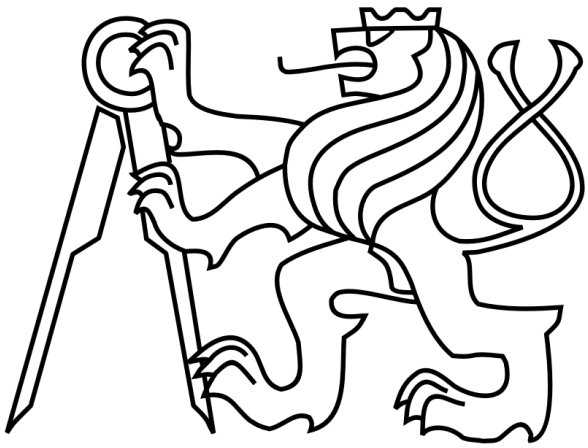
ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
Fakulta elektrotechnická
Katedra měření



DIPLOMOVÁ PRÁCE

Spolupráce senzorů se standardními rozhraními s využitím procesoru ARM

Jaroslav Nesrovnal, 2009

Vedoucí práce: Ing. Jan Fischer, Csc.

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
Fakulta elektrotechnická
Katedra měření

Akademický rok **2007-2008**

ZADÁNÍ DIPLOMOVÉ PRÁCE

Student: **Bc. Jaroslav Nesrovnal**

Obor: **Kybernetika a měření – blok Měřicí a přístrojové systémy**

Název tématu česky: **Spolupráce senzorů se standardními rozhraními s využitím procesoru ARM**

Název tématu anglicky: **ARM Processor Based Sensor Interfacing and Standard Communication**

Zásady pro vypracování:

Navrhněte a realizujte modul s procesorem ARM9 v provedení STR91x, který bude zajišťovat ovládání senzorů a přenos informace na nadřazený systém prostřednictvím standardních rozhraní (např. Ethernet, USB, RS-485, příp. dalších). Orientujte se především na optoelektronické senzory, případně bloky, předzpracovávající obrazovou informaci. Pro tyto senzory navrhněte způsoby spolupráce se standardními rozhraními s využitím výše uvedeného modulu a metody pro přenos dat a jejich následné zobrazení na PC. Vytvořte potřebné programy pro procesor ARM i nadřazené PC.

Posuďte možnost využití některého z operačních systémů vhodných pro vestavné aplikace, např. Micro C-Linux, případně dalších hotových programových bloků. Zpracujte potřebnou dokumentaci a vytvořte další potřebné obvodové a programové prostředky tak, aby modul bylo možno využít i ve výuce mikroprocesorové techniky.

Seznam odborné literatury:

[1] ARM7TDMI-S Technical Reference Manual. ARM ltd. 2001

[2] STR91x Microcontroller Reference Manual. ST Microelectronics, 2006

[3] Česák, P.: Autonomní videoprocesor pro zpracování videosignálu a řízení osvětlovačů. Diplomová práce ČVUT-FEL, Praha 2005

Vedoucí diplomové práce: Ing. Jan Fischer, CSc.

Datum zadání diplomové práce: 10. květen 2007

Termín odevzdání diplomové práce: 23. květen 2008

Prof. Ing. Vladimír Haasz, CSc.
vedoucí katedry

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE Fakulta elektrotechnická -4-

Doc. RNDr. Tomáš Bílek, CSc.
proděkan

V Praze dne 10. 5. 2007

Prohlášení

Prohlašuji, že jsem svou diplomovou práci vypracoval samostatně a použil pouze podklady (literaturu, projekty, software) uvedené v přiloženém seznamu.

V Plané nad Lužnicí dne 19.1.2009

Jaroslav Nesrovnal

Na tomto místě bych rád bych poděkoval vedoucímu mé diplomové práce, Ing. Janu Fischerovi, CSc. za cenné připomínky a odborné konzultace. Dále bych rád poděkoval svým rodičům za podporu během celého mého studia.

Anotace

Tato práce se zabývá návrhem a realizací modulu s jednočipovým mikrokontrolérem STR912 a obvodem fyzické vrstvy Ethernetu STE100P. Vytvořený modul je určen pro ovládání senzorů a dalších bloků a zařízení v oblasti videometrie prostřednictvím standardních rozhraní, včetně Ethernet (IEEE 802.3) 100Base-TX. Jsou navrženy metody přenosu dat na nadřazený systém (PC) protokoly TCP/IP (TCP, UDP, HTTP, telnet) a je vytvořen software pro mikrokontrolér i pro PC. Je zpracován návod k použití modulu a popsány používané i vytvořené programy.

Annotation

The aim of this diploma thesis was to design and realize a module with microcontroler STR912, and Ethernet physical layer interface STE100P. The module is intended to control sensors, other blocks and devices in videometry, through standard interfaces including Ethernet (IEEE 802.3) 100Base-TX. Methods for data transferes to PC over TCP/IP protocols (TCP, UDP, HTTP, telnet) were chosen and software for microcontroller and for PC designed. Finally the manual for module using was created and used programms are discussed.

Obsah

Obsah.....6

1 Úvod.....10

1.1 Lokální a vzdálené ovládání experimentu.....10

1.2 Převodník Ethernetu na vybrané rozhraní.....11

2 Cíle práce.....13

3 Ethernet, protokol TCP/IP.....14

3.1 OSI model síťové komunikace.....14

3.2 IEEE 802.3 - Ethernet.....15

3.2.1 Formát rámce.....16

3.3 TCP/IP.....17

3.4 Vybrané TCP/IP protokoly.....17

3.4.1 IP – Internet Protocol (3. vrstva OSI).....17

3.4.2 ARP – Address Resolution Protocol (3. vrstva OSI).....18

3.4.3 ICMP – Internet Control Message Protocol (3. vrstva OSI).....18

3.4.4 TCP - Transmission Control Protocol.....19

3.4.5 UDP – User Datagram Protocol (4. vrstva OSI).....20

3.4.6 Telnet – Telecommunations Network (7. vrstva OSI).....20

3.4.7 HTTP – Hypertext Transfer Protocol (7. vrstva OSI).....20

4 Výběr vhodného řešení.....21

4.1 Volba firmwaru do mikrokontroléru.....21

4.2 TCP/IP stack.....21

4.3 µClinux.....22

5 Modul s mikrokontrolérem STR75x.....23

5.1 Hardwarová specifikace modulu STR75x.....23

5.2 Programování mikrokontroléru STR75x metodou ICP - UART.....24

5.3 Program Flash Loader Demonstrator.....24

5.4 Vygenerování souboru *.bin z *.axf v prostředí Keil uVision3.....25

5.5 Vygenerování souboru *.bin z *.hex.....25

5.6 Chyby při návrhu a jejich opravy.....25

5.7 Podklady pro výrobu....26
6 Modul s mikrokontrolérem STR912....27
6.1 Návrh plošného spoje....27
6.2 Hardwarová specifikace modulu....28
6.3 Napájení modulu....29
6.3.1 Proudový odběr modulu....29
6.3.2 Napájení ze svorkovnice....29
6.3.3 Napájení z USB portu PC....29
6.4 Mikrokontrolér STR912....30
6.4.1 Základní rysy....30
6.4.2 Řadič MAC/DMA (ENET)....31
6.4.3 Funkce I/O pinů....33
6.5 Ostatní komponenty desky....33
6.5.1 Obvod fyzické vrstvy – STE100P....33
6.5.2 USB....35
6.5.3 Sériové rozhraní UART....35
6.5.4 Sériové rozhraní SSP (SPI)....35
6.5.5 GPIO (vstupy/výstupy), AD převodník....36
6.5.6 RESET....36
6.5.7 Tlačítka....36
6.5.8 JTAG....36
6.5.9 Zálohovací baterie....37
6.6 Knihovny ovladačů periferií 91x_lib a 91x_enet....37
7 Metody nahrávání firmwaru a vývojová prostředí....38
7.1 Nahrávání přeloženého binárního souboru do MCU....38
7.2 Programování metodou ISP a debug....38
7.3 Programování metodou IAP....39
7.4 Vývojová prostředí pro kompilaci kódu....39
7.4.1 Evaluation verze EWARM instalovaná ve virtuálních Windows....41
7.4.2 Evaluation verze uVision3....41
7.4.3 Prostředí RIDE (Raisonance IDE)....41
7.4.4 Nahrání souboru *.hex do mikrokontroléru....43

8 uIP TCP/IP stack a aplikace.....44
8.1 Klíčové mechanismy uIP stacku.....44
8.1.1 Paketový buffer uIP, buffery síťového čipu.....44
8.1.2 Hlavní smyčka programu.....45
8.1.3 Zatížení procesoru zpracováním paketů.....45
8.1.4 Zpožděné potvrzování ACK pakety.....47
8.2 Zdrojové kódy uIP a jejich struktura.....48
8.2.1 Zarovnání dat v paměti (Alignment).....49
8.3 Aplikace uIP.....50
8.3.1 API (Application Program Interface).....50
8.3.2 Podpora více aplikací, směrování podle portů.....52
8.3.3 Vytvoření nové aplikace.....54
8.3.4 HTTP server s podporou CGI skriptů.....57
8.3.5 Aplikace Terminal.....62
8.3.6 Obousměrný převodník rozhraní TCP/IP - RS232.....62
8.4 Přenos větších objemů dat.....63
8.5 Čtení obrazových dat z procesoru BlackFin.....64
8.6 Porovnání výkonu aplikací.....65
8.7 Konfigurační soubory uIP.....66
8.8 Profily nastavení sítě.....66
8.9 Portování uIP na jinou platformu.....67
8.9.1 Změny v kódu uIP.....68
8.9.2 Změny v ostatních částech kódu.....68
9 Návod pro práci s modulem.....69
9.1 Instalace programů RIDE a uVision, nahrání firmware.....69
9.2 Propojení modulu s PC.....70
9.3 Nastavení sítě v PC.....71
9.4 Ověření běhu programu v mikrokontroléru.....72
9.5 Připojení k aplikacím TCP/IP stacku.....72
9.5.1 Aplikace „Terminal“.....72
9.5.2 HTTP server s podporou CGI.....73
9.5.3 Telnet na standardním portu.....76

9.5.4 Aplikace „meas“ - zobrazení obrázku....76
9.6 Další užitečné programy....77
9.6.1 Program Hercules....77
9.6.2 Wireshark....78
9.6.3 Net Profiles....79
9.7 Vzdálený přístup k modulu prostřednictvím sítě LAN nebo WAN....80
9.7.1 Aktivní síťové prvky....80
9.7.2 Ovládání modulů ze stejné lokální sítě....81
9.7.3 Ovládání modulů z jiné sítě (Internetu)....81
10 Závěr....83
11 Seznam obrázků....85
12 Seznam použitých zkratek....87
13 Seznam literatury....89
14 Obrazové přílohy....90
14.1 Modul STR75x – opravená verze....90
14.1.1 Schéma zapojení....90
14.1.2 Seznam součástek....91
14.1.3 Osazovací výkres....92
14.1.4 Fotografie modulu – první verze modulu....92
14.1.5 Vrstva TOP – měřítko 1,5:1 - 89×54 mm....93
14.1.6 Vrstva BOTTOM – měřítko 1,5:1 - 89×54 mm....93
14.2 Modul STR912....94
14.2.1 Schéma zapojení....94
14.2.2 Seznam součástek....95
14.2.3 Rozmístění součástek a nastavení jumperů....96
14.2.4 Osazovací výkres....97
14.2.5 Fotografie modulu....97
14.2.6 Vrstva TOP – měřítko 1:1 - 124×86 mm....98
14.2.7 Vrstva BOTTOM – měřítko 1:1 - 124×86 mm....98
15 Obsah přiloženého DVD....99

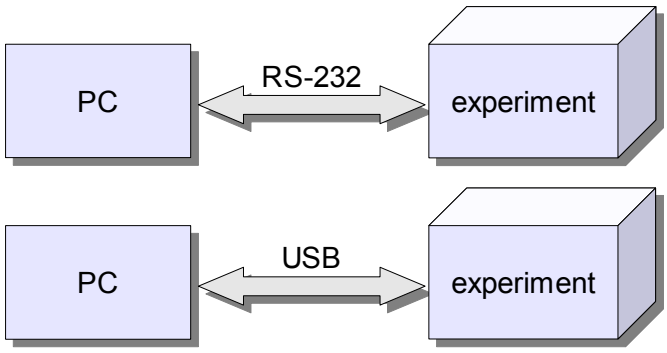
1 Úvod

V měřicích či řídicích aplikacích v průmyslu je používána řada sběrnic pro přenos dat na nadřazený systém, např. sběrnice RS-485, RS-232, CAN, Profibus a další. V poslední době se začínají prosazovat i průmyslové verze sběrnice Ethernet. Řídicí počítač musí obsahovat hardware (v podobě zásuvných karet), schopný po těchto sběrnicích komunikovat. Tato práce se bude zabývat přenosem dat s použitím běžného PC (stolního počítače nebo notebooku v podmínkách školní laboratoře), které disponuje pouze standardními rozhraními RS-232, USB, Ethernet.

RS-232 dosahuje poměrně malých přenosových rychlostí, ale je stále ve velké míře používaným rozhraním, zejména pro jeho dostupnost ve většině (dnes už starších) počítačů a pro jednoduchý komunikační protokol. Moderní osobní počítače už ale sériové porty většinou neobsahují. Oproti tomu sběrnice USB je rychlá, umožňuje připojit více zařízení najednou, avšak pouze na krátké vzdálenosti.

1.1 Lokální a vzdálené ovládání experimentu

V laboratoři videometrie na naší katedře je v současné době použit pro přenos dat mezi úlohami (experimenty) a řídícím PC nejčastěji sériový port (RS-232), nebo sběrnice USB. Propojení experimentu s PC je tedy lokální (viz. obr. 1).

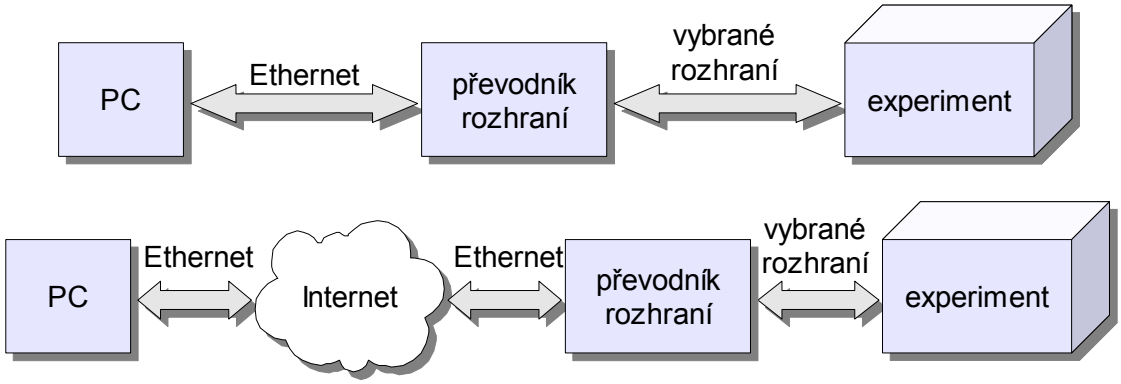


Obrázek 1: Požívaná rozhraní pro lokální ovládání a čtení dat z experimentu.

V některých případech by bylo vhodné umístit experiment do jiné místnosti a přistupovat k němu z měřícího PC vzdáleně. Příkladem může být přesné bezdotykové měření polohy objektů pomocí kamery. Takové měření je vhodné provádět v prostorách s co nejnižší úrovní

otřesů (sklepní místnost) a dobrým zatemněním od okolního světla, které by na měření působilo rušivě.

Řešením tohoto problému je ovládání přes sběrnici Ethernet, která je k dispozici ve většině osobních počítačů a je vhodná jak pro lokální (viz. obr. 2 nahoře), tak pro vzdálený (viz. obr. 2 dole) přístup (v měřítku sítě Internet), při zachování vysoké rychlosti přenosu dat. Výhodou je možnost využít již vybudovanou Ethernetovou síť jako přenosové médium.

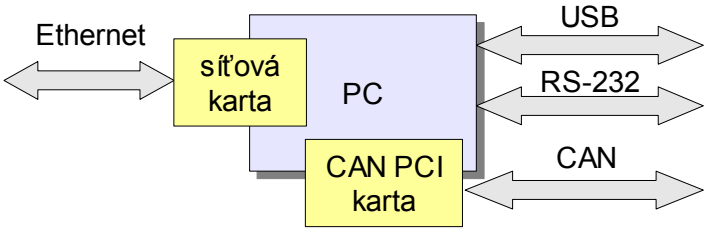


Obrázek 2: Vzdálené ovládání experimentu z lokální sítě a přes Internet.

1.2 Převodník Ethernetu na vybrané rozhraní

Zařízení v laboratoři videometrie Ethernetem nedisponují. Obsahují buďto některá standardní rozhraní, která nalezneme i v PC (RS-232, USB), nebo rozhraní jako SPI, CAN, či pouze vstupně/výstupní brány (analogové nebo digitální). Cílem je připojit tato zařízení k Ethernetu. K tomu je potřeba převodníku (viz. obr. 3) který bude pracovat jako obousměrný most mezi Ethernetem a vybraným rozhraním.

Jako takový převodník se nabízí PC se síťovou kartou LAN (Ethernet). Druhým rozhraním může být RS-232 nebo USB, případně některé další, např. karta CAN v PCI slotu.



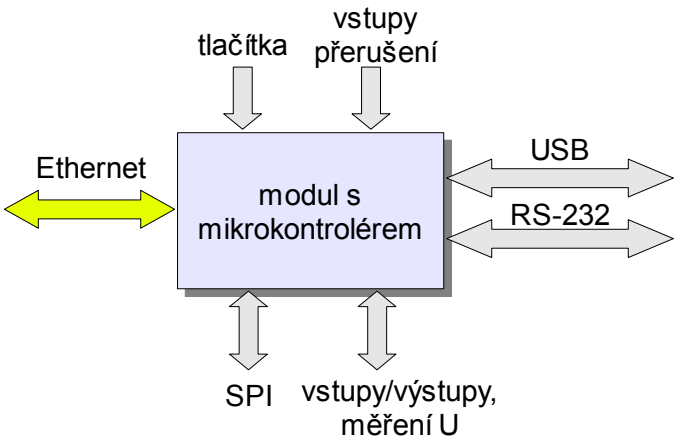
Obrázek 3: PC jako převodník rozhraní.

Toto řešení má ale řadu nevýhod:

- velké rozměry, nemožnost použití v omezenějších podmínkách (malých, hůře přístupných, nebo nebezpečných prostorách)
- velký odběr, hluk a nespolehlivost pohyblivých částí jako jsou ventilátory, pevné disky
- nelze přistupovat přímo na nejnižší hardware

Vhodnější je použití malého zařízení s mikrokontrolérem, který bude mít potřebná rozhraní integrována na čipu a vyvedena na konektory. Existují komerční řešení, ty ale nemají příliš možností, většinou obsahují jen Ethernet a RS-232. Další jejich nevýhodou jsou nedostupné zdrojové kódy.

Návrhem vlastního hardware bude možné přistupovat přímo na hardware (zápis na vstupy, čtení výstupů, měření napětí, generování PWM signálu, atd.), bude vytvořen vlastní zdrojový kód, který může být podle potřeby upravován. Takové zařízení bude předmětem této práce. Ideové blokové schema je na obr. 4.



Obrázek 4: Blokové schema modulu, který bude navržen.

2 Cíle práce

Cílem práce je navrhnout a vytvořit modul obsahující jednočipový mikrokontrolér a obvod fyzické vrstvy Ethernetu. Modul bude obsahovat řadu rozhraní pro připojení různých typů senzorů či obvodových bloků. Bude vytvořeno programové vybavení mikrokontroléru pro přenos dat na nadřazený systém přes rozhraní Ethernet, protokoly TCP/IP (TCP, UDP, telnet, HTTP). Implementaci TCP/IP protokolů pro mikrokontrolér a zejména pak tvorbě aplikací bude věnována největší část této práce. Dále budou vytvořeny nové, nebo použity existující programy pro PC, které budou komunikovat s vytvořeným modulem a z něj přijatá data zobrazovat, ukládat, atd.

Vzhledem k velikosti zdrojového kódu, který bude překládán, nebude možné použít komerčních překladačů ve zdarma dostupných verzích (s omezením velikostí překládaného kódu). Část práce proto bude věnována zhodnocení možností zdarma dostupných překladačů a výběru vhodného z nich.

Některé kapitoly popisující vytvořený firmware, budou popsány podrobněji, aby funkce programu byly jednoduše pochopitelné a bylo možné na vzniklý kód navázat. Aplikace, vytvořené pro TCP/IP stack, budou popsány nejprve v kapitole 8.3, kde budou vysvětleny jejich vnitřní funkce. V kapitole 9.5 pak bude popsáno použití těchto aplikací z pohledu uživatele, který se k nim připojuje z PC.

Modul a jeho programové vybavení bude možné se softwarovými úpravami použít pro různé typy aplikací, jednak pro některé konkrétní úlohy, tak i pro výuku mikroprocesorové techniky na katedře Měření.

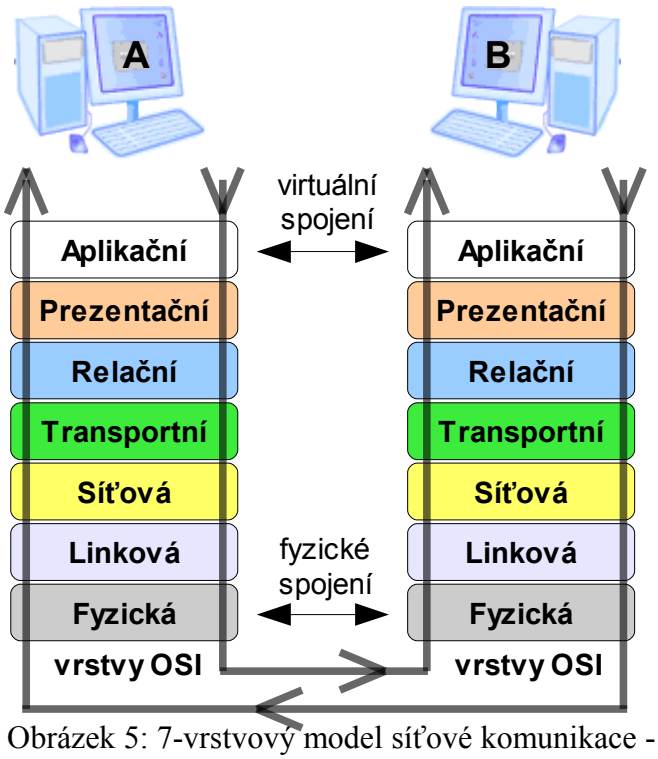
3 Ethernet, protokol TCP/IP

V této kapitole bude podán základní popis rozhraní Ethernet a rodiny protokolů TCP/IP. Popis bude zaměřen na vybrané protokoly, které budou v této práci využívány a implementovány do firmware. Popis bude stručný, pro bližší seznámení s problematikou doporučuji [14].

3.1 OSI model síťové komunikace

Důležitou úlohu v počítačových sítích hrají normy, síťové standardy a doporučení. Díky nim spolu mohou komunikovat síťové prvky různých výrobců. Známé jsou standardy amerického institutu IEEE (Institute of Electrical and Electronic Engineers – viz. www.ieee.org), zejména jejich stále aktualizovaný projekt IEEE-802, který popisuje všechny nejpoužívanější technologie lokálních sítí.

V Evropě je asi nejvyšší autoritou v oblasti tvorby norem ISO – International Standard Organisation. Odtud pochází známý sedmivrstvový model síťové komunikace – OSI (Open System Interconnection) model (viz. obr. 5). OSI model popisuje všechny fáze komunikace mezi stejnorodými i různorodými systémy. Rozděluje činnosti, které probíhají při přenosu informace, do sedmi nezávislých vrstev. Rozhraní mezi těmito vrstvami je přesně popsáno, takže při vývoji komunikačního hardware i software mohou spolupracovat různé specializované firmy. Detailní popis jednotlivých vrstev OSI modelu je popsán v [15].



Obrázek 5: 7-vrstvový model síťové komunikace - OSI model.

3.2 *IEEE 802.3 - Ethernet*

Ethernet je typem lokálních sítí. Pracuje na síťové vrstvě, nedefinuje žádné protokoly, to je záležitostí vyšších vrstev.

Klasický Ethernet využíval sběrnicovou topologii sítě, sdílené médium, ke kterému měly přístup všechny stanice na něj připojené. To ale vedlo ke kolizím na sběrnici v okamžiku, kdy dvě zařízení začala vysílat najednou. K přístupu ke sdílenému médiu se proto používala metoda CSMA/CD (Carrier Sense with Multiple Access and Collision Detection), která dokázala kolizi na médiu detekovat a obnovit vysílání. Tato metoda je využívána v sítích s koaxiálními kabely nebo poloduplexním provozem.

V současnosti nejrozšířenější Ethernet 100Base-Tx používá kroucenou dvojlinku v plně duplexním provozu (full duplex) a díky použití přepínačů (switch) ke kolizi dojít nemůže. Algoritmus CSMA/CD se používá pouze na těchto přepínačích, síť je bezkolizní.

3.2.1 Formát rámce

Ethernetový rámec (frame) v sobě nese data z vyšších vrstev OSI. Může nabývat velikosti 64-1518 bytů (bez prvních osmi bytů určených pro synchronizaci). Jednotlivé položky rámce s vysvětlením znázorňuje tab. 1.

7B	**Synchronizační pole**	Slouží k synchronizaci generátoru hodin síťového adaptéru. Je tvořeno pravidelně se střídajícími jedničkami a nulami – 101010101010 atd.
1B	**Příznak začátku rámce**	Posloupnost 10101011 – indikuje začátek rámce.
6B	**Cílová adresa**	MAC adresa cílové stanice nebo hromadná adresa 11111111 pokud je rámec určen pro všechny stanice na daném segmentu sítě.
6B	**Zdrojová adresa**	MAC adresa stanice, která rámec vyslala.
2B	**Délka rámce**	Počet bytů v datové části rámce.
0-1500B	**Data**	Vlastní přenášená data.
Dle potřeby	**Datová výplň**	U krátkých přenosů se doplňuje délka rámce na 64B pro rychlost 10MBit/s. Nutné ke správné detekci případných kolizí.
4B	**Kontrolní pole rámce**	Kontrolní součet rámce – vypočítá se na vysílací i přijímací straně a obě hodnoty se porovnají. Pokud nejsou stejné, rámec se považuje za chybný a nebude se dále zpracovávat.

Tabulka 1: Formát rámce 802.3.

Poznámka: MAC adresa je 48-bitové číslo přiřazené každému síťovému adaptéru výrobcem. Vyjadřuje se obvykle jako 12 hexadecimálních číslic (např. 0x00153BDEA792). Prvních 24 bitů přiděluje IEEE výrobcům síťových zařízení jako tzv. OUI – Organizational Unique Identifier. Druhých 24 bitů přiděluje každý výrobce tak, aby se nevyskytla žádná dvě zařízení se stejnou MAC adresou.

3.3 TCP/IP

Protokol (rodina protokolů) TCP/IP je hlavním protokolem pro komunikaci v počítačových sítích a síti Internet. Byl navržen tak, aby umožňoval propojení různých počítačů s různými operačními systémy. Původně byl použit v americké vojenské síti Arpanet, která se stala základem dnešního internetu. TCP/IP je nezávislý na fyzické (linkové) přenosové technologii.

