, označí sekvenčním číslem a pošle 3. vrstvě (IP) k přenosu. Dále pak potvrzováním příjmu skupiny paketů, ztracené nebo opožděné pakety příjemce nepotvrdí a odesílatel je pošle znovu.

Efektivní řízení toku spočívá v tom, že nepotvrzuje každý paket, ale skupinu (příjemce informuje odesílatele jaké množství paketů je schopen přijmout čímž je zamezeno přetečení jeho interních bufferů). TCP umožňuje plně duplexní operaci, přijímat i odesílat data současně a multiplexing, možnost datových toků různých aplikací vyšších vrstev najednou prostřednictvím jednoho spojení.

Kromě TCP se ve 4té vrstvě modelu OSI setkáme ještě s protokolem UDP. Tento protokol vyšle data aniž by navazoval jakékoliv spojení s nějakým uzlem. Na rozdíl od TCP posílá data v celém bloku. Očekává tedy od své bezprostředně vyšší vrstvy vždy celý blok dat, který se snaží přenést opět jako celek (v rámci jediného tzv. uživatelského datagramu), a na straně příjemce jej předává své bezprostředně vyšší vrstvě opět jako celek. Tento protokol je nespojový, neexistuje žádná kontrola o doručení jednotlivých datagramů[1].



Obr. 1.1 – architektura TCP/IP

1.2.1 Protokol TCP/IP ve vztahu k referenčnímu modelu OSI

Začátkem 80-tých let byl TCP/IP implementován jako integrální součást Berkley UNIXu verze 4.2. V roce 1983 byl protokol TCP/IP přijat americkou armádou jako standard pro síťové komunikace. Právě IP protokol získal takovou oblibu a je v něm spatřována budoucnost.

IP protokol není proprietární (na rozdíl od např. IPX, DecNET, SNA, …) a není uzavřený, je vyvíjen a rozvíjen na základě široké spolupráce výrobců a uživatelů. Není monolitický, ale je definovatelný na základě modelu OSI,
zajišťuje interoperabilitu mezi různými platformami (PC, Workstation, Mini, Mainframe; Unix, Windows, Mac) a výrobci (IBM, HP, Sun, PC, …). Dále jsou na něm založeny aplikace typu klient/server (např. SAP R/3) a v poslední řadě jsou na něm založeny webové technologie a Internet [13].

7. aplikační
6. presentační
5. spojová
4. transportní
3. síťová
2. linková
1. fyzická

Tab. 1.1 - OSI

Application (5-7) TCP	**Application (5-7) UDP**
Transport (4)	
Internet (3)	
Interface (1 a 2)	

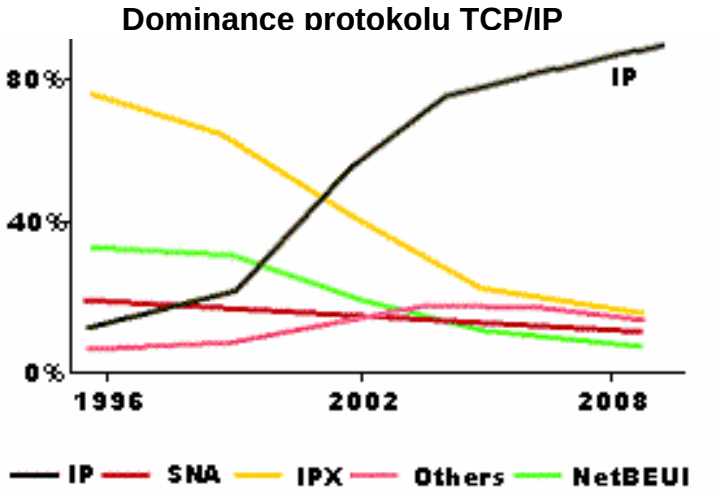
Tab 1.2 - IP

Vrstva Interface není tímto modelem popsána. Zajišťuje protokolům IP funkčnost pro enkapsulaci datagramů a jejich přenos specifickým médiem (včetně relace mezi IP a MAC adresou).
Vrstva Internet používá protokol nazývaný Internetwork Protocol – zkratka **IP** a jeho prostřednictvím zajišťuje :

- služby doručení datagramů bez závislosti na fyzické médium (vrstvu Interface);
- adresní mechanismus
- směrovací schéma pro přenos dat

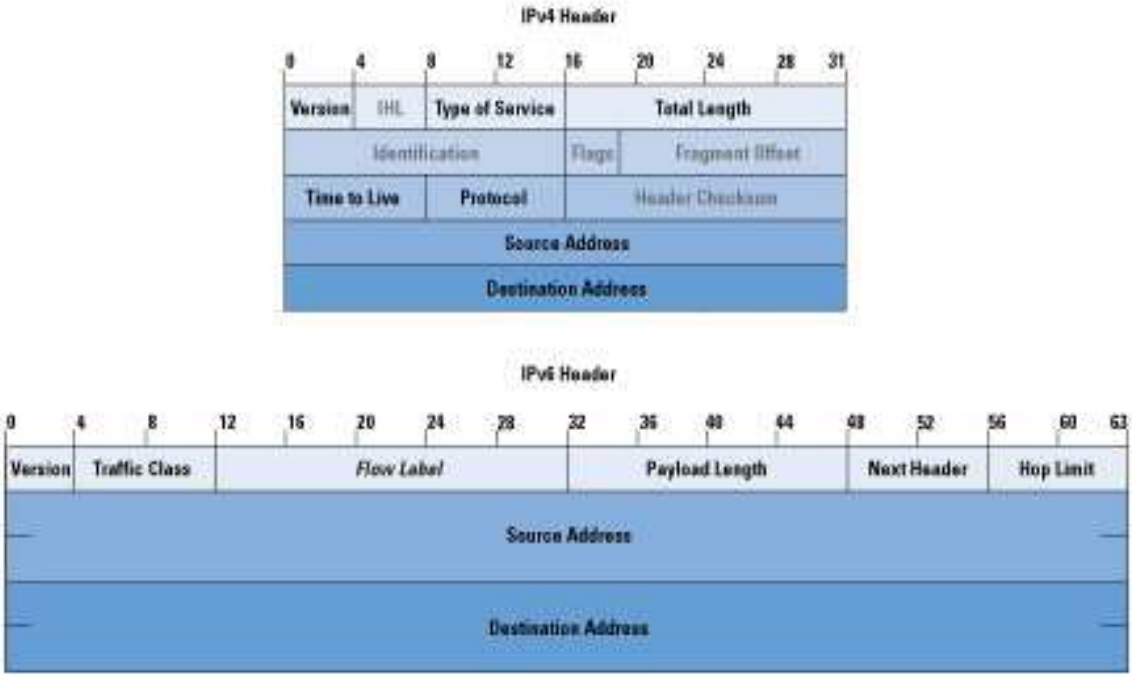
Vrstva Transport zajišťuje spolehlivý přenos dat mezi dvěma koncovými uzly. Míra spolehlivosti odpovídá požadavkům aplikace. Pro přenos dat jsou používány dva typy protokolů – Trasmission Control Protocol známý zkratkou jako TCP a User Datagram Protocol - UDP.

Vrstva Application zahrnuje protokoly specifikující procedury pro uživatele (přihlašování se k serverům, přenos souborů atd.). Procedury (aplikace) jsou rozděleny podle použitého protokolu vrstvy Transport. Jsou jimi protokoly UDP a TCP.



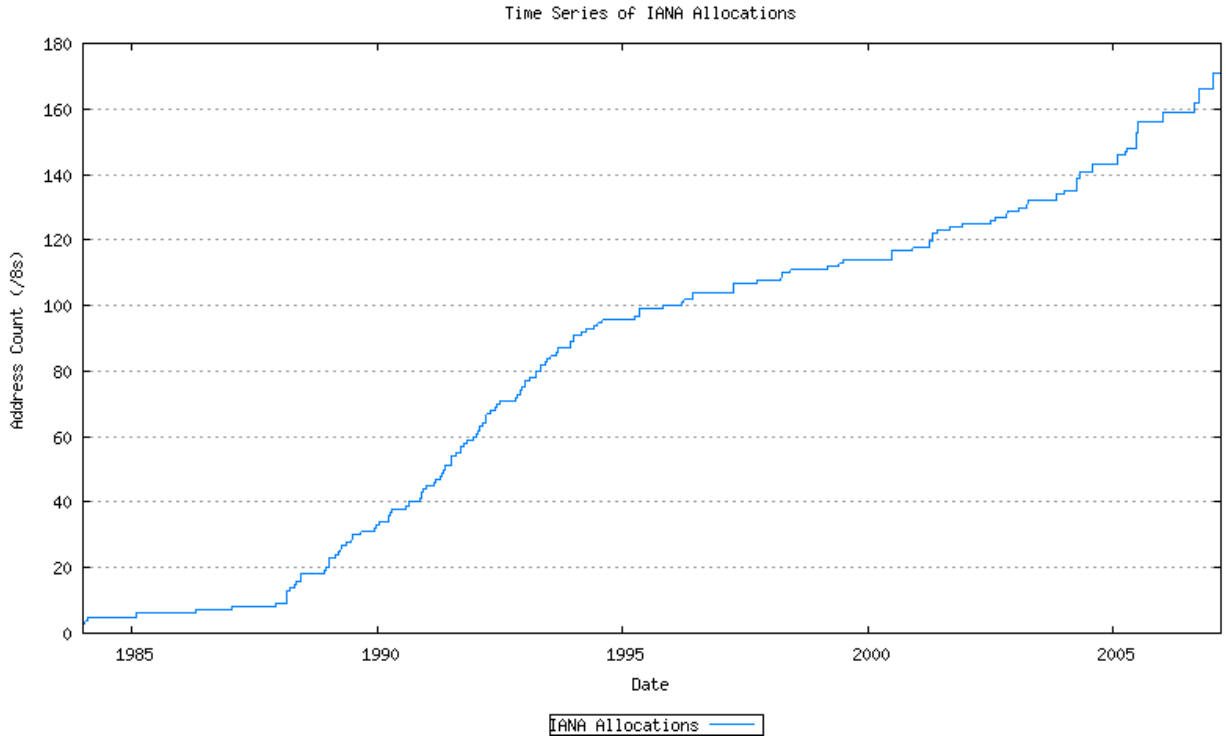
Obr 1.2 – podíl protokolu TCP/IP

Z grafu 1.2 je patrné, že obliba rodiny protokolů neustále roste a posiluje. Není se čemu divit, o rychlý vzestup se dle mého názoru zasloužilo velice rychlé rozšíření sítě internet což odpovídá počátku vzestupu po roce 1996. Je to důkazem i toho, že protokol je velice všestranný a lze ho přizpůsobit a použít téměř pro každou aplikaci. I tento protokol, přesněji spíše nejrozšířenější současná verze IPv4 pomalu začíná narážet na své slabiny. Největší zásluhu na tom nemá nikdo jiný než internet, díky kterému se dostal tak rychle do popředí. V současné době se již začíná narážet na adresní rozsah, který v měřítku běžných sítí až do velikosti MAN není problémem. Bohužel v celosvětovém měřítku ať chceme nebo ne, jednou narazíme na mantinel a setkáme se s problémem dalších volných adres, které však již nebudou k dispozici.



Obr. 1.3 – porovnání adersního rozsahu IPv4 a IPv6

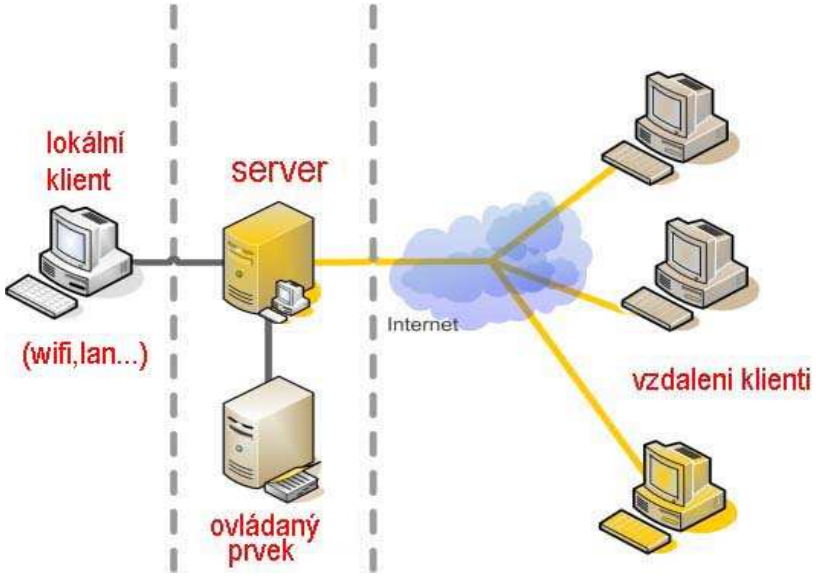
V současné době je sice již znám nástupce označovaný jako IPv6 a dokonce se již v některých aplikacích používá. V pohledu vývoje původní IPv4 je vyvíjeno již přes 30let a teprve v nedávné době se začalo masově využívat. Jaký bude osud nástupce IPv6 by bylo velice malicherné z tohoto pohledu nyní posuzovat. Nicméně nezbývá nám nic jiného, než se nechat překvapit, zda se tato verze bude těšit takové oblibě, nebo jestli ji dříve než se stihne dostat do popředí nahradí některý jiný standart. Vize je velice přesvědčivá, představu, že každé vyrobené elektronické zařízení může mít celosvětově jedinečnou adresu, která bude použitelná téměř kdekoli a nebude potřeba jí měnit nese veliké předpoklady pro budoucnost.



Obr 1.4 – graf přidělování 8bitových prefixů IPv4

2 Návrh aplikace:

Jak již bylo zmíněno úvodem, jedná se o aplikaci typu klient – server. Hlavní výhody tohoto jsou v nezávislosti polohy klienta vůči serveru a ovládanému prvku. Z principu obecné nezávislosti protokolu TCP/IP na fyzické vrstvě modelu ISO/OSI je tedy jedno, zda klient i server jsou spojeny přímo sítovým kabelem nebo je každý na jiném konci světa a spojení je realizováno pomocí sítě internet či nějakého privátního virtuálního okruhu. Další nespornou výhodou je možnost připojení více klientů na jeden server, co nám umožňuje pomocí serveru řídit ovládaný prvek z více míst najednou nebo mít prostě klienta nainstalovaného na více místech a ovládání uskutečnit dle potřeby bez nutnosti fyzického přemisťování klienta.



Obr. 2.1 – topologie klient - server

Může být například klient připojený přímo na místě kde je server, ale pro toto použití by nemusela být aplikace takto řešena. Není problémem, aby byl klient nainstalován v přenosném počítači a poté se k serveru připojit jak po kabelech místí sítě, bezdrátově přes wifi nebo pomocí kterékoli jiné drátové či bezdrátové technologie.

O spojení s ovládacím prvkem se stará v tomto návrhu jen server. Ze strany klienta je toto propojení nepodstatné a není tedy potřeba u klienta řešit a nastavovat připojení, pokud se za server připojí jiný ovládaný prvek. Co mají klient a server společné, jsou příkazy na ovládání. I ty ale lze v klientovi ponechat a případnou úpravu řešit až na serveru. Klient je tudíž absolutně nezávislý na ovládaném prvku a z jeho pohledu není potřeba řešit co konkrétně server ovládá. To umožňuje velkou univerzálnost použití této aplikace jako celku a snadné přizpůsobení dle aktuálních požadavků daného nasazení.

2.1 Cíl řešení

Cílem této diplomové práce je tedy navrhnout funkční software typu klient – server. Dále pak k serveru pomocí USB portu připojit zařízení s jednočipovým mikroprocesorem, které bude pomocí serveru přijímat příkazy od klienta. Na výstupu zařízení s jednočipovým mikroprocesorem budou 4 spínané výstupy. Každý výstup bude možné sepnout samostatně.

Jedna z hlavních vlastností je nezávislost propojení klienta se serverem na fyzické vrstvě přenosového média. Je tedy jedno, zda bude spojení realizováno pomocí kabeláže, wifi technologií, pomocí internetu apod.

Druhým předpokladem je svým způsobem jistá platformová nezávislost, v tomto případě je myšlenka taková, aby bylo snadno nožné aplikaci upravit a spouštět i pod jinými operačními systémy než jsou Microsoft Windows. Tento požadavek je kladen hlavně na serverovou část aplikace u které je větší předpoklad, že jí bude potřeba spouštět například v systémech typu Linux.

Základní požadavky na klienta jsou:

- lehké ovládání
- uživatelsky přívětivé grafické prostředí
- snadná instalace a nastavení
- možnost ověřit spojení se serverem před posláním příkazu
- potvrzení o přijetí příkazu serverem

Základní požadavky na server jsou:

- hardwarová nenáročnost a to jak z hlediska ceny hardware, spotřeby elektrické energie, údržby atd.
- automatické spuštění při startu bez nutnosti po zapnutí konfigurovat server
- možnost snadno upravit a spouštět server na jiném OS než Microsoft Windows

2.2 Ukázková aplikace

V této ukázkové aplikaci bude použití tohoto systému vypadat následovně. Jedná se možnost vzdáleně vypnout a zapnout rádiové jednotky připojené k centrálnímu routeru. Toto je příklad malého wifi převaděče pro distribuci bezdrátového připojení k internetu. Součásti převaděče jsou celkem 3 rádiové jednotky pro multipoint a jednu rádiovou jednotku pro spojení peer to peer (páteřní spoj). Instalace dále obsahuje jeden centrální router. Pasivní části není pro příklad v tomto popisu potřeba uvádět.

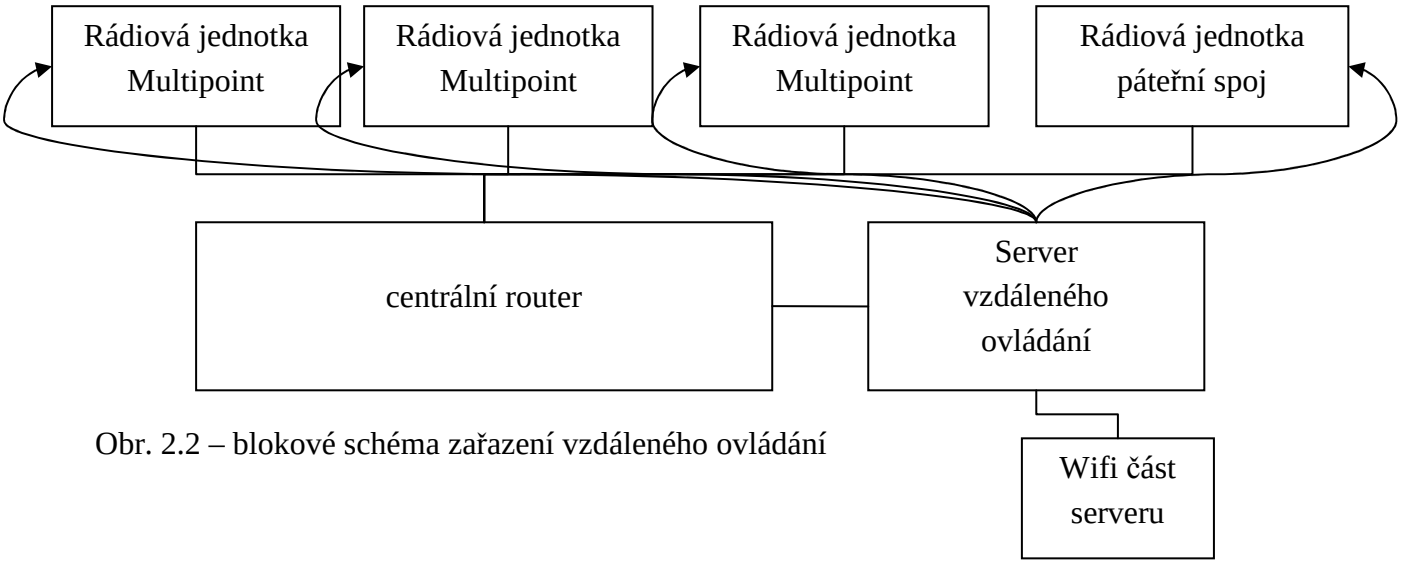
Ze zkušeností je známe, že rádiové jednotky pro multipoint občas trpí tzv. „tuhnutím“. Jedná se o stav, kdy jednotka je sice v síti viditelná jako aktivní, ale nereaguje na žádné podněty. Tímto neduhem trpí zejména jednotky použité jako

multipoint. Jediná pomoc je poté jednotu na chvíli vypnout a poté znova zapnout. Právě k tomuto účelu se hodí použití této aplikace. Centrální router je většinou běžný osobní počítač postavený na platformě x86 s operačním systémem buď Microsoft Windows nebo Linux. V operačním systému je poté spuštěn některý routovací systém. Není proto problémem, aby na routeru byla spuštěna serverová část navrhované aplikace. Podmínkou je, aby router byl vybaven USB portem pro připojení prvku s jednočipovým mikropočítačem. Dále je potřeba dát pozor na nastavení firewallu routeru, aby neblokoval příchozí a ochozí spojení na daných portech určených k ovládání.

Pokud by tuto možnost řešení nedovolovalo, je možné serverovou část nainstalovat na samostatný počítač, který by byl u převaděče připojen jako další nezávislé síťové zařízení. Výhodou může být možnost osazení samostatnou wifi kartou. To nám dává možnost se k serveru připojit bezdrátově, aniž by bylo nutné fyzicky navštívit objekt, kde je převaděč instalován. Dále pak možnost ovládat vzdáleně celý převaděč. Může se stát, že přestane reagovat jednotka u páteřního spoje a poté se ani na serverovou aplikace nebude možné připojit. Nevýhodou je nutnost instalovat další zařízení s ohledem na jeho součásti a spotřebu elektrické energie.

Podmínkou je, aby router byl vybaven USB portem pro připojení prvku s jednočipovým mikropočítačem. Dále je potřeba dát pozor na nastavení firewallu routeru, aby neblokoval příchozí a odchozí spojení na daných portech určených k ovládání.

