VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

WEB SERVER PRO VESTAVĚNÉ APLIKACE

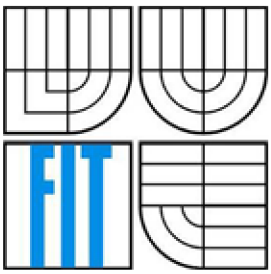
DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

TOMÁŠ PIRKL

BRNO 2007

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

WEB SERVER PRO VESTAVĚNÉ APLIKACE

EMBEDDED WEBSERVER

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

PIRKL TOMÁŠ

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. ŠIMEK VÁCLAV

BRNO 2007

Zadání diplomové práce/5878/2006/xpirkl01

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav počítačových systémů Akademický rok 2006/2007

Zadání diplomové práce

Řešitel: **Pirkl Tomáš**
Obor: Výpočetní technika a informatika
Téma: **Web server pro vestavěné aplikace**
Kategorie: Počítačové sítě

Pokyny:

1. Věnujte se problematice návrhu vestavěných systémů s možností síťové komunikace.
2. Prostudujte standard Ethernet a detailně se věnujte komunikačním protokolům UDP, TCP/IP a HTTP.
3. Navrhněte vhodnou platformu pro implementaci vestavěného web serveru, která bude obsahovat mikrokontrolér, řadič rozhraní Ethernet a paměťový blok.
4. Proveďte výběr vhodných komponent pro účely implementace web serveru na desce plošných spojů.
5. Proveďte realizaci zmíněné desky plošných spojů a její oživení.
6. Prakticky demonstrujte funkčnost vašeho řešení.
7. Uvažujte možnosti dalšího rozšíření, např. využití v kombinaci s platformou FITkit.

Literatura:

- dle pokynů vedoucího

Při obhajobě semestrální části diplomového projektu je požadováno:

- Požadováno splnění prvních čtyř bodů zadání.

Podrobné závazné pokyny pro vypracování diplomové práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva diplomové práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap, které byly vyřešeny v rámci ročníkového a semestrálního projektu (30 až 40% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním paměťovém médiu (disketa, CD-ROM), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Šimek Václav, Ing.,** UPSY FIT VUT
Datum zadání: 1. listopadu 2006
Datum odevzdání: 31. července 2007

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav počítačových systémů a sítí
612 66 Brno, Božetěchova 2

L.S.

doc. Ing. Zdeněk Kotásek, CSc.

vedoucí ústavu

LICENČNÍ SMLOUVA
POSKYTOVANÁ K VÝKONU PRÁVA UŽÍT ŠKOLNÍ DÍLO

uzavřeni mezi smluvními stranami

1. Pan

Jméno a příjmení: Tomáš Pirkl
Id studenta: 15904
Bytem: Starý Maletím 39E, 78901 Maletín
Narozen: 23. 06. 1982, Zábřeh
(dále jen "autor")

a

2. Vysoké učení technické v Brně

Fakulta informačních technologií
Se sídlem Božetěchova 2/1, 612 66 Brno, IČO 00216305
jejímž jménem jedná na základe písemného pověření děkanem fakulty:

_____

(dále jen "nabyvatel")

Článek 1

Specifikace školního díla

1. Předmětem této smlouvy je vysokoškolská kvalifikační práce (VŠKP):
diplomová práce

Název VŠKP: Web server pro vestavěné aplikace
Vedoucí školitel VŠKP: Šimek Václav, ing.
Ústav: Ústav počítačových systémů
Datum obhajoby VŠKP: _____

VŠKP odevzdal autor nabyvateli v:

tištěné formě počet exemplářů: 1
elektronické formě počet exemplářů: 2 (jeden ve skladu dokumentů, 1 na CD)

2. Autor prohlašuje, že vytvořil samostatnou vlastní tvůrčí činností dílo shora popsané a specifikované. Autor dále prohlašuje, že při zpracovávaní díla se sám nedostal do rozporu s autorským zákonem a předpisy souvisejícími a že je dílo dílem původním.
3. Dílo je chráněno jako dílo dle autorského zákona v platném znění.
4. Autor potvrzuje, že listinná a elektronická verze díla je identická.

Článek 2

Udělení licenčního oprávněni

1. Autor touto smlouvou poskytuje nabyvateli oprávnění (licenci) k výkonu pravá uvedené dílo nevýdělečně užít, archivovat a zpřístupnit ke studijním, výukovým a výzkumným účelům. včetně pořizování výpisů, opisů a rozmnoženin.
2. Licence je poskytována celosvětově, pro celou dobu trváni autorských a majetkových práv k dílu.
3. Autor souhlasí se zveřejněním díla v databázi přístupné v mezinárodní síti:
   - □ ihned po uzavření této smlouvy
   - □ 1 rok po uzavření této smlouvy
   - □ 3 roky po uzavření této smlouvy
   - □ 5 let po uzavření této smlouvy
   - □ 10 let po uzavřeni této smlouvy

   (z důvodu utajení v něm obsažených informací)
4. Nevýdělečné zveřejňování díla nabyvatelem v souladu s ustanovením § 47b zákona č. 111/1998 Sb., v platném znění, nevyžaduje licenci a nabyvatel je k němu povinen a oprávněn ze zákona.

Článek 3

Závěrečné ustanovení

1. Smlouva je sepsána ve třech vyhotoveních s platností originálu, přičemž po jednom vyhotovení obdrží autor a nabyvatel, další vyhotovení je vloženo do VŠKP.
2. Vztahy mezi smluvními stranami vzniklé a neupravené touto smlouvou se řídí autorským zákonem, občanským zákoníkem, vysokoškolským zákonem, zákonem o archivnictví, v platném znění a popř. dalšími právními předpisy.
3. Licenční smlouva byla uzavřena na základě svobodné a pravé vůle smluvních stran, s plným porozuměním jejímu textu i důsledkům, nikoli v tísni a za nápadně nevýhodných podmínek.
4. Licenční smlouva nabývá platnosti a účinnosti dnem jejího podpisu oběma smluvními stranami.

V Brně dne: _____

Nabyvatel

Autor

Abstrakt

V této práci je popsána problematika návrhu Embedded systému s možností síťové komunikace. Je zde navrhnuta platforma pro implementaci Embedded web serveru, která obsahuje mikrokontrolér, řadič rozhraní Ethernet a paměťový blok. Dále práce obsahuje popis realizace web serveru na desce pro plošné spoje s rozhraním pro kartu MMC, dále ještě implementaci ovladače pro kartu MMC a souborového systému FAT. Také se zde zabývám síťovými referenčními modely ISO/OSI a TCP/IP standardem Ethernet a komunikačními protokoly UDP, TCP/IP a HTTP.

Klíčová slova

Vestavěný systém, web server, síťová komunikace, referenční model ISO/OSI, model TCP/IP, Ethernet, komunikační protokol UDP, komunikační protokol TCP/IP, komunikační protokol HTTP, mikrokontrolér, řadič rozhraní Ethernet, FAT, MMC driver, DPS.

Abstract

This work discuses Embedded Ethernet systems. Embedded Webserver Platform is designed. This design embrace microkontroler, Ethernet driver, and external SRAM. It includes implementation on PCB with MMC interface and software implementation of MMC driver and FAT file system. Reference model ISO/OSI, reference model TCP/IP, Ethernet and UDP, TCP, IP, HTTP protocol is presented.

Keywords

Embedded system, Webserver, net comunication, reference model ISO/OSI, reference model TCP/IP, Ethernet, protocol UDP, protocol TCP, protocol IP, protocol HTTP, microcontroler, Ethernet driver, FAT, MMC driver, PCB.

Citace

Tomáš Pirkl: Web server pro vestavěné aplikace, diplomová práce, Brno, FIT VUT v Brně, 2007

Web server pro vestavěné aplikace

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením Ing. Václava Šimka. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Tomáš Pirkl
22.5.2007

Poděkování

Poděkovat bych chtěl vedoucímu projektu Ing. Václavu Šimkovi, za poskytnutí veškerých materiálů a při pomoci řešení nejrůznějších problémů. Dále bych ještě rád poděkoval za finanční podporu UPSY na FIT VUT v Brně a vedoucímu tohoto ústavu Doc. Ing. Zdeňku Kotáskovi, Csc..

Obsah

Úvod

1.1 Motivace

Informační technologie jsou stále více aplikovány do celé řady přístrojů vzájemně propojených do rozličných sítí pomocí standardních síťových protokolů. Významnou úlohu v této situaci sehrává neustálý nárůst výkonu a snižování cen výpočetních a komunikačních zařízení. Sítě lokálního a globálního charakteru se dnes již staly nedílnou součástí moderní společnosti. Takovéto sítě mohou tvořit nejen osobní počítače, ale i rozličné mikroprocesorové systémy. Spojením celé řady zařízení a senzorů, vzniká zcela nový komplexní celek, umožňující sběr, sdílení a zpracování dat.

Díky této práci je možné si představit spoustu dalších aplikací. Uvažujme např.: domácí automatizaci, různé senzory, přístroje, bezpečnostní systémy, čtečky karet a ovládání budov, které mohou být snadno řízeny užitím speciálně navrženého softwarového rozhraní nebo pohodlně prostřednictvím internetového prohlížeče odkudkoli na světě. Velká výhoda HTTP serveru v Embedded systémech je, že web browser obstarává celé uživatelské rozhraní. Zobrazování informací může být realizováno jednoduše zasíláním ASCI řetězců (HTML zdrojových textů) klientovi, čímž se dosáhne minimálních požadavků na zdroje.

1.2 Cíl této práce

V této práci se zabývám implementací moderních internetových technologií do Embedded (*vestavěných*) zařízení. Cílem práce je realizace zařízení web serveru s jednočipovým mikropočítačem na desce plošných spojů. Tomu předcházelo seznámení se standardem Ethernet a komunikačními protokoly UDP, TCP, IP a HTTP. Byla navržena platforma pro implementaci vestavěného web serveru, dále jsem vybral vhodné komponenty pro účely implementace. Následoval návrh a realizace samotné desky plošných spojů a potřebného softwarového vybavení.

1.3 Struktura

Následující tedy druhá kapitola se krátce věnuje obecně Embedded systémům a jejím aplikačním možnostem v síťové komunikaci. Dále druhá kapitola popisuje síťové protokoly, zvolený mikrokontrolér, MMC kartu a nakonec se věnuje ještě souborovému systému FAT. Ve třetí kapitole jsou uvedena některá již existující řešení. Čtvrtá kapitola obsahuje návrh web serveru a výběr součástkové základny. Pátá kapitola se soustředí na hardwarovou realizaci. V šesté kapitole je popis softwarového řešení. Sedmá je věnována oživení a testování web serveru a osmá je věnována závěru, kde jsou shrnuty veškeré dosažené výsledky.

1.4 Návaznost na předchozí projekty

Tato práce navazuje na dříve vypracovaný semestrální projekt, ve kterém jsem se věnoval problematice návrhu vestavěných systémů s možností síťové komunikace (kapitola 2.1) dále jsem se v semestrálním projektu zabýval síťovými protokoly a návrhem platformy pro implementaci vestavěného web serveru (kapitola 2.2 a 4).

2 Teoretický úvod

2.1 Embedded systémy

Počítačové komunikační systémy a hlavně Internet hrají stále větší roli v každodenním dění kolem nás. Dnes už to ale není doména jen osobních počítačů nebo pracovních stanic. Víc a víc se uplatňují malé síťové uzly. Pro představu aplikace, které jsou schopné ovládat hardware prostřednictvím internetového prohlížeče, pro přenos a zobrazení stavu senzorů nebo pro automatické generování a odesílání dat při neočekávané události (např. pro účely spolehlivosti bezpečnosti atd.).

2.1.1 Definice

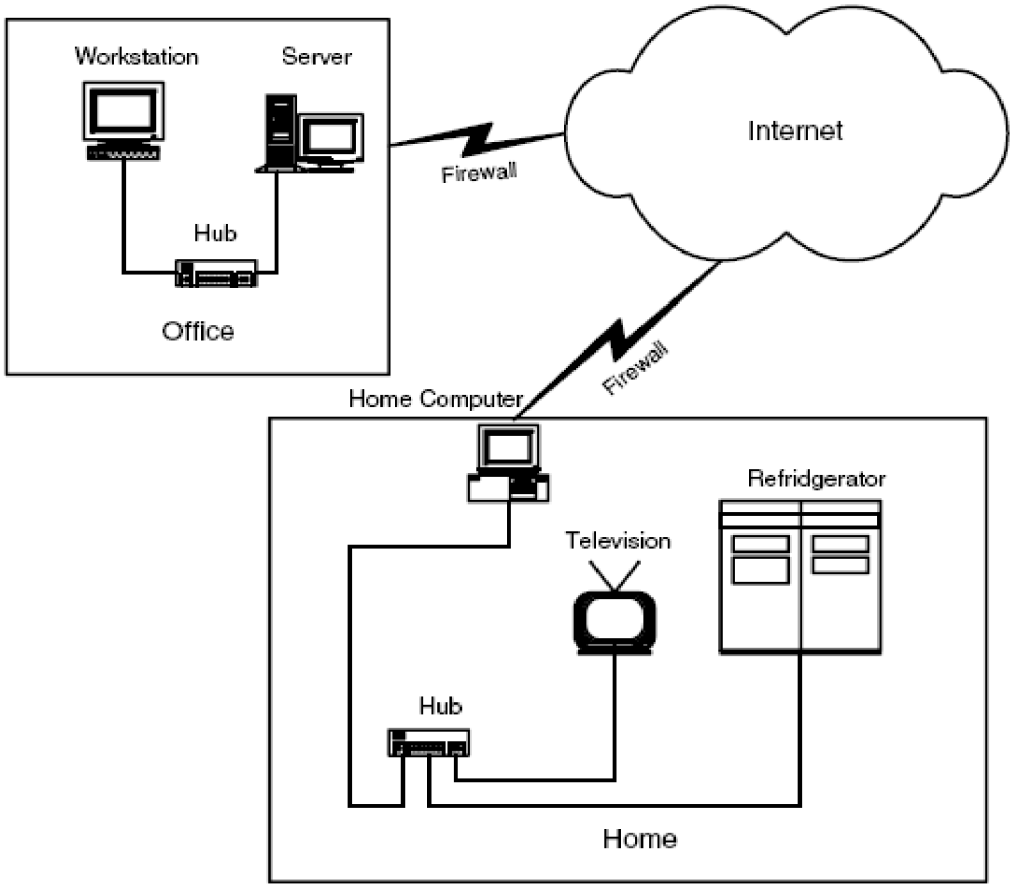
Slovem Embedded obecně označujeme počítačový systém nebo elektronické zařízení, které provádí vyhrazenou funkci nebo takové zařízení, které je navrženo pro potřeby konkrétní „Embedded" softwarové aplikace. Přestože existuje více či méně výstižný český ekvivalent anglického výrazu, budu pro účely přehlednosti a jednoznačnosti v této práci na místě přívlastku používat anglický termín Embedded.

Embedded systémy patří k nejrozšířenější variantě užívání počítačových systémů. Tyto systémy poskytují těm větším, jichž jsou součástí, potřebnou inteligenci. Příklady Embedded systémů najdeme od přenosného přehrávače hudby po užití v raketoplánu. Jedním z požadavků na tyto systémy je to, že musí být tzv. „autonomní" čili schopny plnit své funkce bez zásahu člověka v průběhu poměrně dlouhého časového intervalu. Proto je při vývoji hlavní důraz kladen na energetickou spotřebu, spolehlivost a robustnost. Autonomní činnost je nezbytná především tam, kde reakce člověka mohou být příliš pomalé, nedostatečně předvídatelné nebo nežádoucí. Dále například systémy, které pracují v reálném čase, musí být velmi rychlé při vykonávání daných funkcí. Další klíčovou charakteristikou většiny Embedded systémů je, že by měly být neviditelné (skryté), tj. uživatel by je neměl považovat za počítač.

2.1.2 Embedded systémy a Internet

Embedded systémy se zasloužily o vznik termínu „digital home" (*digitální domácnost*), který se vztahuje k digitalizaci všech forem zařízení spotřební elektroniky a jejich propojení do sítě. Dnes jsou již k dispozici základní prostředky například.: digitalizace médií, levné paměti, přenosná zařízení a rychlé sítě. Tato zařízení budou vybavena chytrým mikroprocesorem, který bude podporovat připojení k Internetu, dále by měla mít webový prohlížeč a obrazovku pro zobrazování informací, nástroje pro tvorbu programů, grafické programy atd., stejně jako osobní počítač. Uživatelé budou

moci kontaktovat zařízení vzdáleně pomocí Internetu, za účelem seřízení nastavení a podobně, tak jak to presentuje Obrázek 1. Více informací k tomuto tématu naleznete v [1] a [2].



Obrázek 1: *Monitorování vybavení domácnosti z kanceláře*

2.2 Síťové protokoly

V této kapitole budou stručně popsány Referenční model ISO/OSI model TCP/IP a technologie Ethernetu a nejčastěji využívané komunikační protokoly. Podrobnější informace lze najít především v [3] a [4], odkud jsem čerpal informace.

2.2.1 Referenční model ISO/OSI

Pro orientaci v oblasti síťových zařízení a technologií se často používá referenční model ISO/OSI, který byl vytvořen mezinárodní organizací ISO (*International Organization for Standardization*) jako jednotný standard pro vzájemné propojování různých systémů. Model se skládá ze sedmi vrstev a jeho schéma uvádí Tabulka **1**.

Fyzická vrstva

Úkol této vrstvy je zdánlivě velmi jednoduchý - zajistit přenos jednotlivých bitů mezi příjemcem a odesilatelem prostřednictvím fyzické přenosové cesty, kterou tato vrstva bezprostředně ovládá. K tomu je ovšem třeba vyřešit mnoho otázek převážně technického charakteru - např. jakou úrovní napětí bude reprezentována logická jednička a jakou logická nula, jak dlouho "trvá" jeden bit, kolik kontaktů a jaký tvar mají mít konektory kabelů, jaké signály jsou těmito kabely přenášeny, jaký je jejich význam, časový průběh apod.

Linková vrstva

Fyzická vrstva poskytuje jako své služby prostředky pro přenos jednotlivých bitů. Bezprostředně vyšší linková vrstva (někdy nazývaná též: spojová vrstva či vrstva datového spoje) pak má za úkol zajistit pomocí těchto služeb bezchybný přenos celých bloků dat (velikosti řádově stovek bytů), označovaných jako rámce (*frames*). Jelikož fyzická vrstva nijak neinterpretuje jednotlivé přenášené bity, je na linkové vrstvě, aby správně rozpoznala začátek a konec rámce, i jeho jednotlivé části.

Na přenosové cestě může docházet k nejrůznějším poruchám a rušení, v jejichž důsledku jsou přijaty jiné hodnoty bitů, než jaké byly původně vyslány. Jelikož fyzická vrstva se nezabývá významem jednotlivých bitů, rozpozná tento druh chyb až linková vrstva. Ta kontroluje celé rámce, zda byly přeneseny správně (podle různých kontrolních součtů). Odesilateli potvrzuje přijetí bezchybně přenesených rámců, zatímco v případě poškozených rámců si vyžádá jejich opětovné vyslání.

Síťová vrstva

Linková vrstva zajišťuje přenos celých rámců, ovšem pouze mezi dvěma uzly, mezi kterými vede přímé spojení. Co ale dělat, když spojení mezi příjemcem a odesilatelem není přímé, ale vede přes jeden či více mezilehlých uzlů? Pak musí nastoupit síťová vrstva, která zajistí potřebné směrování

(*routing*) přenášených rámců, označovaných nyní již jako pakety (*packets*). Síťová vrstva tedy zajišťuje volbu vhodné trasy resp. cesty (*route*) přes mezilehlé uzly, a také postupné předávání jednotlivých paketů po této trase od původního odesilatele až ke konečnému příjemci.

Síťová vrstva si tedy musí "uvědomovat" konkrétní topologii sítě (tj. způsob vzájemného přímého propojení jednotlivých uzlů).

Transportní vrstva

Síťová vrstva poskytuje bezprostředně vyšší vrstvě služby, zajišťující přenos paketů mezi libovolnými dvěma uzly sítě. Transportní vrstvu proto zcela odstiňuje od skutečné topologie sítě a vytváří jí tak iluzi, že každý uzel sítě má přímé spojení s kterýmkoli jiným uzlem sítě.

Transportní vrstvě díky tomu stačí zabývat se již jen komunikací koncových účastníků (tzv. end-to-end komunikací) - tedy komunikací mezi původním odesilatelem a konečným příjemcem.

Při odesílání dat zajišťuje transportní vrstva sestavování jednotlivých paketů, do kterých rozděluje přenášená data, a při příjmu je zase z paketů vyjímá a skládá do původního tvaru. Dokáže tak zajistit přenos libovolně velkých zpráv, přestože jednotlivé pakety mají omezenou velikost.

Relační vrstva

Úkolem této vrstvy je navazování, udržování a rušení relací (*sessions*) mezi koncovými účastníky. V rámci navazování relace si tato vrstva vyžádá na transportní vrstvě vytvoření spojení, prostřednictvím kterého pak probíhá komunikace mezi oběma účastníky relace. Pokud je třeba tuto komunikaci nějak řídit (např. určovat, kdo má kdy vysílat, nemohou-li to dělat oba účastníci současně), zajišťuje to právě tato vrstva, která má také na starosti vše, co je potřeba k ukončení relace a zrušení existujícího spojení.

Prezentační vrstva

Data, která se prostřednictvím sítě přenáší, mohou mít mj. povahu textů, čísel či obecnějších datových struktur. Jednotlivé uzlové počítače však mohou používat odlišnou vnitřní reprezentaci těchto dat - např. střediskové počítače firmy IBM používají znakový kód EBCDIC, zatímco většina ostatních pracuje s kódem ASCII. Podobně jeden počítač může zobrazovat celá čísla v doplňkovém kódu, zatímco jiný počítač v přímém kódu apod. - potřebné konverze přenášených dat má na starosti právě tato prezentační vrstva.

V rámci této vrstvy bývá také realizována případná komprese přenášených dat, eventuálně i jejich šifrování.

Aplikační vrstva

Koncoví uživatelé využívají počítačové sítě prostřednictvím nejrůznějších síťových aplikací - systémů elektronické pošty, přenosu souborů, vzdáleného přihlašování (*remote login*) apod. Začleňovat všechny tyto různorodé aplikace přímo do aplikační vrstvy by pro jejich velkou

různorodost nebylo rozumné. Proto se do aplikační vrstvy zahrnují jen části těchto aplikací, které realizují společné resp. obecně použitelné mechanismy.

model ISO/OSI	model TCP/IP
Aplikační	Aplikační
Prezentační	
Relační	
Transportní	Transportní
Síťová	Síťová
Linková	Síťové rozhraní
Fyzická	

Tabulka 1: *Referenční modely ISO/OSI a TCP/IP*

2.2.2 Model TCP/IP

Pro svůj úzký vztah k síti Internet je soustava protokolů TCP/IP někdy označována také jako Internet Protocol Suite (doslova: soustava protokolů Internetu), nebo též Department of Defense protocol suite (*DoD*).

Tvůrci protokolů TCP/IP vycházeli z předpokladu, že zajištění spolehlivosti je problémem koncových účastníků komunikace, a mělo by tedy být řešeno až na úrovni transportní vrstvy. Komunikační podsíť pak podle této představy nemusí ztrácet část své přenosové kapacity na zajišťování spolehlivosti (na potvrzování, opětné vysílání poškozených paketů atd.), a může ji naopak plně využít pro vlastní datový přenos. Komunikační podsíť tedy podle této představy nemusí být zcela spolehlivá - může v ní docházet ke ztrátám přenášených paketů, a to bez varování a bez snahy o nápravu. Komunikační síť by ovšem neměla zahazovat pakety bezdůvodně. Měla by naopak vyvíjet maximální snahu přenášené pakety doručit (v angličtině se v této souvislosti používá termín: best effort), a zahazovat pakety až tehdy, když je skutečně nemůže doručit - tedy např. když dojde k jejich poškození při přenosu, když pro ně není dostatek vyrovnávací paměti pro dočasné uložení, v případě výpadku spojení apod..

Na rozdíl od referenčního modelu ISO/OSI tedy TCP/IP předpokládá jednoduchou (ale rychlou) komunikační podsíť, ke které se připojují inteligentní hostitelské počítače.

Další odlišnost od referenčního modelu ISO/OSI spočívá v názoru na to, jak má komunikační síť vlastně fungovat. Zatímco model ISO/OSI počítá především se spojovaným přenosem - tedy s mechanismem virtuálních okruhů, TCP/IP naopak předpokládá nespojovaný charakter přenosu v komunikační podsíti - tedy jednoduchou datagramovou službou - což ostatně vyplývá i z představy co možná nejjednodušší komunikační podsítě.

Zatímco referenční model ISO/OSI vymezuje sedm vrstev síťového programového vybavení, TCP/IP počítá jen se čtyřmi vrstvami jak ukazuje Tabulka 1.

Vrstva síťového rozhraní

Nejnižší vrstva, vrstva síťového rozhraní (Network Interface Layer) (někdy též: linková vrstva resp. Link Layer) má na starosti vše, co je spojeno s ovládáním konkrétní přenosové cesty resp. sítě, a s přímým vysíláním a příjmem datových paketů. V rámci soustavy TCP/IP není tato vrstva blíže specifikována, neboť je závislá na použité přenosové technologii.

Vrstvu síťového rozhraní může tvořit relativně jednoduchý ovladač (device driver), je-li daný uzel přímo připojen například k lokální síti či ke dvoubodovému spoji, nebo může tato vrstva představovat naopak velmi složitý subsystém, s vlastním linkovým přenosovým protokolem (např. HDLC apod.). Vzhledem k velmi častému připojování jednotlivých uzlů na lokální sítě typu Ethernet je vrstva síťového rozhraní v rámci TCP/IP často označována také jako Ethernetová vrstva (*Ethernet Layer*).

Síťová vrstva

Bezprostředně vyšší vrstva, která již není závislá na konkrétní přenosové technologii, je vrstva síťová, v terminologii TCP/IP označovaná jako Internet Layer (volněji: vrstva vzájemného propojení sítí), nebo též IP vrstva (IP Layer) podle toho, že je realizována pomocí protokolu IP. Úkol této vrstvy je v prvním přiblížení stejný, jako úkol síťové vrstvy v referenčním modelu ISO/OSI - stará se o to, aby se jednotlivé pakety dostaly od odesilatele až ke svému skutečnému příjemci, přes případné směrovače resp. brány. Vzhledem k nespojovanému charakteru přenosů v TCP/IP je na úrovni této vrstvy zajišťována jednoduchá (tj. nespolehlivá) datagramová služba.

Transportní vrstva

Třetí vrstva TCP/IP je označována jako transportní vrstva (*Transport Layer*), nebo též jako TCP vrstva (*TCP Layer*), neboť je nejčastěji realizována právě protokolem TCP (*Transmission Control Protocol*). Hlavním úkolem této vrstvy je zajistit přenos mezi dvěma koncovými účastníky, kterými jsou v případě TCP/IP přímo aplikační programy (jako entity bezprostředně vyšší vrstvy). Podle jejich nároků a požadavků může transportní vrstva regulovat tok dat oběma směry, zajišťovat spolehlivost přenosu, a také měnit nespojovaný charakter přenosu (v síťové vrstvě) na spojovaný.

Přestože je transportní vrstva TCP/IP nejčastěji zajišťována právě protokolem TCP, není to zdaleka jediná možnost. Dalším používaným protokolem na úrovni transportní vrstvy je například protokol UDP (*User Datagram Protocol*), který na rozdíl od TCP nezajišťuje mj. spolehlivost přenosu - samozřejmě pro takové aplikace, které si to (na úrovni transportní vrstvy) nepřejí.

Aplikační vrstva

Nejvyšší vrstvou TCP/IP je pak vrstva aplikační (*Application Layer*). Jejími entitami jsou jednotlivé aplikační programy, které na rozdíl od referenčního modelu ISO/OSI komunikují přímo s transportní vrstvou. Případné prezentační a relační služby, které v modelu ISO/OSI zajišťují samostatné vrstvy, si zde musí jednotlivé aplikace v případě potřeby realizovat samy.

2.2.3 Ethernet

Ethernet je přenosová technologie. Zajišťuje skutečný přenos dat. V referenčním modelu ISO/OSI pokrývá fyzickou a linkovou vrstvu. V rámci TCP/IP spadá do vrstvy síťového rozhraní.

Standard IEEE 802.3 definuje lineární síťovou architekturu založenou na všesměrovém vysílání (oběžnících), které k síťovému médiu přistupuje způsobem definovaným pomocí CSMA/CD (*Carrier Sense Multiple Acces with Colision Detection*), což je metoda vícenásobného přístupu s detekcí nosné a detekcí kolize. Zařízení, která se řídí touto metodou spolu soupeří o právo přenášet data, protože nemohou přenášet data současně.

Princip metody CSMA/CD lze shrnout do těchto kroků:

- Stanice, která chce vyslat data, testuje stav kanálu.
- Je-li kanál volný, zahájí vysílání. Přitom však musí zaručit dodržení mezirámcové mezery 9,6 miliontin sekundy.
- Je-li kanál obsazen, čeká na jeho uvolnění. Poté ihned po uplynutí mezirámcové mezery zahájí vysílání.
- Během přenosu testuje, zda napěťové úrovně signálu na přenosovém médiu odpovídají očekávaným hodnotám, případně sleduje i jiné parametry vysílaných dat. Je-li během celé doby vysílání signál správný, je považováno na této úrovni vysílání dat za úspěšně ukončené.
- Zjistí-li však stanice nesouhlas, znamená to, že došlo ke kolizi se současným vysíláním další stanice. Proto ihned přeruší přenos rámce a vyšle speciální rušící signál (*jamming signal*), aby kolizi bezpečně rozeznaly i ostatní zúčastněné stanice.
- Po odeslání rušícího signálu čeká stanice po určitou dobu (*backoff*) a poté pokusí opakovaně vysílat jak je popsáno v prvním kroku. Je nutné náhodné stanovení doby čekání po zjištění kolizí aby opakované vysílání nezahájily stanice zúčastněné na kolizi ve stejnou dobu. Pokud rámec není odeslán ani po 16-ti pokusech, je hlášena chyba.