IP adresa, síťová maska

Počítače komunikující pomocí protokolu TCP/IP jsou identifikovány 32-bitovým číslem nazývaným **IP adresa**. Pro snadný zápis a čtení se zapisuje jako čtyři dekadická čísla oddělená tečkou. Každé z nich reprezentuje jeden byte a může nabývat hodnot 0-255 (např. 192.168.250.1). Část z této adresy bude vyhrazena pro adresu **sítě (NetID)**, zbytek udává **adresu počítače v dané síti (HostID)**. S adresou sítě pracují routery směrující jednotlivé pakety. **Síťová maska** je 32-bitové číslo, které rozděluje IP adresu na NetID a HostID. Jedničky v masce reprezentují NetID, nuly pak HostID. Také masku sítě zapisujeme ve formě čtyř dekadických čísel oddělených tečkou (např. 255.255.255.0)

3.4 Vybrané TCP/IP protokoly

K přenosu dat protokolem TCP/IP patři řada standardů a protokolů. V této podkapitole popíši ty z nich, které považuji za důležité pro tuto práci.

3.4.1 IP – Internet Protocol (3. vrstva OSI)

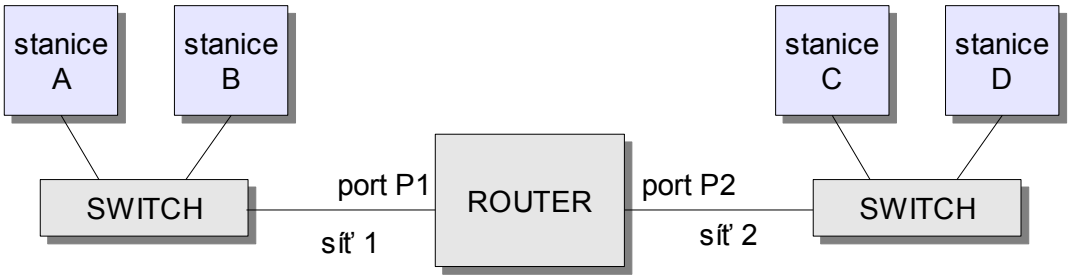
Internet Protocol zajišťuje přenos jednotlivých datagramů po síti. Nese v sobě informace potřebné ke směrování v rozsáhlých sítích. Jde o nespojovanou (connectionless) službu, kde se správné přijetí paketu nepotvrzuje, o spolehlivost se musí postarat protokoly vyšších vrstev, například TCP. Z cílové IP adresy obsažené datagramu musí každý směrovač po cestě určit správný směr k dalšímu směrovači.

3.4.2 ARP – Address Resolution Protocol (3. vrstva OSI)

Pro přenos IP paketů v prostředí lokální sítě je nutné znát nejen IP adresy, ale i MAC adresy jednotlivých síťových adaptérů. K vyhledávání MAC adresy ke známé IP adrese je určen protokol ARP.

Adresy se zjišťují formou dotazů ve všesměrových (broadcast) rámcích. Jsou-li obě stanice na stejném segmentu sítě, je situace jednoduchá. Předpokládejme, že stanice A má IP rámec pro stanici B a potřebuje zjistit její MAC adresu. Vyšle tedy oběžník se žádostí o zjištění MAC adresy k dané IP adrese. Dotaz vyslechnou všechny stanice naslouchající na segmentu, ale jen stanice B odpoví svou vlastní MAC adresou. Stanice A má nyní všechny potřebné informace a vyšle svůj rámec přímo stanici B. Aby se síť nezatěžovala zbytečnými dotazy, ponechá si MAC adresu stanice B také ve své paměti (ARP cache).

Poněkud složitější situace nastane, když každá z komunikujících stanic bude na jiném segmentu sítě (viz. obr. 6). Router v síti směruje na základě IP adres. Žádost o předání IP rámce například od stanice A ke stanici D tedy umí zprostředkovat, ale neumí převádět mezi svými porty MAC adresy. Na dotaz stanice A na MAC adresu stanice D tedy router odpoví svou vlastní MAC adresou na portu P1. Na druhé síti je pak rámec přenesen s použitím MAC adres portu P2 a stanice D.



Obrázek 6: Získání MAC adresy od zařízení na jiném segmentu sítě

3.4.3 ICMP – Internet Control Message Protocol (3. vrstva OSI)

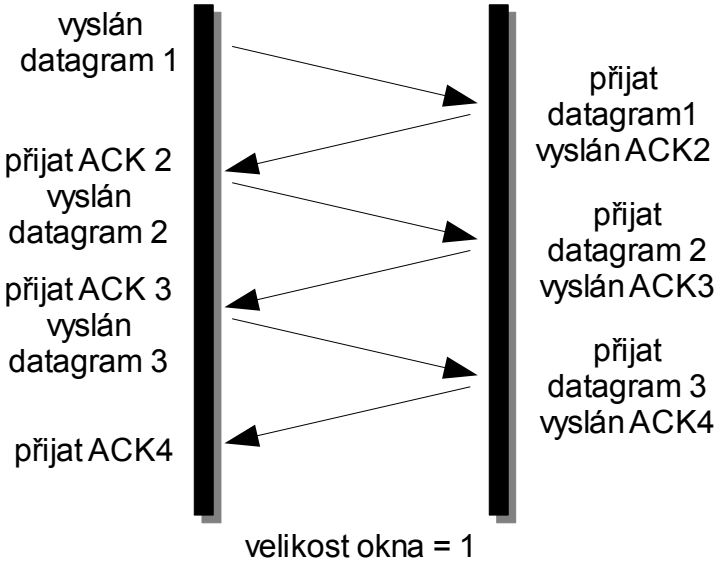
Služební protokol, který slouží k signalizaci mimořádných událostí v sítích postavených na IP protokolu – např. nedoručitelný IP datagram, nedosažitelný uzel, neznámá adresa, nutnost změny směrování, žádost o masku subsítě apod.

ICMP je často používán při zjišťování průchodnosti sítě příkazem ping. Tím je odesílán ICMP paket “Žádost o echo“, na který cílová stanice odpoví ICMP paketem “Echo“.

3.4.4 TCP - Transmission Control Protocol

Protokol TCP je navazuje virtuální spojení mezi vysílací a přijímací stanicí (connection oriented) na začátku přenosu a ukončuje virtuálního spojení po ukončení relace. TCP zajišťuje potvrzování předávaných zpráv a jejich pořadí. Předávané pakety dojdou v pořadí, v jakém byly odvysílány. Aby bylo zřejmé, které aplikace mezi sebou na jednotlivých počítačích komunikují, přidává k IP adrese na každé straně spojení ještě číslo portu. Toto číslo je dvoubajtové, tedy v rozmezí 0-65535. K číslu portu se ještě přidává označení protokolu (TCP nebo UDP, každý protokol má své vlastní porty).

Zpráva určená k přenesení se rozdělí na pakety určité délky, ty se očíslují, opatří IP adresami a vyšlou ve formě datagramů k cílovému počítači. Potvrzování datagramů znázorňuje obr. 7. Po přijetí prvního datagramu vyšle přijímací strana potvrzení ACK (acknowledgement) s číslem o jednotku vyšším. Tím dává najevo úspěšný příjem a požaduje vyslání dalšího datagramu. Nedojde-li k potvrzeni v určitém časovém limitu, vysílání datagramu se opakuje.



Obrázek 7: Potvrzování přijatých TCP/IP datagramů.

Tato metoda přenosu velmi zvyšuje režii a zatížení sítě. Proto automaticky zvyšuje velikost okna – počet za sebou vyslaných datagramů bez potvrzení. Přijímací strana pak potvrzuje až poslední přijatý datagram – opět signálem ACK s číslem o jedničku vyšším. Není-li v určené době série datagramů potvrzena, vyšlou se data znovu a velikost okna se sníží na jedničku.

3.4.5 UDP – User Datagram Protocol (4. vrstva OSI)

Protokol UDP je jednoduchou alternativou protokolu TCP. Je to **nespojovaná (connectionless) služba**, nenavazuje tedy mezi odesílatelem a příjemcem spojení. Odeslané pakety se nepotvrzují, o případné opakování ztraceného datagramu se zde musí postarat aplikační vrstva.

Protokol UDP tedy nevyniká spolehlivostí, ale přesto se najdou aplikace, kde je jeho použití výhodné a nedá se zde nahradit protokolem TCP. Výhodou je menší režie než u TCP protokolu a menší velikost paketu.

3.4.6 Telnet – Telecommunations Network (7. vrstva OSI)

Protokol Telnet zprostředkovává možnost přihlásit se ze vzdáleného počítače pro interaktivní práci na jiném počítači. Telnet naváže TCP spojení a předává cílovému počítači (serveru) jednotlivé znaky tak, jak jsou napsány na klávesnici klienta. Server je zpracovává jako by byly napsány na jeho lokálním terminálu a vzniklé výstupní znaky směruje místo na obrazovku zpět ke klientovi, který je pak lokálně zobrazí. Celá služba je transparentní, neboť uživateli na straně klienta se vše jeví, jako by pracoval přímo na vzdáleném počítači.

Pro telnet jsou podle normy definovány tři základní služby:

- síťový virtuální terminál (NVT – Network Virtual Terminal), který poskytuje standardní rozhraní
- vyjednávání klienta/serveru o nastavení určitých voleb
- symetrické zobrazení terminálu a procesů

3.4.7 HTTP – Hypertext Transfer Protocol (7. vrstva OSI)

Protokol pro komunikaci klientů se serverem WWW. Spojení klienta se serverem je vždy jenom krátkodobé – klient vyšle žádost o určitou stránku, server mu ji odešle a potom spojení ukončí. Při přenosu dalších stránek ze stejného serveru se spojení navazuje vždy znovu. Standardním jazykem pro tvorbu WWW stránek je HTML (Hypertext Markup Language).

Podrobnější popis protokolů rodiny TCP/IP v lit.[14].

4 Výběr vhodného řešení

Řešení hardwarové části modulu je zřejmý. Zadáním byly určeny dvě hlavní komponenty, které bude modul obsahovat – mikrokontrolér STR912 a fyzická vrstva Ethernetu STE100P. Ostatní komponenty a konektory budou zvoleny s ohledem na použití modulu, jeho napájení, atd. Snahou bude vyvést maximum periferií mikrokontroléru na konektory. Hardwarová část je popsána v kap. 6.

Druhou částí řešení bude volba programového vybavení modulu, vývojových nástrojů pro programování a volba programů na straně PC které budou používány pro komunikaci se zařízením.

4.1 Volba firmwaru do mikrokontroléru

Pro mikrokontroléry řady STR zajišťuje výrobce STMicroelectronics knihovny ovladačů všech periferií. Tyto knihovny budou při programování mikrokontroléru STR912 použity.

Druhou část firmware bude tvořit implementace TCP/IP protokolů, takzvaný TCP/IP stack (viz. kap. 8). Třetí, stěžejní částí firmware budou aplikace běžící nad TCP/IP stackem (viz. kap. 8.3).

4.2 TCP/IP stack

TCP/IP stackem se rozumí kód (program), implementující TCP/IP protokol. Většina dnešních operačních systému určených pro PC (Linux OS, Windows) v sobě takový stack obsahuje. Chceme-li implementovat TCP/IP protokol do jednočipového mikroprocesoru, existuje několik možností a projektů které tuto problematiku řeší. Některé TCP/IP stacky jsou komerční placená řešení, některé jsou dostupné v omezených verzích. Následuje přehled TCP/IP stacků, které byly analyzovány (je potřeba poznamenat že situace v této oblasti se rychle vyvíjí):

- InterNiche NicheLite (http://www.iniche.com/nichelite.php) - časově omezená verze
- NUT/OS – v současnosti portován pouze na procesory ATMEGA128
- CMX – placený
- µTasker
- uIP (micro-IP)

- lwIP (leight-weight IP)

Pro účely této práce vyhovují zmíněné poslední dva. Autorem uIP a lwIP je Adam Dunkels, Ph.D., člen S.I.C.S. (Swedish Institute of Computer Science). Tyto implementace jsou určené malá embedded zařízení, především pro 8 a 16-ti bitové mikrokontroléry, s omezenými prostředky programové a datové paměti. Projekty jsou šířeny pod BSD licencí (zdarma dostupné pro nekomerční účely) a podílí se na nich několik dalších vývojářů. uIP je implementován s minimální sadou vlastností pro zajištění plnohodnotné TCP/IP funkcionality. Obsahuje IP, ICMP, UDP a TCP protokoly. Je napsán v programovacím jazyce ANSI-C. lwIP [13] je výkonějším TCP/IP stackem, používá TCP okénko (viz. kap. 3.4.4), vyžaduje více paměti a je složitější. Rozhodl jsem se pro jednodušší, i když méně výkonný uIP stack.

4.3 µClinux

µClinux je alternativou k TCP/IP stacku. Vychází z operačního systému Linux a je určený pro embedded zařízení bez jednotky pro správu paměti MMU (Memory Management Unit). Výhody Linuxu pro použití v malých zařízeních:

- podpora celé řady souborových systémů
- podpora síťových protokolů (TCP/IP)
- neustálý vývoj, opravy chyb, přidávání nových vlastností
- testován velkou komunitou programátorů a uživatelů
- velké množství aplikací

V současnosti je µClinux dostupný ve verzi 2.6 (http://www.uclinux.org/, http://opensrc.sec.samsung.com/). Pro mikrokontroléry STR91x ale není portován. Jeho použití totiž není pro tento procesor vhodné z důvodu malé paměti RAM, pouze 96 kB na čipu. Řešením by bylo připojení externí paměti SRAM. To už ale neodpovídá požadavku zadání, tedy implementovat TCP/IP funkcionalitu do mikrokontroléru jednoduchým způsobem, bez použití externích obvodů. Dalším problémem by bylo složitější portování na platformu STR9, které vyžaduje širší znalosti operačního systému Linux.

5 Modul s mikrokontrolérem STR75x

Předtím, než byl započat vývoj plošného spoje s STR9 a Ethernetem, byla navržena a osazena vývojová deska s mikrokontrolérem STR7, které je věnována tato kapitola. Deska poslouží pro první seznámení s mikrokontroléry ARM a jejich programováním a bude později využita ve výuce mikroprocesorové techniky na katedře Měření.

Poznámka: Výraz mikrokontrolér bude v dalším textu někdy nahrazován zkratkou MCU (Microcontroller Unit).

5.1 Hardwarová specifikace modulu STR75x

- Rozměry: 89×54 mm
- MCU: STR755FR2T6 s 272 kB (256 k+16 k) paměti FLASH a 16 kB paměti SRAM
- Napájení: +5 V
- RS232 sériový port
- 1× napájecí LED
- 4× LED na I/O
- mikrotlačítko pro reset MCU
- 2 mikrotlačítka na I/O pinech
- přepínač bootovacích módů
- trimr na jednom kanále ADC pro testovací účely
- standardní 20-pinový JTAG konektor
- 30 pinů universálních konektorů (I/O piny, ADC, I2C, UART, SPI, PWM, USB – pouze při osazení mikrokontroléry STR751)

Upozornění: celý mikroprocesor používá napájení 3,3 V, I/O piny nejsou +5 V tolerantní
Poznámka: tlačítka jsou bez pull-up rezistorů, je potřeba aktivovat vnitřní pull-up rezistory mikrokontroléru softwarově.

Bližší popis periferií lze nalézt v [19][20].

Plošný spoj byl navržen pro mikrokontrolér STR751, avšak byl osazen typem STR755 (který nemá USB).

5.2 Programování mikrokontroléru STR75x metodou ICP - UART

Mikrokontroléry řady STR75x obsahují interní program, schopný naprogramovat paměť FLASH přes rozhraní UART0. Program se nazývá "ICP bootloader" a je uložen při výrobě do systémové paměti MCU (SystemMemory).

Poznámka: Některé typy mikrokontrolérů STR75x bootloader nemají (viz. aplikační nóta AN2430 [18]). Může jít o některé starší typy, nebo ty které jsou použity v některých vývojových KITech.

Před programováním FLASH paměti přes UART je potřeba spustit procesor v SystemMemory boot módu, nastavením pinů *BOOT0=1* a *BOOT1=0* (viz. obr. 8).

BOOT Mode Selection Pins		Boot Mode	Aliasing	Note
BOOT1	BOOT0			
0	0	Embedded Flash	Embedded FLASH sector B0F0 mapped at 0h	All FLASH sectors accessible except SystemMemory sector
1	0	Embedded SRAM	Embedded SRAM mapped at 0h	
0	1	SystemMemory	SystemMemory mapped at 0h	-
1	1	External SMI	SMI Bank 0 mapped at 0h	-

Obrázek 8: Boot módy mikrokontroléru STR75x (převzato z [20]).

5.3 Program Flash Loader Demonstrator

Na internetu jsem nenalezl žádný volně dostupný program podporující UART bootloader pro STR75x. Od firmy STMicroelectonics jsem získal program Flash Loader Demonstrator (viz. příloha: *DVD\STR755_modul\bootloader_ST*), který je primárně určen pro mikrokontroléry STM32, ale s drobnou úpravou ho je možné použít i pro STR75x. Úprava spočívá v nahrání souboru *STR750F.STmap* (po nainstalování programu) do adresáře *.\Flash Loader\Map*, což rozšíří podporu programu o mikrokontroléry STR75x.

Po spuštění programu vybereme cílový procesor, parametry přenosu po UARTu, a přeložený binární soubor, který chceme nahrát. Ten musí mít příponu **.bin*. Překladače tento typ souboru po překladu obvykle negenerují. V následující kapitole je uveden postup, jakým způsobem vytvořit soubor **.bin* v prostředí *Keil uVision*, které bylo pro práci s mikrokontrolérem použito.

5.4 Vygenerování souboru *.bin z *.axf v prostředí Keil uVision3

uVision3 přímo tento soubor negeneruje, vytváří soubor **.axf*. Soubor **.bin* umí vytvořit program *fromelf.exe*. Ten by měl být po instalaci *uVision3* v adresáři *C:\Keil\ARM\BIN31*. Pokud ne, nalezneme ho v příloze (*DVD\prekladace\ostatni)*. V následujícím odstavci je uveden postup nastavení *uVision,* aby spouštěl *fromelf.exe* automaticky po zkompilování kódu.

Ve vlastnostech projektu, "*Options for Target*", v záložce *USER*, vyplníme do pole "*Run User Program after Rebuild*" cestu:

```
C:\Keil\ARM\BIN31\fromelf.exe --bin --output objects\file.bin
objects\file.axf
```

Program fromelf vygeneruje ze souboru **.axf* soubor **.bin*. Cesty "o*bjects\file.bin"* a "*objects\file.axf"* jsou cesty k souborům z hlavního adresáře projektu.

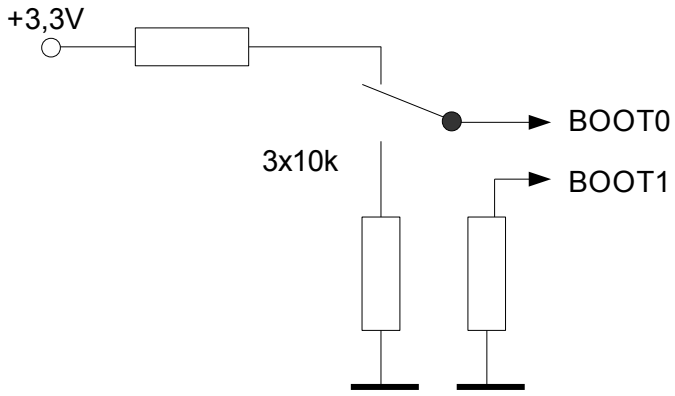
5.5 Vygenerování souboru *.bin z *.hex

Soubor **.bin* můžeme vygenerovat také programem *hex2bin.exe*, který najdeme také v příloze (*DVD\prekladace\ostatni).* Nastavení provedeme analogicky s návodem v předchozím odstavci.

5.6 Chyby při návrhu a jejich opravy

Plošný spoj byl navržen s řadou chyb, které byly po jeho osazení opraveny:

- převodník úrovní ST3232 je napájen 5 V, namísto 3,3 V. Opraveno na ploš. spoji.
- zapojení konektoru **DB-9** je pro připojení **kříženého RS-232 kabelu**, správně měl být navržen pro pro přímý (prodlužovací) kabel. Buďto musí být používán kabel s křížením Rx a Tx, nebo musí být křížení provedeno dodatečně na plošném spoji
- trimr pro AD převodník – chybný návrh pouzdra (prohozené piny), trimr musel být zapájen s křížem ohnutými vývody
- přepínače boot módu – nesprávně navržená pouzdra, doporučuji úpravu zapojení přepínače podle schématu na obr. 9.



Obrázek 9: Volba boot módu mezi FLASH a SystemMemory mode.

5.7 Podklady pro výrobu

Schéma zapojení a plošný spoj byly vytvořeny v programu *OrCAD Capture 10.3* a *Orcad Layout*. Schéma zapojení je v souboru **.DSN*, plošný spoj pak v souboru **.MAX*. Podklady pro výrobu jsou na přiloženém DVD. V adresáři *.\str75x_v1* se nachází první verze plošného spoje, která byla vyrobena a osazena.

Později byla navržena druhá verze ploš. spoje, s následujícími změnami oproti verzi první:

- oprava již zmíněných chyb návrhu
- přidány textové popisky komponent a čísla pinů
- pinové lišty rozšířeny o zemní piny
- přepínače boot módu nahrazeny tlačítky, pro jednodušší volbu boot módu

Schéma zapojení i obrázky vrstev opravené verze jsou k nahlédnutí v obrazových přílohách. Kompletní podklady pak v příloze (*DVD\str75x_v2)*.

6 Modul s mikrokontrolérem STR912

Předchozí kapitola popisovala modul s mikrokontrolérem STR75x, metodu jeho programování a použití ovladačů periferií od výrobce. Tato kapitola bude popisovat hlavní modul s mikrokontrolérem STR912 a fyzickou vrstvou Ethernetu.

6.1 Návrh plošného spoje

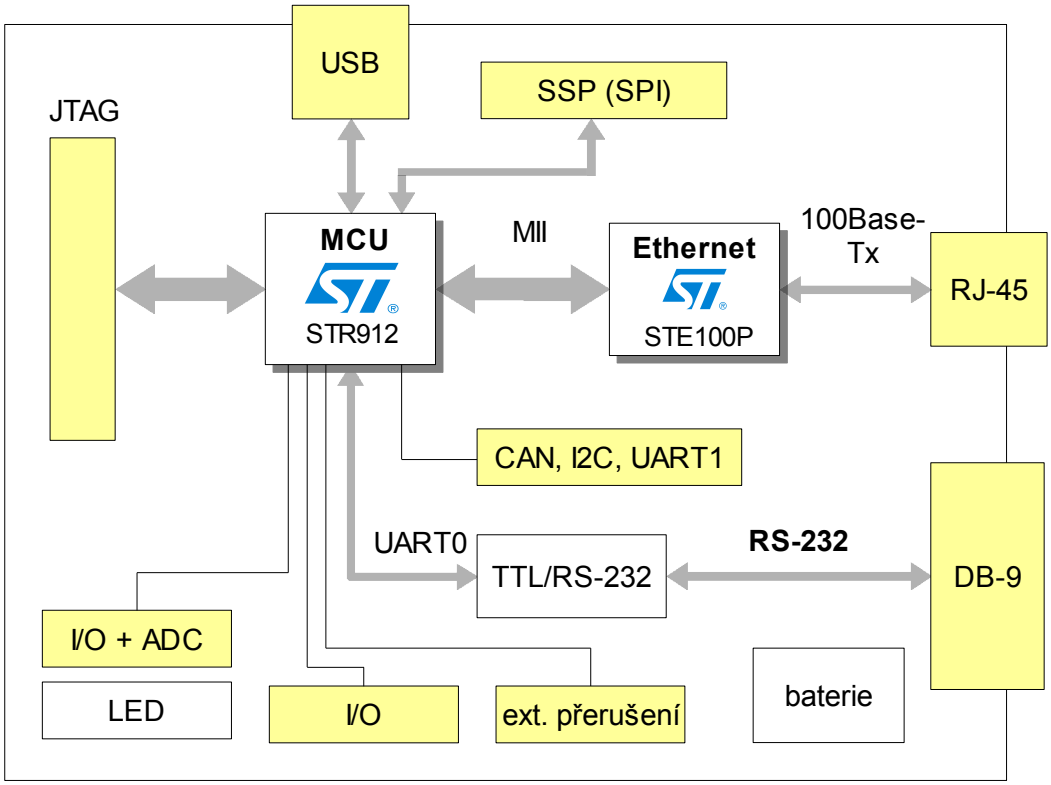
Kostrou vytvořeného modulu je deska plošného spoje. Při jejím návrhu byl kladen důraz v první řadě na správné zapojení obvodů Ethernetu, který je klíčovou částí práce. Postupoval jsem podle aplikačních nót od výrobců součástek, kde jsou popsány detaily, které musí být při návrhu zohledněny, spolu s ukázkovými schematy. Dále byly studovány vývojové kity od několika výrobců, které používají podobných zapojení (viz. tab. 2).

Výrobce	Název produktu	Odkaz na web
STMicroelectronics	STR910-EVAL	www.st.com
Keil	MCB-STR-v3.0	www.keil.com
Hitex	STR912 Eval-Board	www.hitex.com
Embest	STDV912	www.armkits.com
Olimex	STR-E912	www.olimex.com/dev

Tabulka 2: Výrobci vývojových desek a jejich produkty s STR91x.

Kromě obvodů Ethernetu plošný spoj obsahuje další rozhraní jako RS-232, USB, SPI, I/O brány, I2C, CAN. Snahou bylo vyvést co nejvíce periferií na konektory při zachování malých rozměrů plošného spoje. Blokové schéma navrženého modulu znázorňuje obr. 10.

Celý plošný spoj (zejména trojice součástek STR91x, STE100P a konektor RJ-45) byl navržen s ohledem na pravidla návrhu plošných spojů - správné blokování napájení a minimální přerušování spodní vrstvy, kterou tvoří rozlitá zem. Návrh probíhal v programu *OrCAD Layout,* plošný spoj je dvouvrstvý v třídě přesnosti 5, tedy na hranici rozteče pinů pouzdra mikrokontroléru (0,4 mm).



Obrázek 10: Blokové schéma navrženého modulu.

6.2 *Hardwarová specifikace modulu*

- Rozměry: 124×86 mm
- MCU: STR912FAW47X6 s (2048+32) kB interní FLASH a 96 kB interní SRAM
- Napájení: svorkovnice +5.0 V/300 mA nebo USB konektor
- 10/100Mbit Ethernet s DMA (řadič STE100P)
- USB 2.0 (Full Speed 12 MBit/s) SLAVE
- CAN 2.0 / UART1 (bez fyzické vrstvy)
- RS-232 port (UART0)
- SSP, I2C rozhraní
- indikační LED (1 napájecí (red), 2 na síťovém konektoru, 2 (yellow) na I/O)
- 8-bitová brána se signalizačními LED (použití: I/O brána, 8 kanálový 10-bitový ADC)
- druhá 8-bitová I/O brána
- trimr na jednom kanále ADC pro testovací účely
- piny pro externí přerušení

- zálohovací baterie pro RTC jednotku a obsah paměti SRAM
- mikrotlačítko pro reset MCU, 2 mikrotlačítka na I/O pinech
- standardní 20-pinový JTAG konektor

6.3 Napájení modulu

Na modulu se nachází dva konektory pro napájení. Svorkovnice pro připojení stabilizovaného zdroje a USB konektor pro napájení z USB portu počítače. Volba mezi nimi je nastavitelná jumperem na desce modulu (viz. obr. 41).