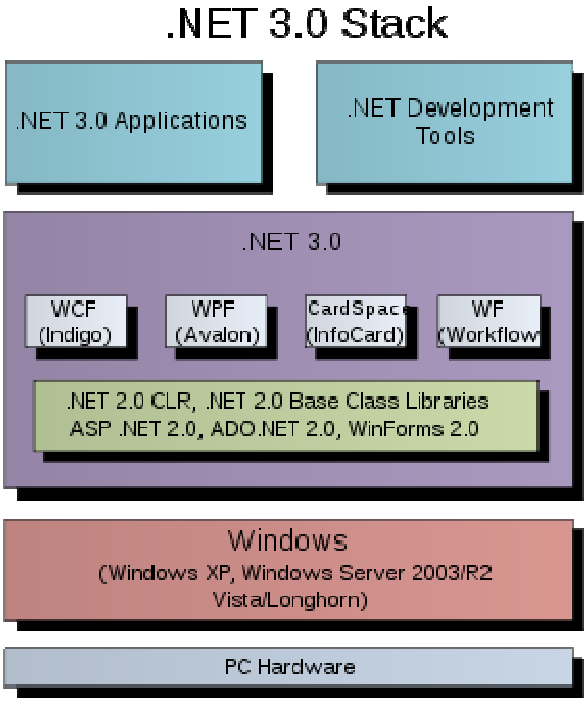
Podmínkou je, aby router byl vybaven USB portem pro připojení prvku s jednočipovým mikropočítačem. Dále je potřeba dát pozor na nastavení firewallu routeru, aby neblokoval příchozí a odchozí spojení na daných portech určených k ovládání.



Obr. 2.2 – blokové schéma zařazení vzdáleného ovládání

2.3 Platformová nezávislost

Jedno z dalších hledisek je, aby aplikace měla jistou platformovou nezávislost a aby ji bylo možné jednoduše upravit a spouštět pod různými operačními systémy. To je jeden z důvodů, proč byl k naprogramování klienta a serveru zvolen jazyk C#. Pro běh aplikace postačí mít na počítači nainstalovaný .NET framework, konkrétně verzi 2.0. Tento požadavek je hlavně na server, u kterého je větší předpoklad, že bude spouštěn i na počítačích s operačním systémem typu linux. Pro běh pod linuxem se server liší jen v jedné třídě. Jedná se o část, která se stará o odesílání příkazů do jednočipového mikropočítače. Běh programu pod operačním systémem Microsoft Windows není z podstaty prostředí .NET problémem. Trochu jiná situace nastane, pokud budeme chtít rozběhnout klienta nebo server pod operačním systémem typu linux.



Obr. 2.3 - .NET Stack

Velice vhodný je pro využití C# v linuxu projekt Mono. Mono Projekt je otevřený výzkum počátečně sponzorován firmou Novell aby vyvinul openource, UNIX verzi Microsoft .NET vývojářská platforma. Jeho cílem je umožnit UNIX vývojářům stavět a rozvíjet cross-platform .NET Aplikace. Projekt poskytuje různé technologie vyvinuté společností Microsoft, které byly nyní předloženy u ECMA ke standardizování[18].

Mono projekt se také zaměřil na vývoj komponent založených na C#, knihoven a běhových prostředí. Nejdůležitější z nich, které jsou vyvíjeny Mono týmem jsou tyto:
- Gtk (http://gtk-sharp.sf.net): vazbu na populární Gtk+ GUI toolkit pro UNIX a Windows systémy. Další vazby jsou dostupné v: Diacanvas-Sharp a MrProject.

- ZipLib (http://www.icsharpcode.net/OpenSource/SharpZipLib/Default.aspx): knihovna pro manipulaci s různými druhy archívů (Zip and tar).
- Tao Framework vazba na OpenGL
-Mono.Directory.LDAP Novell.Directory.LDAP: LDAP přístup pro .NET aplikace.
-Mono.Data podpora pro PostgreSQL, MySQL, Firebird, Sybase ASE, IBM DB2,
-SQLite, Microsoft SQL Server, Oracle, a ODBC datové zdroje.
-Mono.Cairo vazba na Cairo (http://www.cairographics.org) vykrelovací jednotka (Vlastní systém.Je zde implementováno kreslení).
-Mono.Posix Mono.UNIX: vazby pro tvorbu POSIX aplikací s použitím C#.
-Mono.Remoting.Channels.Unix Unix ovládání na bázi socketů
-Mono.Security zdokonalené bezpečnostní a cryptovací prostředí
-Mono.Math tvorba BigInteger a prvočísel
-Mono.Http podpora pro tvorbu vlastních, vestavěných HTTP serverů a běžných HTTP handlerů pro vaše aplikace.
-Mono.XML rozšířená podpora pro XML
-Managed.Windows.Forms (aka System.Windows.Forms) kompletní a multiplatformní,
-System.Drawing based Winforms implementace.
-Remoting.CORBA (http://remoting-corba.sourceforge.net/): A CORBA implementace pro Mono.
-Ginzu implementace pro spicku vzdákeného přístupu ICE (http://www.zeroc.com)

Mono obsahuje mnoho komponent, které vám pomáhají vytvořit aplikace. Common Language Infrastructure (CLI) virtual machine která obsahuje class loader, Just-in-time compiler a garbage collecting runtime. Třídní knihovnu, která spolupracuje s každým jazykem který běží na CLR. Stejně jako .NET kompatibilní třídní knihovny, a také Mono knihovny jsou obsaženy. V poslední řadě compilátor pro C#. V budoucnu budou mož pracovat na jiný kompilátorech, které poukazují na Common Language Runtime.

Windows má kompilátory, které vytváří normu pro virtual machine, jsou to tyto jazyky: (http://msdn.microsoft.com/net/thirdparty/default.asp#lang) Managed C++, Java Script, Eiffel, Component Pascal, APL, Cobol, Perl, Python, Scheme, Smalltalk, Standard ML, Haskell, Mercury a Oberon. CLR a Common Type System (CTS) umožňuje aplikacím a knihovnám aby byly zapsány v celku skládacích se z různých jazyků, které ukazují na bytový kód. To například znamená, že když definujete třídu pro algebraickou manipulaci C#, tato třída může být znovu použita jakýmkoliv jiným jazykem s podporou CLI. Můžete vytvořit třídu v C#, podtřídu v C++ a instanci v Eiffel. Jednoobjektový systém, threading system, class libraries, and garbage collection systém může být sdílen těmito jazyky.

Najdou se samozřejmě rozdíly mezi Mono a .NET. The ".NET výhoda" jsou přídavné služby společnosti Microsoft, jedna část je multiplatformní vývojové prostředí. Mono je implementace pouze vývojového prostředí, ale není to implementace ničeho jiného co by souviselo s .NET, jako je Passport nebo software-as-a-service.

3 Server

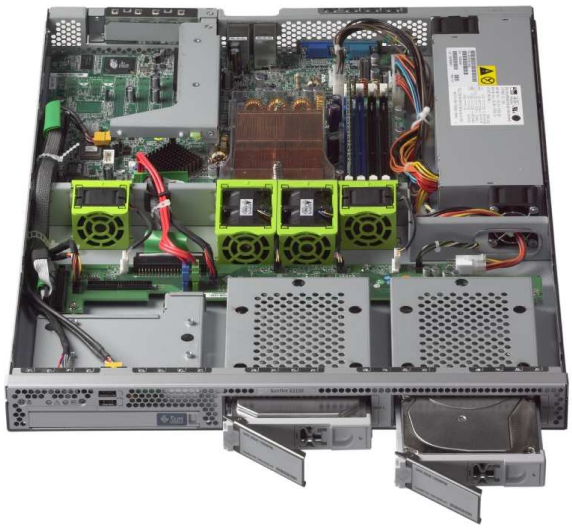
Pod pojmem server si snad téměř každý představí výkonný počítač, poskytující v síti základní síťové služby. informatice obecné označení pro počítač, který poskytuje nějaké služby nebo počítačový program, který tyto služby realizuje. V unixových systémech je označován jako démon (anglicky daemon), v Microsoft Windows pak jako služba (anglicky service). Servery jsou buď umístěny volně nebo ve speciální místnosti, kterou označujeme serverovna (s klimatizací, zabezpečovacím zařízením apod.). Pro úsporu místa se mohou zakládat do speciálních skříní – tzv. rack.

Služby server poskytuje klientům, což označujeme jako model klient-server (odlišné modely jsou peer-to-peer nebo friend-to-friend). Služby mohou být nabízeny v rámci jednoho počítače (lokálně) nebo více počítačům pomocí počítačové sítě (síťové služby). Lokální službou může být například obsluha připojené tiskárny, správa automatických aktualizací a podobně.

Služby, které server poskytuje v lokální síti (LAN) může být například například sdílení disků, tiskáren nebo schopnost ověřit uživatele podle jména a hesla (autentizace). Ve větších sítích, jako je Internet, servery uchovávají a nabízejí webové stránky a poskytují další služby (DNS, e-mail atd.). Poskytování služby zajišťuje speciální program. V unixových systémech je označován jako démon (anglicky *daemon*), v Microsoft Windows pak jako služba (anglicky *service*), který s klientem komunikuje pomocí definovaného protokolu (SMB pro sdílení disků a tiskáren ve Windows, HTTP pro webový server a podobně).

Podle toho, jestli je server vyhrazen jen pro poskytování služeb, nebo může sloužit i uživatelům servery rozlišujeme na:

- dedikovaný – vyhrazený pro speciální účely, bez přímého přístupu uživatelů
- nededikovaný – server slouží uživateli zároveň jako obyčejný počítač



Obr. 3.1 – příklad serveru ve skříní rack 19“

3.1 Základní typy serverů

Jak asi každý tuší, rozlišují se servery podle jejich použití. Serverů je mnoho typů a druhů a některé stroje se speciálním využitím by možná serverem nazval málokdo. Servery lze rozlišovat buď podle hardware nebo podle software a jeho využití. Z pohledu hardware se nejčastěji servery rozlišují na:

- jednoprocesorové, víceprocesorové
- desktopové, rackové
- a další

Jelikož vyvíjená aplikace v tomto případě na hardware velice nenáročná, je pro nás důležitější spíše softwarová stránka věci [14]. Z tohoto pohledu se servery nejčastěji rozlišují na:

- webový server – především v síti Internet poskytuje WWW stránky
- souborový server – slouží např. v podnikové síti jako centrální úložiště dat (dokumentů)
- databázový server – slouží jako úložiště strukturovaných dat (databází)
- tiskový server – zpřístupňuje počítačové tiskárny
- faxový server
- proxy server – zprostředkovává přístup do jiné sítě jiné (např. Internet)
- aplikační server – počítač specializovaný na provoz nějaké aplikace
- herní server – nabízí hraní her s více hráči (multiplayer)
- a další

3.2 Serverové operační systémy

Operační systém obecně znamená základní programové vybavení počítače (tj. software), které je zavedeno do paměti počítače při jeho startu a zůstává v činnosti až do jeho vypnutí. Skládá se z jádra (kernel) a pomocných systémových nástrojů. Hlavním úkolem operačního systému je zajistit uživateli možnost ovládat počítač, vytvořit pro procesy stabilní aplikační rozhraní (API) a přidělovat jim systémové zdroje. Operační systém je velmi komplexní software, jehož vývoj je mnohem složitější a náročnější, než vývoj obyčejných programů.

Server jakožto zařízení poskytující služby v síti je z podstaty velice důležité zařízení. Lze říct, že pokud dojde k výpadku serveru, je celý zbytek sítě takříkajíc k ničemu. Jeden z nejdůležitějších požadavků na server vůbec je jeho stabilita. Pro stabilní a dlouhodobě dobře pracující server je potřeba, aby byl celý jeho návrh dobře

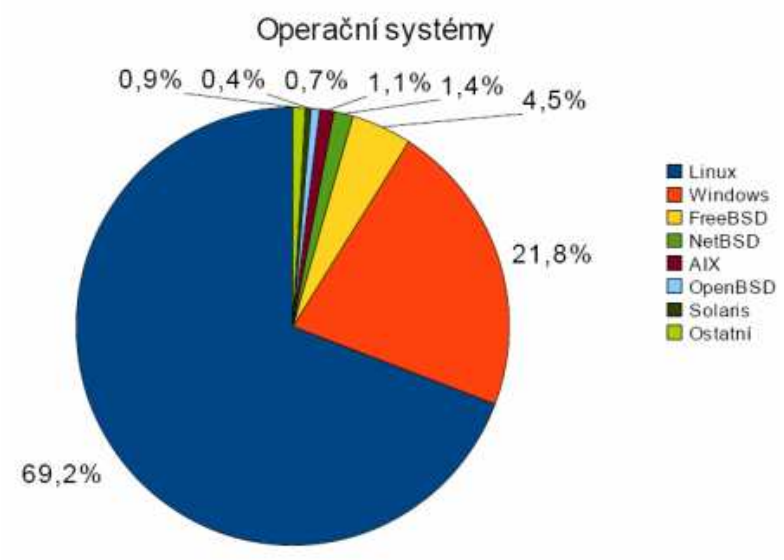
zpracován a navržen přesně jak pro prostředí, ve kterém bude umístěn tak pro služby které bude poskytovat.

Chyby při běhu serveru jsou málokdy způsobeny hardwarem, v převážné většině případu se jedná o špatně navržený nebo nastavený software. Veliký podíl na stabilitě serveru má jeho softwarové vybavení. Z toho má největší podíl operační systém provozovaný na serveru. Podle tabulky 3.1 je vidět, že operační systém je jakýsi prostředník mezi hardwarem počítače a dalším software. Už jen z této hierarchie vyplívá, že nestabilní operační systém má za následek nestabilitu celého počítače jako celku. Návrh serveru jako celku je tedy o mnoho obtížnější, než návrh běžného osobního počítače. To platí jak pro softwarovou stránku tak pro hardwarovou.

APLIKACE (SOFTWARE)
OPERAČNÍ SYSTÉM
BIOS
HARDWARE

Tab. 3.1 - hierarchie v PC

Operačních systémů je na trhu celá řada. Při nejmenším je poznáte podle ceny, která bývá mnohonásobně vyšší, než ceny operačních systémů pro běžná PC. Mohlo by se zdát, že i na poli serverových operačních systémů bude mít největší zastoupení celosvětový softwarový gigant Microsoft. V tomto případě se dá říct, že zdání klame. Podle grafu na obrázku 3.2 je vidět, že nejpoužívanější serverový operační systém je Linux. Graf může být trochu zkreslený, protože Linux jako takový není konkrétní systém. Je to jen klon UNIXové architektury a systémů postavených na této bázi je celá škála a nazývají se distribuce. Mezi nejznámější v České Republice nejspíše patří Debian, Suse linux, Fedora, Red-Hat, Slackware, Gentoo a mnoho dalších.



Obr. 3.2 – zastoupení jednotlivých serverových OS

4 Softwarové řešení aplikace – klient

U aplikace na straně klienta je oproti serverové části jeden velice podstatný rozdíl, byť nemusí být na první pohled až tak patrný a kde kdo by nad ním mávl rukou. Jedná se o to, že uživatelé, kteří budou tento software využívat, se nesetkají osobně se serverovou částí, ale s klientem. Za první dojem, který aplikace na klienta učiní je tedy odpovědná pouze tato část.

Je důležité, aby klientská část aplikace byla uživatelsky přívětivá, lehce ovladatelná a v neposlední řadě by měla dobře graficky vypadat. Po zvažování jsem se rozhodl, že klient bude vytvořen v grafickém prostředí pomocí Windows Form. Z osobních zkušeností si moc dobře nedovedu představit běžného uživatele, jak používá k ovládání aplikace příkazový řádek. Klient je záměrně řešen tak, aby uživatel nemusel zadávat parametry pro nastavení. Vše co bylo možné, je zakomponováno jako součást zdrojového kódu. Jediný parametr, který musí uživatel po spuštění klienta zadat, je IP adresa serveru, na který má být připojen.

Programovací jazyk je použit C#, programováno v Microsoft Visual Studiu 2005, .NET framework 2.0, aplikace typu Windows Form.

4.1 Klady a zápory řešení ve Windows Form

Je pochopitelné, že každé řešení má vždy nějaké výhody na úkor něčeho jiného. Jak tomu bývá snad v každém odvětví, vždy je něco za něco. Pro programátora je nejdůležitější srozumitelný a co nejjednodušší zdrojový kód, pro uživatele zase co nejlépe vypadající aplikace s pokud možno co nejlehčím ovládáním. Z osobních zkušeností si moc dobře nedovedu představit běžného uživatele, jak používá k ovládání aplikace příkazový řádek. Na druhou stranu převážná většina dnes běžně dostupných počítačů je sama o sobě dostatečně výkonná. Co se týče operačního systému, Microsoft Windows, ve kterých je aplikace spustitelná, dnes obsahuje většina prodávaných počítačů.

Hlavní výhody tohoto řešení jsou:

- grafické prostředí a možno vytvořit vzhledově příjemnou aplikaci
- lehké ovládání s využitím myši
- snadné upravení aplikace v případě potřeby přidání/odebrání tlačítek apod.
- snadné přizpůsobení aplikace dle požadavků uživatele na vzhled a ovládání

Zápory tohoto řešení jsou:

- zbytečně složitější zdrojový kód aplikace
- potřeba více součástí z operačního systému
- horší platfomová nezávislost na operačním systému a hardwaru
- vyšší nároky na hardware klientské stanice

4.2 Návrh zdrojového kódu

Zdrojový kód programu je rozdělen do čtyřech samostatných tříd. Jedná se o program.cs, tcpsender.cs, tcpreceiver.cs a form1.cs. Form1 zastupující formulář je obsahuje jednak grafický návrh formuláře v Form1.Designer.cs a samotný zdrojový kód formuláře a automaticky generovaný kód Microsoft Visual Studiem. Namespace celého programu je pojmenováno TCPklient.

4.2.1 Třída Program.cs

V této třídě není nic jiného, než vstupní bod programu a spuštění komponent pro formulář a formulář samotný. Obsahuje jedinou metodu a tou je statická metoda Main() bez návratové hodnoty.

Použité direktivy v této třídě jsou:

- System
- System.Windows.Forms

Zdrojový kód metody Main():

```
...
static void Main()
        {
                Application.EnableVisualStyles();
                Application.SetCompatibleTextRenderingDefault(false);
                Application.Run(new Form1());
        }
...
```

4.2.2 Třída Tcpreceiver.cs

Funkce této třídy je přijímání hlášení ze serveru. Obsahuje dvě metody. Metoda prijmipotvrzeni() se stará o navázání komunikace ze serverem a o následné zobrazení přijatého hlášení pomocí komponenty MessageBox. Nemá návratový typ a ani nepřijímá data při volání. Pro navazování spojení slouží funkce ze třídy TcpListener, která je součástí .Net frameworku. Jedná se o přenos typu stream. Tělo metody je uzavřeno v bloku Try-Catch, zajišťuje pro případ chyby uzavření streamu a tcplisteneru. TcpListener je nastaven tak, aby naslouchal na portu TCP 2020 z jakékoli IP adresy. Je to z důvodu, že Tcpklient může být spuštěn pokaždé z jiného místa v síti a odpadá nutnost vždy měnit nastavení klienta podle použitých IP adres.

Metoda Run() je zde z důvodu spuštění metody prijmipotvrzeni() ve vlákně. Je to z důvodu, že využívaná metoda AcceptTcpClient() ze třídy TcpListener je blokující. Pokud by nebyla spuštěna ve vlákně, došlo by k zablokování běhu celého programu. Je

vždy spuštěno pouze jedno vlákno v okamžiku, kdy se očekává hlášení ze serveru. Po přijetí hlášení je vlákno ukončeno.

Použité direktivy v této třídě Tcpreceiver jsou:

- System.Net
- System.Net.Sockets
- System.IO
- System.Threading
- System.Windows.Forms

Zdrojový kód metody prijmipotvrzení():

```
...
public static void prijmipotvrzeni()
        {
        TcpListener tcpListener = new TcpListener(IPAddress.Any,2020);
        TcpClient client = null;
            try
            {
             tcpListener.Start();
             if (!tcpListener.Pending())
               {
                  client = tcpListener.AcceptTcpClient();
                  Stream clientstream = client.GetStream();
                  StreamReader reader =
                  new StreamReader(clientstream);
                   string data = reader.ReadToEnd();
                   reader.Close();
                   client.Close();
                   MessageBox.Show(data);
                }
              tcpListener.Stop();
            }
            Catch
...
```

Zdrojový kód metody Run:

```
public static void Run()
        {
            Thread tcprecieverthread = new
            Thread(tcpreceiver.prijmipotvrzeni);
            tcprecieverthread.Start();
        }
```

4.2.3 Třída Tcpsender

Na první dojem by se mohlo zdát, že tato třída je v celém programu nejdůležitější. Dle mého názoru to sice takto může vypadat, ale podle mne jsou důležité všechny části programu stejně. Tcpsender se stará o navázání spojení se serverem a odeslání daného příkazu. Obsahuje jednu metodu SendData(). V metodě jsou využity funkce třídy Tcpclient z .NET frameworku. Tělo této metody je opět uzavřeno v bloku Try-Catch. V tomto případě nejenom z důvodu uzavření Tcpclienta v případě chyby při běhu, ale výjímku způsobí i stav, kdy spojení se serverem nelze navázat. Toho je využito i pro výpis chybového hlášení pomocí komponenty MessageBox.

Metoda nemá návratový typ, ale přijímá data ze dvou proměnných typu string. Jedno jsou samotná data, což není nic jiného než zaslaný příkaz pro server a druhá proměnná obsahuje IP adresu serveru, kam se má příkaz zaslat. Tím je umožněno z jednoho spuštěného klienta zasílat příkazy na více serverů. Spojení je navazováno na portu TCP 2010 a zadanou IP adresu serveru.

Použité direktivy v této třídě Tcpreceiver jsou:

- System
- System.IO
- System.Net
- System.Net.Sockets
- System.Windows.Forms

Zdrojový kód metody prijmipotvrzeni():

```
...
public static void SendData(string data, string ipserver)
        {
            TcpClient client = null;
            try
            {
                //pripojime se k serveru na port 2010
                client = new TcpClient(ipserver, 2010);
                //ziskame proud pripojeni a zapiseme do nej data
                Stream clientstream = client.GetStream();
                tcpreceiver.Run();
                StreamWriter writer = new StreamWriter(clientstream);
                writer.Write(data);
                writer.Close();
            }
            Catch
...
```

4.2.4 Třída Form1.cs

Jak již bylo zmíněno na začátku čtvrté kapitoly, programování pomocí formulářů Windows Form má své výhody i nevýhody. Z pohledu zdrojového kódu, kterým se zabýváme v této části se jedná o velkou nevýhodu a to ve směru automaticky generovaného zdrojového kódu, který je velice obsáhlý. Další nevýhodou je, že formulář jako takový potřebuje pro svůj běh mnoho komponent z .NET frameworku a tím i zvyšuje náročnost na systém. Zdrojový kód celého formuláře zde rozebírat nebudu, v případě zájmu je k nahlédnutí na přiloženém CD nosiči s elektronickou verzí této práce. Zmíním zde jen ukázku odeslání příkazu pomocí metody SendData() ze třídy Tcpsender.