2.2.4 Protokol ARP

Protokol ARP (*Address Resolution Protocol*) je součástí síťové vrstvy modelu ISO/OSI a vrstvy Internet modelu TCP/IP. Protokol převádí logickou adresu síťové vrstvy na hardwarovou adresu (MAC adresu síťového zařízení).

Když se zařízení pokusí o navázání relace s aplikací umístěnou na vzdáleném počítači, může použít buď jeho název nebo logickou adresu síťové vrstvy. Název počítače musí být přeložen na adresu síťové vrstvy a ta následně na hardwarovou adresu linkové vrstvy. Pokud je při navazování relace použita logická adresa síťové vrstvy, odpadá nutnost provést překlad adresy. Vzdálená strana však stále musí přeložit logickou adresu síťové vrstvy na MAC adresu síťové karty, aby bylo možné datagramy správně doručit. Protokol ARP, definovaný v [8], zajišťuje dynamický překlad v této poslední fázi přenosu datagramu. Hlavička ARP je dlouhá 28 bytů. Obrázek 2 ukazuje její formát.

0 31

Hardware Type		Protokol Type
HLEN	PLEN	Operation
Sender HA		
Sender HA		Sender IP
Sender IP		Target HA
Target HA		
Target IP		

Obrázek 2: *Hlavička protokolu ARP*

Pole Hardware Type specifikuje linkový protokol používaný na LAN. Protocol Type specifikuje typ síťového protokolu . Pole HLEN určuje délku linkové adresy a pole PLEN délku síťové adresy. Standardně je tedy HLEN = 6 a PLEN = 4. Pole Operation určuje o jakou operaci jde. Žádost (*ARP request*) má hodnotu 1 a odpověď (*ARP reply*) má hodnotu 2. Pak již následuje linková adresa odesilatele, IP-adresa odesilatele, linková adresa příjemce (v dotazu vyplněna nulami) a IP-adresa příjemce.

2.2.5 Internet Protokol

Protokol IP (Internet Protokol) odpovídá vrstvě Internet modelu TCP/IP a Síťové vrstvě modelu ISO/OSI. Protokol IP nabízí nespolehlivé, nespojované doručování datagramů. Z této definice vyplývá, že IP nezaručuje, že datagram IP dorazí do cílového místa. Datagram je odeslán a očekává se, že dorazí do svého cíle. Protokol IP k němu přidá logické adresy síťové vrstvy odesílajícího a cílového počítače a při doručování spoléhá na protokoly jiných vrstev, které mají zajistit úspěšné doručení.

Pokud dojde během doručování k problému, spoléhá IP na protokol ICMP, který má při výskytu chyby odeslat do zdroje zprávu s detailním popisem chyby (datagram je při nemožnosti dalšího doručení zahozen). Spolehlivost přenosu mají zajistit protokoly vyšších vrstev, například TCP.

Základní funkcí protokolu IP je logické adresování na úrovní síťové vrstvy a doručení informací (ve formě datagramů) mezi počítači. Adresování IP

Protokol IP dále provádí například fragmentaci datagramů a jejich zpětné skládání, pokud velikost původního datagramu přesahuje přípustnou hranici, takže musí dojít k jeho rozdělení na menší díly. Protože IP patří mezi nespojované protokoly, nevyžaduje navazování spojení mezi oběma koncovými počítači. Nezajišťuje doručování v pevně daném sledu, potvrzování příjmu ani nijak jinak neřídí tok datagramů po síti mezi koncovými počítači. Každý datagram je chápán jako samostatná entita. IP pouze provede adresaci datagramu a poté ho odešle, dál se o jeho doručení nezajímá.

Protokol IP přijímá proud dat od protokolů UDP a TCP, rozděluje tyto informace do bloků, každý blok doplní adresou a připraví do podoby datagramu, který poté může být odeslán po síti na cílový počítač. Směrovače a směrovací protokoly se starají o výběr cesty, kterou datagram mezi odesílatelem a cílovým počítačem putuje.

Obrázek 3 obsahuje formát datagramu IP podle definice [5]. Hlavička protokolu IP sestává nejméně ze 20 bytů, pokud neobsahuje přídavné volby.

0 31

Version	IHL	Type of Service	Total Length	
Identification			Flags	Gragment Offset
TTL		Protocol	Header Checksum	
Source Adressa				
Destination Adress				
Options				Padding

Obrázek 3: Hlavička protokolu IP

2.2.6 Protokol UDP

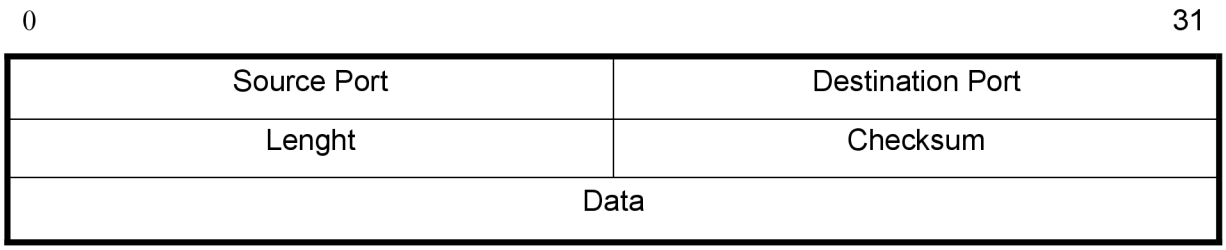
Protokol UDP (*User Datagram Protocol*) [7] entitám (procesům, úlohám, programům) aplikační vrstvy zpřístupňuje nespolehlivé a nespojované, zato ale rychlé a efektivní přenosové služby síťové vrstvy (protokolu IP). Můžeme si jej představit jako jednoduchou "obálku" nad protokolem IP, která nijak nemění povahu ani kvalitu jeho přenosových služeb, ale pouze je zprostředkovává své bezprostředně vyšší vrstvě. V podstatě jediné, co UDP zajišťuje navíc, je multiplexování a demultiplexování datagramů (viz minulý díl seriálu), tedy rozlišování různých příjemců, resp. odesilatelů v rámci téhož hostitelského počítače - podle čísla portu.

Každý aplikační program, který se rozhodne používat transportní služby protokolu UDP, tak na sebe přebírá odpovědnost za zajištění takové úrovně spolehlivosti přenosů, jakou sám potřebuje. Sám se také musí vyrovnávat i s dalšími důsledky, které vyplývají z nespolehlivého a nespojovaného

charakteru přenosových služeb protokolu UDP - jako je např. zajišťování správného pořadí doručovaných datagramů, eliminace duplicitních datagramů apod.

Protokol UDP dostává od své bezprostředně vyšší vrstvy bloky dat, které se snaží vkládat celé do jednotlivých datagramů síťové vrstvy (*IP datagramů*). Proto se také těmto blokům na úrovni transportní vrstvy říká uživatelské datagramy (*user datagrams*), nebo též UDP datagramy (*UDP datagrams*).

Hlavičku UDP datagramu tvoří čtyři položky v rozsahu 16 bitů, které po řadě vyjadřují číslo portu odesílatele a příjemce, délku UDP datagramu, a kontrolní součet - viz Obrázek 4.



Obrázek 4: *Hlavička UDP datagramu.*

2.2.7 Protokol TCP

Protokol TCP (*Transmission Control Protocol*) [6] je protokolem transportní vrstvy, a je jedním ze dvou alternativních protokolů, které síťový model TCP/IP na úrovni této vrstvy nabízí. Narozdíl od protokolu UDP nabízí TCP na úrovni transportní vrstvy pomalejší, zato ale spolehlivé služby.

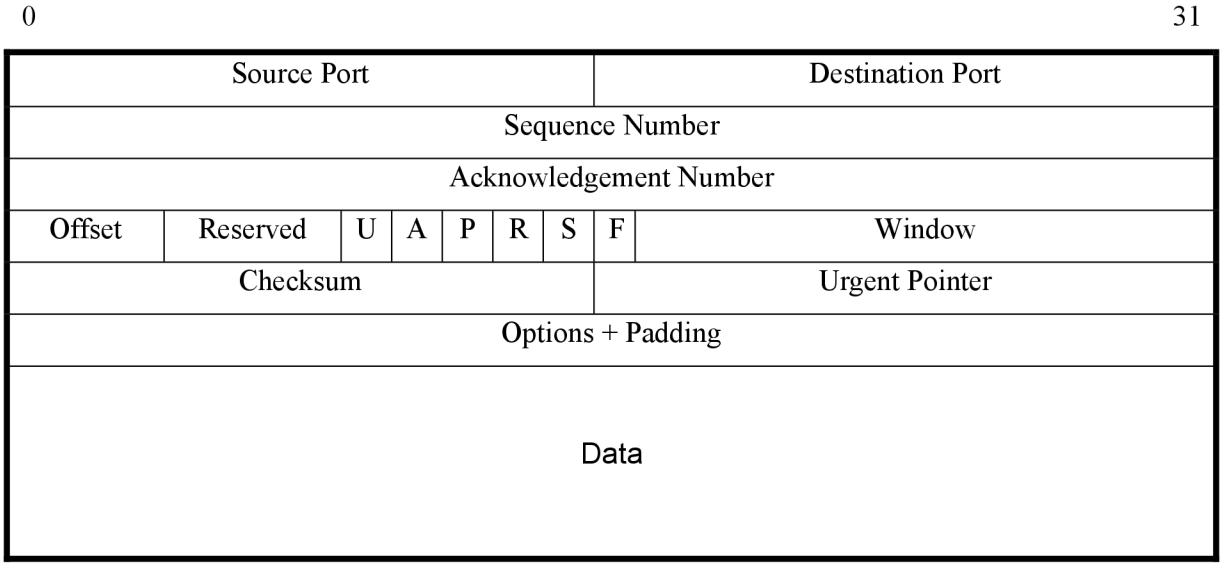
Další významný rozdíl mezi protokoly UDP a TCP spočívá v tom, že zatímco UDP nabízí nespojované (*connectionless*) služby, protokol TCP nabízí své přenosové služby jako spojované (*connection-oriented*). Před každou výměnou dat mezi dvěma uzly tedy musí být nejprve navázáno spojení, a po skončení přenosu musí být toto spojení zase zrušeno.

Protokol TCP se snaží nabízet své bezprostředně vyšší vrstvě přenos jednotlivých bytů. Očekává tedy, že entity aplikační vrstvy mu na straně odesilatele budou postupně předávat jednotlivé byty (přesněji: osmibitové oktety), a na straně příjemce si je zase budou po jednom vyzvedávat. Ve skutečnosti nejsou jednotlivé bity přenášeny každý zvlášť. Protokol TCP na straně odesilatele postupně akumuluje jednotlivé byty do vhodné vyrovnávací paměti (*bufferu*), a po jejím naplnění odešle celý její obsah najednou - v tzv. segmentu, jak se nazývá blok, přenášený protokolem TCP. Analogicky na straně příjemce, kde se datový obsah segmentu ukládá do vyrovnávací paměti, a jednotlivé byty jsou entitám aplikační vrstvy poskytovány z této vyrovnávací paměti. Celý mechanismus sdružování jednotlivých bytů do bloků je plně v režii protokolu TCP, který se přenosem větších celků snaží optimalizovat využití přenosových cest. Pro vyšší vrstvu je tento mechanismus neviditelný - vyšší vrstva pracuje s představou proudu jednotlivých bytů.

Protokol TCP používá tzv. kladné potvrzování (*positive acknowledgement*), což znamená, že potvrzuje úspěšně přijatá data, a na chybně přijatá data nereaguje nijak (ty pak odesilatel znovu vyšle na základě vypršení časového limitu, ve kterém očekával jejich kladné potvrzení). Ve své základní

podobě tzv. jednotlivého kladného potvrzování (*stop&wait positive acknowledgement*), kdy odesilatel před odesláním každého dalšího bloku čeká na kladné potvrzení naposledy vyslaného bloku, je tento mechanismus značně neefektivní. Protokol TCP proto používá kladné potvrzování ve variantě tzv. kontinuálního potvrzování (*continuous acknowledgement*), známého též jako tzv. metoda okénka (*sliding window*). Podstata této varianty spočívá v tom, že odesilatel může odeslat další blok (resp. několik dalších bloků) ještě dříve, než dostane potvrzení o přijetí bloku předchozího. O tom, kolik bloků může takto vyslat "dopředu", rozhoduje velikost pomyslného okénka.

Protokol TCP přenášené bloky dat označuje jako segmenty, a mohou mít různou délku. V případě, že některý segment není přenesen bezchybně (resp. není kladně potvrzen do vypršení časového limitu), je jeho obsah vyslán znovu, a po něm také obsah všech následujících segmentů. Jde tedy o tzv. opakování s návratem (*Go-Back-N*). V protokolu TCP potvrzovanými jednotkami dat nikoli celé bloky (segmenty), ale jednotlivé byty (přesněji oktety).



Obrázek 5: *Formát hlavičky TCP, jejíž délka je obvykle 20 bytů, pokud nejsou použity žádné volby. Hlavička nemusí obsahovat žádné volby ani data.*

2.2.8 Protokol HTTP

Obecné informace o protokolu

HTTP (*Hypertext Transfer Protocol*) je bezestavový protokol aplikační úrovně. Využívá služeb protokolů TCP/IP, ale obecně je požadován pouze protokol poskytující spojované služby. Jeho nejčastější použití je přenos dokumentů (HTML stránek, obrázků) v síti WWW. Protokol je nezávislý na přenášených datech a lze ho tedy využít i jinde. Detailní popis je možné nalézt v [9].

První verze protokolu pojmenovaná HTTP/0.9 se v Internetu začala používat od roku 1990. Byl to jednoduchý protokol pro přenos holých dat přes Internet. V další verzi HTTP/1.0 byl protokol rozšířen hlavně o přenos *hlaviček*. Jsou to vlastně krátké zprávy ve formátu podle tehdy již existujícího standardu MIME (*Multipurpose Internet Mail Extensions*), které obsahují dodatečné in-

formace o přenášených datech — například datum poslední změny dokumentu, délku, použité kódování, informace o serveru nebo klientském programu.

Poslední verze HTTP/1.1 je proti předchozím v mnohém rozšířena. Podporuje mimo jiné přenos více dokumentů při jednom transportním spojení, automatický výběr typu dokumentu, jeho jazyka a kódování a také přenos vybrané části dokumentu. Zajímavou vlastností je podpora více virtuálních serverů na jedné síťové adrese. Tyto novinky byly k dispozici již ve verzi HTTP/1.0, ale pouze jako volitelná rozšíření. Ne ve všech implementacích byly tedy podporovány.

Každá verze protokolu je vždy zpětně kompatibilní s předchozími verzemi. Poslední verzi podporuje například klientský program Microsoft Internet Explorer 6. Nejznámějším serverem je Apache.

Protokol je typu požadavek/odpověď (*klient/server*). Každá zpráva (požadavek i odpověď) se skládá ze záhlaví a těla zprávy. Klient posílá na server požadavek obsahující tzv. *metodu,* URI (*Uniform Resource Identifier*), verzi protokolu, hlavičky a případně data v těle požadavku. Server odpovídá verzí protokolu a stavovým kódem. Následují hlavičky s informacemi o dokumentu a o serveru a na konci je tělo odpovědi obsahující většinou přenášený dokument.

Informace jsou přenášeny jako obyčejné textové řetězce. Výjimkou je tělo s dokumentem, což jsou obecně binární data, která mohou být navíc kódována. Nejdůležitější informace, jako jsou metoda, URI, kód chyby a verze protokolu, se nacházejí v první řádku záhlaví požadavku/odpovědi. Dalšími položkami záhlaví jsou hlavičky. Tělo zprávy je od záhlaví odděleno prázdným řádkem.

Stavové kódy

Stavový kód informuje klienta většinou o úspěšném zpracování požadavku a o chybách, které mohly nastat při vyřizování požadavku. Stavový kód tvoří trojciferné číslo doplněné o krátký text popisující význam kódu.

Kódy jsou rozděleny podle první číslice do pěti tříd. Kód začínající číslicí 1 je pouze informačního charakteru. Kód začínající číslicí 2 je kód úspěchu. Klient se tak dozví, že jeho požadavek byl úspěšně přijat a zpracován. Číslicí 3 začíná kód zprávy, která informuje klienta, že musí uskutečnit další akci, aby se úspěšně vyřídil jeho požadavek. V další třídě jsou kódy začínající číslicí 4. Server tak odpovídá na chybný dotaz klienta nebo na požadavek, který nemůže být splněn. Poslední skupinou jsou kódy začínající číslicí 5. Ty posílá server v případě, že při plnění zřejmě platného požadavku dojde na straně serveru k chybě.

Konkrétní stavové kódy jsou uvedeny v [9] i s jejich definicí. Klient není povinen rozumět všem stavovým kódům, ale musí rozumět alespoň třídě, do které kód patří.

Metody protokolu HTTP

Základní definované metody jsou GET a HEAD. Přeje-li si klient získat ze serveru nějaký dokument, pošle v požadavku právě metodu GET následovanou jednoznačným identifikátorem zdroje (*URI*) -

obvykle se jedná o jméno souboru na serveru. Existuje-li požadovaný dokument, server ho klientovi pošle. Kromě toho obsahuje odpověď ještě hlavičky s dalšími informacemi o dokumentu. Podmíněnou metodou GET se rozumí zpráva, která navíc obsahuje v záhlaví podmínku přenosu dokumentu (například hlavičky *If-Modified-Since* nebo *If-None-Match*). Dokument se přenáší jen při splnění uvedených podmínek, čímž se omezí použití sítě. Ze stejných důvodů je zavedena tzv. částečná metoda GET, která obsahuje hlavičku Range. Tím je umožněno přenést pouze část dokumentu.

Metoda HEAD vypadá podobně jako metoda GET. Server na tento požadavek posílá tutéž odpověď, jakou by poslal v případě požadavku GET s tím rozdílem, že zpráva obsahuje pouze záhlaví (neposílá se tělo s dokumentem). Ke klientovi se tedy dostanou jen hlavičky obsahující informace o dokumentu. Dá se tak i například zjistit, jestli se dokument od posledního přenosu změnil. V případě že ne, nemusí se žádný přenos dokumentu uskutečňovat a stačí například použít jeho dočasnou kopii uloženou v lokální vyrovnávací paměti. Metoda se také často používá k testování platnosti a dostupnosti hypertextových odkazů. Kompletní seznam metod s jejich popisem lze najít v [3] nebo [9].

2.3 Mikrokontrolér Atmel ATmega128

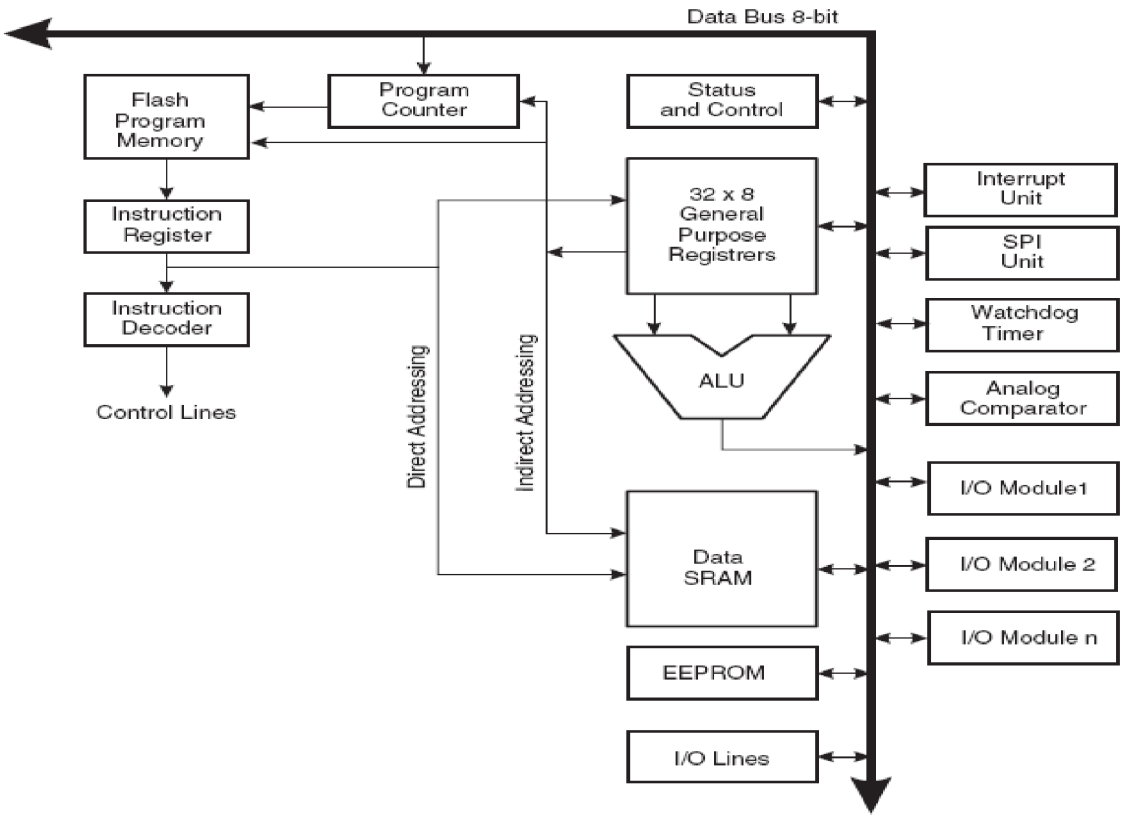
2.3.1 Obecná charakteristika mikrokontroléru

Atmel ATmega128 reprezentuje rodinu výkonných nízkopříkonových 8-bitových mikrokontrolérů CMOS, založených na vylepšené architektuře AVR RISC – viz. Obrázek 6. Mikrokontroléry ATmega128 jsou schopné vykonat většinu ze svých 133 instrukcí v jediném hodinovém cyklu, to znamená navýšení jejich výkonu.

Jádro AVR kombinuje bohatou instrukční sadu s 32 univerzálními pracovními registry. Všechny tyto registry jsou přímo spojeny s Aritmeticko Logickou Jednotkou (ALU), to umožňuje během vykonávání jedné instrukce přístup ke dvěma nezávislým registrům současně v jednom hodinovém cyklu. Výsledná architektura je tak schopna dosahovat mnohem vyššího výkonu než klasické CISC mikrokontroléry.

Mikrokontroléry AVR využívají koncepci Harwardské architektury, což znamená oddělenou paměť pro program a pro data. Program umístěný v programové paměti je prováděn s jednoduchým překrýváním instrukcí (*pipeline*). Zatímco jedna instrukce je prováděna, druhá je přesouvána Z programové paměti.

ATmega128 nabízí tyto parametry: 128KB systémově programovatelné flash paměti, která umožňuje čtení během zápisu, 4KB EEPROM, 4KB SRAM, 53 univerzálních I/O linek, 32 univerzálních pracovních registrů, Real Time Counter (*RTC*), čtyři flexibilní časovače/čítače s komparačními režimy a PWM, 2 x USART, dvoudrátové sériové rozhraní, 8-mi kanálový 10-ti bitový ADC s volitelnými stupni diferenčních vstupů s programovatelným ziskem, programovatelný Watchdog časovač s interním oscilátorem, port SPI, rozhraní JTAG odpovídající standardu IEEE 1149.1 a 6 softwarově řízených úsporných režimů.



Obrázek 6: *Blokové schéma architektury AVR [12]*

2.3.2 Přerušovací systém

Při provádění relativních skoků či instrukcí volání je přímo přístupný adresový prostor. Většina AVR instrukcí má formát jednoho 16 bitového slova. Každá adresa programové paměti obsahuje 16 nebo 32 bitovou instrukci. Při provádění obsluhy přerušení a volání podprogramu se návratová adresa programového čítače (*Program Counter*) ukládá do zásobníku. Zásobník je umístěn v datové paměti SRAM a tudíž je omezen jenom velikostí paměti SRAM a jejím volným místem. Všechny uživatelské programy se musí inicializovat v inicializační (*reset*) části programu, před prováděním podprogramů nebo obsluhy přerušení. Šestnácti bitový ukazatel zásobníku je přístupný pro čtení i zápis v I/O prostoru.

Systém přerušení má vlastní řídicí registry umístěné v I/O prostoru a navíc bit ve stavovém registru pro zákaz/povolení všech přerušení. Všechna různá přerušení mají oddělený vektor přerušení v tabulce vektorů přerušení umístěné na začátku programové paměti. Priorita těchto přerušení je dána umístěním jejich vektorů V tabulce přerušení. Čím nižší má vektor přerušení adresu, tím větší má prioritu.

2.3.3 Paměťový prostor

Tato sekce popisuje dostupný paměťový prostor v čipu ATmega128. Architektura AVR má k dispozici dva hlavní paměťové prostory, datovou a programovou paměť. Navíc má ATmega ještě 4kB paměť EEPROM pro ukládání konfiguračních nebo jiných dat. Celkem jsou tedy k dispozici tři paměťové prostory přímo na čipu.

Flash paměť

ATmega128 má k dispozici 128kB na čipu je to systémově programovatelná flash paměť pro uchování programu. Protože všechny instrukce AVR jsou široké 16 nebo 32 bitů, paměť je organizována jako 64k x 16. Pro bezpečnost softwaru, je tato paměť rozdělena na dvě sekce, bootovací a programovou sekci. Až 8 kB paměti FLASH lze využít pro zavaděč (*bootloader*) mikroprocesoru.

Paměť vydrží přinejmenším 10000 zápisových nebo mazacích cyklů. Programový čítač ATmega 128 má šířku 16 bitů, pro adresování 64k adres programu. Paměť lze programovat pomocí rozhraní SPI nebo JTAG.

SRAM datová paměť

ATmega128 poskytuje dvě různé konfigurace datové paměti SRAM, které ilustruje Tabulka ***2***.

Konfigurace	Interní datová paměť SRAM	Externí datová paměť SRAM
Normální mód	4096	do 64k
Mód kompatibilní s ATmega103	4000	do 64k

Tabulka 2: *Konfigurace paměti SRAM [12]*

ATmega128 je komplexní mikrokontrolér s mnoha periferními jednotkami, které jsou podporovány na 64 adresách rezervovaných pro operační kód pro I/O instrukce. Pro rozšířený I/O prostor, který je v rozsahu adres $60 - $FF paměti SRAM, pro něj mohou být využity pouze instrukce ST/STS/STD a LD/LDS/LDD.

Prvních 4352 adres datové paměti pro registry, I/O paměť, rozšířenou I/O paměť a interní datovou SRAM. Prvních 32 je rezervováno pro soubor registrů, dalších 64 pro standardní I/O paměť, pak 160 adres pro rozšířenou I/O paměť a 4096 adres pro interní data SRAM.

U mikrokontroléru ATmega128 je možno použít i externí paměť SRAM. Tato paměť bude využívat zbytek 64K adresového prostoru. Tento prostor je mapován hned za interní SRAM. Soubor registrů, vstupně/výstupní, rozšířený vstupně/výstupní a interní SRAM jsou mapovány na prvních 4352 bytů, je-li tedy užit 64KB (65536 bytů) externí paměti, je k dispozici 61184 bytů externí paměti.

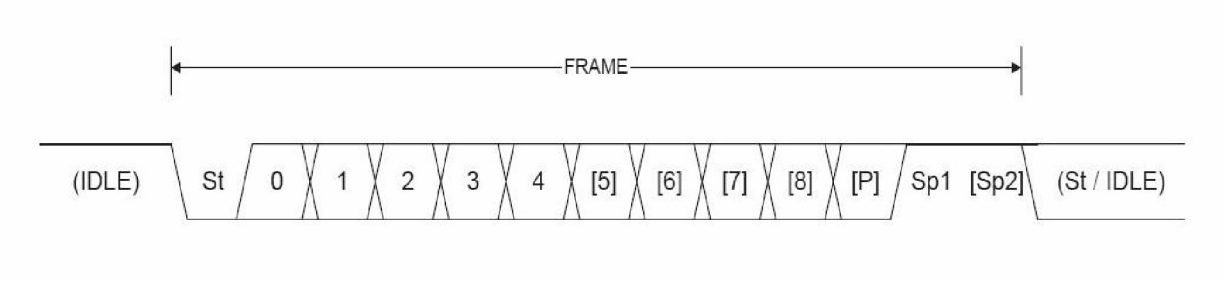
2.3.4 USART

USART (*Univerzal Synchronous and Asynchronous Reciever andTransmitter*) je obvod schopný obousměrné komunikace rychlostmi jednotek až desítek baud. Umožňuje jak synchronní, tak asynchronní přenos s možností nastavení 5- 9 datových bitů, nastavení počtu stop-bitů a parity. Umožňuje také multiprocesorovou komunikaci.

Já používám toto rozhraní pro komunikaci s PC přes RS232. Mnozí výrobci uvedli před lety na trh převodníky RS232/USB, takže není problém při komunikaci přes USB rozhraní. To se hodí zvláště při napojení na notebook.

Jednotka USART je složena ze tří hlavních bloků. Z vysílače, přijímače a generátoru hodin. Já používám toto rozhraní pro komunikaci s PC přes RS232, pin XCK a obvod synchronizace z vnějších hodin je tedy nevyužit. Pro nastavení požadované bitové rychlosti slouží 16-bitový registr UBRR. Detaily jsou v [12].

