JIHOČESKÁ UNIVERZITA V ČESKÝCH BUDĚJOVICÍCH

PEDAGOGICKÁ FAKULTA

KATEDRA FYZIKY

Mikrokontroléry Atmel AVR použití HW vývojového kitu STK500, vývojového prostředí AVR Studio a programovacího jazyka Bascom.

BAKALÁŘSKÁ PRÁCE

Vedoucí práce: Ing. Michal Šerý

Autor: Jiří Volf

Anotace

Cílem práce je popis 8-bitových mikroprocesorů AVR série s RISC architekturou, instrukčního souboru a programovacích prostředí AVR Studio a Bascom. Dále se práce zabývá komunikací mikroprocesoru s okolím se zaměřením na konkrétní typ ATmega32.

Práce je rozdělena do čtyř částí. V první části se zabývám obecnou historií a vývojem logických programovatelných obvodů. Druhá část je věnována mikrontroléru ATmega32 jeho základním parametrům, HW jádru a instrukční sadě. Ve třetí části se zabývám srovnáním softwarových programovacích nástrojů AVR Studio a Bascom , možnou spoluprácí s hardwarovým vývojovým kitem STK 500. Čtvrtá část je zaměřena na praktickou ukázku komunikace MCU a IPchipu Nano SocketLan pomocí sady AT příkazů. Konstrukce modulu umožňuje vybavit jakékoli zařízení TCP/IP komunikací.

Annotation

Goal of this work is a description of 8-bit microprocessors of AVR series with RISC architecture, instruction set and programming environment, Bascom and AVR Studio. Furthermore, this work deals with communications with the microcontroller environment, focusing on a specific type ATmega32.

The work is divided into four parts. The first section deal with the general history and development of programmable logic circuits. The second part is attended to the microcontroller ATmega32 its basic parameters, HW core and instruction set. In the third part we deal with a comparison of software programming tools and Bascom AVR Studio and possible cooperation with a hardware development kit STK 500th The fourth part is focused on practical demonstration of communication and MCU IPchip Nano-SocketLAN using a set of AT commands. The design module allows any device equipped with TCP / IP communications.

Prohlášení:

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Michala Šerého a uvedl v seznamu literatury všechny použité literární a odborné zdroje.

Prohlašuji, že v souladu s § 47b zákona č. 111/1998 Sb. v platném znění souhlasím se zveřejněním své bakalářské 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.

České Budějovice, 2009 Podpis:

Poděkování:

Děkuji panu Ing. Michalu Šerému, za velmi užitečnou metodickou a odbornou pomoc, kterou mi poskytl při zpracování této bakalářské práce.

České Budějovice, 2009 Jiří Volf

Obsah :

Úvod

Tématem bakalářské práce je seznámení s programovatelnými obvody ATMEL a ověření teoretických základů na praktické konstrukci TCP/IP modulu.

Základem celé konstrukce je AVR mikroprocesor ATmega32 a hotový IPmodul Nano SocketLan od firmy Connect One. Právě užití této moderní součástkové základny ve spojení se silným programovacím prostředím Bascom a vývojového kitu STK500 celou konstrukci značně zjednodušuje a rozšiřuje oblast užití i mimo profesionální praxi. Vhodným naprogramováním a uspořádáním modulu lze vybavit a připojit jakékoli zařízení komunikují po sériové lince přímo do internetu. Během konstrukce modulu bylo využito možností dalších vývojových prostředků jako např. návrhový systém pro plošné spoje Eagle (free verze), WYSIWYG editor webových stránek (free verze), výroba plošných spojů fotocestou apod.

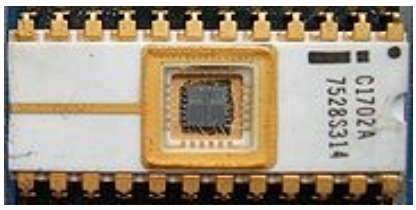
K výběru tématu jsem byl inspirován zájmem o programovatelné obvody a jejich užití v systémech ovládání a řízení, zvláště v době kdy se celá tato problematika přenosu informace řeší pomocí již velmi dostupné internetové sítě.

1. Historie programovatelných obvodů

První integrovaný obvod zkonstruoval Jack St. Clair Kilby z firmy Texas Instruments již v roce 1958. Kilbyho integrovaný obvod byl na základní desce z germania o velikosti 11 × 1,6 mm a obsahoval jediný tranzistor s několika pasivními součástkami. Tento vynález si nechal patentovat v roce 1964 . V témže roce si také připsal konstrukci první kapesní kalkulačky založené na integrovaném obvodu zpracovávající jednoduché matematické operace. V roce 2000 byl oceněn Nobelovu cenu za fyziku.

1.1 EPROM, EEPROM, FLASH

EPROM (*Erasable Programmable Read-Only Memory)*. Jedná se o pamět typu ROM-RAM, jejíž obsah je mazatelný ultrafialovým zářením. Pokud chceme paměť nově programovat je nutné paměť nejdříve kompletně smazat. K programování se používá většinou několikanásobně vyšší napětí než ke čtení (typ. 12 V nebo 25 V proti 5 V napájecího napětí). Paměť se používá k uložení dat (např. firmware), často u malosériové výroby, kde není vyžadována možnost měnit obsah paměti v již zabudovaném zařízení.



Obr. č. 1 – Paměť EPROM (okénko slouží k vymazání obsahu UV zářením)

EEPROM (*Electronically Erasable Programmable Read-Only Memory)*. Jedná se o elektricky mazatelnou paměť typu ROM-RAM. Paměť má umožňuje menší počet zápisů než paměť typu flash. Před novým naprogramováním musíme pomocí elektrického signálu smazat celý její obsah. (Výhoda oproti EPROM, pro smazání bylo nutné použití ultrafialového světla) Využití této paměti je jako úložiště (např. firmware) u zařízení, kde nedochází často k přepisům paměti. Nyní se již od použití této paměti upouští a využívá se paměti typu flash.

FLASH je nevolatilní[1] elektricky programovatelná paměť s libovolným přístupem. Vnitřní organizace paměti je v blocích na rozdíl od pamětí typu EEPROM, lze programovat každý blok samostatně (nedochází k přepisu ostatních bloků). Paměť se používá jako paměť typu ROM např. pro uložení firmware (např. ve vestavěných zařízeních). Výhodou této paměti je, že ji lze znovu naprogramovat (např. přeprogramování novější verzí firmware) bez vyjmutí ze zařízení s použitím minima pomocných obvodů. Flash paměť se používá jako výměnné datové médium ve formě paměťových karet. Flash paměť se používá i v discích Solid State SSD jako vestavěná paměť, kde ji označujeme jako paměťové technologické zařízení Memory Technology Device.

[1] Data jsou uchována i bez přítomnosti napájecího napětí

1.1.1 Funkce flash paměti

Data jsou ukládána v poli tranzistorů (plovoucí brány), zvaných „buňky“, každá z nich obvykle uchovává 1 bit informace. Jedna brána je kontrolní brána (CG-control gate) druhá je plovoucí brána (FG-floating gate) navzájem izolované vrstvou oxidu. Protože je FG izolovaný jeho izolační vrstvou oxidu, každé elektrony na něj přivedené jsou „uvězněny“ a tím pádem je uložena informace. Když jsou na FG nějaké elektrony, tak modifikují (částečně ruší) elektrické pole přicházející z CG, což modifikuje prahové napětí (Vt) buňky. Buňka je čtená umístěním specifického elektrického napětí na CG, elektrický proud pak buď teče, nebo neteče, a to v závislosti na Vt buňky, které je závislé na počtu elektronů na FG. Tato přítomnost nebo nepřítomnost elektrického proudu je přeložena na 1 a 0, představující uložená data.

Flash buňka je naprogramovaná (nastavená na specifickou hodnotu) spuštěním toku elektronů ze zdroje do odvodu. Přivedení velkého napětí na CG pak poskytne dostatečně silné elektrické pole pro jejich vysátí na FG. Pro vymazání flash buňky je velký napěťový rozdíl přiveden mezi CG a zdroj, což odvede elektrony pryč skrz kvantový tunel. Současné flash paměti jsou rozdělené do vymazatelných částí nazývaných buď bloky, nebo sektory. Všechny paměťové buňky v rámci jednoho bloku musí být vymazány současně [18].

1.2 RISC Architektura

RISC - zkratka pochází z anglického originálu *Reduced Instruction Set Computer*, přeloženo: počítač s *redukovanou* instrukční sadou.

Během 70. let 20. století vědci ukázali, že většina programů prováděných na tehdejších počítačích využívala pouze malou část ze všech dostupných strojových instrukcí procesoru (tehdejší překladače nedokázaly efektivněji využít všech instrukcí). Také složitý přístup do paměti zpomaloval provádění operací. Z toho vyplynulo, že složitější operace (mikrokód) efektivněji vykoná posloupnost jednodušších instrukcí, které lze provádět s vyšší frekvencí.

Shrnutí typických rysů RISC procesorů:

- procesor komunikuje s pamětí po sběrnici
- redukovaná sada strojových instrukcí obsahuje hlavně jednoduché instrukce
- délka provádění jedné instrukce je vždy jeden cyklus
- mikroinstrukce jsou HW implementovány na procesoru tím je zvýšena rychlost jejich provádění
- využívají řetězení instrukcí

Superpočítač CDC 6600 navržený v roce 1964 byl prvním z RISCových strojů. Jeho CPU měla 74 operačních kódů (tj. částí instrukcí), v porovnání se 400 u 8086.

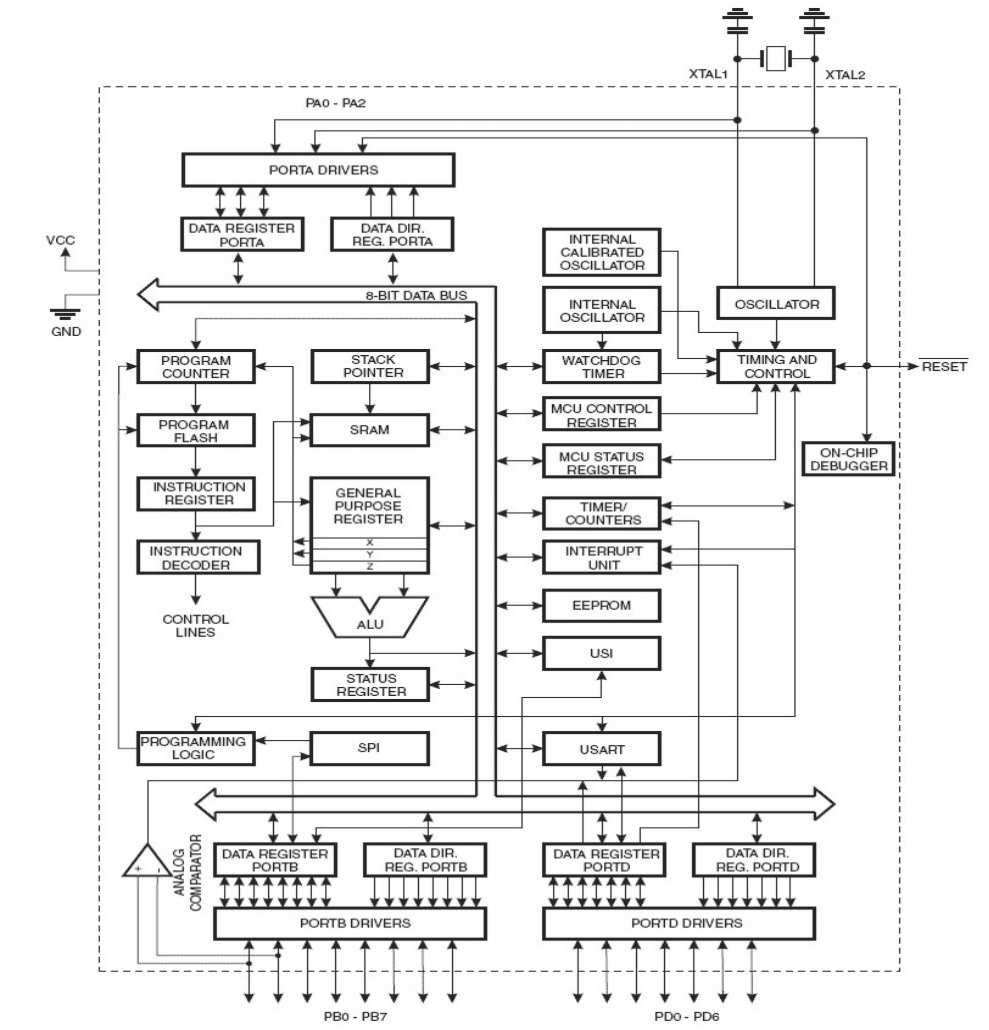
V dnešní době je prakticky každý moderní procesor založen na architektuře RISC, přestože pro ně mnohdy neplatí její základní charakteristiky – instrukční sady jsou rozšířeny o speciální povely pro práci s multimédii (MMX, SSE, 3DNow!), a instrukce trvají různě dlouhou dobu. Na druhou stranu ale masivně využívají pipelining – instrukce jsou načítány až 31 kroků dopředu a průběžně distribuovány mezi výkonné jednotky. Výstupy jsou poté řazeny tak, aby byl zachován sled jejich postupného zpracování [18].

Mezi nejznámější výrobce procesorů RISC patří IBM (např. řada PowerPC), Intel (většina jeho procesorů je ale řazena mezi CISC, nebo označována jako tzv. „post-RISC“) a Sun Microsystems (např. řada Sparc).

1.3 AVR-Rodina

Jednočipový mikropočítač (MCU). Procesor s univerzálním jádrem, s kterým jsou současně zaintegrovány základní periferní obvody, takže je schopen samostatné funkce. Za průkopníky v této kategorii můžeme považovat 8bitový procesor Intel i8051, který poprvé integroval všechny základní periferie (jádro procesoru, paměť RAM, EEPROM, čítače a časovače) na jediném čipu a 16bitový technologický procesor Siemens SAB 80C166, který poprvé integroval A/D převodníky, komunikační linky a masivní systém čítačů/časovačů/přerušení (následníky řady 80166 dnes vyrábí Infineon (řada C167 a C166 SV2) a SGS Thomson (řada ST10)).

AVR je označení pro rodinu 8bitových mikročipů typu RISC s harvardskou architekturou od firmy Atmel. Za zrodem AVR stojí studenti Alf-Egil Bogen a Vegard Wollan z Norského technického institutu. Na trhu se tyto mikroprocesory začaly objevovat od roku 1997.



Obr. č. 2 – RISC Architektura

Základní typy

1. ATtiny – využívají se v jednoduchých a malých elektronických obvodech
   1.1. Paměť flash pro uložení programu 1 – 8 kB
   1.2. Pouzdro 6 – 32 pinů (orientačně)
   1.3. Omezená sada integrovaných rozhraní

2. ATmega – výkonné mikročipy, mají JTAG[2] rozhraní, větší flash RAM, více integrovaných rozhraní.
   2.1. Paměť Flash pro uložení programu 4 – 256 kB

[2] Joint Test Action Group je standard definovaný normou IEEE 1149.1. Jedná se o architekturu pro testování plošných spojů, programování flash pamětí apod.

2.2. V pouzdrech 23 – 100 pinů (orientačně)
2.3. Rozšířená sada instrukcí (instrukce násobení a instrukce pro přístup k větší programové paměti)
2.4. Široká sada integrovaných rozhraní

Druhy pamětí

Paměti integrované na čipu jsou typu Flash (programová paměť), EEPROM (trvalá paměť dat), SRAM (paměť dat) a jsou všechny uvnitř čipu AVR.

Programová paměť (Flash)

Instrukce programu jsou uloženy v paměti Flash (10000x přepisovatelné), uchovávající obsah i po vypnutí napájení. Velikost Flash bývá uvedena v označení součástky (např. řada ATmega32x má 32 kB Flash). Nelze použít vnější paměť pro program, veškerý kód prováděný jádrem AVR musí být uvnitř Flash.

EEPROM

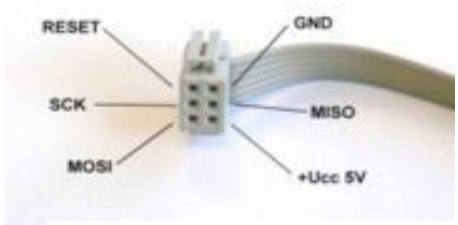
Téměř všechny AVR mikrokontroléry mají interní elektricky mazatelnou programovatelnou paměť (EEPROM). Stejně jako Flash i EEPROM uchovává svůj obsah i po vypnutí napájení (garantovaný počet přepsání je 100000x). Používá se pro uložení různých nastavení za běhu mikrokontroléru.

Programování MCU

K naprogramování AVR mikroprocesoru je nutný AVR programátor a příslušný software pro kompilaci programového kódu do strojového kódu. Nejčastěji se používá nástroj vyvinutý firmou Atmel a to program *„AVR studio“*. V tomto programu lze pak tvořit za pomocí GNU asembleru, C/C++ program který po kompilaci je skrze programátor (např. STK500) nahrán do programové paměti mikroprocesoru.