Použité direktivy ve třídě Form1.cs:

- System
- System.Windows.Forms

Zdrojový kód použití metody SendData():

```
private void provePrikaz_Click(object sender, EventArgs e)
      {
            tcpsender.SendData(commandBox.Text, serverIP.Text);
      }
```

4.2.5 Grafické rozhraní pomocí formuláře Windows Form

I přes veškeré zápory vytváření aplikací pomocí formulářů sebou toto řešení přináší i některé výhody. Nejdůležitější je nejspíše možnost snadno vytvořit uživatelsky přívětivou aplikaci s grafickým rozhraním. Tohoto jsem využil pro vytvoření klientské části aplikace, se kterou se bude běžný uživatel setkávat nejčastěji. Mezi další výhody lze zařadit i jednodušší ovládání s využitím kurzoru myši.

Pro sestavování grafické aplikace se využívá GUI (Graphical User Interface) [8]. Není to nic jiného než uživatelské rozhraní, které umožňuje ovládat počítač pomocí interaktivních grafických ovládacích prvků. Na monitoru počítače jsou zobrazena okna, ve kterých programy zobrazují svůj výstup. Uživatel používá klávesnici, myš a grafické vstupní prvky jako jsou menu, ikony, tlačítka, posuvníky, formuláře a podobně. Použité nástroje v aplikaci z Microsoft Visual Studia 2005 jsou následující:

- formulář windows form
- Button (tlačítko)
- Label (popis)
- TexBox (pole vložení textu)

Obr. 4.1 – grafické rozhraní TCP klienta

5 Softwarové řešení aplikace – server

Ačkoli se funkčně jedná o aplikaci jako celek, je serverová část od klientské velice odlišná. Server je tvořen jako konzolová aplikace, nikoli grafická. V tomto případě ani není nutné, aby byla serverová část řešena jako grafická aplikace. K funkčnímu serveru mají přístup ve většině případů jen administrátoři, kteří pro svou práci grafické rozhraní většinou ani nepotřebují.

Musíme brát v úvahu i funkci serveru jako takovou, když už přeci jen dojde k vypnutí serveru ať vlivem externího zásahu nebo výpadkem elektrické energie, je vždy potřeba, aby server po svém zapnutí sám naskočil bez nutnosti zásahu obsluhy. V tom se skrývá problém s udržením a pamatováním nastavení serverové aplikace. V případě této aplikace je nastavení serveru řešeno nikoli zadáním a pamatováním parametrů po spuštění serveru, ale přednastavením serveru a přizpůsobením operačního systému pro běh. Rozhodl jsem se takto z důvodu, že server sám o sobě potřebuje pouze tři parametry.

Prvním je IP adresa počítače na kterém je spuštěn a ta je tak jako tak zadaná v operačním systému. Druhým jsou čísla TCP portu pro spojení s klientem a ty není potřeba měnit ani přenastavovat. Posledním parametrem je název virtuálního COM portu pro připojení prvku s jednočipovým mikroprocesorem. V serveru je přednastavený na COM15 a po při instalaci virtuálního portu je potřeba zadat v operačním systému tento název.

U aplikace typu server nesmíme opomenout ještě jednu důležitou vlastnost a tou je logování provozu. V případě testování serveru je důležité mít možnost sledovat co server skutečně dělá, případně na kterém kroku se zastaví. V tomto případě je logování řešeno výpisem stavů přímo do konzole v textovém režimu. Server zde po každém

přijatém příkazu vypisuje přijatý příkaz, hlášení, které bylo odesláno zpět klientovi a zda byl příkaz v pořádku odeslán na mikrokontrolér.

5.1 Výhody konzolového řešení

Toto řešení má nespornou výhodu v nenáročnosti aplikace jako takové. Jelikož server je zařízení, které je přizpůsobené pro nepřetržitý běh, je důležité, aby i serverová aplikace byla pro tuto podmínku odladěná a testovaná. Čím bude serverová aplikace složitější a náročnější, tím hůře půjde odladit a přizpůsobit konkrétnímu použití. Je zde nežádoucí jakýkoli vygenerovaný zdrojový kód, ve kterém se programátor obtížně orientuje a hledá případné chyby, jak tomu bývá u aplikací využívající GUI a formuláře Windows form.

5.2 Návrh zdrojového kódu

Jmenný prostor (namespace) do kterého spadají všechny třídy je nazván TCPserver. Zdrojový kód serverové části aplikace je rozdělen do pěti tříd. Jedná se o třídy Program.cs, Listener.cs, Verifier.cs, Tcpclient.cs a Port.cs. Programovací jazyk je použit C#, programováno v Microsoft Visual Studiu 2005, .NET framework 2.0, aplikace je řešena jako konzolová.

5.2.1 Třída Program.cs

Obdobně jako v případě klientské části i zde tato třída obsahuje vstupní bod programu. Je zde však jeden podstatný rozdíl. U klienta se inicializoval formulář, zde tomu tak není. Při spuštění je zde volána metoda StartListening() ze třídy Listener.cs. Pro případ nečekané chyby je volání metody uzavřeno v bloku pro správu výjímek Try-Catch. Spuštění opět provádí jediná metoda obsažená ve třídě, metoda Main().

Použité direktivy ve třídě Program.cs:

- System

Zdrojový kód metody Main():

```
...
static void Main(string[] args)
        {
            try
            {
                Console.WriteLine("Bc. Jan Šítal, AMVTp 2008/2009");
                Listener.StartListening();
            }
            catch (Exception ex)
...
```

5.2.2 Třída Listener.cs

Jak už z názvu trochu vyplývá, tato třída naslouchá a stará se o klienty, kteří se na server připojují. Obsahuje jednu obsáhlou metodu StartListening(). Začátek metody se provede pouze po spuštění. Jedná se o vytvoření jedné instance TcpListener a proměnné tcpclient. Zbytek těla je obsažen v takzvané nekonečné smyčce. V průběhu jednoho cyklu server přijme data od klienta, zjistí IP adresu a port klienta, ověří příkaz od klienta pomocí třídy Verifier.cs a podle výsledku ověření buď odešle příkaz na mikrokontrolér a klientovi potvrzení o provedení příkazu nebo příkaz vrátí zpátky s chybovým hlášením. Listener naslouchá na portu 2010 z jakékoli IP adresy. Metoda StartListening() nepřijímá žádné parametry a nemá návratový typ.

Použité direktivy ve třídě Program.cs:

- System
- System.Net
- System.Net.Sockets
- System.IO

Zdrojový kód nekonečné smyčky metody StartListening():

```
...
do
   {
      //zacneme naslouchani na urcenem portu
      listener.Start();
      //pockame na pripojeni klien
      client = listener.AcceptTcpClient();
      string remonteendpoint = lient.Client.RemoteEndPoint.ToString();
      //po pripojeni si vyzvedneme proud a nacteme z nej data
      Stream clientStream = client.GetStream();
      StreamReader reader = new StreamReader(clientStream);
      //ulozi prijata data
      string dataodklienta = reader.ReadToEnd();
      // overime prijaty prikaz
      Verifier.ValidaceDat(ref dataodklienta, remonteendpoint);
      if (dataodklienta != "chyba")
         {
           Console.WriteLine("prikaz potvrzen: {0}", dataodklienta);
           port.ZapisNaPort(dataodklienta);
         }
       clientStream.Close();
       client.Close();
       //zastavime naslouchani
       listener.Stop();
          }
       while(true);
...
```

5.2.3 Třída Verifier.cs

Tato třída se stará pouze o validaci přijatých příkazů od klienta. Testuje přijatý příkaz zda se jedná o příkaz pro mikrokontrolér, žádost o potvrzení stavu serveru nebo chybu. Obsahuje konstanty v podobě předdefinovaných příkazů, které musejí být shodné s příkazy kterým rozumí připojený mikrokontrolér. Třída má jednu metodu ValidaceDat(), která při volání přijímá dva parametry. Jeden je formátu string a jedná se o přijatý příkaz od klienta určený k ověření. Druhým je též string, ale tentokráte se jedná o takzvaný RemonteEndPoint, což je adresa klienta, ze které byla data přijata. Metoda má též návratový typ string. Zpět do smyčky metody StartListening() vrací buď zvalidovaný příkaz nebo chybu. Zároveň volá metodu PosliPotvrzeni() ze třídy tcpclient a předává hlášení ze serveru pro klienta od kterého byla data přijata.

Použité direktivy ve třídě Program.cs:

- System

Zdrojový kód metody ValidaceDat():

```
public static string ValidaceDat(ref string data, string
remontendpoint)
   {
    switch (data)
       {
      case command1:
      tcpclient.PosliPotvrzeni(potvrzeniprikazu, remontendpoint);
      return data;
...
      case potvrzenispojeni:
      tcpclient.PosliPotvrzeni("server pripojen", remontendpoint);
      Console.WriteLine("spojeni potvrzeno : {0}", data);
      data = "chyba";
      return data;
      default:
      tcpclient.PosliPotvrzeni("chybny prikaz", remontendpoint);
      Console.WriteLine("chybny prikaz : {0}", data);
      data = "chyba";
      return data;
            }
       }
...
```

5.2.4 Třída Tcpclient.cs

Metoda obsažená v této třídě se nestará o nic jiného než o navázání spojení s klientem od kterého byla přijatá data a zaslání hlášení. Jejím obsahem je jediná metoda PosliPotvrzeni(). Metoda přijímá dva parametry a to jsou data, která se mají zaslat klientovi (v tomto případě se jedná o hlášení) a adresu klienta. Oba parametry jsou typu string. Sama o sobě nemá návratovou hodnotu. I když se jedná o jednu metodu, její práci lze pro vysvětlení rozdělit na dvě části. Prvním krokem je zjištění IP adresy klienta, kterému se zasílá hlášení. Z metody StartListening() získáme sice adresu klienta, ale jedná se o celou IP adresu včetně čísla TCP portu, ze kterého byla data odeslána. To je pro naše použití nežádoucí, protože komunikace mezi serverem a klientem probíhá na dvou přesně daných portech.

První část metody se tedy stará pouze o získání samotné IP adresy klienta bez dalších znaků. Poté už jen následuje navázání spojení s klientem na portu 2020 a odeslání hlášení. Je zde použitý stream. Pro navázání spojení s klientem a odeslání dat je využita instance třídy TcpClient. Vzniklé proměnné se předá samotná IP adresa klienta a data, která má klient obdržet. Tělo druhé části metody je uzavřeno opět v bloku Try-Catch z důvodu správy výjímek, může zde dojít k situaci, že nebude možné zpět s klientem spojení navázat, ale je nutné opět uzavřít TcpClienta a instanci StreamWriteru.

Použité direktivy ve třídě Program.cs:

- System
- System.Net.Sockets
- System.IO

Zdrojový kód pro získání IP klienta:

```
...
string adresa = remontendpoint;
char[] pocet = adresa.ToCharArray();
int delkaadresy = adresa.Length;
char znak;
string z = "";
string ipadresa = "";
int index = 0;
znak = pocet[index];
while (index != delkaadresy && z != ":")
    {
     znak = pocet[index];
     index++;
     z = znak.ToString();
     if (z != ":")
       {
       ipadresa = ipadresa + z;
       }
    }
...
```

5.2.5 Třída Port.cs

Poslední důležitou součástí serverové části je třída Port.cs. Tato třída má na starosti spojení s jednočipovým mikroprocesorem. Spojení je realizované pomocí portu USB, který je emulován na virtuální sériovou linku. Obsahuje metodu ZapisNaPort(), která při volání přijímá jednu proměnou typu string a to není nic jiného než zvalidovaný příkaz pro mikrokontrolér. Metoda nemá návratový typ. Funkce je taková, že při zavolání otevře port, nastaví rychlost a poté do něj zapíše určená data. Poté do konzole vypíše textové potvrzení o odeslání dat. V opačném případě chybu. Tělo je opět uzavřeno v bloku Try-Catch z důvodu výjímky v podobě uzavřeného portu nebo náhlé chyby, která by způsobila, že port zůstane otevřený. Pro přístup na port je použita instalnce třídy SerialPort.

Použité direktivy ve třídě Program.cs:

- System
- System.IO.ports

Zdrojový kód metody ZapisNaPort

```
...
public static void ZapisNaPort(string data)
    {
     SerialPort serialPort1 = new SerialPort();
     try
      {
      // odeslani prikazu na port
      serialPort1.PortName = "COM15";
      serialPort1.BaudRate = 9600;
      serialPort1.Open();
      if (serialPort1.IsOpen)
         {
          serialPort1.Write(data);
          Console.WriteLine("prikaz zapsan na port : {0}", data);
         }
      else
         {
         Console.WriteLine("PORT UZAVREN! : {0}", data);
         }
         serialPort1.Close();
      }
      Catch
...
```

6 Doporučený hardware pro server

O této aplikaci lze říct, že je jednoúčelová i když účelovost lze hodnotit až při konkrétní implementaci a hardwarově nenáročná. Ačkoli hardware pro server je vždy o něco dražší než běžné komponenty pro počítače typu PC, u tohoto případu to nehraje zas až tak velkou roli. Aplikace je spustitelná na téměř jakémkoli obyčejném počítači typu PC. Pořád je to ale server a očekává se od něj bezproblémový a stálý chod. I když není striktně požadavek na hardware, je stejně nutné instalovat platformu, na které bude běh programu odzkoušený. Návrhu doporučeného hardware se tedy nevyhneme ani v tomto případě. Je tedy potřeba vybírat komponenty s ohledem na jejich kvalitu, výkon a cenu. Někdy je nutné vzít v úvahu i spotřebu elektrické energie i když se zajisté často setkáme s tím, že toto hledisko je až na posledních místech a mnozí se jím vůbec nezabývají. Řešení je vždy určitým kompromisem mezi těmito hledisky.

6.1 Požadavky na hardware

Jeden z nejdůležitějších požadavků na server je stabilita. Podle toho musejí být také vhodně zvolené jednotlivé komponenty. Server pro aplikaci software vzdáleného ovládání nemusí být příliš výkonný, software sám o sobě není náročný na systém. Z tohoto důvodu vychází jako vhodná platforma zařízení formátu ITX. Výhoda komponent tohoto formátu jsou malé rozměry, nízká spotřeba elektrické energie a pasivní chlazení. Pokud však máme plně využít výhody, které nám poskytuje platforma formátu ITX, je potřeba, aby i ostatní komponenty splňovaly určité předpoklady.

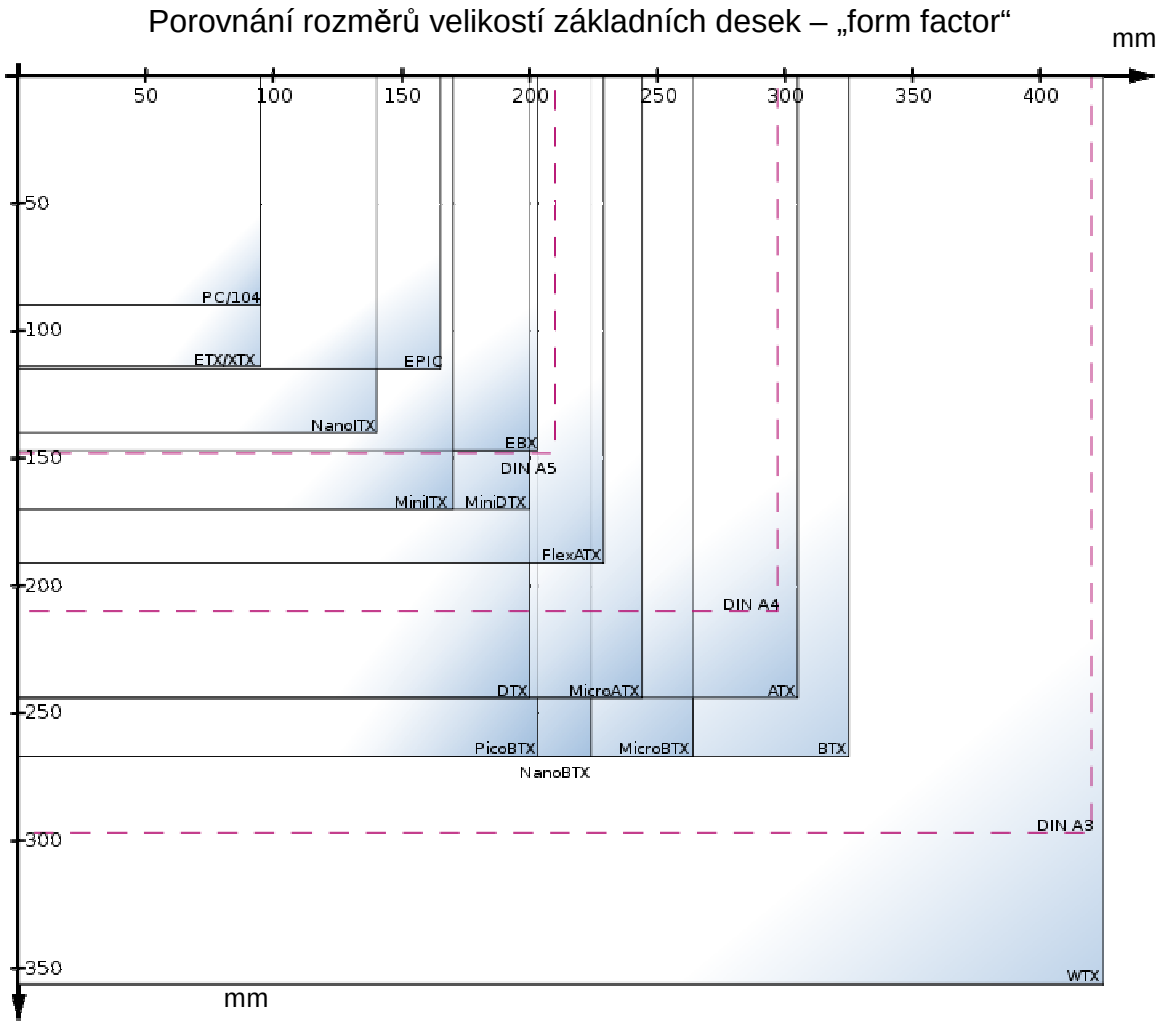
Zodpovědně musíme přistupovat k výběru pevného disku. V současnosti je jich velký počat druhů a mohlo by se zdát, že na výběru až tolik nezáleží. To bychom museli vybírat jen podle ceny pevného disku, pak by výběr byl velice snadný. V tomto případě se však nejedná o běžný kancelářský počítač, ale o serverovou aplikaci.

6.2 Server s platformou Via Epia

Jak již bylo zmíněno na úvod kapitoly, vhodným kandidátem jsou základní desky formátu ITX. Firma VIA Technologies prosazuje formát nejmenších základních desek typu mini-ITX s rozměry 170x170 mm, nano-ITX 120x120 mm, pico-ITX 100x72 mm a mobile-ITX 75x45 mm. Tyto desky stačí většinou osadit pouze operační pamětí a pevným diskem, čímž se z nich stane plnohodnotný počítač.

Výhodou je, že se dají zpravidla uchladit pasivně díky nízké spotřebě a malému vyzařovanému teplu. Na těchto deskách jsou kromě chipsetu integrovány i procesor

a grafická karta. Desky mini-ITX navíc obsahují ještě jeden rozšiřující slot PCI, některé dokonce nemají procesor integrovaný, ale obsahují 479pinový socket pro mobilní (notebookové) procesory. Nejmenší z řady ITX, mobile-ITX, má integrovánu i operační paměť a je konkurentem pro UMPC. Mohlo by se zdát, že tato platforma bude pro naše použití optimální. Je však potřeba vzít v úvahu ještě cenu celé sestavy. Nejedná se jen o základní desku, je potřeba navrhnout celou sestavu ze vším všudy.



Obr. 6.1 – „form factor“ – standardizované rozměry základních desek

Z grafu na obrázku 6.1 je patrné, že formát mini-ITX není zdaleka nejmenší standart pro základní desky. Hardware pro aplikaci serverové částí tohoto softwaru by mohl být postaven i na některé menší platformě, ale u tohoto ji hraje velkou roli cena komponent a jejich dostupnost. Zatímco Platformy Via-Epia formátu mini-ITX jsou zcela běžně dostupné na domácím trhu, jiné platformy menších rozměrů jsou vcelku obtížně k sehnání a nebo se jedná o desky určené již pro konkrétní aplikace bez možnosti jakékoli úpravy nebo rozšíření. Server postavený na této již nemusí být jednoúčelový. Výkon procesorů u platforem Via-Epia je celkem vysoký a je možné, aby na serveru kromě serverové části aplikace vzdáleného ovládání byly spuštěny jakékoli

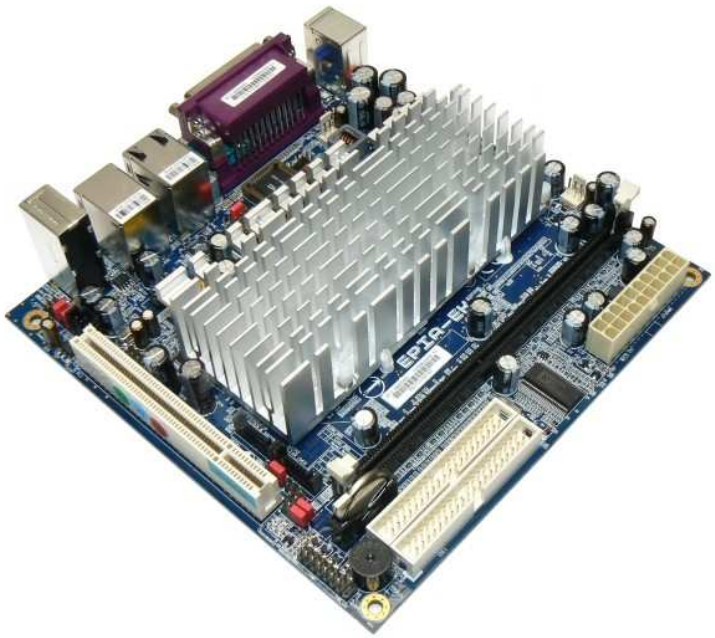
jiné služby. Mohlo by se jednat o ftp server, zálohovací systém, ukládání z logování provozu sítě a podobně.