Vysílač obsahuje posuvný registr do nějž se zápisem datového registru UDR vloží vysílaná data, která jsou dále automaticky zpracována. Jakmile je UDR připraven pro další zápis dat, je vyvoláno přerušení. Podle požadavku je vypočtena sudá nebo lichá parita. Řídící logika pak v rytmu hodin generátoru bitové rychlosti vysílá jednotlivé bity doplněné o start, paritu a stop bity přes výstupní budič na pin TxD. Jakmile je přenos dokončen, je vyvoláno další přerušení. Rámec sériového přenosu presentuje Obrázek 7, kde ST značí start-bit, 0-4 povinné datové bity, 5-9 volitelné datové bity, Sp1 a Sp2 povinný a volitelný stop-bit a IDLE je klidový stav na lince.



***Obrázek 7**: Rámec sériového přenosu [12]*

Přijímač přijímá sériový tok bitů pinem RxD. Nejprve se provádí filtrace za účelem odstranění šumu a rušení a regenerace tvaru signálu. Rekonstruovaným hodinovým signálem se řídí přijímací posuvný registr, který postupně střádá datové bity. Z přijatých dat se spočítá a překontroluje parita. Po dokončení příjmu slova je vyvoláno přerušení a přijatá data lze přečíst z registru UDR.

Další nastavení USARTu a čteni stavových informací se provádí přes tři 8-bitové registry UCSRA, UCSRB a UCSRC. Kompletní popis je v [10] nebo [12].

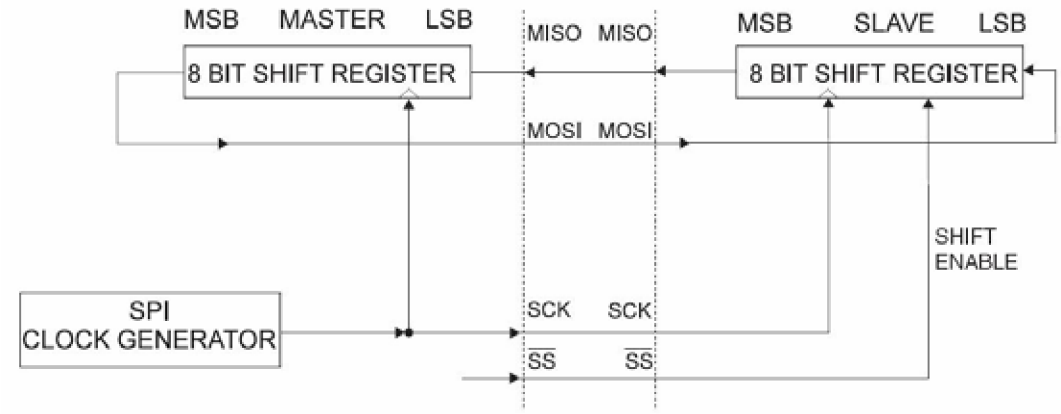
2.3.5 Způsoby programování

Programová paměť mikrokontroléru je „In System Programmable Flash memory". Znamená to, že kromě klasického, paralelního naprogramování této paměti, je možné i sériové naprogramování přímo

v systému. Při paralelním programování, které se používá u většiny jednočipových mikropočítačů či mikrokontrolérů se využívá toho, že příslušný obvod je navržen tak, aby po připojení programovacího napětí na určitý vývod obvodu se provedlo přepnutí vývodů, které jsou v normálním režimu vývody I/O portů, tak, že nyní jsou tyto vývody obvodu připojeny k adresovým a datovým vývodům vnitřní programové paměti, takže lze do této paměti paralelně zaznamenat data, což je vlastně příslušný program. Po naplnění této paměti snížení napětí na vývodu umožňujícím přepínaní do/z programovacího módu a uvedení procesoru do počátečního stavu (resetování) pak mikropočítač pracuje podle právě naprogramovaného programu. Je zřejmé, že při tomto způsobu programování je třeba, aby při programování byl k I/O vývodům připojen programátor a naopak aby byly odpojeny od jakýchkoli jiných obvodů, např. periferií. Proto je při každém programování potřeba obvod vyjmout z objímky, popř. vyletovat z plošného spoje a vložit do programátoru. Tato nevýhoda odpadá při sériovém programování, kdy mikrokontrolér zůstává v aplikaci a pomoci několika signálů připojených k programátoru se dá jednoduše naprogramovat.

Serial Peripheral Interface (SPI)

Rozhraní SPI (*Serial Peripheral Interface*) poskytuje vysokorychlostní synchronní přenos dat mezi mikrokontrolérem a okolními prostředky nebo dvěma mikrokontroléry. Významná funkce ATmega128 v SPI režimu je synchronní datový přenos na třech linkách, který řídí buď master nebo salave. Já jsem se rozhodl toto rozhraní použít pro účely přenosu dat mezi mikrokontrolérem a MMC kartou.



Obrázek 8: *Propojení Master – Slave přes SPI rozhraní [12]*

Obrázek 8 ukazuje propojení master a slave zařízení přes SPI rozhraní. Obě zařízení mají osmi bitový posuvný registr, ve kterém připravují data pro komunikaci. Samotnou komunikaci řídí master generováním hodinového signálu SCK. Data z registrů jsou postupně posílána druhé straně, než je poslán celý byte. Po poslání paketu master synchronizuje slave nastavením signálu SS (Slave Select) na logickou jedničku.

V konfiguraci mikrokontroléru jako master, musí nejdříve program zaručit řízení signálu SS. Potom se přenese byte dat do SPDR (*SPI Data Register*) a pošle se zařízení slave. Po poslání bytu se

zastaví generátor hodinového taktu a nastaví se příznak ukončení transakce SPIF (*SPI Interrupt Flag*) v SPSR (*SPI Status Register*). Jak je nastavený SPIE (*SPI Interrupt Enable*) bit v SPCR (*SPI Kontrol Register*), bylo vyvolané přerušení. Master potom může pokračovat posíláním dalšího bytu, nebo oznámit konec nastavením signálu SS.

Pokud jde o slave zařízení, tak musí čekat dokud nedojde k nastavení SS na logickou nulu. Pak je možné posílat data z SPDR. Po jednom bytu se nastaví příznak ukončení přenosu a po nastavení SPIE bitu je opět možné přenášet nová data.

Rozhraní najdeme na vývodech portu B (PB0-PB3). Další informace nejdete v [12] nebo [10].

JTAG

JTAG je zkratka pro Joint Test Action Group, což je skupina výrobců integrovaných obvodů a jejich prodejců, usilující o dosažení dohody, která by stanovila princip konstrukce integrovaných obvodů. Jimi navrhnutá norma IEEE 1149.1 definuje rozhraní integrovaného obvodu a způsob komunikace na tomto rozhraní. Definuje tedy testovací logické obvody které mohou být zařazeny do integrovaných obvodů za účelem testování propojení mezi jednotlivými komponentami na již osazených deskách, testování integrovaného obvodu samotného a sledování integrovaného obvodu při jeho normální činnosti.

Já v tomto projektu využívám rozhraní JTAG k programování a verifikaci FLASH paměti a FUSE bitů procesoru. Dále ho lze použít pro programování a verifikaci paměti EEPROM a LOCK bitů. Pro přenos kódu do paměti jsem využil rychlý USB programátor PRESTO a jeho software JTAG Player od firmy ASIX. Aby bylo možné data přenést, je potřeba použít na WinAVR vygenerovaný kód ještě další program AVRsvf. Je to program spouštěný z příkazové řádky sloužící pro převod na kód formátu SVF (*Serial Vector Format*), se kterým umí pracovat rozhraní JTAG procesorů řady Atmel AVR.

Tento systém využívá k programování čtyři piny [12], TCK, TMS, TDI a TDO. Data do obvodu vstupují pomocí pinu TDI a pinem TDO z něj vystupují zpět do programovacího rozhraní. Úroveň na pinu TDI je zachycena ve speciálním registru a po aktivní hraně hodinového signálu TCK (nezávisí na hodinovém signálu aplikace) je naprogramována příslušná buňka. TMS signál slouží k aktivaci programovacího režimu.

Aby bylo možné toto rozhraní používat je nutné správně nastavit FUSE bity procesoru ATmega128, přesněji jde o pojistku JTAGEN. Pojistky se nastavují v jednom z parametrů programu AVRsvf. Musí se dbát na jejich správné nastavení protože by mohlo dojít k nevratným změnám na čipu a tím k jeho trvalému znehodnocení.

2.3.6 Vývojové nástroje

Pro programování v jazyce C existuje několik programů. Mezi nejrozšířenější patří asi WinAVR [14], který je k dispozici zdarma. Tento program je vystavěn na kompilátoru GCC. Další program je

CodeVisionAVR [15]. Nástrojem pro testování a ladění je například AVR Studio [16]. Já jsem použil WinAVR ve verzi 20060421.

Pro samotné programování paměti čipu ATmega128 jsem používal programátor PRESTO [17], který umožňuje programovat přes rozhraní SPI a JTAG.

2.4 MultiMediaCard (MMC)

Jde o univerzální přenosné medium pro nenákladné ukládání dat. Protože tato karta byla navrhována hlavně pro užití v mobilních zařízeních, byl kladen důraz na její mobilitu a také na její cenu. Jinak řečeno důraz byl kladen na její nízkou spotřebu a velkou datovou propustnost na jejím rozhraní při komunikaci s okolím.

Komunikace karty s okolím je založena na pokročilé sedmi pinové sběrnici pracující na nízkých úrovních napětí. Komunikační protokol je definován jako část standardu MMC a je uveden jako MultiMediaCard režim. Aby byla zachována kompatibilita s existujícími kontroléry, MMC karta nabízí ještě kromě MMC režimu další alternativní režim, který je založen na SPI standardu.

Pro zachování napěťové kompatibility byly v standardní specifikaci definovány dva typy MMC, které se liší úrovní napájecího napětí. Jde o typy High Voltage MMC a Low Voltage MMC.

2.4.1 Systémové vlastnosti

MultiMediaCard Systém má velké množství systémových vlastností, které zahrnují:

- Pokrytí širokého spektra aplikací např. od tzv. chytrých mobilních telefonů, a PDA po digitální záznamníky a hračky.
- Usnadnění práce designérům, kteří vyhledávají vývoj aplikací s jejich vlastními pokročilými a dokonalejšími rysy.
- Udržování kompatibility a souladu se současnými elektronickými, komunikačními a datovými standardy.

Následuje výčet těch hlavních vlastností MultiMediaCard Systému:

- Je určena jak pro mobilí tak stacionární aplikace.
- Možné úrovně systémového napětí presentuje Tabulka 3.

	High Voltage MMC	Low Voltage MMC
Komunikace	2.7-3.6	1.65-1.95, 2,7-3.6
Přístup k paměti	2.7-3.6	1.65-1.95, 2,7-3.6

Tabulka 3: *Úrovně systémového napětí*

- Jsou konstruovány jako read-only, read/write a I/O karty.
- Podporují proměnlivou frekvenci hodin 0-20 MHz, 0-26MHz nebo 0-52MHz.
- Maximální rychlost datového přenosu je 416Mbitů/s.

- Umožňují ochranu dat heslem.
- Podporují opravu poškozených paměťových polí.
- Zajišťují ochranu proti zápisu.
- Protokolové vlastnosti v závislosti na komunikačním protokolu:

MultiMediaCard Mode	SPI Mode
Sběrnice s deseti vodiči(hodiny, 1 bit pro řízení, 8-bitová datová sběrnice)	Sériová sběrnice se třemi vodiči (hodiny, Darwin, dataOut) + CS signál pro výběr karty
Výběr karty pomocí přiřazení unikátních adres	Výběr karty pomocí hardwarového signálu CS
Pro jednu sběrnici jedna karta	Karta potřebuje vyhrazený CS signál
Jednoduchá identifikace a přiřazení pracovní adresy	Výběr karty skrz hardwarový signál CS
Proti chybám chráněný datový přenos	Proti chybám nechráněný přenos dat.
Sekvenční a blokově jednoduché/vícenásobné příkazy pro čtení/zápis	Blokově jednoduché/vícenásobné příkazy pro čtení/zápis

Tabulka 4: *MMC operační módy podle specifikace verze 4.1*

- Definice dvou typů karet MMC Plus a MMC Mobile.

2.4.2 Architektura

Paměťová karta MMC přenáší data skrz konfigurovatelný počet signálů sběrnice. Signály pro komunikaci jsou:

- **CLK:** Každá změna tohoto signálu řídí přenos jednoho bitu příkazové linky a jednoho bitu na všech datových linkách. Frekvence se může pohybovat mezi nulou a maximální možnou. Pro přenos doporučena frekvence do 20MHz, pro identifikační mód 400kHz.
- **CMD:** Tento obousměrný příkazový kanál slouží pro inicializaci karty a přenos příkazů. Signál CMD má dva operační módy: open-drain pro inicializační mód, a push-pul pro rychlý přenos příkazů. Příkazy jsou posílány z masteru na sběrnici MultiMediaCard do karty a odpovědi zase z karty hostiteli.
- **DAT0-DAT7:** Jsou to obousměrné datové kanály. Signály DAT operují v módu push-pull. MultiMediaCard obsahuje interní pull-upy pro všechny datové linky. Defaultně je po zapojení nebo resetu, pro přenos dat připraven pouze DAT0. Širší datovou sběrnici buď DAT0-DAT3 nebo DAT0-DAT7 je možné aktivovat pro datové přenosy použitím kontroléru MMC.

Karta je přímo připojena k signálům sběrnice. Následující tabulka ukazuje kontakty karty:

Číslo pinu	Název	Typ[1]	Popis
1	DAT3	I/O/PP	Data
2	CMD	I/O/PP/OD	Příkaz/Odpověď
3	V_{SS1}	S	Napájení zem
4	V_{DD}	S	Napájení
5	CLK	I	Hodiny
6	V_{SS2}	S	Napájení zem
7	DAT0	I/O/PP	Data
8	DAT1	I/O/PP	Data
9	DAT2	I/O/PP	Data
10	DAT4	I/O/PP	Data
11	DAT5	I/O/PP	Data
12	DAT6	I/O/PP	Data
13	DAT7	I/O/PP	Data

Tabulka 5:*Rozhraní MultiMediaCard*

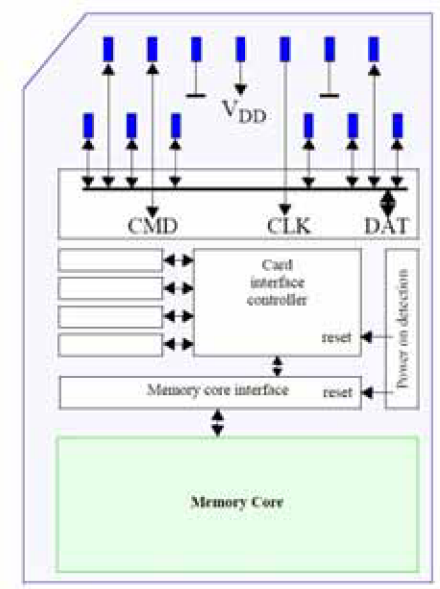
K inicializaci karty je potřeba jen CMD kanál. V další tabulce je pak uvedena sada registrů.

Název	Šířka [B]	Popis	Implementace
CID	16	Card IDentification numer - obsahuje informace pro identifikaci karty.	Povinná
RCA	2	Relative Card Address – udává dynamicky přidělovanou adresu karty.	Povinná
DSR	2	Drive Stage Register – konfigurace ovladačů.	Volitelná
CSD	16	Card Specific Data – informace o konfiguraci karty.	Povinná
OCR	4	Operation Condition Register – popisuje typ napájení karty.	Povinná
EXT_CSD	512	Extend Card Specific Data – obsahuje informace o specifických vlastnostech karty a vybraných módech.	Povinná

Tabulka 6: *Registry MultiMediaCard*

Hostitel má možnost resetovat kartu tím, že jí odpojí a opět připojí napájení, nebo ji lze resetovat pomocí příkazu CMD0 (GO_IDLE_STATE). Architekturu MMC a rozložení pinů znázorňuje Obrázek 9.

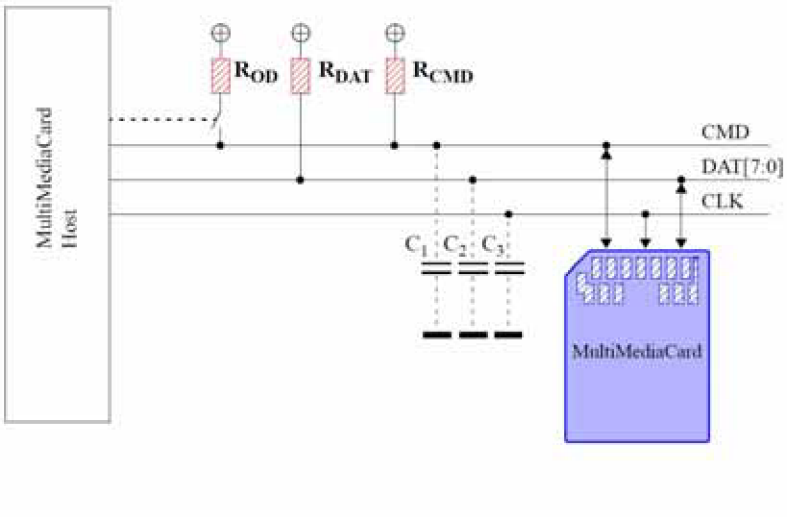
[1] S: zdroj napájení; I: vstup; O: výstup; PP: push-pull; OD: open-drain; NC: nepřipojeno(logická 1)



Obrázek 9: *Architektura MultiMediaCard [18][18]*

2.4.3 MMC režim

Tento režim využívá všech deseti komunikačních linek a tří napájecích linek. Po resetu, ke kterému dojde po zapojení, musí hostitel inicializovat kartu speciálním protokolem.

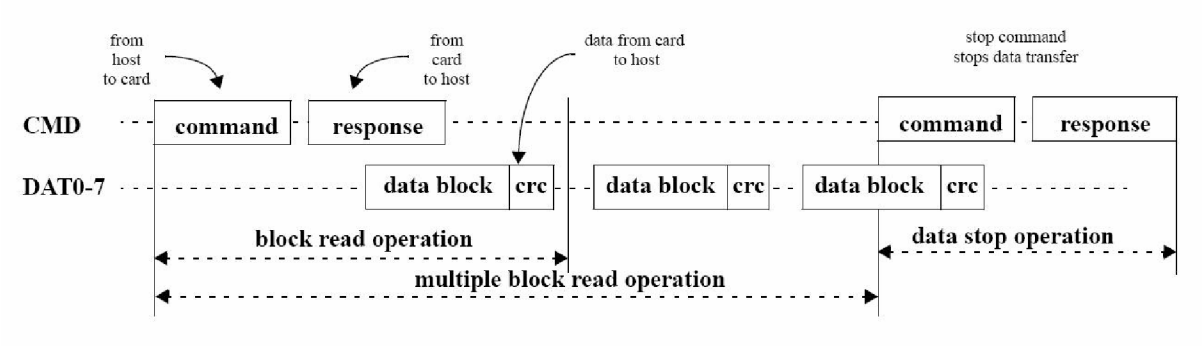


Obrázek 10: *Připojení karty k adaptéru v režimu MMC [18]*

K jednomu hostiteli je možné připojit víc karet MMC. Jejich adresování pak probíhá pomocí pracovních adres, které jim jsou přiděleny během inicializace. Každá zpráva je reprezentovaná jedním z následujících znaků:

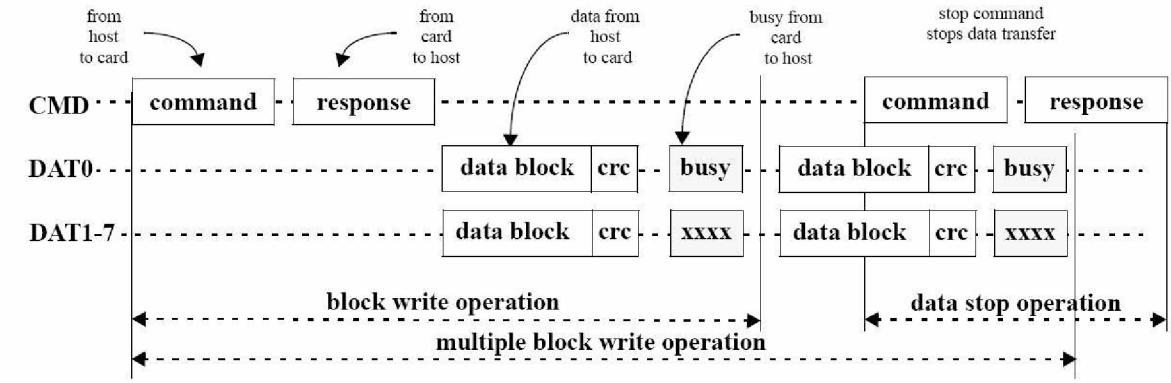
- příkaz – posílá sériově host po CMD lince kartě. Je to znak, který startuje operaci.
- odpověď – posílá karta sériově po CMD lince hostu jako odpověď na doručený příkaz.
- data – jsou posílána skrze datové linky z karty hostu nebo obráceně. Linek pro datové přenosy se pohybuje od 1(DAT0) přes 4(DAT0-DAT3) po 8(DAT0-DAT7).

Kromě případu kdy se využívá pouze jedné datové linky jsou data posílána po datových linkách v blocích, ke kterým jsou připojeny CRC kontrolní bity. Jak operace čtení tak zápisu podporuje přenos po jednom bloku (sigle) tak i přenos po větším počtu bloků (multiple block transmition).



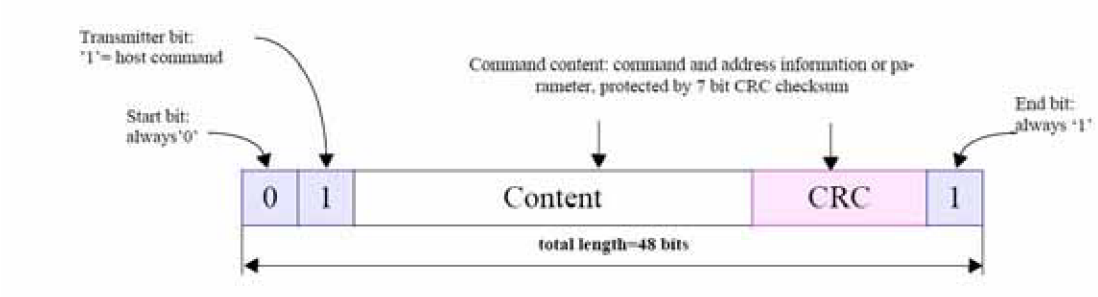
Obrázek 11: *(Multi) Bloková operace READ [18]*

Obrázek 12 ilustruje a jak je vidět, tak při zápisu se užívá po odeslání dat čekací signál BUSY na signalizaci dokončení operace. Nezáleží přitom na počtu použitých datových linek.



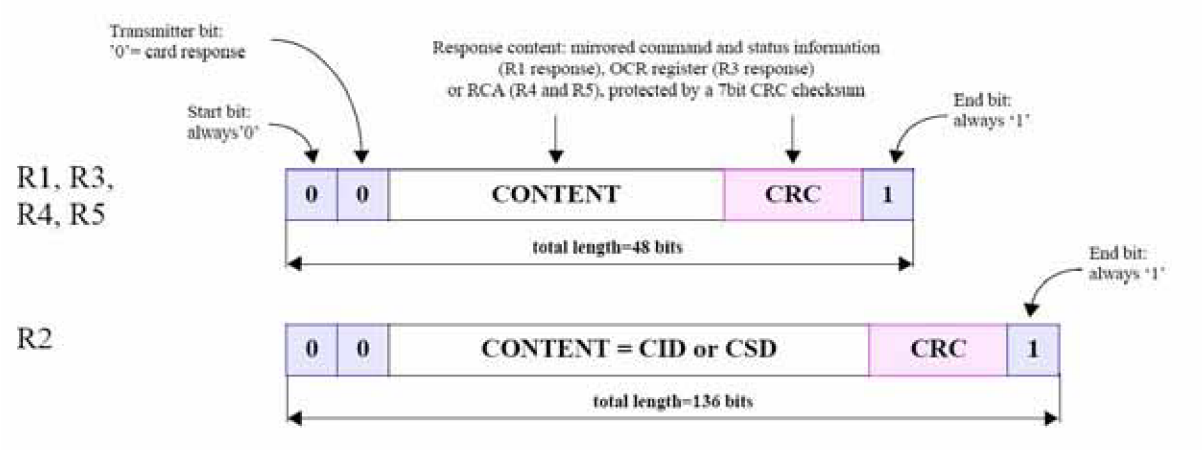
Obrázek 12: *(Multi) Bloková operace WRITE [18]*

Formát příkazového rámce ilustruje Obrázek 13. Vždy začíná startovacím bitem s hodnotou 0 a končí bitem 1. Délka rámce je 48 bitů. Každý rámec je chráněn sedmi CRC kontrolními bity.



Obrázek 13: *Formát příkazového rámce [18]*

Formát odpovědi má pět variant kódovacích schémat závisejících na obsahu. Jejich délka je buď 48 nebo 136 bitů.



Obrázek 14: *Formát rámce s odpovědí [18]*

Tento mód je zde uveden pouze z informačních důvodů. Pro bližší informace o tomto režimu doporučuji nahlédnout do [18]. Pro připojení karty MMC k mikrokontroléru web serveru bude použit následně uvedený režim SPI.

2.4.4 SPI režim

SPI režim je druhý podporovaný komunikační protokol pro karty MMC. Režim je navrhnut pro komunikaci SPI kanálem, který je běžně podporován microkontroléry Motorola a Atmel. Režim se volí během prvního resetu (tzn. první zapnutí) a nemůže být změněn dokud nedojde k vypnutí.

SPI standartd definuje pouze fyzickou vrstvu a nikoli kompletní protokol pro přenos dat. Jinak implementace SPI využívá MMC protokolu a jeho sady příkazů. Nevýhodou SPI je ztráta výkonu oproti MMC módu (nižší přenosová rychlost, hardware CS).

Topologie sběrnice

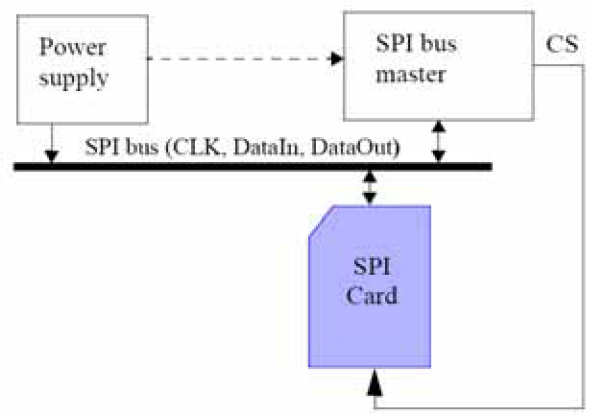
MultiMediaCard SPI komunikační kanál je tvořen čtyřmi následujícími signály:

- **CS:** signál pro výběr čipu (karty)
- **CLK:** hodinový signál
- **DataIn:** datový signál pro přenos z hostitele do karty
- **DataOut:** datový signál pro přenos z karty do hostitele

Další charakteristikou je přenos po bytech, který je implementovaný v kartě. Všechny datové rámce jsou tvořeny násobkem bytů, přičemž každý z nich je zarovnán vůči signálu CS. Identifikaci a adresování karty zajišťuje hardwarový Chip Select (CS) signál. CS signál musí být aktivní během trvání všech operací SPI (příkaz, odpověď, data).

Obousměrné linky CMD a DAT linky nahrazují jednosměrné linky DataIn a DataOut signály. To zabraňuje vykonávání příkazů během přenosu dat tedy zabraňuje sekvenčním a multi-blokovým operacím. SPI kanál podporuje pouze jedno-blokový přenos příkazů.

Připojení pinů MMC v SPI režimu předvádí Tabulka 7.



Obrázek 15: *SPI sběrnice a MMC [18]*

MMC mód			SPI mód		
Název	Typ[2]	Popis	Název	Typ	Popis
DAT3	I/O/PP	Data	CS	I	Chip Select
CMD	I/O/PP/OD	Příkaz/Odpověď	DI	I/PP	Data In
V_{SS1}	S	Napájení zem	VSS	S	Napájení zem
V_{DD}	S	Napájení	VDD	S	Napájení
CLK	I	Hodiny	SCLK	I	Hodiny
V_{SS2}	S	Napájení zem	VSS2	S	Napájení zem
DAT0	I/O/PP	Data	DO	O/PP	Data Out
DAT1	I/O/PP	Data	NC		
DAT2	I/O/PP	Data	NC		
DAT4	I/O/PP	Data	NC		
DAT5	I/O/PP	Data	NC		
DAT6	I/O/PP	Data	NC		
DAT7	I/O/PP	Data	NC		

Tabulka 7: *Konfigurace pinů SPI rozhraní*

Protokol SPI

Komunikace v SPI módu se skládá z příkazů, odpovědí a datových bloků. Celou komunikaci mezi hostitelem a kartou řídí hostitel (master), který každou sběrnicovou transakci začíná nastavením signálu CS na logickou nulu. Adresovaná karta vždy odpovídá na příkaz. Pokud nastane problém se zpřístupněním dat, následuje místo bloku dat chybová odpověď.Na každý datový blok poslaný kartě během operace zápisu je požadována reakce odpovídajícím datovým rámcem. Datový blok může být tak velký, jak je velký jeden blok pro zápis na kartě a tak malý jako jeden byte. Částečné .blokové

[2] S: zdroj napájení; I: vstup; O: výstup; PP: push-pull; OD: open-drain; NC: nepřipojeno(logická 1)

operace čtení nebo zápisu jsou povolené v závislosti na hodnotě registru CSD. V režimu SPI jsou přístupné pouze registry OCR, CSD a CID.