K naprogramování mikroprocesoru je možno využít několik rozdílných programátorů, nejčastěji však:

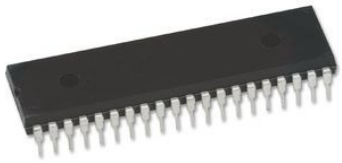
- STK200
- STK500
- AVR Prog
- USBasp
- The Dragon (USB)
- JTAG (IEEE 1149.1)



Obr.č. 3 – Možné rozložení programovacích pinů

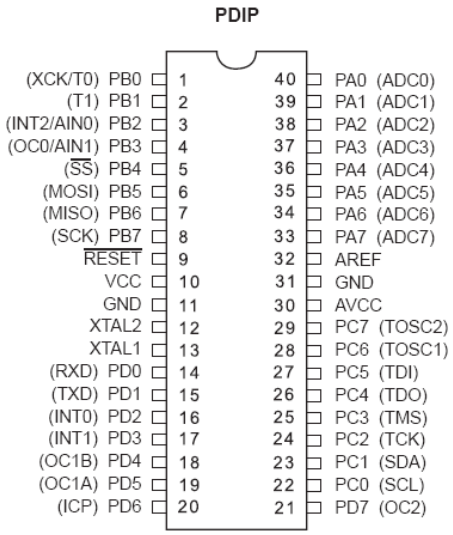
2. ATmega32

Pro základ celé konstrukce byl zvolen 8-bitový mikroprocesor firmy ATMEL v pouzdře DIL40 – Atmega32L-8PU



Obr. č. 4 – pouzdro DIL40

2.1 Základní (katalogové) parametry



Obr. č. 5 – Rozložení vývodů ATmega32 – pouzdro DIP40

- instrukční soubor obsahuje 131 instrukcí
- 32 8bitových registrů
- čtyři 8bitové vstupní/výstupní porty
- hodinový kmitočet 0-8MHz výpočetní výkon 8MIPS
- napájecí napětí 2,7 – 5,5 V
- programová paměť typu flash 32kbytů, až 10 000 cyklů přepsání
- 2 Kb datové SRAM Paměti
- 1 Kb EEPROM 100 000 cyklů přepsání
- programování pomocí SPI nebo JTAG rozhraní
- 2x 8bitový čítač, 2x 16bitový čítač, watchdog
- 4x PWM kanál
- analogový komparátor
- 8x 10bitový A/D převodník
- vestavěné rozhraní : USART, SPI, I2C, micro-wire
- interní RC oscilátor (možnost softwarové kalibrace)
- spotřeba < 1µA ve Stand-by režimu

2.2 Jádro, popis registrů

V konstrukci AVR mikroprocesorů je použita hardvardská architektura tedy paměť dat a programu jsou vzájemně odděleny. Data zapsané do paměti dat není možné chápat jako instrukci, ale pouze jako operandy při vykonávání programu z toho plyne větší bezpečnost při programování a větší výpočetní výkon. Mikrokontroléry v tomto provedení provádí i složité instrukce během jediného hodinového cyklu [3].

Mikrokontrolér disponuje čtyřmi vstupně-výstupními porty označenými PA až PD. Všechny tyto porty mohou pracovat obousměrně s výstupním proudem až 20 mA (buzení LED). Tyto porty jsou sdíleny se zabudovanými hardwarovými periferiemi například USART, I2C atd. v případě jejich užití není již možno požít příslušný vývod portu jinak. Dále obvod disponuje zabudovaným kalibrovatelným RC oscilátorem s možností připojit vnitřní děličku a získat požadovaný kmitočet.

Registry mikroprocesoru

Registrové pole – skupina 32 registrů 8bitové délky, většina těchto registrů se stává vstupním nebo výstupním operandem programového sledu instrukcí. Toto definované registrové pole značně zjednodušuje zápis programu a vede k jeho rychlejšímu provádění (bližší popis funkce jednotlivých registrů je uveden v datasheetu k danému obvodu). Užití funkce záleží na konkrétní programové aplikaci.

PC (Program counter) čítač programu, tento registr uchovává adresu právě prováděné instrukce v paměti programu. Tento registr není přímo přístupný. Jeho obsah se mění o 1 s každým krokem programu. Při provádění skoku v programu je do tohoto registru zapsána adresa s cílem skoku v programu.

SREG (Status Register) registr příznaků (flags-vlajky) jednotlivé bity registu je možno programově číst a vyhodnocovat výsledky operací například zda došlo k přetečení registru po sčítání, zda je dělení se zbytkem atd.

SP (Stack Pointer) ukazatel vrcholu zásobníku, pokud program pracuje s řadou výsledků, které jsou ukládány do zásobníku je práce s nimi pomocí ukazatelů, neboť fyzický přesun je programově časově náročný.

Vstupně-Výstupní registry název pro speciální registry jež ovládají, řídí vestavěné hardwarové periferie. Mohou být například SREG, SPH, SPL

Toto je výčet nejpoužívanějších registrů. Jejich počet, funkce se liší s každým typem mikroprocesoru. Kompletní popis je vždy výrobcem uveden v manuálu k danému obvodu tzv. datasheet[3].

[3] Katalogový list (datasheet) je dokumentace součástky, ve které se nacházejí všechny potřebné údaje pro její aplikaci.Většina výrobců poskytuje katalogové listy svých součástek na internetu zdarma a to v anglickém a čínském jazyce.

2.3 Instrukční soubor

Je soubor všech instrukcí (ATmega32 celkem 131), které mikroprocesor zná a umí je vykonat. *Instrukce* nejmenší jednotka programu k programátorskému užití. Jako příklad uvedu jednoduchou operaci sčítání obsahu dvou registrů, kdy výsledek je uložen v jednom z nich.

K této operaci potřebujeme dvě instrukce `mov` a `add` a dva uživatelské registry `registr1` a `registr2.`

`mov` – naplní registr danou hodnotou
`add` – sečte obsah dvou registru a výsledek uloží do jednoho z nich

samotná část programu v jazyce assembler může vypadat takto:

```
mov    registr1, 1         ; do registru1 zapíše číslo 1
mov    registr2, 3         ; do registru2 zapíše číslo 3
add    registr1, registr2  ; sečte a uloží do registr1 (v registr1 je číslo 4)
```

Spojením takovýchto jednoduchých operací do logických celků vzniká *program*. Obecně se dá uvažovat, že jeden řádek programu potřebuje k vykonání jeden hodinový cyklus, což je jedním z charakteristických rysů mikroprocesorů AVR.

Podle druhu vykonávané funkce můžeme instrukce rozdělit do několika skupin:

a) instrukce pro práci s registry procesoru (naplnění,porovnání,matematické operace)
b) instrukce skoků
c) instrukce operující s pamětí mikroprocesoru
d) instrukce pro práci s periferiemi mikroprocesoru
e) ostatní (obsluha přerušení, zjišťování a nastavování stavu procesoru aj.)

Kompletní výčet a popis jednotlivých instrukcí je vždy specifický pro konkrétní mikroprocesor a je popsán výrobcem v manuálu (datasheet).

3. Vývojové nástroje

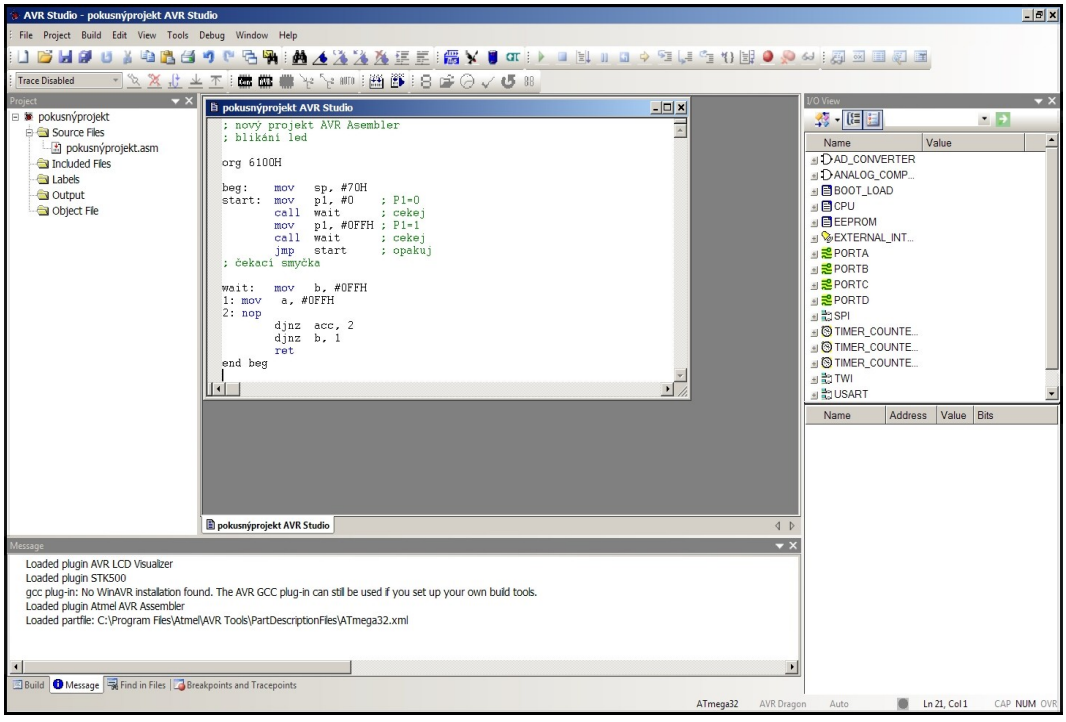
Pro práci a vývoj aplikací s mikroprocesory AVR jsou přímo určeny softwarové a hardwarové nástroje.

Softwarové nástroje podle svého druhu umožňují psát aplikace v různých programovacích jazycích (Assembler, C/C++, Pascal, Basic atd.)

Hardwarové nástroje jsou většinou koncipovány jako základní desky obsahující podpůrné obvody pro programování a ladění mikroprocesorových aplikací.

3.1 AVR Studio

Jedná se o integrované vývojové prostředí podporované přímo výrobcem obvodů AVR, pro vývoj programů s mikroprocesory. Aplikace je možné psát v programovacím jazyce *Assembler* s možností integrace překladačů jazyka C/C++. Pro snadnější práci s obvody obsahuje rovněž simulátor procesorů AVR a přímo podporuje základní druhy ladících nástrojů ATMEL.



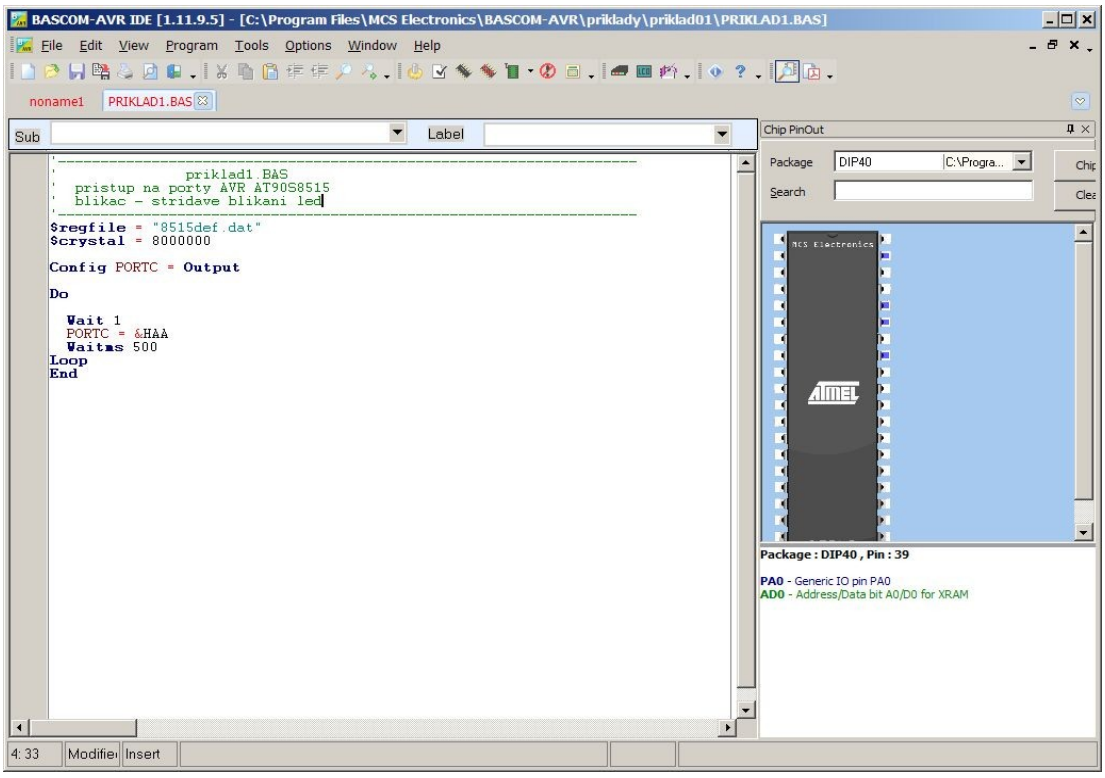
Obr. č. 6 – AVR Studio základní okno programu

Tento nástroj nebyl používán při vývoji programu ale pouze ve spojení s vývojovým kitem STK500, k jeho oživení a k případnému přeprogramování pojistek mikroprocesoru. Důvodem je podpora paralelního programování mikroprocesorů, které je vyžadováno při přeprogramování pojistek mikroprocesoru. (například nechtěné vypnutí možnosti sériového SPI programování). K samotnému vývoji a ladění programu bylo použito jazyka Bascom.

3.2 Bascom

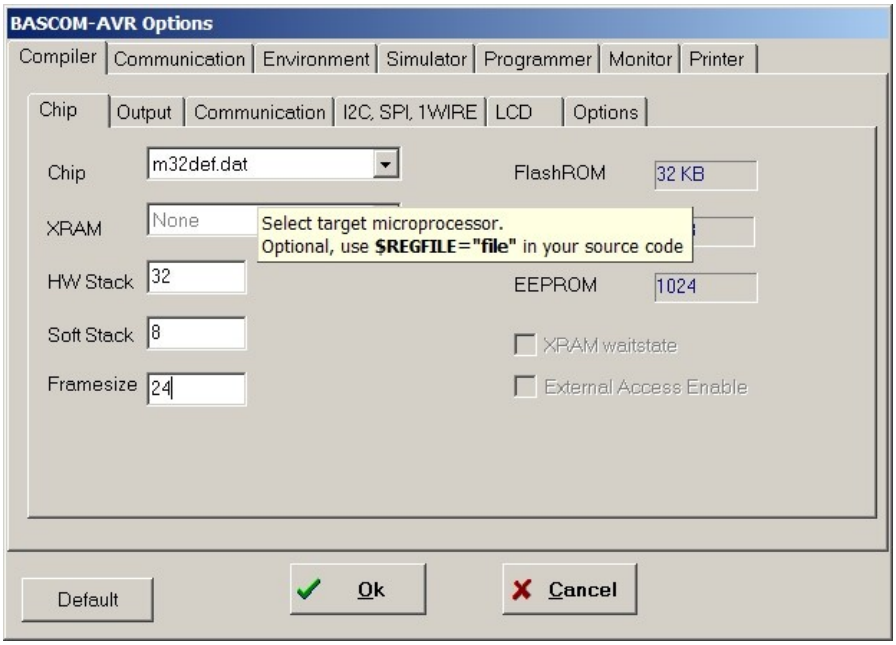
S rozšiřující se nabídkou a dostupností programovatelných obvodů se objevuje potřeba uživatelů bez hlubších programovacích znalostí mít jednoduchý programovací jazyk. V roce 1995 vytvořil Mark Alberts nový programovací jazyk pojmenovaný Bascom. Tento jednoduchý jazyk umožňuje i začátečníkům napsat program pro jednočipové mikropočítače a mikrokontroléry [7]. Proto je také Bascom vybrán k hlubšímu popisu v této práci a je v něm vytvořen ovládací program pro TCP/IP modul. Pro náš účel je použita demo verze, kterou nabízí firma MCS electronics. Tato verze je plně funkční jen velikost kódu je omezena na max. velikost 4kB, což je pro naší aplikaci plně dostačující.

Prostředí je napsáno jako klasický windowsovský program, tedy ovládání je plně intuitivní. Výchozím oknem programu je integrované vývojové prostředí (IDE), odtud jsou dostupné veškeré funkce programu (editace kódu, překlad, debugging, atd.)



Obr. č. 7 – Bascom základní okno programu (IDE)

Před samotným psaním programu je nutné správně zvolit typ mikroprocesoru se kterým budeme pracovat (Obr. č. 8) tím je zajištěno správné pojmenování použitelných registrů. Toto je důležité například při použití jiného typu mikroprocesoru. Není nutné složité přepisování programu. Například pro přímý přistup na portA mikroprocesoru, slouží v Bascomu příkaz : `portA` , ale každý mikroprocesor má tento port na jiné adrese, a tu bychom museli přepisovat na každém místě programu, tímto krokem je zajištěna jednoduchá kompatibilita programu pro více mikroprocesorů bez složitých úprav hlavního programu.