Na desce jsou osazeny dva stabilizátory, 1,8 V pro jádro mikroprocesoru a 3,3 V pro vstupně-výstupní obvody mikroprocesoru a další obvody modulu.

6.3.1 Proudový odběr modulu

Odběr mikrokontroléru je 1,7 mA/MHz, co je při maximální pracovní frekvenci 96 MHz proud 163 mA. V módu sleep má MCU odběr pouze 55 μA. Odběr ethernetového řadiče je 100 mA při maximálním vytížení. Spolu s dalšími komponentami desky je celkový proud odebíraný modulem **300mA**. S ohledem na proudový odběr musí být dimenzován napájecí zdroj.

6.3.2 Napájení ze svorkovnice

Napájecí napětí musí být stabilizované, minimálně 4,3 V (úbytek na stabilizátorech je cca 1 V). Doporučeno je napětí **5 V**, které **nesmí** být výrazně překročeno. Při vyšším napětí by se stabilizátory příliš zahřívaly. Vstupní kondenzátory jsou dimenzovány pouze na 10 V.

6.3.3 Napájení z USB portu PC

Druhým způsobem napájení modulu je USB port PC. Využívá se napájecího vodiče (+5 V) a zemního vodiče. Podle specifikace může být k USB sběrnici připojeno zařízení s odběrem max. 500 mA (v případě že bude jediným zařízením na celé sběrnici). V praxi to bývá u stolních počítačů i vyšší proud. Při nedostatku výkonových rezerv může systém zařízení odmítnout.

Tento typ napájení modulu je vhodný zejména pro testování programového vybavení z PC, pro jednoduché připojení klasickým USB kabelem typu A-B. Je ale nutné vzít v úvahu

počet současně připojených zařízení a jejich proudové odběry, aby USB sběrnice nebyla přetěžována.

6.4 Mikrokontrolér STR912

6.4.1 Základní rysy

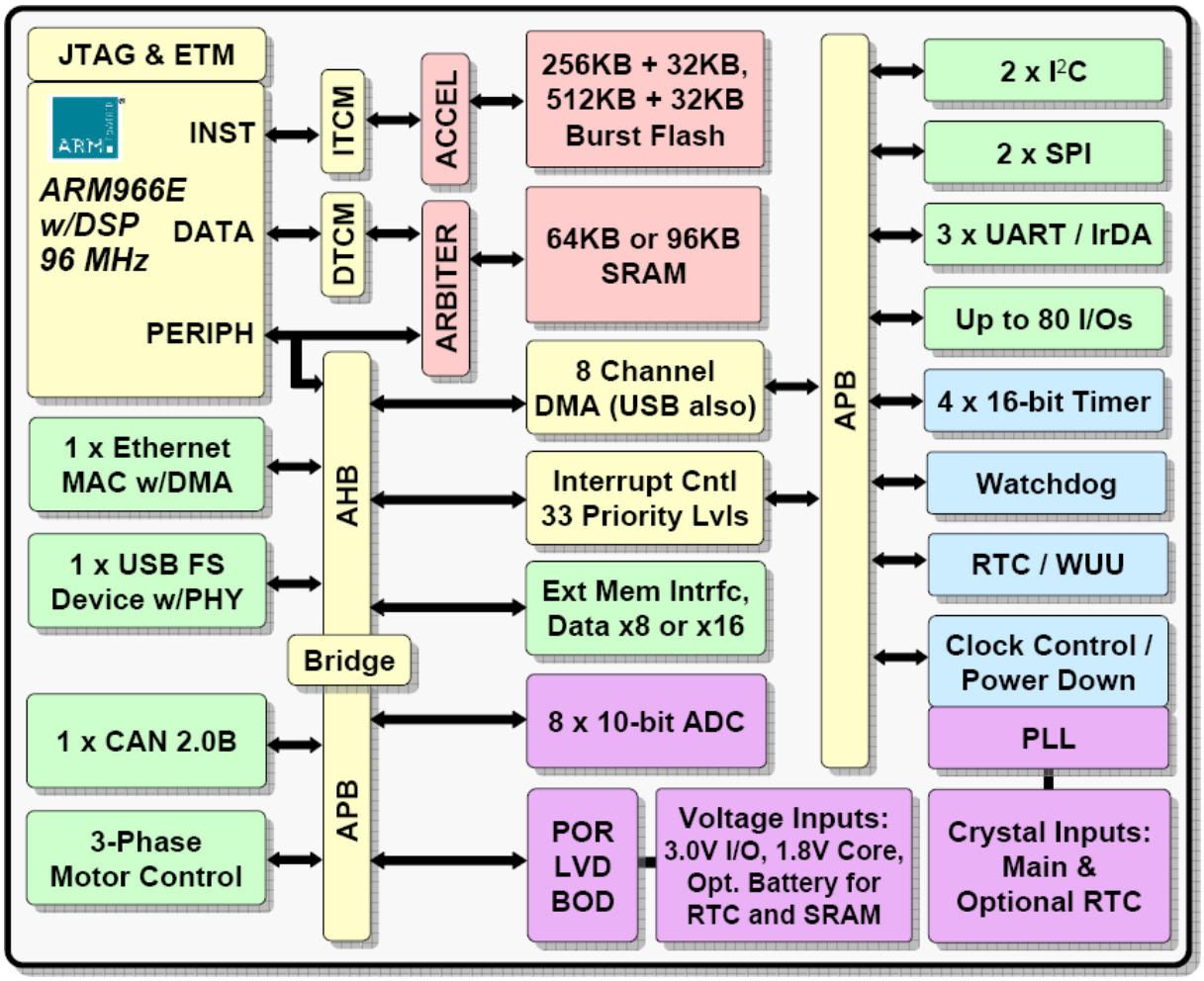
STR91xF je řada mikrokontrolérů s jádrem ARM od firmy STMicroelectronics. Na jednom čipu kombinuje 16/32-bitové jádro ARM966E-S s RISC (redukovaná instrukční sada), dvou-bankovou paměť FLASH, paměť SRAM, a množství periferií. Blokové schema vnitřní struktury je na obr. 11.

Použitý typ má označení STR912FAW47X6 a je ve 128-pinovém pouzdře LQFP128. Jedná se o nejvýkonnější typ z řady STR91x, jeho hlavní vlastnosti znázorňuje tab. 3.

Frekvence (Mhz)	96
Flash (kB)	2048 + 32
RAM (kB)	96
Hlavní periferie	Ethernet, USB, CAN, UART, I2C, SSP

Tabulka 3: Základní vlastnosti mikrokontroléru STR912FAW47X6.

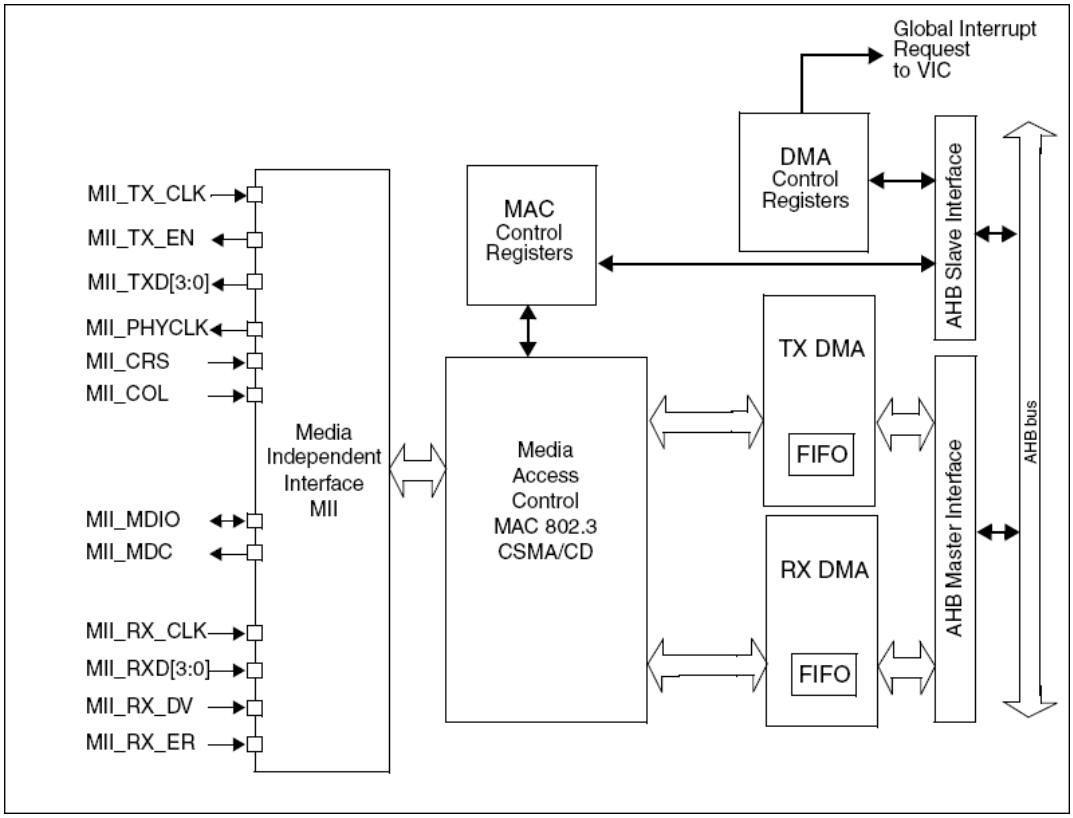
Jádro ARM966E-S má oddělenou paměť dat a programu (Harvardská architektura), což dovoluje procesoru načítat instrukci a současně číst nebo zapisovat data. Výhodou takového uspořádání je významné snížení počtu cyklů na instrukci. Navíc je použita 5-stavová pipeline, která zvyšuje množství paralelních operací a tím i výkon.



Obrázek 11: Blokové schéma vnitřní struktury mikrokontroléru STR912 (převzato z [3]).

6.4.2 Řadič MAC/DMA (ENET)

Mikrokontrolér obsahuje blok ENET, která sestává z řadiče MAC (Media Access Controller) kompatibilního s normou IEEE 802.3, a řadiče DMA. Vnitřní blokové zapojení bloku ENET znázorňuje obr. 12. K řadiči MAC je možné přes standardní rozhraní MII (Media Independent Interface) připojit externí fyzickou vrstvu Ethernetu (PHY). Připojení přes MII je realizováno 18ti signálovými vodiči. Fyzická vrstva je pak připojena do sítě LAN metalickým vedením (kroucenou dvojlinkou), optickým kabelem, či bezdrátově (Wireless LAN), podle použitého obvodu fyzické vrstvy. Obvod fyzické vrstvy použitý v této práci je popsán v kap. 6.5.1.



Obrázek 12: Blokové schéma bloku ENET (MAC/DMA) (převzato z [5]).

MAC vrstva je zodpovědná za zapouzdřování dat do Ethernet rámců před odesláním a separování rámců po přijetí, včetně detekce chyb. Dále řídí přístup ke sdílenému médiu, včetně započetí vysílání dat a zotavení při chybě vysílání.

MAC má následující vlastnosti:

- přenosové rychlosti 10 a 100 Mb/s
- podpora poloduplexního (half duplex) a plně duplexního (full duplex) módu
- v poloduplexním módu je podporována metoda CSMA/CD
- při vysílání generuje a při příjmu ověřuje 32-bitové CRC
- módy filtrování adres pro fyzické a multicast adresy (zahazování rámců)

Ethernet MAC rozhraní umožňuje pro rychlé přenosy dat využít 32-bitový DMA kanál a ušetřit tak procesorový čas. DMA kanál umožnuje:

- přímý přesun vysílaných rámců z SRAM paměti do MAC
- přímý přesun přijatých rámců z MAC do SRAM

6.4.3 Funkce I/O pinů

Během resetu jsou všechny piny na portech 0-9 ve vstupním módu vysoké impedance. V tomto módu zůstanou do té doby, než jim firmware mikroprocesoru přiřadí nějaké funkce. Každému z pinů lze přiřadit jedna z až pěti funkcí. Tato vlastnost rozšiřuje možnosti při návrhu plošného spoje, periferie je možné na piny namapovat více způsoby a vhodně je rozmístit. Přiřazení periferií k pinům pro navržený plošný spoj znázorňuje tab. 4 (význam zkratek je zřejmý ze schematu zapojení, viz. příloha).

Poznámka: všechny GPIO jsou +5 V tolerantní.

	GPIO brány									
	GPIO0	GPIO1	GPIO2	GPIO3	GPIO4	GPIO5	GPIO6	GPIO7	GPIO8	GPIO9
pin 0	ETH	ETH			GPIO4	UART0	EXTINT	GPIO7		
pin 1		ETH			GPIO4	UART0	EXTINT	GPIO7	LED1	
pin 2	ETH	ETH	I2C	CAN	GPIO4	ETH-CLK	EXTINT	GPIO7	LED2	
pin 3	ETH	ETH	I2C	CAN	GPIO4	ETH	EXTINT	GPIO7		
pin 4	ETH	ETH			GPIO4	SSP0-SCK		GPIO7		
pin 5	ETH	ETH		BTN1	GPIO4	SSP0-MOSI		GPIO7		
pin 6	ETH	ETH		BTN2	GPIO4	SSP0-MISO		GPIO7		
pin 7	ETH	ETH			GPIO4	SSP0-NSS	USB SW	GPIO7		

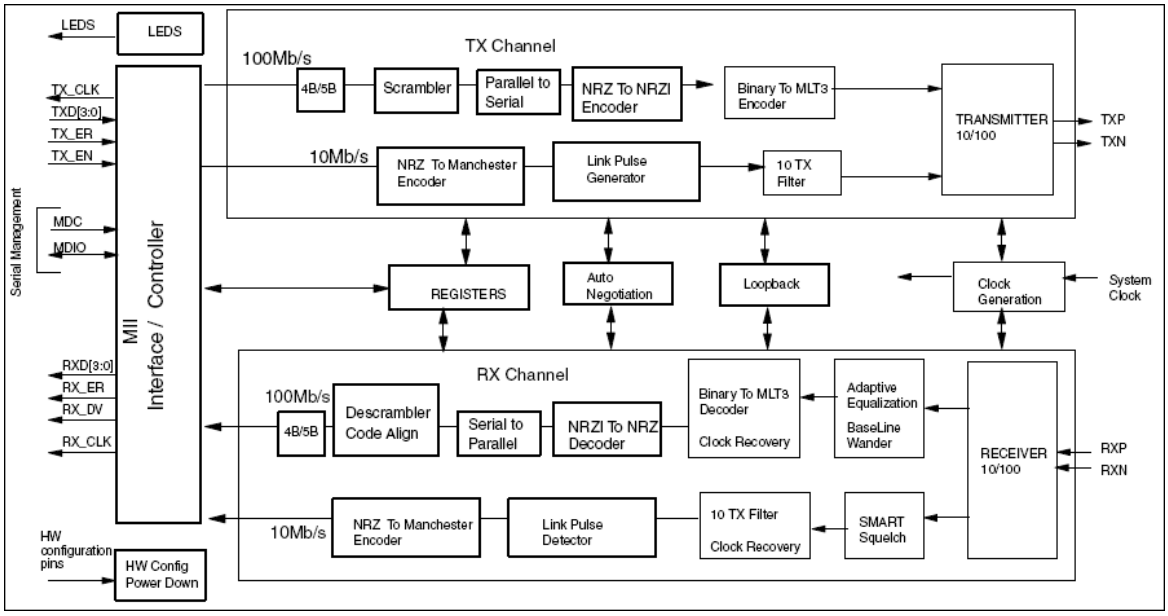
Tabulka 4: Mapování periferií k jednotlivým pinům.

6.5 Ostatní komponenty desky

6.5.1 Obvod fyzické vrstvy – STE100P

Fyzická vrstva obecně definuje přenosové médium (optické vlákno, kroucená dvojlinka, atd.), konektory, elektrické úrovně, kódování pro přenosový kanál atd.

Fyzická vrstva Ethernetu (PHY) je na modulu realizována obvodem STE100P (blokové schéma viz obr. 13). Obvod obsahuje MII (Media Independent Interface) rozhraní, pro připojení k MAC vrstvě mikrokontroléru. Napájecí napětí obvodu je 3,3 V.



Obrázek 13: Vnitřní blokové zapojení obvodu STE100P (převzato z [7]).

Vlastnosti:

- full duplex, half duplex (CSMA/CD) módy v rychlostech 10 a 100 Mbps
- nastavení módu pomocí Auto-Negotiation (nastaví komunikační rychlost na nejrychlejší možnou úroveň s ohledem na rychlost připojeného zařízení)
- fyzické rozhraní 10Base-T nebo 100Base-TX (max. délka segmentu 100m)
- přijímací paketové filtry
- výstupy na signalizační LED konektoru RJ-45 (indikace připojení, aktivity Rx/Tx, kolizí)

Při návrhu plošného spoje bylo potřeba dodržet několik pravidel v souladu s aplikační nótou [7].

Obvod STE100P pracuje na vysokých frekvencích (25 MHz), je proto důležité správné připojení blokovacích kondenzátorů mezi napájecí piny a zem (co nejblíže k obvodu), aby se snížil vysokofrekvenční šum. Analogová a digitální sekce napájení jsou odděleny feritovými tlumivkami, dostatečně dimenzovanými na špičkový proud až 250 mA/STE100P. Ty zabraňují průniku rušení do analogových sekcí obvodu.

K fyzické vrstvě je připojen konektor RJ-45 přes dva páry vodičů RxP, RxN a TxP a TxN. Párové vodiče jsou vedeny co nejblíže u sebe a jsou co možná nejkratší. V jejich nejbližším

okolí nejsou žádné součástky, které by mohly být rušené. Součástí konektoru RJ-45 jsou dvě diody LED. Zelená pro signalizaci připojení do sítě LAN (link), oranžová pro signalizaci příjmu dat.

6.5.2 USB

Rozhraní USB 2.0 je třídy Full-speed (12 Mbit/s), zařízení typu SLAVE. Připojení k USB konektoru je realizováno přes obvod elektrostatické ochrany (ESD), který chrání USB obvody mikrkontroléru před elektrostatickým výbojem.

Připojení k PC lze nastavit jumperem J5 dvěma způsoby (viz. obr. 41). Pokud je J5 v poloze *NORM*, je modul ihned po připojení k PC detekován jako Full-speed zařízení. Pokud je v poloze *SW DRIVE*, je připojení/odpojení od PC řízeno pinem P6.7 mikroprocesoru (softwarově).

6.5.3 Sériové rozhraní UART

Modul obsahuje dvě sériová rozhraní UART, velmi podobných průmyslovému standardu UART 16C550.

První z nich (UART0) je připojeno přes převodník úrovní TTL/RS-232 na konektor DB-9. Vyvedeny jsou signály Rx, Tx a GND. Hardwarové řízení toku zapojeno není. UART0 se propojuje s okolím sériovým prodlužovacím kabelem (bez křížení).

Druhé sériové rozhraní (UART1) je vyvedeno na universální piny přímo (bez převodu úrovní).

6.5.4 Sériové rozhraní SSP (SPI)

Mikroprocesor obsahuje dvě nezávislá SSP rozhraní. Na modul je vyvedeno jedno z nich, SSP0. SSP podporuje standardní průmyslový SPI protokol, ale také SSI a Microwire protokoly. Vlastnosti:

- 3 nebo 4-drátová plně duplexní sběrnice (MISO, MOSI, SCK, NSS)
- MASTER nebo SLAVE mód
- programovatelný hodinový kmitočet
- programovatelná polarita a fáze
- nastavitelná velikost datového rámce

SPI je obecně určeno pro komunikaci mezi mikroprocesory a dalšími integrovanými obvody na krátké vzdálenosti. Vyvedený konektor umožňuje připojit externí zařízení.

6.5.5 GPIO (vstupy/výstupy), AD převodník

Na universální konektor jsou z MCU vyvedeny dvě osmibitové I/O brány, GPIO4 a GPIO7. Obě jsou chráněny rezistory 330 R v sérii.

Možnosti použití **GPIO4**:

- digitální I/O brána (pro čtení nebo zápis logických stavů)
- analogový vstup pro osmikanálový analogově-číslicový převodník
- brána vyvedena na jeden konektor (piny P4.0 – P4.7)
- signalizace aktuálního stavu brány přes budič na LED diody

Možnosti použití **GPIO7**:

- pouze jako digitální I/O brána
- rozdělena na dva samostatné konektory (piny P7.0 – P7.5 a P7.6 – P7.7)

Externí přerušení

Pro možnost připojení zdroje externího přerušení pro mikrokontrolér jsou na konektor vyvedeny čtyři piny P6.0 – P6.3.

AD převodník

- 8-kanálový
- 10-bitový s postupnou aproximací, čas konverze 0,7 us
- vstupní rozsah 0 – 3.6 V
- na kanál 0 je pro testovací účely připojen trimr (který je možné jumperem odpojit)

6.5.6 RESET

Tlačítko RESET je připojeno na pin RESET_IN mikrokontroléru. Po jeho stisku je vyvolán systémový reset a program v mikrokontroléru se začne vykonávat od začátku.

6.5.7 Tlačítka

K dispozici jsou dvě uživatelská tlačítka, BTN1 a BTN2.

6.5.8 JTAG

Standardní 20-pinový JTAG konektor je určen k připojení debuggeru (ULINK, j-link). Přes JTAG konektor je nahráván program do MCU.

6.5.9 Zálohovací baterie

Pro zachování obsahu RAM paměti i po odpojení napájení je připojena zálohovací lithiová baterie. Je určena také pro udržování času reálných hodin a data RTC jednotky.

6.6 Knihovny ovladačů periferií 91x_lib a 91x_enet

Firma STMicroelectronics nabízí pro mikrokontroléry STR91x knihovnu ***91x_library***, napsanou v jazyce C. Knihovna obsahuje soubor rutin, datových struktur a maker, které tvoří ovladače pro všechny standardních periferie MCU. Ovladač každé periferie sestává ze sady funkcí, které pokrývají funkcionalitu periferie. Knihovna usnadňuje vývoj programů, není nutné podrobně studovat specifikace periferií a jednotlivé registry. Pro obsluhu periferie ENET (řadič MAC/DMA) je k dispozici samostatná knihovna ***91x_enet***. Ta bude využita k odesílání a příjmu paketů od síťového rozhraní.

Seznam ovladačů periferií, které byly použity při tvorbě firmwaru mikrokontroléru:

- 91x_scu: System Control Unit (jednotka pro napájení, reset, hodinový signál)
- 91x_fmi: Flash Memory Interface
- 91x_adc: analogově/digitální převodník
- 91x_gpio: konfigurace I/O pinů, čtení, zápis na piny
- 91x_rtc: jednotka reálného času
- 91x_uart: periferie UART
- 91x_enet: periferie ENET (Ethernet)
- 91x_vic: řadič přerušení
- 91x_it: funkce obsluhy prerušení
- 91x_conf.h: konfigurační soubor, definice periferií které budou použity

Knihovny jsou dobře zdokumentovány (viz. [8],[9]) a existují k nim ukázkové příklady jejich použití – examples (viz. [3]).

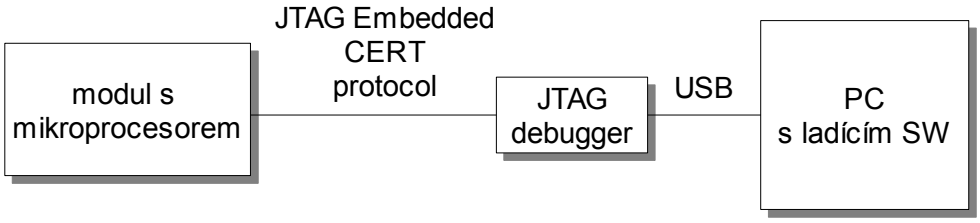
7 Metody nahrávání firmwaru a vývojová prostředí

7.1 Nahrávání přeloženého binárního souboru do MCU

Kód programu pro mikrokontrolér se potřeba zkompilovat překladačem, výsledkem je binární soubor (firmware). Ten je potřeba do MCU nějakým způsobem nahrávat. Pro tento typ MCU to lze provést několika způsoby. Prvním a v této práci jediným využívaným způsobem je programování přes JTAG konektor. To je metoda ISP. Dále existuje možnost programování metodou IAP. Obě tyto metody jsou popsány v následujících podkapitolách.

7.2 Programování metodou ISP a debug

Metodou ISP (In System Programming) lze mikroprocesor naprogramovat přes rozhraní JTAG. K tomu je potřeba debugger, zařízení které převádí ladící data a příkazy mezi USB sběrnicí počítače a JTAG protokolem v mikroprocesoru (viz. obr. 14).



Obrázek 14: Připojení JTAG debuggeru.

JTAG rozhraní v mikroprocesoru STR91x umožňuje:

- programovat celou paměť FLASH
- ladění (debug) běžícího programu

Po nahrání firmwaru přes JTAG mikroprocesor zavádí program z paměti FLASH, a to z jedné ze dvou bank. V základním nastavení z banky 0. Tento typ MCU nemá bootloader, jako například mikrokontroléry řady STR75x.

Programování i odesílání ladících příkazů probíhá ve vývojovém prostředí, ve kterém je kód překládán. Příkladem ladících operací může být spuštění, zastavení, či krokování

programu, nastavování breakpiontů (míst v kódu, kde bude vykonávání programu pozastaveno), watchpointů, atd.

7.3 Programování metodou IAP

IAP (In Application Programing) je metoda programování paměti FLASH v cílovém zařízení za běhu programu. Program je možné nahrávat do MCU využitím některého rozhraní mikroprocesoru (UART, CAN, USB, Ethernet). Abychom mohli využívat IAP, je nutné do MCU nahrát ovladač IAP pro vybrané rozhraní, které chceme pro programování používat.

K nahrání ovladače je potřeba:

- hardwarové přemapování banky bank1 FLASH paměti na adresu 0x00 (bank0 bude přemapována na adresu 0x80000)
- nahrání ovladače do bank1 FLASH přes JTAG pomocí programátoru/debuggeru

Nahrávání programu přes zvolené rozhraní (UART):

- odeslání binárního souboru s programem z Hyperterminálu MS Windows přes UART
- IAP ovladač uložený v bank1 mikroprocesoru tento soubor ukládá do bank0
- softwarové přemapování bank0 na adresu 0x00 a bank1 na 0x80000
- po skončení přenosu skok na adresu bank0 a kde se začne vykonávat uživatelský program.

Firma STMicroelectronics nabízí na svých internetových stránkách zdrojové kódy ovladačů pro IAP. Pro STR91x jsem testoval ovladač pro IAP přes UART, který se podařilo správně nahrát do bank1, ale na soubor odeslaný z Hyperterminálu již ovladač nereagoval. Chyba mohla být v chybném přemapování paměti, které by mělo být provedeno pomocí nástroje CAPS, nebo v kódu ovladače.

Pro další experimenty s IAP doporučuji zkontrolovat webové stránky výrobce [3], zda nevyšla novější verze ovladače. Ke stažení je také ovladač IAP přes Ethernet.

7.4 Vývojová prostředí pro kompilaci kódu

Kód, který má být v mikroprocesoru spuštěn, musí být nejprve přeložen kompilátorem pro cílový procesor. Existuje řada programů pro kompilaci, které podporují řadu STR91x.

Některé jsou komerční, nabízené ke stažení pouze v omezené verzi, jiné jsou zdarma, ale zase neobsahují tolik možností.

Testované programy a jejich vlastnosti:

- Keil uVision3 V3.60
- IAR Embedded Workbench for ARM
- Raisonance RIDE 6.10
- Hitex HiTOP 5.2 – Tantino, Tanto

Výv. prostředí	uVision	EWARM	RIDE	HiTOP
možnosti	kompilace/debug	kompilace/debug	kompilace/debug	kompilace/debug
podpora ULINK	ANO	NE	NE	NE
podpora J-Link	ANO	ANO	NE	NE

Tabulka 5: Možnosti vývojových prostředí.