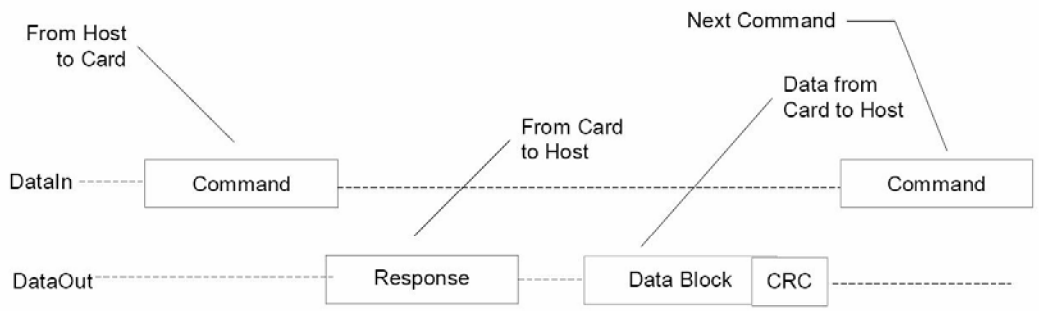
Výber režimu a inicializace

MMC karta se po připojení nastavuje do režimu MMC. Karta přejde do režimu SPI pokud je CS signál v logické nule během přijetí reset příkazu CMD0. Po přechodu do režimu SPI odpovídá karta SPI R1 odpovědí. Jediná cesta, jak se vrátit opět do režimu MMC je vypnout a následně opět zapnout zařízení (*power cycle*).

Přenos dat

Každý přenášený rámec je chráněn CRC bity. Mód SPI nabízí po inicializaci nechráněný režim. Hostitel může spouštět nebo vypínat CRC kontrolu příkazem CMD59 (*CRC_ON_OFF*).

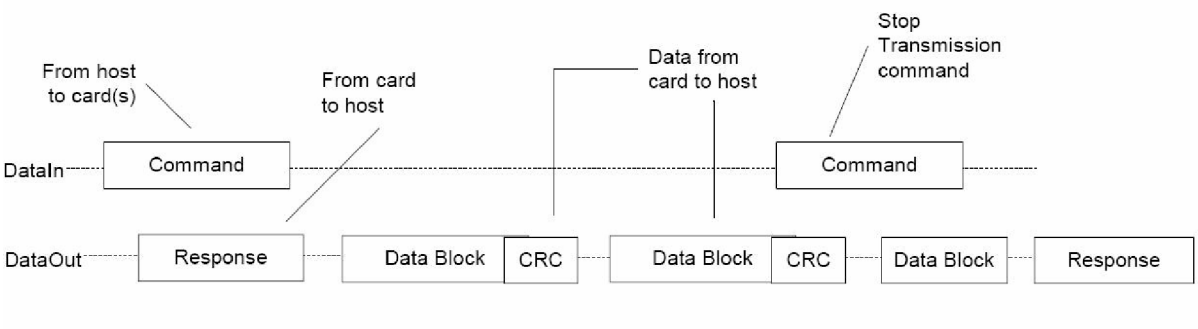
SPI režim podporuje čtení po jednom nebo po více blocích. Jak ukauje Obrázek 16, data i odpovědi jsou obě přenášeny od karty na lince Data Out.



Obrázek 16: *Operace čtení po jednom bloku [19]*

Základní jednotkou pro přenos dat je blok jehož maximální velikost je definována v CSD (*READ_BL_LEN*). CMD17 (*READ_SINGLE_BLOCK*) inicializuje čtení jednoho bloku.

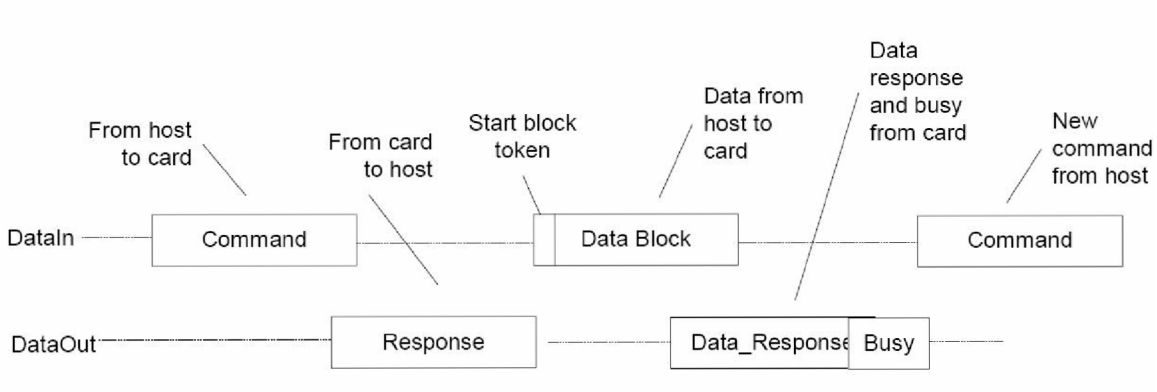
CMD18 (*READ_MULTIPLE_BLOCK*) startuje přenos posloupnosti několika bloků. Počet bloků pro tuto operaci není nijak definován. Karta bude přenášet jednotlivé bloky tak dlouho dokud neobdrží příkaz pro zastavení přenosu CMD12 (*STOP_TRANSMISSION*).



Obrázek 17: *Multi bloková operace čtení [19]*

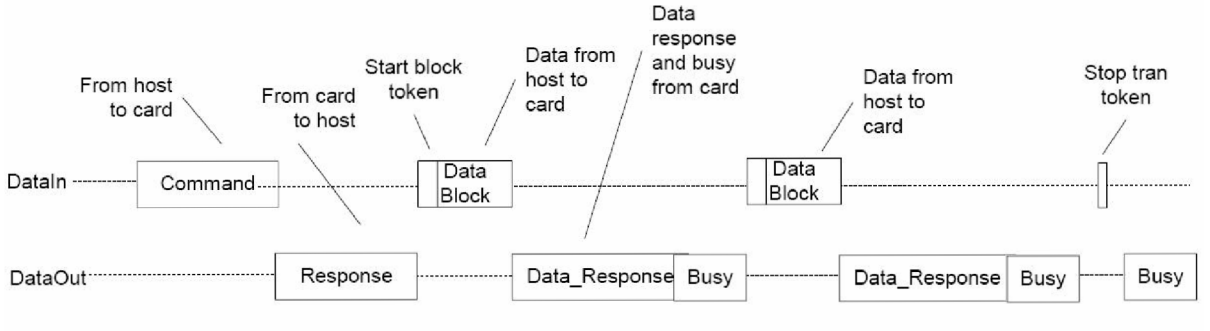
Zápis opět podporuje operace pro zápis po jednom bloku nebo více bloků. Po obdržení příkazu pro zápis (CMD24 nebo CMD25) karta odpoví a čeká na blok dat od hostitele.

Každý datový blok začíná bytem Start blok. Po obdržení datového bloku MMC karta odpovídá paketem Data response a pokud byla data obdržena bez chyb dochází k samotnému zápisu. Karta po dobu programování vysílá po sběrnici busy pakety.



Obrázek 18: *Operace zápisu jednoho bloku [19]*

Při multi-blokové operaci zápisu, se posílání dat ukončuje paketem Stop tran. Počet bloků opět není nijak definován. Karta přijímá datové bloky dokud neobdrží paket Stop tran.



Obrázek 19: *Operace multi-blokového zápisu [19]*

Po ukončení operace programování host kontroluje jak tato operace dopadla zasláním příkazu CMD13 (*SEND_STATUS*).

Proces programování karty je během stavu busy imunní vůči resetu signálu CS. Přerušit programování karty lze pouze příkazem CMD0, ale může dojít k zničení datové struktury. Host by měl hlídat, aby k tomu nedošlo.

Čtení obsahu informačních registrů CID a CSD lze provádět pomocí jednoduchých blokových operací pro čtení. Karta odpovídá standardními datovými bloky dlouhými 16 bytů a 16 bitovým CRC.

Příkazy SPI režimu

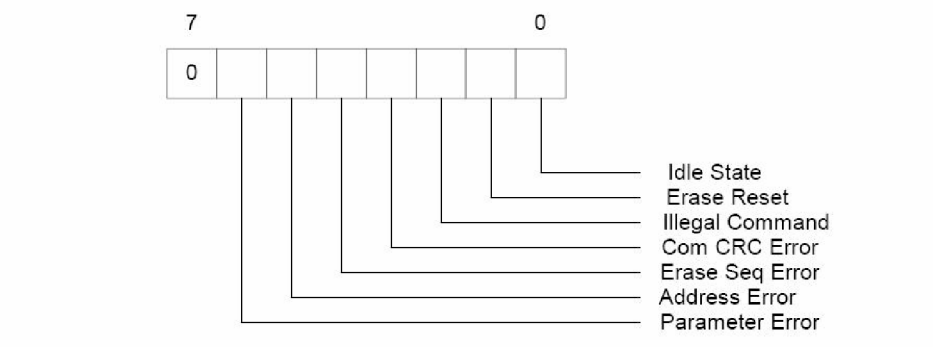
Všechny příkazy v SPI režimu mají délku 48 bitů a přenos začíná tím nevýznamnějším. Formát příkazu ilustruje Obrázek 20.

0	1	bit 5....bit 0	bit 31....bit 0	bit 6....bit 0	1
start bit	host	command	argument	CRC7	end bit

Obrázek 20: *Formát paketu příkazu*

Pro detailní popis příkazů doporučuji nahlédnout do [18]. Binární kódy příkazů jsou definovány jejich pořadovým číslem např. pro CMD0 to je „000000“ nebo pro CMD39 to je „100111“.

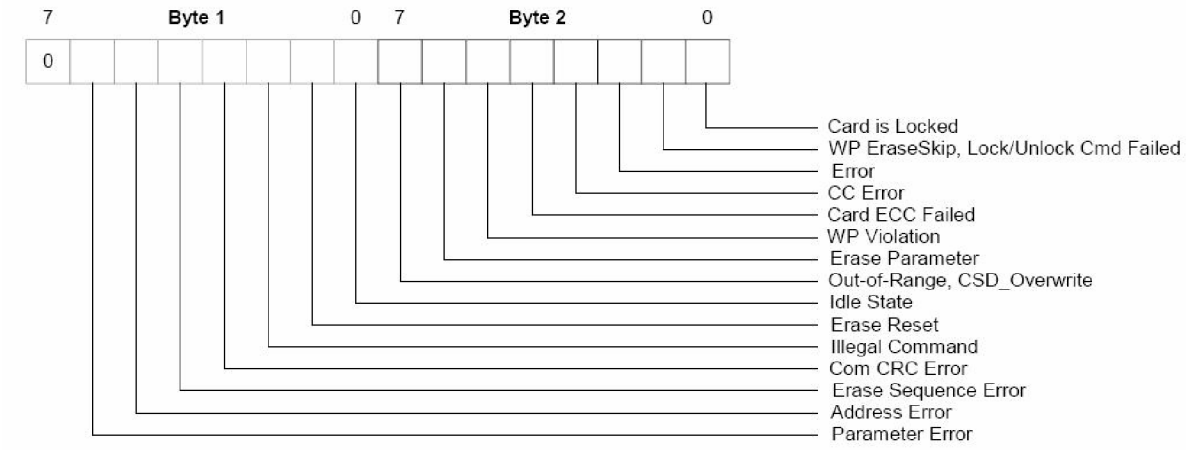
Odpovědi, které karta posílá jako reakce na příkazy nebo chyby, jsou R1, R1b, R2, R3 a Data-response. Formát odpovědi R1 je posílán jako reakce na všechny příkazy kromě SEND_STATUS. První MSB bit je 0, ostatní pak indikují výskyt určité chyby, tak ukazuje Obrázek 21.



Obrázek 21: *Formát R1 odpovědi [19]*

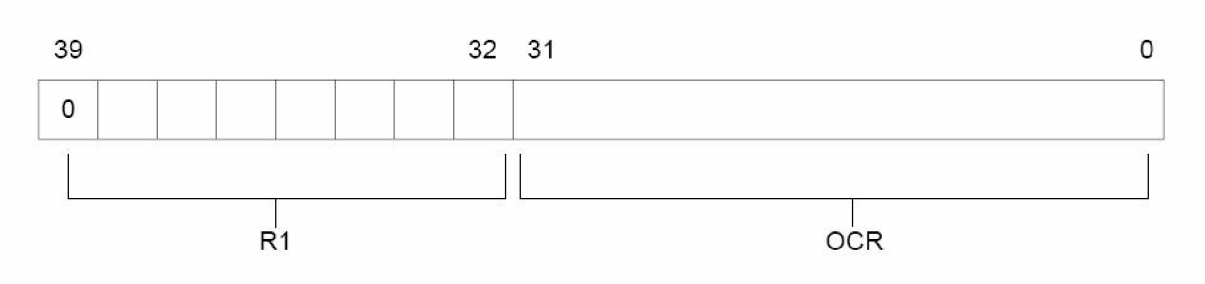
R1b je odpověď R1 rozšířená o signál busy. Signál busy má k dispozici libovolný počet bytů. Nulová hodnota indikuje stav busy. Nenulová hodnota naopak indikuje připravenost pro příjem dalšího příkazu.

Karta posílá 2 byty dlouhou odpověď R2 jako reakci na příkaz SEND_STATUS. První byt je identický s R1 a druhý - viz.Obrázek 22.



Obrázek 22: *Formát odpovědi R2 [19]*

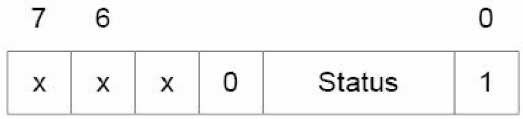
R3 odpověď je posílána jako reakce na obdržený příkaz READ_OCR. Délka odpovědi je 5 bytů. Struktura prvního bytu (MSB) je shodná s R1 a další 4 byty obsahují data OCR registru.



Obrázek 23: *Formát odpovědi R3 [19]*

Každý blok dat zapsaný na kartu je potvrzen odpovědí Data-response. Odpověď je jeden byte dlouhá a má formát, který ukazuje Obrázek 24. Význam stavových bitů:

- 010 – data akceptována
- 101 – data zamítnuta pro chybu v CRC
- 110 – data zamítnuta pro chybu při zápisu



Obrázek 24: *Formát odpovědi Data-response [19]*

Při výskytu chyby během operace multi-blokového zápisu host přeruší operaci příkazem CMD12 (*STOP_TRANSMISSION*). Konkrétní chybu je možné zjistit příkazem CMD13 (*SEND_STATUS*).

Další podrobnější informace o MultiMediaCard mediu jsou k nalezení v [19].

2.5 File Alocation Table (FAT)

Protože jsem se rozhodl ukládat data HTML stránek na MMC kartu budu se v této kapitole věnovat souborovému systému, který bude na tomto médiu použit. Souborový systém FAT [20] uchovává informace o alokovaném prostoru v tabulkách.

Jako většina technologií spojená s informačními technologiemi i FAT file systém se postupně vyvíjel a zdokonaloval. Dnes existují tři verze: FAT12, FAT16, FAT32. Některé vlastnosti jednotlivých verzí předvádí Tabulka 8.

Vlastnosti	FAT12	FAT16	FAT32
Rok vydání	1977	1987	1996
Max velikost svazku	32 MB	2 GB	8 TB
Max velikost souboru	32 MB	2 GB	4 GB
Max počet souborů	4077	65517	268 435 437

Tabulka 8: *Vlastnosti FAT*

Významný rozdíl v typech FAT představuje počet bitů vyhrazených pro umístění jednoho záznamu na disku. Od této velikosti se odvozuje i název typu FAT. Platí, že čím větší je šířka, tím vetší počet záznamů je možné adresovat. Poslední verze FAT32 už dosahuje slušné velikosti svazků. Ale nevýhodou je hranice maximální velikosti souboru, která už v dnešní době nedostačuje.

FAT12 se dnes už stává zastaralým a je využitelný pro média s velikostí do 16MB, a proto už není běžně podporován. To se netýká FAT16 a FAT32. 16 a 32 jednoduše představuje velikost clusteru v bitech, ačkoliv FAT32 využívá pouze 28 bitů a 4 jsou rezervované. FAT32 využívá svazky větší velikosti efektivněji než FAT16 a také je méně citlivý na chyby, protože používá záložní kopii důležitých datových struktur v boot recordu.

2.5.1 Master Boot Rekord MBR

Tabulka MBR se nachází v nultém sektoru daného média a popisuje jednotlivé oddíly na něm se nacházející. Na začátku sektoru je 446 bytů spustitelného kódu. Za touto oblastí se nachází čtyři 16 bytové popisy jednotlivých oddílů.

Offset	Velikost	Popis
000H	446	Spustitelný kód zavaděče
1BEH	16	1. položka tabulky rozdělení disku (PAT)
1CEH	16	1. položka tabulky rozdělení disku (PAT)
1DEH	16	1. položka tabulky rozdělení disku (PAT)
1EEH	16	1. položka tabulky rozdělení disku (PAT)
1FEH	1	Znak 0x55
1FFH	1	Znak 0xAA

Tabulka 9: *Struktura MBR*

Strukturu jednotlivých popisů uvádí Tabulka 10. Každý záznam obsahuje informace o schopnosti bootování, adresu začátku, adresu konce svazku na médiu a počet sektorů v oddílu.

Offset	Délka[B]	Popis
00H	1	Stav bootovacího příznaku (0 = neaktivní, 80H = aktivní)
01H	3	Začátek – adresa hlavy, cylindru a sektoru
04H	1	Typ oddílu (Tabulka 11)
05H	1	Konec – adresa hlavy, cylindru a sektoru
08H	4	Adresa počátečního sektoru (relativně k počátku média)
0CH	4	Počet sektorů oddílu

Tabulka 10: *Záznam vlastností oddílu v MBR*

Typ(hodnota)	Popis
00H	Neznámý/žádný
01H	Fat12
04H	FAT16
05H	Rozšířený oddíl DOS
06H	FAT16
0BH	FAT32
0CH	FAT32 s LBArozšířením
0EH	FAT16 s LBA
0FH	Rozšířený oddíl DOS s LBA

Tabulka 11: *Typy oddílu FAT*

Rozšířený oddíl DOS umožňuje větší počet oddílů na médiu než jsou základní čtyři. Ukazuje na strukturu, podobnou MBR, která obsahuje popis dalšího oddílu FAT a případně odkaz na další rozšířený oddíl. Offset je ovšem vztažen k pozici aktuální tabulky, je tedy nutné ho přičíst k pozici udávané v popce offset. Poslední rozšířený oddíl obsahuje již pouze poslední oddíl FAT.

2.5.2 Struktura oddílu systému FAT

Každý svazek s FAT systémem má strukturu, kterou ukazuje Tabulka 12.

Boot sector	Reserved sectors	FAT č. 1	FAT č. 2	Root dir (jen FAT12/16)	Data

Tabulka 12: *Struktura FAT oddílu*

První sektor obsahuje BPB (*Bios Parameter Block*). BPB kromě mnoha dalších informací o oddílu obsahuje např. zavaděč OS. Po Boot sektoru volitelně následuje několik rezervovaných sektorů. Jejich počet udává položka Reserved Sector Count v BPB. Důležité jsou dvě kopie alokačních tabulek soborů (*File Alocation Table*). Root adresář se nachází za kopií tabulek u typů FAT12 a FAT16. FAT32 má svůj kořenový adresář v datové části.

Boot Record

Obsahuje všechny potřebné informace pro správné nastavení oddílu. Podrobné informace o uvádí Tabulka 13.

Název	Offset [B]	Velikost [B]	Popis
BS_jmpBoot	00H	3	Adresa spustitelné oblasti
BS_OEMName	03H	8	Název systému
BPB_BytsPerSec	0BH	2	Počet bytů na sektor (512, 1024, 2048, 4096)
BPB_SecPerClus	0DH	1	Sektorů na cluster
BPB_RsvdSecCnt	0EH	2	Rezervované sektory
BPB_NumFATs	10H	1	Počet FATtabulek
BPB_RootEntCnt	11H	2	Počet položek v Root Direktory (pro FAT32 = 0)
BPB_TotSec16	13H	2	Celkový počet sektorů (pro FAT32 = 0)
BPB_Media	15H	1	Typ média (F8H = přenositelná, F0H = nepřenositelná)
BPB_FATSz16	16H	2	Počet sektorů na FAT (pro FAT32 = 0)
BPB_SecPerTrk	18H	2	Sektorů na stopu (pro média s geometrií)
BPB_NumHeads	1AH	2	Počet hlav (pro média s geometrií)
BPB_HiddSec	1CH	4	Počet skrytých sektorů
BPB_TotSec32	20H	4	Celkový počet sektorů pro FAT32

Tabulka 13: *Struktura Boot Record*

Další části Boot Sektoru mají odlišnou strukturu pro FAT16 a FAT32. Obsahují informace o adrese kořenového adresáře (pro FAT32), názvu oddílu, identifikačním klíči, verzích. Velikost je pro obě verze stejná a to 26 bytů. Některé důležité položky této struktury pro FAT16 a FAT32 uvádí Tabulka 14.

Název	Offset [B]	Velikost [B]	Popis
FAT16			
BS_VolLab	2BH	11	Jméno oddílu
BS_FilSysType	36H	8	Typ: FAT12 nebo FAT16
FAT32			
BPB_FATSz32	24H	4	Velikost FAT tabulky v sektorech
BPB_RootClus	2CH	4	Adresa prvního clusteru kořenového adresáře
BS_VolLab	47H	11	Jméno oddílu
BS_FilSysType	52H	8	Typ: FAT32

Tabulka 14: *Některé další části Boot Recordu závislé na typu FAT*

FAT.

FAT je vlastně zřetězená struktura sloužící k uchování informací o přiřazených datových clusterech k souborům a adresářům.

Základní informace

Celkový počet sektorů oddílu vyčteme ze struktury Boot Rekord z položky *BPB_TotSec16*, pokud je rovna nule najdeme ho v *BPB_TotSec32*. Podobně je to s velikostí FAT tabulky, ta je uložena v položce *BPB_FATSz16*, pokud je rovna nule najdeme tuto hodnotu v *BPB_FATSz32*.

Jak už bylo výše uvedeno FAT tabulky se nacházejí za oblastí rezervovaných sektorů. První sektor tabulky pro FAT32 se dá určit pomocí následujícího vztahu:

$$firstDataSec = BPB\_resvdSecCnt + BPB\_NumFATs * FATSz + rootDirSecs,$$

kde $FATSz$ udává velikost FAT tabulky a počet sektorů kořenového adresáře $rootDirSecs$ vypočteme takto:

$$rootDirSecs = (BPB\_RootEntCnt * 32 + BPB\_BytsPerSec - 1) / BPB\_BytsPerSec.$$

Potom adresa prvního sektoru v N-tém clusteru je:

$$firstSecClusN = (N - 2) * BPB\_SecPerClus + firstDataSec.$$

Tato adresa je relativní k adrese prvního sektoru oddílu.

Pro FAT32 je tabulka tvořena polem položek o velikosti 4B, z nichž je použito 28 bitů. Nejvyšší 4 bity jsou rezervovány a neměly by se měnit. Pro velikost sektoru 512B se tedy v jednom sektoru nachází 128 položek.

Každá položka obsahuje adresu následujícího clusteru souboru nebo adresáře. Hodnota větší nebo rovna 0FFFFFF8H znamená konec řetězce, tedy to že daný cluster je poslední. Číslo prvního clusteru řetězce je uloženo ve dvou položkách vlastností adresáře. Soubor o nulové délce, který dosud nemá alokována data, má v těchto položkách hodnotu 0. Hodnota 0FFFFFF7H značí vadný cluster. Hodnota 0 označuje volný cluster. Nultá a první položka (cluster) mají speciální význam a nepoužívají se.

Hodnota	Popis
00000000H	Volný cluster
00000001H – 00000002H	Zakázaná hodnota
00000003H – FFFFFFF6H	Adresa dalšího clusteru
FFFFFFF7H	Vadný cluster
FFFFFFF8 - FFFFFFFFH	EOF – konec souboru (řetězce clusterů)

Tabulka 15: *Možné hodnoty položek FAT*

Adresářová struktura

Informace o umístění adresářů a souborů nacházíme v rodičovském adresáři. Nejvyšší místo v hierarchii má kořenový adresář. Jeho adresu najdeme v tabulce Boot Rekord. Adresář obsahuje jednotlivé položky dlouhé 32 bytů, jejichž strukturu popisuje Tabulka 16.

Název	Offset	Velikost [B]	Popis
DIR_Name	00H	11	Jméno souboru nebo adresáře (+ přípona souboru)
DIR_Attr	0BH	1	Atributy položky (viz.)
DIR_NTRes	0CH	1	Rezervováno
DIR_CrtTimeTenth	0DH	1	Čas vytvoření – setiny sekundy
DIR_CrtTime	0EH	2	Čas vytvoření – granularita jsou 2 sekundy
DIR_CrtDate	10H	2	Datum vytvoření
DIR_LstAccDate	12H	2	Datum posledního přístupu
DIR_FstClusHi	14H	2	Číslo prvního clusteru (horní polovina)
DIR_WrtTime	16H	2	Čas poslední změny
DIR_WrtDate	18H	2	Datum poslední změny
DIR_FstClusLo	1AH	2	Adresa prvního clusteru (dolní polovina)
DIR_FileSize	1CH	4	Velikost souboru v bytech

Tabulka 16: *Struktura jedné položky v adresáři*

Formát času a data je přesněji uveden v následujících tabulkách. Aby se ušetřilo místo v struktuře adresářové položky, je struktura data a času umístěna do16 bitů, jak to ilustruje Tabulka 17 a Tabulka 18.

H4	H3	H2	H1	H0	M5	M4	M3	M2	M1	M0	S4	S3	S2	S1	S0

Tabulka 17: *Struktura času (H – hodina: 0-23, M – minuta: 0-59, S.2 – sekunda: 0-59 s krokem 2)*

Y6	Y5	Y4	Y3	Y2	Y1	Y0	M3	M2	M1	M0	D4	D3	D2	D1	D0

Tabulka 18: *Struktura data (H – hodina: 0-23, M – minuta: 0-59, S.2 – sekunda: 0-59 s krokem 2)*

Byte atributů rozlišuje o jakou položku se vlastně jedná, zda-li soubor nebo adresář, či jmenovku oddílu a další vlastnosti, viz Tabulka 19. Pokud má daný bit hodnotu 1, je atribut aktivní.

Hodnota	Popis
01H	Položka pouze pro čtení
02H	Skrytá položka
04H	Systémová položka
08H	Jméno svazku(pouze pro kořenový adresář)
10H	Adresář
20H	Archivní položka
0FH	Dlouhé jméno

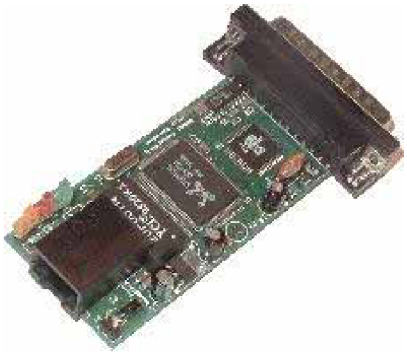
Tabulka 19: *Atributy položky*

3 Exitující řešení web serveru

V prostředí Internetu se dnes můžeme setkat s celou řadou řešení, které se zabývají podobným problémem – implementací web serveru pro Embedded systémy. Dále je stručně popsáno několik podobných projektů.

3.1 PicoWeb

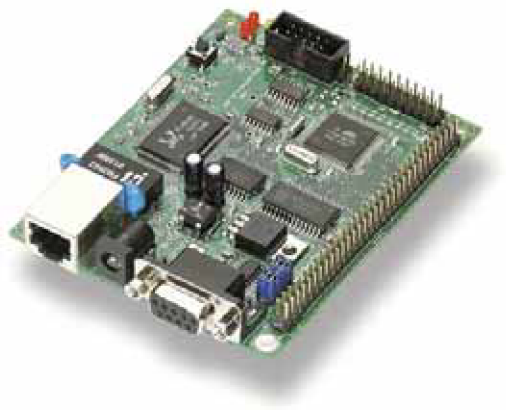
PicoWeb využívá mikrokontrolér Atmel AT90S8515 a řadič Ethernetu Realtek RTL8019AS. Na stránkách http://www.picoweb.net/index.htm jsou další podrobnější informace.



Obrázek 25: *PicoWeb*

3.2 Ethernut

Projekt Ethernut vzniknul v Německu a jeho autorem je Harald Kipp. Jedná se v současné době o jeden z nejprogresivnějších projektů v oblasti Embedded ethernetu. Projekt je vyvíjen a šířen pod BSD licencí a tvoří ho několik verzí kompletních řešení založených na mikrokontrolérech ATmega128, ATmega2561 nebo AT91R40008 od firmy Atmel. Součástí projektu je operační systém Nut/OS a jeho nadstavba Nut/Net, která představuje implementaci TCP/IP protokolového modulu. Další informace o projektu jsou dostupné na internetové adrese http://www.ethernut.de/



Obrázek 26: *Ethernut 1.3*

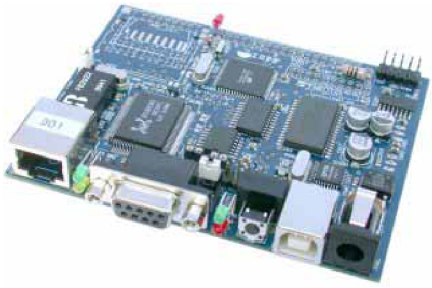
3.3 Web51

Web51 je projekt založený na připojení mikrokontroléru Atmel AT89C8252 k síťovému řadiči Realtek RTL8019AS. Výsledkem je velmi levné rozhraní do sítě Ethernet. Na stránce http://web51.hw.cz/ jsou dostupné další informace o tomto řešení.