Obr. č. 8 – Bascom – výběr mikroprocesoru

Nyní je možný samotný zápis programu. Další výhodou tohoto prostředí pro široké použití jsou již předdefinované knihovny pro obsluhu běžných periferií mikroprocesoru (USART, I2C, atd.). V našem případě bylo pro komunikaci mikroprocesoru s okolím použita hardwarová sériová linka USART. Pro nejzákladnější obsluhu této linky slouží v Bascomu pouze dva příkazy:

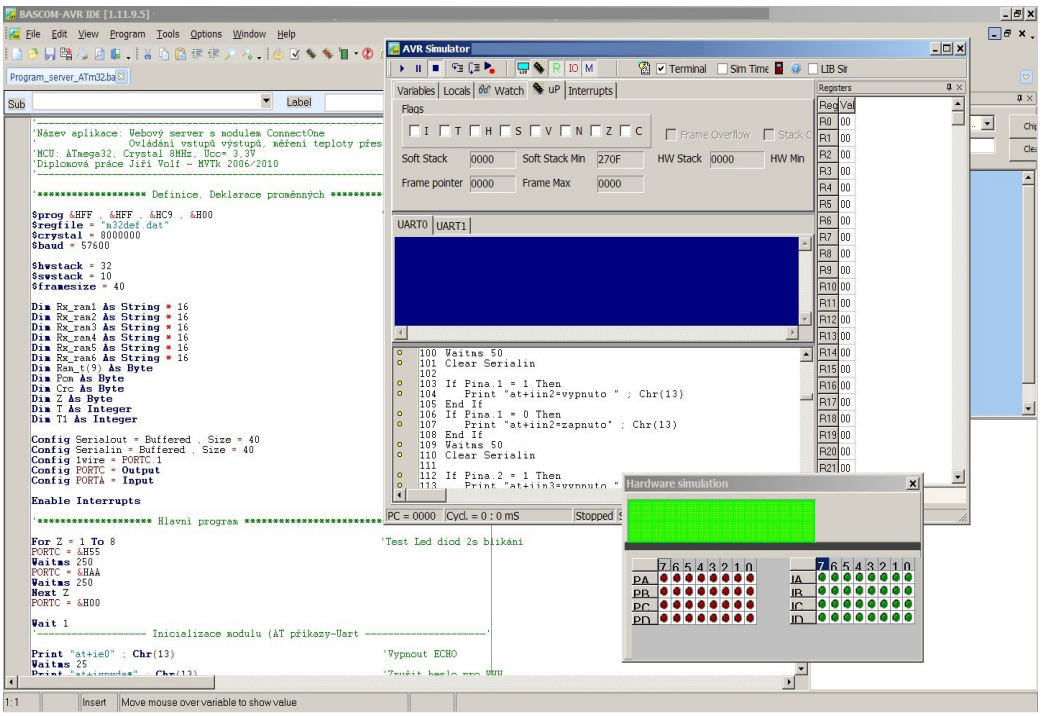
`Print` = vysílání Tx
`Input` = příjem Rx

Pokud je v programu zadán řetězec:

```
Print = "A"
```

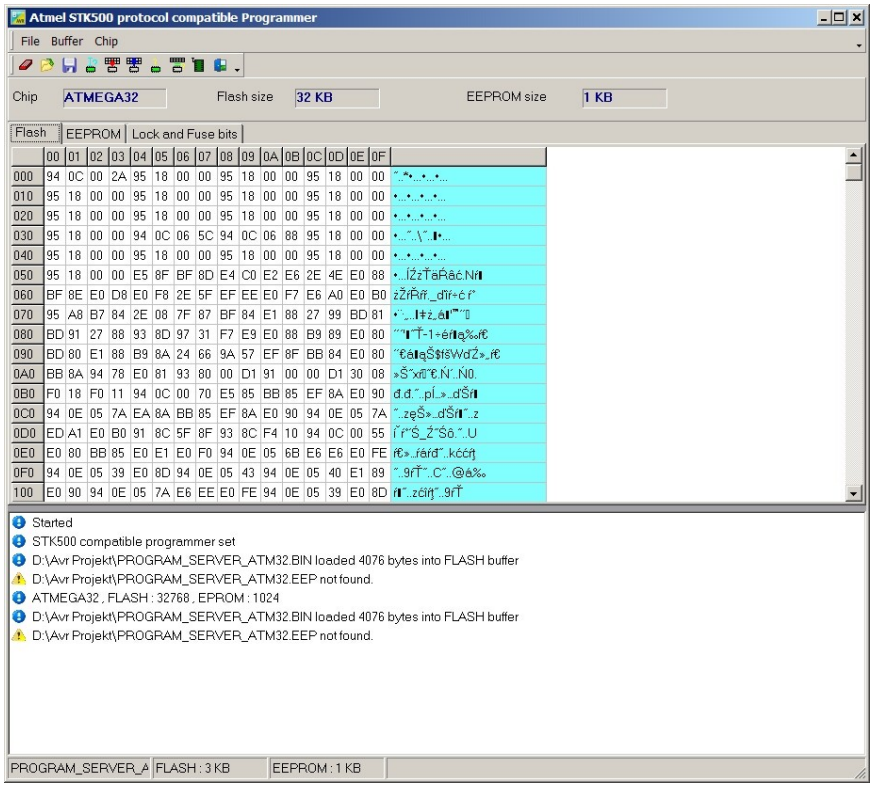
Je automaticky zajištěno vyslání znaku „A“ dle standardů sériové linky RS232. Právě tato vlastnost umožňuje použití jazyka Bascom pro náročnější aplikace bez hlubších programovacích znalostí. Samozřejmostí je také editace těchto knihoven v případě nestandardního specifického použití, nebo kompletní tvorba knihoven vlastních. Díky těmto vlastnostem je práce v tomto prostředí přehledná a rychlost vývoje větší. V případě velmi konkrétního požadavku na obsluhu je možné část programu psát v základním jazyce Assembler a tento kód vložit do hlavního programu pomocí direktivy `asm`.

Pro testování programu je možné využít simulátoru mikroprocesorů který je součástí prostředí. Pomocí grafického znázornění je možné program procházet po jednotlivých krocích a sledovat stav hardwaru (registry, porty, atd.)



Obr. č. 9 – Bascom – simulátor

Prostředí Bascom podporuje hardwarový vývojový kit STK500 a připojuje se pomocí sériové linky. Po připojení dojde k ověření zda osazený mikroprocesor odpovídá nastavení programu a po zkompilování je možné program nahrát do flash paměti mikroprocesoru.



Obr. č. 10 – Bascom – okno programátoru

3.3 STK500

Hardwarový vývojový kit STK500 je určen pro programování MCU firmy Atmel a následné ladění aplikací s nimi. Kit primárně spolupracuje se softwarovým prostředím AVR Studio, avšak jeho univerzálnost nevylučuje použití i s jiným softwarem například Bascom, WinAVR, Codevision atd. V tomto popisu se zaměřím na jeho použití s programovacím prostředím Bascom. Kit je opatřen osmi programovacími sokety pro vložení AVR obvodů a následnému naprogramování. Programování obvodů je možné pomocí dvou způsobů:

- AVR In-Systém Programing (ISP)
- High Voltage Programování (Paralelní)

AVR obvody je možné programovat i mimo základní desku kitu, a to přímo v aplikaci pomocí 6-ti nebo 10-ti žilového kabelu. Pro programování obvodů jiných pouzder než DIL je kit vybaven expansními konektory pro připojení redukcí (například pro pouzdro TQFP).

Obr. č. 11 – Sada vývojového kitu STK 500

Systémové požadavky:

- Procesor 486 a vyšší
- 16Mb RAM
- volné místo na disku dle použité sw aplikace
- Win95-Vista
- Comport 115200 baud (RS232) (popř. adaptér USB-RS232)
- DC zdroj 10-15V / 500mA

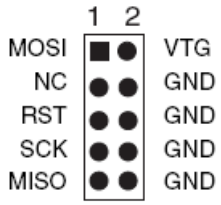
Základní vlastnosti STK500:

- Kompatibilní s AVR studio
- Rozhraní RS232 (programování, ovládání)
- Regulovatelný napájecí zdroj 10-15V
- Patice pro DIL8, DIL20, DIL28, DIL40, mikrokontroléry
- Reprogramování mikrokontroléry
- 8 tlačítek pro ladění aplikací
- 8 LED pro ladění aplikací
- AVR I/O porty vyvedeny na kolíkové konektory
- Přídavný RS232 port

Vývojový kit se k osobnímu počítači připojuje pomocí sériové linky RS232 (novější PC a notebooky již nejsou sériovým portem vybaveny a je nutné použít převodník USB-RS232). Ve spojení s Bascomem bylo vyzkoušeno několik modelů převodníků a se všemi fungovalo základní programování bez problémů, jen některé neumožňovali zpětnou verifikaci naprogramovaného obvodu, a programování pojistek. Toto bylo odstraněno ve spojení s AVR Studiem (blíže v popisu AVR studia).

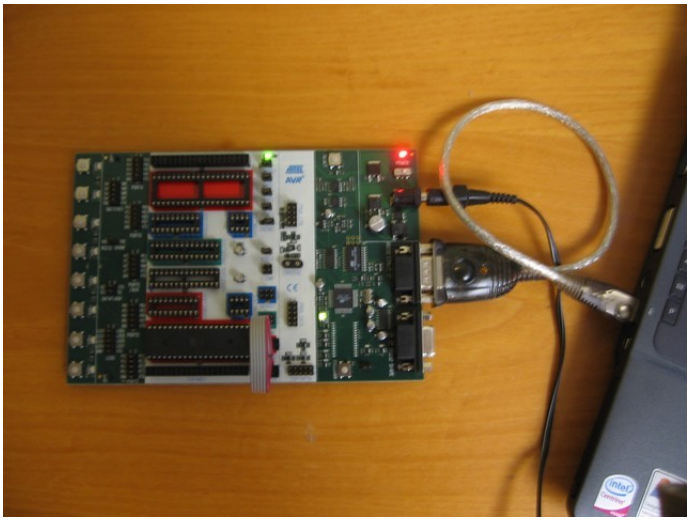
Obr. č. 12 – Převodník USB/RS232

Mikroprocesor byl programován mimo STK500 a to přímo v aplikaci pomocí 10ti žilového kabelu (Obr. č. 37.) Tato vlastnost kitu umožňuje programování a ladění aplikace téměř v reálných podmínkách.



Obr. č. 13 – Zapojení ISP konektoru na STK500

Programovací prostředí Bascom ve spojení s vývojovým kitem STK500 tvoří velmi výkonný dostupný a spolehlivý nástroj pro vývoj AVR aplikací. Během vývoje konstrukce nedocházelo k vážnějším potížím s užitím této sestavy.

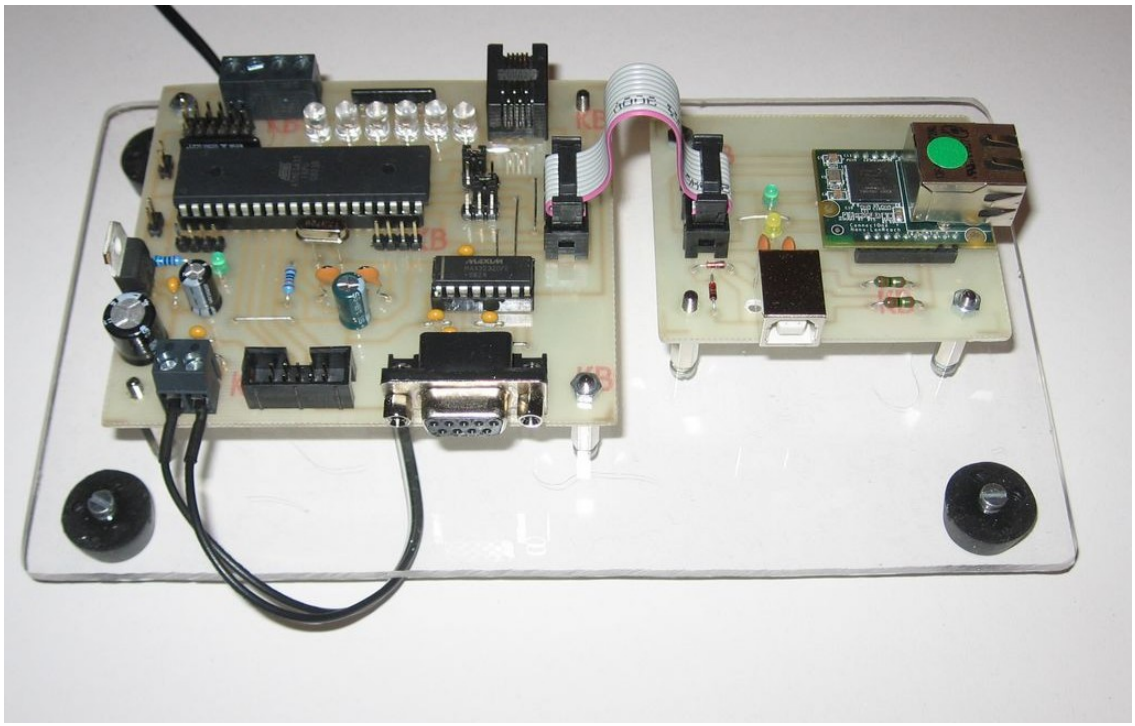


Obr. č. 14 – Fotografie připojení STK500 k PC

4. Konstrukce TCP/IP modulu

4.1. Popis konstrukce

Cílem praktické konstrukce TCP/IP modulu je ověření funkce výše popsaného vývojového kitu STK500 s programovacím prostředím Bascom včetně seznámení s IPmodulem Nano SocketLan.



Obr. č. 15 – Fotografie oživeného modulu

Modul je koncipován částečně jako vývojový prostředek pro práci s IPmodulem Nano SocketLan. Na základní desce je umístěn převodník RS232 / UART 3,3V pro komunikaci:

a) PC – MCU
b) PC – Nano SocketLan
c) MCU – Nano SocketLan

Nastavení komunikace je možné pomocí jumperů (popis níže). Dále deska obsahuje:

1- 6x Led 2mA (připojeno na Port.C přes rezistor 680 Ohmů)
2- 6x Jumper lišta (připojeno na Port.D rozpojeno=1, spojeno=0)
3- Konektor RJ pro jednoznačné připojení např. I2C, 1-wire, atd.

4- Svorkovnice pro připojení analogových/digitálních signálů
5- Kolíkové lišty pro připojeni analogových/digitálních signálů
6- MLW konektor pro přímé programováni MCU pomocí ISP
7- Zdroj 3,3V-1A

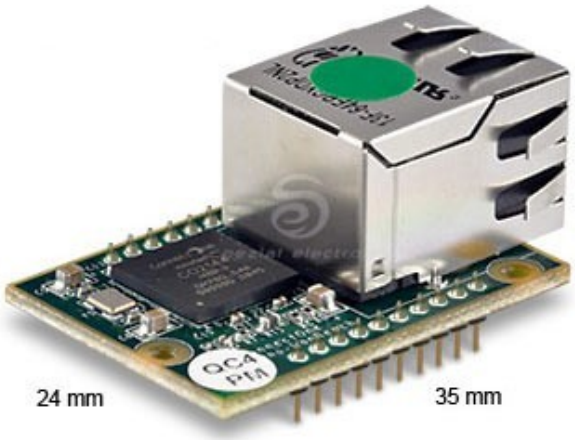
Deska modulu (montážní) obsahuje patici pro IPmodul, USB konektor pro možnou konfiguraci modulu, indikační LED (napájení - zelená, síťová komunikace - oranžová)

Obě desky jsou vzájemně propojeny pomocí 10-ti žilového plochého kabelu. Napájení je řešeno pomocí externího nestabilizovaného síťového zdroje 12V/1A, připojeného pomocí 2-pólové svorkovnice ARK. Pro pohodlnou manipulaci s oběma deskami je celý komplet umístěn na jednu společnou základní desku z plexiskla.

4.2 Nano SocketLan

IPmodul Nano SocketLan je moderní elektronická součástka (modul) umožňující připojení jakéhokoli zařízení komunikujícího po sériové lince do internetu. V konstrukci modulu je využito IPchipu CO2144. Obvod je integrován do pouzdra LFBGA-144 a obsahuje 32 bitové jádro ARM7TDMI a 256kB rychlou pamět SRAM. Dále je modul vybaven 2MB programovou Flash pamětí obsahující operačního systému iChipOS. Modul je určen pro připojení do klasické sítě ethernet typu 10/100BaseT [15].

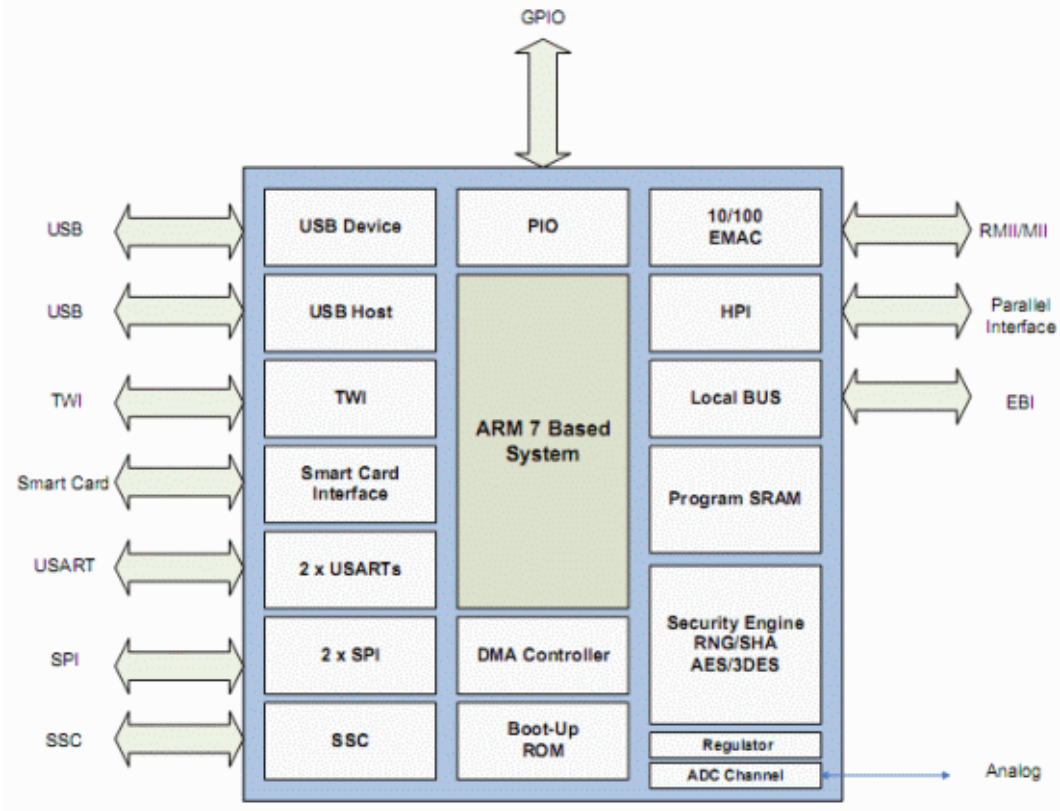
Nejsou potřeba žádné síťové ovladače či vyžadována složitá konfigurace modulu. Po jednoduché konfiguraci, je modul okamžitě připraven poskytnout své nadřazené aplikaci plnou konektivitu do sítě internet. Komunikace s okolím je možná prostřednictvím několika standardních komunikačních linek. K dispozici jsou navzájem oddělené linky typu **UART**, **SPI**, a **USB**.