Výběr vhodného kompilátoru zabral poměrně dost času. Ani jeden ze čtyř testovaných nešlo použít jednoduše pro celý proces kompilace. Některé části kódu bylo potřeba odladit a protože k dispozici byly JTAG debuggery ULINK a J-Link, výběr ladících programů se zúžil na uVision a EWARM, které jsou ale zdarma jen v omezených verzích.

Omezení uVision RealView Microcontroller Development Kit:

- programy větší než 32 kB paměti programu a dat nelze kompilovat, assemblovat, linkovat ani ladit debuggerem
- kompletní přehled omezení je na adrese http://www.keil.com/demo/limits.asp

Omezení volně stažitelné verze IAR Embedded Workbench:

- verze *Kickstart* má omezení velikosti kódu na 32 kB, ale není omezena časově, je vhodná pro menší programy
- verze *Evaluation* je časově omezena na 30 dní, pak se stane nefunkční; velikostí kódu omezena není, je proto vhodná pro programy, které nelze kvůli své velikosti přeložit v *Kickstart* verzi

7.4.1 Evaluation verze EWARM instalovaná ve virtuálních Windows

Velikost zdrojových kódů, které jsem překládal, byla větší než limit 32 kB. Musel jsem tedy použít verzi *Evaluation*. Po vypršení časově omezené licence ale nestačí program odinstalovat a znovu s novou licencí nainstalovat. Předchozí licence je totiž zapsána v registrech operačního systému Windows, odkud se mi ji nepodařilo odstranit.

Tuto situaci jsem obešel instalací virtualizačního nástroje který umožňuje spouštět další (hostované) operační systémy uvnitř hostitelského OS. Použil jsem *VirtualBox* od firmy *Sun Microsystems* a vytvořil v něm novou instalaci Windows XP. Celý operační systém se uloží do jednoho souboru **.vdi*, který je potřeba před instalací *Evaluation* verze programu EWARM zálohovat. Když máme vytvořenu zálohu, můžeme nainstalovat *Evaluation* verzi EWARM. Po 30ti dnech používání programu a vypršení licence stačí obraz s virtuálními Xp odstranit, ze zálohy zkopírovat počáteční *.vdi obraz a nainstalovat v něm EWARM s novou licencí, opět použitelnou na 30 dní.

Tento postup se může zdát jako složité řešení, ale chceme-li ladit programy větší než 32 kB, je toto jedna z možností. Nevýhodou je o něco pomalejší překlad kódu a také nutnost nastavit v programu *VirtualBox* ty hardwarové komponenty počítače, které chceme na virtuálním stroji používat (USB, sériový port, síťová karta, sdílené disky). Stejně lze *VirtualBox* používat pod OS Linux.

7.4.2 Evaluation verze uVision3

Jak již bylo zmíněno výše, testovací verze prostředí *uVision3* má stejně jako *EWARM Kickstart* limit kódu pro překlad. uVision3 bude v této práci využíváno k nahrávání binárního souboru **.hex* do mikroprocesoru (omezení testovací verze platí pro kompilaci a debug, nahrávání programu do MCU je bez omezení).

Instalace Keil RealView Microcontroller Development Kit V3.20 je jednoduchá, není třeba ji popisovat. Byla používána verze 3.20 (viz. *DVD\prekladace\Keil devtool*), nejnovější verze je dostupná na adrese www.keil.com.

Poznámka: ukázkové projekty v adresáři s instalací: \Keil\ARM\Examples\ST\STR91xLib

7.4.3 Prostředí RIDE (Raisonance IDE)

Toto vývojové prostřední od výrobce Raisonance podporuje řady mikroprocesorů 8051, ARM, ST7, STM8 a další. Programový balík Rkit-ARM pro mikroprocesory ARM je

použitelný bez jakýchkoli omezení, což je jeho velké plus a proto jsem jej zvolil pro další práci. Postup nastavení programu bude popsáno podrobněji. Spolu s prostředím RIDE je potřeba nainstalovat GNU ARM kompilátor (www.gnuarm.com) který je pod GNU GPL licencí, tedy volně dostupný.

Instalace balíku RKit

- spustit instalátor Raisonance RKit 6.1 (DVD příloha: *prekladace\Rkit_6_1.exe*)
- vybrat součásti: STRx ToolChain, GNUARM 4.0

Otevření projektu:

- nejjednodušší je vycházet z již hotového projektu, *Project → Open*
- prostředí RIDE s otevřeným projektem a zobrazením hlavních částí programu viz. obr. 15.

Vytvoření nového projektu:

nastavení cílového procesoru:

- *Options → Target → Device → STR912FAW47*

nastavení typu paměti pro kód:

- *Options → Target → Options → Boot Mode → FLASH*

zadání cest ke zdrojovým souborům **.h* a **.inc*:

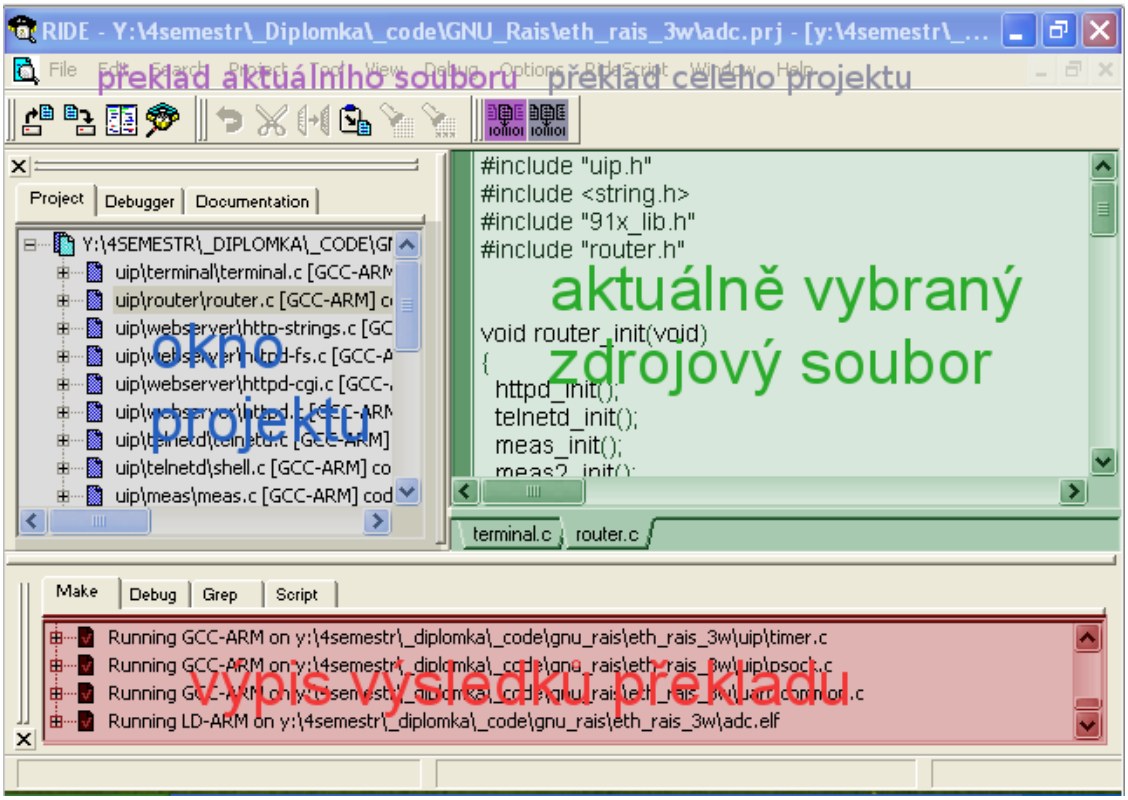
- *Options → Project → Directories → Include*
- zadávat relativní cesty od projektu (**.prj*), nebo absolutní cesty
- oddělovačem mezi jednotlivými cestami je středník

přidání složky, zdrojového souboru

- v okně projektu pravé tlačítko myši - *Add group, Add node*

Kompilace kódů

Kompilace se provádí pomocí *Make* nebo *Build*. Výsledkem kompilace je, mimo jiné, binární soubor *.hex.



Obrázek 15: Rozvržení komponent programu RIDE.

7.4.4 Nahrání souboru *.hex do mikrokontroléru

Binární soubor **.hex*, který je výsledkem překladu (kompilace) kódu lze nahrát do mikroprocesoru přes rozhraní JTAG (metoda ISP). Použijeme programátor ULINK v kombinaci s vývojovým prostředím Keil uVision3. Následující postup ukazuje nastavení projektu v uVision3 pro nahrávání souboru **.hex*:

Project → *New uVision Project*

- zadáme jméno souboru, vybereme cílový mikroprocesor (STMicroelectronics, STR912FAW47), start-up soubory do projektu nepřidáváme
- v *Project Workspace*, *Target1* – pravé tlačítko – *Options for Target* → Output → *Name for Executable* – zadáme cestu k hex souboru včetně názvu s příponou (*..\soubor.hex*)
- binární soubor nahrajeme z menu *Flash* → *Download*

Aby se program v mikrokontroléru spouštěl ihned po nahrání, nastavíme následující:

Flash → *Configure Flash Tools* → *Utilities* → *ULINK* → *Settings* → *Reset and Run*

8 uIP TCP/IP stack a aplikace

Jak již bylo řečeno, pro tuto práci byl zvolen uIP TCP/IP stack s open-source licencí. Zajišťuje TCP/IP konektivitu pro malá embedded zařízení, především pro mikrokontroléry s omezenými prostředky programové a datové paměti.

uIP může běžet samostatně v hlavní smyčce programu, nebo jako úloha v multitasking systému. uIP podporuje celou řadu protokolů z rodiny TCP/IP, od protokolů nejnižších vrstev jako ARP, který překládá IP adresy na MAC adresy, až po protokoly aplikační vrstvy OSI modelu, například HTTP.

Hlavní výhody uIP:

- dobře zdokumentovaný a okomentovaný zdrojový kód
- malé nároky na paměť FLASH i RAM
- podpora protokolů ARP, IP, UDP, ICMP (ping) a TCP
- podpora několika současně aktivních TCP spojení
- podpora několika současně naslouchajících serverů
- zdarma pro komerční i nekomerční použití
- RFC kompatibilní implementace protokolů TCP a IP, včetně řízení toku

Nevýhody:

- není implementováno časové okno pro TCP, což se negativně projevuje na výkonu

8.1 Klíčové mechanismy uIP stacku

8.1.1 Paketový buffer uIP, buffery síťového čipu

uIP používá jeden globální paketový buffer pro přijatá i odesílaná data a fixní tabulku, ve které jsou udržovány informace o spojení. Velikost bufferu je definovatelná, v základu je nastaven na maximální velikost paketu. Přijde-li ze sítě nový paket, ovladač síťového zařízení ho uloží do globálního paketového bufferu a dá to na vědomí TCP/IP stacku. Ten paket zpracuje a zjistí-li, že obsahuje aplikační data, zavolá příslušnou aplikaci. Aplikace v první řadě musí překopírovat data do vlastního bufferu, pokud s nimi chce dále pracovat, protože globální buffer může být po návratu z aplikace přepsán novým příchozím či odchozím

paketem. Přijde-li ze sítě nějaký další paket v okamžiku, kdy aplikace zpracovává paket současný, musí být zařazen do fronty.

To není záležitostí TCP/IP stacku, ale síťového rozhraní s buffery na čipu. V této práci použitý obvod STE100P má velikost přijímacího bufferu 4 kB a vysílacího také 4 kB. Pokud se zaplní, budou další pakety zahazovány, což vede ke snížení rychlosti. Proto je důležité nesetrvávat v aplikaci příliš dlouhou dobu a neblokovat běh hlavní smyčky programu. Ta periodicky kontroluje stav fronty příchozích paketů, aby některé nebyly zahozeny. Pokud je ke kopírování paketů mezi síťovým rozhraním a aplikačním bufferem využit DMA kanál, bude operace kopírování prováděna bez účasti procesoru.

8.1.2 Hlavní smyčka programu

Hlavní smyčka programu (viz. obr. 16) je umístěna v dolní části souboru *main.c*, představuje ji nekonečný cyklus “while(1) {}“. Periodicky kontroluje dvě podmínky:

- jestli přišel paket ze sítě
- jestli nastal periodický timeout

Podle toho jsou volány příslušné funkce uIP stacku (funkce s prefixem uip_).

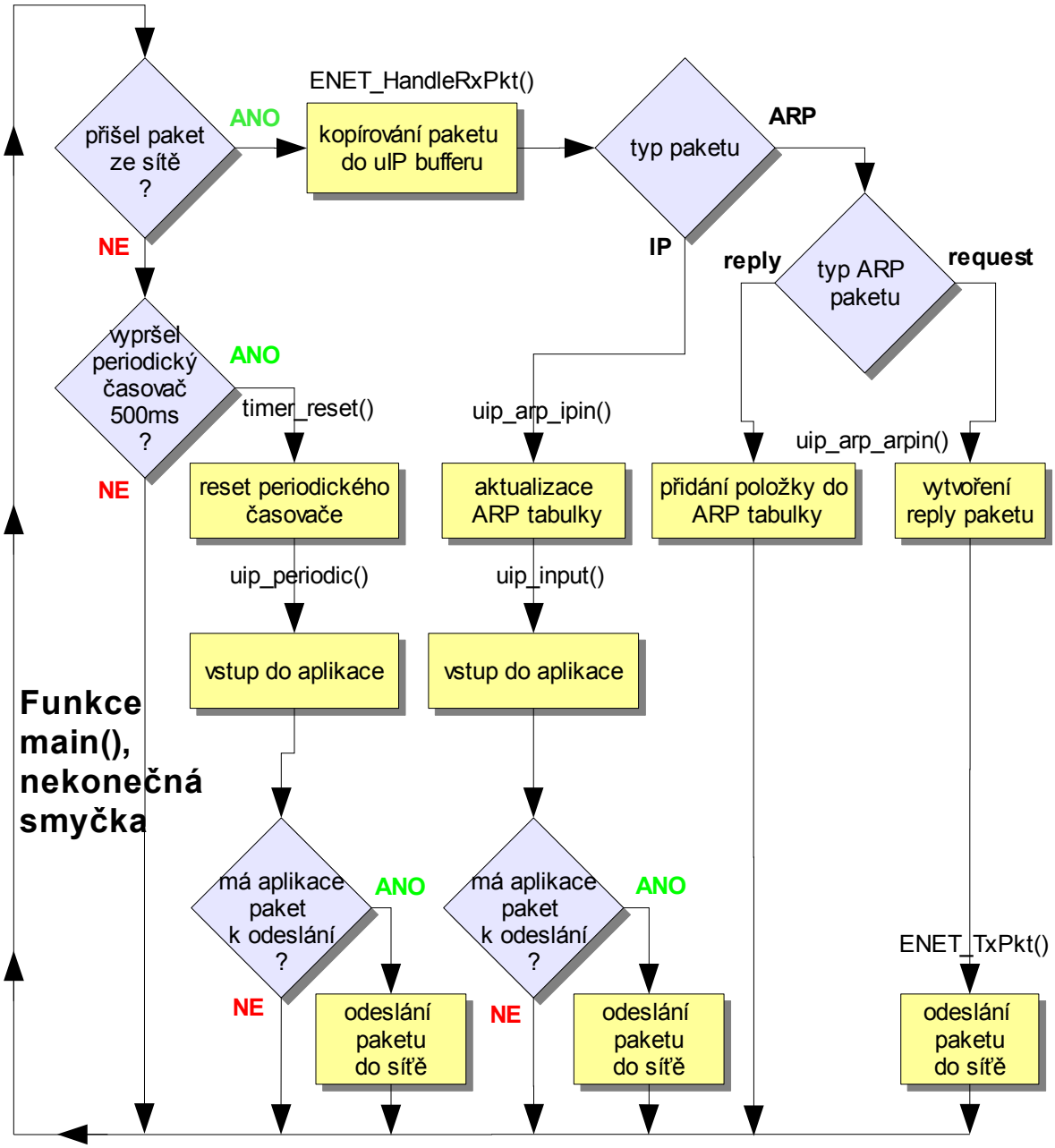
8.1.3 Zatížení procesoru zpracováním paketů

Byl měřen čas, který uIP stack potřebuje ke zpracování příchozího paketu a následné odeslání odchozího paketu maximální délky. Měření bylo samozřejmě závislé na typu mikrokontroléru (STR912, frekvence jádra 96MHz, DMA kanál). Z PC byl odeslán paket (1 byte TCP/IP dat) a měřen čas do přijetí paketu (1500 bytů TCP/IP dat) od modulu.

Následující operace uIP stacku jsou časově nejnáročnější:

- výpočet IP,TCP kontrolních součtů (CRC) příchozího paketu
- přepínání kontextu
- operace kopírování odchozího paketu z aplikace do uIP bufferu
- výpočet IP,TCP kontrolních součtů odchozího paketu

Naměřený čas se rovnal přibližně **0,9 ms**. Po tento čas je běh hlavní smyčky blokován uIP stackem (pokud je tedy zpracováván paket). To je potřeba vzít v úvahu, pokud chceme v hlavní smyčce provádět další akce.



Obrázek 16: Hlavní smyčka programu v souboru main.c.

Když je paket přijat, z hlavní smyčky by měla být vyvolána vstupní obslužná funkce *uip_input()*. Tato vstupní funkce vždy okamžitě vrací a stack nebo aplikace pro kterou byl příchozí paket určen může produkovat jeden nebo více "reply" paketů, které budou odeslány do sítě. Pak by měl být volán ovladač síťového zařízení k odeslání těchto paketů.

K řízení TCP mechanismů (které jsou závislé na časovačích) jsou použity periodické časovače. Když hlavní smyčka usoudí, že timeout vypršel, měla by se zavolat obslužná funkce časovače *uip_periodic()*.

8.1.4 Zpožděné potvrzování ACK pakety

Většina TCP "přijímačů" má implementován takzvaný "Delayed ACK" algoritmus (zpožděné potvrzování přijatých dat), který slouží ke snížení počtu odeslaných prázdných ACK paketů (vysvětlení ACK viz. kap. 3.4.4). TCP přijímač používající tento algoritmus bude posílat ACK až po posledním přijatém TCP segmentu (počet segmentů bez potvrzení závisí na velikosti časového okna). ACK je poslán také v případě, že není přijat žádný segment ve specifickém časovém okně. Velikost časového okna může být až 500 ms, ale obvykle používanou hodnotou je 200 ms.

TCP odesílatel (v tomto případě uIP), který v jednom časovém okamžiku může obsluhovat jen jeden odchozí TCP segment, bude reagovat pomalu, pokud komunikuje se zařízením používajícím Delayed ACK. Protože přijímač přijímá pouze jeden segment v čase, bude čekat 200 ms předtím, než může být ACK odeslán. Tímto zpožděním je maximální přenosová rychlost velmi omezena.

Degradace přenosové rychlosti kvůli Delayed ACK se projeví pouze když uIP vysílá data, při příjmu dat se neprojeví. Maximální přenosová rychlost může být vyjádřena rovnicí

$$p = \frac{s}{t + t_d}$$

kde:

s (B) ... velikost segmentu (v bytech)

t (s) ... doba, pro přenos informace od zdroje k cíli a zpět

t_d (s) ... velikost časového okna (typicky 200-500 ms)

p (B/s) ... maximální přenosová rychlost

Uvažujeme-li s = 1500, t_d = 200 ms, t zanedbáme, tak nejvyšší dosažitelná rychlost je **7 kB/s**.

Degradaci výkonu lze zabránit úpravou, která je popsána v dokumentaci uIP stacku v kapitole "uIP TCP throughput booster hack" (je potřeba přidat modul *uip-split* do kódu uIP). Trik spočívá v rozdělení odchozího TCP segmentu do dvou. Tuto úpravu jsem

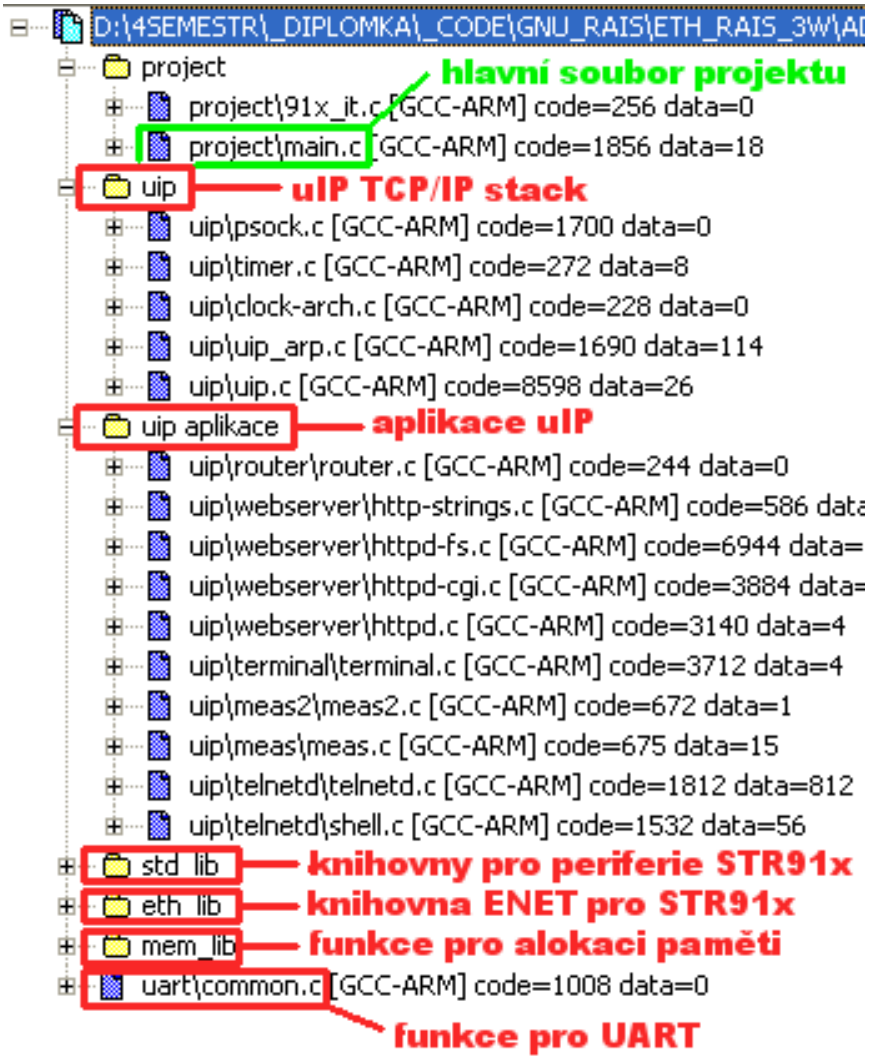
vyzkoušel, avšak nepodařilo se mi kód zkompilovat. Podle zdrojů na Internetu by ale tato úprava měla být funkční a použitelná.

V aplikacích, kde bude potřeba dosahovat vyšších rychlostí přenosu, bude použita metoda odesílání potvrzovacích zpráv od příjemce zpět, okamžitě po příjmu dat. Zprávy odesílá komunikující program (musí tak být naprogramován). Při příjmu paketu je aplikace v uIP vyvolána okamžitě, bez čekání na paket s příznakem potvrzení (ACK). Tímto způsobem bude dosaženo rychlostí několikanásobně vyšších.

8.2 Zdrojové kódy uIP a jejich struktura

Zdrojové kódy poslední verze uIP 1.0 včetně dokumentace jsou ke stažení na adrese [11]. Z této verze jsem při vývoji firmware vycházel, přičemž všechny změny které byly ve zdrojových kódech provedeny jsou doplněny komentáři přímo v kódu, nebo jsou zřejmé z textu této práce. Kód celého projektu sestává z několika částí, struktura zdrojových kódu v projektu vývojového prostředí RIDE je zobrazena na obr. 17.

Poznámka: *DVD příloha obsahuje adresáře se zdrojovými kódy pro různá vývojová prostředí a v řadě verzí. Všechny zdrojové kódy pro mikroprocesor, které budou v této práci popisovány, se budou týkat adresáře DVD\STR912_modul\code\RIDE_GNUARM\eth_rais_3w. Projekt v tomto adresáři obsahuje všechny vytvořené aplikace, které po nahrání do MCU běží současně. Projekt je vytvořen v prostředí RIDE (viz. kap. 7.4.3).*



Obrázek 17: Struktura zdrojových souborů (vývoj. prostředí RIDE).

8.2.1 Zarovnání dat v paměti (Alignment)

V některých zdrojových souborech uIP stacku je přistupováno ke strukturám adresově. Tyto struktury obsahují další struktury či proměnné, u kterých je potřeba zajistit, aby byly zarovnány v paměti těsně za sebou. Bez toho nemůže stack fungovat, proměnné by byly v paměti posunuté.

Příklad takové struktury v souboru uip_arp.h:

```
struct uip_eth_hdr {
  struct uip_eth_addr dest;
  struct uip_eth_addr src;
  u16_t type;
};
```

Která používá následující strukturu v souboru uip.h:

```
struct uip_eth_addr {
  u8_t addr[6];
};
```

Přístup k proměnné *type* adresově (v souboru main.c):

```
#define BUF ((struct uip_eth_hdr *)&uip_buf[0])
...
BUF->type == ...
```

Kód uIP předpokládá, že použitý překladač bude zarovnávat struktury 1-bytově, tedy těsně za sebe. Takto zarovnává překladač IAR EWARM. Jiné překladače, například GNU, Keil, zarovnávají 4-bytově, tedy na celá slova, protože pracujeme s 32-bitovým procesorem. Je možné, že 1-bytové zarovnávání lze vnutit překladači nastavením některého přepínače při kompilaci, ale nebyl nalezen způsob jak toto provést. Problém byl vyřešen přidáním atributu *__attribute__ ((packed, aligned (1)))* za klíčové slovo struct.

Takto vypadá upravená definice struktury uip_eth_addr:

```
struct __attribute__ ((packed, aligned (1))) uip_eth_addr {
  u8_t addr[6];
};
```

Protože byl pro kompilaci používán překladač GNU, musely být atributy přidány ke všem strukturám, u kterých to bylo potřeba.

8.3 Aplikace uIP

8.3.1 API (Application Program Interface)

API definuje způsob, jak bude aplikační program interagovat s TCP/IP stackem. Běžně používané API pro TCP/IP je BSD socketové API, které je použité na Unixových systémech, nebo jemu podobné Microsoft Windows WinSock API. Tyto API potřebují podporu od multitasking (víceúlohového) systému a v uIP nejsou implementovány. uIP zajišťuje dvě API: protosockety (zjednodušené BSD socket API) a "raw" API, založené na událostech, které pracuje na nižší úrovni než protosockety, ale používá méně paměti.

uIP „raw" API používá rozhraní řízené událostmi. Aplikace běžící nad uIP je implementována jako funkce v jazyce C, která je volána uIP stackem jako odezva na jisté

události. uIP volá aplikaci když jsou data přijata, když data byla úspěšně odeslána druhé straně, nebo bylo navázáno nové spojení s druhou stranou, nebo pokud data musí být odeslána opakovaně. Aplikace je také periodicky vyvolávána ke kontrole, zda má nějaká data k odeslání. To se děje pomocí pouze jedné „callback“ funkce a je na aplikaci, jakým způsobem bude směrovat data podle portů k jednotlivým službám (podaplikacím). Protože aplikace je schopna reagovat na příchozí data a požadavky od zpracovávaného spojení okamžitě, jakmile TCP/IP stack přijme paket, může být i v takto jednoduchém systému dosaženo rychlé odezvy.