6.2.1 Základní deska

Ať vybíráte jakoukoli komponentu, nemusí se jednat zrovna o základní desku, ale i o cokoli jiného. Každý výrobce nabízí velké množství modelů a někdy jsou rozdíly obtížně rozpoznatelné. Obdobně je tomu i u platforem Via-Epia, těchto základních desek je též celá řada. Nároky pro vybíraný server nejsou nikterak striktní a náročné, základní rozhodující faktory jsou:

- formát mini-ITX
- nízká spotřeba elektrické energie
- pasivní chlazení
- 2x port USB
- 1x Ethernet channel
- VGA výstup
- snadná dostupnost ostatních nutných komponent

Pro nasazení serverové části aplikace není potřeba, aby základní deska měla nějakou zvláště vysokou výbavu. Dokonce by to mohlo být nežádoucí. V porovnání všech pro a proti jednotlivých nabízených modelů vychází nejrozumněji základní deska s označením VIA EPIA EK8000EG [19]. Napájení základní desky je řešeno pomocí běžného formátu ATX, je potřeba, aby i skříň měla napájecí zdroj s takovýmto výstupem.



Obr 6.2 – Základní deska VIA EPIA EK8000EG

VIA EPIA EK-Series Mini-ITX Board Specifications	
Model Name	VIA EPIA EK8000EG
Processor	1.0GHz VIA Luke CoreFusion Processor 800MHz Fanless VIA Luke CoreFusion Processor
Chipset	VIA VT8237R Plus South Bridge
Onboard LAN	VIA VT6103L 10/100Mbps Ethernet PHY + VIA VT6107 10/100Mbps Ethernet controller
System Memory	1 x DDR 400 DIMM Socket, Up To 1GB Memory Size
VGA	Integrated VIA UniChrome™ Pro AGP graphics with MPEG-2/4 decoding acceleration
Expansion Slots	1 x PCI Slot
Onboard IDE	2 x UltraDMA 133/100/66 Pin Connectors
Onboard Audio	VIA VT1618 AC'97 Codec with 6-channel Support
Onboard I/O Connectors	3 x Serial port connectors for COM2/3/4 (5V/12V selectable) 1 x Digital I/O connector 1 x WP connector for BIOS flash 1 x SMBus connector 1 x LVDS/DVI/TTL module connector, for 18/24-bit dual channel LVDS panel 1 x CD audio-in connector 1 x CIR connector (switchable for KB/MS) 1 x Front panel connector 2 x Fan connectors for CPU and system fans 1 x ATX power connector
Back Panel I/O	1 x PS2 mouse port 1 x PS2 keyboard port 1 x Parallel port 1 x VGA port 1 x Serial port 2 x RJ-45 LAN ports 4 x USB 2.0 ports 3 x Audio jacks: Line-out, Line-in and Mic-in (vertical, Smart 5.1 support)
BIOS	Award BIOS, LPC 4/8Mbit flash memory
Operating System	Windows 2000/XP, Linux, Win CE, XPe
Software Application	VIA FliteDeck™ Suite • FlashPort-Live BIOS Flash • MissionControl-H/W Monitoring Remote SNMP Management • SysProbe-Live DMI Browser
System Monitoring & Management	CPU temperature reading CPU voltage monitoring Wake-on-LAN, Keyboard-Power-on, Timer-Power-on, Watch Dog Timer Fan control System power management AC power failure recovery
Operating Temperature	0 ~ 50 ° C
Operating Humidity	0% ~ 95% (relative humidity; non-condensing)
Form Factor	Mini-ITX (6- layer), 17 cm x 17 cm

Tab. 6.1 – parametry základní desky VIA EPIA EK8000EG

6.2.2 Operační paměť

Aby byl server funkční jako celek, jen samotná základní deska nestačí. Vybraný model obsahuje navíc akorát integrovaný procesor, ale to je vše. Pro kompletní sestavu je nutné doplnit operační paměť, pevný disk a skříň včetně napájecího zdroje.

Výběr operační paměti usnadňuje výrobcem uvedené specifikace použitelného paměťového modulu. Jedná se o modul DDR SDRAM PC3200 400MHz. Ačkoli se jedná v současnosti o starší typ pamětí, s jeho dostupností není prozatím problém. V současnosti jsou nejvíce rozšířené moduly s typovým označením DDR2. Výhody novějších typů pamětí jsou v rychlosti sběrnice a možné vyšší kapacitě. Ani jedno z těchto hledisek pro nás není v této aplikaci rozhodující. Jelikož i stabilita systému je z části pamětí, je zvolen modul od renomovaného výrobce.

Pro server jsem zvolil paměť od výrobce Kingston, produktové číslo KVR400X64C3A/1G, jedná se o model o kapacitě 1GB DDR 400MHz PC3200 CL3 128Mx64 BOX. Paměť je dodávána v boxovaném balení a výrobce uvádí doživotní záruku [24]. Paměť se může chlubit certifikací výrobců základních desek a je vhodná i pro menší servery.

Typ	SDRAM DDR
Kapacita	1GB
Certifikace	PC 3200
Max. pracovní frekvence	400MHz
CAS latency (CL)	3

Tab. 6.2 – Specifikace operační paměti

Obr. 6.3 – Paměťový modul Kingston KVR400X64C3A/1G

6.2.3 Pevný disk

Další komponentou, která nesmí ve funkční sestavě chybět je pevný disk. Na trhu je velké množství typů od různých výrobců. Zvolit nejvhodnější pro sestavu je o něco náročnější než tomu bylo u paměťového modulu. Největší dilema je mezi tím, zda zvolit klasický pevný disk s mechanickými částmi nebo některý z pevných disků SSD, které žádnou mechanikou část nemají. Základní deska nám umožňuje dle tabulky 6.1 připojit jakýkoli disk, který má rozhraní IDE UltraDMA 133/100/66. To nám opět trochu zužuje výběr. Nejdůležitějším parametrem je potřebná kapacita pevného disku. U této sestavy musí být dostatečně dimenzovaná pouze na použitý operační systém, serverovou část navrhnutého softwaru a případně některé další spuštěné aplikace. Je patrné, že největší část zabere operační systém. Pokud vezmeme v úvahu operační systém Microsoft Windows XP Professional, tak včetně ovladačů, potřebných součástí a instalované aplikace vzdáleného ovládání je potřebná kapacita cca 10GB. Jelikož tento server nebude sloužit jako úložiště dat, pořizovat pevný disk s několikanásobně větší kapacitou je zbytečné.

Pro porovnání vezmeme jeden disk klasického mechanického provedení, ale o velikosti 2,5“ . Menší provedení o velikosti 2.5" je voleno s ohledem na skříně pro základní desky typu ITX. Počítačové skříně typu ITX mají jednak menší rozměry pro instalaci komponent, ale také slabší napájecí zdroje. Klasický pevný disk o velikosti 3,5“ je nevhodný i z důvodu vyšší spotřeby elektrické energie a tím pádem i zátěže napájecího zdroje. Dále pak z důvodu, že tyto pevné disky jsou původem určeny pro přenosné počítače a tím pádem mají vyšší odolnost [20]. Pro srovnání jsem vybral model od výrobce Seagate model Momentus 2.5" 5400.3 40GB, 8MB cache, 5400ot, PMR. Typové označení ST94811A.

Obr. 6.4 – pevný disk Seagate ST94011A

Pevný disk Seagate Momentus 2.5" ST94011A - drive specification	
Formatted Gbytes (512 bytes/sector)	40
Guaranteed sectors	78,140,160
Bytes per sector	512
Physical read/write heads	2
Discs	1
Cache (Mbytes)	2
Recording density, BPI (bits/inch max)	642,000
Track density. TPI (tracks/inch max)	100,780
Areal density (Mbits/inch2 max)	65
Spindle speed (RPM)	5,400
Internal data transfer rate OD (Mbytes/sec max)	48.25
Sustained data transfer rate OD (Mbytes/sec)	34.5
I/O data-transfer rate (Mbytes/sec max)	100
ATA data-transfer modes supported	PIO modes 0–4; Multiword DMA modes 0–2; Ultra DMA modes 0–5
Height	9.5 +/-0.2 mm (0.374 +/-.0078 inches)
Width	69.85 +/-0.25 mm (2.75 +/-0.0098 inches)
Length	100.2 +/-0.25 mm (3.945 +/-0.0098 inches)
Weight (typical)	99 grams (0.218 lb.)
Average latency (msec)	5.6
Power-on to ready (sec typical)	3.9
Standby to ready (sec typical)	2.6 sec
Startup current 5V (peak)	1.2 amps
Track-to-track seek time (msec typical)	1.5 (read), 2.0 (write)
Average seek time (msec typical)	12.0
Average seek, read (msec typical)	12.0
Average seek, write (msec typical)	14.0
Full-stroke seek (msec)	22 (typical); 26 (max)
Seek power (typical)	2.4 watts
Read/write power (typical)	2.4 watts
Idle mode (typical)	1.2 watts
Standby mode	0.36 watts (typical)**
Sleep mode	0.36 watts (typical)**
Voltage tolerance (including noise)	5V ± 5%
Ambient temperature	5°to 55℃ (operating)–40°to 6 5℃ (nonoperating)
Temperature gradient (℃ per hour max)	20℃ (operat ing)30℃ (nonoperating)
Relative humidity (noncondensing)	5% to 90% (operating)5% to 95% (nonoperating)
Vibration, operating (max displacement may apply below 10 hz)	1.0 Gs (0 to peak, 5–500 Hz)
Vibration, nonoperating (max displacement may apply below 22 hz)	5.0 Gs (0 to peak, 5–500 Hz)
Mean time between failures (power-on hours)	330,000 at 25℃, Max case temperature: 60℃

Tab. 6.3 – parametry pevného disku Seagate ST94011A

Jako druhého kandidáta pevný disk typu SSD bez mechanických částí. Výhody tohoto disku je v mechanické odolnosti, nehlučnosti a spotřebě elektrické energie. Co se týče parametrů kapacity a ceny, tak je vždy výhodnější klasický pevný disk s mechanickými částmi. V současné době jsou pevné disky tohoto typu spíše novinkou a do podvědomí veřejnosti se spíše stále dostávají.

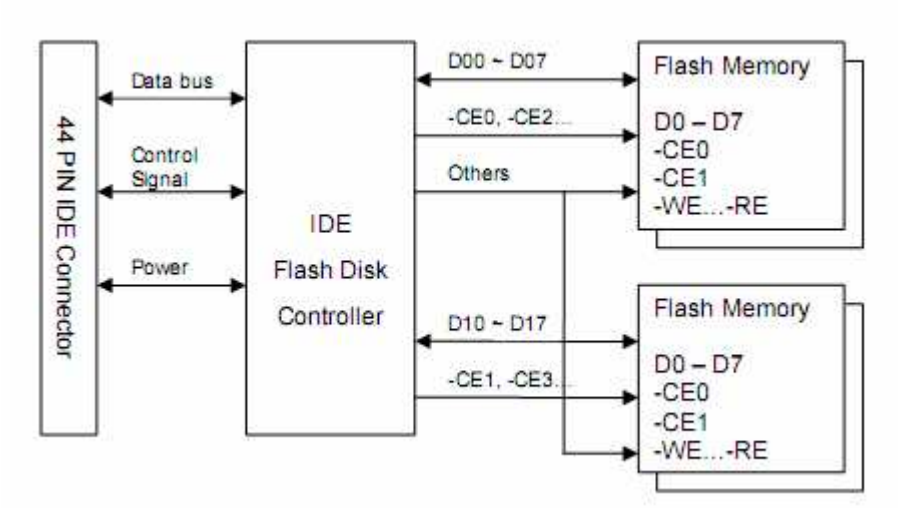
Zkratka SSD (**Solid-state drive)** je v informačních technologiích typ datového média, která používá pro ukládání dat flash paměť [15]. SSD disk emuluje pevný disk tak, aby ho bylo místo něj možné snadno použít. SSD disky však trpí i mnoha problémy, které jsou dány jejich konstrukcí. Flash paměti mají omezenou životnost maximálním počtem zápisů do stejného místa, který je výrazně nižší, než u klasických pevných disků. Operační systémy k nim obvykle přistupují díky kompatibilitě jako k normálním pevným diskům a tak dochází k degradaci jejich výkonu. Ukazuje se, že SSD disky nejsou v žádném případě vhodné pro všechny aplikace. Jejich použití je potřeba pečlivě zvážit. V případě navrhované sestavy nehrají zápory těchto disků velikou roli a mohli bychom je bez obav použít.

Pokud po sestavě serveru požadujeme v co největší míře bezúdržbovost systému a mechanickou odolnost vůči prostředí, neváhal bych tento druh disku použít. Podmínkou je, že na serveru poběží pouze aplikace vzdáleného ovládání a pro žádné jiné služby se server nebude využívat. Pokud by zbytečně docházelo k přístupu na systému disk a tím ke zpomalení celého systému, není aplikace tohoto disku vhodná. Pro sestavu jsem zvolil pevný disk od výrobce TRANSCEND 32GB SSD disk 2.5" IDE PATA. Rychlost přenosu dat při čtení maximálně 26MB/s a při zápisu maximálně 8MB/s (při UDMA Mode4). Více parametrů naleznete v tabulce 6.4.

Obr 6.5 – pevný disk SSD Transcend TS32GSSD25-M

Pevný disk Transcend TS32GSSD25-M - drive specification	
Form Factor	2.5-inch HDD
Storage Capacities	32 GB
Length (mm)	100.0 0 ± 0.40
Height	7.40 ± 0.15
Input Voltage	5V ± 5%
Weight	80 g
Connector	44-Pin standard IDE/ATA connector (Pitch 2.0 mm)
Operating Temperature	0 □ to 70 □
Storage Temperature	- 40 □ to 85 □
Power Consumption Write	78.0 (mA)
Power Consumption Read	57.7 (mA)
Power Consumption Standby	1.5 (mA)
Data Reliability	Built-in 4 symbol/page correction ECC
Data Retention	10 years
Connector Durability	10,000 times
MTBF	3,600,000 hours
ATA Compatibility	ATA/ATAPI 4, PIO Modes 0 – 4, UDMA Modes 0 - 4
Compliance	CE, FCC and BSMI
Operating Humidity (Non condensation)	5% to 95%
Storage Humidity (Non condensation)	5% to 95%

Tab. 6.4 – parametry pevného disku Transcend TS32GSSD25-M



Obr 6.6 – blokové schéma SSD disku

6.2.4 Skříň platformy mini-ITX

Poslední komponentou, která chybí u návrhu serveru je skříň, ve které bude instalován. Vybraná skříň musí být pro vybranou základní desku, která je formátu mini-ITX. Dále musí obsahovat minimálně jednu pozici pro pevný disk o rozměru 2,5“. Je třeba dát pozor na napájecí zdroj, který je součástí. Vybraná základní deska má na připojení napájení běžný ATX konektor. Napájecí zdroje, které bývají součástí mini-ITX skříní se mohou v tomto ohledu velice lišit. Převážná většina skříní je určená vždy přímo pro jeden typ základní desky. Při výběru skříně pro server je potřeba vzít v úvahu ještě jedno hledisko, které vychází z místa, kde bude server umístěn. Skříně lze rozdělit na dva základní typy a to typu desktop nebo rack.

Rackové technologie jsou určeny převážně do oblasti datových sítí. Vzhledem k tomu, že zařazení vyvíjeného softwaru klient-server je jako jedno z možných použití určen pro vzdálené ovládání rádiových zařízení, tak i navrhovaný server nepochybně do této oblasti patří. Rackové skříně jsou praktičtější a mají spoustu výhod, ale jen v případě, že budou umístěné v datovém rozvaděči. Pokud tomu tak není, skříň typu rack nemá žádný význam. Nehledě na to, že skříně typu rack jsou vždy dražší než klasické desktopové. Rackovým skříním bývá ještě často vytýkáno, že počítače v nich umístěné se hůře chladí. Z tohoto důvodu v nich bývají instalované silné ventilátory o malém průměru, které jsou velice hlučné. Rackový systém není v žádném případě vhodný do prostor, kde je nežádoucí hluk. Aplikace serveru je určena právě do prostředí datových sítí a je veliký předpoklad, že veškeré technologie bude potřeba umístit do datových rozvaděčů. Prostředí, ve kterých se poté bude spíše průmyslového charakteru, proto ani hluk z rackových počítačů nevadí. Pro server je zvolena skříň EMCO EM-162 o rozměru 19“ 1,25U [23]. V této skříni je i dostatek místa pro ovládací prvek s jednočipovým mikroprocesorem, pokud by bylo vhodné, aby byl přímo fyzicky součásti serveru.

Obr 6.7 – skříň rack 19“ 1,25U pro platformu mini-ITX, pohled zhora zepředu (nahoře) a na zadní panel (dole)

Skříň EMCO EM-162 - parametry	
Rozměry - rack	1,25U / 19"
Rozměry (š x v x h)	482 x 53 x 200 mm
Barva	černý čelní panel
Externí pozice	1x slim 5.25"
Interní pozice	1x 2,5"
Formát základní desky	170 x 170mm mini-ITX
Chlazení	2x 40x40x10mm (volitelné)
PCI slot	1x
Rozšiřující karta:	1x32 Bit 1 slot PCI (volitelná)
Čelní panel	- Power tlačítko - Power LED - HDD LED
Napájecí zdroj	AC/DC 90W ATX bez ventilátoru
Hmotnost čistá	3 kg
Hmotnost hrubá	3,6 kg

Tab. 6.5 – parametry rack skříně Emco EM-162

6.2.5 Finální sestava

Celá šestá kapitola byla věnována návrhu hardwaru pro server, jehož základem je mini-ITX platforma Via – Epia. Sestava s touto platformou je dostatečně výkonná a proto nabízí daleko širší využití, než jen jako stroj pro serverovou část aplikace vzdáleného ovládání. Z tohoto důvodu i finálně navržená sestava na této platformě není navrhnuta jen jako jednoúčelová. V návrhu je brán ohled na co nejmenší náročnost na údržbu, co nejnižší spotřebu elektrické energie a přijatelné rozměry pro implementaci. V návrhu jsem se příliš doposud nezabýval cenou jednotlivých komponent. Ceny jsou uvedeny včetně DPH a jsou pouze orientační.

Hardware pro server		Cena (Kč)
Základní deska:	VIA EPIA EK8000EG mini-ITX	4430 ,-
Operační paměť:	Kingston KVR400X64C3A/1G	915,-
Pevný disk:	Seagate Momentus ST94011A	1130,-
Skříň:	EMCO EM-162	3920,-
Cena celkem:		10395,-
Navýšení ceny při použití SSD pevného disku:		+ 2480,-

Tab. 6.6 – cena serverové sestavy

Z cenového navýšení při použití SSD pevného disku místo klasického je patrné, že se nejedná o zanedbatelnou položku. Použití SSD disku je proto velice zvážit. SSD disky si zajisté naleznou své specifické použití, ale než budou schopni nahradit doposud používané klasické mechanické, bude ještě nějakou dobu trvat.

6.3 Server s platformou Alix Wrap

I když se může zdát server postavený na platformě Via-Epia jako ideální, ve skutečnosti tomu tak být nemusí. Předchozí návrh je sice vynikající co se týče poměru cena výkon, ale jen pokud by byl využit i k jiným účelům, než jen provozu aplikace vzdáleného ovládání. Opět tedy trochu záleží na možném využití přímo v místě instalace. Předchozí sestava je navrhnuta spíše jako víceúčelová. Platforma, o které se zmíním v této kapitole, bude využita pouze jako jednoúčelová. Bude sloužit jen pro provozování aplikace vzdáleného ovládání. U této platformy nemáme možnost takového výběru dodatečných komponent, jako u Via-Epia. Platformy Alix Wrap jsou určeny pro vytváření počítačů, které mají průmyslové využití, zejména v automatizaci [21]. Setkat se s nimi můžeme například i u CNC strojů. Využití je opravdu

6.3.1 Základní deska

Je možná trochu neobjektivní uvádět u této varianty pojem základní deska. Nejedná se o samostatnou základní desku, ale již o celou platformu, na které jsou všechny potřebné součástí integrovány. Nevýhodou může být, že v tomto případě není moc možností v případě potřeby o rozšíření. Návrh je však určen jako jednoúčelový pro konkrétní nasazení a v tomto případě není podle mého názoru brát ohledy na možnosti rozšíření. Jedná se o výrobek firmy PC Engines GmbH. Konkrétně o model Alix 1d LX800.

Platforma je postavena na čipsetu AMD, rovněž procesor je od stejného výrobce. Z dalších potřebných součástí pro použití deska obsahuje vlastní operační paměť, VGA výstup, USB port, LAN kontrolér a další součásti, které jsou pro plánované využití nemají opodstatnění. Formát desky je mini-ITX low profile. Jedna z velkých výhod použití této platformy je napájení. Narozdíl od platformy Via-Epia je zde řešeno konektorem jack 3.5“. Deska je tedy napájena adaptérem a nepotřebuje žádný napájecí zdroj typu ATX a podobně. Tato výhoda se projeví jak při samotné montáži zařízení, protože skříň je daleko menších rozměrů, tak na spotřebě zařízení, které je dáno pouze spotřebou použitého napájecího adaptéru.