Obr. č. 16 – IPmodul Nano SocketLan

Nano SockeLan je možné nakonfigurovat do následujících provozních režimů:

a) **SerialNET Serial to LAN Bridge** - transparentní můstek mezi sériovým USARTem a Ethernet LAN, tedy UART přes LAN, s vysokou rychlostí 3Mbps. Jedná se režim přenosu sériové linky přes LAN a tedy lze signály sériové linky (Rx, Tx, CTS, RTS) přenést přes internetovou síť do celého světa.
b) **Full Internet Controller mode** – jednoduchý procesor lze snadno připojit do internetu a využít bohaté možnosti LAN modulu jako např. E-Mail, FTP, SSL, zabudovaný web server atd. Lze použít s jakýmkoli hardware.
c) **PPP emulation** – umožňuje připojit existující aplikaci (např. s GPRS modemem) používající PPP, připojit se beze změny hardware nebo software přímo do internetové sítě.



Obr. č. 17 – Blokové schéma IPchipu CO2144

Podporované protokoly:

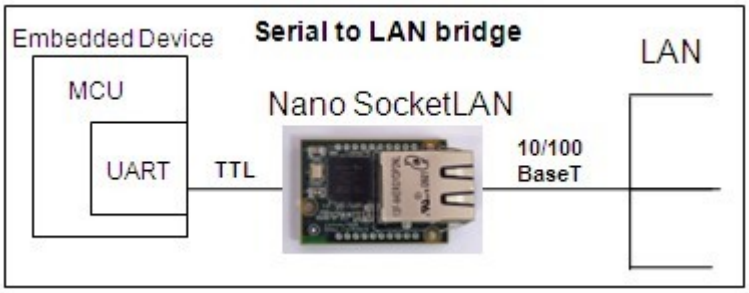
ARP, ICMP, IP, UDP, TCP, DHCP, DNS, NTP, SMTP, POP3, MIME, HTTP, FTP TELNET

Bezpečnostní protokoly: SSL3/TLS1, HTTPS, FTPS, RSA, AES-128/256, 3DES, RC-4, SHA-1, MD-5

Protokoly akcelerované hardwarem: AES, 3DES a SHA

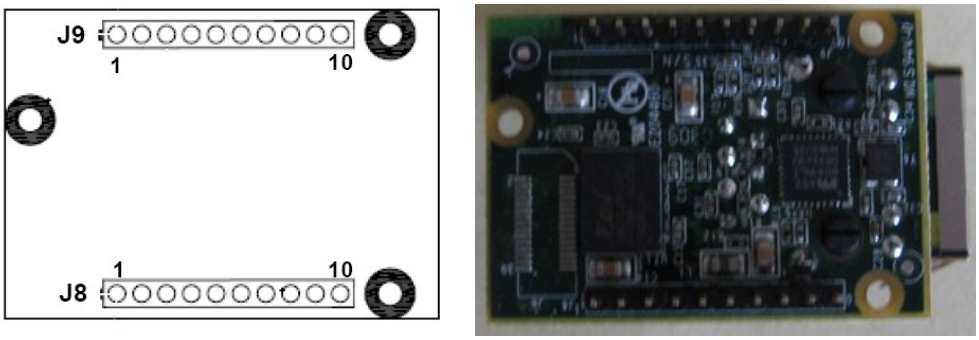
Počet současně otevřených socketů: 10

V konstrukci TCP/IP modulu byla ověřena funkčnost režimů **SerialNET** a **Full Internet Controller mode**.



Obr. č. 18 – Schematické znázornění režimu SerialNET

Vývody modulu jsou realizovány pomocí kolíkové lišty s roztečí 2mm.



Obr. č. 19 – Vývody modulu (pohled zespodu)

Pin:	*Název:*	*Typ:*	*Popis:*
1	GND	Nap.	
2	Vdd	Nap.	
3	RxD0	Vstup	USART 0 příjem
4	TxD0	Výstup	USART 0 vysílání
5	nCTS0	Vstup	
6	nRTS0	Výstup	
7	DATA_RDY	Výstup	Data k vysílání
8	MSEL	Vstup	HW reset
9	nRESET	Vstup	SW reset
10	ACT_LINK	Výstup	LAN Link LED

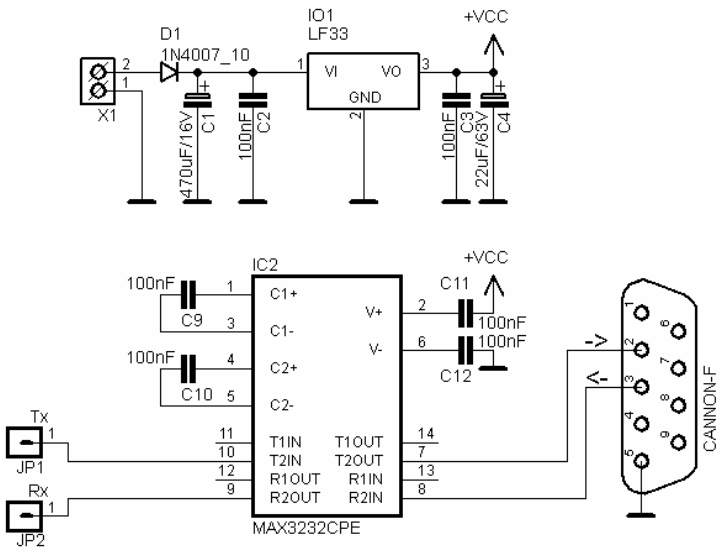
Tab. č. 1 – Vývody modulu-lišta J8

Pin:	*Název:*	*Typ:*	*Popis:*
1	nSPI1_CS	Vstup	SPI
2	SPI1_CLK	Vstup	SPI (max. 12 MHz)
3	SPI1_MISO	Výstup	SPI
4	SPI1_MISO	Vstup	SPI
5	SPI1_INT	Výstup	SPI
6	Readlines	Výstup	IPchip OK
7	DDM	Analog.	USB data -
8	DDP	Analog.	USB data +
9	SPEED	Výstup	LED 10/100M
10	GND	Nap.	

Tab. č. 2 – Vývody modulu-lišta J9

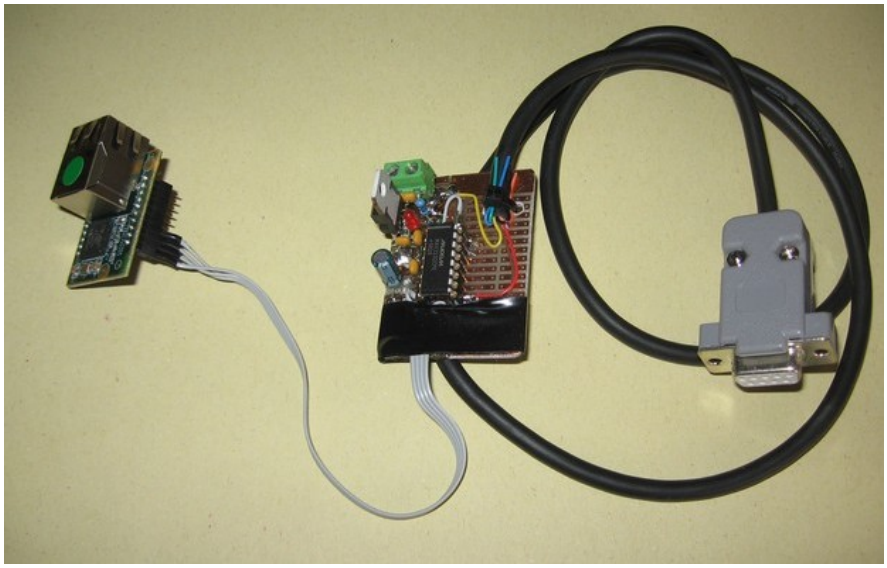
4.2.1 Nastavení

První nakonfigurování modulu je nutné pomocí komunikační linky RS232. celý modul pracuje v 3,3 voltové logice a není „5V tolerantní“. Proto je nutné pro připojení k PC použít jednoduchý převodník jehož základ tvoří integrovaný obvod MAXIM3232 v pouzdru DIL16. Obvod pracuje na principu nábojové pumpy a převádí napěťové úrovně sériové linky na 3,3V logiku.



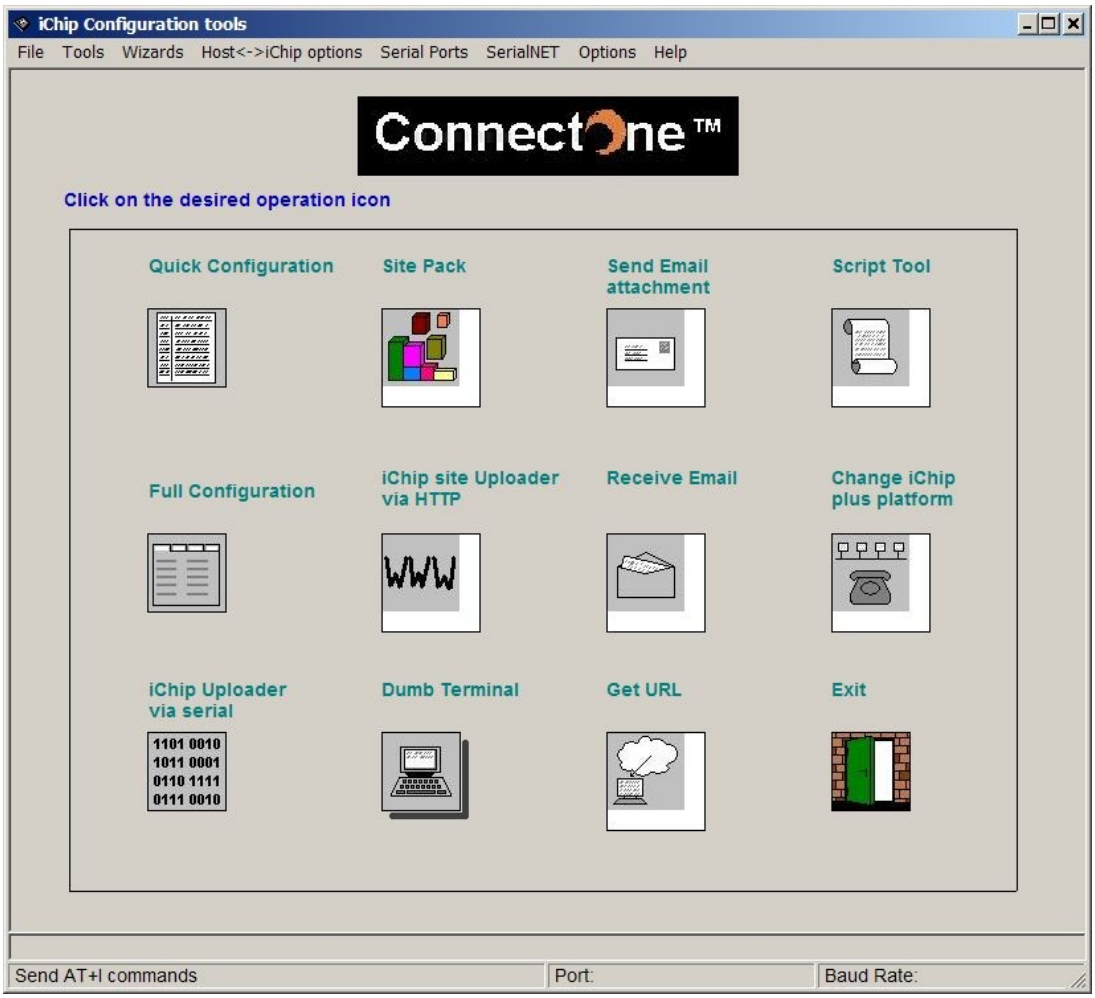
Obr. č. 20 – Schéma převodníku RS232 / USART 3,3V

Pro první zapojení byl celý převodník realizován na univerzální DPS.



Obr. č. 21 – Fotografie převodníku RS232 / USART 3,3V

Konfigurace se provádí pomocí softwarové aplikace iChipConfig, která je k dispozici na stánkách výrobce modulu. Po jednoduché instalaci a spuštění dojde k zobrazení výchozího okna. Modul je také možné kompletně konfigurovat pomocí textového terminálu a sady AT příkazů (viz. Příloha 5).



Obr. č. 22 – Výchozí okno iChipConfig utility

Výchozí okno konfiguračního programu obsahuje více konfiguračních nástrojů:

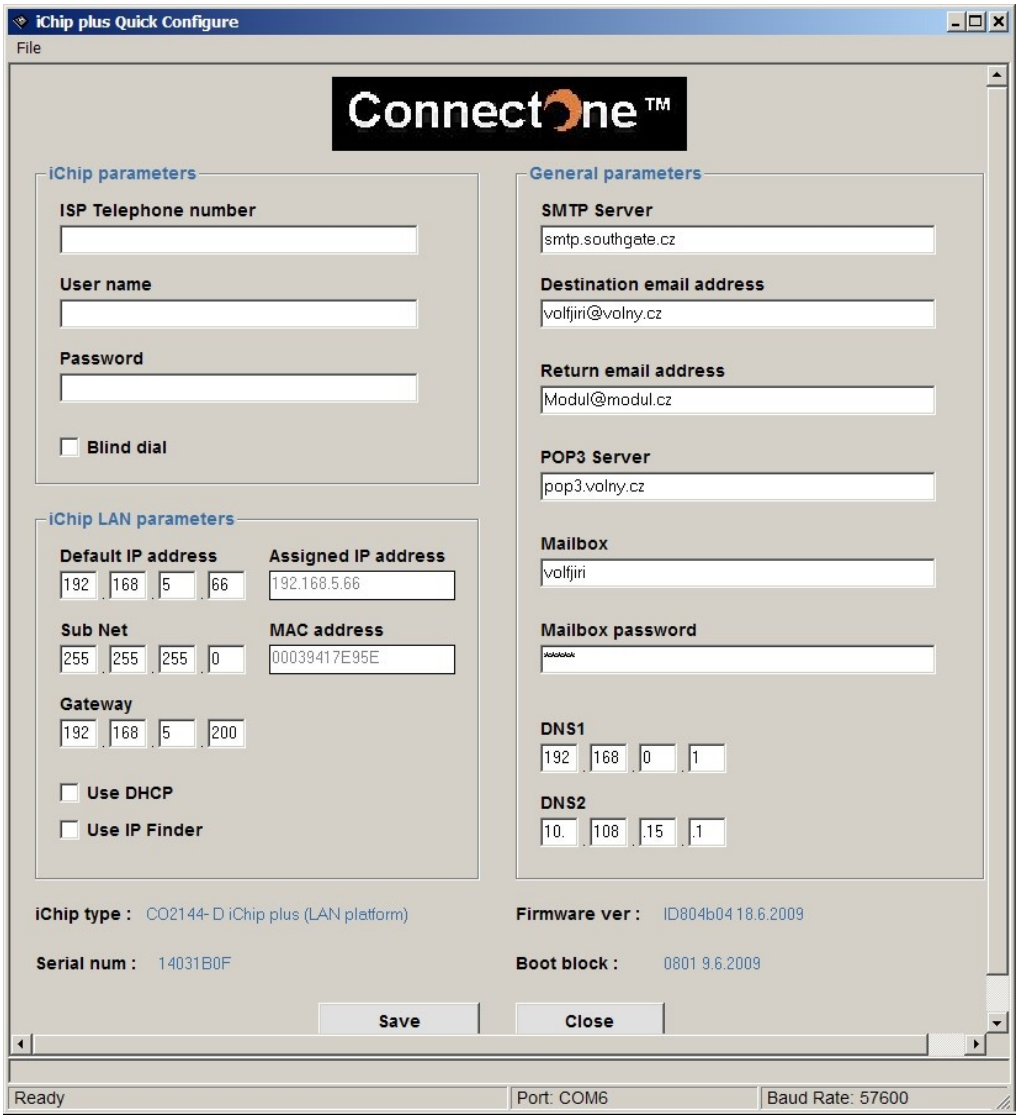
- Quick Configuration: slouží k rychlé konfiguraci modulu, síťových parametrů (IP adresa, brána DNS atd.)
- Full Configuration: kompletní nastavení modulu a režimů
- iChip Uploader via seriál: slouží k update vnitřního firmware přes USART
- Site pack: nástroj pro konverzi www stránek do jediného image souboru pro nahrání do modulu.
- iChip site uploader via http: slouží k nahrání www image
- Send Email attachment: při správné konfiguraci modulu umožní odeslat email s přílohou
- Recieve Email: přijme email pomocí protokolu pop3
- Get URL: zadání www stránky pro automatický update firmware

- Script editor: editor skriptů, které je možné spouštět při staru modulu (sekvence ATpříkazů)
- Change ichip plus platform: nastavení módu provozu

Horní lišta okna obsahuje průvodce pro rychlé nastavení módů provozů s nejběžnějšími parametry. Celé nastavení je možné uložit do souboru a nahrát do modulu přes konfigurační www stránky. Po správném připojení modulu k PC přes převodník úrovní, software automaticky detekuje modul a komunikační rychlost.

Obr. č. 23 – Stavová lišta s nalezeným modulem

Pro základní zpřístupnění modulu přes LAN síť byla použita volba „Quick-Configuration.“



Obr. č. 24 – Okno Quick Config

Pro nejzákladnější nastavení postačí zadat pouze IP adresu sítě, masku a bránu sítě.

V tomto případě takto:

IP adresa : 192.168.5.66
Maska : 255.255.255.0
Brána : 192.168.5.200 (není nutné při užití v neroutované síti)

Po této konfiguraci a připojení modulu do sítě pomocí standardního síťového UTP-cat.5. kabelu, modul okamžitě odpovídá na příkaz ping. Tímto je modul prvotně oživen a zapojen do sítě k dalšímu postupu.

Nyní je také možné ověřit komunikaci pomocí terminálu. Zvolíme „Dump Terminál“. Zde je možné modul obsluhovat a konfigurovat pomocí sady AT příkazů (kompletní seznam viz. Příloha 5).