Obrázek 27: *Web51*

3.4 Arthernet

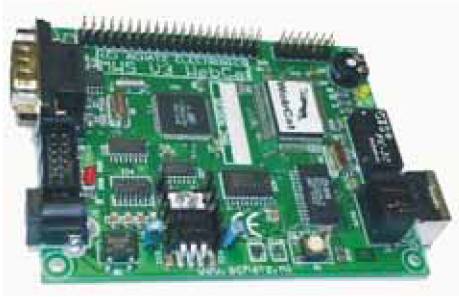
Po hardwarové stránce Arthernet vychází také z mikrokontroléru ATmega128, řadiče RTL8019, bývá dále osazen 512kB RAM, 4MB/8MB paměti FLASH-ROM. Co má navíc oproti ostatním řešením je USB rozhraní. Softwarově má stejné vybavení jako Ethernut.



Obrázek 28: *Arthernet*

3.5 WebCat

WebCat je projekt, který je hardwarově kompatibilní s projektem Ethernut ve verzi 1.3. A jelikož z tohoto projektu vychází, tak používá i stejné softwarové vybavení.



Obrázek 29: *WebCat*

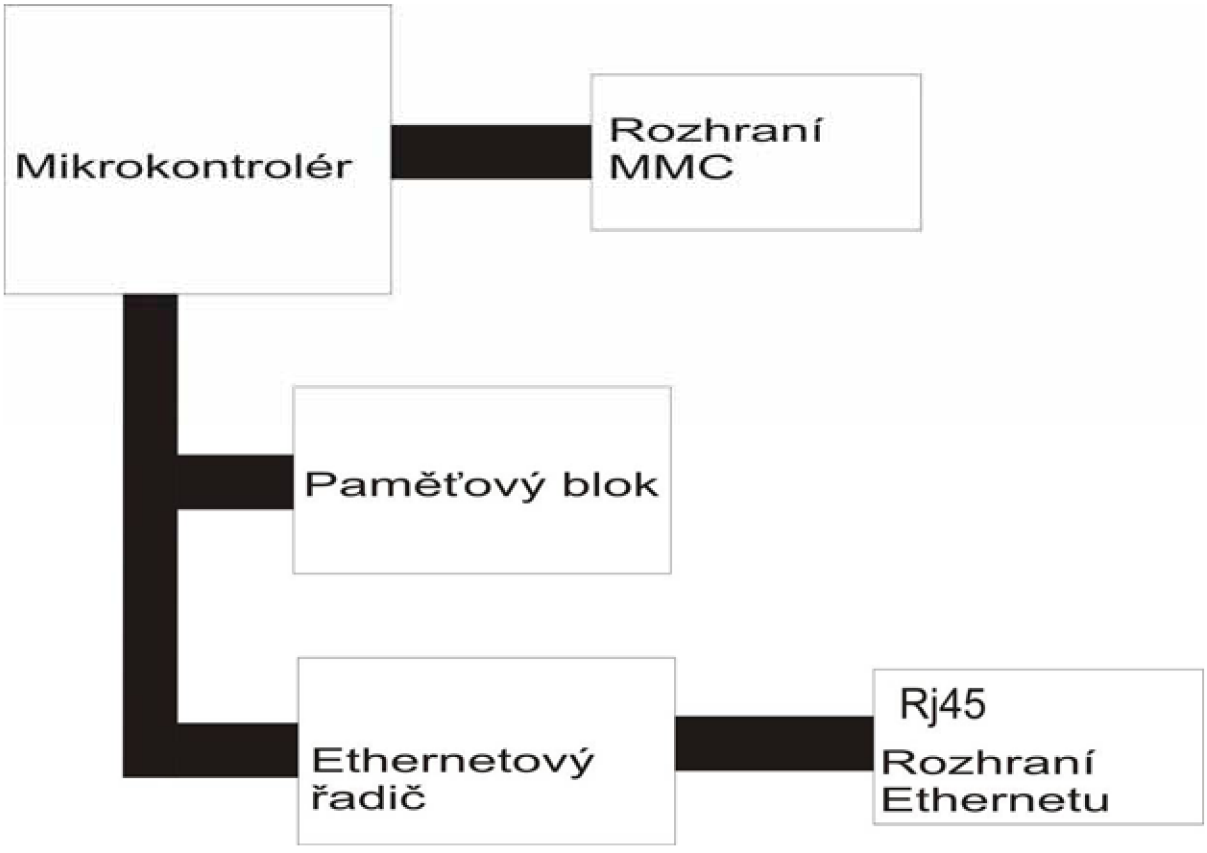
4 Návrh web serveru

4.1 Návrh platformy web serveru

Platforma, kterou jsem zvolil pro další návrh Embedded web serveru, obsahuje mikrokontrolér, Ethernetový řadič, paměťový blok, rozhraní pro paměťovou kartu MMC a další komunikační rozhraní.

Rozhodl jsem se navrhnout hardware, který bude kompatibilní s hardwarem projektu Ethernut ve verzi 1.3. Budu tak moci využít realtime operační systém NUT/OS, který byl vytvořen pro jednodušší návrh aplikací takto navrženého HW.

Protože samotné http stránky jsou v projektu Ethernut uloženy v programové paměti procesoru je značně omezená paměťová kapacita pro takové stránky nebo případně pro jiná data se kterými by takové zařízení mohlo pracovat. S tím souvisí i snížení životnosti procesoru, která je spjata s omezeným počtem pro přepis paměti FLASH. Proto jsem se rozhodl rozšířit platformu o rozhraní pro MMC kartu. Na této paměťové kartě budou uloženy http stránky. Bude je možné jednoduše aktualizovat a kapacita pro stránky bude závislá pouze na kapacitě MMC karty, kterou je možno kdykoli vyměnit za jinou kartu s větší kapacitou.



Obrázek 30: *Navrhovaná platforma web serveru jako blokové schéma*

4.2 Součástková základna

Výběr procesoru je jedním z kritických rozhodnutí, které ovlivňuje úspěch celého projektu. Hlavním cílem je volba procesoru s nejnižší pořizovací cenou při splnění všech požadavků stanovených zadáním. Stále více periferií je integrováno přímo na čipu. Tato skutečnost se pozitivně projevuje do nárůstu spolehlivosti (testování již během vývoje) a komfortu při ovládání (dedikované instrukce procesoru). Vyloučení externích prvků sebou přináší i snížení požadavků na velikost desky plošných spojů.

V komplexnějších zapojeních však můžeme narazit na problém výskytu několika druhů napájecích napětí, na což je třeba brát patřičný zřetel.

Dalším kriteriem při samotné realizaci projektu jsou možnosti při návrhu a testování softwaru. Vhodná volba vývojových a podpůrných nástrojů nám ušetří spoustu zbytečných komplikací. Není důvod se obávat nasazení vyšších programovacích jazyků. Kvalita výsledného kódu u moderních kompilátorů je srovnatelná s programem napsaným v jazyku symbolických adres.

Protože jsem se snažil vytvořit HW, který bude kompatibilní s operačním systém Ethernut musím vzít také v úvahu podporu ze strany operačního systému NUT/OS.

4.2.1 Mikrokontrolér

Kritéria pro výběr mikrokontroléru:

- integrace pamětí typu RAM, FLASH, EEPROM na čipu,
- integrace inteligentních periferií na čipu (UART, ADC, PWM, PCA, atd.),
- snížená spotřeba,
- snížené vyzařování, rušení,
- podpora programování ISP, IAP,
- dostatečná rezerva I/O pinů,
- provozní teploty pro vojenské a průmyslové aplikace,
- návaznost na ostatní použité součástky,
- dostupnost softwarového vybavení pro návrh programu procesoru,
- zavedená architektura procesoru,
- dostupnost procesoru na trhu,
- perspektiva produkce procesoru (kompatibilní náhrady) v budoucnu,
- cena,
- podpora OS NUT/OS.

I přes horší dostupnost v maloobchodní síti a trochu vyšší cenu jsem nakonec zvolil produkt ATmega128 od firmy Atmel. U tohoto mikrokontroléru je dostatečná paměťová kapacita. Díky architektuře RISC je tento procesor také dostatečně výkonný. Na čipu je integrováno mnoho

inteligentních periferií. K dispozici je také dostatek vývojových nástrojů a je podporovaný operačním systémem NUT/OS.

4.2.2 Řadič Ethernetu

Kritéria pro výběr řadiče Ethernetu:

- 8-bitová datová sběrnice,
- komunikace přes jednoduché I/O operace,
- jednočipové řešení ethernetového řadiče,
- integrovaná paměť RAM pro vysílání a příjem dat,
- rychlost minimálně 10 Mb/s,
- napájecí napětí 5 V,
- volně dostupná programová a hardwarová dokumentace.

Dostupný požadavkům vyhovující řadič, pro který jsem se rozhodl, je RTL 8019AS produkovaný firmou Realtek.

4.2.3 Paměť

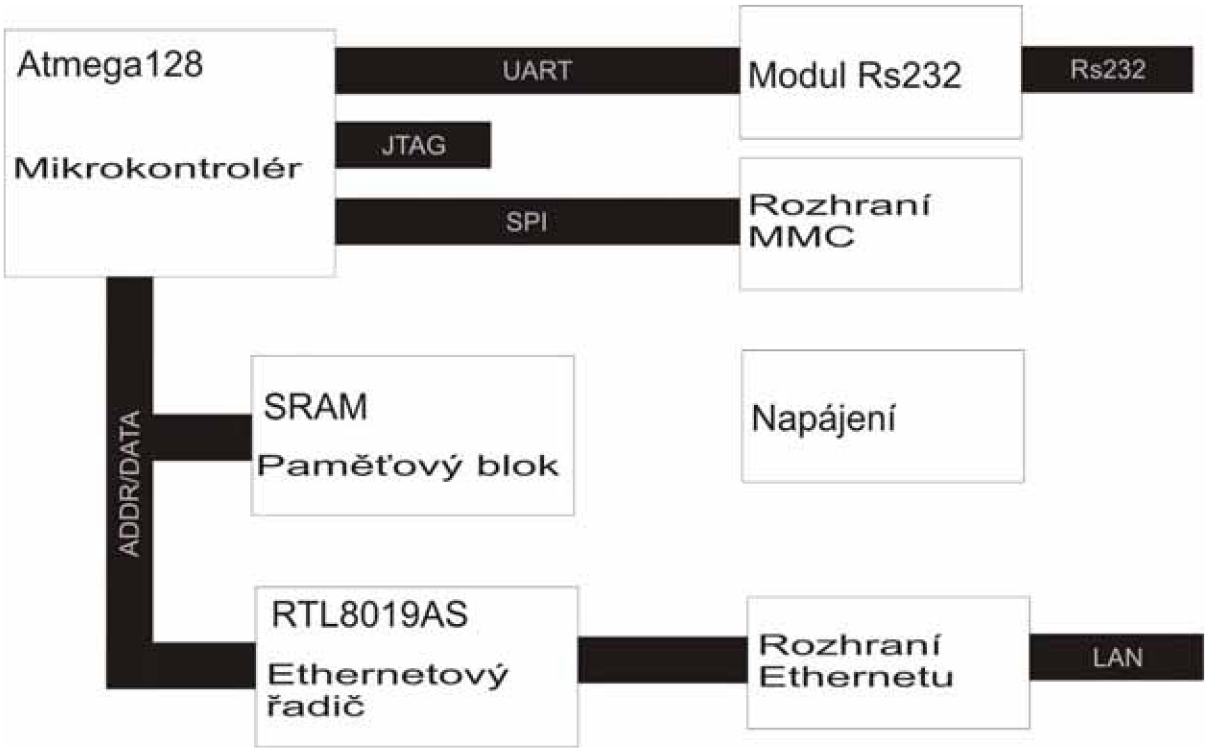
Mikrokontrolér ATmega128 podporuje 64Kb externí datové paměti. V tomto případě bude použito 32Kb paměti, protože 64Kb paměti SRAM na jednom čipu s potřebným časováním jsem nebyl schopen v maloobchodní síti získat.

4.2.4 Podpůrné obvody

Jednočipový mikropočítač a ethernetový řadič potřebují pro svoji základní funkci podpůrné obvody. Jedná se ve skrze o součástky běžně dostupné v maloobchodní síti (kondenzátory, odpory, krystaly, stabilizátory napětí, diody, převodník úrovní TTL/RS232, D-Buffery, atd.). Vzhledem k minimalizaci a zvýšení spolehlivosti celého zařízení je vhodné použít součástky v provedení SMD. Kompletní seznam všech použitých součástek a typů jejich pouzder je uveden jako příloha.

4.2.5 Blokové schéma web serveru

Jak presentuje Obrázek 31, k mikrokontroléru je pomocí datové a adresní sběrnice připojena externí paměť SRAM a ethernetový řadič. Přes integrované rozhraní mikrokontroléru USART je pak připojeno asynchronní sériové komunikační rozhraní RS-232. Programování paměti FLASH a EEPROM mikrokontroléru je možno provádět metodou ISP přes JTAG rozhraní. Rozhraní SPI mikrokontroléru, které se běžně používá také k programování mikrokontroléru, bude využito pro připojení rozhraní MMC karty.



Obrázek 31: *Blokové schéma navrhovaného web serveru*

5 Konstrukce web serveru

Po provedení návrhu a výběru hlavních komponent jsem byl postaven před samotnou fyzickou realizaci. Znamenalo to zabezpečení všech součástek, návrh, výrobu a osazení desky plošných spojů. Návrh vychází z doporučeného zapojení obvodu RTL8019AS a procesoru ATmega128. Na internetu jsem získal několik dalších referenčních zapojení pro oba obvody, které stačilo aplikovat a připravit potřebné podklady pro výrobu. Pro návrh schématu jsem si zvolil produkt Eagle [22].

V porovnání s projektem Ethernut není tento web server postaven na čtyřvrstvém plošném spoji, ale pouze dvouvrstvém, je rozšířen o rozhraní pro MMC kartu. Protože MMC karta musí být napájena napětím 3.3V, jsou na desce nutné dva rozvody pro dvě různé úrovně napětí 5V a 3V3. Vzhledem k tomu, že deska je pouze dvouvrstvá a obsahujet šachtu pro MMC a její další podpůrné obvody, došlo k částečnému zvětšení plochy desky plošných spojů.

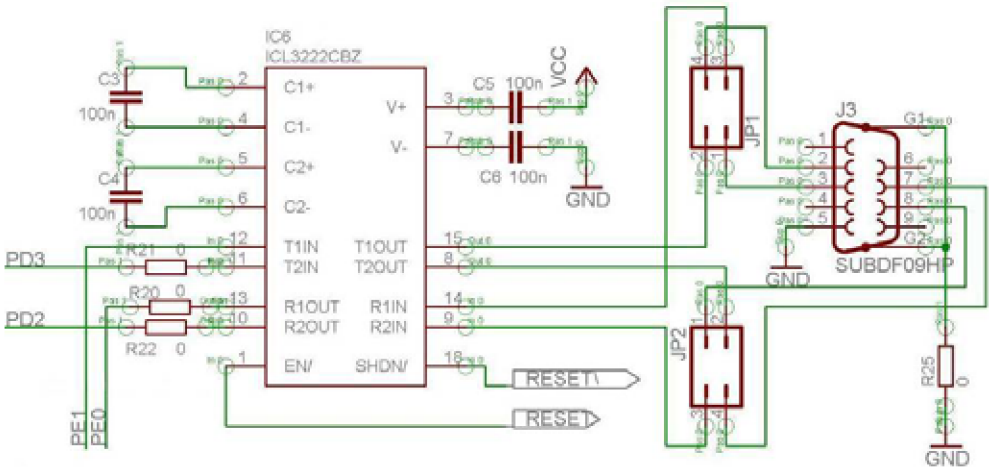
Kompletní obvodová schémata, návrh plošného spoje, výrobní data plošného spoje a seznam použitých součástek jsou součástí přílohy.

5.1 CPU

Hodinový takt mikrokontroléru ATmega128 je generován 14,7456 MHz krystalem (Y1), který by mohl být nahrazen krystalem s frekvencí až do 16 MHz. Další 32,768 kHz krystal (Y2) řídí vnitřní asynchronní časovač, který slouží jako realtime softwarové hodiny. Manuální reset procesoru je řízen prostřednictvím obvodu MAX825 (IC10). Níže jsou dále podrobněji presentována všechna připojená rozhraní procesoru.

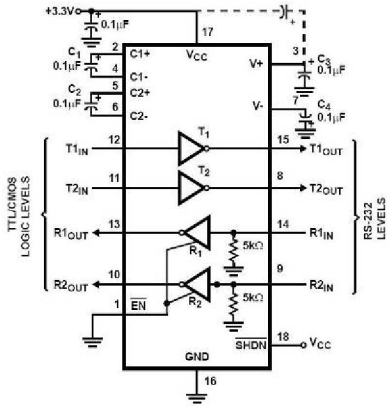
5.1.1 Rozhraní RS232

RS232 je rozhraní pro sériový přenos informací vytvořené původně pro komunikaci dvou zařízení do vzdálenosti 20m. Pro větší odolnost proti rušení je informace po propojovacích vodičích přenášena větším napětím, než je standardních 5V. Pro dnešní počítače kde již toto rozhraní téměř vymizelo, existují převodníky RS232/USB. Protože USART procesoru ATmega používá standardní 5V úrovně, musí se pro komunikaci s PC použít převodník úrovní na RS232.



Obrázek 32: *Schéma zapojení sériového portu*

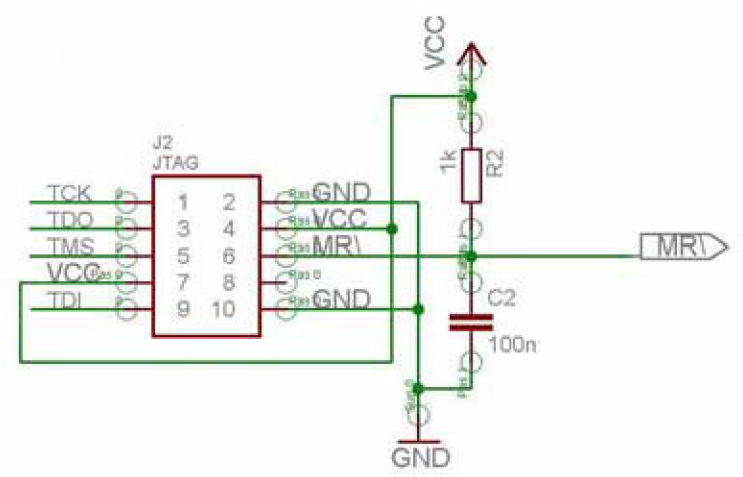
Logické úrovně TTL a RS232 jsou mezi sebou převáděny obvodem ICL3222 (IC6). Obvod ICL3222 v sobě obsahuje dvojici převodníků TTL/RS322, dvojici převodníků RS322/TTL a měnič – viz. Obrázek 33. Zabudovaný měnič napětí na principu nábojové pumpy potřebuje ke své činnosti pouze čtyři externí kapacitory C3 až C6. Linka se připojuje přes standardní konektor CANON9.



Obrázek 33: *Schéma obvodu ICL3222CBZ*

5.1.2 JTAG rozhraní

JTAG rozhraní je vyvedeno dle požadavků výrobce čipu ATmega128. Je připojeno na signály TDI (Test Data In), TDO (Test Data Out), TMS (Test Mode Select), TCK (Test Clock), které jsou součástí rozhraní mikrokontroléru PortF [12]. Aby přes toto rozhraní bylo možné monitorovat externí RESET/ signál je připojen i signál MR/. Kondenzátor C2 a rezistor R2 slouží k zamezení vzniku nevyvolaného resetu.

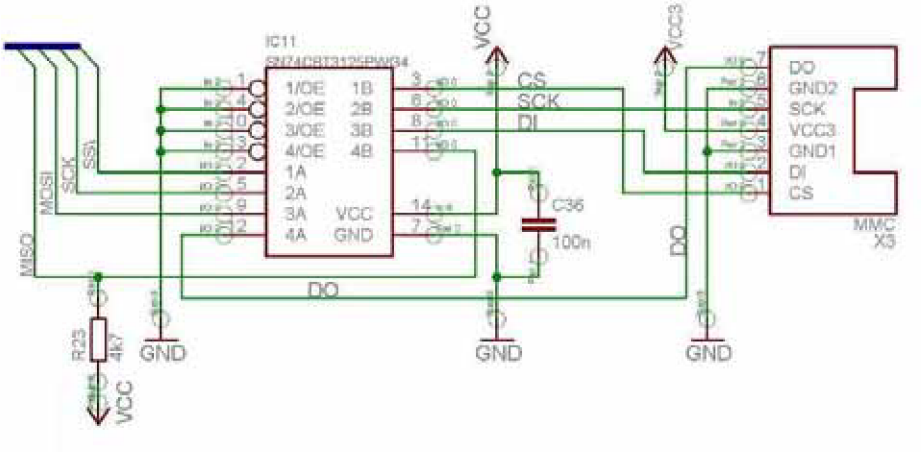


Obrázek 34: *Signály připojené ke konektoru JTAG*

5.1.3 MMC rozhraní

MMC karta může být provozována ve dvou módech Multimedia Card Mode a Serial Peripheral Interface (SPI) Mode, v tomto případě je zapojena v SPI módu. Na konektor pro kartu MMC jsou přivedeny signály SPI rozhraní mikrokontroléru PortB. SPI rozhraní je tvořeno čtyřmi signály SS (*Slave Select*), SCK (*SPI Bus Serial Clock*), MISO (*SPI Bus Master Intput/Slave Output*), MOSI (*SPI Bus Master Output/Slave Input*).

Protože logika MMC karty používá napěťovou úroveň 3V3, bylo nutné mezi mikrokontrolér a MMC konektor umístit převodník napěťových úrovní SN74CBT3125 (IC11). Zapojení konektoru a převodníku napěťových úrovní presentuje Obrázek 35.



Obrázek 35: *Zapojení MMC slotu a převodníku napěťových úrovní*

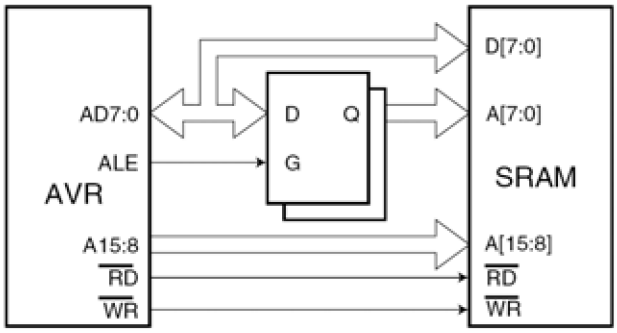
5.2 Rozšiřující porty

Aby možnosti využití web serveru byly mnohem větší než pouhá komunikace přes sériové rozhraní nebo rozhraní Ethernet a aby byla možnost využít všechny komunikační porty a periferie mikrokontroléru ATmega128, byly na desce prostřednictvím konektoru SV1 vyvedeny PortD, PortB, PortE, adresová sběrnice, datová sběrnice, řídící signály WR/ (*Write Strobe*), RD/ (*Read Strobe*), ALE (*Address Latch Enable*), napájení 5V, napájení 3V3 a GND. Dále jsou na rozšiřující port připojeny signály MR/ a RESET/.

Pro vyvedení analogových vstupů A/D převodníků procesoru, tedy PortF slouží konektor J1. Toto rozhraní dále obsahuje napájení 5V a signál AREF, což je analogový kontrolní pin pro A/D převodníky. Tabulky s detailním zapojením jsou uvedeny v příloze.

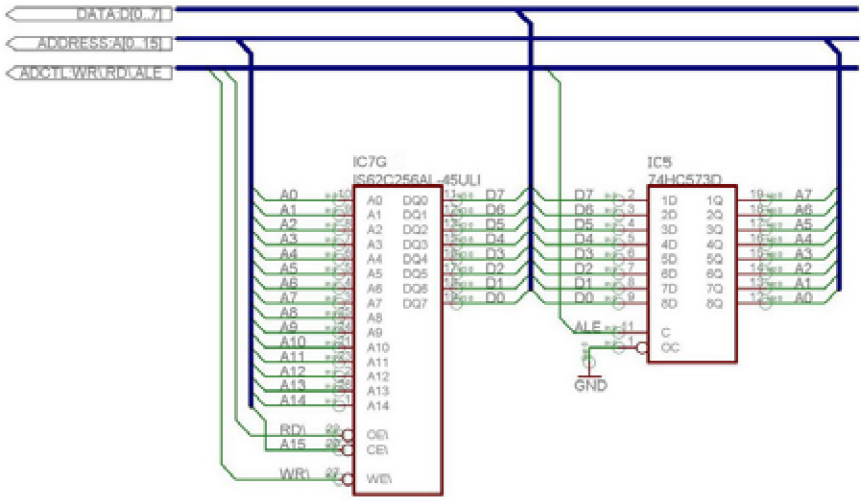
5.3 SRAM

Externí paměť SRAM IS62C256AL (IC7) je řízena pomoci sdílené adresové a datové sběrnice kde dolních 8 bitů je společných jak pro data tak pro adresu. Proto bylo nutné do zapojení vložit pomocné D-buffery 74HC573D (IC5).



Obrázek 36: *Připojení externí SRAM k ATmega128 [12]*

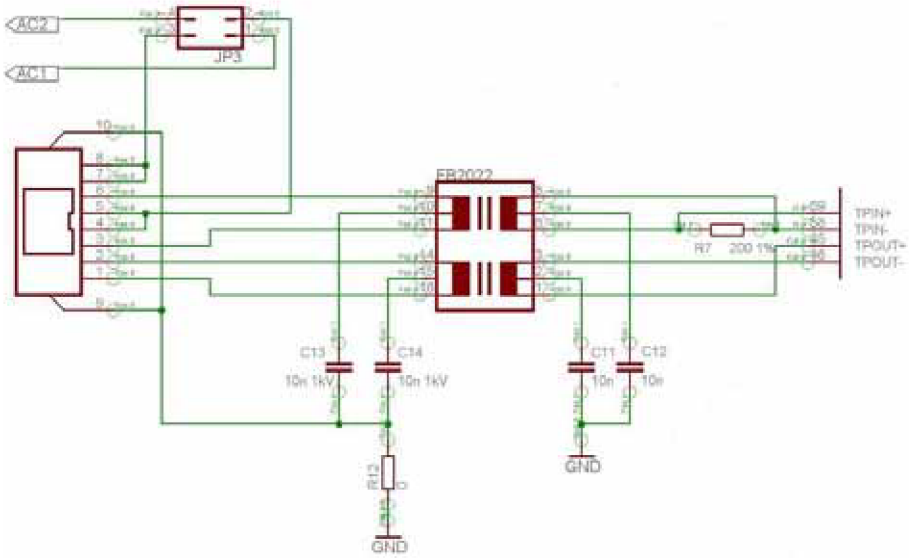
Na straně procesoru je pro multiplexovanou adresovou a datovou sběrnici vyhrazen PortA a pro horních 8 bitů adresy pak PortC. Paměť je dále řízena signály RD/ (*Read Strobe*),WR/ (*Write Strobe*), ALE (*Address Latch Enable*).



Obrázek 37: *Zapojení klopných obvodů D 74HC573D*

5.4 Ethernet rozhraní

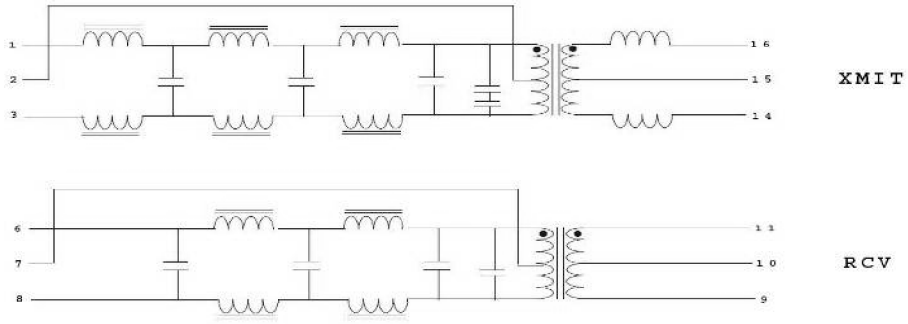
Síť Ethernet se k zařízení připojuje pomocí standardního konektoru RJ45, který je v tomto případě proveden se stíněním. Obvod RTL8019AS [13] v sobě obsahuje veškeré digitální a analogové obvody mimo vstupních/výstupních filtrů a oddělovacích transformátorů. Proto je nedílnou součástí zapojení impulsní transformátor s hybridním filtrem, jehož úkolem je budit linku Ethernet a zároveň dodávat dostatečně kvalitní signál pro další zpracování řadičem. Impulsní transformátor zde též vytváří galvanické oddělení Ethernetu a linky RS232, která je galvanicky spojena se zbytkem zařízení. Doporučuje se použít obvod FB2022.



Obrázek 38: *Schéma zapojení obvodu FB2022*

Jak uvádí Obrázek 39, obvod obsahuje dvojici dolních propustí s frekvencí 17MHz a dva oddělovací transformátory. Toto zapojení slouží k odstranění vyšších harmonických vznikajících vlivem toho, že výstup z obvodu má digitální charakter a je potřeba mu "zaoblit" hrany tak, aby co

nejvíce připomínal sinusovku a ne obdélník. Odfiltrováním harmonických signálů se přes UTP kabeláž přenáší pouze užitečný signál a nikoliv nežádoucí rušení.



Obrázek 39: *Schéma obvodu FB2022*

Kapacitory C11 až C14 slouží k potlačení nesymetrických rušivých signálů a uzemňují pro střídavý signál středy oddělovacích transformátorů.

Přijímaný signál prochází z konektoru RJ45 (X1) přes hybridní filtr přímo do obvodů řadiče. Přijímač je na straně řadiče impedančně zakončen pomocí odporu R7.