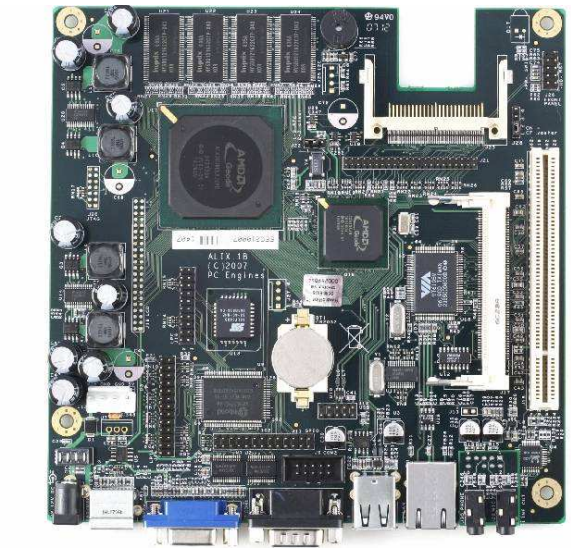
S komptabilitou této platformy, jak by se mohlo na první pohled zdát není též problém. Stále se jedná o systém typu x86. Lze proto na ní instalovat téměř jakýkoli operační systém. Jediné omezení se kterým se setkáme je ve výkonu procesoru a velikosti operační paměti. Bohužel v tomto případě je dosti striktní. V současné době již není mnoho operačních systémů, kterým stačí pro běh 256MB operační paměti a procesor o frekvenci 500MHz.

Jelikož je sestava s touto platformou navrhnuta jako jednoúčelová pro konkrétní použití, může být použit i operační systém bez jakýchkoli dalších doplňků, navržený též na míru pro běh na této platformě. Vhodným kandidátem pro tuto platformu je operační systém založený na některém linuxovém jádře. Na rozdíl od systému Microsoft Windows, jakýkoli druh operačního systému založeném na linuxovém jádře nám

umožňuje nainstalovat jen a pouze součásti, které jsou nutné pro běh. Zkompilovat navrženou instalaci přímo pro použitý hardware a nainstalovat opravdu jen potřebné součásti. Ušetří se tím náročnost systému na hardware. Upravený a přizpůsobený systém pro danou platformu se nám odvděčí ještě vyšší stabilitou celého systému a i vyšším výkonem celé sestavy. U serverů je toto té velice žádoucí.

6.3.2 Základní rysy platformy Alix 1D LX800

• AMD Geode LX CPU, 500 MHz (LX800) 5x86 CPU,
• 256 KB cache (64K data + 64K instruction + 128K L2)
• 1 Ethernet channel (Via VT6105M)
• 1 miniPCI + 1 PCI socket (3.3V) for 802.11 wireless cards and other expansion
• 256 MB DDR SDRAM, 64 bit wide for high memory bandwidth
• 512 KB flash for Award system BIOS.
• CompactFlash + 44 pin IDE header for user's operating system and application
• 12V DC supply through DC jack
• 2 serial port (DB9 male + 10 pin header)
• 1 parallel port (26 pin header)
• Combined PS/2 keyboard + mouse port
• VGA port
• 4 USB 2.0 ports (2 on rear panel connector + 2 on 10 pin header)
• AC97 audio codec (line in / line out on board, headphone + mic on 10 pin header)
• Header for LPC bus (use for flash recovery or I/O expansion)
• GPIO header for user expansion
• Optional header for TFT LCD interface, I2C bus
• Front panel header for power switch, reset, hard disk and power LED
• Buzzer for "beeps"
• Socketed RTC battery
•Power supply +12V DC, ~ 0.007A off state, typical about 0.4A active
•Peak power can be higher, suggest a 15W supply.
•Center pin = positive, sleeve = ground, 2.5 mm diameter.
•Temperature range 0 to 50°C.
•Dimensions ALIX.1C = 6.7 x 6.7" (170 x 170 mm)



Obr. 6.8 – deska Alix 1D LX800

6.3.3 Pevný disk pro Alix 1D LX800

Mohlo by se zdát, že požadavky na pevný disk budou pro tuto platformu obdobné jako u sestavy s deskou Via-Epia. Alix 1D obsahuje sice klasické IDE ATA-100 rozhraní, ale připojovat klasický pevný disk je pro toto použití zbytečné. Z hlediska kapacity pro plánované využití bez problémů stačí 4GB. Operační systém pro tuto platformu je navržen na míru a tím i s minimálními požadavky na kapacitu pevného disku. Klasický pevný disk s mechanickými částmi je sice z hlediska pořizovací ceny vhodným kandidátem, ale narazíme zde na problém s připojením napájení. Platforma je napájena pouze adaptérem a ten nemá patřičný výstup pro napájení pevného disku. Pro tuto platformu je zvolen jiný typ pevného disku. Nejedná se až tak o pevný disk v pravém slova smyslu, ale o Compact Flash kartu. Compact Flash karty nejsou původně určeny jako náhrada pevného disku, ale pro tuto sestava jsou ideálním kandidátem. Samotná deska Alix 1D LX800 obsahuje slot přímo pro připojení CF karty.

V principu se jedná zpravidla o paměti typu RAM (Random Access Memory) s technologií Flash, která umožňuje uchovávat data v paměti i bez přítomnosti napájecího napětí. Kapacita těchto pamětí se pohybuje od 8MB do 4GB. Paměti typu Compact Flash (CF) mají poněkud větší nejenom rozměry, ale zejména pak tloušťku 3.3 mm. Obsahují přídavné elektronické obvody (řadič) pro usnadnění komunikace s jiným externím zařízením a také speciální konektor. Právě pomocí speciálního dvouřadého konektoru se pak karta propojuje s okolním zařízením. Jediným omezením i když v tomto případě ne nějak zásadním je maximální kapacita těchto karet a jejich rychlost. Operační systémy, které jsou vhodné pro použití na tomto médiu fungují tak, že po načtení běží v operační paměti a pevný disk vůbec nevyužívají. Pro sestavu je volena karta od výrobce Kingston, z důvodu spolehlivosti a záruky výrobce [24].

Obr 6.9 – CompactFlash karta Kingston

Základní parametry Compact Flash karty:

- Capacities — 2GB, 4GB
- Dimensions — 1.43" x 1.68" x 0.13" (36.4mm x 42.8mm x 3.3mm) - CF Type I
- Operating Temperature — 0^o to 60^o C / 32^o to 140^o F
- Storage Temperature — -20^o to 85^o C / -4^o to 185^o F
- Voltage — 3.3v / 5v
- Standardized — complies with CompactFlash Association specification standards
- Small — one-third the size of a full-size PC card
- Easy — plug-and-play
- Guaranteed — lifetime warranty
- Versatile — compatible with PC Card Type II adapters
- Economical — autosleep mode preserves system battery life

Aby bylo možné CompactFlash kartu využívat jako náhradu pevného disku, je potřeba jí připojit k základní desce pomocí slotu, který již platforma Alix 1D LX800 obsahuje. Karta je poté emulována a pro počítač se tváří, jakoby byl připojen klasický pevný disk. O některých nevýhodách jsem se zmiňoval v předchozí kapitole u SSD disků, CF karty jsou na tom úplně stejně. V této sestavě, která je však navrhnuta jednoúčelově a operační systém je navrhnutý přímo pro toto použití lze tyto nevýhody pominout. Toto řešení pevného disku je optimální pro jednoduchost a bezúdržbovost celého systému. Cena karty je táž velice příznivá a nedá se s náklady na klasický pevný disk vůbec srovnávat. Běžně se tohoto provedení náhrady pevného disku využívá i v případě jiných aplikací. Některé platformy mají dokonce již integrovanou paměť připravenou pro instalaci operačního systému s možností připojení externí CompactFlash karty. Příkladem mohou být platformy pro datově sítě od výrobce Mikrotik. Nejedná se tedy o žádnou žhavou novinku, jen se toto řešení moc doposud nedostalo do podvědomí veřejnosti. Není se ani čemu divit, protože oblast zaměření, kde tohoto lze efektivně využít je velice úzká.

6.3.4 Skříň pro Alix1D LX800

Jako každou sestavu i tuto je potřeba složit do uzavřené skříně. V tomto případě nám nezbývá nic jiného, než do návrhu zahrnout skříň dodávanou přímo výrobcem. Rozměry platformy nejsou standardní, podobají se sice platformě mini-ITX, ale kdo by zkoušel komptabilitu, byl by po chvíli zklamán. Na skříních dodávaných výrobcem není nic špatného, ale pokud by bylo potřeba do nich vestavět ještě některé další komponenty, tak to bohužel nelze. Ve skříni prostě není místo na cokoli dalšího.

Obr. 6.10, 6.11 – skříň pro Alix D1

6.3.5 Napájecí zdroj pro Alix 1D LX800

Vybraná platforma jak již bylo zmíněno v úvodu kapitoly nepotřebuje žádný speciální napájecí zdroj. Na napájecí vstup můžeme přivést stejnosměrné napětí od 12 V do 24V. Deska tedy obsahuje vlastní napěťový stabilizátor. Je třeba počítat s tím, že sestava bude při plném vytížení odebírat při 12A ze zdroje proud kolem 1A. Při použití nedostatečně dimenzovaného zdroje může dojít jak k poškození zdroje, tak základní desky. Napájecí adapter vhodný pro použití k desce Alix 1D LX800 je od výrobce Sunny s označením SUN18V/30W.

Tento vybraný model adaptéru je vhodný pro tuto navrhovanou sestavu. Není počítáno, že by deska byla osazena ještě některými dalšími součástmi. Pokud by byl využit ještě miniPCI slot některou kartou pro bezdrátovou síť wifi 802.11 a PCI slot s jakoukoli komponentou, bude nutné sestavu napájet silnějším adaptérem. Zvolený model má při 18V výstupní maximální proud 1.6 A. Pravděpodobně by to stačilo pro napájení sestavy bylo dostačující, ale adaptér by byl zatížen blízko u svého maxima, zbytečně by se přehříval a zkrátila by se mu doba jeho životnosti. Deska Alix 1D má ještě jednu výhodu a tou je možnost napájení formou pasivního POE pomocí LAN portu. Redukce nutná pro napájení pomocí LAN je vyobrazena na obrázku 6.12. Zda využít nebo nevyužít tuto možnost je na zvážení pro danou instalaci.

Obr. 6.12 – napájecí adaptér Sunny

Parametry zdroje SUN18V/30W	
Ostatní:	Zkrat, přepětí, proudové přetížení cUL, TUV, CB, CE, PSE, Nemko, CCC
Provozní teplota:	0 - 40 ℃
Rozměry:	103 x 52 x 34 mm
Hmotnost:	0,15 kg
Výstupní výkon:	30 W
Výstupní napětí:	18 V
Vstupní napětí:	100 - 240 V
Frekvence napětí:	50~60 Hz
Max. výstupní proud:	1,67 A

Tab 6.7 – parametry zdroje Sunny

Obr. 6.13 – pasivní POE redukce

6.3.6 Podporované operační systémy a shrnutí

Podporované operační systémy platformou Alix 1D	
FreeBSD	FreeBSD (on older versions, may need to disable USB 2.0 in BIOS, or disable the EHCI driver) Monowall (FreeBSD based) pfSense (FreeBSD based) STYX (FreeBSD based Secure Internet Appliances, in German)
Linux	iMedia ALIX Linux - highly recommended for a quick start - fits nicely on a 512MB CF card. DD-WRT gOS 3 Gadgets fli4l one disk router (in German) LEAF Linux Embedded Appliance Firewall Meshlium (wireless mesh system based on Voyage) OpenWRT (working with Kamikaze version) Voyage Linux (slimmed down Debian) Xubuntu Linux (customer reported success with 8.04). Zeroshell
NetBSD	NetBSD (versions 4, 5)
OpenBSD	OpenBSD (X11 and audio support in 4.3-current) Flashdist OpenBSD installer
Commercial	Ikarus OS Embed-it (OpenBSD based) Microsoft Windows XP Home Mikrotik RouterOS

Tab. 6.8 – podporované operační systémy platformou Alix 1D

Jak je z tabulky 6.8 patrné, i když se u Alix 1D jedná o nestandardní platformu, na podporovaných operačních systémech to znát není. Je možné použít jak komerční tak nekomerční operační systém a to se může příznivě projevit na celkové ceně sestavy včetně softwarového vybavení. Je překvapující, že i na této platformě lze provozovat komerční systém Microsoft Windows XP, který je určen úplně do jiného segmentu trhu [21].

Na závěr šesté kapitoly ještě uvedu cenový přehled navrhované sestavy s deskou Alix 1D LX800. Je dobré mít vždy při výběru možnost porovnání s platformou Via-Epia. Při výběru je potřeba brát v úvahu, že sestavu se základem Via-Epia je možné použít víceúčelově, kdežto od sestavy s deskou Alix 1D širší možnost využití očekávat nelze. Ceny jsou orientační a jsou uvedeny včetně DPH.

Hardware pro server		Cena (Kč)
Základní deska:	Alix 1D LX 800	3460 ,-
Pevný disk:	Compact Flash Kingston 4GB	360,-
Skříň:	PCengines pro Alix 1D	380,-
Napájecí adapter	Sunny 18V, 1.67A stejnosměrný	310,-
Cena celkem:		4510,-

Tab. 6.9 – cena serverové sestavy s Alix 1D LX800

7 Ovládací prvek s mikrokontrolérem

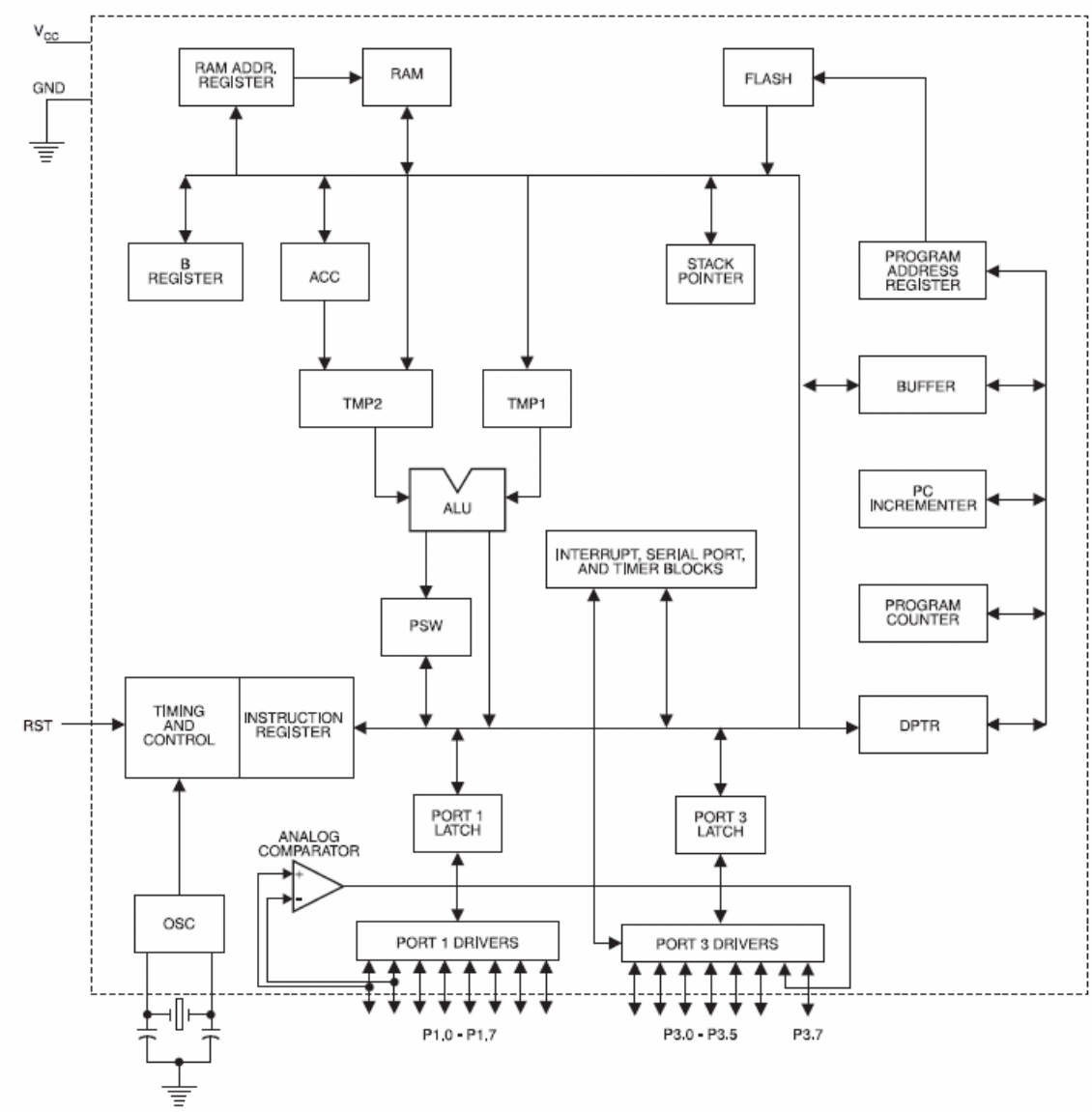
Pro využití navrhnutého softwaru pro vzdálené ovládání je nutná k celému systému ještě jedna komponenta. Jedná se o ovládací prvek, který je připojený k serveru a na který server odesílá přijaté příkazy. V ovládacím prvku je využit jednočipový mikroprocesor, na jehož vstupu je připojené rozhraní serveru. Řešení s mikroprocesorem je vhodné zejména z důvodu, že jeho software lze snadno upravit pro ovládání jednotlivých výstupů. To umožňuje značnou variabilitu využití systému vzdáleného ovládání jako celku. Není proto problémem implementace na míru dle konkrétních požadavků. Jedná se o mikroprocesor od výrobce Atmel AT89C2051.

7.1 Mikroprocesor AT89C2051

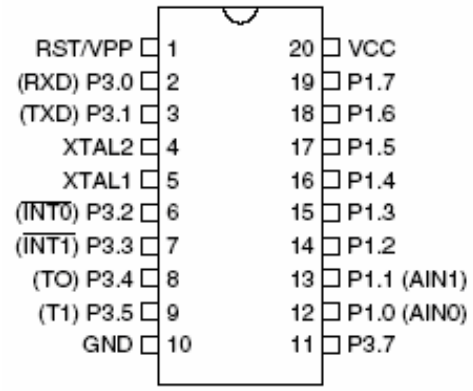
Mikroprocesor AT89C2051 je obdobou mikroprocesoru AT89C51, ale má omezený počet portů a je použito pouzdro DIL20. AT89C2051 je 8bitový jednočipový mikroprocesor s harvardskou strukturou, u které je oddělena programová a datová paměť. Procesor, jehož vnitřní struktura je blokově zobrazena na obr 2.2, je schopen samostatné činnosti po připojení vnějšího piezokeramického rezonátoru a jednoho napájecího napětí 5 V. Na čipu procesoru je umístěna vlastní procesorová jednotka CPU, která je vnitřní společnou sběrnicí propojena s pamětí programu ROM nebo EPROM o kapacitě 4 kB, s pamětí RAM o kapacitě 128 bytů a se dvěmi vstupně výstupními branami, které zajišťují styk procesoru s vnějšími periferiemi [22].

Pro snadnější styk s periferiemi je procesor vybaven řadičem přerušení, který zpracovává 5 zdrojů přerušení (2 externí, od každého ze dvou časovačů a od sériového kanálu). Jednotlivá přerušení mají definovanou prioritu na každé ze dvou úrovní priority. Čítače, které usnadňují realizaci časování, jsou 16bitové s hodinovým signálem odvozeným z interního generátoru hodin nebo z vnějších vstupů. Pro snazší sériový styk s nadřazeným počítači nebo jiným spolupracujícími procesory je vybaven duplexním (obousměrným) sériovým kanálem. Procesor může pracovat s jednotlivými bity vnitřní paměti RAM a interních periferií. Pro základní kmitočet 12 MHz trvají instrukce 1 μs nebo 2 μs s tím, že nejdelší jsou instrukce násobení a dělení, které trvají 4 μs.

Obdobné procesory vyrábí mnoho dalších výrobců. Procesory se od sebe liší v počtu portů, čítačů/časovačů, sériových kanálů (UART), v závislosti na tom pak i počtem vývodů pouzdra, provedením pouzdra, velikostí a typem paměti, vybaveností dalšími periferiemi jako je obvod hodin reálného času (RTC), analog.komparátor, sběrnice I2C, sběrnice USB nebo CAN, programovatelný WatchDog, A/D a D/A převodník atd. Existuje nepsaný standard ve značení verzí mikroprocesoru podle použitého technologického typu paměti programu (ROM). Verze bez jakékoliv vnitřní paměti ROM začíná číslem 80, verze s pamětí v provedení OTP číslem 83, verze s pamětí EPROM číslem 87 a verze s pamětí FLASH/EEPROM začíná číslem 89.



Obr. 7.1 - Blokové cháma AT89C2051



Obr 7.2 – zapojení vývodů AT89C2051

7.2 Komunikace serveru s mikroprocesorem

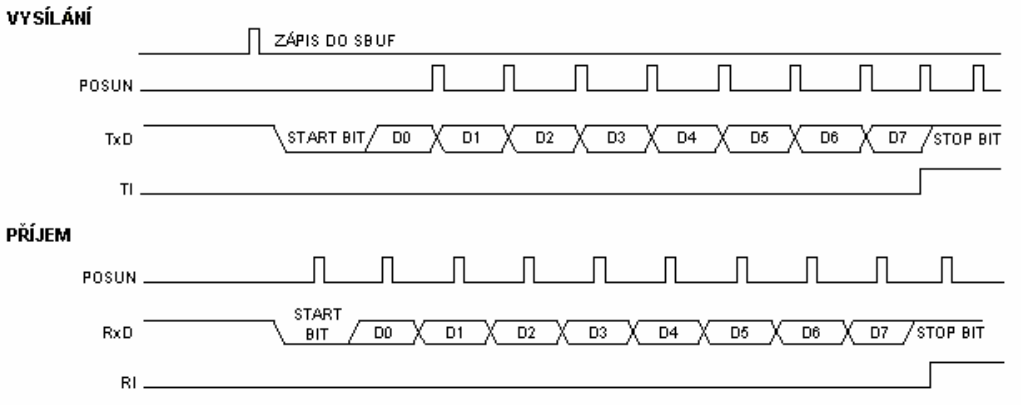
Mikroprocesor AT89C2051 obsahuje přímo vstup a výstup pro komunikaci pomocí sériové linky. Označení vývodů je P3.0 – Rx data a P3.1 – Tx data. Není problém této možnosti využít, ale pro propojení s počítačem není toto řešení vhodné. Většina současných počítačových sestav již standardně neobsahuje port pro sériovou linku, jak tomu bývalo dříve. Pro připojení periferie k počítači je nejvhodnější využít dnešní standard, který se nazývá USB(Universal Serial Bus). Abychom byli schopni propojit počítač, který obsahuje pouze USB port s jednočipovým mikroprocesorem, který obsahuje pouze vstup a výstup v podobě sériové linky, je nutné mezi ně zařadit převodník z USB na sériový port.