Tvar AT příkazu musí začínat „AT+i“ a zakončen je znakem <CR> (enter). Tento znak ale většina terminálu odesílá automaticky po stisku klávesy enter. Navíc je přidán znak <LF> (nový řádek) tento ale pro vykonání příkazu není nutný. Tyto znaky se nezobrazují ale jsou linkou odesílány. Tedy skutečný tvar odeslaného řetězce vypadá takto:

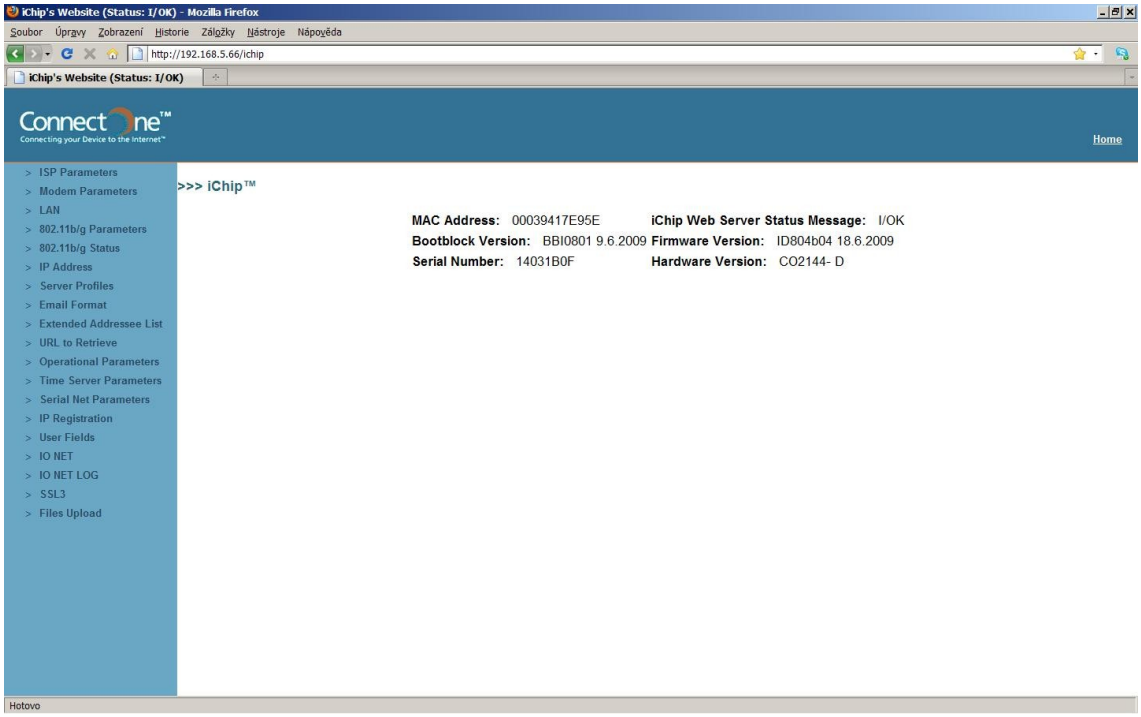
AT+i <příkaz> <CR><LF>

Skutečná odpověď modulu potom vypadá takto:

I/OK<CR><LF>

S tímto skutečným tvarem přijaté odpovědi je nutné počítat při návrhu programu pro MCU. Zvláště jsou-li znaky dekódovány jeden po druhém pomocí ukazatelů.

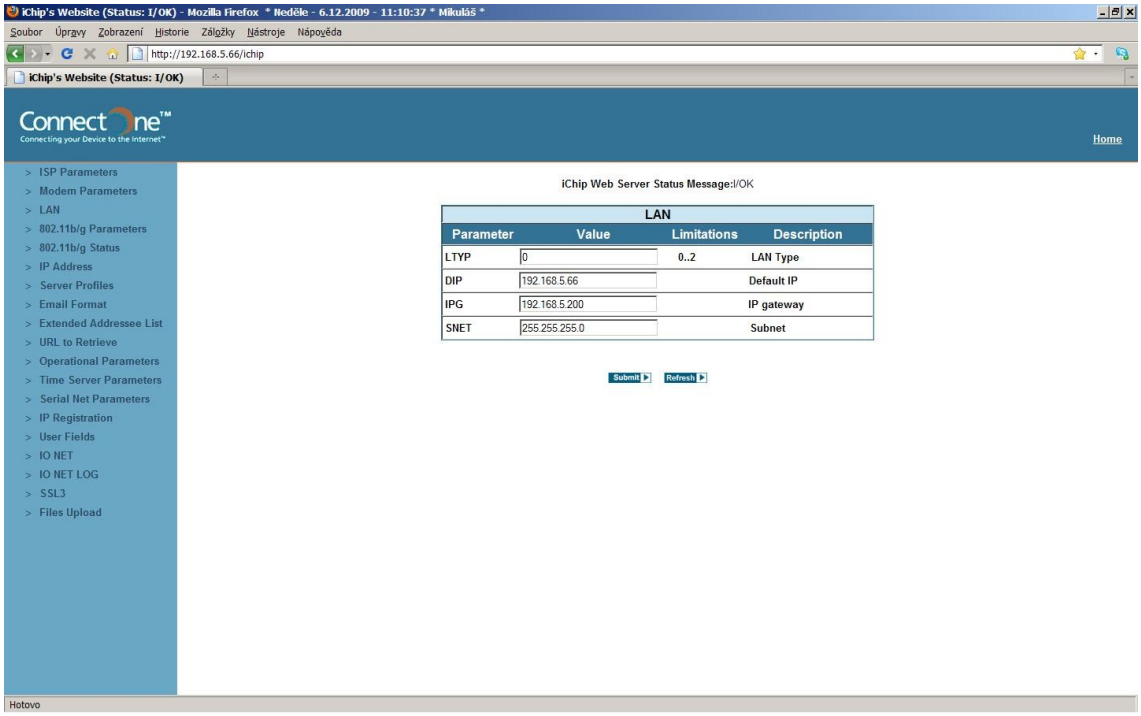
V okně terminálu nyní zadáme „AT+i“ a pokud je vše v pořádku musí následně modul vrátit odpověď „I/OK“. Nadále je možné modul konfigurovat pomocí Ichipconfigu nebo pomocí interního www serveru. V základním nastavení je tento server vypnut a je nutné ho aktivovat příkazem „AT+iwww“ po té již stačí v jakémkoli webovém prohlížeči zadat adresu „http://192.168... <ip adresa modulu>/ichip.html“ a dojde k načtení konfiguračních webových stránek. Na „http://192.168...<ip adresa modulu>“ jsou pak dostupné uživatelské webové stránky. Pokud požadujeme po zapnutí ne-standardní nastavení modulu, je možné ve v nástroji „script tool“ sestavit jednoduchou sekvenci příkazů která bude nahrána hned po zapnutí modulu.



Obr. č. 25 – Konfigurační webové stránky

Obr. č. 26 – Detail menu pro nastavení

Pro nastavení základních síťových parametrů vybereme z menu položku LAN a můžeme zadat parametry. Jedná se o analogii k nastavení pomocí iChipconfigu k volbě „Quick configuration“.



Obr. č. 27 – Podstránka nastavení LAN parametrů

Tyto parametry je také možné zadat pomocí terminálu a AT příkazů např.

- zadání IP adresy:

  AT+iDIP=“192.168.5.66“ (tímto je nastavena IP adresa)

  Odpověď „I/OK“

- kontrola zadání:

  AT+iDIP?

  Odpověď „I/(192.168.5.66)“

  v případě chyby:

  Odpověď „I/ERROR“

Celkem je tedy možné modul konfigurovat třemi způsoby:

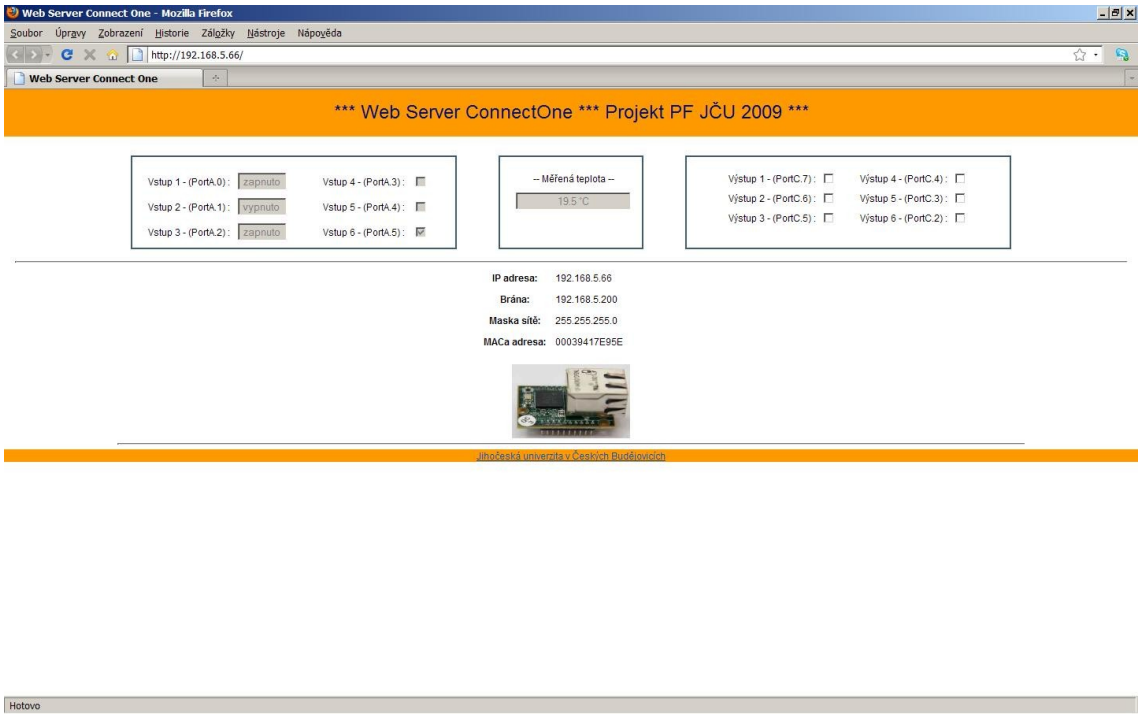
a) pomocí USART a utility iChipconfig

b) pomocí webového rozhraní

c) pomocí USART vhodně naprogramovaným MCU

4.2.2 Vytvoření, nahrání webových stránek

Modul disponuje vnitřní 256Kb velkou EEPROM pamětí pro uložení uživatelských webových stránek. Tato vlastnost byla zvolena k ověření při praktické konstrukci celého modulu.



Obr. č. 28 – Uživatelské webové stránky

Pro přehlednost jsou vytvořeny velmi jednoduché webové stránky ve free verzi wysiwyg editoru, ale stránky je možné rozšířit o další prvky:

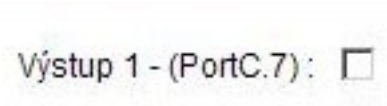
- link na externí stránky
- obrázky
- grafiku
- java aplety
- wap atd.

Toto uživatelské prostředí v modulu umožňuje velmi praktickou vlastnost definování vlastních proměnných jejichž hodnotu je možné měnit např. pomocí USART a připojeného MCU a taktéž naopak. Názvem proměnné může být libovolný text o maximální délce 256 znaků a uzavřen do „~“. Velikost webových stránek názvů a obsahů proměnných nesmí překročit právě 256Kb. Přeprogramování stánek lze proměnné velmi jednoduše vytvářet uvedeným způsoben.

Uvedu příklad programování a obsluhy checkboxu (zaškrtávací pole) k ovládání výstupních led diod.

Tento html kód vytvoří na stránkách checkbox s názvem proměnné „out1“.

```
<td>Výstup 1 - (PortC.7) :</td>
<td><input type="checkbox" name="out1" ~out1~ value="checked"
     onchange="document.outform.submit()"></td>
```



Obr. č. 29 – Naprogramovaný checkbox

Po spuštění kódu se v paměťovém prostoru modulu vytvoří prostor pro proměnnou s ná-zvem „out1“, která může nabývat dvou stavů dle zaškrtnutého pole a to:

a) out1=“checked“ (zaškrtnuto)
b) out1=“ “ (nezaškrtnuto)

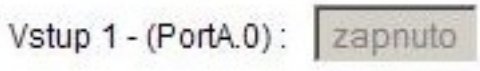
Nyní je možné například pomocí připojeného mikroprocesoru po sériové lince v pravidelných cyklech odesílat dotazy na stav proměnné „out1“ a to takto:

dotaz na stav (hodnotu) : AT+iout1?
odpověď : “ “ nebo “checked“

Odpověď je možné pomocí jednoduchého podmiňovacího cyklu programově vyhodnotit a rozsvítit příslušnou LED. V případě odeslání hodnoty zpracované mikroprocesorem a následné zobrazení na www stránkách je postup obdobný:

Tento html kód vytvoří na stránkách textové pole, které zobrazí hodnotu proměnné s názvem „in1“.

```
<td>Vstup 1 - (PortA.0) :</td>
 <td><input type="text" size="6" name="in1" value="~in1~"
      disabled style="text-align: center"></td>
```



Obr. č. 30 – Naprogramované textové pole

Nyní pomocí připojeného mikroprocesoru po sériové lince v pravidelných cyklech odesíláme hodnotu proměnné „in1“ v závislosti na stavu PortA.0 takto:

a) PortA.0 = 0 – MCU odešle : AT+iin1=“vypnuto“
b) PortA.0 = 1 – MCU odešle : AT+iin1=“zapnuto“

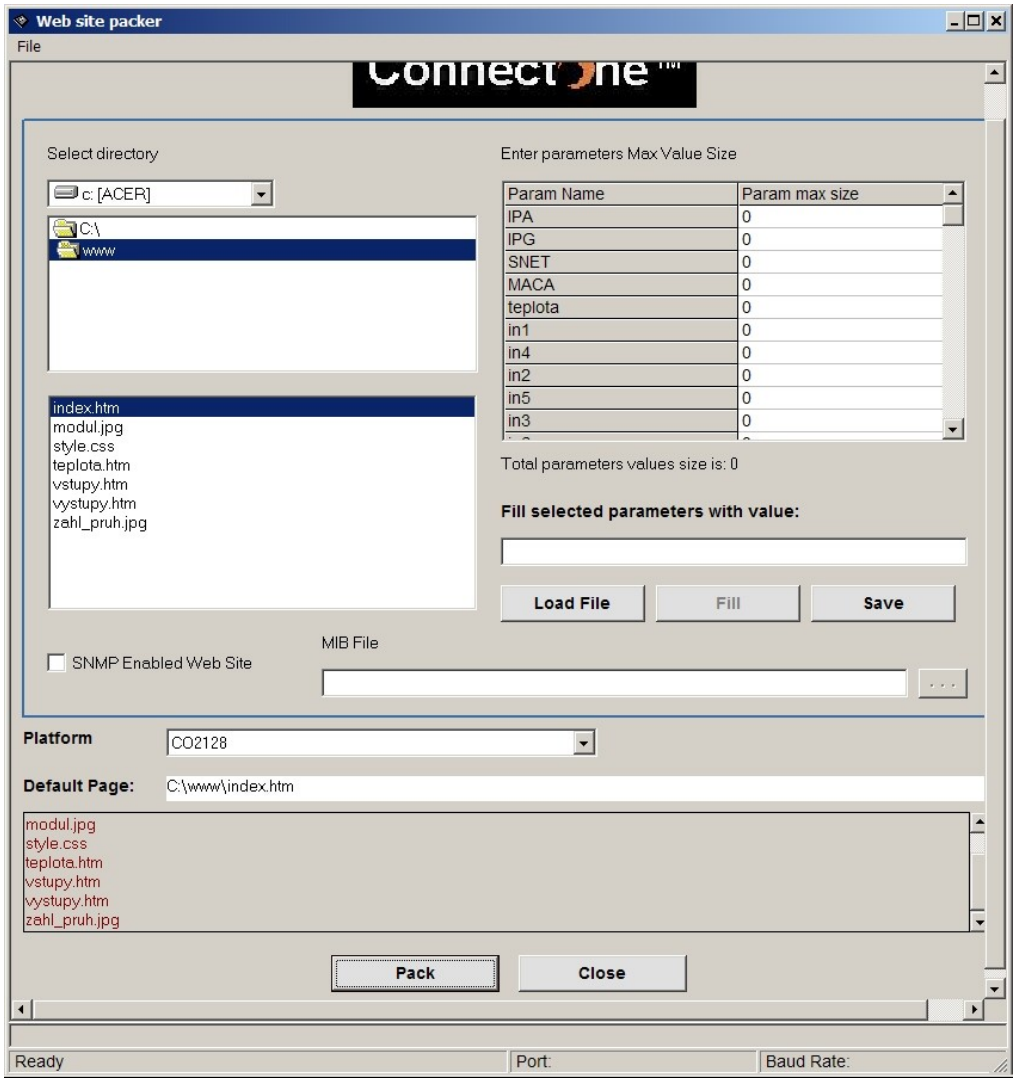
Dále je možné na stránkách zobrazit nastavené parametry modulu například IP adresu:

```
td align="center"><strong>IP adresa: ~IPA~ </strong></td>
```