8.3.1.1 Aplikační události

Aplikace musí být implementována jako funkce v jazyce C, *UIP_APPCALL*, kterou uIP volá vždy když nastane událost. Každé události přísluší testovací funkce, která je použita k rozlišení mezi různými událostmi. Testovací funkce jsou implementovány jako C makra, která jsou vyhodnocována jako nulová nebo nenulová. Jisté události se mohou stát současně s jinými (např. mohou přijít nová data a současně být data potvrzena).

8.3.1.2 Přijímání dat

Když je uIP testovací funkce *uip_newdata()* nenulová, znamená to, že vzdálený host na daném spojení právě odeslal nová data. Na nová data ukazuje ukazatel *uip_appdata*. Velikost dat je přístupná funkcí *uip_datalen()*. Důležité je, že data budou po návratu z aplikační funkce přepsána jinými. Proto aplikace musí na příchozí data přímo reagovat, nebo je zkopírovat do vlastního bufferu pro pozdější zpracování.

8.3.1.3 Odesílání dat

Aplikace odesílá data použitím funkce *uip_send()*, která má dva parametry – ukazatel na data k odeslání a délku dat. Aplikace může uložit data přímo do paketového bufferu, na ukazatel *uip_appdata* voláním funkce *uip_send(uip_appdata, len),* kde *len* je délka dat. Tím lze ušetřit místo v RAM paměti, pokud jí je nedostatek, a také proces odesílání dat urychlit, protože se data nemusí do paketového bufferu kopírovat. Aplikace umožňuje odeslat jen jeden paket/spojení, není možné volat *uip_send()* při obsluze aplikace více než jednou, jinak budou odeslána jen data z posledního volání.

8.3.1.4 Polling (dotazování)

Když je spojení nečinné, uIP se dotazuje aplikace vždy, když vyprší periodický časovač. Aplikace používá testovací funkci *uip_poll()* ke kontrole, zda uIP vznesl dotaz.

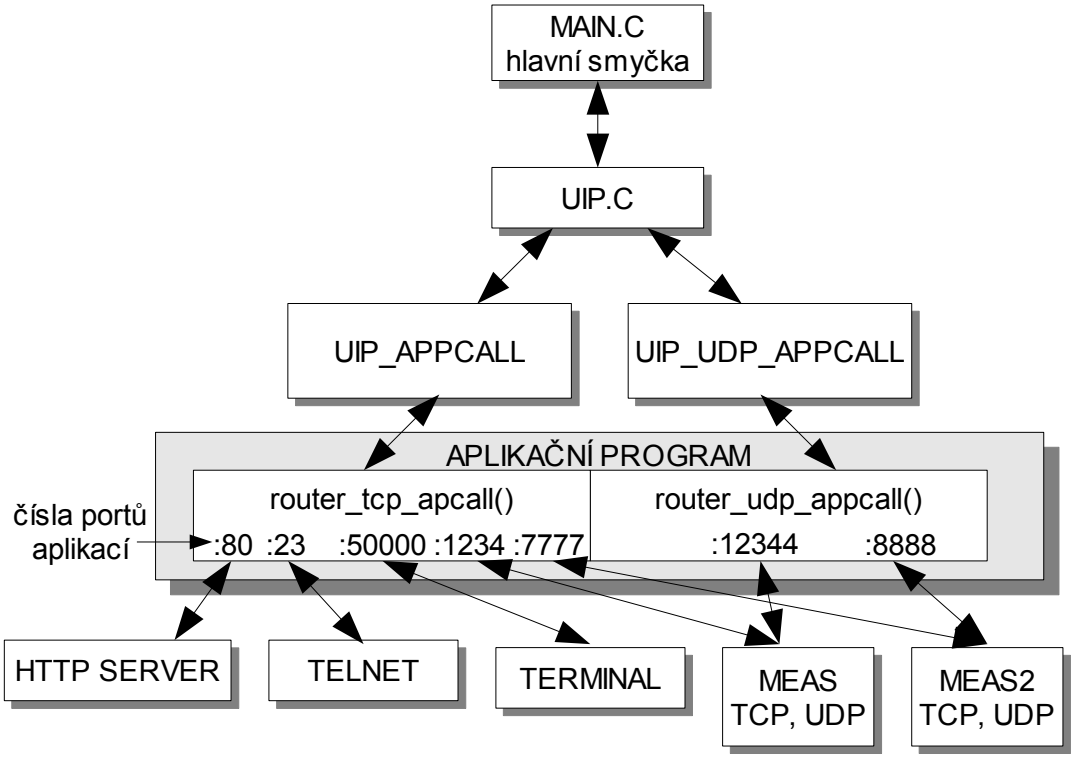
Dotazování má dva účely. Prvním je dávat aplikaci periodicky najevo, že spojení je nečinné, což dovoluje aplikaci zavřít spojení, která byla nečinná příliš dlouho. Druhým účelem je dovolit aplikaci odeslat nová data, která byla vyprodukována. Aplikace totiž může odeslat data jen když je vyvolána od uIP a proto "poll" událost je jediná cesta jak odeslat data na jinak nečinné spojení.

Podrobnější popis API a dalších vlastností uIP je vysvětlen v dokumentu [12].

8.3.2 Podpora více aplikací, směrování podle portů

Podpora běhu několika aplikačních programů současně, není v uIP příliš vhodně řešena. Proto jsem tuto vlastnost implementoval vytvořením aplikace router (soubory *router.c, router.h*). Aplikaci bude nazývána zkráceně **uip router.**

Soubor *router.c* obsahuje funkci *router_tcp_appcall* (pro TCP) a *router_udp_appcall* (pro UDP). Tyto funkce jsou volány uIP stackem pomocí maker *UIP_APPCALL* (TCP)*, resp. UIP_UDP_APPCALL* (UDP) vždy, když nastane nějaká událost (přijatá data, potvrzená data, atd.). Úkolem routeru je předat řízení uIP aplikaci, pro kterou jsou data určena, a to v závislosti na typu protokolu (TCP, UDP) a čísle portu, na němž aplikace data očekává (viz. tab. 6 a tab. 7). Porty jsou aplikacím přiřazeny při inicializaci uIP stacku pomocí funkce *router_init()*. Na obr. 18 je znázorněn princip uip routeru se směrováním na aplikace, které budou vytvořeny a v dalších kapitolách popsány.



Obrázek 18: Směrování na jednotlivé aplikace pomocí uip routeru.

```
#define UIP_APPCALL     router_tcp_appcall

void router_tcp_appcall(void)
{
  switch(uip_conn->lport)
  {
    case HTONS(LOCAL_PORT_TCP_APP1):
      app1_tcp();
      break;
    case HTONS(LOCAL_PORT_TCP_APP2):
      app2_tcp();
      break;
    ...
    ...
  }
}
```

Tabulka 6: Směrování TCP paketů na aplikaci podle portu (soubor router.c)

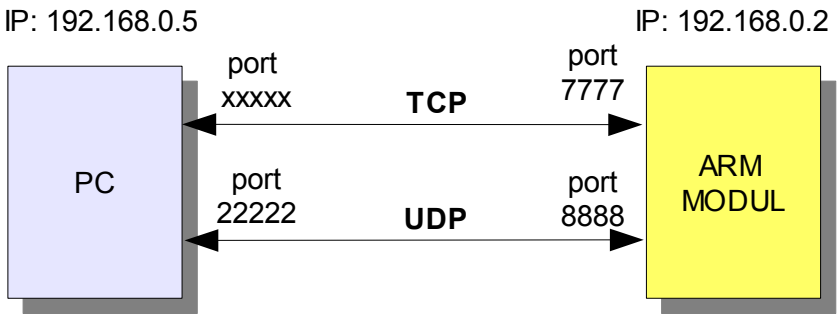
```
#define UIP_UDP_APPCALL     router_udp_appcall

void router_udp_appcall(void)
{
  switch(uip_udp_conn->lport)    //uip_udp_conn!!
  {
    case HTONS(LOCAL_PORT_UDP_APP1):
      app1_udp();
      break;
    case HTONS(LOCAL_PORT_UDP_APP2):
      app2_udp();
      break;
    ...
    ...
  }
}
```

Tabulka 7: Směrování UDP paketů na aplikaci podle portu (soubor router.c)

Vytvoření uip routeru si vyžádalo drobnou úpravu souboru *uip.h*. Ve strukturách *uip_conn* a *uip_udp_conn* byla nadefinována pro každou aplikaci proměnná, ve které je uchováván stav aplikace (informace o otevřeném spojení). V základním kódu uIP 1.0 byl definován stav jen jeden, pouze pro jednu aplikaci.

8.3.3 Vytvoření nové aplikace

Při vytváření nové aplikace je potřeba si říci, k jakému účelu bude aplikace sloužit, jaký bude používat komunikační protokol a na jakém portu bude naslouchat. Následující příklad ukazuje vytvoření aplikace používající TCP protokol na portu *7777* a UDP protokol na portu *8888*. Druhá strana UDP spojení bude mít UDP port *22222* (viz. obr. 19). V podstatě se jedná o dvě nezávislé aplikace umístěné v jednom zdrojovém souboru.



Obrázek 19: TCP a UDP porty

Jednotlivé kroky nutné pro vytvoření nové aplikace:

1) vytvoření zdrojových souborů (*.c), (*.h)

Nejprve je potřeba vytvořit zdrojové soubory, které budou obsahovat kód aplikace. Nazveme je *meas2.c* a *meas2.h* a umístíme je do stejné podsložky projektu (*./uip*) jako ostatní aplikace. Vytvoříme pro ně adresář s názvem *meas2*.

2) přidání souborů do stávajícího projektu (postup je uveden pro vývojové prostředí RIDE)
   - do souborového stromu projektu přidáme (*Add node*) soubor *meas2.c*
   - do *Option-Project-Directories-Include* přidáme cestu k hlavičkovému souboru: *.\uip\meas\meas2.h*
   - do souboru *router.c,* do funkce *router_init()* přidáme volání funkce *meas2_init()*, která slouží k nastavení naslouchajících portů. Zvolíme vhodná čísla portů (viz. aplikace již vytvořené)
   - do souboru *uip-conf.h* přidáme řádek: *#include "meas2.h"* pro připojení kódu aplikace k TCP/IP stacku.

3) definování aplikace v routeru (TCP)

   do souboru *router.c*, do funkce *router_tcp_appcall()* přidáme volání aplikace podle portu:

```
case HTONS(7777):
meas2_appcall();
break;
```

   do souboru *meas2.c*, do funkce *meas2_init()* přidáme funkci pro naslouchání na portu :

   uip_listen(HTONS(7777));

   v souboru *meas2.h* nadefinujeme strukturu pro uložení stavu aplikace

```
struct meas2_state
{
  char state;
};
```

   do souboru *uip.h*, do struktury *uip_conn* do typu *union* definujeme proměnnou podle vytvořeného typu meas2_state:

```
struct meas2_state     meas2_appstate;
```

4) definování aplikace v routeru (UDP)

do souboru *main.c* přidáme řádky:

```
struct uip_udp_conn *c2;

uip_ipaddr(ipaddr, 192,168,0,5);            //nastaveni remote IP
c2 = uip_udp_new(&ipaddr, HTONS(22222));    //remote IP, port

if(c2 != NULL) {
      uip_udp_bind(c2, HTONS(8888));  //Bind a UDP connection to a
local port
}
```

vysvětlení výše uvedeného kódu:

- alokace paměti pro nové UDP spojení, jeho přiřazení k IP adrese a portu zařízení (druhé strany), se kterým bude přes UDP komunikováno (vzdálené PC)
- vytvořenému UDP spojení přiřadíme lokální UDP port *8888*

do souboru *router.c*, do funkce *router_udp_appcall()* přidáme volání aplikace podle portu

```
case HTONS(8888):
      meas2_udpapp();
      break;
```

v souboru *meas2.h* nadefinujeme strukturu pro uložení stavu UDP aplikace

```
struct meas2_udp_state
{
  char state;
};
```

do souboru *uip.h*, do struktury *uip_udp_conn*, do typu *union* definujeme proměnnou podle vytvořeného typu meas2_udp_state:

```
struct meas2_udp_state     meas2_udp_appstate;
```

8.3.4 HTTP server s podporou CGI skriptů

Jednou z aplikací, která je obsažena ve zdrojových kódech uIP stacku je jednoduchý HTTP server s podporou CGI schopný číst HTML stránky z ROM souborového systému (paměti FLASH). Pomocí internetového prohlížeče se lze z počítače na HTTP server připojit a prostřednictvím HTML stránek z mikrokontroléru získávat data, nebo mu data odesílat. Vytvořením vlastní ovládací HTML stránky získáváme možnost ovládání mikroprocesoru přes grafické rozhraní. Aplikace podporuje jednoduchý CGI (Common Gateway Interface) skriptovací jazyk (CGI obecně definuje způsob předávání dat mezi serverem a CGI skriptem).

HTTP server se skládá z těchto modulů které jsou v adresáři *uip/webserver*:

- *httpd.c* - http server, obsahuje vstupní callback funkci aplikace
- *httpd-cgi.c* – CGI skriptovací interface, napsaný s využitím protosocketů (viz. [12])
- *httpd-fs.c* – ROM souborový systém
- *httpd-fsdata.c* – výsledek kompilace HTML stránek pomocí perl skriptu (vysvětleno dále)

Dále adresář *uip/webserver* obsahuje:

- adresář test – HTML stránky a perl skripty pro jejich převod

Soubor httpd.c je hlavním modulem HTTP serveru. Obsahuje funkci *httpd_appcall()*, ktrerá je volána uip routerem. Protože HTTP server komunikuje protokolem TCP/IP, je využívána funkce *router_tcp_appcall()*. Tuto funkci volá uIP vždy, když chce předat nějaká data serveru, nebo zkontrolovat zda server má nějaká data k odeslání. HTTP server naslouchá na portu 80, což je standardní číslo portu pro webové servery. Je možné toto číslo změnit na jinou hodnotu, ale pak je potřeba při přístupu na tento server zadávat v internetovém prohlížeči kromě IP adresy explicitně číslo portu (např: 10.1.1.1:12345).

Nebudou dále vysvětlovány implementační detaily jednotlivých součástí HTTP serveru, to je obsahem podrobně popsané dokumentace k uIP. Budou ale vysvětleny změny provedené na převzatém kódu.

Dynamické a statické HTML stránky

HTML stránky které budou na serveru spuštěny, mohou být statické nebo dynamické (dynamicky generované).

Statická HTML stránka je po vyžádání odesílána přímo žadateli (internetovému prohlížeči) v podobě, v jaké je uložena v souborovém systému.

Do dynamické HTML stránky, předtím než je odeslána, může být vložen další kód (dynamicky vytvářený). Vkládaným kódem mohou být hodnoty proměnných jazyka C. Tímto způsobem je možné vyčítat z mikrokontroléru různé stavy, výsledky operací, naměřené hodnoty, atd. Dynamickou HTML stránku představuje soubor s příponou *.shtml. Na místě, kam má být v souboru *.shtml vložen dynamický obsah, musí být vložena sekvence znaků "%!", za kterou následuje mezera a řetězec reprezentující funkci, která obsah generuje. Tato funkce (CGI funkce) bude popsána v dalším textu.

Princip funkce HTTP serveru

Když server přijme od webového prohlížeče požadavek na zaslání určité stránky, je vyvolána funkce *handle_connection()*. Požadavek může navíc obsahovat tzv. POST data. To jsou data zadaná uživatelem do formuláře v HTML stránce, nebo například jméno tlačítka, na které uživatel klikl. HTTP server POST data rozparsuje (rozdělí na části), vyhodnotí je a podle výsledku může provést určitou akci.

Poznámka: Implementaci čtení POST dat z HTML stránky jsem oproti původní verzi rozšířil. Data jsou parsována v souboru httpd.c, ve funkci handle_input() a uložena do struktury "postdata".

Vkládání dynamického obsahu do HTML souboru

K tomuto účelu slouží velmi jednoduché CGI rozhraní implementované v souboru *httpd-cgi.c*. Zde jsou pomocí makra *HTTPD_CGI_CALL()* definovány CGI funkce, které vytvářejí dynamický HTML obsah a jim příslušící řetězec, podle kterého bude funkce vyvolávána. O tomto řetězci již byla řeč v odstavci popisujícím dynamické HTML stránky.

Generování souboru HTML stránek pro MCU

HTML stránky se nacházejí v adresáři *./webserver/test/httpd-fs.*

Aby mohly být stránky se všemi komponentami uloženy do FLASH paměti mikroprocesoru, musí být předtím převedeny do formátu, kterému "rozumí" souborový systém. K tomu slouží dva skripty v jazyce Perl (přípona .pl) v adresáři *./webserver/test:*

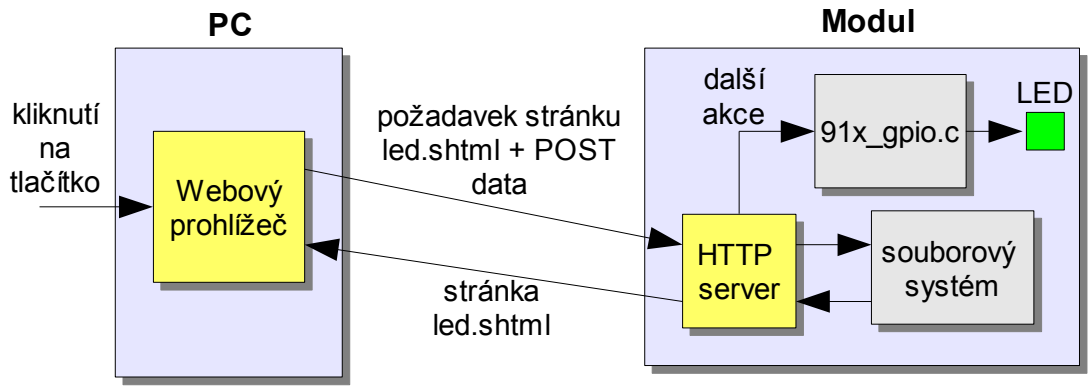
- *makefsdata.pl*
- *makestrings.pl* – ze souboru *http-strings* (soubor s nadefinovanými páry proměnná-řetězec) generuje soubory *http-strings.h* a *http-strings.c*

Vstupem skriptu *makefsdata.pl* jsou html soubory, příp. další soubory z adresáře httpd-fs. Skript převádí jednotlivé soubory z tohoto adresáře do podoby pole bytů a ukládá je do souboru httpd-fsdata.c.

Pro spouštění skriptů je zapotřebí překladač skriptovacího jazyka Perl. V OS Windows můžeme použít například "Active Perl" od společnosti "Active State", který je po registraci ke stažení. Po jeho nainstalování jsou soubory s příponou .pl registrovány k tomuto programu a lze je spouštět dvojklikem. Po spuštění obou skriptů překopírujeme tři vygenerované soubory (*http-strings.h, http-strings.c, httpd-fsdata.c*) do adresáře webserver.

Komunikaci mezi PC a modulem (viz. obr. 20) lze shrnout do následujících kroků:

- zadání IP adresy modulu a cesty souboru do internetového prohlížeče na PC (např.: *111.222.333.444/soubor.html*)
- odeslání požadavku (pokud nebyla zadána cestu k souboru, odešle se požadavek na soubor *index.html*)
- modul přijme požadavek, spouští funkci *handle_connection()*
- obsahuje-li požadavek POST data, uloží se do příslušných proměnných struktury *post*
- funkce *handle_script()* hledá ve vyžádaném souboru sekvenci znaků, uvozující CGI skript; pokud ji najde spustí se příslušná CGI funkce
- modul odešle soubor a zobrazí jeho obsah v prohlížeči



Obrázek 20: Princip čtení HTML stránky z mikrokontroléru.

Konkrétní aplikace ovládání LED diody z HTML stránky

Obr. 20 se váže také k následujícímu příkladu konkrétního použití HTTP serveru, kdy je po kliknutí na tlačítko v prohlížeči a přijetí požadavku od PC, odeslána stránka *led.shtml* a rozsvícena LED dioda GPIO4.3.

Hlavní stránka *index.html* bude mimo jiné obsahovat formulář s polem pro zadání textového řetězce a jedno tlačítko (viz. tab. 8). Po kliknutí na tlačítko se odešle požadavek na odeslání souboru *led.shtml*, obsahujícího CGI skript.

```
<FORM ACTION="led.shtml" METHOD="GET">
  <INPUT TYPE="text" NAME="led_app">
  <INPUT TYPE="SUBMIT" NAME="led3" VALUE="3">
</FORM>
```

Tabulka 8: Část kódu stránky index.html, vytvářející formulář.

HTML stránky *index.html* a *led.shtml* je potřeba přeložit Perl skripty a vygenerované soubory překopírovat do adresáře serveru.

V souboru *httpd-cgi.c* definujeme novou CGI funkci pomocí makra (vysvětlení parametrů viz. tab.9):

```
HTTPD_CGI_CALL(led, "led-retezec", led_ptr);
```

led	jméno CGI funkce
led-retezec	řetězec v shtml souboru (za sekvencí “%!“)
led_fce	pointer na CGI funkci

Tabulka 9: Vysvětlení parametrů makra HTTPD_CGI_CALL

Implementace CGI funkce *generate_led_fce*:

```
static unsigned short
generate_led_fce(void *arg)
{
    if (*(post->button_value) == '3')
       GPIO_WriteBit(GPIO4, GPIO_Pin_3, Bit_SET);
       //nastavi pin 3 do log 1

    //zapis dynamicky generovanych dat do bufferu
    return snprintf((char *)uip_appdata, UIP_APPDATA_SIZE,
          "LED%c zapnuta<BR>\r\n", *(post->button_value));
}

static
PT_THREAD(led_stats(struct httpd_state *s, char *ptr))
{
  PSOCK_BEGIN(&s->sout);
  PSOCK_GENERATOR_SEND(&s->sout, generate_led_stats, s);
  PSOCK_END(&s->sout);
}
```

Celý projekt přeložíme a výsledný binární soubor nahrajeme do MCU. Připojíme se k serveru z internetového prohlížeče. Načte se stránka s vytvořeným formulářem. Do textového pole zadáme například *„zadanytext“* (obsah není podstatný) a odešleme tlačítkem.

Data TCP/IP paketu, která jsou odeslána serveru po kliknutí na tlačítko:

```
GET /led.shtml?led_app=zadanytext&led3=3 HTTP...
```

Server odešle vyžádaný soubor *led.shtml*, doplněný o dynamický HTML kód *„LED3 zapnuta
“ a* prohlížeč soubor zobrazí.

HTML stránka je jeden ze způsobů jak přistupovat k mikroprocesoru přes Ethernet, konkrétně přes TCP/IP protokol. Je to uživatelsky pohodlné, ovládací prvky i zobrazená data jsou v grafické podobě.

8.3.5 Aplikace Terminal

Aplikace *„Terminal“* byla vytvořena za účelem ovládat modul z textového režimu, jednoduchými příkazy. *Terminal* podobně jako HTTP server naslouchá na TCP/IP portu. Bylo zvoleno číslo portu *50000*. K aplikaci se lze připojit z PC klientem telnet. Testovanými klienty byl základní telnet ve Windows XP, dále program PUTTY (Windows) a telnet klient v OS Linux. Jejich chování při odesílání zpráv se mírně liší:

- putty, linux telnet – odesílá paket po napsání zprávy a stisku klávesy *ENTER*
- windows telnet– odesílá paket po napsání každého znaku (nepříliš vhodné)

Po navázání TCP/IP spojení klienta s aplikací bude aplikace očekávat příchozí data od uživatele z PC. Aplikace reaguje jen na určité příkazy, které byly nadefinovány v souboru *teminal.c*. Jeden příkaz se může skládat z více podpříkazů oddělených mezerou. Seznam příkazů i s nápovědou je vypsán po zadání příkazu „help“ v telnet klientu. Pomocí těchto textových příkazů lze provádět následující funkce:

- čtení a zápis logických stavů brány GPIO4, ovládání LED diod
- přepínání módu brány GPIO4 mezi analogovým (ADC mód) a digitálním (I/O mód)
- zápis na sériový port ve třech formách - ASCII znak, řetězec znaků, binární číslo

Další příkazy a odpovídající akce, které má MCU provádět je možné do kódu aplikace *Terminal* jednoduše přidávat.

Poznámka: ADC a I/O mód jsou mnou pojmenované módy brány GPIO4, které je možné přepínat proměnnou ADC_MODE v souboru main.c.

8.3.6 Obousměrný převodník rozhraní TCP/IP - RS232

Aplikace *meas2* umožňuje obousměrný převod ASCII zpráv mezi Ethernetem (TCP/IP) a RS-232. Zpráva zadaná do TCP/IP terminálu na PC je přeposílána na RS-232 port modulu (UART0). Analogicky opačným směrem. Aplikace *meas2* má nastaven TCP port na 7777.

TCP/IP terminálem může být telnet, RS-232 terminálem např. program Hercules (viz. kap. 9.6.1). Konec odesílaného řetězce musí být zakončen stiskem klávesy *ENTER*

(v obou terminálech). Systémy Windows odesílají *ENTER* jako dvojici znaků <CR>,<LF> (0x0A, 0x0D), unixové systémy jako <LF>. Aplikace *meas2* podporuje oba způsoby zakončení. Znaky <CR>,<LF> jsou aplikací *meas2* z přijaté zprávy odstraňovány a protějšímu terminálu je zpráva odesílána bez nich. To je z toho důvodu, aby zařízení připojené na sériový port RS-232 a ovládané pomocí přesně definovaných zpráv, nedostávalo navíc nežádoucí znaky.

Ve směru "sériový port - telnet" jsou data čtena přímo v hlavní smyčce programu. Po přijetí celé zprávy je vyvolána aplikace *meas2* funkcí *uip_poll()*. Ta zprávu odešle telnet klientu. Ve směru "telnet – sériový port" je aplikace vyvolána okamžitě po příjmu zprávy telnet klientem. Odesílání na sériový port se děje v aplikaci.

Vytvořená funkce převodníku Ethernet/RS-232 umožňuje nasadit modul tam, kde je potřeba nahradit sériový port modernějším Ethernetem, nebo prodloužit sériovou linku.

Poznámka: Ve směru "sériový port – telnet" aplikace spolehlivě funguje i při rychlosti UARTu 115 kb/s v případě, že meas2 je jedinou aplikací, která uIP stack zatěžuje. Při větší zátěži uIP stacku (a tím delším oběhu hlavní smyčky), musí být rychlost UART snížena, nebo čtení dat řešeno jiným způsobem.

8.4 Přenos větších objemů dat

Dosud byly popisovány aplikace, které nevyžadovaly vysoké rychlosti přenosu. Jednalo se o textové zprávy, případně malé soubory. V této podkapitole budou rozebrány možnosti přenosu většího množství dat uIP stackem. Pro otestování takového přenosu je ve FLASH paměti uložen obrázek, v rozlišení 720×576 pixelů. Pro přenos obrázku do PC byla vytvořena aplikace *meas* (v adresáři *./uip/meas)*. Pro přenos byly použity protokoly TCP i UDP pro porovnání jich vlastností.

Aplikace naslouchá na TCP portu *1234* a na UDP portu *12344*. Pro připojení na oba porty byly vytvořeny programy na PC (příloha *DVD\programy*), Blackfin_TCP a Blackfin_UDP, které data čtou a zobrazují (programy se liší v podstatě jen typem protokolu). Základem programů je zobrazovač vytvořený na naší katedře Milošem Fenykem, programovaný v prostředí Visual Studio v jazyce C# který čte obrazová data ze sériového portu, připojuje k nim BMP hlavičku a zobrazuje je jako BMP obrázek. Přidal jsem podporu TCP/IP a UDP/IP protokolů a také definoval potvrzovací zprávy, aby se zamezilo degradaci rychlosti v důsledku zpožděného potvrzování paketů síťovou kartou počítače (viz. kap. 8.1.4).