7.2.1 Sériová komunikace

Procesor obsahuje plně duplexní sériový kanál, který umožňuje komunikaci ve standardním 8 a 9bitovém asynchronním režimu nebo 8bitovém synchronním režimu s pevnou přenosovou rychlostí. Tím byla výrazně usnadněna komunikace s nadřízeným počítačem, např. typu PC, k jejich realizaci je v současné době zapotřebí jeden integrovaný obvod MAX232, MAX233 zajišťující převod úrovní TTL sériového kanálu na úrovně V24 (PC). Plně duplexní sériový kanál umožňuje současně vysílat i přijímat hodnoty po tomto kanálu, který tvoří minimálně tři vodiče (RxD, TxD a zem). Přijímací kanál je vybaven vyrovnávacím registrem, do kterého je uložena právě přijatá hodnota, čímž je umožněn okamžitý příjem další hodnoty. Převzatá hodnota musí být však převzata dříve, než je dokončen příjem následující hodnoty, který by způsobil přepsání původní hodnoty. Procesor není vybaven příznaky, které indikují ztrátu přijaté hodnoty, chybu rámce a parity nebo indikaci přerušení, které jsou obvykle u specializovaných obvodů [16].

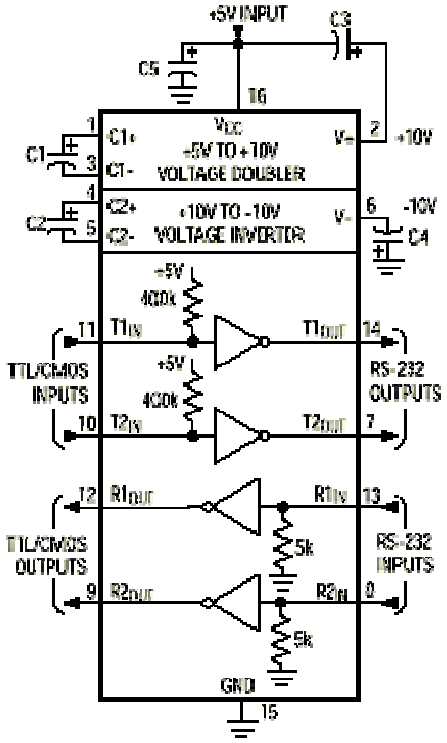
Sériový kanál může pracovat ve čtyřech módech v závislosti na naprogramování registru SCON a nejvyššího bitu v registru PCON. Pro připojení k převodníku USB / Serial bude mikroprocesor komunikovat po sériové lince v módu 1.

Mód 1 – 8bitový UART obr. 2.4. Hodnoty se vysílají výstupem TxD a přijímají vstupem RxD a skládají se z deseti intervalů, určených převrácenou hodnotou přenosové rychlosti v baudech pro přenos jednotlivých bitů. První bit je vždy nulový (log. 0) a představuje tzv. start bit, po němž následuje 8 přenesených bitů počínaje bitem s nejnižší váhou a posledním, který je vždy nastaven na 1 (log. 1) a představuje tzv. stop bit. Při příjmu se stop bit ukládá do bitu RB8 v registru SCON. Přenosová rychlost je volitelná a je určena periodou přetečení čítače/časovače 1 a hodnotou nejvyššího bitu v registru PCON. Pro přenosovou rychlost můžeme za předpokladu, že čítač 1 pracuje v módu 2, snadno odvodit vztah pro přenosovou rychlost.



Obr. 7.3 – stavy sériové linky v módu 1

U navrhovaného ovládacího prvku s jednočipovým mikroprocesorem je využit pouze výstup s označením P3.0 Rx data. Rychlost komunikace je zvolena na 9600 Bit/s. Přenosová rychlost je odvozena od frekvence oscilátoru, v případě tohoto mikroprocesoru od frekvence krystalu, který je připojen na vstupy XTAL1 a XTAL2. Krystal slouží pro mikroprocesor zároveň jako zdroj hodinového signálu. Abychom mohli využívat výše uvedenou rychlost sériové linky, musí být v návrhu zvolen krystal s optimální frekvencí, v tomto případě 11,059200 MHz.



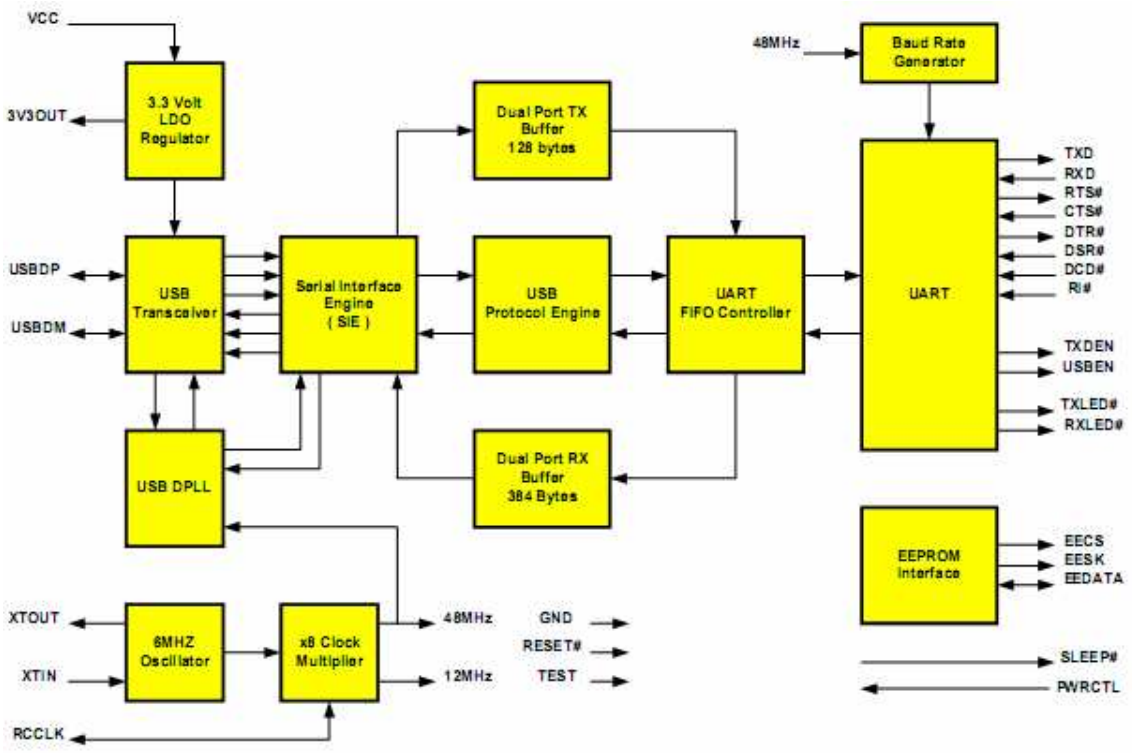
Obr. 7.4 – vnitřní zapojení obvodu MAX232

7.2.2 Převodník USB / Sériový port

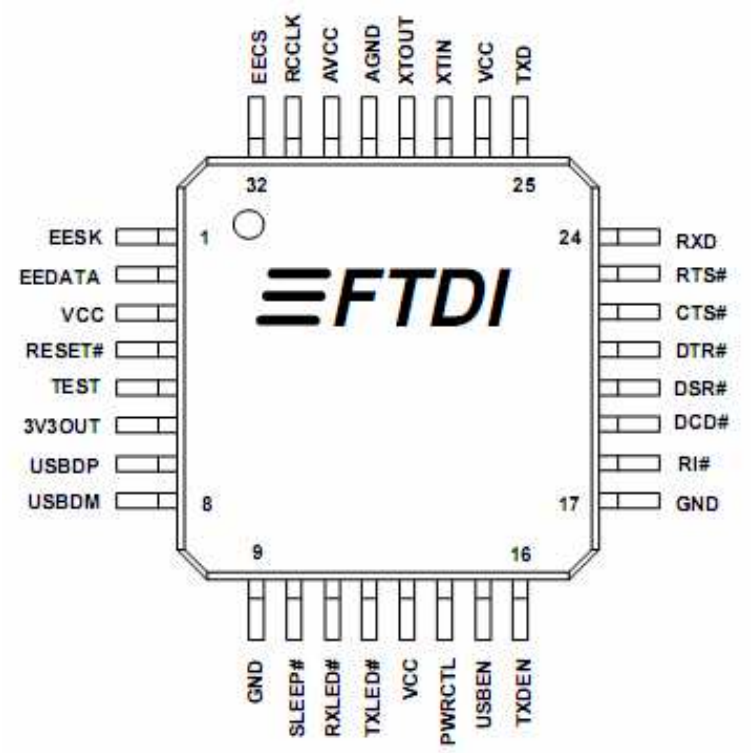
Aby bylo možné připojit sériovou linkou jednočipový mikroprocesor k serveru pomocí USB portu, je nutné použití převodníku mezi těmito sběrnicemi. Požadavky na převodník jsou následující:
- připojení k USB portu verze 1.1 i 2.0
- podpora ovladačů pro operační systémy Microsoft Windows i Linux
- sériový výstup kompatibilní s AT89C2051 a MAX232
- napájení převodníku z USB portu.

Mechanicky je převodník proveden jako kabel o délce 1 m zakončený na jedné straně konektorem USB typu A, který se zasune do příslušného portu v počítači a na straně druhé krytkou, která obsahuje elektroniku a ze které je vyveden 9pinový konektor Sub-D (Cannon) pro přímé připojení . Převodník se v systému chová stejně jako standardní sériový port. Lze tedy takovýto "virtuální" COM používat nejrůznějšími aplikacemi standardně přistupujících na COM v rozsahu komunikačních rychlostí od 300bps po 115.2kbps. Instalaci ovladačů je potřeba udělat jen jednou při prvním připojení konvertoru. Při připojení dalších konvertorů nebo při opakovaném připojení již systém instalaci ovladačů nevyžaduje. Pokud jsou USB porty funkční (povoleny v setupu počítače a správně nainstalovány v operačním systému), systém automaticky okamžitě pozná přítomnost nového zařízení a požádá Vás o nainstalovaní potřebných ovladačů. Nový port se nastaví automaticky jako nejbližší volný COM, avšak ve správci zařízení jej můžeme nastavit libovolně v rozmezí COM1 až COM256. Převodník je možné kdykoliv odpojit nebo připojit bez nutnosti restartovat systém.

Základem převodníku je integrovaný obvod s označením FTDI FT8U232BM. Jedná se o velice kompatibilní obvod [9]. Mezi jeho základní vlastnosti tohoto obvodu patří:
- vyrovnávací paměť pro příjem dat do PC 384 byte
- vyrovnávací paměť pro vysílání dat z PC 128 byte
- plně hardwarové řízení toku dat (RTS/CTS, DTR/DSR/DCD, RI)
- hardwarová podpora XON/XOFF
- protokol USB 1.1, USB 2.0 kompatibilní
- funkce režimu s nízkou spotřebou (USB suspend mode)
- napájení převodníku ze sběrnice USB, nepoužívá se žádné další napájení
- ovladače pro Linux release 2.40 jsou přímo součástí systému



Obr. 7.5 – blokové schéma FT8U232BM

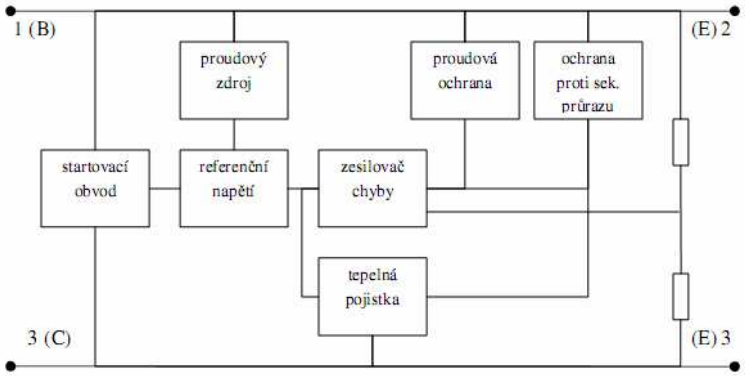


Obr 7.6 – označení vývodů obvodu FT8U232BM

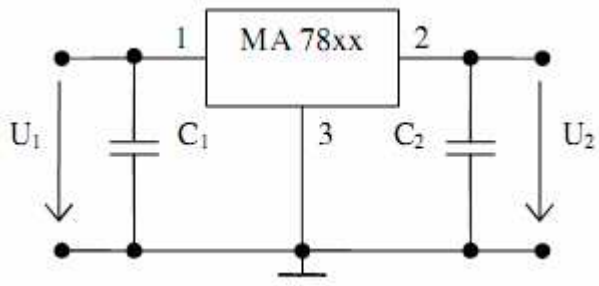
7.2.3 Návrh ovládacího prvku

Ovládací prvek slouží k vykonávání příkazů, které přijme se serverové stanice. Obsahuje jednočipový mikroprocesor z důvodu, aby bylo možné snadné přizpůsobené konkrétním podmínkám ovládání zvolené instalace. Pro nynější použití, které spočívá v možnosti vzdáleně restartovat rádiové částí bezdrátového převaděče, jsou použity 4 výstupy. Na těchto výstupech jsou připojeny relé jako spínač. Každý definovaný příkaz ze serveru vždy na 5s rozepne jedno relé. Modul se skládá z několika částí, viz. obr 2.3. Stabilizátor s diodou proti opačné polaritě a svítivou diodou, která signalizuje připojení napájecího napětí. Stabilizátor stabilizuje stejnosměrné napájecí napětí z 12 V na 5 V. Aby procesor mohl samovolně pracovat, musí k němu být připojeno napájecí napětí a krystal s dvěma kondenzátory. Pak už jen stačí tlačítko s kondenzátorem pro reset mikroprocesoru. Další vývody jsou už jen vstupy a výstupy. Obvod MAX 232 s několika kondensátory převádí TTL logiku z 5 V na 24 V.

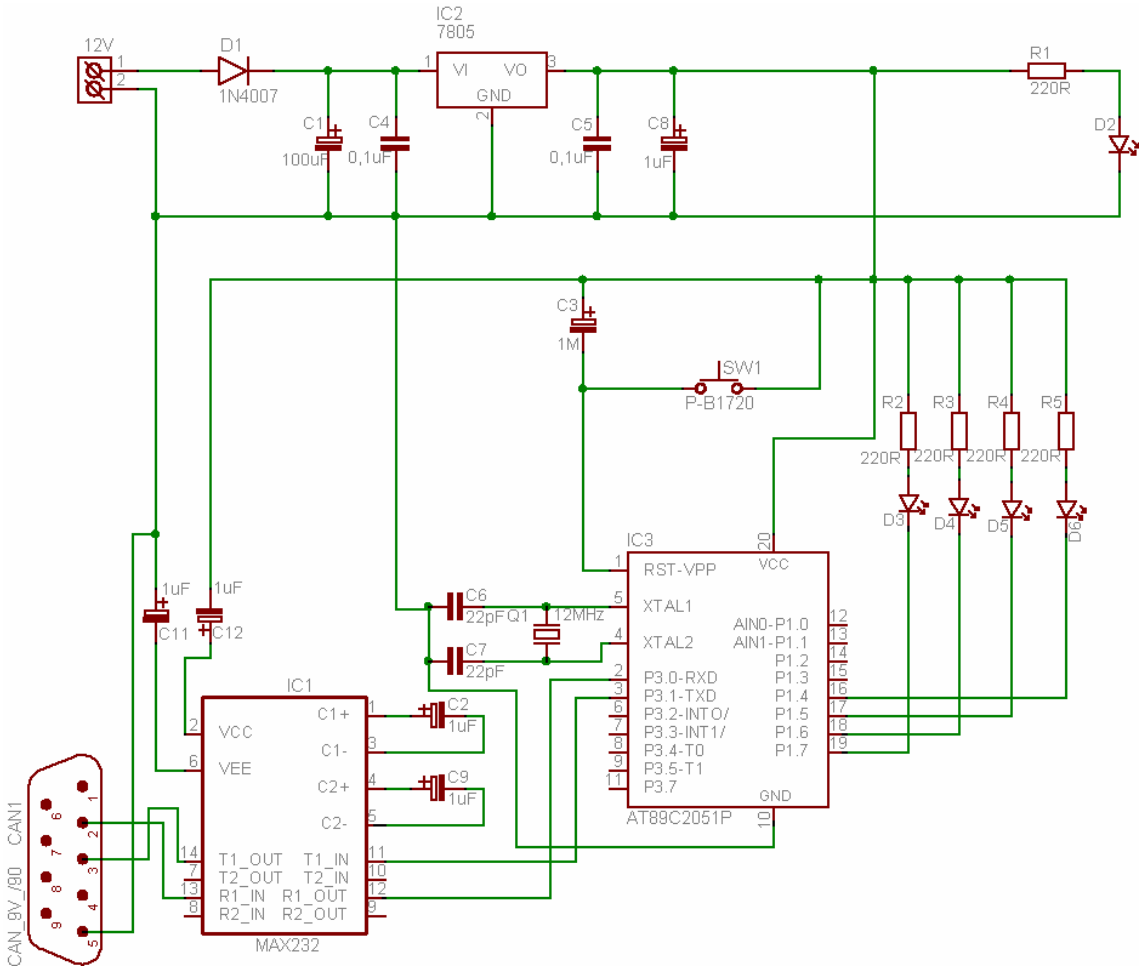
Pro ukázku funkčního modulu jsou výstupy jednočipového mikroprocesoru připojeny k LED diodám. Po přijetí příkazu se LED dioda rozsvítí po dobu 5s obdobně, jako by bylo rozepnuto relé. Modul je napájen stejnosměrným napětím, které může být v rozmezí 9 – 24V. Tato možnost je dána stabilizátorem napětí MAA 7805, který je součástí návrhu. Použití tohoto stabilizátoru napětí je velice vhodné, zapojení je jednoduché a obvod bez problémů splňuje parametry, které jsou potřeba pro napájení mikroprocesoru a ostatních součástí.



Obr 7.7 – Vnitřní blokové schéma MAA7805



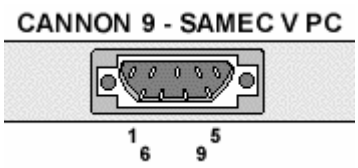
Obr. 7.8 – Schéma zapojení MAA7805



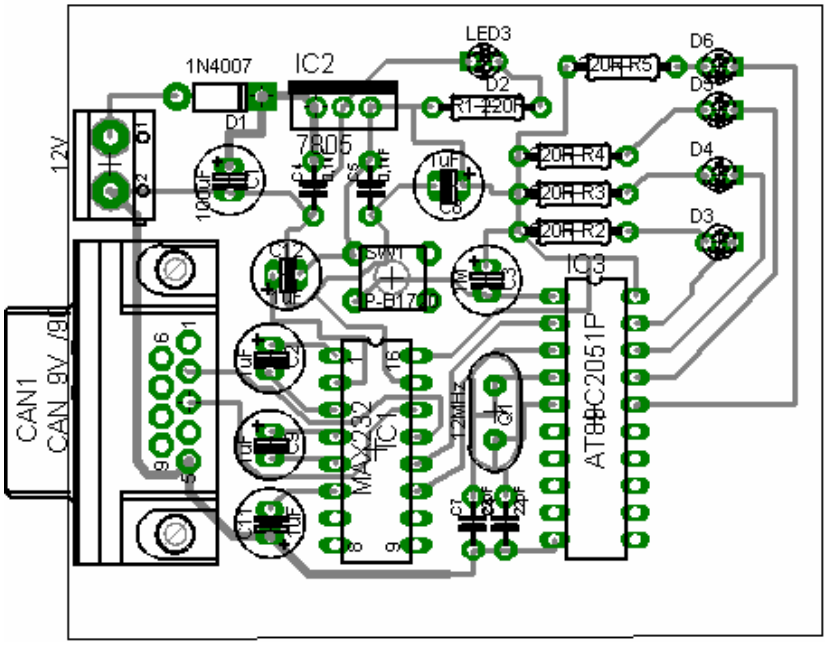
Obr. 7.9 – schéma zapojení ovládacího prvku s AT89C2051

Cannon 9			
PIN	NÁZEV	SMĚR	POPIS
1	CD	<--	Carrier Detect
2	RXD	<--	Receive Data
3	TXD	-->	Transmit Data
4	DTR	-->	Data Terminal Ready
5	GND	---	System Ground
6	DSR	<--	Data Set Ready
7	RTS	-->	Request to Send
8	CTS	<--	Clear to Send
9	RI	<--	Ring Indicator

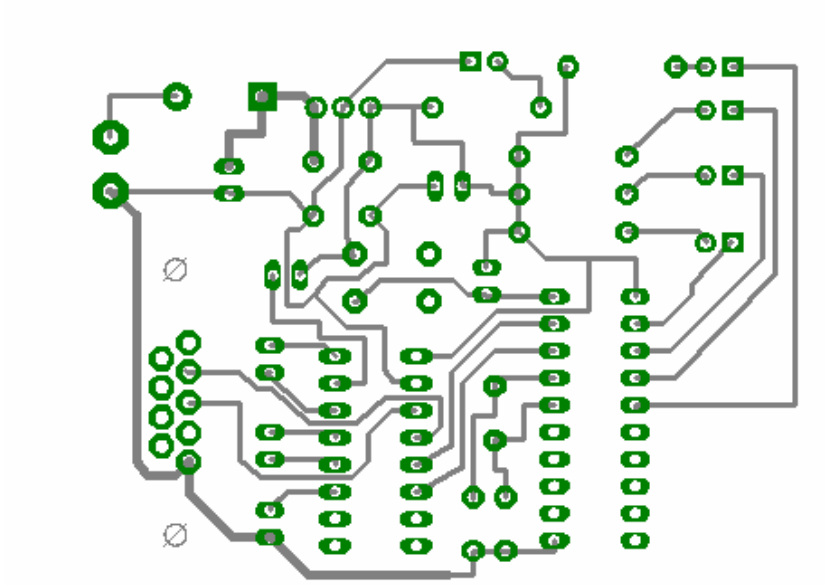
Tab. 7.1 – piny Cannon9



Obr. 7.10 – konektor samec Cannon9



Obr 7.11 – deska plošného spoje ovládacího prvku - osazení součástek



Obr. 7.12 – deska plošného spoje ovládacího prvku

Označení	Typ součástky	Počet kusů
D1	Dioda 1N4007	1x
IC2	Stabilizátor MA7805	1x
R1	Rezistor 220	1x
R2, R3, R4, R5	Rezistor 20	4x
D2, D3, D4, D5, D6	Dioda LED	5x
C1	Kondenzátor elektrolyt. 100 uF, 24V	1x
C4, C5	Kondenzátor keramický 0,1 uF	2x
C6, C7	Kondenzátor keramický 22pF	2x
C2,C3, C8, C9, C11, C12	Kondenzátor elektrolyt. 1 uF, 24V	6x
Q1	Rezonátor 12MHz	1x
IC1	Integrovaný Obvod MAX 232	2x
IC3	Mikroprocesor AT89C2051P	1x
SW1	Tlačítko P-B1720	1x
CAN1	Konektor Cannon 9	1x

Tab. 7.2 – seznam součástek ovládacího prvku

7.2.4 Program pro Mikroprocesor AT89C2051P

Program pro mikroprocesor AT89C2051P je navržen tak, aby stéle testoval přicházející spojení sériové linky na pinu P3.0 Rx data. V momentě, kdy zaznamená událost, přijme data a uloží do akumulátoru. Další instrukcí data porovná s tabulkou znaků, která je shodná s příkazy, které zasílá server. Pokud nalezne shodu, podle přijatého znaku na 5s přepne jeden z pinů P1.4, P1.5, P1.6, P1.7, na log 0. Tím dojde k sepnutí obvodu a v demonstračním modulu k rozsvícení příslušné LED diody. Po stanovené době opět přepne příslušný výstup zpět na výchozí stav a vrazí se na začátek smyčky. Opět čeká na přicházející spojení. V průběhu vykonávání přijatého příkazu procesor nereaguje na jakékoli podněty, které by v tu chvíli byly zaznamenány na vstupu P3.0 Rx data.