IP adresa: 192.168.5.66

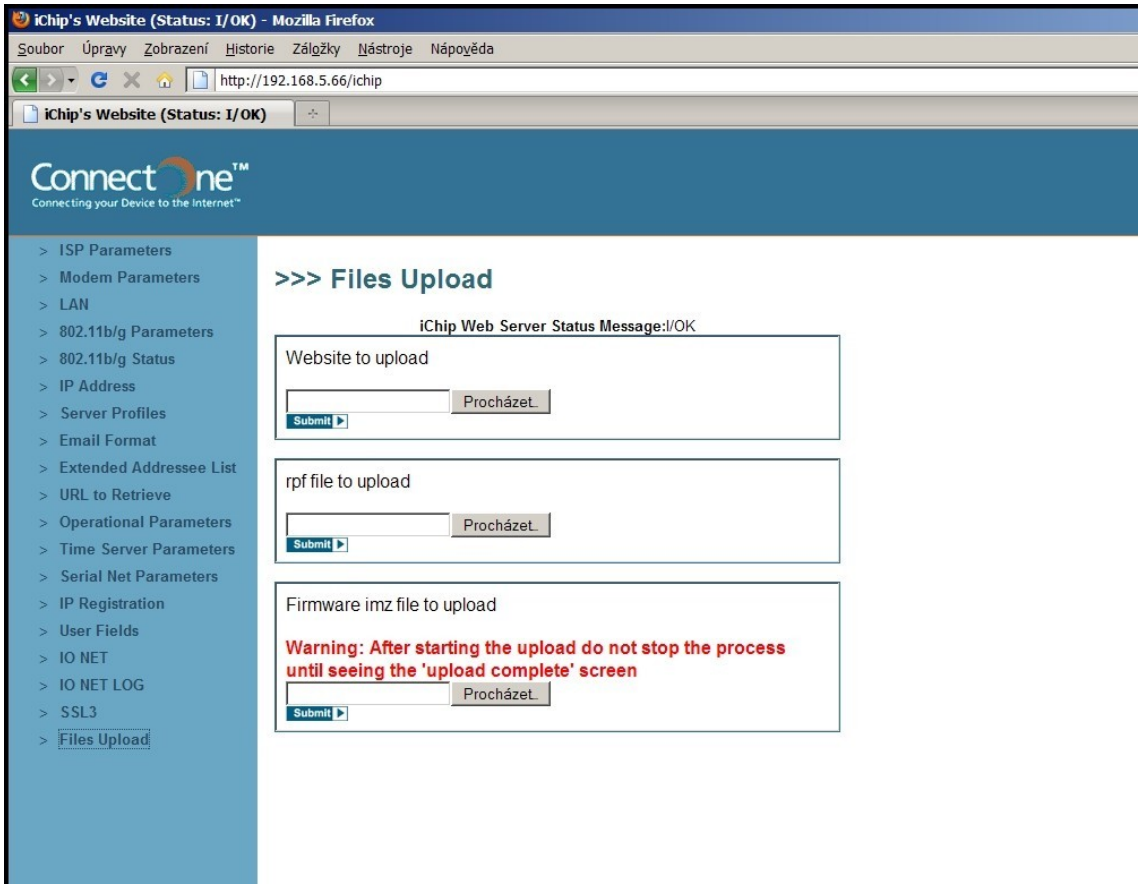
Obr. č. 31 – Výsledek kódu

Pomocí těchto jednoduchých základů je možné vytvořit velmi silnou aplikaci pro řízení nebo monitorování nejrůznějších zařízení přes internetovou síť. Pro větší názornost je program MCU (blíže v jeho popisu) doplněn o měření a zobrazení teploty pomocí čidla firmy Dallas DS18S20 komunikujícího po jednovodičové sběrnici 1-wire. Celé webové stránky je před samotným nahráním do modulu nutné zkompilovat do jediného image souboru pomocí utility obsažené v iChipconfig a to „Site pack“



Obr. č. 32 – Site pack (kompilátor webových stránek)

V okně kompilátoru zadáme cestu k souborům webových stránek. Výchozí stránka musí mít název index.htm. V položce „Platform“ vybereme typ použitého chipu CO2128 nebo novější CO2144. V okně „Parameters max value“ přidělíme jednotlivým proměnným velikost hodnoty (v bitech) nyní stiskneme tlačítko „pack“ pokud velikost stránek a proměnných nepřekročí 256Kb bude vygenerován soubor website.img a ten je možné nahrát do modulu pomoci iChipconfig nebo pomocí konfiguračního webového rozhraní.



Obr. č. 33 – Podstránka pro nahrání image souboru

Po restaru modulu a spuštění www serveru jsou na adrese: *„http:// (ip adresa modulu)“* dostupné uživatelské webové stránky viz. Obr.č. 28

4.3 Popis programu MCU

Celý program je napsán v programovacím prostředí Bascom a je určen pro mikroprocesor ATmega32 taktovaným externím krystalem 8 MHz.

Program je rozdělen do tří bloků:

1- Deklarace a definice proměnných a podprogramů
2- Hlavní tělo programu
3- Podprogramy

V první části je definován typ mikroprocesoru , nastavení programovacích pojistek, použitý krystal, nastavení parametru USARTu (přenosová rychlost, vstupní a výstupní buffer) a alokace paměťového prostoru pro uživatelské proměnné.

Druhá část obsahuje tělo programu, v podstatě se jedná o skoky na podprogramy vykonávající hlavní funkci modulu. Celý děj je uzavřen do smyčky opakující se po 200 ms.

Třetí část obsahuje podprogramy obsluhující komunikaci mezi USARTem MCU a modulem Nano SocketLan.

a) podprogram sejme stav vstupů Port.A 0..2
   stav = „0“ odešle se „vypnuto“
   stav = „1“ odešle se „zapnuto“
   podprogram sejme stav vstupů Port.A 3..5
   stav = „0“ odešle se „ “
   stav = „1“ odešle se „checked“

Obr. č. 34 – Výsledek operace podprogramu

b) podprogram kontroluje stav checkboxu na uživatelských www
   odešle dotaz,
   odpověď : zaškrtnuto= „checked“ rozsvítí příslušnou LED
   nezaškrtnuto = „ “ zhasne příslušnou LED
   jednotlivé odpovědi jsou uloženy do Rx_bufferu a následným cyklem dekodovány (postupné čtení bufferu=nastavování výstupů)

Obr. č. 35 – Ovládání Checkoxu (zaškrtnutí rozsvěcuje příslušné LED)

c) podprogram pro komunikaci s DS18S20 po sběrnici 1-wire

   výsledná teplota je vypočítána z hodnoty přijaté z čidla a odešle se jako řetězec do modulu pro zobrazení na www

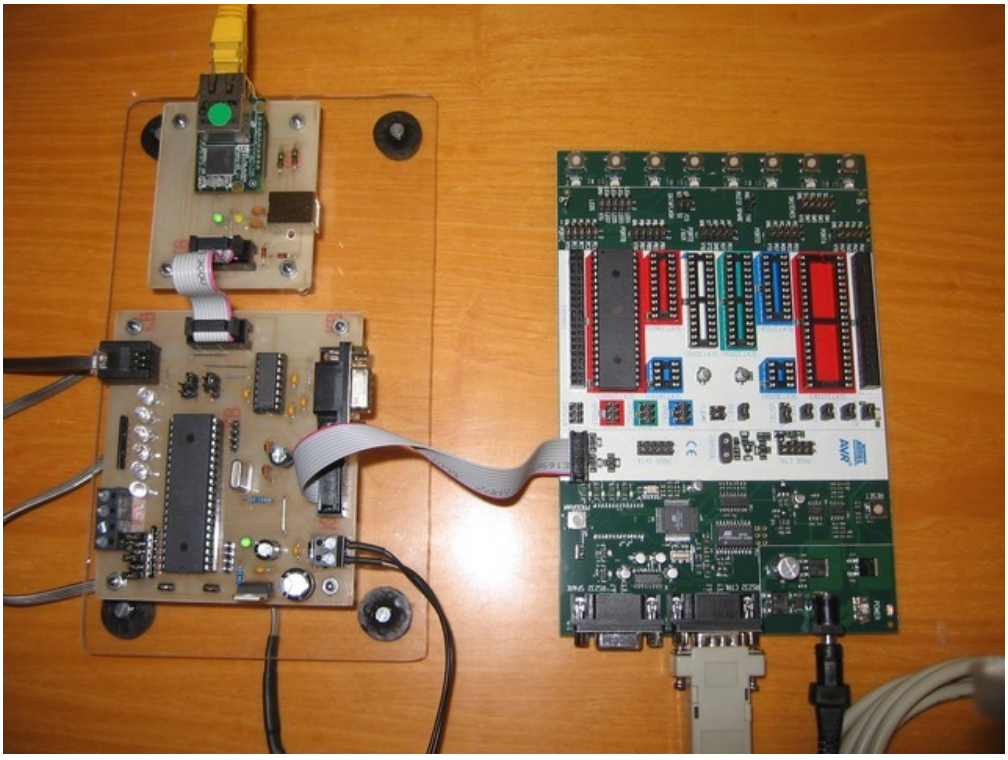
Obr. č. 36 – Zobrazení teploty

Celková velikost kódu je téměř 4 Kbyty což je maximální velikost kódu ve volné verzi programovacího prostředí Bascom. Celý program je možné řešit úsporněji například obsluha USARTu pomocí cyklů, ale pro názornost je zvolena tato přehlednější varianta.

4.4 Naprogramování, ladění MCU

Obr. č. 37 – Vývojová sestava (modul, STK500, PC, Bscom)

Mikroprocesor je programovaný přímo v modulu pomocí ISP rozhraní STK500. Propojení modulů je realizováno pomocí 10ti žilového kabelu. Po nahrání programu do MCU je tak možné okamžitě ověřit jeho funkci v reálných podmínkách.



Obr. č. 38 – Propojení modulu s STK500

V první fázi vývoje programu pro mikroprocesor byla ověřena funkce integrovaného USART rozhraní s PC. V prostředí bascom slouží k obsluze USART základní příkazy „Print“ a „Input“ pro zápis a čtení. Byl napsán jednoduchý program, kdy mikroprocesor přijal z PC znaky a odeslal je zpět na terminál ve windows.

V další fázi byl program rozšířen o spolehlivou základní komunikaci mezi mikroprocesorem a modulem Nano socketLan a dekódování přijatých znaků v závislosti na podmínce „IF – THEN – ELSE“ Pro zvýšení spolehlivosti přijatých znaků byl v programu použit softwarový buffer. Dále byla ověřena komunikace 1-wire mezi mikroprocesorem a teplotním čidlem DS18S20 výsledná naměřená hodnota byla během ladění odesílána na terminál ve windows.

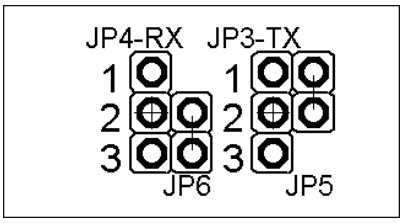
Poslední fáze vývoje byla již směřována ke konkrétní spolupráci s vytvořenými www stránkami. Obsluha výstupních portů (rozsvěcení LED v závislosti na zatržení checkboxu), zobrazení stavu vstupních portů (zapnuto, vypnuto – dle zkratovacích propojek „jumperů“), zobrazení měřené teploty přímo na www stránkách.

Bližší popis jednotlivých částí programu je v kapitole *4.3 Popis programu MCU.*

4.5 Oživení kompletního modulu

Po vizuální kontrole osazení DPS před osazením aktivních součástek ověříme správné napájecí napětí na jednotlivých paticích (MCU, Modul, MAX3232). Komunikační jumpery nastavíme do polohy PC - Nano SocketLan viz. Tab.č 3. Modul připojíme k PC na COM port, popřípadě přes USB převodník, osadíme modul Nano SocketLAN a připojíme napájení. Provedeme základní nastavení Nano SocketLan popsané v kapitole *4.2.1 nastavení.*

Dále osadíme naprogramovaný mikroprocesor, komunikační jumpery nastavíme do polohy MCU-Nano SocketLan. Po zapnutí napájení po dobu 2s blikají výstupní led jako indikace běhu programu, po té MCU odešle sadu inicializačních AT příkazů do modulu. Nyní pokud je modul kompletně konfigurován a připojen do internetové sítě odešle email „Modul OK“ a zpřístupní uživatelské webové stránky, které jsou dostupné na jakémkoli PC v síti přes internetový prohlížeč a je možné ověřit kompletní funkci.



Obr. č. 39 – Popis nastavení jumperů pro komunikaci

Komunikace:	*Jumper*	*Pin*	*Jumper*	*Pin*
PC - Nano SocketLan	JP4-RX	2-3	JP3-TX	2-3
PC - MCU	JP4-RX	1-2	JP3-TX	1-2
MCU - Nano SocketLan	JP6	2-3	JP5	1-2

Tab. č. 3 – nastavení jumperů pro komunikaci

Závěr

Cílem této bakalářské práce je seznámení s moderní součástkovou základnou pro TCP/IP komunikaci. Ověření programování a ladění AVR aplikací pomocí kitu STK500 v konstrukci s modulem Nano SocketLan.

Vývoj celé konstrukce potvrdil univerzálnost a spolehlivost použití hotového modulu Nano SocketLan ve spojení s vhodně naprogramovaným mikroprocesorem ATmega32, což velmi jednoduše umožňuje právě programovací prostředí Bascom. Díky těmto poznatkům je možné dovybavit starší zařízení komunikující po RS232 bezproblémovou TCP/IP komunikací a přenášet tak data internetovou sítí. Popřípadě realizovat jednoduchou stavovou telemetrii nebo dálkové ovládání zařízení pomocí internetové sítě.

Z tohoto důvodu je také celá konstrukce pojata jako univerzální vývojový modul pro případné pohodlné odladění konkrétní aplikace.

Seznam použitých zdrojů

Tištěné zdroje:

[1] Matoušek, D.: *Práce s mikrokontroléry ATMEL AT89C2051 1.díl*. Praha, BEN, 2002. 248 s. ISBN 80-7300-048-2

[2] Matoušek, D.: *Práce s mikrokontroléry ATMEL AVR 3.díl*. Praha, BEN, 2003. 376 s. ISBN 80-7300-088-1

[3] Matoušek, D.: *Práce s mikrokontroléry ATMEL AVR ATmega16 4.díl*. Praha, BEN, 2006. 320 s. ISBN 80-7300-174-8

[4] Matoušek, D.: *Práce s mikrokontroléry ATMEL AT89LP2052 5.díl*. Praha, BEN, 2006. 200 s. ISBN 80-7300-205-1

[5] Matoušek, D.: *Práce s mikrokontroléry ATMEL AT89S8252 2.díl*. Praha, BEN, 2002. 304 s. ISBN 80-7300-066-0

[6] Hrbáček, J.: *Komunikace mikrokontroléru s okolím*. Praha, BEN, 1999. 152 s. ISBN 80-86056-36-8

[7] Váňa, V.: *Mikrokontroléry ATMEL AVR - Bascom*. Praha, BEN, 2004. 144 s. ISBN 80-7300-115-2

[8] Dostálek, L, Kabelová, A.: *Velký průvodce protokoly TCP/IP a systémem DNS*. Praha, Computer Pres, 2000. 435 s. ISBN 80-7226-323-4

Webové zdroje:

[9] http://www.atmel.com

[10] http://www.hw.cz

[11] http://www.asix.cz

[12] http://www.earchiv.cz

[13] http://united-nuke.openland.cz/

[14] http://www.mcselec.com

[15] http://www.spezial.cz

[16] http://www.eagle.cz

[17] http://www.mcselec.com

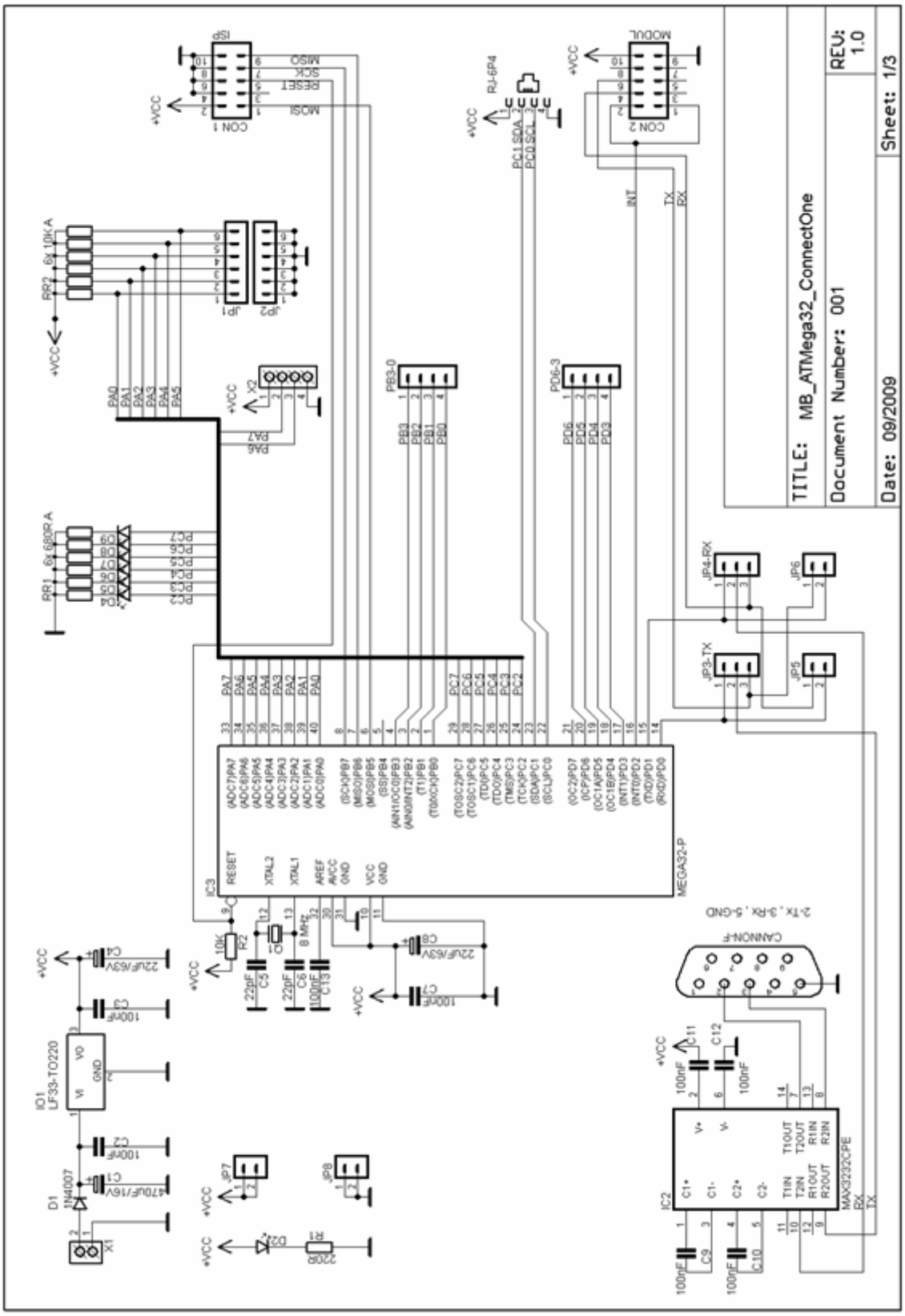
[18] http://www.wikipedia.org

Seznam obrázků

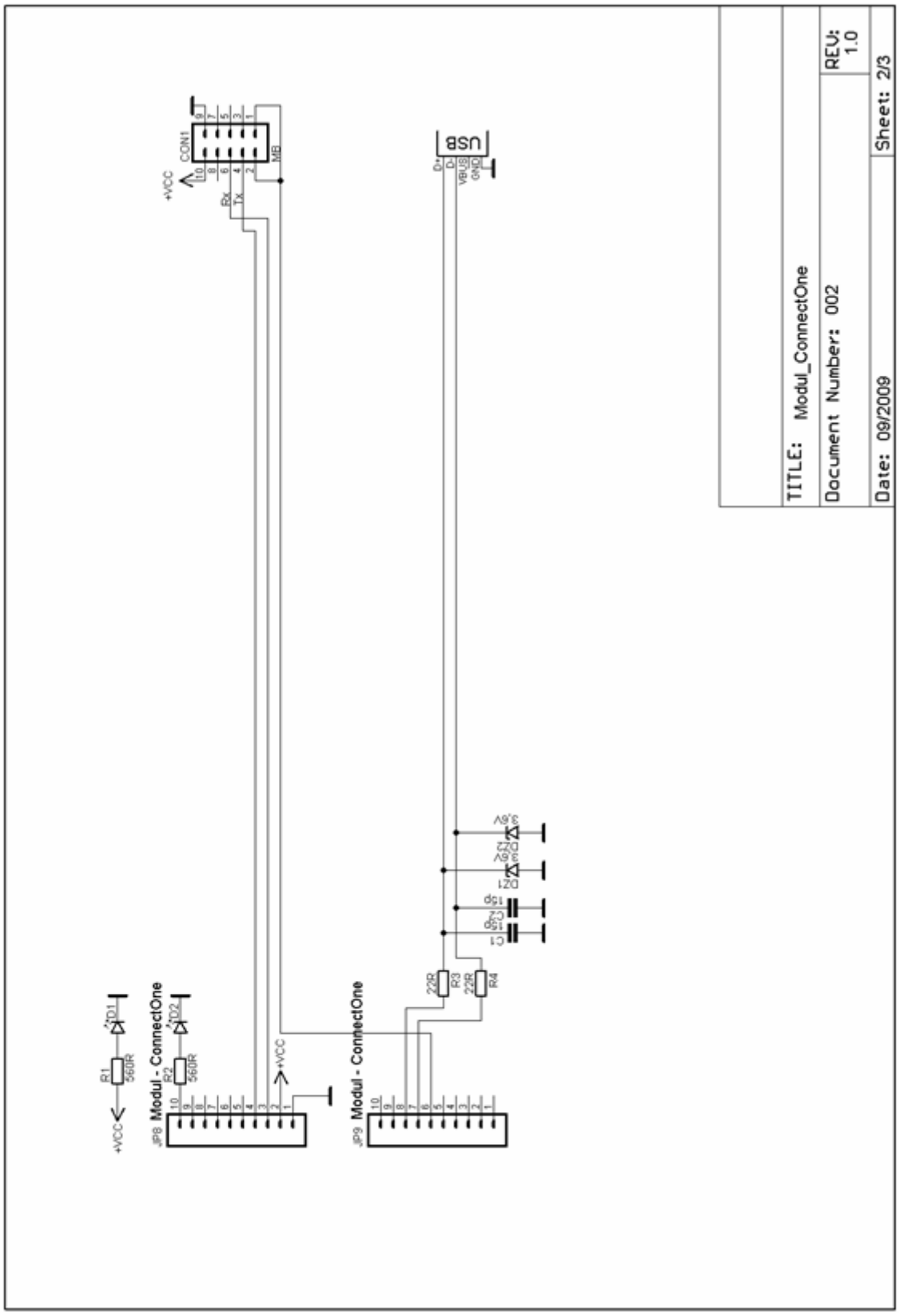
Seznam tabulek

Přílohy

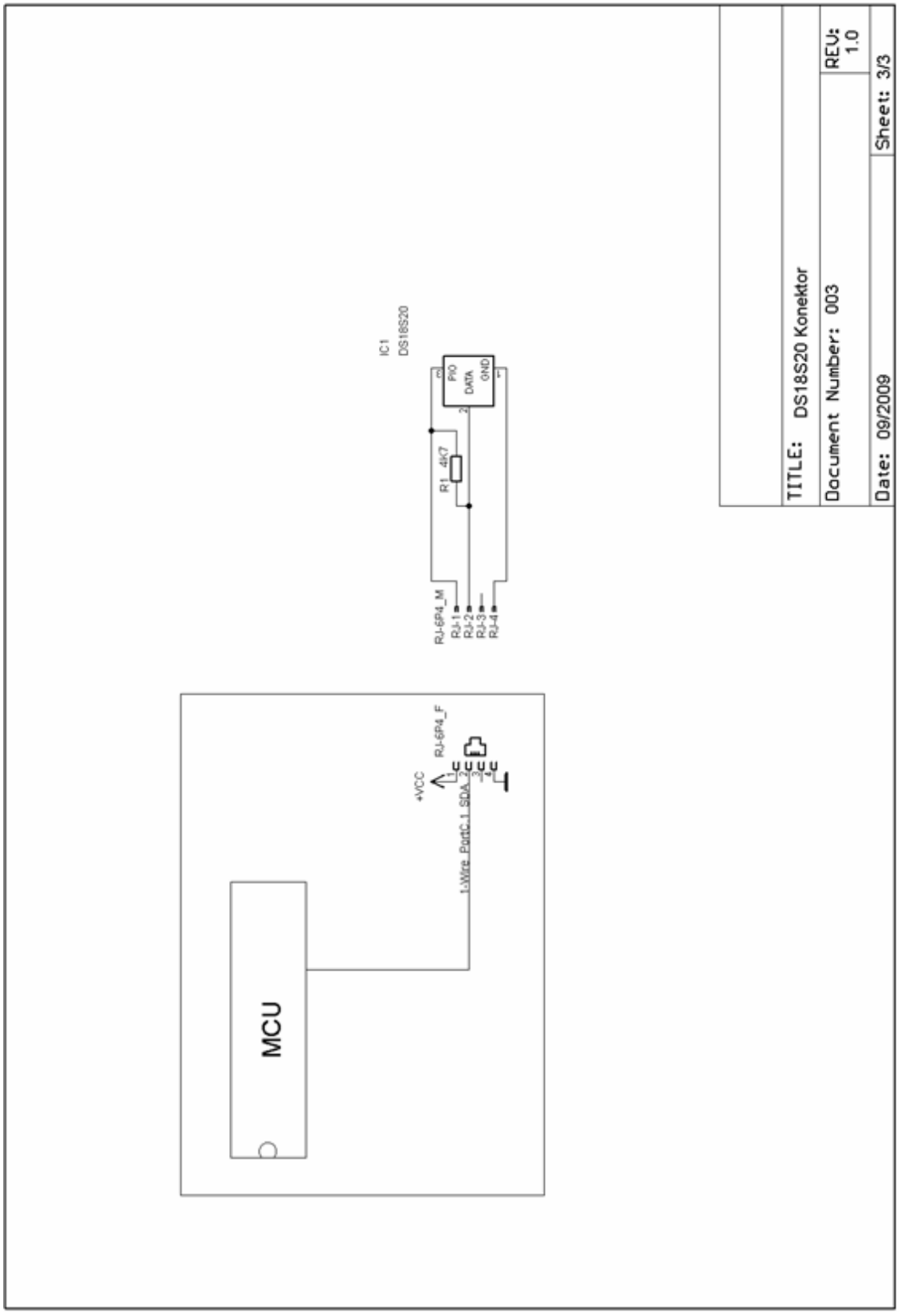
Příloha 1 – Schéma modulu



Obr. č. 1.1 – Schéma základní desky

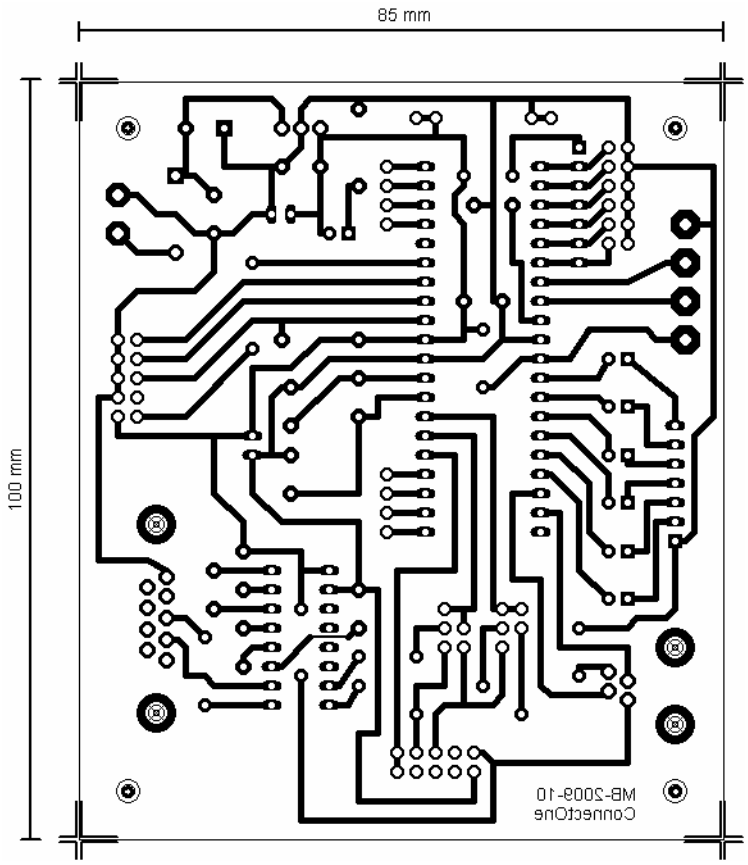


Obr. č. 1.2 – Schéma desky modulu

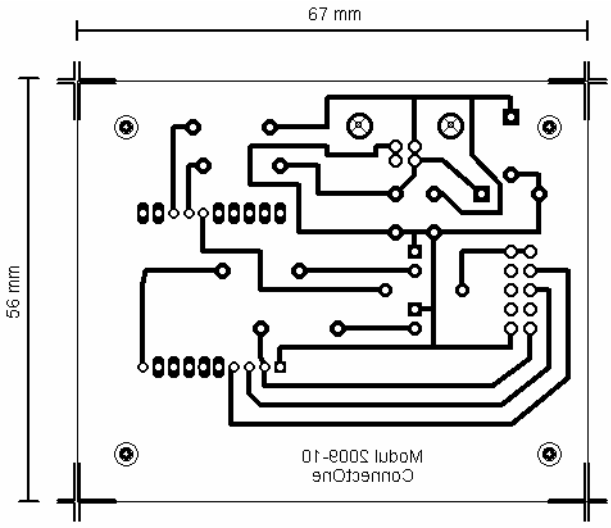


Obr. č. 1.3 – Schéma konektoru DS18S20

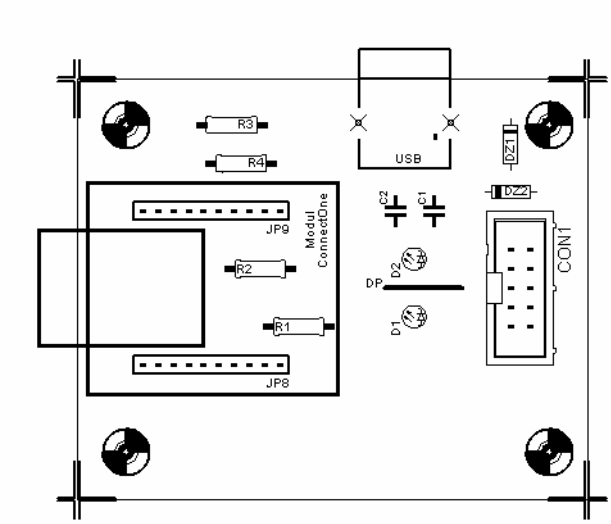
Příloha 2 – Předlohy DPS



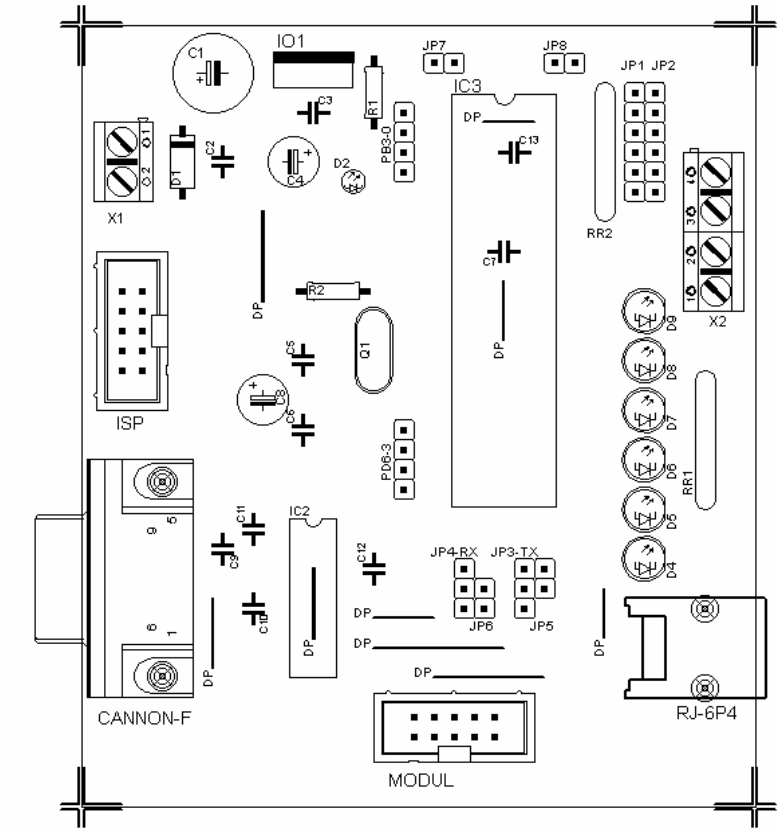
Obr. č. 2.1 – Předloha DPS - MB (pohled ze strany součástek)



Obr. č. 2.2 – Předloha DPS - Modul (pohled ze strany součástek)



Obr. č. 2.3 – Modul - osazovací plán



Obr. č. 2.4 – MB - osazovací plán

Příloha 3 – Seznam součástek

Seznam součástek – Základní deska (MB):	
název:	*hodnota:*
R1	Rezistor – 220R – 0,6W, vel. 0207
R2	Rezistor – 10K - 0,6W, vel. 0207
RR1	Rezistorová síť – 6x220R typ.A
RR2	Rezistorová síť – 6x10K typ.A
C1	Kondenzátor – 470uF/16V, Elektrolytický
C4, C8	Kondenzátor – 22uF/63V, Elektrolytický
C2, C3, C7, C9-13	Kondenzátor - 100 nF, Keramický
C5, C6	Kondenzátor - 22 pF, Keramický
Q1	Krystal HC-49U, Q=8.000.000 Hz
D1	Dioda 1A – 1N4007
D2	LED 3mm – 2mA
D4-9	LED 5mm – 2mA
IO1	Monolitický stabilizátor LF33 – 3,3V - 150mA
IC2	Převodník úrovní MAX3232P – DIL 16
IC3	MCU ATmega32L – P – DIL 40
JP1, JP2	Konektorové kolíky 6 pin, RM=2,54 mm
JP3, JP4	Konektorové kolíky 3 pin, RM=2,54 mm
JP5, JP6, JP7, JP8	Konektorové kolíky 2 pin, RM=2,54 mm
PB3-0, PB6-3	Konektorové kolíky 4 pin, RM=2,54 mm
CON1,2 - ISP, Modul	Počítačový konektor na plochý kabel MLW10 – 10pin
CANNON – F	Počítačový konektor cannon 9 pin samice do DPS – CAN9Z/90
RJ-6p4-F	Telefonní konektor do DPS – RJ6p4
X1	Svorkovnice ARK500/2
X2	Svorkovnice ARK500/4

Tab. č. 3.1 – Seznam součástek základní desky

Seznam součástek – deska modulu ConnectOne (Modul):	
název:	*hodnota:*
R1, R2	Rezistor – 560R - 0,6W, vel. 0207
R3, R4	Rezistor – 22R - 0,6W, vel. 0207
C1, C2	Kondenzátor – 15 Pf, Keramický
D1, D2	LED 3mm – 2mA
DZ1, DZ2	Zenerova dioda 3,6V – 0,5W
JP8, JP9	Konektorové dutinky 10 pin, RM=2 mm
CON1	Počítačový konektor na plochý kabel MLW10 – 10pin
X1	Konektor USB-B do DPS – USB1X90B PCB

Tab. č. 3.2 – Seznam součástek desky modulu

Seznam součástek – Teplotní čidlo:	
název:	*hodnota:*
R1	Rezistor – 4k7 - 0,6W, vel. 0207
DS18S20	Čidlo teploty Dallas 1-Wire - TO92
RJ-6p4-M	Telefonní konektor na kabel – krepovací – RJ6p4
Kabel4pol	Kabel telefonní 4pol. cca 30 cm