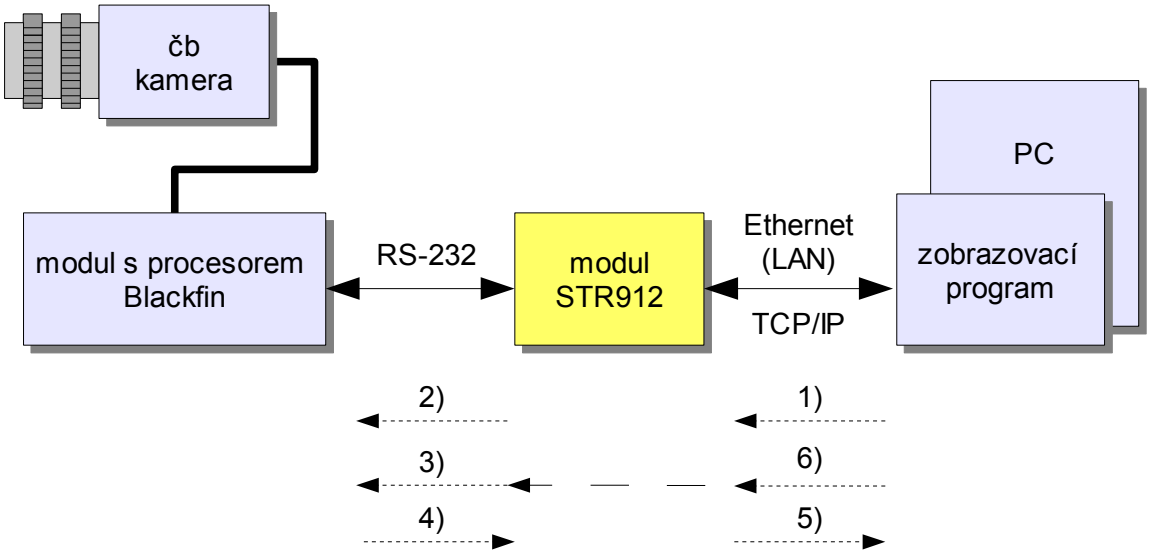
Hodnoty pixelů obrázku jsou uloženy v proměnné *obr[]*. Obrázek je odesílán po řádcích, každý řádek (720 bytů) je odeslán v jednom paketu. Každý paket je potvrzen zprávou, a po přenesení celého obrázku je zaslán požadavek na nový obrázek. UDP protokolem byla dosažena rychlost přenosu **1,3 MB/s**. Připojení k aplikaci z PC je popsáno v kap. 9.5.4.

Protokol UDP je ve své podstatě nespojovaný protokol, který nezaručuje, že data dojdou k příjemci ve správném pořadí a zda vůbec dojdou. Má menší velikost paketu než TCP, měl by být proto rychlejší. Na uIP stacku jsem ale nenaměřil příliš velký rozdíl rychlostí (i při vypnutém výpočtu kontrolního součtu UDP paketu).

8.5 Čtení obrazových dat z procesoru BlackFin

Pro demonstraci čtení dat z externího zařízení byl sestaven experiment znázorněný na obr. 21. Systém s kamerou a signálovým procesorem zpracovává obraz a odesílá jej na sériovou linku RS-232. Úlohou navrženého modulu je přeposílání dat ze sériové linky na Ethernet pro jejich zobrazování na PC.

Černobílá kamera snímá plošný obraz, který je dekódován na vývojové desce EZ-KIT Lite BF533 se signálovým procesorem Blackfin. Blackfin je naprogramován tak, že na vyžádání odesílá postupně jednotlivé řádky obrazu. Obrázek je v rozlišení 180x144 pixelů. Modul se spuštěnou aplikací *Kamera* (soubor *kamera.c* v adresáři *uip*), zprostředkovává data mezi Blackfin modulem a zobrazovačem na PC. Zobrazovacím programem na PC je již zmíněný program Blackfin_TCP, mírně modifikovaný aby komunikoval s aplikací *Kamera*.



Obrázek 21: Experiment čtení dat ze signálového procesoru Blackfin.

Proces čtení dat

Proces čtení a přeposílání dat je vysvětlen následujícím seznamem operací, které se provádějí za sebou. Čísla operací korespondují s čísly u šipek v obr. 21. Umístění a směr šipky symbolizuje, v jaké části blokového schematu se operace odehrává.

1) požadavek od zobrazovače – TCP/IP paket s ASCII znakem 'f' (new frame) – zašli nový obrázek
2) požadavek na Blackfin na započetí zasílání nového obrázku (hexadecimálně 0x38)
3) požadavek na Blackfin na zaslání nového řádku (ASCII znak 'r' – new Row)
4) Blackfin odesílá řádek o 180 bytech
5) odeslání TCP/IP paketu zobrazovacímu programu
6) potvrzení přijetí dat, paket s ASCII znakem 'a' (potvrzovací, pro vyvolání aplikace) modul STR912 pokračuje bodem 3)
   po vyčtení všech 144 řádků zobrazovač obrázek vykreslí a pokračuje bodem 1)

Apliace kamera.c

Čtení řádku rozhraním UART v mikrokontroléru probíhá přímo v aplikaci *Kamera*. Komunikační rychlost je 115 kb/s, takže vyčtení celého řádku zabere čas přibližně 15ms. Čas, který stráví program v aplikační funkci, by měl být co nejkratší (viz. kap. 8.1.1), aby nedošlo ke ztrátě dalšího příchozího paketu (viz. obr. 16), který může být určen i pro jinou, souběžně běžící aplikaci a který by musel být znovu vyžádán.

8.6 Porovnání výkonu aplikací

uIP stack je schopen velmi rychle reagovat na příchozí data (pakety), ale nevyniká velkým výkonem při odesílání dat. Jak již bylo uvedeno, maximální rychlost odesílání dat z uIP je 7 kB/s.

Pro odesílání jednoduchých HTML stránek je tato rychlost dostatečná, reálná rychlost je dokonce vyšší než 7 kB/s, protože některé pakety putují i opačným směrem, což komunikaci urychluje. Pomocí programu Wireshark (viz. kap. 9.6.2) byla naměřena rychlost odesílání HTML obsahu okolo 40 kB/s. Stejně tak pro textové zprávy odesílané programem telnet je základní rychlost dostatečná..

V aplikacích, kde je potřeba přenášet data vyšší rychlostí, bylo použito metody potvrzovacích zpráv, která byla popsána v kap. 8.4. S tímto nastavením je dosaženo rychlostí přes 1 MB/s.

8.7 Konfigurační soubory uIP

Před kompilací uIP stacku je možné nastavit jeho paramety pomocí konfiguračních souborů. K tomu je určen v první řadě soubor *uip-conf.h*. Dále pak *uipopt.h*, ve kterém jsou definovány podrobnější parametry.

uip-conf.h

V tomto souboru jsou pomocí datových typů, které používá konkrétní mikroprocesor, definovány 8-mi a 16-ti bitové proměnné, které používá kód uIP stacku. Jinak budou následující definice typů vypadat na 8-bitovém mikrokontroléru a jinak na 32-bitovém.

```
typedef unsigned char u8_t;
typedef unsigned short u16_t;
```

Dalším, na typu MCU závislým parametrem je *UIP_CONF_BYTE_ORDER*, kterým se nastavuje endianita mikrokontroléru (u architektury ARM little-endian). Dalšími konstantami s prefixem *UIP_CONF_* jsou nastavovány parametry jako velikost uIP bufferu, počet spojení, atd. Nakonec jsou v tomto souboru includovány hlavičkové soubory vytvořených aplikací, které mají v uIP běžet.

uipopt.h

V tomto souboru je podrobnější nastavení konfigurace. Důležité je nastavení MAC adresy pro ARP modul stacku. To je možné dvěma způsoby:

1) fixní MAC adresa - v souboru *uipopt.h*: *UIP_FIXEDETHADDR 1*
   MAC bude nastavena podle konstant *UIP_ETHADDR0 - UIP_ETHADDR5*
2) nefixní MAC adresa - v souboru *uipopt.h*: *UIP_FIXEDETHADDR 0*
   požadovaná MAC se nastaví v souboru *main.c*, do pole *mymac[6*]

V obou případech je nutno stejnou MAC adresu jako pro ARP modul nastavit také v souboru ovladače jednotky ENET, *91x_enet.h.*

8.8 Profily nastavení sítě

Pokud bude modul připojován střídavě do dvou sítí, bylo by vhodné obě nastavení sítě (IP adresy, síťové masky, atd.) definovat v kódu a před kompilací zvolit, které nastavení sítě se použije, aby nemusel být kód programu pro každou síť vždy přepisován.

Oba profily jsou definovány v souboru *main.c*. Na začátku souboru byla vytvořena konstantu preprocesoru *TEST_IP* pro volbu profilu. Ta může mít dva stavy.

#define TEST_IP 1

- pokud bude modul připojen přímo k vývojovému PC kříženým kabelem (ladění kódu, testování aplikací)

#define TEST_IP 0

- pokud je firmware modulu hotov a modul připojen do konkrétní sítě LAN nebo do Internetu

	#define TEST_IP 1	**#define TEST_IP 0**
uip_sethostaddr() - IP	192.168.0.2	147.32.116.187
uip_setdraddr() - DR	192.168.0.254 - nepodstatné	147.32.116.1
uip_setnetmask() - NM	255.255.255.0	255.255.255.0
uip_udp_new() - RIP	192.168.0.5	147.32.116.128

Tabulka 10: Dva profily nastavení sítě

- **IP** = IP adresa modulu
- **DR** (default router) = IP adresa routeru, na kterou jsou pakety směrovány v případě, že cílové zařízení není ve stejné síti; pro přímé spojení s PC je adresa routeru nepodstatná
- **NM** (net mask) = síťová maska
- **RIP** (remote IP) = IP adresa vzdáleného zařízení (PC) – pro UDP

8.9 Portování uIP na jinou platformu

uIP lze portovat na kteroukoli jinou platformu (jiný mikrokontrolér), 8-bitovou, 16-bitovou i 32-bitovou. MCU může mít ethernetový řadič integrován na čipu nebo externě připojen. Důležité je mít pro konkrétní ethernetový řadič napsán ovladač, a vyřešenu komunikaci mezi řadičem a mikroprocesorem. V kódu uIP stacku není potřeba provádět velké změny.

Dalším aspektem portování je, že pro různé typy mikrokontrolérů se používají různá vývojová prostředí a překladače, která mohou mít svá specifika. Jedním z nich je alignment (zarovnávání) struktur v paměti. Kód uIP stacku musí být překládán s alignmentem 1 byte, (viz. kap. 8.2.1), to je nutné zkontrolovat v dokumentaci k používanému překladači.

8.9.1 Změny v kódu uIP

Následující zdrojové soubory je při portování potřeba pozměnit, nebo přinejmenším zkontrolovat:

uipopt.h

Nastavení MAC adresy: UIP_ETHADDR0 – UIP_ETHADDR5. Musí korespondovat s MAC adresou v Ethernet řadiči (v tomto případě je uložena v souboru ovladače k němu).

uip-conf.h

- *typedef unsigned char u8_t; - unsigned char* změnit na 8-bitový typ procesoru (na většině mikroprocesorů beze změny)
- *typedef unsigned short u16_t; - unsigned short* změnit na 16-bitový typ procesoru (u 8-bitových mikroprocesorů to bude *unsigned int*)
- *#define UIP_CONF_BYTE_ORDER LITTLE_ENDIAN* – změnit podle endianity procesoru (little-endian, middle-endian, big-endian)

clock-arch.h, clock-arch.c, clock.h, timer.c

- definovat typ pro *clock_time_t*, a přiřadit funkci, která měří čas
- upravit implementaci funkce timer_expired() podle toho, jakým způsobem bude čas měřen (čas může být měřen např. RTC jednotkou nebo časovači mikroprocesoru)

uip_arch.h

- není nutné ho měnit, pouze pokud chceme použít vlastní implementaci 32-bitové aritmetiky

8.9.2 Změny v ostatních částech kódu

main.c

Includován soubor *91x_enet.h*, což je hlavičkový soubor k ovladači síťového řadiče. Místo něj includovat soubor ovladače pro použitý řadič. Totéž platí pro knihovnu ovladačů periferií *91x_lib.h*, zejména pro ty periferie, které jsou TCP/IP stackem využívány (RTC jednotka).

Změny, nutné provést v hlavní smyčce funkce:

- Volání funkce *uip_len = ENET_HandleRxPkt(uip_buf)* přejmenovat na funkci ovladače síťového rozhraní, která po zavolání zkopíruje paket z bufferu síť. rozhraní do bufferu *uip_buf* a velikost paketu uloží do *uip_len.*
- Volání funkce *ENET_TxPkt(&uip_buf[0], uip_len)* přejmenovat na funkci, která po zavolání zkopíruje paket z *uip_buf* do bufferu síť. rozhraní.

9 Návod pro práci s modulem

Tato kapitola slouží jako uživatelský manuál pro práci s vytvořeným modulem. Budou zde popsány jednotlivé kroky a nastavení, které je potřeba provést, aby mohl být modul propojen s PC. Bude uveden návod, jak mikrokontrolér naprogramovat vytvořeným programem a připojit se ke všem vytvořeným aplikacím TCP/IP stacku.

Všechny potřebné programy na PC byly provozovány OS Windows XP. V OS Linux nebyl nalezen způsob, jak modul naprogramovat, protože neexistuje linuxový ovladač pro debugger ULINK. To je jediný problém, ostatní věci jako kompilace kódu GNU kompilátorem, použití telnet terminálu a sériové linky jsou v Linuxu možné. Následující postupy budou tedy popsány pro Microsoft Windows XP.

9.1 Instalace programů RIDE a uVision, nahrání firmware

Hlavním programem, který je nutné nainstalovat je *RKit*. Balík obsahující grafické vývojové prostředí RIDE a GNUARM kompilátor. Dále je potřeba nainstalovat vývojové prostředí Keil uVision3, které bude využito k nahrávání firmwaru do MCU a které obsahuje ovladač pro ULINK.

Finální projekt se zdrojovými kódy pro mikroprocesor se nachází v DVD příloze v adresáři *STR912_modul\code\RIDE_GNUARM\eth_rais_3w*. Projekt je vytvořen ve vývojovém prostředí RIDE. Spouštěcím souborem je soubor **.prj*. Kód lze přeložit a nahrát následujícím postupem:

- *Project → Open → eth_rais_3w\adc.prj*
- projekt obsahuje všechny aplikace, které byly vytvořeny, včetně všech potřebných knihoven pro periferie a Ethernet
- po zkompilování kódu příkazem *Build* nebo *Make* se vytvoří soubor **.hex* v adresáři projektu
- otevřeme adresář *HEX_flash* který je v adresáři projektu a spustíme soubor *hex_flash.uv2*; otevře se projekt programu *uVision*, který je nakonfigurován tak, aby nahrál vygenerovaný **.hex* soubor do modulu. Postup nahrávání firmwaru byl popsán v kap. 7.4.4.

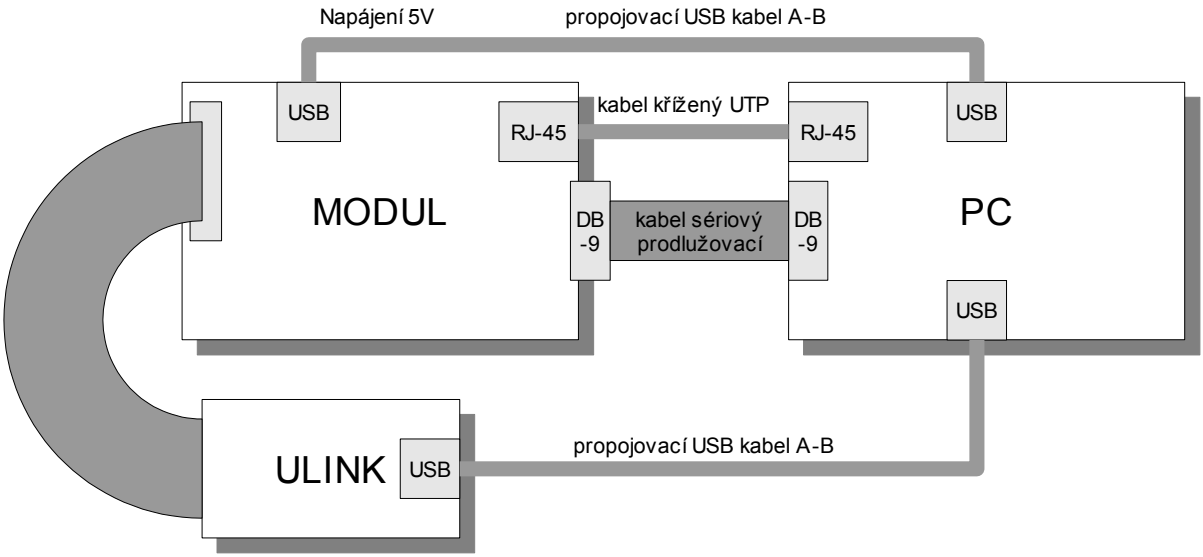
9.2 Propojení modulu s PC

Následující návod popisuje přímé propojení modulu s PC. Je připojeno napájení, komunikační rozhraní i programátor. Tato sestava je určena k propojení na krátkou vzdálenost, pro nahrávání kódu a testování.

Pro napájení modulu zvolíme jednodušší způsob, USB port PC, pokud je tedy dostatečně dimenzován (viz. kap. 6.3.3). K tomu je potřeba USB kabel A-B. Na modulu musí být propojkou správně nastavený zdroj napájení USB (viz. obr. 41).

Dále připojíme USB debugger ULINK. Konektorem JTAG k modulu, USB kabelem k PC. Kabel je také typu A-B. Sériový port propojíme s PC „prodlužovacím“ RS-232 kabelem.

Poslední věcí je Ethernet kabel. Modul je schopen nejvyšší komunikační rychlosti 100 Mbit/s (Ethernet 100Base-Tx). Většina síťových karet v počítačích podporuje stejnou rychlost 100 Mbit/s. Použijeme klasický UTP kabel (kroucená dvojlinka), s konektory RJ-45, který se používá v počítačových sítích. Důležité je, aby kabel byl **křížený** (nekřížený použijeme k propojení se switchem, hubem, atd). Zapojení je zřejmé z obr. 22.

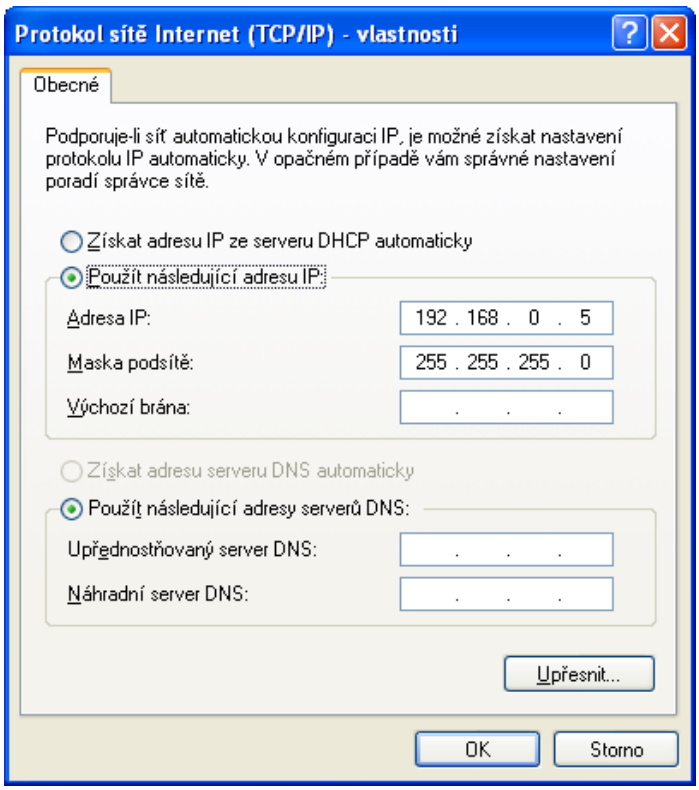


Obrázek 22: Propojení modulu s PC.

9.3 *Nastavení sítě v PC*

V modulu je staticky nastavena IP adresa **192.168.0.2** a maska podsítě **255.255.255.0**. PC nastavíme na stejnou podsíť následujícím způsobem:

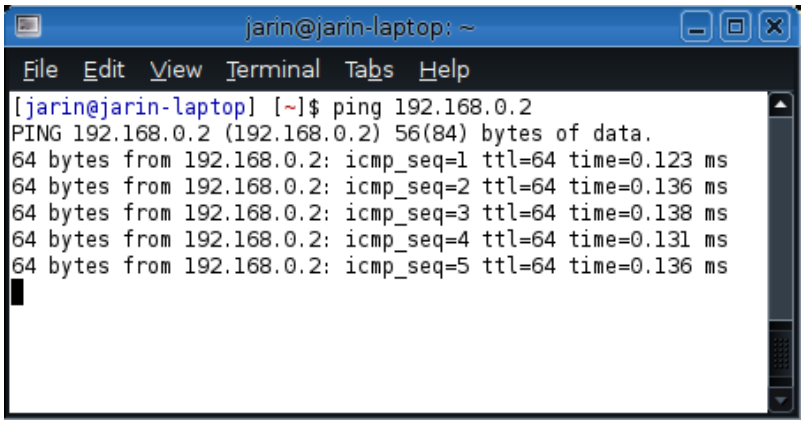
1. Otevřeme *Síťová připojení*
2. Vybereme síťovou kartu LAN – pravé tlačítko – *Vlastnosti*
3. Protokol sítě Internet (TCP/IP) – *Vlastnosti*
4. V otevřeném okně (viz. obr. 23) vyplníme IP adresu. Její první tři oktety se musí shodovat s IP adresou modulu. Poslední oktet musí být v rozsahu 0-255. Vyplníme **192.168.0.5**, protože právě taková je registrována v modulu jako vzdálená IP. Maska podsítě se musí shodovat s maskou nakonfigurovanou v modulu.



Obrázek 23: Karta nastavení sítě v PC

9.4 Ověření běhu programu v mikrokontroléru

Po připojení napájení se program začne vykonávat, což signalizují oranžové LED diody blikající v rytmu časovače uIP stacku. Funkčnost sítě lze zjistit otevřením terminálu na PC a odesláním požadavku ping k modulu.



Obrázek 24: Test dostupnosti cílového zařízení příkazem "ping".

9.5 Připojení k aplikacím TCP/IP stacku

9.5.1 Aplikace „Terminal“

K aplikaci *Terminal* se připojíme klientem telnet, ve Windows otevřením příkazové řádky (*Start* → *Spustit* → *cmd)* a napsáním příkazu telnet **192.168.0.2 50000**. Seznam příkazů a odpovídajících akcí které je možné provádět, je uveden v kap. 8.3.5. Aplikaci můžeme vyzkoušet nastavením LED diody na pinu GPIO4.5: **gpio4 set 5**.

Obrázek 25: Připojení na aplikaci "Terminal" klientem telnet v MS Windows

9.5.2 HTTP server s podporou CGI

Do internetového prohlížeče zadáme IP adresu modulu. Ze souborového systému v MCU je vyčten obsah, který se zobrazí jako dvě stránky v rámci pod sebou (viz. obr. 29). V horní HTML stránce jsou ovládací prvky, ve spodní stránce se zobrazují výsledky, které se automaticky aktualizují každé 4 sekundy. Horní stránka umožňuje ovládání těchto funkcí:

- nastavování LED diod a čtení jejich stavu
- čtení AD převodníku, zobrazení reálného času z RTC jednotky
- řízení polohovacího mechanismu

9.5.2.1 Nastavování a čtení stavu LED diod brány GPIO4

Tato funkce je aktivní pouze pokud je brána GPIO4 v módu I/O (nastavení do I/O módu viz. kap. 8.3.5). Po kliknutí na jedno z osmi tlačítek se na modulu rozsvítí daná LED, připojená na port 4 a aktuální stav brány se zobrazí ve spodní stránce. Tlačítko „aktuální stav“ přečte a zobrazí stav brány tamtéž.

9.5.2.2 Zobrazení stavu RTC a AD převodníku

Po kliknutí na odkaz *“Zobraz hodnoty ADC a RTC“* se ve spodní stránce začnou zobrazovat hodnoty:

- reálného času běžícího v RTC jednotce mikroprocesoru, který je udržován baterií i po odpojení modulu od napájení
- naměřených hodnot napětí na kanálu 0 AD převodníku, ale pouze je-li brána GPIO4 v módu ADC

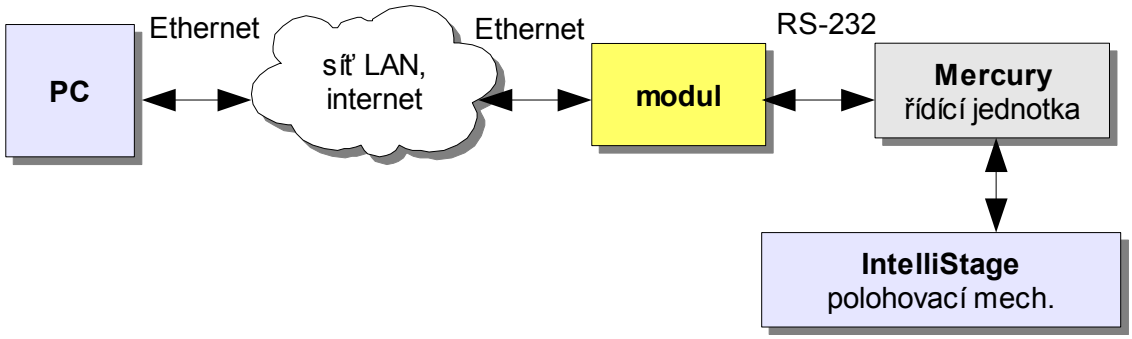
9.5.2.3 Řízení polohovacího mechanismu pro kameru

Vzdálené ovládání experimentu, kterým je přesné měření polohy kamerou, bylo zmíněno v kap. 1. Kamera měří polohu objektu, který je do určité polohy nastaven precizním polohovacím mechanismem. Celá úloha bude umístěna ve sklepních prostorách, proto je potřeba motor řídit vzdáleně. S tím bylo počítáno ve vytvořené HTML stránce, kde byla vytvořena tlačítka pro ovládání polohovadla (viz. obr. 29).

Řídící jednotka má název *Mercury C-862* (viz. obr. 26), precizní polohovací mechanismus nese označení *IntelliStages M-511* (viz. obr. 27). Blokové schéma propojení s PC je na obr. 28. Řídící jednotka zpracovává příkazy ze sériové linky a generuje PWM signál pro motor polohovadla a také čte informace o stavu motoru.