Celý program je uzavřen v nekonečné smyčce. Pouze při spuštění mikroprocesoru dochází k vykonání instrukcí, které nastavují proměnné a mód sériové komunikace. Sériová linka je nastavena do módu 1. Program pro jednočipový mikroprocesor Atmel AT89C2051P je napsán v jazyce symbolických adres Assembler. Tento jazyk je použit z důvodu menší náročnosti na paměť procesoru, možnost detailního a přesného sledování chodu procesoru. Zdrojový kód programu pro Atmel AT89C2051 je dostupný v příloze č. 1 této diplomové práce.

8 Instalace a nastavení software

Pro správnou funkci software pro vzdálené ovládání je potřeba, aby bylo vše správně nainstalováno a nastaveno. Jedině tak je možné zajistit opravdu bezproblémovou funkci. V této poslední kapitole se budeme věnovat instalaci, nastavení a ovládání softwaru jak serverové, tak klientské části. Dále pak správnému nastavení a používání převodníku a ovládacího prvku s jednočipovým mikroprocesorem. Nakonec bude uvedeno základní nastavení síťového routeru pro příklad propojení klienta a serveru pomocí sítě internet. Zde se setkáme s problémem, který spočívá ve známém překladu adres NAT.

8.1 Serverová část software

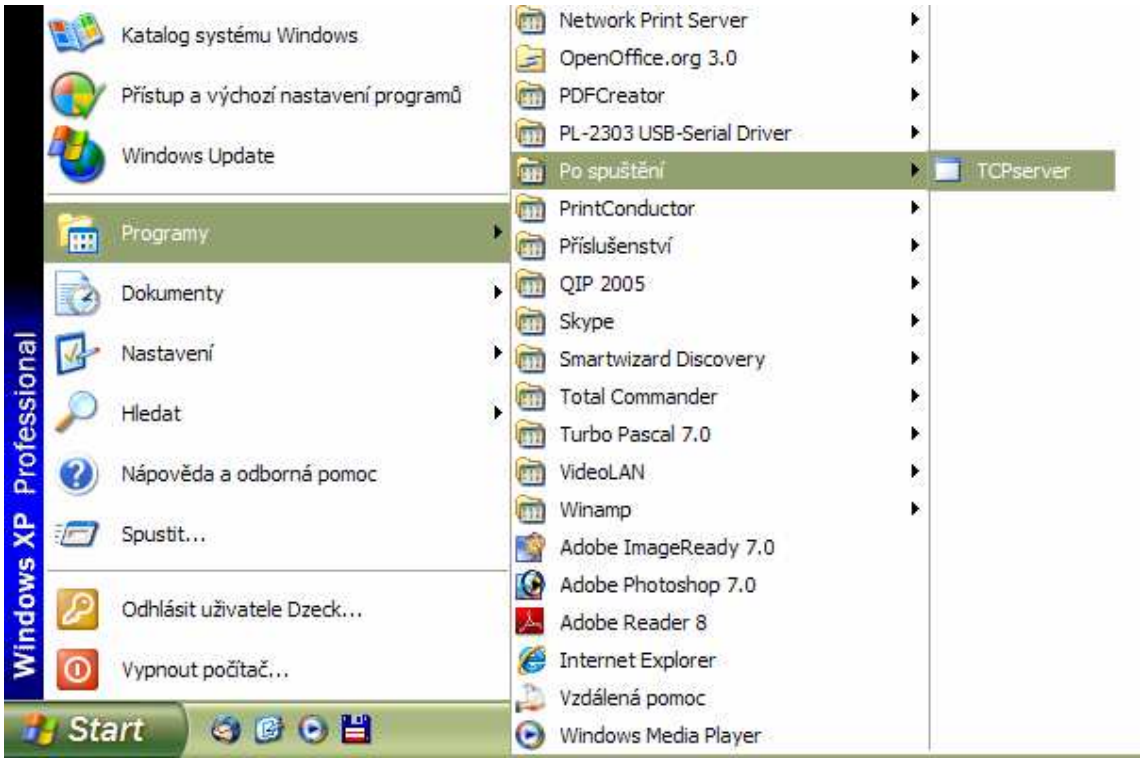
Jelikož je serverová část navržena jako konzolová aplikace, po nainstalování a spuštění je vidět pouze černé okno příkazového řádku s nápisem „server ready“. Serverovou aplikaci není potřeba po spuštění jakkoli nastavovat nebo upravovat. V tomto případě je server správně nainstalován a připraven k použití. Nesmíme zapomenout, že server naslouchá na portu TCP 2010 a odesílá potvrzení na portu TCP 2020. Pokud počítač na kterém je serverová část aplikace provozována obsahuje firewall, je nutné tyto čísla portů povolit. V opačném případě nebude možné na server navázat spojení nebo bude serveru znemožněno zasílání potvrzení klientským stanicím. Samozřejmostí je, aby počítač, na kterém je server spuštěn, měl přidělenou platnou IP adresu do patřičné sítě. Server pro svůj běh vyžaduje, aby v operačním systému byl nainstalován .NET framework 2.0 nebo vyšší

8.1.1 Instalace software

Stabilní verze serveru, která je otestovaná a připravená k použití je TCPserver_1_0_0_5. Jakoukoli jinou verzi nepoužívejte a neinstalujte. Tato verze je otestována a vyzkoušena pouze pro operační systémy:

- Microsoft Windows XP home SP1, SP2, SP3
- Microsoft Windows XP profesionall SP1, SP2, SP3
- Microsoft Windows XP
- Microsoft Windows Server 2003 R2

Z instalačního CD spusťte v adresáři TCPserver soubor setup.exe. Při výzvě upozornění zabezpečení zvolte tlačítko „instalovat“. Nechte proběhnout instalační proces a po nainstalování se serverová aplikace automaticky spustí. U serverové aplikace je ještě důležité, aby po zapnutí počítače se automaticky spustil i serverový software. Pro zajištění spuštění je potřeba přidat zástupce spouštěcího souboru serveru do složky v nabídce start – „Po spuštění“ viz. obr 8.1.



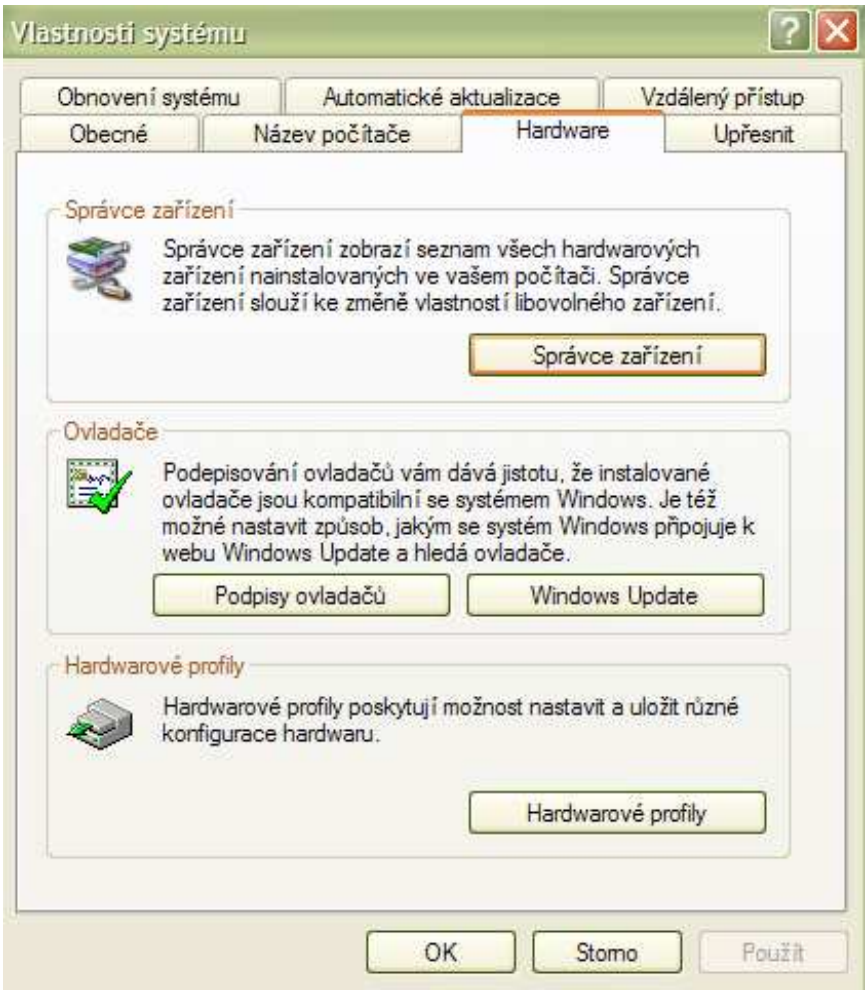
Obr 8.1 – složka „Po spuštění“

8.1.2 Nastavení sériové linky a firewallu

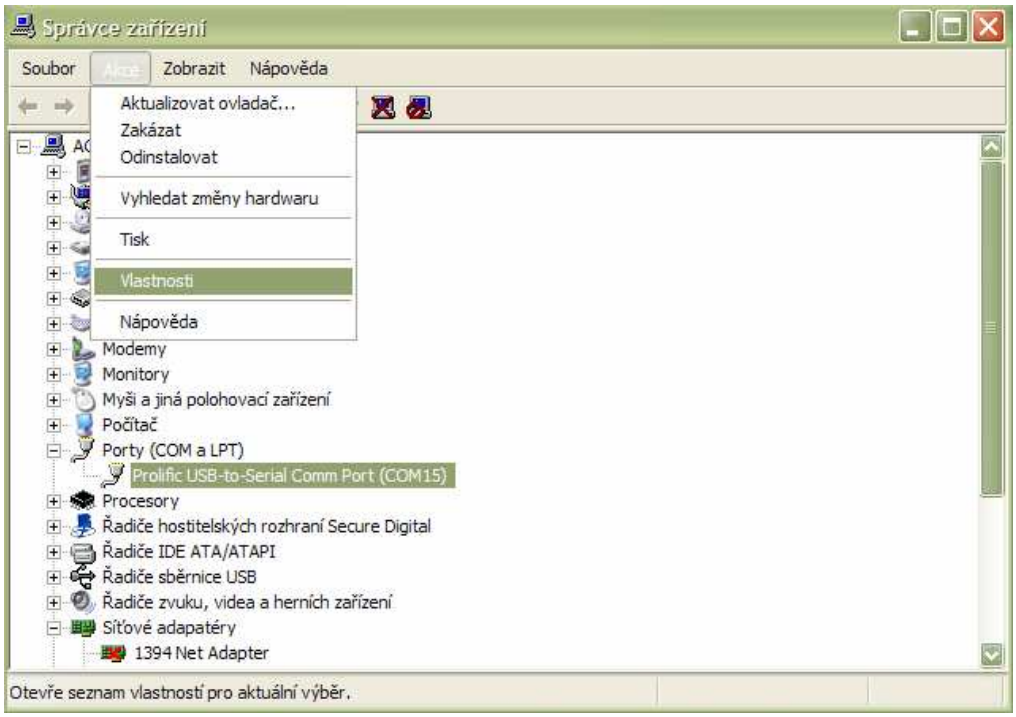
Poslání serverové aplikace je přijetí příkazů od klientů a posílání na ovládací prvek. S jednočipovým mikroprocesorem je zajištěna komunikace pomocí sériové linky. Jelikož používáme USB převodník, je v počítači nainstalován virtuální sériový port. Jediný parametr, který je potřeba pro server v počítači nastavit je adresa tohoto portu. Je nutné, aby byl převodník již nainstalován. Poté je potřeba nastavit adresu com portu na COM15 a změnit další parametry sériového portu.

Postup změny nastavení je nabídka Start – Nastavení - Ovládací panely – Systém – Záložka hardware – kliknout na tlačítko „Správce zařízení“, obr 8.2. Poté ve okně Správce zařízení rozklikněte ve stromu položku „Porty (COM a LPT)“ a vyberte položku „Proliftic USB-to-Serial Comm port“, obr 8.3. Po výběru zvolte v menu „Akce“ položku „vlastnosti“. V zobrazeném okně zvolte záložku „nastavení portu“. Jednotlivé položky nastavte dle obr 8.4. Poté klikněte na tlačítko „upřesnit“ a v okně upřesňující nastavení nastavte jednotlivé položky dle obr. 8.5. Pokud je vše nastaveno, je server připraven k činnosti.

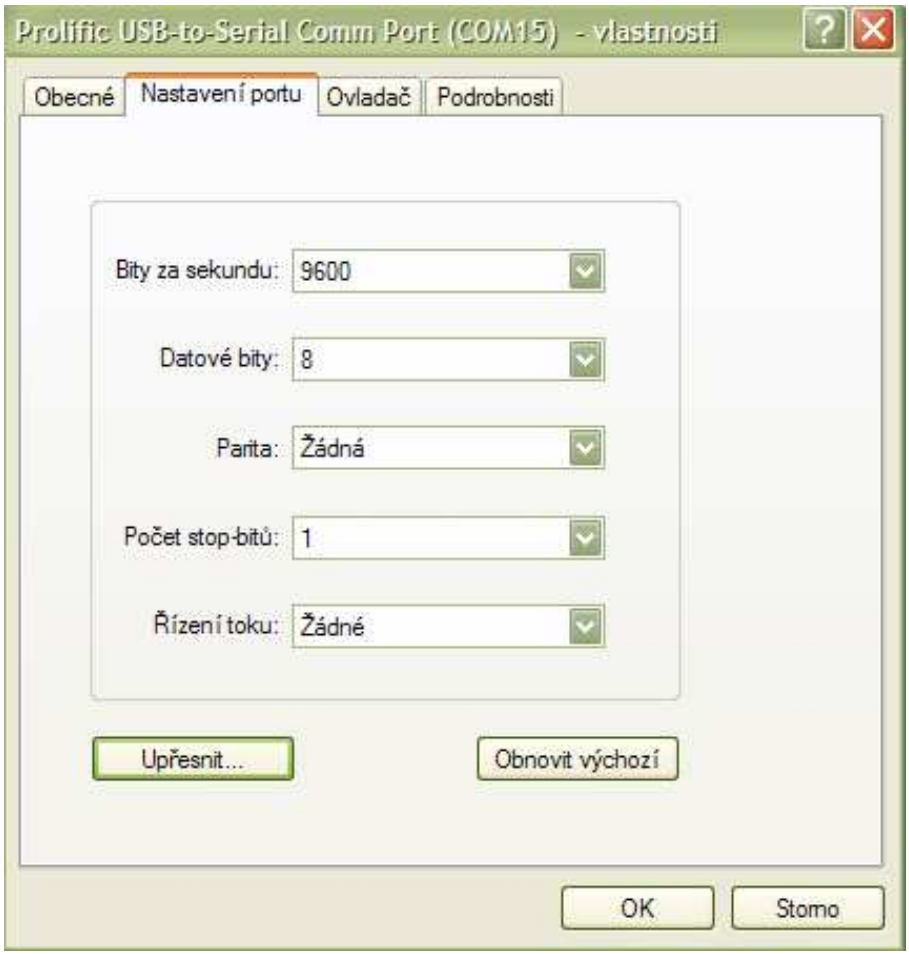
Server naslouchá na portu TCP 2010 a odesílá potvrzení na portu TCP 2020. Je nutné mít ve firewallu tyto porty otevřené. Nastavení firewallu v Microsoft Windows naleznete na obr. 8.6.