5.5 Řadič Ethernetu

Jako řadič sítě Ethernet je v zařízení použit obvod RTL8019AS (IC2). Byl běžně používán v síťových kartách počítačů PC a umožňuje komunikovat dle standardu IEE802.3a (10Base2) nebo IEE802.3i (10Base-T). V tomto případě byl zvolen standard 10Base-T, neboť propojení koaxiálním kabelem dle 10Base2 je již zastaralé. UTP kabeláž je navíc kompatibilní se standardem 100Base-T. Dále obvod umožňuje režim komunikace v plném duplexu a podporuje režim sníženého odběru energie.

Celé zapojení řadiče RTL8019AS i s pomocnými obvody vychází z dokumentace výrobce obvodu a jeho aplikačních poznámek. Jelikož je použit osmibitový řídící procesor, je i samotný řadič konfigurován do osmibitového režimu uzemněním pinu IOCS16 přes odpor R9.

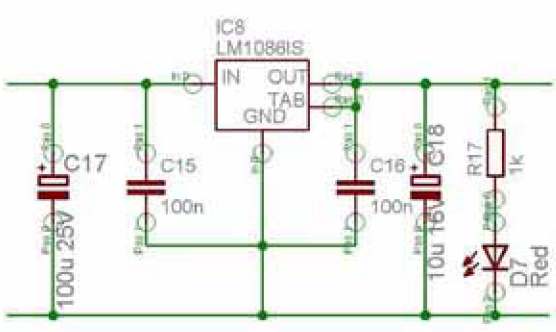
Obvod RTL8019AS je v minimalizovaném zapojení, tzn. bez externí paměti EEPROM 93C46, která se používá pro uložení MAC adresy, konfigurace, atd. V tomto řešení jsou tyto údaje uloženy přímo v paměti procesoru. Komunikace s řadičem probíhá v 8-bitovém režimu, umožňující adresovat 8 kB SRAM integrované uvnitř řadiče. Obvod je konfigurován do osmibitového režimu uzemněním pinu IOCS16 přes odpor R9. Paměť je určena pro příjem a vysílání Ethernetových rámců. Po přijetí Ethernetového rámce, řadič automaticky generuje požadavek o přerušení pro mikroprocesor ATmega128. Maximální dosažitelná rychlost činí 10 Mb/s.

Vizuální sledování stavu komunikace na rozhraní Ethernet umožňují LED diody D1 a D2 připojené k napájení přes odpory R5 a R6. Dioda D1 (žlutá) indikuje stav linky (svítí když je vše v pořádku), D2 (zelená) indikuje přenos po médiu. Signály pro LED generuje sám řadič. Obvod má samostatný externí oscilátor (Y2), je to krystal s frekvencí 20MHz a kapacitory C9 a C10.

5.6 Napájení

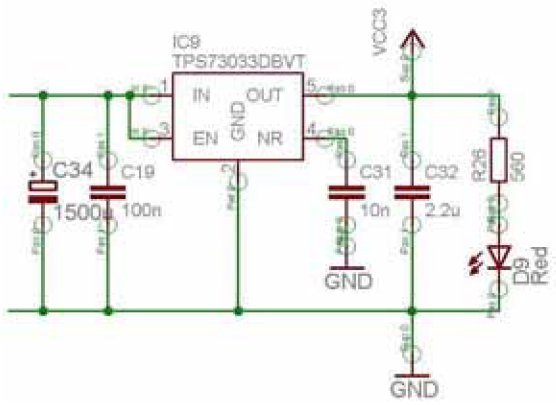
Logika desky je řízena napájením 5V a 3.3V. Deska je proto vybavena dvěma vlastními stabilizátory napětí (IC8 a IC9).

Deska potřebuje nestabilizovaný zdroj DC 8-12V , který snese zatížení proudem 400mA.. Zdroj se připojuje přes standardní konektor 2.1mm. Tento vstup je chráněn pojistkou F1 a usměrňovacím můstkem postaveným z Schottkyho diod D3 až D6. Dále je napájecí napětí vedeno na filtrační elektrolytický kapacitor C17. Obvod LM1086IS (IC8) je v doporučeném zapojení uváděném výrobcem, C15 a C16 blokují celé zapojení před zákmity. Kapacitor C18 souží jako hlavní výstupní filtrační kapacitor.



Obrázek 40: *Zapojení stabilizátoru LM1086IS pro 5V*

Druhý stabilizátor TPS73033DBVT (IC9) je výhradně určen pro napájení MMC karty. Je zapojen podle doporučení výrobce. Na vstupu proti GND je připojen kapacitor C19, který slouží pro zajištění stability, lepší zákmitové charakteristiky, potlačení šumu a vlnění. Pro stabilizaci vnitřního kontrolního cyklu je mezi výstup a GND připojen výstupní kapacitor C32. Mezi pin NR a GND pak ještě kapacitor C31 který spolu s vnitřním rezistorem tvoří filtr typu dolní propust pro zmírnění nežádoucího rušení.



Obrázek 41: *Zapojení stabilizátoru TPS730DBVT pro 3V3*

Napájet desku je umožněno prostřednictvím použití pinů 4,5 a pinů 7,8 konektoru RJ45. V tomto případě musí být zkratovány piny 1 a 3 a piny 2 a 4 přepínače JP3. Stejně jako v případě konektoru 2.1mm je i tento vstup chráněn pojistkou F1 a proti přepólování usměrňovacím můstkem. Pro napájení přes Ethernetový kabel je potřeba speciální zdroj napětí. Bez připojení takového zdroje

není vhodné zkratovat přepínač JP3, protože by mohlo dojít ke zničení zařízení připojených na linku Ethernet.

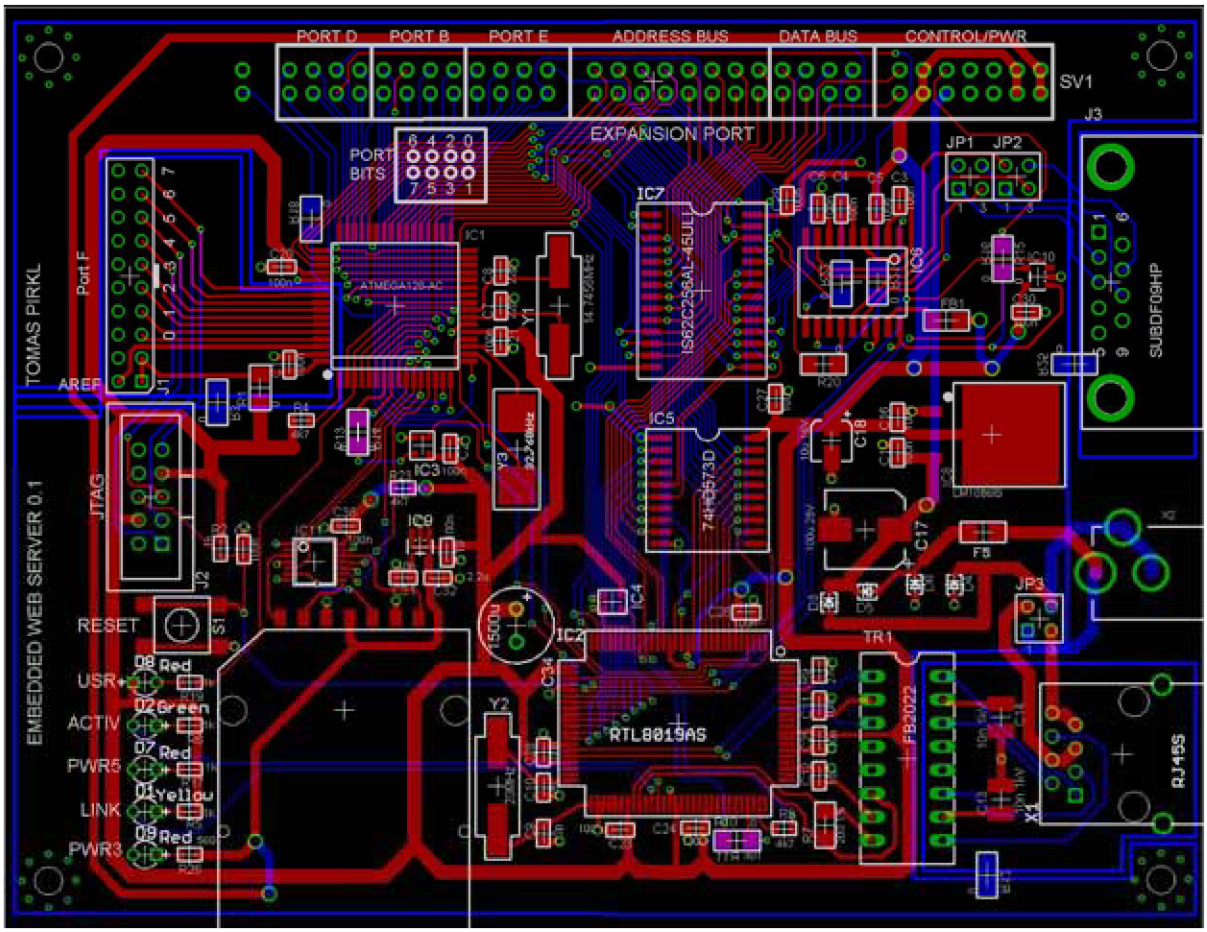
Další možnost napájení, kterou tato deska má je připojení DC signálu na rozšiřující port SV1. Takto lze napájet samotnou desku nebo zařízení k ní připojené. Tento vstup není nijak chráněn, špatná polarita nebo vysoké napětí může zařízení zničit.

Vzhledem k delšímu rozvodu napájení po povrchu desky plošných spojů jsem se rozhodl do rozvodu 5V umístit, jako další stabilizaci napětí, radiální elektrolytický kapacitor C34 s nízkou impedancí. Každý obvod na desce je také vybaven svými blokovacími kapacitory. Jakmile je napájení připojeno k jakémukoli vstupu popsanému výše, bude to signalizovat svítivá dioda D7 pro 5V a dioda D9 pro 3,3V. Proud diodami je omezen pomocí do série zapojených rezistorů R26 a R17.

5.7 Návrh a výroba DPS web serveru

Poté co jsem vytvořil schéma zapojení s ohledem na dostupnost jednotlivých komponent v maloobchodní síti, jsem přistoupil k návrhu DPS. Vzhledem k nákladům na výrobu jsem se rozhodl pro dvouvrstvé řešení.To bylo přibližně dvakrát levnější než deska se čtyřmi vrstvami. Plošné spoje byly navrhnuty ručně bez použití autorouteru v prostředí programu Eagle [22].

Vrchní vrstva plošného spoje je kromě signálových rozvodů použita ještě k rozvodu napájení VCC 5V a VCC 3V3. Na spodní vrstvě je pak metodou rozlití mědi rozvedena zem GND. Rozměry celé desky web serveru jsou přibližně 13cm x 10cm. Desku plošných spojů vytvořenou v prostředí Eagle s rozmístěním jednotlivých komponent presentuje následující Obrázek 42.



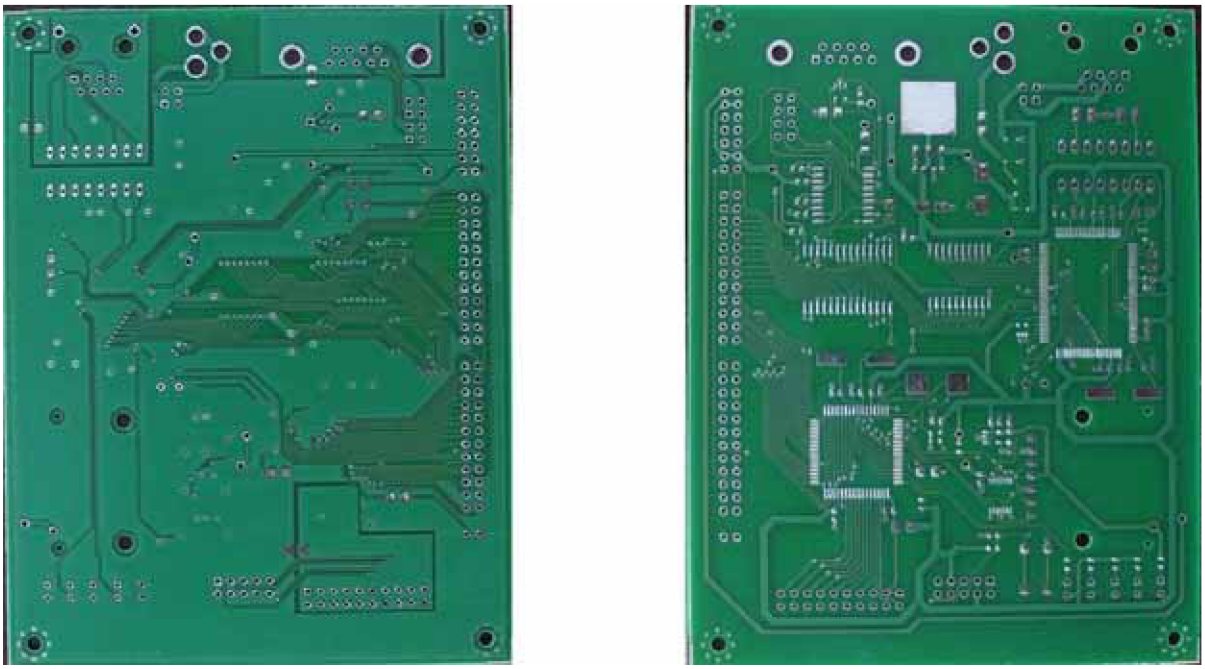
Obrázek 42: *Navrhnuté plošné spojů.*

Pro bližší specifikaci výroby ve firmě Gatema s.r.o. pak byla použita dále uvedená výrobní data. Jde o plošný spoj oboustranný, prokovený s vrtanými otvory po prokovení ještě dovrtanými. Pro výrobu desky byl použit materiál Lamplex FR4 firmy Lamitech Czech s.r.o. tloušťky 1,5 mm s naplátovanou mědí tloušťky 35 um.

Vodiče, pájecí očka i otvory jsou pokryty žárově nanášenou pájkou metodou HAL (Hot Air Levelling). Jedná se o horkovzdušné nanášení roztavené pájky Sn63Pb37 na povrch spojů a otvorů. Deska se ponoří do roztavené pájky a při vytahování je z obou stran ofukována horkým vzduchem.

Nepájivá maska je dvousložková fotocitlivá. Byla nanášena sítotiskem na měděný povrch z obou stran desky. Nepájivá maska slouží jako ochrana těch částí desky, které nemusí být pájeny.

Deska prošla při výrobě elektrickým testem a jednotlivé kusy desek byly odděleny stříháním. Výsledný výrobek uvádí Obrázek 43. Jak můžete vidět tak na výsledné DPS chybí servisní potisk. Došlo k tomu mou nezkušeností v oblasti a chybnou komunikací s firmou Gatema.

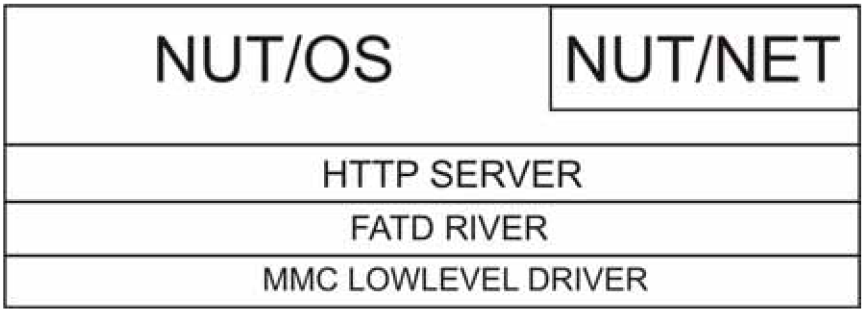


Obrázek 43: *Neosazená DPS*

6 Softwarové vybavení web serveru

Po hardwarové realizaci jsem postoupil k realizaci softwarového vybavení web serveru. Rozhodl jsem se pro řízení tohoto zařízení použít volně dostupný operační systém NutOS šířený pod GNU GPL licencí. Tento operační systém pochází od firmy Egnite a byl navrhnut přímo pro embedded zařízení s mikrokontroléry řady AVR. Výhodou Nut/OS jsou zdarma poskytnuté nástroje ne nepodobné standardním operačními systémům a tím i pohodlí a nebývalá rychlost vývoje aplikací oproti projektům, kde není žádný operační systém použit.

Protože hardware, který jsem navrhl není díky již uvedeným rozšířením plně kompatibilní s NUT/OS, bylo nutné navrhnout chybějící ovladače a upravit již implementovanou aplikaci HTTP serveru. Co se týká softwaru pro práci s MMC kartou bylo nutné implementovat ovladač MMC karty umístěné na SPI sběrnici v SPI režimu a ovladač pro práci se souborovým systémem FAT.



Obrázek 44: *SW vrstvy systému*

6.1 NUT/OS

Při tvorbě operačního systému Nut/OS autoři vycházeli z projektu Liquorice vytvořeného Davem Hudsonem. Nut/OS je napsán v jazyku C pro překladač AVR GNU C a ImageCraft AVR . Vše co potřebuje ke své činnosti, je součástí jeho zdrojových kódů, což je nesmírnou výhodou. Jedná se o více-úlohový operační systém s kooperativním multitaskingem a systémem řízení priorit. Je podporovaná dynamická správa paměti. Jednotlivé úlohy lze dynamicky vytvářet a ukončovat za běhu programu. Dále je implementována podpora časovačů, systém zpráv, fronty událostí a jednoduchý systém souborů, který jsem se rozhodl nahradit souborovým systémem FAT. Přístup k periferiím mikroprocesoru je řešen pomocí ovladačů jednotlivých zařízení (USART, RTL8019, atd.). Programátor má k dispozici služby operačního systému nad kterými může relativně snadno vytvářet vlastní aplikace.

Bližší informace k systému a jeho implementaci naleznete v [24] a [25].

NUT/NET

Nut/Net je nadstavbou Nut/OS a představuje implementaci TCP/IP protokolového zásobníku. Na rozdíl od osobních počítačů je limitován dostupnou výpočetní a paměťovou kapacitou. I přes to Nut/Net podporuje práci s UDP a TCP sokety. V současné době jsou implementovány následující protokoly ARP, IP, ICMP, UDP, TCP, TELNET, DHCP, DNS, HTTP, TFTP, PPP.

Nut/Net obdobně jako Nut/OS poskytuje aplikační rozhraní, které využívá programátor při vytváření vlastní aplikace.

6.2 Vývoj aplikací

Vývoj aplikací se provádí na platformě MS Windows nebo Linux. Pro obě platformy je k dispozici instalační balík obsahující zdrojové kódy, dokumentaci a programy nutné k okamžitému zahájení vývoje. Je na výběr ze dvou kvalitních kompilátorů, volně dostupného AVR GNU C nebo komerčního ImageCraft AVR.

Na desce web serveru je vyveden konektor pro JTAG programování. K samotnému nahrání firmwaru slouží programátor PRESTO a jeho aplikace JTAG player for Presto. Konkrétní postup instalace, konfigurace a kompilace operačního systému NUT/OS i s jeho aplikacemi je popsán v manuálu, který je součástí přílohy a v [23][23].

6.3 Implementace ovladače pro MMC

Implementované funkce pro inicializaci karty, sběrnice SPI a funkce pro datovou komunikaci jsou uloženy ve zdrojovém souboru `mmcdrv.c` a hlavičkovém souboru `mmcdrv.h`. Při implementaci jsem vycházel z informací v kapitole 2.4.

`int MMCInit()` Tato rutina provádí konfiguraci portu SPI a inicializaci MMC karty.

Inicializace SPI začíná nastavením CS na logickou jedničku (deaktivujeme kartu), dále nastavením pinů na vstupní nebo výstupní v registru DDRB (Port B Data Direction Register). Následně aktivuje SPI rozhraní příkazem SPIE (SPI Enable) v SPCR (SPI kontrol Register) registru, dále se vybere režim master pomocí příznaku MSTR (Master/Slave) v SPCR a ještě se zvolí pracovní frekvence (ponechána na maximu) pomocí příznaků SPR0, SPR1 (SPI Clock Rate Select 0 a 1) v registru SPCR a příznaku SPI2X (Double SPI Speed Bit) v registru SPSR (SPI Status Register).

Následně uvolní vstupní buffer karty. Vybere kartu nastavením CS na logickou nulu (aktivujeme kartu). Provede reset karty zasláním příkazu CMD0 za kterým následuje zasílání příkazu CMD1. Jakmile se nám karta ozve nastavíme opět CS na logiskou jedničku.

int MMCMountAllDevices() – Funkce, která načte z registru CSD (Card Specific Data register), potřebná data pro přístup k datům na kartě (velikost a počet sektorů) do struktury typu DRIVE.

int MMCGetSectorSize() – Ze struktury typu DRIVE, zjistí a vrátí velikost sektoru.

DWORD MMCGetTotalSectors() – Ze struktury typu DRIVE zjistí a vrátí počet sektorů.

int MMCReadSectors() – Funkce čte pomocí příkazu CMD17 v parametru předaný počet sektorů.

6.4 Implementace souborového systému FAT

Při řešení jsem vycházel z informací uvedených v kapitole 2.5. Implementovaný blok funkcí pro čtení ze souborových systémů FAT16 a FAT32 je v souborech fat.c, fat.h a fatdrv.h. K implementaci zápisu souborů do systému FAT už jsem nepřistoupil. Struktury, které jsem pro práci s FAT vytvořil, jsou:

_FAT32FileDataTime – Je struktura pro datum a čas vytvoření souboru.

_FAT32DirectoryEntry – Je struktura pro jednu položku adresáře.

_FAT32DirectoryEntryLong – Je struktura pro jednu položku adresáře s dlouhým jménem.

_FAT32PartitionEntry – Je struktura pro vlastnosti oddílu v MBR.

_FAT32PartionTable – Je struktura pro uložení Master Boot Rekord.

_bpbfat16 – Je struktura pro Bios Parameter Block pro FAT16.

_bpbfat32 – Je struktura pro Bios Parameter Block pro FAT32.

_FAT32BootRecord – Je struktura pro uložení Boot Rekordu.

_fat_entry_table16 – Je struktura určující počet položek na sektor pro FAT16.

_fat_entry_table32 – Je struktura určující počet položek na sektor pro FAT32.

_drive_info – Je struktura informací o zařízení.

_fhandle – Je struktura pro uchování vlastností souboru.

Funkce, které s těmito strukturami pracují, jsou uvedeny dále.

static void Mount() – Zjistí pomocí funkcí ovladače MMC karty její parametry.

static void UnMount() – Uvolní všechny získané informace pro práci s MMC kartou.

static int FATInit() – Přidělí procesu vlákno. Inicializuje MMC kartu pomocí volání fukcí ovladače.

static DRIVE_INFO* GetDriveByDevice () – Načte informace potřebné pro práci s MMC kartou do struktury typu NUTDEVICE.

`static NUTFILE *FATFileOpen()` – Otevře existující soubor pro čtení. Naplní strukturu typu `FHANDLE` základními vlastnostmi souboru. Vrací ukazatel na strukturu typu `NUTFILE`, do které přiřadí zařízení na kterém se soubor nachází a parametry souboru ze struktury `FHANDLE`.

`static int FATFileClose()` – Uzavře soubor dříve otevřený voláním `FATFileOpen()`.

`long FATFileSize()` – Vrátí velikost souboru v bytech, kterou vyčte ze struktury `NUTFILE`.

`static DWORD FindFile()` – Prochází adresářovou strukturou a hledá soubor s daným jménem.

`int FATFileRead()` –Dokud se nedostane na požadovaný počet přečtených bytů z dat souboru, tak přečte jeden sektor a následně zjišťuje kolik bytů z něj bude ještě požadovat, zároveň testuje úmysl číst z dalšího sektoru a příznak konce souboru. Pokud se dostane na konec clusteru načítá další funkcí `GetNextCluster()`.

`int FATIOCtl()` – Zpřístupňuje implementované funkce driveru operačnímu systému NUT/OS.

6.5 Aplikace HTTP server pro NUT/OS

Softwarová implementace aplikace HTTP serveru se díky použití RTOS NUT/OS a jeho nadstavby NUT/NET provedla jako úprava již implementované ukázkové aplikace, která využívá schopností NUT/OS a NUT/NET. Implementace aplikace HTTP serveru je v souboru `httpserv.c`. Provedl jsem nalinkování knihovny Fat.h, dále registraci kořenového adresáře pro HTTP server a registraci nových ovladačů pro FAT a MMC. Tím se nové periferie inicializují a přidají do systémového seznamu zařízení.

7 Oživení a testování web serveru

Následovalo testování funkčnosti zařízení a jeho oživení. Ještě před samotným prvním spuštěním jsem po připojení web serveru k napájení testoval správnou úroveň napětí na všech napájecích pinech důležitých komponent na desce.

Následně už jsem přistoupil k testování funkčnosti ve spojení s operačním systémem NUT/OS. Bylo nutné nakonfigurovat NUT/OS pro můj HW, potom následovalo sestavení knihoven pro NUT/OS. Pak už stačilo jen z již hotových připravených příkladů k testování použít aplikaci BaseMon, která je schopna postupně testovat komunikaci po lince RS232, SRAM paměť, Řadič Ethernetu a síťové rozhraní.

Bližší informace k oživení a testování web serveru jsou v manuálu, který je součástí příloh. Instalace a konfigurace NUT/OS je popsána v [23].

K prvnímu testování celého systému jsem díky HW kompatibilitě mohl použít aplikaci BaseMon, která byla vytvořena pro testování HW projektu Ethernut. Ta ověřuje funkčnost základních bloků hardwaru. Jak je vidět níže na výpisu z terminálu připojeného k desce nejdříve testuje Externí paměť, posléze síťové rozhraní a vstupní/výstupní porty.

```
BaseMon 4.2.0
Nut/OS 4.3.2.1 beta
Compiled by AVRGCC for ATmega128
Baudrate select = 7
External RAM Test... 28416 bytes
Detecting NIC...     RTL8019AS
Testing NIC...       OK
I/O Port Test...     OK
```

Další výkonnostní testy jsem chtěl provést před odevzdáním diplomového projektu, ale web server začal vykazovat velmi nestabilní chování, které je způsobeno s největší pravděpodobností nějakou hardwarovou chybou. Projevuje se to, jak v mé aplikaci pro čtení dat z MMC, tak v testovací aplikaci BaseMon a to vždycky jiným typem chyby. K těmto chybám začalo docházet po častém nahrávání firmwaru a velkém počtu restartů celého systému.

Z výše uvedených důvodů budu muset opět přistoupit k testování funkčnosti HW. Příčinou by mohlo být chybné napájení, jeho úrovně už jsem překontroloval. Ale problém s napájením by mohl být v šumu nebo v nějakém rušení. Protože už jsem neměl přístup potřebnému vybavení nemohl jsem to zatím ověřit. Dalším možným problémem by mohl být resetovací obvod. Ze stejných důvodů jako v předchozím případě nejsem momentálně schopen to ověřit. Další chyba, která by mohla být příčinou takového chování je chybný rozvod některého signálu nebo vadný podpůrný obvod.

8 Závěr

Během této práce jsem se seznámil s implementací moderních internetových technologií do Embedded zařízení. Dále jsem nastudoval síťové referenční modely ISO/OSI a TCP/IP, standard Ethernet a komunikační protokoly UDP, TCP, IP a HTTP. Navrhl jsem platformu pro implementaci vestavěného web serveru a vybral vhodné komponenty pro účely implementace web serveru na desce plošných spojů. Také jsem vyhledal hodné vývojové nástroje pro návrh softwaru.

V dalším kroku projektu jsem navrhl konkrétní zapojení, v maloobchodní síti obstaral všechny komponenty a podpůrné obvody. Potom bylo nutné nechat vyrobit desku plošných spojů a tu posléze osadit. Následně se přistoupilo k implementaci souborového systému a ovladače pro MMMC kartu pro NUT/OS. Dalším krokem bylo oživení, testování a na konec zhodnocení celého systému.

Podařilo se mi tedy postavit web server na desce plošných spojů, který vychází z původního projektu Ethernut. Moje verze se liší počtem použitých vrstev na desce plošných spojů. Oproti původním čtyřem u Ethernutu používám pouze dvouvrstvou desku. To znamenalo komplikace při návrhu DPS. Kromě signálových spojů jsem musel navrhnout nové cesty pro napájení na horní straně DPS a nové cesty pro zem na straně spodní. Použití dvou vrstev znamenalo snížení nákladů na výrobu, ale na druhou stranu to způsobilo částečný nárůst velikosti plochy DPS. Zvýšení plochy DPS je také způsobeno i rozšířením web serveru o rozhraní pro MMC kartu. Na desce přibyla šachta pro MMC kartu, stabilizátor napájení pro 3V3 a kromě dalších podpůrných obvodů ještě převodník napěťových úrovních pro komunikaci mezi MMC kartou a mikrokontrolérem.

S tímto rozšířením potom souvisí i návrh softwarového vybavení. Pro NUT/OS v kombinaci s mikrokontrolérem ATmega128 jsem implementoval ovladač pro MMC kartu a dále ještě knihovnu funkcí pro čtení ze souborového systému FAT.

Díky tomuto rozšíření značně vzrostla kapacita pro http stránky nebo pro jakákoli jiná data zpracovávaná tímto systémem. Také se prodlouží životnost FLASH paměti mikrokontroléru tedy mikrokontroléru ATmega128 jako celku, protože pro změnu stránek už není nutné kompilovat a znovu nahrávat celou aplikaci, tak jak tomu bylo v případě původního projektu Ethernut.

Budoucnost tohoto projektu vidím v připojování nejrůznějších periferií, jako jsou webové kamery, teploměry nebo jakákoli jiná zařízení u který bude třeba s nimi pracovat vzdáleně přes rozhraní Ethernet.