Tab. č. 3.3 – Seznam součástek teplotní čidlo

Příloha 4 – výpis ovládacího programu pro MCU ATmega32

```
'-------------------------------------------------------------------------------------------------------'
'   Název aplikace: Webový server s modulem ConnectOne                                                  '
'                  Ovládání vstupů výstupů, měření teploty přes webové rozhraní                        '
'   MCU: ATmega32, Crystal 8MHz, Ucc= 3,3V                                                              '
'   Bakalářská práce - Jiří Volf - MVTk 2006/2010                                                       '
'-------------------------------------------------------------------------------------------------------'


'********************  Definice, Deklarace proměnných  ***********************'

$prog &HFF , &HFF , &HC9 , &H00                         'konfigurace pojistek
$regfile = "m32def.dat"
$crystal = 8000000
$baud = 57600

$hwstack = 32
$swstack = 10
$framesize = 40

Dim Rx_ram1 As String * 16                              'řetězec 16 bytů
Dim Rx_ram2 As String * 16
Dim Rx_ram3 As String * 16
Dim Rx_ram4 As String * 16
Dim Rx_ram5 As String * 16
Dim Rx_ram6 As String * 16
Dim Ram_t(9) As Byte
Dim Pom As Byte                                                                                         '
“Číslo“ 1byte
Dim Crc As Byte
Dim Z As Byte
Dim T As Integer
Dim T1 As Integer

Config Serialout = Buffered , Size = 40                 'výstupní buffer
Config Serialin = Buffered , Size = 40                  'vstupní buffer
Config 1wire = Portc.1                                  'sběrnice 1-Wire
Config Portc = Output                                   'inicializace portu
Config Porta = Input                                    'inicializace portu

Enable Interrupts                                       'povoleni přerušeni UDRE

'********************   Hlavní program  ************************************'
For Z = 1 To 8                                          'test Led diod 2s blikání
Portc = &H55
Waitms 250
Portc = &HAA
Waitms 250
Next Z
Portc = &H00

Wait 1


'----------------------- Inicializace modulu (AT příkazy-Uart) ---------------------------------'

Print "at+ie0" ; Chr(13)                               'vypnout ECHO
Waitms 25
Print "at+iwpwd=*" ; Chr(13)                           'zrušit heslo pro WWW
Waitms 25
Print "at+irpg=*" ; Chr(13)                            'zrušit heslo parametry
Waitms 25
Print "at+iwww:3" ; Chr(13)                            'spustit www server
Waitms 25
```

```
Print "at+iema:Modul  OK" ; Chr(13) ; "." ; Chr(13)     'odeslat email
Waitms 100

Clear Serialin                                          'ukazatel na zacatek Rx bufferu

Do                                                      'hlavní programová smyčka

Gosub Vstupy                                            'skok na podprogram
Gosub Vystupy
Gosub Dekoduj
Gosub Mereni_t

Waitms 200

Loop

End                                                     'úplný konec programu

'========================= Podprogramy ================================='

'-----------------------------------------------------------------------------------------------------'
' podprogram sejme stav vstupů MCU portA, a odešle stav (řetězec) do modulu          '
'-----------------------------------------------------------------------------------------------------'


Vstupy:

If Pina.0 = 1 Then                                      'pin je v 1 odešli:vypnuto
   Print "at+iin1=vypnuto " ; Chr(13)
End If
If Pina.0 = 0 Then                                      'pin je v 0 odešli:zapnuto
   Print "at+iin1=zapnuto" ; Chr(13)
End If
Waitms 50
Clear Serialin

If Pina.1 = 1 Then
   Print "at+iin2=vypnuto " ; Chr(13)
End If
If Pina.1 = 0 Then
   Print "at+iin2=zapnuto" ; Chr(13)
End If
Waitms 50
Clear Serialin

If Pina.2 = 1 Then
   Print "at+iin3=vypnuto " ; Chr(13)
End If
If Pina.2 = 0 Then
   Print "at+iin3=zapnuto" ; Chr(13)
End If
Waitms 50
Clear Serialin

If Pina.3 = 1 Then
   Print "at+iin4=" "" ; Chr(13)
End If
If Pina.3 = 0 Then
   Print "at+iin4=checked" ; Chr(13)
End If
Waitms 50
Clear Serialin
```

```
If Pina.4 = 1 Then
   Print "at+iin5=" "" ; Chr(13)
End If
If Pina.4 = 0 Then
   Print "at+iin5=checked" ; Chr(13)
End If
Waitms 50
Clear Serialin

If Pina.5 = 1 Then
   Print "at+iin6=" "" ; Chr(13)
End If
If Pina.5 = 0 Then
   Print "at+iin6=checked" ; Chr(13)
End If
Waitms 50
Clear Serialin

Return                                                  'návrat do hlavní smyčky


'-----------------------------------------------------------------------------------------------------------'
'  podprogram kontroluje hodnotu checkboxu na www                                                           '
'  zatrženo (check) - do ram zapíše "checked"                                                               '
'  nezatrženo       - do ram zapíše " "                                                                     '
'-----------------------------------------------------------------------------------------------------------'


Vystupy:

Clear Serialin : Waitms 10

Print "at+iout1?" ; Chr(13)                             'stav checkboxu?
Input Rx_ram1                                           'zapiš do RAM
Waitms 5
Clear Serialin                                          'ukazatel na zač. bufferu

Print "at+iout2?" ; Chr(13)
Input Rx_ram2
Waitms 5
Clear Serialin

Print "at+iout3?" ; Chr(13)
Input Rx_ram3
Waitms 5
Clear Serialin

Print "at+iout4?" ; Chr(13)
Input Rx_ram4
Waitms 5
Clear Serialin

Print "at+iout5?" ; Chr(13)
Input Rx_ram5
Waitms 5
Clear Serialin

Print "at+iout6?" ; Chr(13)
Input Rx_ram6
Waitms 5
Clear Serialin

Return                                                  'návrat do hlavní smyčky
```

```
'--------------------------------------------------------------------------------'
'  podprogram nastaví výstupy dle obsahu RAM                                     '
'  "checked" výstup = 1                                                          '
'  " "       výstup = 0                                                          '
'--------------------------------------------------------------------------------'

Dekoduj:                                                 'dekóduje Rx_ram

If Rx_ram1 = "checked" Then
   Portc.7 = 1
End If
If Rx_ram1 = " " Then
   Portc.7 = 0
End If

If Rx_ram2 = "checked" Then
   Portc.6 = 1
End If
If Rx_ram2 = " " Then
   Portc.6 = 0
End If

If Rx_ram3 = "checked" Then
   Portc.5 = 1
End If
If Rx_ram3 = " " Then
   Portc.5 = 0
End If

If Rx_ram4 = "checked" Then
   Portc.4 = 1
End If
If Rx_ram4 = " " Then
   Portc.4 = 0
End If

If Rx_ram5 = "checked" Then
   Portc.3 = 1
End If
If Rx_ram5 = " " Then
   Portc.3 = 0
End If

If Rx_ram6 = "checked" Then
   Portc.2 = 1
End If
If Rx_ram6 = " " Then
   Portc.2 = 0
End If

Return                                                   'návrat do hlavní smyčky

'--------------------------------------------------------------------------------'
'  podprogram měření teploty pomocí DS 18s20                                     '
'  výsledná teplota je odeslána pomocí AT+i (hodnota) do modulu                  '
'                                                                                '
'--------------------------------------------------------------------------------'

Mereni_t:

   1wwrite &HCC : 1wwrite &H44                           'inicializace čidla
   Waitms 300
```

```
   Gosub Cti1820                                        'čtení registrů 9 bytů

   If Err = 1 Then
      Print "at+iteplota=DS 18S20 - odpojeno" ; Chr(13) 'chyba 1-wire
   Else
   If Crc = 0 Then
      Print "at+iteplota=" ; T ; "." ; T1 ; " °C" ; Chr(13) 'zobraz teplotu
   Else
      Print "at+iteplota=!! chyba !!" ; Chr(13)         'chyba  CRC
   End If
   End If

Return


Cti1820:

   1wreset                                              'reset
   1wwrite &HCC                                         'čtení RAM DS1S20
   1wwrite &HBE                                         'čtení registrů 9 bytů

   Ram_t(1) = 1wread(9)

   1wreset                                              'reset

   Gosub Crcit

   If Crc = 0 Then                                      'CRC OK
      Pom = Ram_t(1) And 1

   End If

   If Pom = 1 Then

      Decr Ram_t(1)                                     'výpočet teploty
      T = Makeint(ram_t(1) , Ram_t(2))
      T = T * 50
      T = T - 25
      T1 = Ram_t(8) - Ram_t(7)
      T1 = T1 * 100
      T1 = T1 / Ram_t(8)
      T = T / 100
      T1 = T1 / 10

   End If

Return


Crcit:                                                  ' CRC součet

   Crc = 0
     For Z = 1 To 9
       Pom = Crc Xor Ram_t(z)
       Crc = Lookup(pom , Crc8)
   Next Z

Return
```

```
Crc8:
Data 0 , 94 , 188 , 226 , 97 , 63 , 221 , 131 , 194 , 156
Data 126 , 32 , 163 , 253 , 31 , 65 , 157 , 195 , 33 , 127
Data 252 , 162 , 64 , 30 , 95 , 1 , 227 , 189 , 62 , 96
Data 130 , 220 , 35 , 125 , 159 , 193 , 66 , 28 , 254 , 160
Data 225 , 191 , 93 , 3 , 128 , 222 , 60 , 98 , 190 , 224
Data 2 , 92 , 223 , 129 , 99 , 61 , 124 , 34 , 192 , 158
Data 29 , 67 , 161 , 255 , 70 , 24 , 250 , 164 , 39 , 121
Data 155 , 197 , 132 , 218 , 56 , 102 , 229 , 187 , 89 , 7
Data 219 , 133 , 103 , 57 , 186 , 228 , 6 , 88 , 25 , 71
Data 165 , 251 , 120 , 38 , 196 , 154 , 101 , 59 , 217 , 135
Data 4 , 90 , 184 , 230 , 167 , 249 , 27 , 69 , 198 , 152
Data 122 , 36 , 248 , 166 , 68 , 26 , 153 , 199 , 37 , 123
Data 58 , 100 , 134 , 216 , 91 , 5 , 231 , 185 , 140 , 210
Data 48 , 110 , 237 , 179 , 81 , 15 , 78 , 16 , 242 , 172
Data 47 , 113 , 147 , 205 , 17 , 79 , 173 , 243 , 112 , 46
Data 204 , 146 , 211 , 141 , 111 , 49 , 178 , 236 , 14 , 80
Data 175 , 241 , 19 , 77 , 206 , 144 , 114 , 44 , 109 , 51
Data 209 , 143 , 12 , 82 , 176 , 238 , 50 , 108 , 142 , 208
Data 83 , 13 , 239 , 177 , 240 , 174 , 76 , 18 , 145 , 207
Data 45 , 115 , 202 , 148 , 118 , 40 , 171 , 245 , 23 , 73
Data 8 , 86 , 180 , 234 , 105 , 55 , 213 , 139 , 87 , 9
Data 235 , 181 , 54 , 104 , 138 , 212 , 149 , 203 , 41 , 119
Data 244 , 170 , 72 , 22 , 233 , 183 , 85 , 11 , 136 , 214
Data 52 , 106 , 43 , 117 , 151 , 201 , 74 , 20 , 246 , 168
Data 116 , 42 , 200 , 150 , 21 , 75 , 169 , 247 , 182 , 232
Data 10 , 84 , 215 , 137 , 107 , 53

'=====================     Konec podprogramů    ============================='
```

Příloha 5 - AT+i příkazy modulu Nano SocketLan

Command	Function	Parameters/Description
AT+i	Command prefix	Required to precede all commands
Host Interface		
En	Echo Mode	n=0 Do not echo host characters n=1 Echo all host characters (default upon power-up) This command is equivalent to and interchangeable with ATEn.
Parameter Database Maintenance		
<*par*>=*value -or-* <*par*>:*value*	Set parameter	*value* stored in parameter <*par*> in nonvolatile memory. <*par*> retains set value indefinitely after power down.
<par>~value	Assign single session parameter value	*value* is assigned to parameter <*par*> for the duration of a single Internet session. Following the session, the original value is restored.
<par>?	Read parameter	Parameter value is returned.
<par>=?	Parameter allowed values	Returns the allowed values for this parameter.
FD	Factory Defaults	Restores all parameters to factory defaults.
Status Report		
RP<*i*>	Request status report	Returns a status report value based on <i>.
Connection		
BDRA	Auto baud rate mode	Forces iChip into auto baud rate detection mode.
UP	Connect to Internet	Forces iChip to go online, establish an Internet session, and optionally register its IP address.
TUP	Triggered Internet session mode	Enters a mode in which iChip goes online in response to triggers from external signals. It also supports a special Always Online mode.
DOWN	Perform a software reset	Performs a software reset. Forces iChip to terminate an Internet session and go offline.
PING	PING a remote system	Sends a PING message and waits for its echo response.
FOPN	Open FTP link	Opens an FTP command socket to a remote FTP server. If iChip is not online, it is connected. Once an FTP link is established, it

		can be used to carry out operations on the server‘s file system.
FOPS	Open secure FTP link	Opens an FTP link and negotiates an SSL3/TLS1 connection on the control channel. All following FTP operations in this session are performed over an SSL3/TLS1 connection.
FDL	FTP directory listing	Retrieves the remote FTP server‘s file directory listing. The full server-dependent listing is returned.
FDNL	FTP directory name list	Retrieves the remote FTP server‘s file directory listing. Only file names are returned.
FMKD	FTP make directory	Creates a directory on a remote FTP server.
FCWD	FTP change directory	Changes a remote FTP server‘s current directory.
FSZ	FTP file size	Retrieves the size of a file stored on a remote FTP server.
FRCV	FTP file receive	Downloads a file from a remote FTP server.
FSTO	FTP file store	Opens a file for upload to a remote FTP server. If the file already exists, it is overwritten.
FAPN	FTP file append	Opens a file on a remote FTP server for appending. If the file does not already exist, it is created.
FSND	FTP file send	Sends data to a file on a remote FTP server. The file must be already open by a previous FSTO or FAPN command.
FCLF	FTP close file	Closes the currently open file on an FTP server. Any data uploaded to the file with the FSND command is retained on the server.
FDEL	FTP delete file	Deletes a file from a remote FTP server‘s file system.
FCLS	FTP close	Closes an FTP link.
STCP:<*host*>, <*port*>[,<*lport*>]	Socket TCP	Opens and connects a TCP socket. If iChip is not online, it is connected. The responding system is assumed to be a server listening on the specified socket. Returns a handle to the socket.

SUDP: <*host*>,<*rport*> [,<*lport*>]	Socket UDP	Opens, connects, and optionally binds a UDP socket. If iChip is not online, it is connected. Returns a handle to the socket.
LTCP: <*port*>,<*backlog*>	Listening socket	Opens a TCP listening socket on <*port*>. Allows a maximum of <*backlog*> concurrent connections. Returns a handle to the socket. Up to two listening sockets are supported.
LSST:<*hn*>	Listening socket status	Returns a list of active socket handles accepted for a listening socket identified by handle <*hn*>.
SST:<*hn*>	Single socket status	Returns status of a single socket identified by handle <*hn*>. A subset of RP4 report.
SCS:<*hn*>	Socket connection status	Returns status of a single socket identified by handle <*hn*>. A subset of RP4 report. Does not report number of buffered characters.
SSND[%]: <*hn*>,<*sz*>:<*stream*>	Socket send	Sends a byte stream of size <*sz*> to the socket identified by handle <*hn*>. The % flag indicates automatic socket flush.
SRCV:<*hn*> [,<*max*>]	Socket receive	Receives a byte stream from the socket identified by handle <*hn*>. Accepts up to <*max*> bytes. If <*max*> is not specified, all available bytes are retrieved.
GPNM:<*hn*>	Get peer name	Retrieves peer name (<*IP*>:<*port*>) of a remote connection to the TCP/UDP socket specified by socket handle <*hn*>.
SDMP:<*hn*>	Dump socket buffer	Dumps all buffered data currently accumulated in a socket‘s input buffer. The socket remains open.
SFSH[%]:<*hn*>	Flush socket‘s outbound data	Flushes (sends immediately) data accumulated in a socket‘s outbound buffer. If the flush-and-acknowledge flag (!) is specified, iChip waits for peer to acknowledge receipt of the TCP packet.

[!]SCLS:<*hn*>	Close socket	Closes a TCP/UDP socket. If that socket is the only socket open and the stay online flag (!) is not specified, iChip terminates the Internet session and goes offline.
SSL:<*hn*>	SSL3/TLS1 socket connection	Negotiates an SSL3/TLS1 connection over an active TCP socket
MCM	Interlaced modem command	Sends an interlaced AT command to the modem while it is online.
Wireless LAN		
WLTR	WLAN transmisssion rate	Sets the maximum allowable WLAN transmission rate.
WLPW	WLAN Tx power	Sets the transmission power of the Marvell WLAN chipset.
WRFU	WLAN radio up	Turns on radio transmission of the Marvell WLAN chipset.
WRFD	WLAN radio down	Turns off radio transmission of the Marvell WLAN chipset.
WRST	Reset WLAN chipset	Performs a hardware reset of the Marvell WLAN chipset.
WLBM	WLAN b mode	Sets the Marvell WLAN chipset to 802.11/b mode.
WLGM	WLAN g mode	Sets the Marvell WLAN chipset to 802.11/g mode.

Příloha 6 – nosič dat CD:

Přiložené CD obsahuje: Kompletní Bakalářskou práci ve formátu Microsoft Word (*.doc) a Adobe Reader (*.pdf). Kompletní dokumentaci k výrobě TCP/IP modulu (schémata, předlohy DPS). Dokumentace k užitým součástkám – Datasheet. Zdrojové kódy programu pro MCU a webové rozhraní. Software (iChipConfig) pro základní nastavení modulu Nano SocketLan.