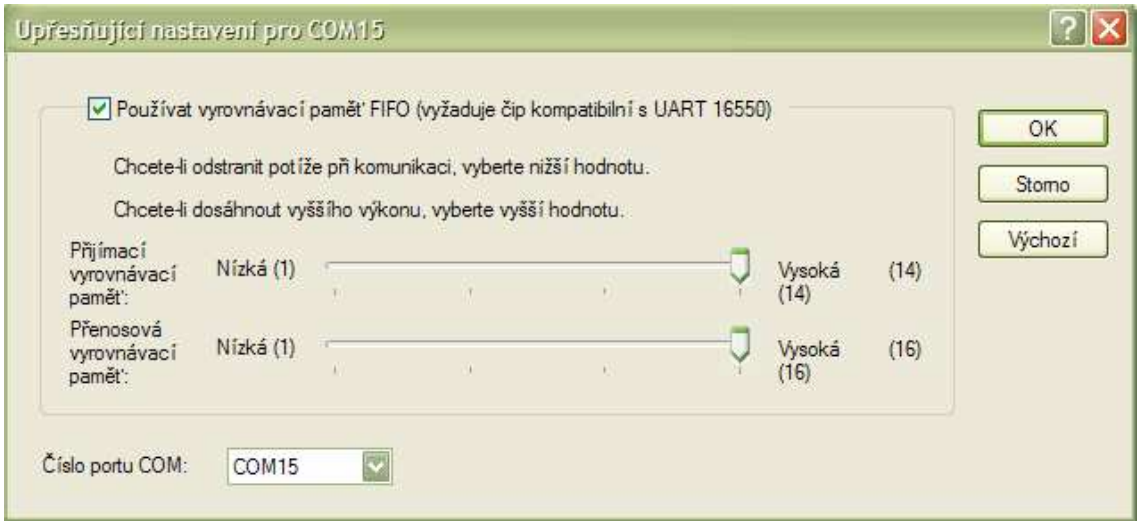
Obr 8.2 – spuštění Správce zařízení



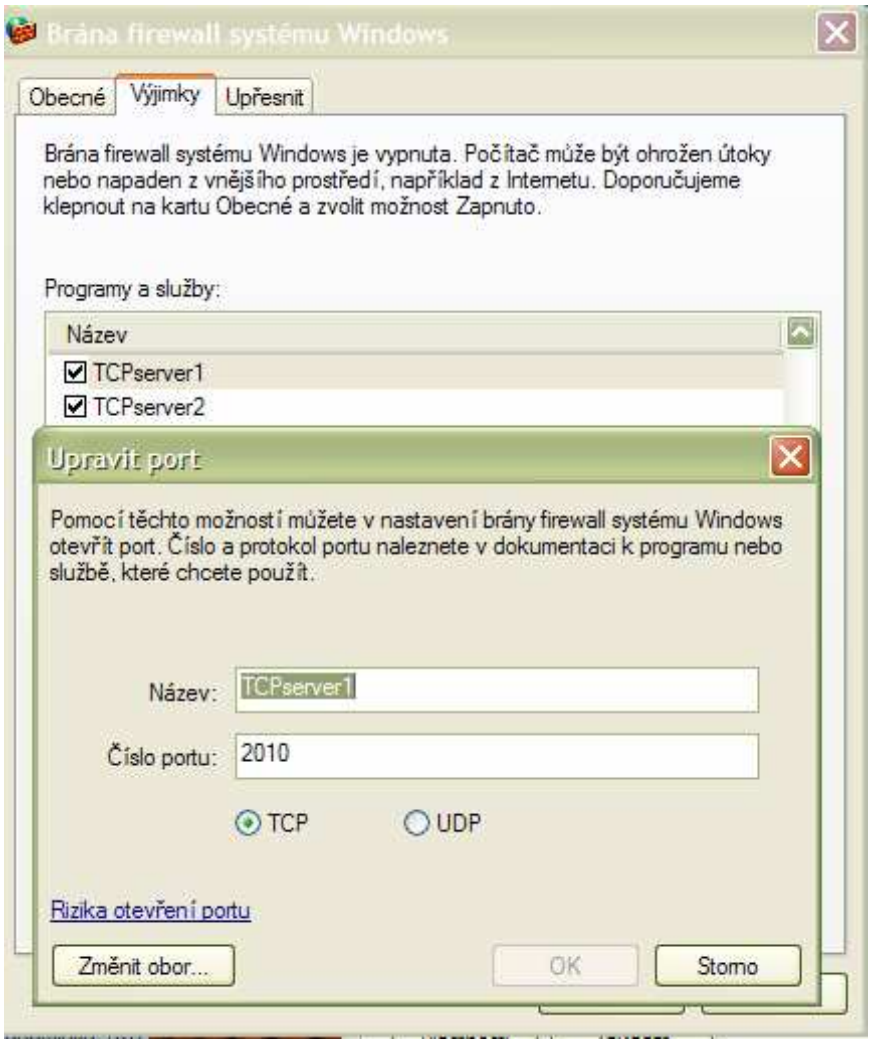
Obr 8.3 – možnosti nastavení COM portu



Obr 8.4 – nastavení com portu



Obr. 8.5 – upřesňující nastavení com portu



Obr. 8.6 – nastavení firewallu ve Microsoft Windows XP

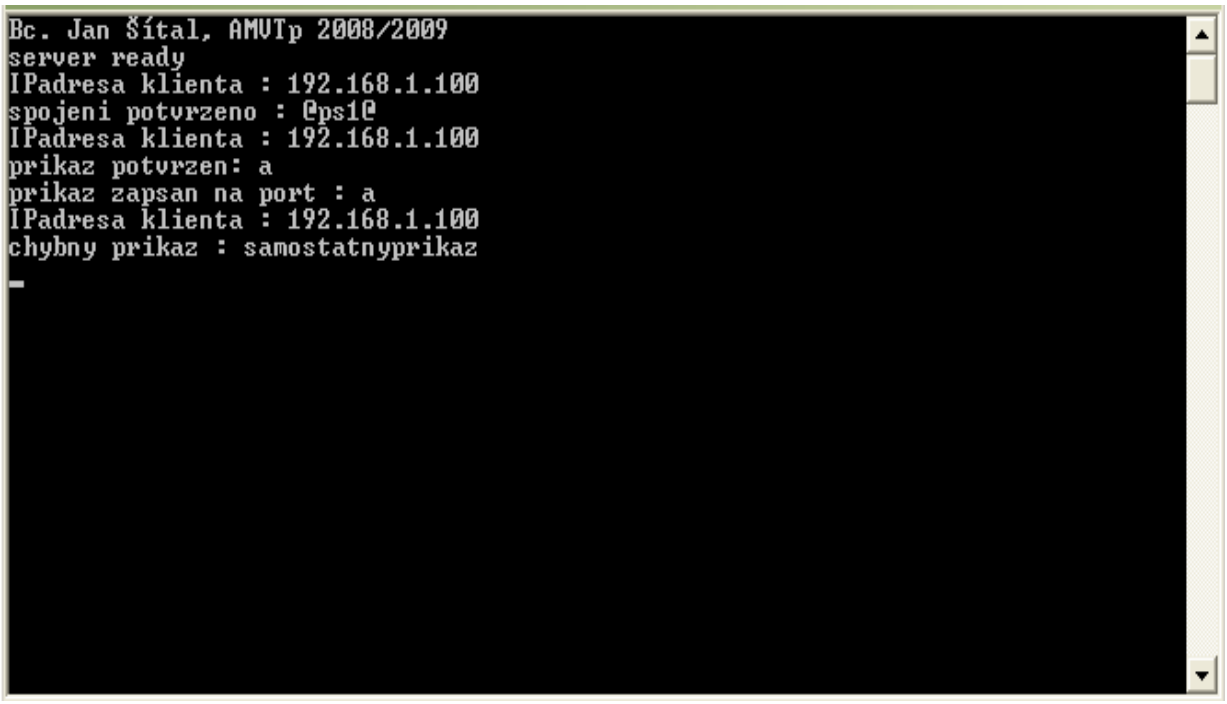
Používaných firewallů jakožto ochran v sítích či jednotlivých počítačů je celé řada a každý má své specifické nastavení. Filozofie povolení portů je však stále stejná. Na obrázku 8.6 je ukázka nastavení firewallu, který je součástí operačního systému Microfsoft Windows. Jsou dvě možnosti, jak povolit serveru komunikaci po datové síti. Buď povolit program jako takový, bez žádného dalšího omezení firewallem nebo povolit jen patřičná čísla jednotlivých portů.

První popisovaná možnost není příliš doporučena, může se tím narušit bezpečnost celého systému, kde je serverová aplikace spuštěna. Na obrázku 8.6 je příklad nastavení otevření jednoho TCP portu. Pro server je nutné, aby byly otevřeny porty dva. Proto na obrázku 8.6 vidíte v okně, které je v pozadí dva řádky. První povoluje TCP port 2010 a druhý TCP port 2020.

8.1.3 Provoz serveru a hlášení

Serverová aplikace je navržena tak, aby nebylo potřeba nastavovat parametry v software, ale v základním nastavení operačního systému. Při provozu není potřeba u serveru cokoli měnit nebo nastavovat. Serverová aplikace je závislá jen na nastavení operačního systému. Pokud chceme, aby se serverová aplikace spustila automaticky po spuštění počítače, je nutné přidat zástupce do složky „Po spuštění“, jak je popsáno v kapitole 8.1.1.

Každý síťový software by měl mít možnost výpisů aktuálního stavu při komunikaci. I u této serverové aplikace máme možnost sledovat její aktuální činnost. Hlášení je řešeno jednoduchým způsobem. Server do konzolového okna textově vypisuje po řádcích pod sebe vždy IP adresu klienta ze které byl příkaz přijat, přijatý příkaz, a poté stav co bylo s příkazem provedeno. Takto je možno sledovat činnost serveru v případě poruchy nebo chyby při vykonávání příkazů. Ukázka je k vidění na obr. 8.7. Server zaznamenává jakýkoli přijatý příkaz IP adresu odesilatele, stav příkazu a co s ním bylo provedeno.



Obr. 8.7 – výpis hlášení serveru

8.2 Klientská část aplikace

Klientská část software je navržena jako aplikace Windows. Důvody, výhody a nevýhody tohoto řešení jsou popsány v kapitole 2. Po spuštění aplikace se zobrazí okno spuštěného programu a software je připraven k použití. Pro provoz klienta není nutné jakékoli další nastavení ze strany operačního systému. V počítači, na kterém je klient spuštěn je nutné jediné nastavení, a to, aby byl připojen k síti a byla přidělena patřičná IP adresa. Pokud je v operačním systému zapnut firewall, je potřeba mít povolené porty TCP 2010 a 2020 stejně jak tomu je i u serverové části aplikace. Požadavek na operační systém je jediný a to aby byl nainstalován .NET framework 2.0 nebo vyšší. Pokud nebude některý z portů povolen, klient buď vůbec nenaváže spojení serverem nebo nebude moct přijímat hlášení ze strany serveru.

8.2.1 Instalace software

Stabilní verze serveru, která je otestovaná a připravená k použití je TCPklient_1_0_0_3. Jakoukoli jinou verzi nepoužívejte a neinstalujte. Tato verze je otestována a vyzkoušena pouze pro operační systémy:

- Microsoft Windows XP home SP1, SP2, SP3
- Microsoft Windows XP profesionall SP1, SP2, SP3
- Microsoft Windows XP 64
- Microsoft Windows Vista basic, premium, ultimate

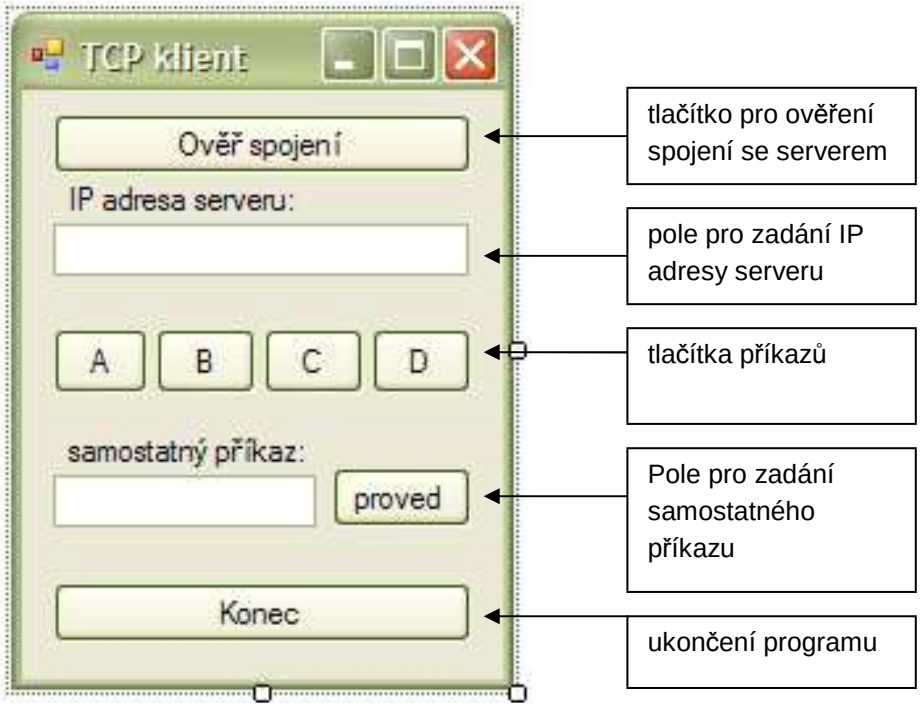
Z instalačního CD spusťte v adresáři TCPklient soubor setup.exe. Při výzvě upozornění zabezpečení zvolte tlačítko „instalovat“. Nechte proběhnout instalační proces a po nainstalování se serverová aplikace automaticky spustí. Pokud vše proběhne a po nainstalování dojde ke automatickému spuštění software, proběhla instalace úspěšně. Klientskou část aplikace je možné provozovat na jakémkoli počítači, kde je nainstalován jako operační systém jeden z výše uvedených a .NET framework 2.0 nebo vyšší.

8.2.2 Ovládání klientské části

Klienta je možné ovládat v grafickém prostředí Microsoft Windows pomocí klávesnice i myši. Po spuštění je potřeba zadat jen IP adresu serveru, na který mají být příkazy odeslány do pole označené „IP adresa serveru“. Před samotným odesláním příkazu je doporučeno ověřit, zda je zadaný server dostupný. K tomu slouží tlačítko „Ověř spojení“. Po jeho stisknutí se zobrazí informační box se zprávou zda je server dostupný či nikoli. IP adresu je potřeba zadat ve standardním formátu včetně teček mezi

číslicemi. Pokud bude zadána špatná IP adresa, tak po stisknutí tlačítka ověření spojení serveru nebo po pokusu o odeslání příkazu klient zobrazí chybu v informačním boxu.

Předdefinované příkazy je možné odesílat pomocí tlačítek s označením A,B,C,D. Každé tlačítko ovládá jeden výstup jednočipového mikroprocesoru v ovládacím prvku. Kromě předefinovaných příkazů je možné zasílat na server i samostatné příkazy. K tomu slouží pole s označením „samostatný příkaz“. Stačí do pole zapsat příkaz a poté stisknout tlačítko „proved“ a příkaz bude odeslán na server. Ukončení klienta může být provedeno stisknutím tlačítka „Konec“ nebo stisknutím křížku v pravém horním rohu okna klienta. Popis okna klienta je uveden na obrázku 8.8.



Obr 8.8 – popis okna klienta

8.3 Spojení pomocí sítě internet

Zasílání příkazů na server je samozřejmě možné i pomocí sítě internet. Ostatně, bylo by v dnešní době chybou aby toto spojení nebylo možné. Při přenosu pomocí sítě internet je potřeba řešit daleko více faktorů ovlivňujících spojení mezi klientem a serverem. Je to dáno tím, že na cestě spojení je daleko více aktivních síťových prvků, spojení je realizováno přes více samostatných sítí a ve spojení hraje roli více protokolů, než jen samotné TCP/IP. Problémy spojení mohou též způsobovat firewally routerů v síti internet, přes které je spojení realizováno.

8.3.1 Spojení pomocí veřejných IP adres

Nejjednodušší situace, která může nastat, je pokud bude mít klient i server přímo přidělenou veřejnou IP adresu. V tomto případě se nic neliší od běžného propojení pomocí sítě LAN. Je potřeba akorát ověřit, zda na cestě není někde blokován TCP port 2010 nebo 2020. Pokud tomu tak není, spojení může být bez problémů realizováno a není potřeba jakkoli upravovat nastavení aktivních síťových prvků, které se účastní spojení.

8.3.2 Spojení při překladu adres NAT

Pokud se dostaneme do situace, kde nebude mít klient ani server přidělenou veřejnou IP adresu přímo, nezbývá nám nic jiného, než upravit nastavení aktivních síťových prvků. Na straně serveru je nutné, aby součástí sítě byl aktivní prvek, který bude mít přidělenou veřejnou IP adresu. Může se jednat o router nebo síťový server, který slouží na daném místě jako brána do sítě internet s překladem adres NAT. Totéž musí být zajištěno i na straně klienta. Na tomto aktivním síťovém prvku je potřeba nastavit takzvané přesměrování portů. Na straně serveru je potřeba přesměrovat port TCP 2010 a na straně klienta port TCP 2020 tak, aby ukazoval na počítač kde jsou tyto aplikace spuštěny. Nastavení spočívá v tom, že se routeru řekne, aby vše, co přijde na jeho veřejnou IP adresu s daným číslem portu, tak přesměroval na počítač, kde je spuštěna aplikace. V pravidle pro router se využívá Destination-Nat a je potřeba zadat číslo zdrojového a cílového portu (v tomto případě obojí buď 2010 nebo 2020) a IP adresu počítače, na kterém je spuštěn server nebo klient. Pokud je toto nastavené, neměl by být s propojením jakýkoli problém.

8.3.3 Spojení pomocí VPN tunelu

Další možností, jak server vzdáleně ovládat je propojení počítačů pomocí některé formy VP tunelu. Je to řešení vcelku jednoduché a efektivní. Podmínkou je, aby na počítači serveru nebo bráně přes kterou je server připojený do internetu bylo možné spustit některý VPN server. Může se jednat o PPTP, L2TP apod. V tomto případě potřebujeme veřejnou IP adresu pouze na straně, kde je umístěn server, u klienta potřeba není a nemusíme nikde nastavovat ani jakékoli přesměrování portů. Po vytočení a spojení tunelu zadáme u klienta přímo interní IP adresu serveru, jako kdyby jsme byli připojeni ve stejném segmentu sítě. Ostatně to je také účel VPN tunelů jako takových.

Závěr

Zaměření této diplomové práce je spíše pro praktické využití. Obsaženou teoretickou část jsem zvolil vždy k danému tématu pro spíše pro doplnění dané problematiky. Je zde několik částí ze kterých se práce skládá a v každé je praktická část mým vlastním návrhem. Jedná se o software typu klient – server, kde komunikace probíhá pomocí protokolu TCP/IP v aplikační verstě, je tím pádem nezávislá na použitém přenosovém médiu ani na síťových prvcích přes které je spojení uskutečněno. Návrh doporučeného hardware a ovládacího prvku s jednočipovým mikroprocesorem. Propojení ovládacího prvku je řešeno pomocí USB portu a následného převodu na sériovou linku pro jednočipový mikroprocesor.

Aplikace, která je obsažena v této práci je spíše ukázkou. Pro praktické využití je potřeba vždy jak klientskou část software, tak jednočipový mikroprocesor obsažený v ovládacím prvku vždy přizpůsobit na míru danému využití. V současné je software s ovládacím prvkem nasazen pro vzdálený restart rádiové části vysílače využité pro připojení uživatelů k bezdrátové síti (multipoint). Testování probíhá v mé vlastní síti a zatím nebyl zaznamenán žádný závažný problém s nasazením software.

Využití software je velice široké a univerzální. Od aplikací jednoduchých jakými může být elektronický vrátný ovládaný například přes wifi, vypínání a zapínání spotřebičů, možnost ovládat například dveře nebo bránu z různých míst s využitím stávající datové sítě apod. až po aplikace složitější, kdy je možné pomocí jednočipového mikroprocesoru nejenom zasílat příkazy, ale i stahovat data a posílat je na server, kde mohou být přístupná vzdáleným klientům. Právě obousměrná komunikace mezi serverem a mikroprocesorem a ukládání dat serverem je vize dalšího vývoje tohoto softwaru a jeho případných aplikací. U jednočipového mikroprocesoru lze pro komunikaci s okolím využít minimálně 8 vstupů/výstupů a tím je dána možnost širokého a univerzálního využití.

Seznam použitých zdrojů

Tištěné zdroje:

[1] Kabelová, A., Dostálek, L.: Velký průvodce protokoly TCP/IP a systémem DNS, Praha, Computer Press, 2002
[2] Robert Láníček: Elektronika (obvody, součástky, děje), BEN, Praha, 1999
[3] Jan Vobecký, Vít Záhlava: Elektronika (součástky a obvody principy a příklady), Grada Publishing, Praha 7, 2000
[4] Hrbáček, J.: Komunikace mikrokontroléru s okolím 1. BEN – technická literatura
[5] Hrbáček, J.: Komunikace mikrokontroléru s okolím 2. BEN – technická literatura
[6] David Matoušek: Práce s mikrokontroléry Atmel AT89C2051. Ben – technická literatura, 2002
[7] John Sharp: Microsoft Visual C# 2005 krok za krokem – Computer Press, 2006
[8] Mareš Amadeo: 1001 tipů a triků pro C# - Computer Press, 2008

Webové zdroje:

[9] http://www.hw.cz
[10] http://www.mcu.cz
[11] http://www.elweb.cz
[12] http://www.aradio.cz
[13] http://www.svetsiti.cz
[14] http://www.svethardware.cz
[15] http://www.zive.cz
[16] http://www.dhservis.cz
[17] http://www.volny.cz/fuksam
[18] http://www.programujte.com
[19] http://www.via.com
[20] http://www.seagate.com
[21] http://www.pcengines.ch
[22] http://www.atmel.cz
[23] http://www.emko.cz
[24] http://www.kingston.com

Seznam obrázků

Seznam tabulek

Přílohy

Příloha 1 - program pro Atmel AT89C2051:

```
;*************************************************************************
;programmed by Bc. Jan Šítal, AMVTp 2008/2009
;*************************************************************************
START:          MOV     SCON,#52H               ;Nastaveni modu seriove linky
                MOV     TMOD,#20H               ;Nastaveni seriove linky
                MOV     TH1,#0FDH               ;Nastaveni rychlosti a ser.liny,c/c
                MOV     TL1,#0FDH               ;Nastaveni kvuli 1.spusteni
                SETB    TR1                     ;Spusteni casovace
                MOV     P1,#0FFh                ;Nastaveni vystupu
                MOV     DPTR,#0C000H            ;Nastaveni ukazatele
                CLR     A                       ;Vymazani akumulatoru
                MOVC    A,@A+DPTR               ;Prepnuti konektoru ser.linky
                MOV     P1,#00001111b           ;Test diod pri prvnim spusteni
                CALL    ZPOZDENI                ;Zpozdeni diody sviti
                MOV     P1,#0FFh                ;Nastaveni vystupu
                CALL    ZPOZDENI                ;Zpozdeni diody zhasle
                MOV     P1,#00001111b           ;Nastaveni vystupu
                CALL    ZPOZDENI                ;Zpozdeni diody sviti
                MOV     P1,#0FFh                ;Nastaveni vystupu

PR:             CALL    PRIJEM                  ;Spusteni prijmu ser.linky,ceka na data
                CJNE    A,#'a',POKR             ;Porovnani akumulatoru se znakem a
                CLR     A                       ;Vymazani akumulatoru
                CLR     P1.4                    ;Prepnuti vystupu p1.4 log0
                CALL    ZPOZDENI                ;Zpozdeni 1s vystup p1.4 v log0
                CALL    ZPOZDENI                ;Zpozdeni 1s vystup p1.4 v log0
                SETB    P1.4                    ;Prepnuti vystupu p1.4 log 1

POKR:           CJNE    A,#'b',POKR2            ;Porovnani akumulatoru se znakem b
                CLR     A                       ;Vymazani akumulatoru
                CLR     P1.5                    ;Prepnuti vystupu p1.5 log0
                CALL    ZPOZDENI                ;Zpozdeni 1s vystup p1.5 v log0
                CALL    ZPOZDENI                ;Zpozdeni 1s vystup p1.5 v log0
                SETB    P1.5                    ;Prepnuti vystupu p1.5 log 1

POKR2:  CJNE    A,#'c',POKR3                    ;Porovnani akumulatoru se znakem c
                CLR     A                       ;Vymazani akumulatoru
                CLR     P1.6                    ;Prepnuti vystupu p1.6 log0
                CALL    ZPOZDENI                ;Zpozdeni 1s vystup p1.6 v log0
                CALL    ZPOZDENI                ;Zpozdeni 1s vystup p1.6 v log0
                SETB    P1.6                    ;Prepnuti vystupu p1.5 log 1

POKR3:  CJNE    A,#'d',POKR4                    ;Porovnani akumulatoru se znakem d
                CLR     A                       ;Vymazani akumulatoru
                CLR     P1.7                    ;Prepnuti vystupu p1.7 log0
                CALL    ZPOZDENI                ;Zpozdeni 1s vystup p1.7 v log0
                CALL    ZPOZDENI                ;Zpozdeni 1s vystup p1.7 v log0
                SETB    P1.7

POKR4:          CLR     A                       ;Vymazani akumulatoru
                JMP     PR                      ;Skok na navesti PR

PRIJEM: JNB     RI,$                            ;Testovani priznaku seriove linky
                MOV     A,SBUF                  ;Naplneni stradace
                CLR     RI                      ;Nulovani priznaku
                RET                             ;Ukonceni podprogramu
```

```
ZPOZDENI:       MOV     R3,#5                   ;Zpozdeni 1s
CAS1:           MOV R1,#254                     ;Naplneni registru
CAS2:           DJNZ R0,CAS2                    ;Snizeni R0 a porovnani
                MOV R2,#107                     ;Naplneni R2
                DJNZ R2,$                       ;Snizeni R2 a porovnani
                DJNZ R1,CAS2                    ;Snizeni R1 a porovnani
                NOP                             ;Prodleva 1 cyklu
                MOV R2,#8                       ;Naplneni R2
                DJNZ R2,$                       ;Snizeni R2 a porovnani
                DJNZ R3,CAS1                    ;Snizeni R2 a porovnani
                RET                             ;Ukonceni podprogramu
KON:    END
```

Příloha 2 – nosič dat CD:

Na přiloženém nosiči dat CD v adresáři `zdrojovekody` jsou umístěny kompletní zdrojové kódy aplikace klient i aplikace server a programu pro Atmel AT89C2051. V adresáři `instalace` jsou umístěny instalační soubory těchto aplikací klient a server a soubor ve formátu .hex pro programátor jednočipového mikroprocesoru. Dále je na tomto CD v adresáři `diplomovaprace` umístěna elektronická podoba této diplomové práce ve formátu .DOC a .PDF.