Při dalším vývoji bych se zaměřil na dokončení knihovny pro práci se souborovým systémem FAT a to hlavně na podporu zápisu na MMC kartu. Pak bude potřeba odstranit nestabilní chování obvodu, které začal náhle zatím z neznámých důvodů vykazovat.

Literatura

[1] Schwarz, J., Růžička, R., Strnadel, J.: Mikroprocesorové a vestavěné systémy. Studijní opora. Brno, leden. 2006.

[2] Wikipedia. Embedded systém [online]. 2007. Dokument dostupný na URL http://cs.wikipedia.org. (leden 2007)

[3] Osterloh, Heather: TCP/IP Kompletní průvodce. SoftPress, 2003.

[4] Rosa, Zdeněk: Principy činnosti a technické vybavení počítačových sítí. Pardubice, Systemconsult 1992.

[5] Postel, Jon: Internet Protocol, RFC 791 [online]. USC/Information Sciences Institute, září 1981. Dokument dostupný na URL ftp://ftp.rfc-editor.org/in-notes/rfc791.txt (leden 2007).

[6] Postel, Jon: DOD Standard Transmition Control Protocol, RFC 761 [online]. USC/Information Sciences Institute, leden 1980. Dokument dostupný na URL ftp://ftp.rfc-editor.org/in-notes/rfc761.txt (leden 2007).

[7] Postel, Jon: User Datagram Protocol, RFC 768 [online]. Information Sciences Institute, srpen 1980. Dokument dostupný na URL ftp://ftp.rfc-editor.org/in-notes/rfc768.txt (leden 2007).

[8] Plummer, C. David: An Ethernet Adres Resolutin Protocol, RFC 826 [online]. DCP@MIT-MC, listopad 1982. Dokument dostupný na URL ftp://ftp.rfc-editor.org/in-notes/rfc826.txt (leden 2007).

[9] Fielding, R., Gettys, J., Mogul, J., Frytysk, H., Masinter, L., Leach, P., Berners-Lee, T.: Hypertext Transfer Protokol, RFC 2616 [online]. Network Working Group červen 1999. Dokument dostupný na URL ftp://ftp.rfc-editor.org/in-notes/rfc2616.txt (leden 2007).

[10] Váňa, Vladimír.: Mikrokontroléry ATMEL AVR popis procesoru a instrukční soubor. Praha, BEN – technická literatura 2003.

[11] Šubrt, Vladimír: Mikrokontroléry ATMEL AVR vývoj aplikací. 1993. Praha, BEN – technická literatura 2002.

[12] Atmel Corporation. Atmel ATmega128 Datasheet. 2006 [online]. Dokument dostupný na URL http://www.atmel.com/dyn/resources/prod_documents/doc2467.pdf (kvěren 2007).

[13] Realtek Semi-conductor Corporation. RTL8019AS Specification [online]. 2001. Dokument dostupný na URL http://www.chipcad.hu/download/8019as.pdf (květen 2007).

[14] WinAVR. Oficiální stránka [online]. 2007. Dokument dostupný na URL http://winavr.sourceforge.net/ (květen 2007).

[15] Code Vision. Oficiální stránka [online]. 2007. Dokument dostupný na URL http://www.hpinfotech.ro/index.html (květen 2007).

[16] Atmel Corporation. AVR Studio 4 [online]. 2007. Dokument dostupný na URL: http://www.atmel.com/dyn/products/tools_card.asp?tool_id=2725 (květen 2007).

[17] ASIX. Presto. Oficiální stránka [online]. 2007. Dokument dostupný na URL http://www.asix.cz/a6_presto.htm (květen 2007).

[18] MMCA Technical Committee. MMC Systém Summary [online]. 2005. Dokument dostupný na URL http://www.sandisk.com/Assets/File/OEM/Manuals/ProdManRS-MMCv1.3.pdf (květen 2007).

[19] SanDisk Corporation. MultiMediaCard Produkt Manual [online]. 2005. Dokument dostupný na URL http://www.mmca.org/compliance/buy_spec/MMCA_System_SummaryV41.pdf (květen 2007).

[20] Microsoft. FAT32 File System Specification [online]. 2000. Dokument dostupný na URL http://www.microsoft.com/whdc/system/platform/firmware/fatgen.mspx (květen 2007).

[21] Wikipedia. File Allocation Table [online]. 2007. Dokument dostupný na URL http://cs.wikipedia.org/wiki/File_Allocation_Table (květen 2007).

[22] Eagle. Oficiální stránka [online]. 2007. Dokument dostupný na URL http://www.elcad.cz/eagle/ (květen 2007).

[23] Ethernut. Software manual [online]. 2005. Dokument dostupný na URL http://www.ethernut.de/pdf/enswm24e.pdf (květen 2007).

[24] Ethernut. NUT/OS and NUT/NET API [online]. 2006. Dokument dostupný na URL http://www.ethernut.de/api/main.html (květen 2007).

[25] Ethernut. NUT/OS Threads, Events and Timers [online]. 2002. Dokument dostupný na URL http://www.ethernut.de/pdf/entet100.pdf

Přehled zkratek

ADC (Analog to Digital Converter) - analogově digitální převodník
ALU (Arithmetic Logic Unit) - aritmeticko logická jednotka
ARP (Address Resolution Protocol) - v počítačových sítí s IP protokolem se používá k získání ethernetové (MAC) adresy sousedního stroje
AVR - řada mikrokontrolérů firmy Atmel
CISC (Complex Instruction Set Computer) - architektura procesorů s velkou sadou procesorových instrukcí
CSMA/CD (Carrier Sense Multiple Access / Collision Detection) - přístupová metoda Ethernetu
DOD (Department Of Defense) - referenční komunikační model
DPS (Deska Plošných Spojů)
EEPROM (Electrically Erasable Programmable Read Only Memory) - paměť PROM elektricky mazatelná
EPROM (Erasable Programmable Read-Only Memory) - semipermanentní typ paměti typu Rom-RAM, jejíž obsah je mazatelný ultrafialovým zářením
FLASH - nevolatilní (semipermanentní) paměť typu RAM (s náhodným přístupem), elektricky programovatelná
GCC (The GNU Compiler Collection) - je sada kompilátorů vytvořených v rámci projektu GNU
HDLC (High-level Data Link Control) - asynchronní linkový protokol pro přenos rámců
HTML (Hypertext Markup Language) - značkovací jazyk
HTTP (Hypertext Transfer Protocol) - protokol pro přenos hypertextových dokumentu
I/O (Input/Output) – vstup/výstup
IAP (In-Application Programming) - programovaní mikroprocesoru ATmega128 pomocí zavaděče s podporou protokolu STK500
ICMP (Internet Control Message Protocol) - protokol řídících zpráv internetu
IP (Internet Protocol) - je datový protokol používaný pro přenos dat přes paketové sítě
ISO/OSI (International Standards Organization / Open Systen Interconnection) - referenční komunikační model
ISP (In-System Programmer) - rozhraní, které slouží jak pro programování, tak pro připojení ISP periférií
JTAG - rozhraní programátoru, ale hlavně debugovacího nástroje, který umožňuje reálný pohled do právě běžící aplikace v procesoru
MAC (Media Access Control) - jedinečná adresa síťového zařízení přidělená výrobcem
MCU (Micro Control Unit) - mikrokontrolér, řídící jednotka mikropočítače
MIME (Multipurpose Internet Mail Extensions) - internetový standard pro formátování e-mailů

MMC (MultiMedia Card) – standard pro paměťové karty do mobilních zařízení

OTP (One-Time-Programmable) - jednou programovatelné paměti

PCA (Programmable Counter Array) - programovatelné čítačové pole

PWM (Pulse Width Modulator) - pulzně šířkový modulátor

RAM (Random access memory) - paměť s libovolným (náhodným) přístupem

RISC (Reduced Instruction Set Computer) - procesor s redukovanou instrukční sadou

ROM (Read-Only Memory) - typ paměti, jejíž obsah nelze přepsat běžným způsobem

SMD (Surface Mount Device) - označení součástek pro plošné spoje

SPI (Serial Peripheral Interface) - sériové synchronní programovací rozhraní

SRAM (Static Random Access Memory) - statické RAM, uchovávají informaci v sobě uloženou po celou dobu, kdy jsou připojeny ke zdroji elektrického napájení

TCP (Transmission Control Protocol) - spojově orientovaný protokol pro přenos toku bytů na transportní vrstvě se spolehlivým doručováním

TCP/IP (Transmission Control Protocol/Internet Protocol) - referenční komunikační model

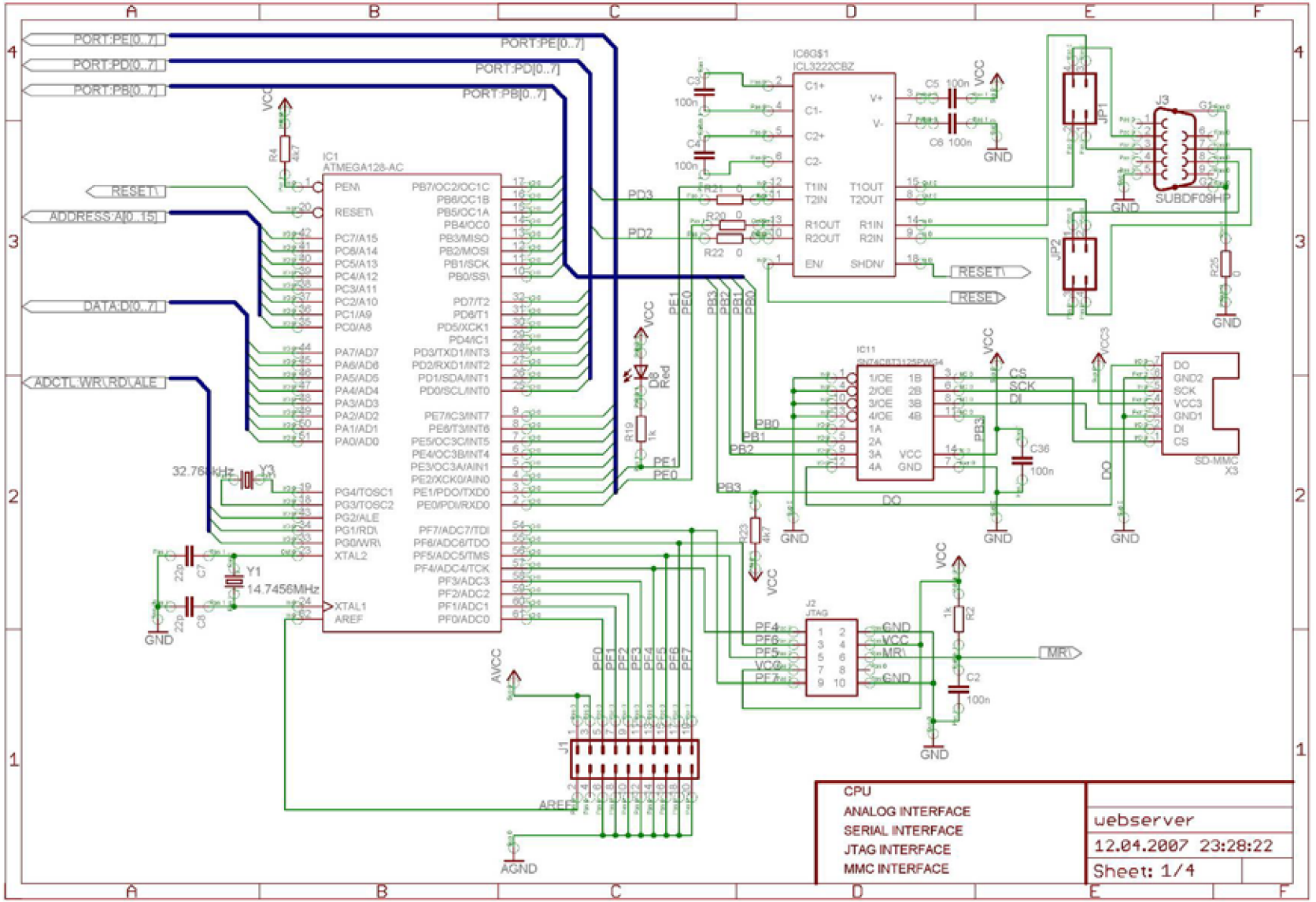
UART (Universal Synchronous and Asynchronous serial Receiver and Transmitter) - rozhraní schopné obousměrné komunikace, a to jak synchronní, tak i asynchronní, s vysokou možnou přenosovou rychlostí

UDP (User Datagram Protocol) - protokol nespolehlivého přenosu zpráv

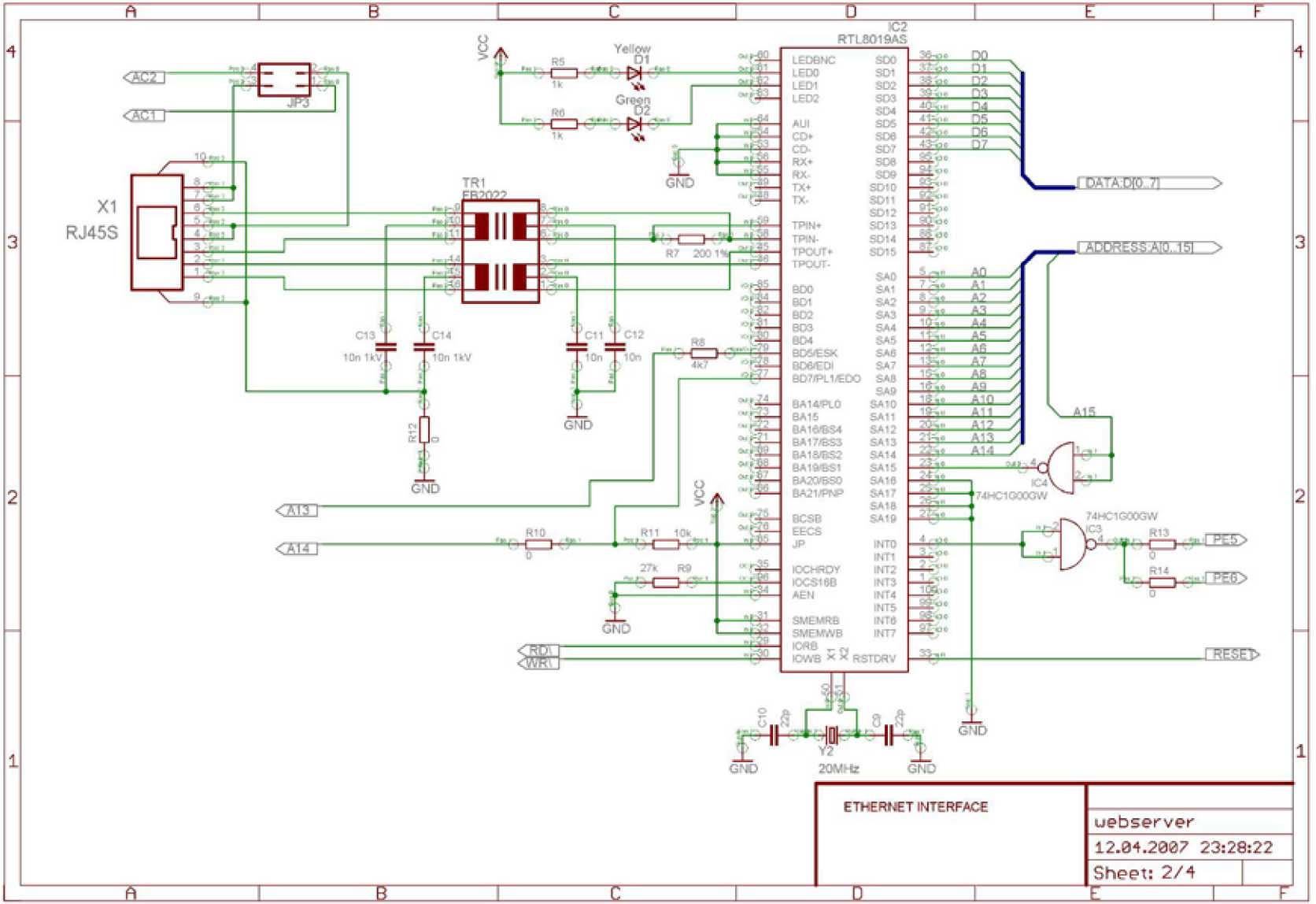
URI (Uniform Resource Identifier) - jednotný identifikátor zdroje

WWW (World-Wide Web) - celosvětová počítačová síť

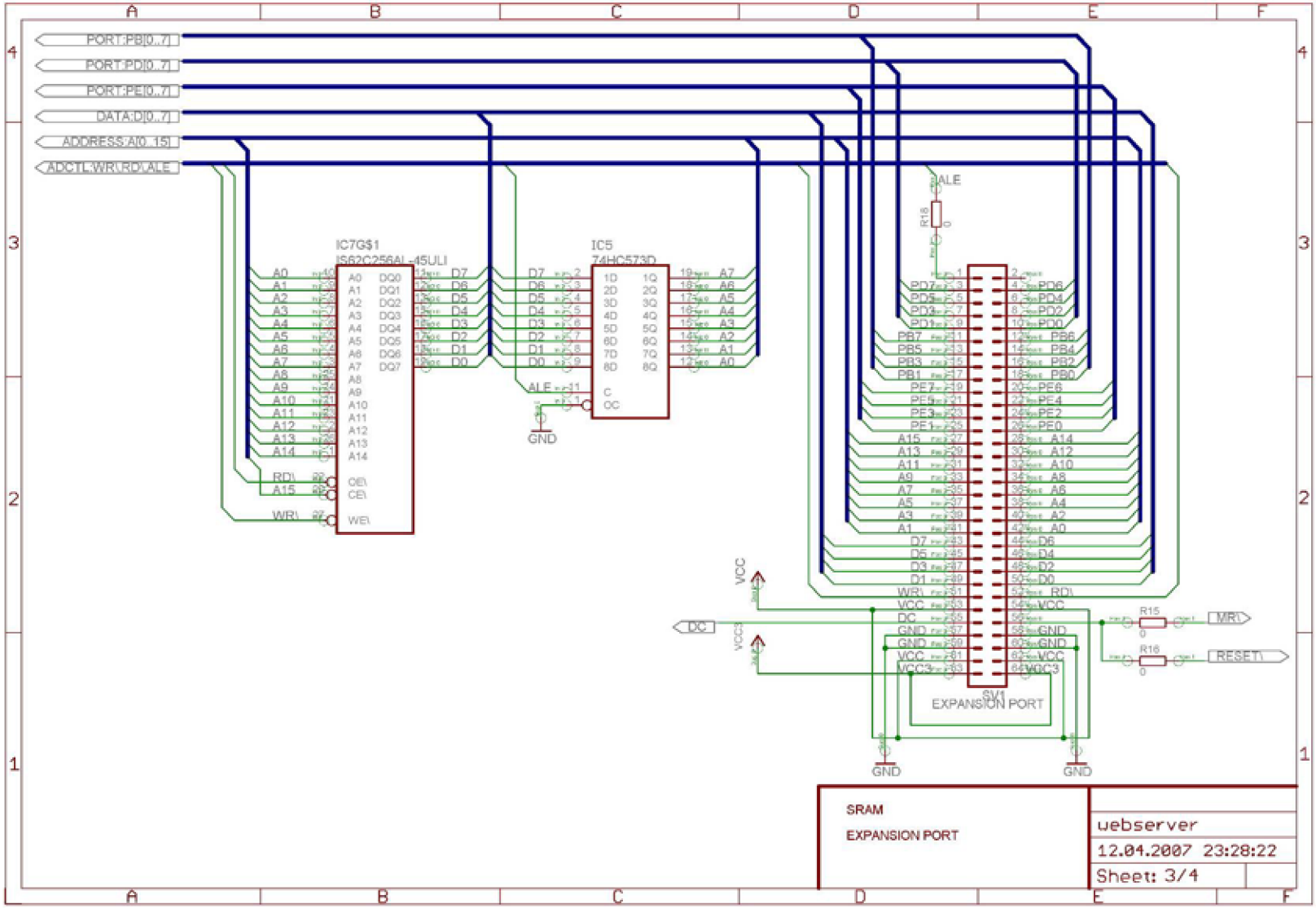
Seznam příloh



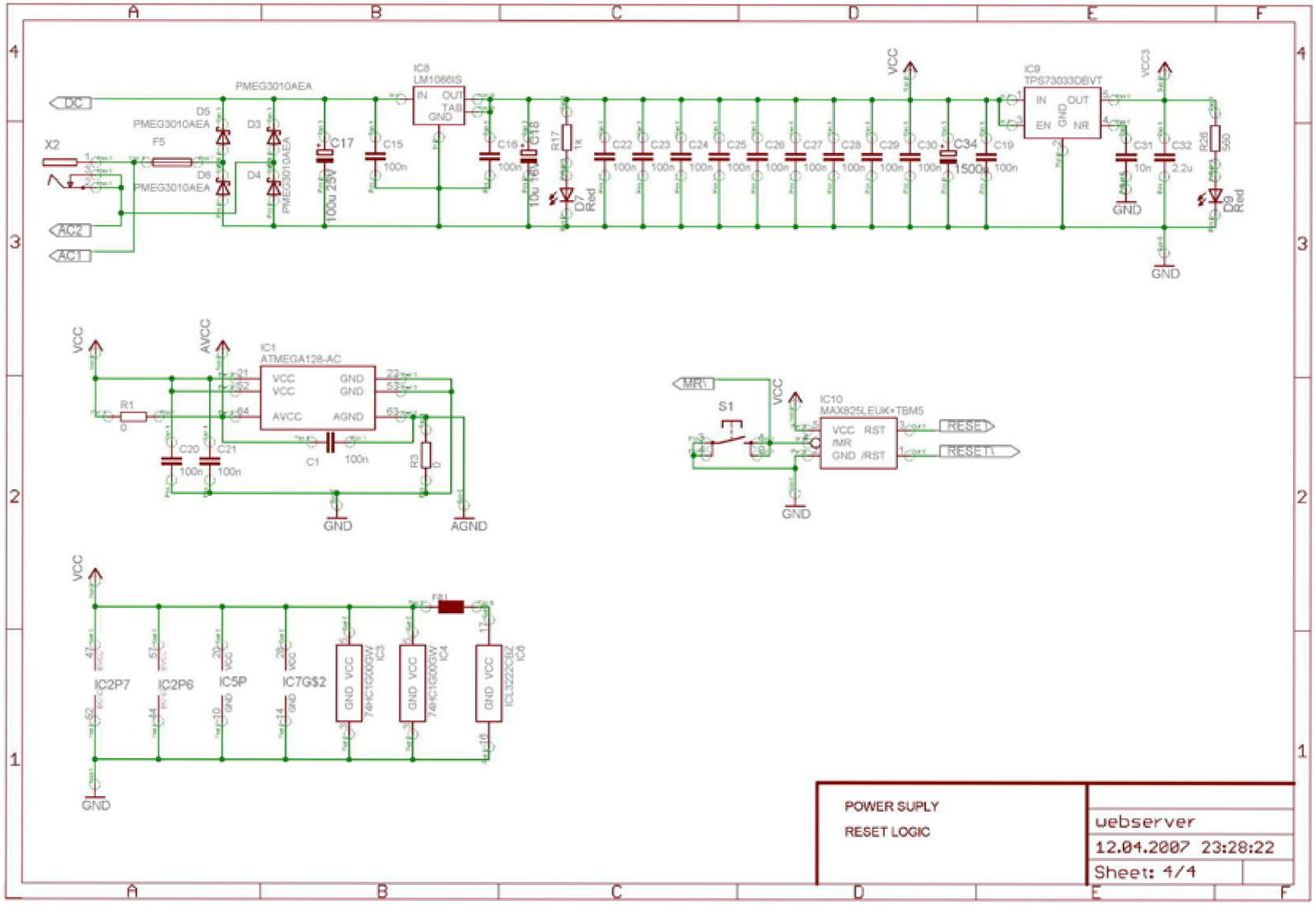
Obrázek 45: *Obvodové schéma (CPU, Analog, Seriál,JTAG, MMC interface)*



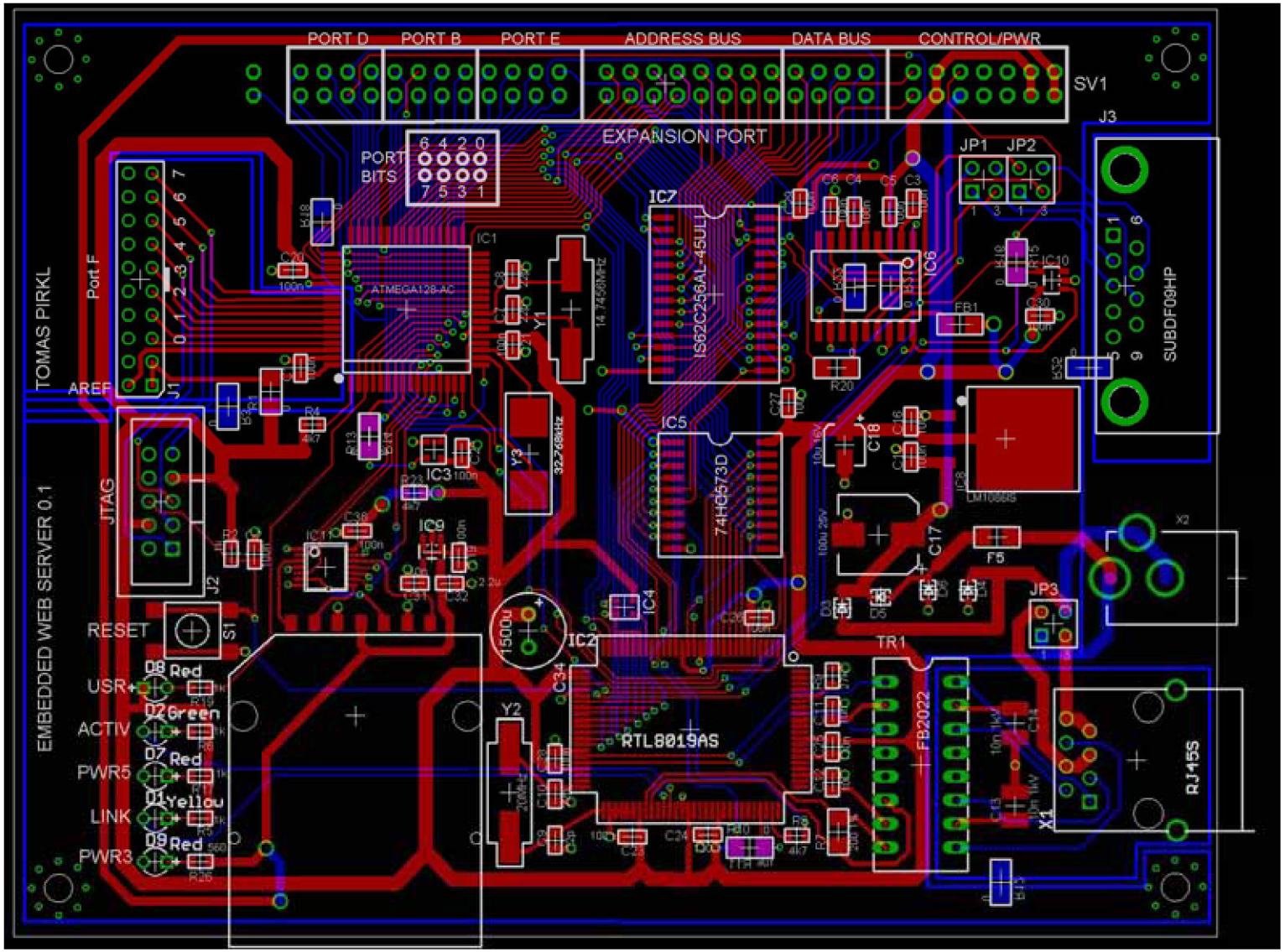
Obrázek 46: *Obvodové schéma (Ethernet interface)*



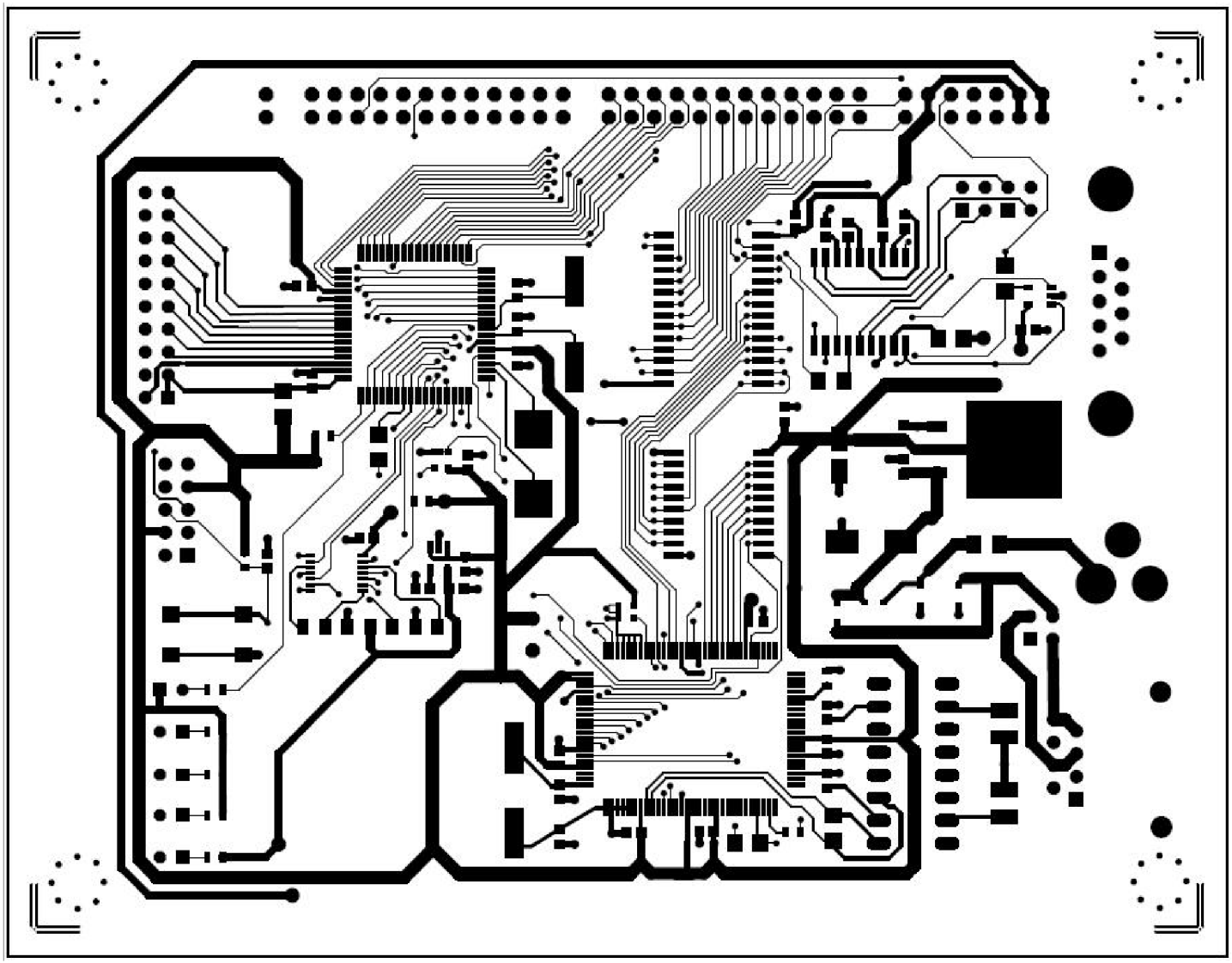
Obrázek 47: *Obvodové schéma (SRAM, Expansion port)*