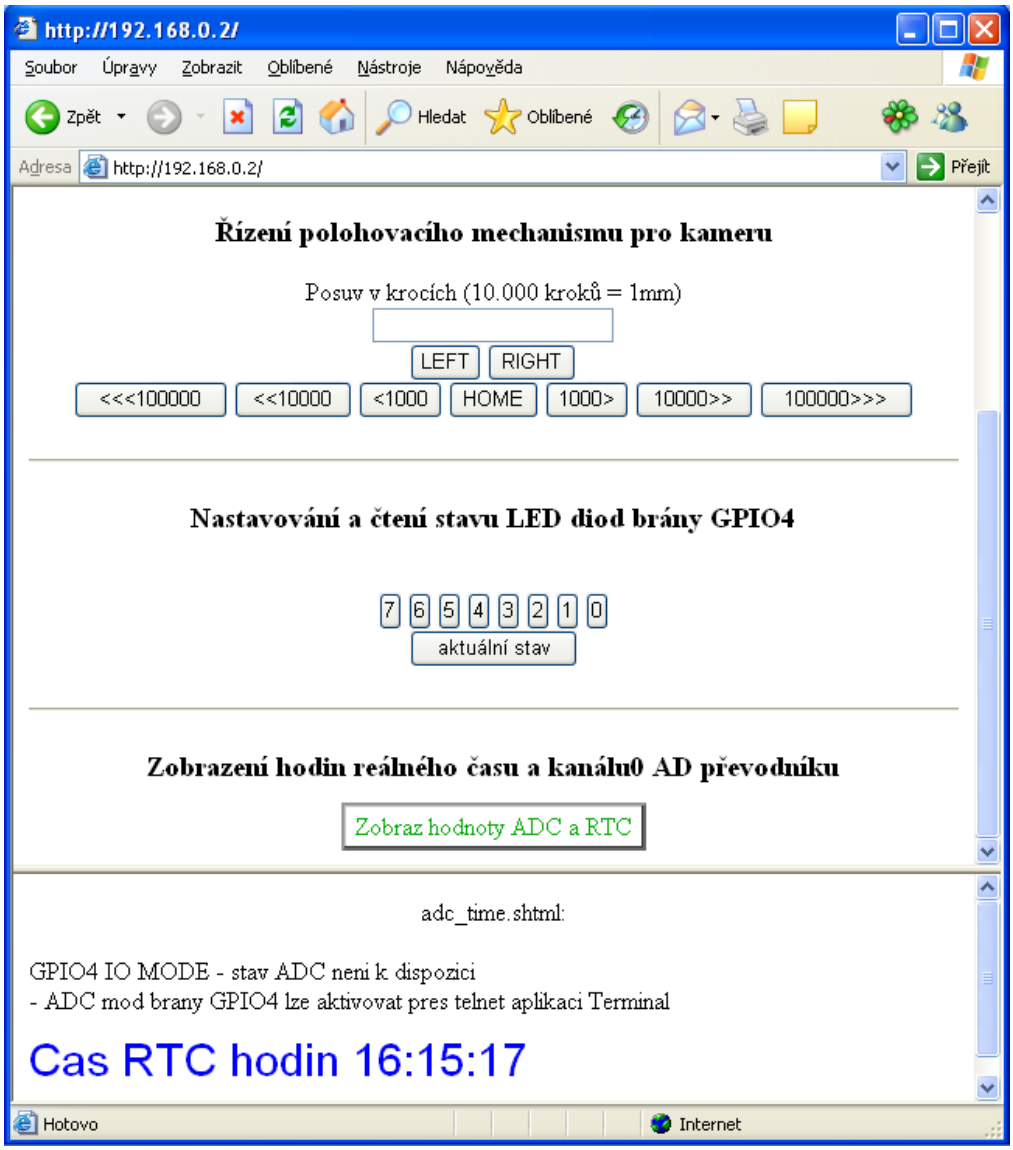
Obrázek 26: Mercury C-862.

Obrázek 27: IntelliStages M-511.



Obrázek 28: Blokové schéma ovládání polohovacího mechanismu z PC přes Ethernet.

HTML stránka obsahuje tlačítka pro ovládání polohovadla. Šestice tlačítek v řadě slouží k posuvu pojezdu o pevně daný počet kroků. Tlačítko *HOME* vrací pojezd do výchozí polohy (střed). Tlačítky *LEFT* a *RIGHT* zajistíme posuv o počet kroků zadaný do textového pole. Po stisku některého z tlačítek odesílá prohlížeč data, která modul přijme a generuje patřičná data na sériovou linku. Ty jsou přijaty řídící jednotkou, která řídí motor. Tlačítky je možné řídit posun o určitý počet kroků oběma směry a umístění polohovacího mechanismu do základní polohy.



Pro ovládání pojezdu můžeme využít také program *Terminal*, který dokáže přesměrovat zprávy z telnet klienta na sériovou linku. Řetězec "RIDICIZPRAVA" odešleme příkazem **send uart0 RIDICIZPRAVA.**

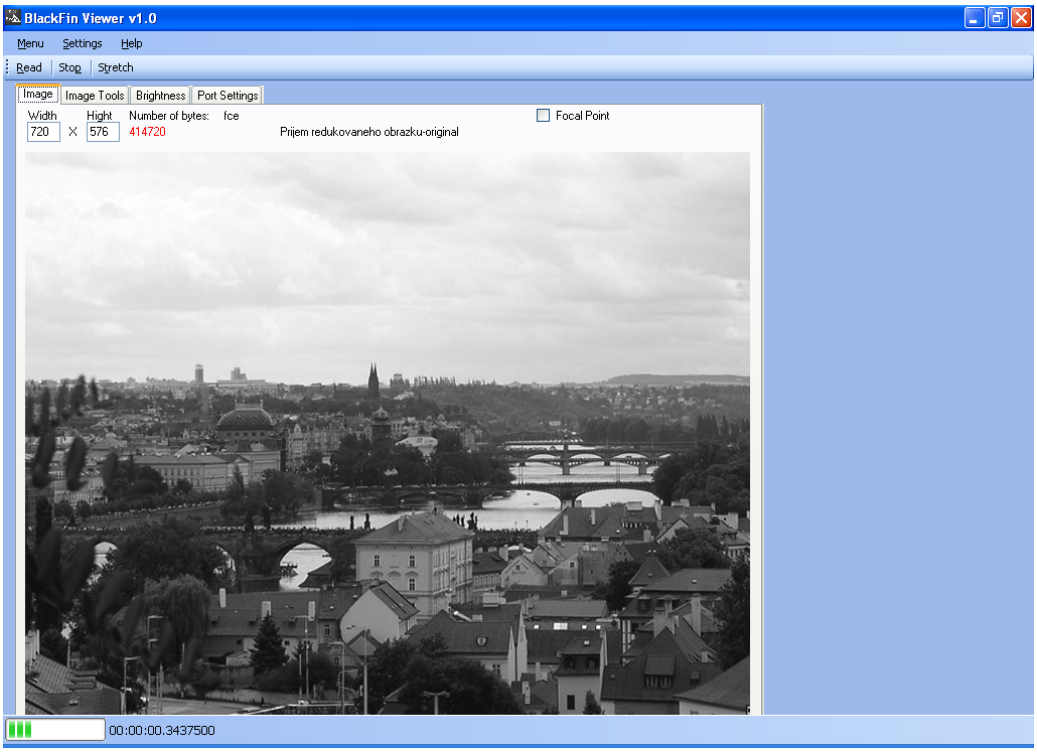
9.5.3 Telnet na standardním portu

Aplikace *Telnet (soubory telnetd.c, shelld.c)*, je v uIP stacku původní. Telnet naslouchá na standardním **TCP portu 23**. K aplikaci se připojíme z příkazové řádky, zadáním příkazu **telnet 192.168.0.2** (viz. obr. 30). Podporováno je jen několik příkazů, zajímavý je výpis statistiky přijatých/odeslaných/ztracených paketů.

9.5.4 Aplikace „meas“ - zobrazení obrázku

Zobrazovací programy pro data od aplikace *meas* jsou v DVD příloze v adresáři *programy/BF_TCP* (komunikace TCP/IP protokolem) a *programy/BF_UDP* (komunikace UDP/IP protokolem). Zobrazovací program se po stisku tlačítka *AUTO* automaticky připojuje přes Ethernet na aplikaci *meas* běžící v modulu. Na obr. 31 je okno programu BF_UDP, ve kterém je cyklicky načítán obrázek z modulu UDP/IP protokolem v rozlišení 720×576 pixelů. V levé dolní části okna je zobrazován čas načtení snímku v sekundách.

Je dosaženo frekvence přibližně 3 snímky/sekundu.



9.6 Další užitečné programy

9.6.1 Program Hercules

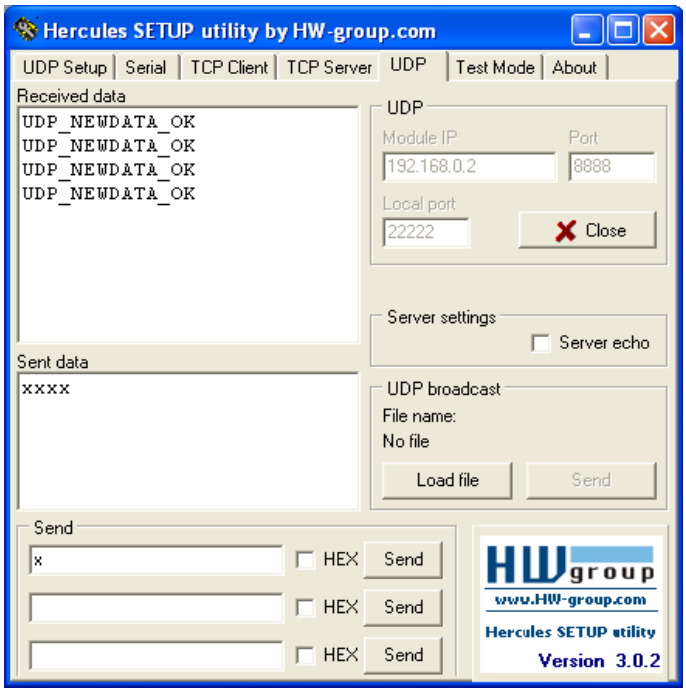
Program Hercules je užitečným nástrojem, který plní funkci TCP (telnet) klienta, UDP klienta, nebo terminálu RS-232. Je tvořen jedním *.exe souborem bez nutnosti instalace. Program je dostupný jako freeware (http://www.hw-group.com).

9.6.1.1 Připojení programem Hercules na TCP port

Připojení na vzdálený TCP/IP port je jednoduché. Po otevření záložky *TCP Client* je potřeba zadat IP adresu a cílový port. Otevřeme záložku TCP Client v programu Hercules. Tlačítkem *Connect* otevřeme spojení, a pak již je možné pomocí tlačítka *Send* odesílat zadaný text serveru. Je možné odesílat data v ASCII nebo HEX formátu. Odeslaná i přijatá data program zobrazuje. Přijatá data lze ukládat (logovat) do souboru (pravé tlačítko na okno *Received Data* → *Log to File*).

9.6.1.2 Připojení programem Hercules na UDP port

Otevřeme záložku UDP v programu Hercules. Kromě portu na který se připojujeme, je potřeba zadat také lokální port PC. V případě, který ukazuje obrázek, je lokální port 22222 a koresponduje s vytvořeným UDP spojením v souboru main.c: *uip_udp_new(&ipaddr, HTONS(22222))*.



9.6.2 Wireshark

Program Wireshark (dříve známý jako Ethereal) je protokolový analyzer. Mezi jeho nejčastější použití patří analýza a ladění problémů v počítačových sítích, vývoj software, vývoj komunikačních protokolů a studium síťové komunikace. Program je pod open source licencí.

Wireshark nabízí grafické uživatelské rozhraní a mnoho možností uspořádání a filtrování zobrazených informací. Aplikace umí přepnout síťovou kartu do promiskuitního režimu a díky tomu dokáže zachytávat veškerou komunikaci na připojeném médiu (i pakety, které nejsou určeny pro ni).

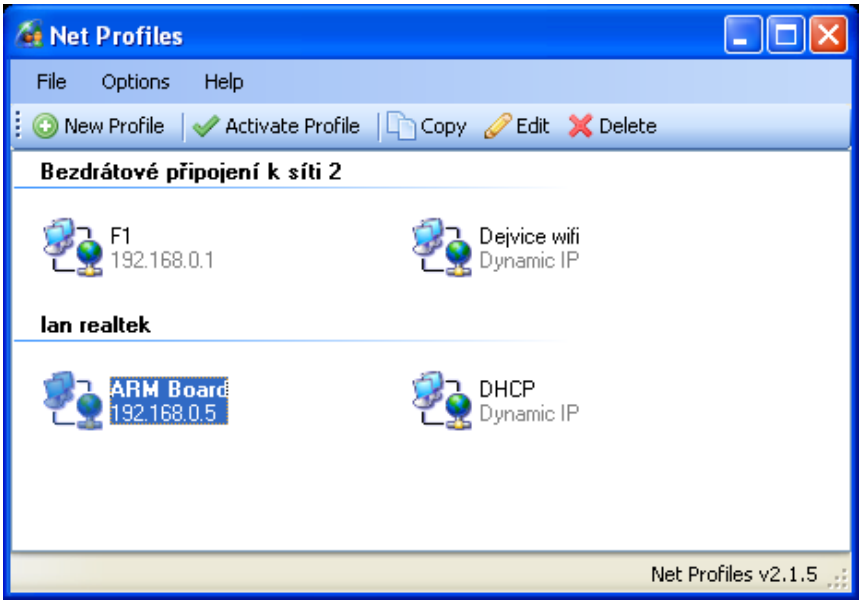
Na obr. 33 je znázorněna zachycená komunikace na síti po odeslání ICMP požadavku *ping* od PC k modulu. Prvním ARP paketem odeslaným všem připojeným zařízením se PC

dotazuje na MAC adresu zařízení kterému je ping určen. Druhým ARP paketem modul odpovídá svou adresou MAC. Další pakety jsou již požadavky a odezvy ICMP *ping*.

1	0.000000	AsustekC_a7:6d:c5	Broadcast	ARP	Who has 192.168.0.2? Tell 192.168.0.5
2	0.000094	3com_03:04:05	AsustekC_a7:6d:c5	ARP	192.168.0.2 is at 00:01:02:03:04:05
3	0.000102	192.168.0.5	192.168.0.2	ICMP	Echo (ping) request
4	0.000201	192.168.0.2	192.168.0.5	ICMP	Echo (ping) reply
5	0.994113	192.168.0.5	192.168.0.2	ICMP	Echo (ping) request
6	0.994217	192.168.0.2	192.168.0.5	ICMP	Echo (ping) reply
7	1.994272	192.168.0.5	192.168.0.2	ICMP	Echo (ping) request
8	1.994383	192.168.0.2	192.168.0.5	ICMP	Echo (ping) reply

9.6.3 Net Profiles

Net Profiles je program rozšiřující nastavení sítě v MS Windows. Windows umožňují nastavit ve vlastnostech sítě pouze jednu konfiguraci, buď statickou adresu, nebo dynamicky přidělovanou (DHCP). Pokud je počítač střídavě připojován do různých sítí, je velmi nepohodlné toto nastavení vždy měnit pro danou síť. *Net Profiles* umožňuje vytvářet více profilů nastavení sítě (viz. obr. 34), které je možné jednoduše přepínat. Program je freeware.



Obrázek 34: Vytvořené profily v programu Net Profiles.

Při práci s modulem byl program Net Profiles používán. Jeden profil byl nastaven na statickou IP adresu pro komunikaci s modulem po kříženém kabelu a druhý pro připojení k internetu.

9.7 Vzdálený přístup k modulu prostřednictvím sítě LAN nebo WAN

Doposud byla řeč o přímém propojení moulu s PC přes UTP kabel. V této kapitole budou popsány možnosti přístupu k modulu, nebo k více modulům z počítače, který se nachází v lokální síti LAN, nebo v celosvětové síti WAN (Internetu).

9.7.1 Aktivní síťové prvky

Jsou to zařízení, která slouží ke vzájemnému propojování v počítačových sítích a přitom nejsou jen pasivními mechanickými záležitostmi (jakými jsou např. kabely, konektory, apod.).

Switch (přepínač) je aktivní síťový prvek, propojující jednotlivé segmenty sítě. Pracuje na druhé vrstvě OSI modelu. Switch obsahuje porty (neplést s porty u TCP/IP), na něž se připojují síťová zařízení nebo části sítě. Switche nahradily dříve používané HUBy (rozbočovače), které signál jednoduše kopírovaly na všechna připojená zařízení. Výhodou oproti HUBu je vyšší výkon (stanice připojené k různým rozhraním switche navzájem nesoutěží o médium) znamená přínos i pro bezpečnost sítě, protože médium již není sdíleno a data se vysílají jen do rozhraní, jímž je připojen jejich adresát.

Router (směrovač) je aktivní síťový prvek, který přeposílá datagramy směrem k jejich cíli. Tento proces se nazývá routování a probíhá na třetí vrstvě modelu OSI. Routery se používají k propojení dvou a více sítí. Router je potřeba nakonfigurovat pro danou situaci, ve které bude použit. Konfigurace většinou probíhá pomocí HTML stránky.

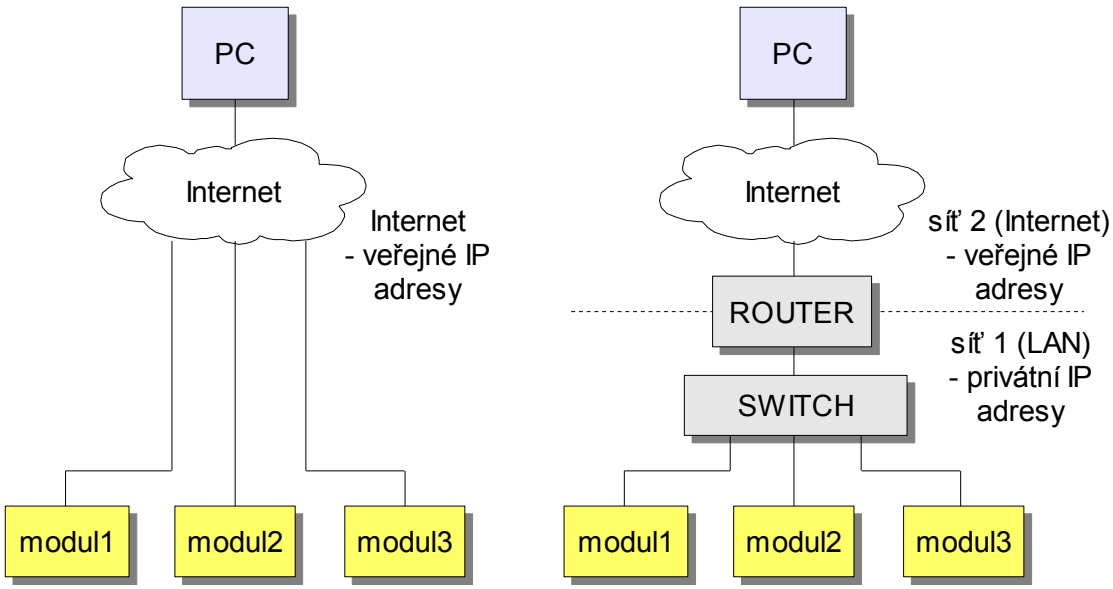
Zjednodušeně řečeno, switch použijeme, pokud potřebujeme síť rozvětvit, router v případě, že potřebujeme dvě sítě propojit. K propojování s těmito aktivními prvky se používají **rovné (nekřížené)** UTP kabely. Tato zařízení si křížení nastaví hardwarově. Některé dokáží dokonce rozpoznat, je-li k nim připojen křížový či rovný kabel a podle toho se přizpůsobit.

9.7.2 Ovládání modulů ze stejné lokální sítě

Pokud chceme propojit například 3 moduly s jedním PC lokálně, je situace jednoduchá. Všechny zařízení připojíme do jednoho switche. IP adresy nastavíme všechny například na 192.168.0.1 – 192.168.0.4.

9.7.3 Ovládání modulů z jiné sítě (Internetu)

Zde je situace o něco složitější. V síti Internet se používají **veřejné IP adresy**. V principu tak každé zařízení, které chceme do Internetu připojit, musí mít jedinečnou veřejnou IP adresu. Pokud jsou pro každý modul i pro PC k dispozici veřejné IP adresy, budou na sebe zařízení "vidět" přímo (viz. obr. 36 vlevo). Ale v situaci, kdy je modul připojen v privátní síti LAN, která je připojena do Internetu prostřednictvím routeru (viz. obr. 36 vpravo), bude celá



Obrázek 36: Připojení modulu do sítě Internet (vlevo – přímo, vpravo – přes router)

privátní síť viditelná v Internetu pod jedinou veřejnou IP adresou. Adresy v lokální síti budou **neveřejné**, z Internetu skryté (např. 192.168.0.X, 10.X.X.X). Výhodou neveřejných IP adres je vyšší ochrana proti útokům z Internetu.

Pro přístup více zařízení z lokální sítě na Internet pod jedinou veřejnou adresou, používají routery techniku překladu síťových adres NAT (Network Address Translation). NAT umožňuje překládat privátní IP adresy z lokální sítě na jedinečnou veřejnou IP adresu, která slouží pro vstup do jiné sítě (např. Internetu). Překládanou adresu si uloží do NAT tabulky pod náhodným portem, při odpovědi si v tabulce vyhledá port a pošle pakety na adresu přiřazenou k danému portu.

10 Závěr

Modul byl navržen a zkonstruován. Jeho hlavními komponentami je mikrokontrolér STR912 a obvod fyzické vrstvy Ethernetu STE100P s maximální přenosovou rychlostí 100 Mbit/s. Kromě Ethernetu modul obsahuje řadu standardních rozhraní (USB Full-speed, SPI, RS-232, I/O brány, atd.).

Část práce byla věnována výběru vhodného překladače. Velikost výsledného kódu po překladu je 280 kB, což neumožňuje použití překladačů, kterými disponují vývojová prostředí Keil uVision nebo IAR EWARM ve zdarma dostupných (a tím bohužel velikostí kódu omezených) verzích. Pro překlad kódu byl zvolen překladač GNUARM.

Byly prozkoumány možnosti implementace protokolů TCP/IP pro použitý mikrokontrolér, včetně systému μClinux. Tento systém má TCP/IP funkcionalitu implementovánu, avšak pro svůj běh má vysoké nároky na paměť RAM (až 4 MB) proto bylo jeho použití zamítnuto. Zvolen byl TCP/IP stack uIP 1.0 (micro IP) a vytvořeny či upraveny aplikace, které prostřednictvím uIP stacku budou komunikovat s PC. Vytvořené aplikace a změny provedené v kódu uIP byly detailně popsány.

Vytvořené aplikace jsou zaměřeny na ovládání senzorů, obvodových bloků a zařízení v oblasti videometrie. Jednou z aplikací je HTTP server, umožňující odesílat HTML stránky do PC. Dále lze s modulem komunikovat prostřednictvím terminálu telnet pomocí nadefinovaných textových zpráv. Byly vytvořeny aplikace pro rychlý přenos dat do PC protokoly TCP a UDP. Přenosem obrazových dat z FLASH paměti mikrokontroléru a jejich zobrazováním programem v PC bylo dosaženo rychlosti 1,3 MB/s protokolem UDP a 1,1 MB/s protokolem TCP. Další funkcí modulu je obousměrný most (převodník) mezi rozhraními TCP/IP – RS232.

Funkčnost modulu a jeho programového vybavení byla otestována a demonstrována řízením polohovacího mechanismu pro bezdotykové měření polohy přes Ethernet. Úlohu je možné řídit z prostředí HTML stránky v internetovém prohlížeči nebo textovými zprávami z prostředí klienta telnet. Dále byla demonstrována úloha čtení obrazových dat ze signálového procesoru Blackfin, připojeného přes RS-232 k modulu a jejich zobrazování na PC přes Ethernet.

Nakonec byly navrženy způsoby připojení několika modulů do lokální počítačové sítě (LAN) a do Internetu, s použitím aktivních síťových prvků.

Jednotlivé body zadání práce byly splněny. Byly navrženy hardwarové i softwarové prostředky pro zadané účely a zpracována dokumentace. Pro případný další vývoj doporučuji implementovat do uIP stacku modul *uip-split*, který řeší problém degradace výkonu uIP stacku vhodnějším způsobem. Pro výkonnější aplikace doporučuji vyzkoušet TCP/IP stack *lwip*.

11 Seznam obrázků

Obrázek 1: Požívaná rozhraní pro lokální ovládání a čtení dat z experimentu.....10
Obrázek 2: Vzdálené ovládání experimentu z lokální sítě a přes Internet.....11
Obrázek 3: PC jako převodník rozhraní.....11
Obrázek 4: Blokové schema modulu, který bude navržen.....12
Obrázek 5: 7-vrstvový model síťové komunikace - OSI model.....15
Obrázek 6: Získání MAC adresy od zařízení na jiném segmentu sítě.....18
Obrázek 7: Potvrzování přijatých TCP/IP datagramů.....19
Obrázek 8: Boot módy mikrokontroléru STR75x (převzato z [20]).....24
Obrázek 9: Volba boot módu mezi FLASH a SystemMemory mode.....26
Obrázek 10: Blokové schéma navrženého modulu.....28
Obrázek 11: Blokové schéma vnitřní struktury mikrokontroléru STR912 (převzato z [3]).....31
Obrázek 12: Blokové schéma bloku ENET (MAC/DMA) (převzato z [5]).....32
Obrázek 13: Vnitřní blokové zapojení obvodu STE100P (převzato z [7]).....34
Obrázek 14: Připojení JTAG debuggeru.....38
Obrázek 15: Rozvržení komponent programu RIDE.....43
Obrázek 16: Hlavní smyčka programu v souboru main.c.....46
Obrázek 17: Struktura zdrojových souborů (vývoj. prostředí RIDE).....49
Obrázek 18: Směrování na jednotlivé aplikace pomocí uip routeru.....53
Obrázek 19: TCP a UDP porty.....54
Obrázek 20: Princip čtení HTML stránky z mikrokontroléru.....60
Obrázek 21: Experiment čtení dat ze signálového procesoru Blackfin.....64
Obrázek 22: Propojení modulu s PC.....70
Obrázek 23: Karta nastavení sítě v PC.....71
Obrázek 24: Test dostupnosti cílového zařízení příkazem "ping".....72
Obrázek 25: Připojení na aplikaci "Terminal" klientem telnet v MS Windows.....72
Obrázek 26: Mercury C-862.....74
Obrázek 27: IntelliStages M-511.....74
Obrázek 28: Blokové schéma ovládání polohovacího mechanismu z PC přes Ethernet.....74
Obrázek 29: HTML stránka načtená z mikrokontroléru.....75

Obrázek 30: Aplikace telnet na portu 23....76
Obrázek 31: Zobrazovač obrazových dat přijatých přes Ethernet....77
Obrázek 32: Program Hercules, připojení k UDP portu....78
Obrázek 33: Výpis zachycené komunikace na síti programem Wireshark....79
Obrázek 34: Vytvořené profily v programu Net Profiles....79
Obrázek 35: Propojení PC s více moduly pomocí prvku switch....81
Obrázek 36: Připojení modulu do sítě Internet (vlevo – přímo, vpravo – přes router)....81
Obrázek 37: Seznam součástek....91
Obrázek 38: Osazovací výkres....92
Obrázek 39: Vrstva TOP....93
Obrázek 40: Vrstva BOTTOM....93
Obrázek 41: Rozmístění součástek a nastavení jumperů....96
Obrázek 42: Osazovací výkres....97
Obrázek 43: Vrstva TOP....98
Obrázek 44: Vrstva BOTTOM....98

12 Seznam použitých zkratek

ACK – Acknowledge – příznak TCP/IP segmentu, označující potvrzení úspěšného příjmu dat

ADC – Analog Digital Converter – analogově/číslicový převodník

API – Application Programming Interface – rozhraní pro programování aplikací

ARP – Address Resolution Protocol – protokol pro vyhledání adresy MAC podle IP adresy

CGI – Common Gateway Interface – skriptovací jazyk

CSMA/CD – Carrier Sense Multiple Access with Collision Detection – metoda přístupu k médiu používaná v Ethernetu

DMA – Direct Memory Access – přístup do paměti bez účasti procesoru

MCU – Microcontroller Unit – mikrokontrolér (mikroprocesor + periferie na jednom čipu)

MAC – Media Access Control – řadič přístupu k médiu / MAC adresa

GPIO – General Purpouse I/O – vstupně/výstupní víceúčelová brána mikrokontroléru

HTML – Hypertext Markup Language – jazyk pro vytváření HTML stránek

HTTP – Hypertext Transfer Protocol – internetový protokol pro přenos dat (HTML stránek)

I2C – Inter-Integrated Circuit – multi-master sériová sběrnice

IAP – In Application Programing – metoda programování mikroprocesoru v aplikaci

ICMP – Internet Control Message Protocol – protokol Internetu pro odesílání chyb. zpráv

IEEE – Institute of Electrical and Electronic Engineers – organizace, normy

IP – Internet Protocol – základní protokol Internetu

ISP – In System Programing – metoda programování mikroprocesoru (např. přes JTAG)

MS Windows – operační systém Microsoft Windows

NAT – Network Address Translation – překlad síťových adres routerem

OS – operační systém

OSI – norma, sedmivrstvý model síťové komunikace

RAM – Random Access Memory – paměť s náhodným přístupem

RFC – Request for Comments – standardy, popisující Internetové protokoly

SPI – Serial Peripheral Interface – sériová sběrnice používaná nejčastěji pro komunikaci mezi integrovanými obvody

RJ-45 – konektor (koncovka) UTP Ethernetových kabelů

RS-232 – standard, sériové rozhraní k propojení dvou zařízení

TCP – Transmission Control Protocol – spojově orientovaný protokol se zárukou doručení dat, jeden ze základních protokolů Internetu

Telnet – Telecommunication Network – protokol na aplikační vrstvě, používaný v počítačových sítích pro spojení klient-server pomocí TCP

UDP – User Datagram Protocol – nespojovaný protokol v počítačových sítích, bez záruky doručení dat

uIP – Micro IP – TCP/IP stack určený pro mikrokontroléry

USB – Universal Serial Bus – sériová sběrnice pro připojování zařízeni k PC

UTP – Unshielded Twisted Pair – typ kabelu, nestíněná kroucená dvojlinka, používaná v počítačových sítích

13 Seznam literatury

[1] The Insider's Guide to The STR91x ARM9,
http://www.hitex.com/index.php?id=download-insiders-guides

[2] ARM966E-S Technical Reference Manual,
http://www.arm.com/documentation/ARMProcessor_Cores/

[3] Documents and files for family STR9, http://www.st.com/mcu/familiesdocs-101.html

[4] STR91x Datasheet, [3]

[5] STR91x Reference Manual (UM0216), [3]

[6] STR91x Hardware development getting started (AN2339), [3]

[7] STE100P - Single port fast ethernet transceiver (datasheet + AN1301), [3]

[8] STR91xFA Firmware Library (UM0233), [3]

[9] STR91xFA ENET Firmware Library (UM0248), [3]

[10] uIP TCP/IP stack, http://www.sics.se/~adam/uip/index.php/Main_Page

[11] Zdrojové kódy uIP 1.0, http://www.sics.se/~adam/uip/index.php/Download

[12] Dunkels, A. (Swedish Institute of Computer Science): *The uIP 1.0 Reference Manual,*
http://www.sics.se/~adam/uip/index.php/Documentation

[13] lwIP - Lightweight TCP/IP stack, http://savannah.nongnu.org/projects/lwip/

[14] Peterka, J.: *Rodina protokolů TCP/IP, verze 2.2*, http://www.earchiv.cz/l213/index.php3

[15] Peterka, J.: *Referenční model ISO/OSI,* http://www.earchiv.cz/a92/a212c110.php3

[16] Záhlava, V.: *OrCAD 10*. Grada Publishing, a.s., 2004, ISBN 80-247-0904-X

[17] Záhlava, V.: *Návrh a konstrukce desek plošných spojů*. Vydavatelství ČVUT, 2005, ISBN 80-01-03351-1

[18] Documents and files for family STR7, http://www.st.com/mcu/familiesdocs-86.html

[19] STR75x Datasheet, [18]

[20] STR75x Reference Manual (RM0003), [18]

[21] STR75x Hardware development getting started (AN2419), [18]

[22] STR75x Software Library (UM0218), [18]

14 Obrazové přílohy

14.1 Modul STR75x – opravená verze

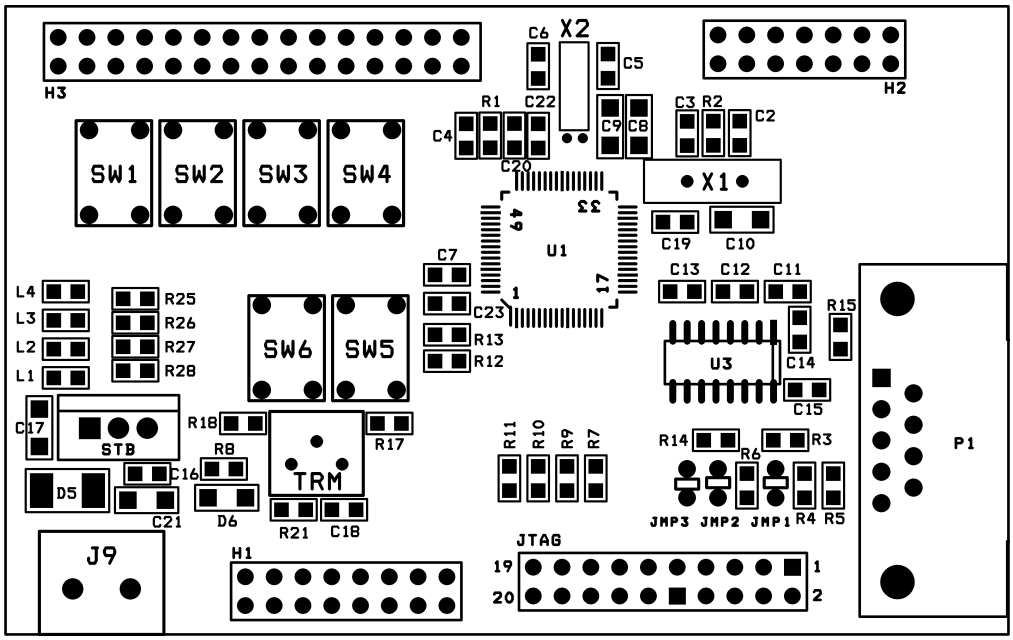
Schéma zapojení, seznam součástek, vrstvy plošného spoje, fotografie hotového modulu (následující obrazové přílohy se týkají přepracované verze modulu STR75x, pouze fotografie vyobrazuje první verzi modulu).

14.1.1 Schéma zapojení

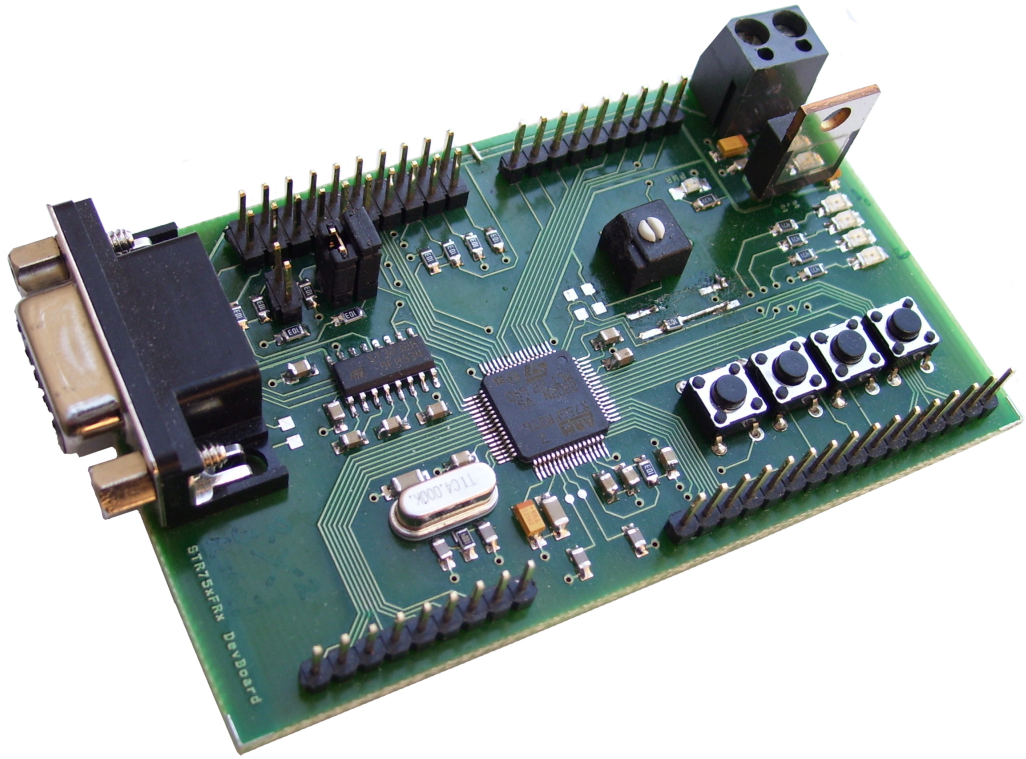
14.1.2 Seznam součástek

part reference	název/hodnota	poznámka
integrované obvody		
U1	str751	mikrokontrolér
U3	ST3232_16P	převodník TTL/232
STB	MC33269T	stabilizátor
ostatní polovodičové součástky		
L1, L2, L3, L4	LED	diody LED
X2	32k	krystal
X1	4M	krystal
D5	DIODE	ochranná dioda
rezistory		
R8	470	
R25,R26,R27,R28	1k	
R1,R3,R4,R5,R6,R7, R9,R10,R11,R12,R13, R14,R17,R18,R21	10k	
R2	1M	
R15	nezapojen	
TRM	10k	
kondenzátory		
C2,C3,C5,C6	22p	
C7	33n	
C4,C11,C12,C13,C14, C15,C18,C19,C20, C22,C23	100n	
C16	330n	
C9,C10	1u	
C8,C17,C21	10u	
tlačítka		
SW4	RESET	
SW1	TL1	uživ. tlačítko
SW2	TL2	uživ. tlačítko
SW3	TL3	uživ. tlačítko
SW5	BOOT0	boot mode
SW6	BOOT1	moot mode
konektory, jumpery		
J9	napajeci_kon	svorkovnice
JTAG	JTAG_CON	konektor JTAG
H1	HEAD1	I/O piny
H2	HEAD2	I/O piny
H3	HEAD3	I/O piny
P1	CONNECTOR DB9	
JMP1,JMP2,JMP3	JUMPER	

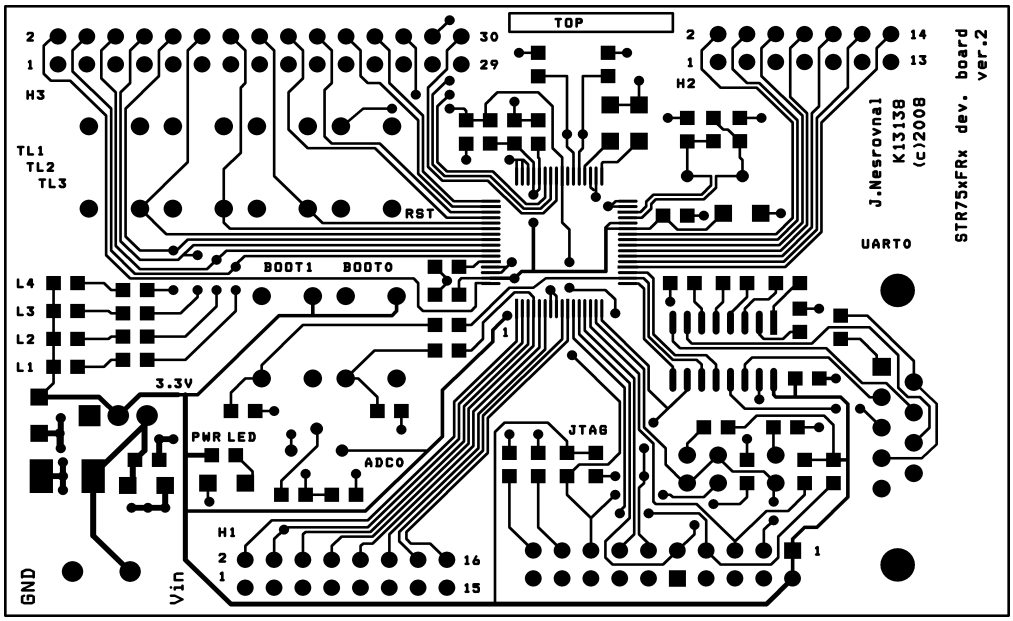
14.1.3 Osazovací výkres



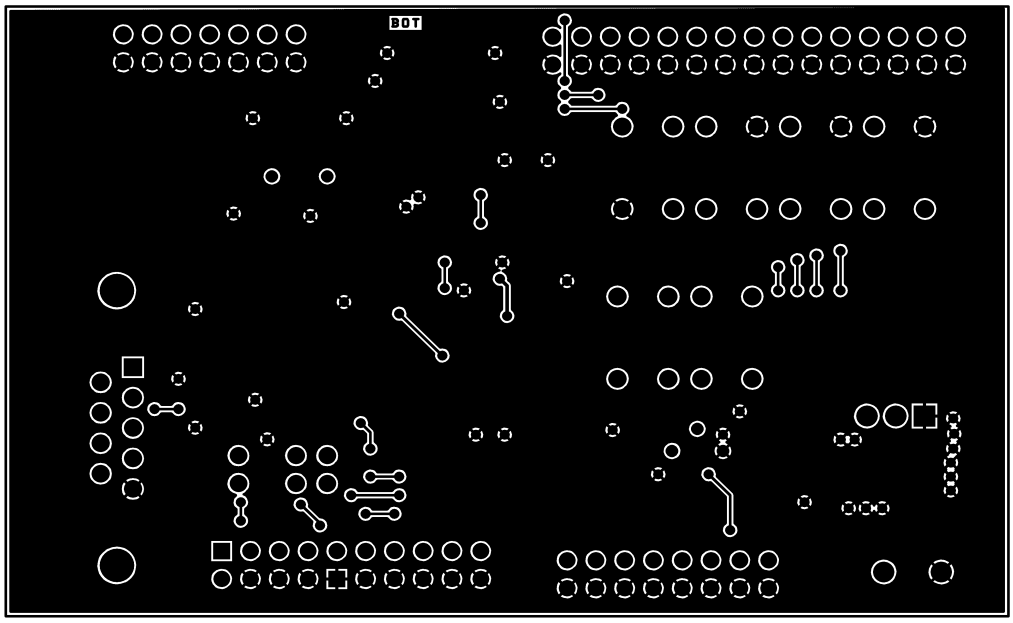
14.1.4 Fotografie modulu – první verze modulu



14.1.5 Vrstva TOP – měřítko 1,5:1 - 89×54 mm



14.1.6 Vrstva BOTTOM – měřítko 1,5:1 - 89×54 mm



Obrázek 40: Vrstva BOTTOM.

14.2 Modul STR912

Schéma zapojení, seznam součástek, nastavení jumperů, osazovací výkres, vrstvy plošného spoje, fotografie modulu

14.2.1 Schéma zapojení

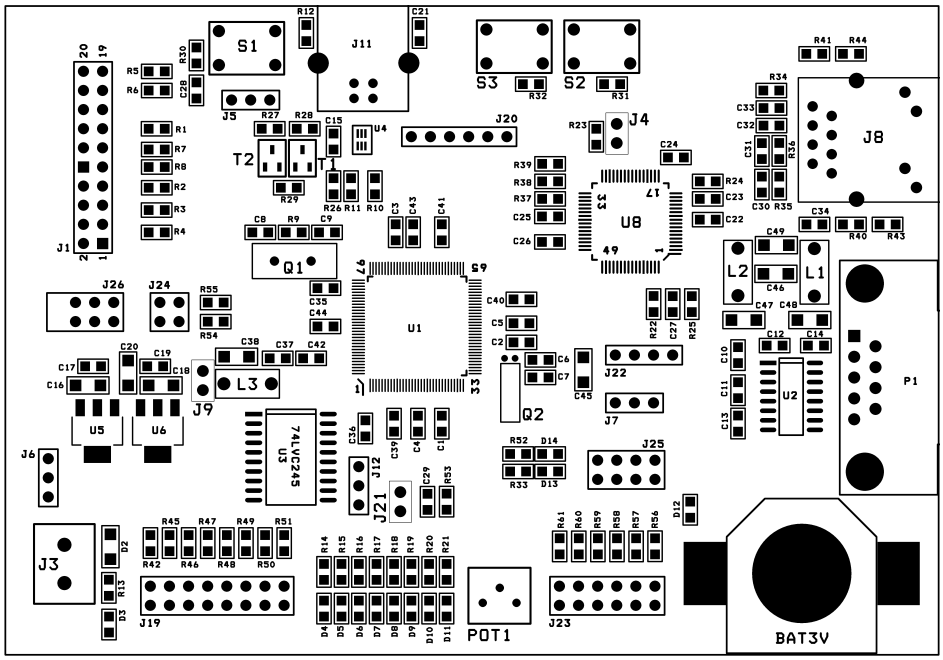
14.2.2 Seznam součástek

part reference	název/hodnota	poznámka
integrované obvody		
U1	STR912FW44_LQFP128	
U8	STE100P_TQFP64	
U2	ST3232	TTL/RS-232
U3	74LVC245	budič LED
U4	USBLC-6	ESD ochrana
U6	LD1117S18	stabilizátor
U5	LD1117S33	stabilizátor
ostatní polovodiče		
D12	1N4148	dioda baterie
T1,T2	BC850CL	
D2	SM4004	ochr. dioda
Q1	25M	krystal
Q2	32.768k	krystal RTC
rezistory		
R25	0R	
R10,R11	22R	
R35,R36	49.9R	
R34	100R	
R14,R15,R16,R17, R18,R19,R20,R21, R40,R41,R42,R45, R46,R47,R48,R49, R50,R51,R54,R55, R56,R57,R58,R59, R60,R61	330R	
R13,R33,R52,R53	1k	
R26	1k5	USB full speed
R24	5k+/-1%	ref. proud
R1,R2,R3,R4,R5, R6,R7,R8,R22, R23,R28,R29,R30, R31,R32,R37,R38, R39, R43,R44	10k	
R27	100k	
R9,R12	1M	
POT1	10k	trimr
kondenzátory		
C30,C31	10p	
C6,C7,C8,C9	22p	k xtalu MCU
C21	4,7n	
C1,C2,C3,C4,C5, C10,C11,C12,C13, C14,C15,C17,C19, C22,C23,C24,C25, C26,C27,C28,C29, C32,C33,C34,C35, C36,C37,C39,C40, C41,C42,C43,C44	100n	

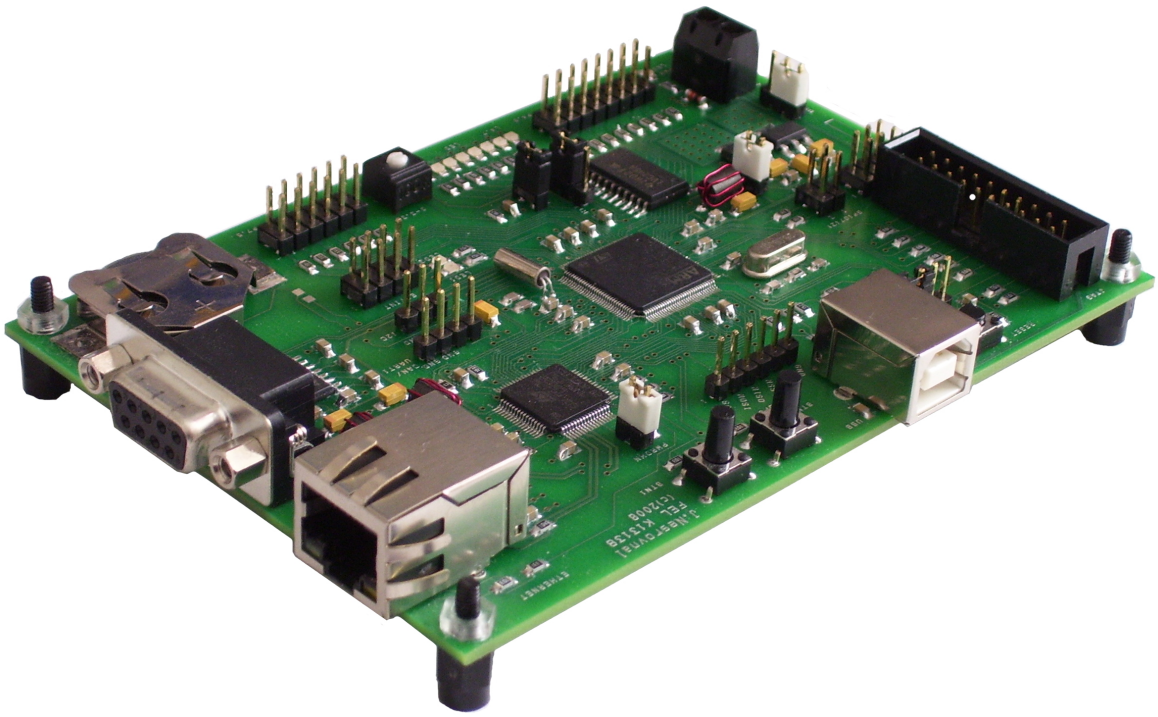
part reference	název/hodnota	poznámka
kondenzátory		
C16,C18,C20, C38,C45	10uF	
C46,C47,C48,C49	22uF	
diody LED		
D3	LED	napájecí
D11	L4.0	LED GPIO4
D10	L4.1	
D9	L4.2	
D8	L4.3	
D7	L4.4	
D6	L4.5	
D5	L4.6	
D4	L4.7	
D13	LED1	GPIO8.1
D14	LED2	GPIO8.2
tlačítka		
S1	RESET	
S2	BTN1	
S3	BTN2	
jumpery		
J9	AV	
J4	PWRDWN	
J6	USB/EXT	
J3	V_EXT	
J5	USB_1k5	
J12	BUS FUNC	
J21	ADC0	
konektory		
J19	GPIO4	I/O brána 4
J23	GPIO7 (CAST1)	I/O brána 7
J24	GPIO7 (CAST2)	I/O brána 7
J11	USB_B	USB kon.
J20	SSP0	
J22	CAN	
P1	DB9 FEMALE	CANON kon.
J25	EXTINT	
J8	RJ-45_dipl	
J7	I2C	
J1	JTAG	
J26	3.3V_OUT	výstupy 3.3V
ostatní součástky		
B1	BAT3V	baterie
L1,L2,L3	BEAD	ferit

14.2.3 Rozmístění součástek a nastavení jumperů

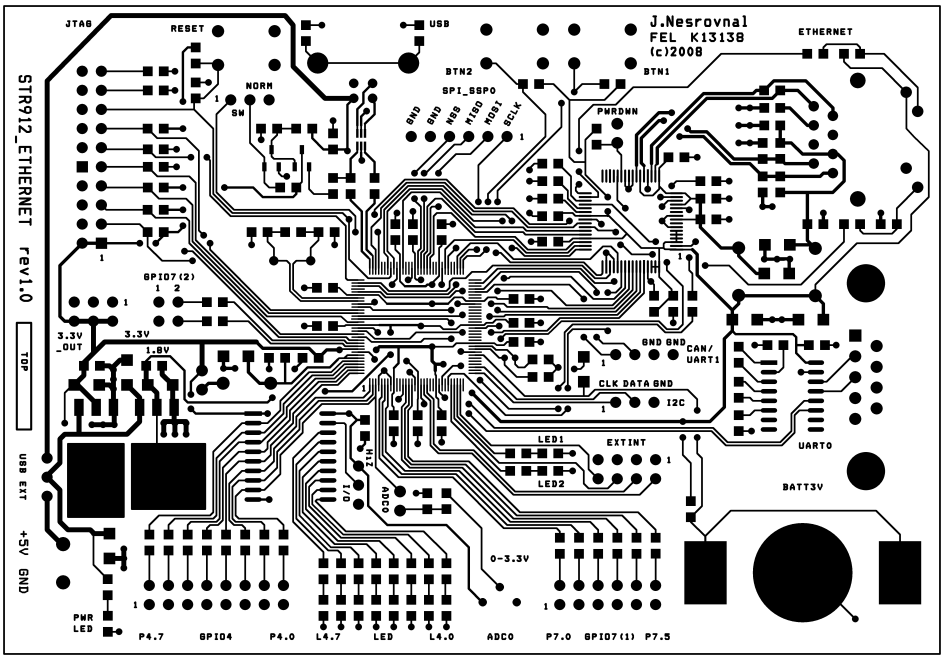
14.2.4 Osazovací výkres



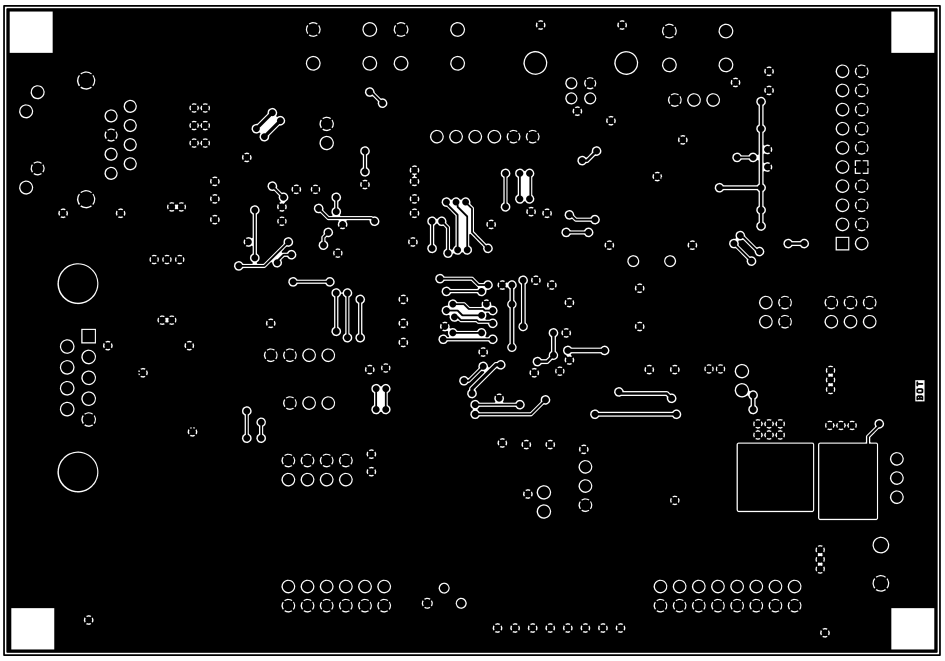
14.2.5 Fotografie modulu



14.2.6 Vrstva TOP – měřítko 1:1 - 124×86 mm



14.2.7 Vrstva BOTTOM – měřítko 1:1 - 124×86 mm



Obrázek 44: Vrstva BOTTOM.

15 Obsah přiloženého DVD

- soubor PDF s tímto textem

- adresář OrCAD_Data – podklady pro výrobu plošných spojů
- adresář STR755_modul – zdrojové kódy a dokumentace
- adresář STR912_modul – zdrojové kódy a dokumentace
- adresář prekladace – použité překladače zdrojových kódů
- adresář programy – programy pro PC
- adresář foto – fotografie vytvořeného hardware