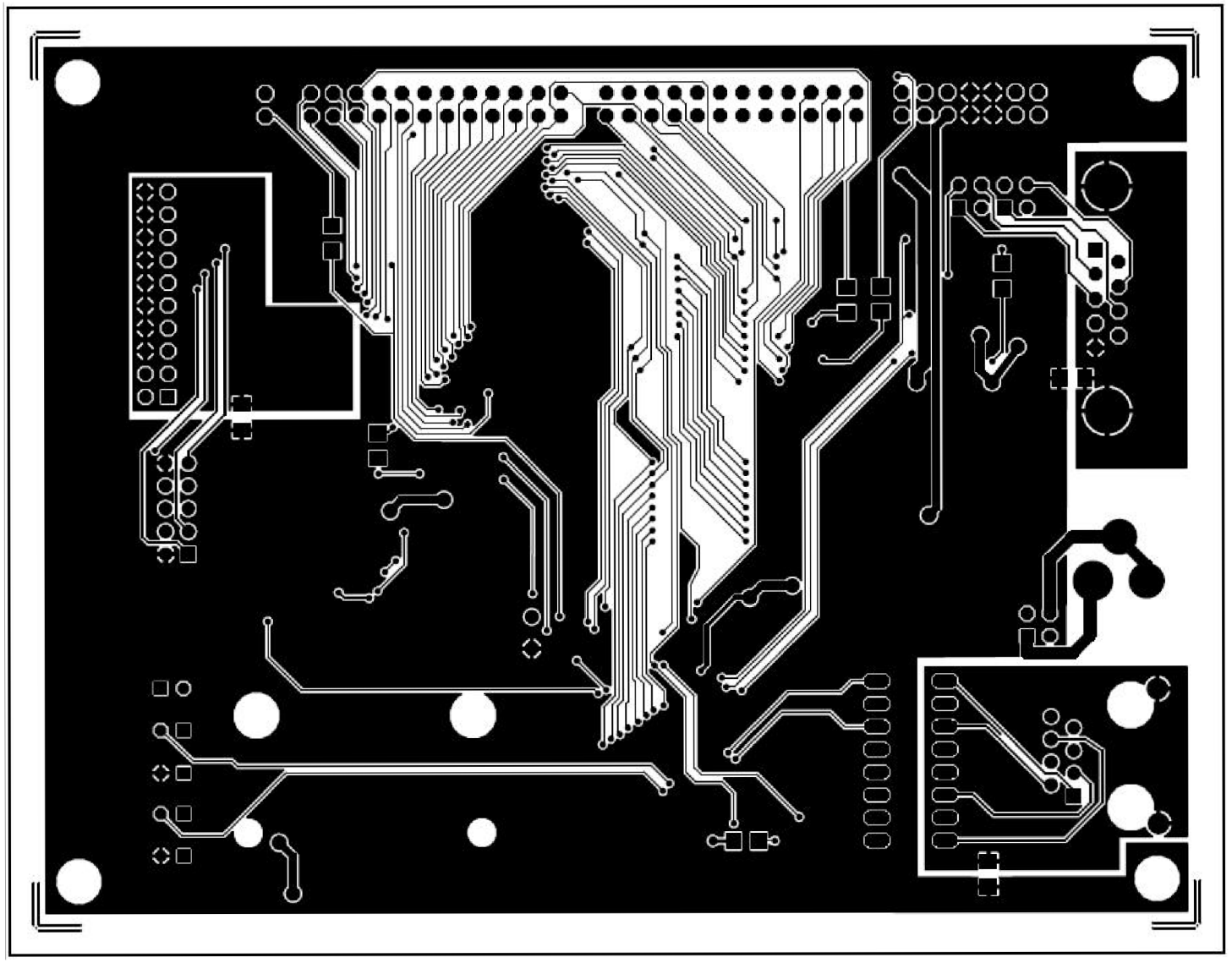
Obrázek 48: *Obvodové schéma (Power supply, Reset logic)*



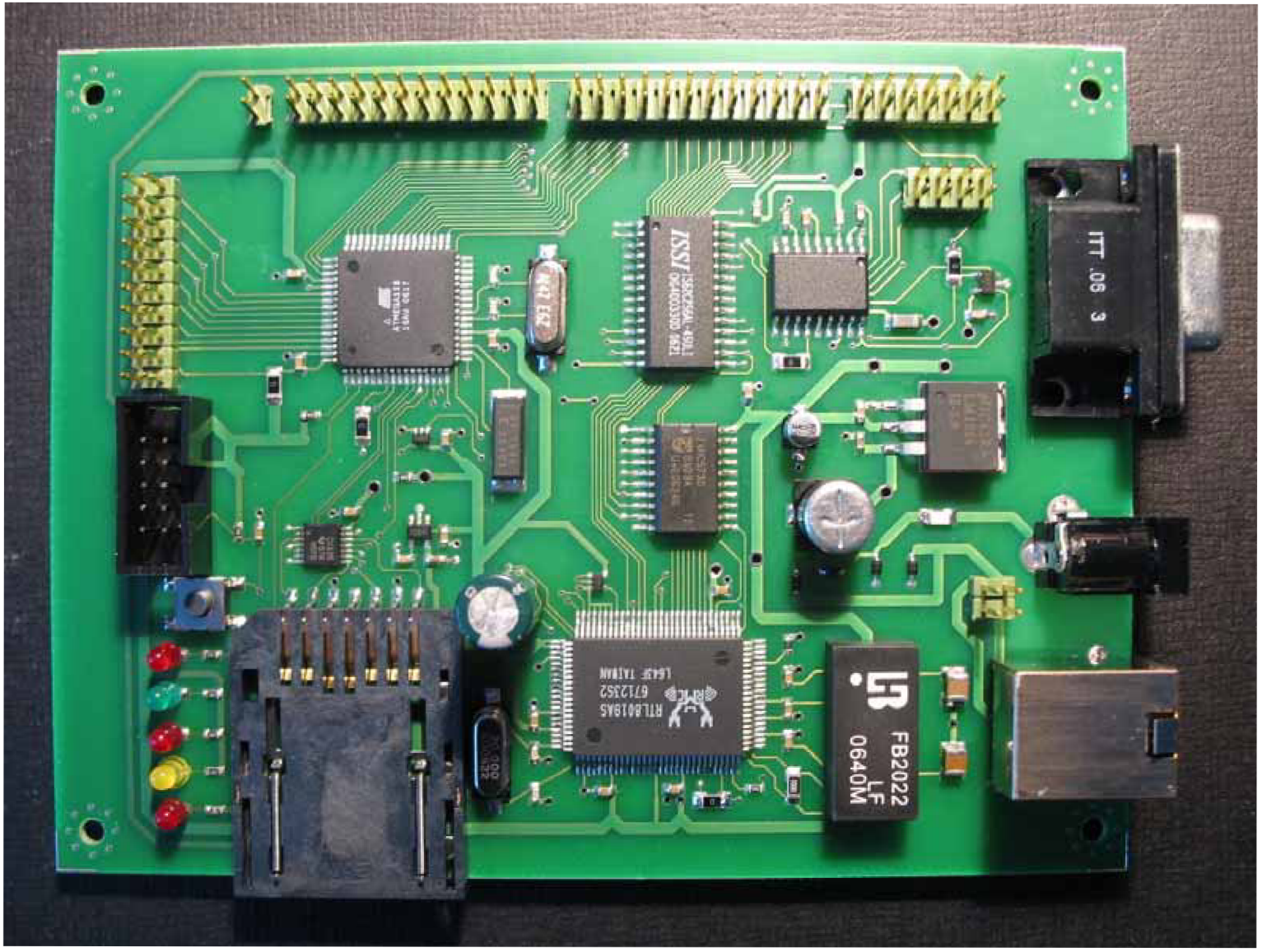
Obrázek 49: *Návrh DPS*



Obrázek 50: *Výrobní data (Vrchní část DPS)*



Obrázek 51: *Výrobní data (Spodní část DPS)*



Obrázek 52: Osazená DPS

Seznam použitých součástek

Součástka	Hodnota	Zařízení	Pouzdro	Knihovna	List
C1	100n	C0603	C0603	webserver	4
C2	100n	C0603	C0603	webserver	1
C3	100n	C0603	C0603	webserver	1
C4	100n	C0603	C0603	webserver	1
C5	100n	C0603	C0603	webserver	1
C6	100n	C0603	C0603	webserver	1
C7	22p	C0603	C0603	webserver	1
C8	22p	C0603	C0603	webserver	1
C9	22p	C0603	C0603	webserver	2
C10	22p	C0603	C0603	webserver	2
C11	10n	C0603	C0603	webserver	2
C12	10n	C0603	C0603	webserver	2
C13	10n 1kV	C	C1210K	webserver	2
C14	10n 1kV	C	C1210K	webserver	2
C15	100n	C0603	C0603	webserver	4
C16	100n	C0603	C0603	webserver	4
C17	100u 25V	ALUCAP8.0	ALUC8.0	webserver	4
C18	10u 16V	ALUCAP4.0	ALUC4.0	webserver	4
C19	100n	C0603	C0603	webserver	4
C20	100n	C0603	C0603	webserver	4
C21	100n	C0603	C0603	webserver	4
C22	100n	C0603	C0603	webserver	4
C23	100n	C0603	C0603	webserver	4
C24	100n	C0603	C0603	webserver	4
C25	100n	C0603	C0603	webserver	4
C26	100n	C0603	C0603	webserver	4
C27	100n	C0603	C0603	webserver	4
C28	100n	C0603	C0603	webserver	4
C29	100n	C0603	C0603	webserver	4
C30	100n	C0603	C0603	webserver	4
C31	10n	C0603	C0603	webserver	4
C32	2.2u	C0603	C0603	webserver	4
C34	1500u	E-KAP'3.5'	E3,5-8	webserver	4
C36	100n	C0603	C0603	webserver	1
D1	Yellow	LED3MM	LED3MM	webserver	2
D2	Green	LED3MM	LED3MM	webserver	2
D3	PMEG3010AEA	PMEG3010AEA	SOD323	webserver	4
D4	PMEG3010AEA	PMEG3010AEA	SOD323	webserver	4

D5	PMEG3010AEA	PMEG3010AEA	SOD323	webserver 4
D6	PMEG3010AEA	PMEG3010AEA	SOD323	webserver 4
D7	Red	LED3MM	LED3MM	webserver 4
D8	Red	LED3MM	LED3MM	webserver 1
D9	Red	LED3MM	LED3MM	webserver 4
F5	Fuse	OMNI-BLOK	R1206	webserver 4
FB1	FB1	FB1	R1206	webserver 4
IC1	ATMEGA128-AC	ATMEGA128-AC	TQFP64	webserver 1
IC2	RTL8019AS	RTL8019AS	QFP100	webserver 2
IC3	74HC1G00GW	74HC1G00GW	SOT353-OLD	webserver 2
IC4	74HC1G00GW	74HC1G00GW	SOT353-OLD	webserver 2
IC5	74HC573D	74HC573D	SO20W	webserver 3
IC6	ICL3222CBZ	ICL3222CBZ	SO18W300	webserver 1
IC7	IS62C256AL-45ULI	IS62C256AL-45ULI	SOP28	webserver 3
IC8	LM1086IS	LM1086IS	DDPAK-ST	webserver 4
IC9	TPS73033DBVT	TPS73033DBVT	SOT23-5	webserver 4
IC10	MAX825LEUK+TBM5	MAX825LEUK+TBM5	SOT23-5L	webserver 4
IC11	SN74CBT3125PWG4	SN74CBT3125PWG4	TSSOP14	webserver 1
J1		CON2X10PINHEAD	PINHD10X2	webserver 1
J2	JTAG	HDR2X5BOX	HDR2X5BOX	webserver 1
J3	SUBDF09HP	SUBDF09HP	F09HP	webserver 1
JP1		JP2Q	JP2Q	webserver 1
JP2		JP2Q	JP2Q	webserver 1
JP3		JP2Q	JP2Q	webserver 2
R1	0	R1206	R1206	webserver 4
R2	1k	R0603	R0603	webserver 1
R3	0	R1206	R1206	webserver 4
R4	4k7	R0603	R0603	webserver 1
R5	1k	R0603	R0603	webserver 2
R6	1k	R0603	R0603	webserver 2
R7	200 1%	R1206	R1206	webserver 2
R8	4k7	R0603	R0603	webserver 2
R9	27k	R0603	R0603	webserver 2
R10	0	R1206	R1206	webserver 2
R11	10k	R1206	R1206	webserver 2
R12	0	R1206	R1206	webserver 2
R13	0	R1206	R1206	webserver 2
R14	0	R1206	R1206	webserver 2
R15	0	R1206	R1206	webserver 3
R16	0	R1206	R1206	webserver 3
R17	1k	R0603	R0603	webserver 4

R18	0	R1206	R1206	webserver 3
R19	1k	R0603	R0603	webserver 1
R20	0	R1206	R1206	webserver 1
R21	0	R1206	R1206	webserver 1
R22	0	R1206	R1206	webserver 1
R23	4k7	R0603	R0603	webserver 1
R25	0	R1206	R1206	webserver 1
R26	560	R0603	R0603	webserver 4
S1	Button	OMRON_10XX	B3FS-1010	webserver 4
SV1	EXPANSION PORT	ML64L'2'	ML64L-DIF	webserver 3
TR1	FB2022	ISO10BT	DIL16	webserver 2
X1	RJ45S	RJ45S	RJ45	webserver 2
X2		DC'2'	DC2.1N	webserver 4
X3	SD-MMC	SD-MMC	MMC-SD	webserver 1
Y1	14.7456MHz	HC18U-VSMD	HC49SM	webserver 1
Y2	20MHz	HC18U-VSMD	HC49SM	webserver 2
Y3	32.768kHz	MC-146	SMD-B	webserver 1

Popis rozšiřujícího portu

Pin	Signál	Popis
1	VCC3	Stabilizovaných 3V3
3	VCC5	Stabilizovaných 5V
5	GND	Zem
7	GND	Zem
9	MR\	Vstup pro reset
11	VCC5	Stabilizovaných 5V
13	RD\	Signál pro čtení z paměti
15	D0	Bit 0 datové sběrnice
17	D2	Bit 2 datové sběrnice
19	D4	Bit 4 datové sběrnice
21	D6	Bit 6 datové sběrnice
23	A0	Bit 0 adresové sběrnice
25	A2	Bit 2 adresové sběrnice
27	A4	Bit 4 adresové sběrnice
29	A6	Bit 6 adresové sběrnice
31	A8	Bit 8 adresové sběrnice
33	A10	Bit 10 adresové sběrnice
35	A12	Bit 12 adresové sběrnice
37	A14	Bit 14 adresové sběrnice
39	PE0	Bit 0 portu E
41	PE2	Bit 2 portu E
43	PE4	Bit 4 portu E
45	PE6	Bit 6 portu E
47	PB0	Bit 1 portu B
49	PB2	Bit 1 portu B
51	PB4	Bit 1 portu B
53	PB6	Bit 1 portu B
55	PD0	Bit 1 portu D
57	PD2	Bit 1 portu D
59	PD4	Bit 1 portu D
61	PD6	Bit 1 portu D
63	NC	Neppřipojeno

Pin	Signál	Popis
2	VCC3	Stabilizovaných 3V3
4	VCC5	Stabilizovaných 5V
6	GND	Zem
8	GND	Zem
10	DC	Nestabilizovaný vstup
12	VCC5	Stabilizovaných 5V
14	WR\	Signál pro zápis do paměti
16	D1	Bit 1 datové sběrnice
18	D3	Bit 3 datové sběrnice
20	D5	Bit 5 datové sběrnice
22	D7	Bit 7 datové sběrnice
24	A1	Bit 1 adresové sběrnice
26	A3	Bit 3 adresové sběrnice
28	A5	Bit 5 adresové sběrnice
30	A7	Bit 7 adresové sběrnice
32	A9	Bit 9 adresové sběrnice
34	A11	Bit 11 adresové sběrnice
36	A13	Bit 13 adresové sběrnice
38	A15	Bit 15 adresové sběrnice
40	PE1	Bit 1 portu E
42	PE3	Bit 3 portu E
44	PE5	Bit 5 portu E
46	PE7	Bit 7 portu E
48	PB1	Bit 1 portu B
50	PB3	Bit 1 portu B
52	PB5	Bit 1 portu B
54	PB7	Bit 1 portu B
56	PD1	Bit 1 portu D
58	PD3	Bit 1 portu D
60	PD5	Bit 1 portu D
62	PD7	Bit 1 portu D
64	ALE	Platnost adresy A0-7

Popis analogového portu

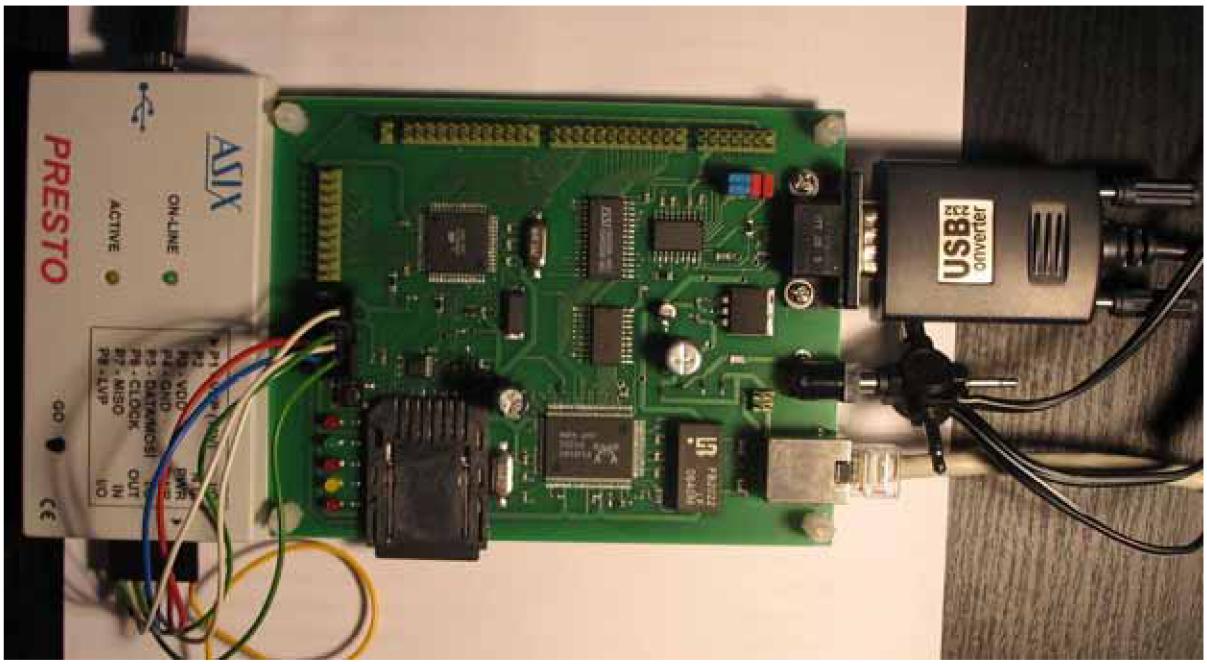
Pin	Signál	Popis
1	AVCC5	Napájení 5V
3	AVCC5	Napájení 5V
5	PF0	Analogový vstup 1
7	PF1	Analogový vstup
9	PF2	Analogový vstup
11	PF3	Analogový vstup
13	PF4	Analogový vstup
15	PF5	Analogový vstup
17	PF6	Analogový vstup
19	PF7	Analogový vstup

Pin	Signál	Popis
2	AREF	Kontrola analogu
4	AGND	Zem
6	AGND	Zem
8	AGND	Zem
10	AGND	Zem
12	AGND	Zem
14	AGND	Zem
16	AGND	Zem
18	AGND	Zem
20	AGND	Zem

Manuál

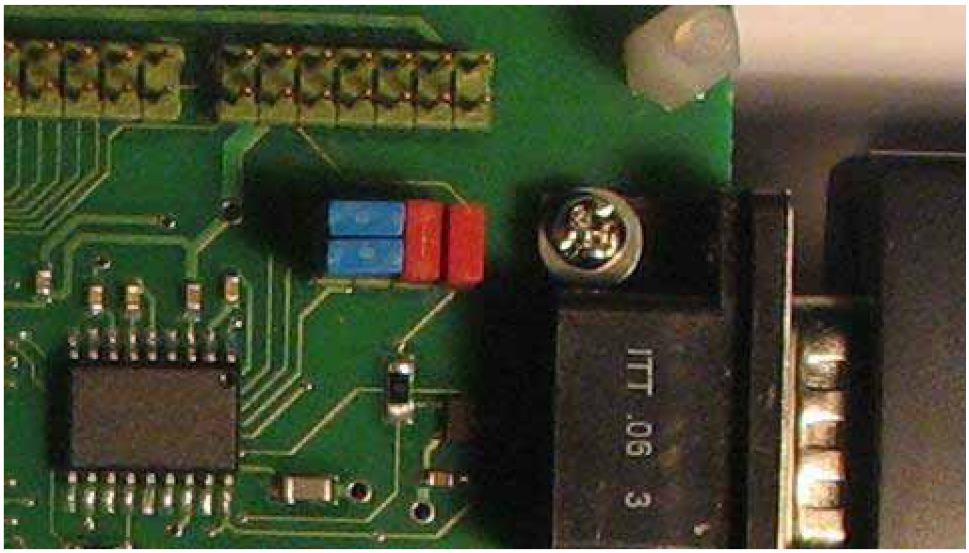
Připojení desky

Desku položte tak, aby se svými kontakty nijak nedotýkala žádných vodivých nebo jiných materiálů, které by mohly desku zničit. Nepoužívejte jako podložku ani antistatický pytlík, protože je eletroinduktivní, ani plastovou podložku, protože by mohlo dojít k elektrostatickému výboji. Desku tedy položte nejlépe na dřevěný povrch nebo na papír. Nebo ji vybavte distančními šrouby které ji budou udržovat nad povrchem podkladu.



Připojte port DB-9 RS232 na volný COM port použitím sériového kabelu nebo pomocí převodníku RS232/USB k volnému USB portu na vašem počítači.

Přednastavená konfigurace přepínačů:



Dále použijte ethernetový kabel pro připojení desky do hubu nebo přepínače. Nebo křížený ethernetový kabel pro připojení do vašeho PC.

Na kartu MMC nakopírujte jakékoli html stránky s libovolným html kódem s úvodní stránkou `index.html`. Dále do šachty pro MMC kartu tuto kartu vsuňte.

Připojte napájecí zdroj (DC 9-12V, 300 mA minimum, konektor 2.1 mm) do do konektoru pro napájení. Zdroj by neměl být připojen k elektrické síti dříve, než je připojen do konektoru pro napájení desky.

Připojte napájecí zdroj do elektrické sítě. Následně by se měly rozsvítit dvě červené diody signalizující funkční napájení 5V a 3V3.

Signalizace led diodami umožňuje sledovat provoz na Ethernetu, RS232 a funkčnost napájení 5V a 3V3.

Pomocí terminálu se připojte k příslušnému COM portu s těmito parametry (115200 baud, no parity, 8 data bits, 1 stop bit, disable hardware RTS/CTS, disable software XON/XOFF).

Přednastavená konfigurace Web serveru v souboru `httpserv.c`:

- IP adresa: `10.0.0.99`
- MAC adresa: `\x00\x0A\x59\x00\xAB\x58`
- Maska sítě: `255.255.255.0`
- Brána: `10.0.0.1`

Následně by se po restartu měl v terminálu objevit tento výpis:

```
Nut/OS 4.3.2.1 beta HTTP Daemon...Configure eth0...OK
10.0.0.99 ready

Init MMC interface...ready
Mount MMC...ready
```

Pak už po zadání adresy http://10.0.0.99/ by měl váš prohlížeč zobrazit stránky umístěné na MMC kartě.

Instalace NUT/OS

Nejdříve je nutné nainstalovat překladač WinAVR (já použil verzi 20060421). Pak už můžeme spustit instalaci NUT/OS (já použil verzi 4.3.2). Po instalaci musí proběhnout konfigurace NUT/0S (spouští se pomocí `nutconf.exe`). Při konfiguraci použijte přednastavené hodnoty pro Ethernut 1.3 rev G tedy soubor ethernut13g.conf. Posléze nastavte konfiguračním rozhraní správně cesty pro aplikace a pro překladač (popsáno v [23]). Pak už proveďte kompilaci NUT/OS jejímž výsledkem jsou následující knihovny:

- `libnutcrt.a`
- `libnutcrtf.a`
- `libnutdev.a`
- `libnutfs.a`
- `libnutnet.a`
- `libnutos.a`
- `libnutpro.a`
- `nutinit.o`

Pokud vám překlad selže, tak ještě zkuste v příkazovém řádku nastavit tyto cesty (podle umístění jednotlivých aplikací ve vašem PC):

- `set PATH=c:\WinAVR\bin;c:\WinAVR\utils\bin;%PATH%`
- `SET PATH=c:\ethernut-4.3.2\nut\tools\win32;%PATH%`

Následně můžete zkusit sestavit knihovny pomocí:

- `make clean`
- `make install`

Instalace NUT/OS je detailně popsána v [23].

Instalace SW

Pokud již máme nainstalován, zkonfigurován NUT/OS a sestaveny všechny knihovny pro jeho funkci je potřeba ještě přeložit samotnou aplikaci http serveru s ovladači pro MMC kartu a ovladači pro FAT. Nakopírujte adresář `MMCWS` (je na CD) do adresáře `\ethernut-4.3.2\nutapp\`. Poté opět na příkazové řádce nastavte cesty:

- `set PATH=c:\WinAVR\bin;c:\WinAVR\utils\bin;%PATH%`
- `SET PATH=c:\ethernut-4.3.2\nut\tools\win32;%PATH%`

Následně přeložte obsah adresáře pomocí příkazů:

- `make clean`
- `make`

Tak získáte soubor `httpserv.hex`. Tento soubor ještě musíme pomocí `avrsvf.exe` přeložit do formátu SVF (Seriál Vector Format), se kterým už umí pracovat programátor PRESTO. Překlad se provede příkazem:

- `avrsvf -datmega128 -s -e -ifhttpserv.hex -pf -vf -f0xFF833F -F -ovhttpserv.svf -mp`

Parametry avrsvf jsou k nahlédnutí po zadání příkazu:

- `avrsvf -h`

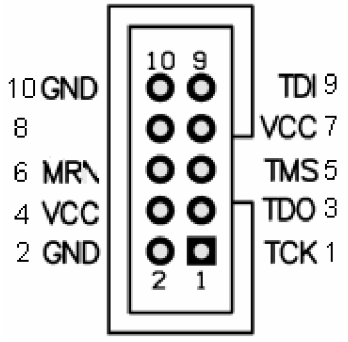
Velký pozor dávejte na nastavení Fuse bits [12] parametrem `f0xFF833F`. Při nesprávném nastavení může dojít k nevratnému zničení mikrokontroléru ATmega128. Výsledkem překladu do formátu SVF je soubor `httpserv.svf`, se kterým už umí pracovat SW programátoru PRESTO.

Následuje připojení karty k RS232, Ethernetu, k Prestu a k napájení. Potom spustíme JTAG Player for Presto předložíme mu soubor `httpserv.svf` a spustíme programování paměti mikrokontroléru ATmega128. Pak už můžeme funkčnost ověřit pomocí nějakého internetového prohlížeče.

Programátor PRESTO

Připojení programátoru PRESTO k portu JTAG na desce je popsáno následující tabulkou a obrázkem.

PRESTO	JTAG		
Pin	Název	Typ[3]	Pin (port na desce)
1	USR[4]	O	NC
2	NC	NC	NC
3	VCC	PWR	4
4	GND	PWR	2
5	TDI	O	9
6	TCK	O	1
7	TDO	I	3
8	TMS	O	5



[3] PWR: napájení; I: vstup; O: výstup; NC: nepřipojeno

[4] V programátoru volitelná funkce TRST, SCK nebo vlastní (NENÍ NUTNÉ PŘIPOJIT)