VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

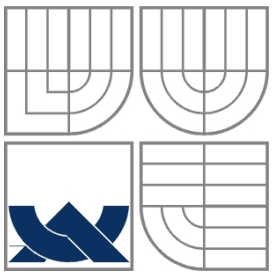
VESTAVĚNÝ WEBOVÝ SERVER ZALOŽENÝ NA PLATFORMĚ FREESCALE

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

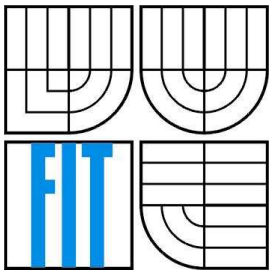
AUTOR PRÁCE
AUTHOR

Bc. PETER ŠILON

BRNO 2010



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

VESTAVĚNÝ WEBOVÝ SERVER ZALOŽENÝ NA PLATFORMĚ FREESCALE

EMBEDDED WEB SERVER BASED ON FREESCALE PLATFORM

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. PETER ŠILON

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. JOSEF STRNADEL, Ph.D.

BRNO 2010

Abstrakt

Vstavaný webový server (ďalej len VWS) je webový server bežiaci na systéme s obmedzenou výpočetnou kapacitou a pamäťou. Zabudovaním webového servera do vstavaného zariadenia so sieťovým rozhraním, získame nové možnosti riadenia a správy na diaľku. Webové technológie ponúkajú užívateľsky príjemné GUI, jednoduchosť a nízku cenu. Táto práca sa zaoberá návrhom VWS s využitím prostriedkov od firmy Freescale. VWS musí obsahovať slot pre pamäťovú kartu, periférie na doske alebo signály rozhraní vyvedené do konektoru pre pripojenie externých zariadení. Obslužný software bude postavený na niektorej z bezplatných TCP/IP implementácií pre vstavané systémy.

Abstract

Embedded web server (EWS) is a web server that runs on a device with limited computing power and memory. By implementing a web server into embedded device with network connectivity we achieve new way of long distance management and control. Web technologies offer user-friendly GUI, simplicity and low prices. The aim of this project is a design of the EWS using Freescale resources. EWS have to have a slot for memory card, on–board peripheries or interfaces signals attached to connecter to allow connection of external devices. Firmware will be built on one of the freeware TCP/IP implementation for embedded systems.

Kľúčové slová

Vstavaný webový server, vstavaný ethernet, ColdFire, MCF52233, InterNiche TCP/IP stack, RTOS

Keywords

Embedded web server, embedded ethernet, ColdFire, MCF52233, InterNiche TCP/IP stack, RTOS

Citácie

Šilon Peter: Vestavěný webový server založený na platformě Freescale, diplomová práce, Brno, FIT VUT v Brně, 2010

Vestavěný webový server založený na platformě Freescale

Prehlásenie

Prehlasujem, že som túto diplomovú pracú vypracoval samostatne pod vedením Ing. Josefa Strnadela, Ph.D.
Uviedol som všetky literárne pramene a publikácie, z ktorých som čerpal.

……
Peter Šilon
01.05.2010

Poďakovanie

Na tomto mieste by som chcel poďakovať *Ing. Josefovi Strnadelovi, Ph.D* za pomoc pri riešení diplomovej práce a svojim rodičom, ktorí ma nielen finančne podporovali počas celého štúdia na vysokej škole.

Obsah

Úvod

Protokoly ako Fieldbus, ControlNet, Interbus alebo CAN boli vyvinuté za účelom sieťovej komunikácie medzi real–time vstavanými systémami. Nevýhodou týchto protokolov je rýchlosť (1 Mbps) a prenos väčšieho objemu dát. Naopak výhodou je determinizmus pri viacnásobnom prístupe na zdieľané médium. Pre hard real–time systémy je determinizmus kľúčový, pretože oneskorenie dát môže mať katastrofické následky. Soft real–time systémy, pre ktoré nie sú oneskorené dáta kritické, môžu používať známe vysokorýchlostné protokoly počítačových sietí LAN.

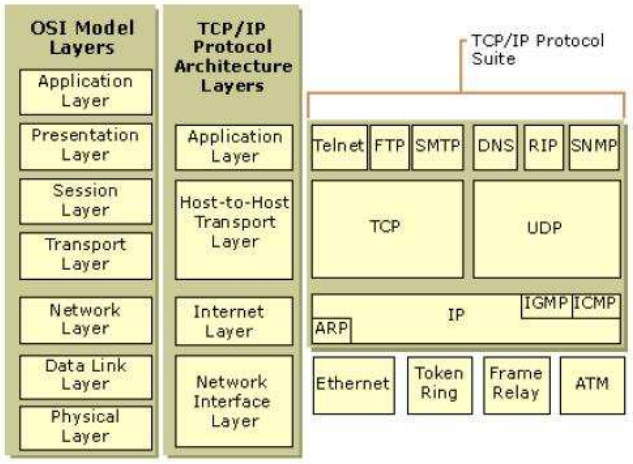
Ethernet má niekoľko vlastností, ktoré ho robia ideálnou technológiou pre distribuované vstavané systémy. Je decentralizovaný, neexistuje centrálny bod siete ktorý môže zlyhať a žiaden paket sa nestratí. Ethernet má vo svojej základnej verzii rýchlosť 10 Mbps, čo je viac ako ostatné protokoly pre vstavané systémy. Firmware môže byť vyvíjaný a testovaný na klasickom PC a mal by byť prenositeľný na cieľové zariadenie pretože vrstva MAC (Medium Access Control) musí byť identická. Použitím protokolu TCP/IP môžu byť vstavané systémy globálne dostupné.

Práca je logicky rozčlenená do štyroch častí. Na začiatku je teoretický úvod do problematiky sieťovej komunikácie a pripojenia vstavaných systémov k internetu. Postupne sú diskutované sieťové protokoly v poradí v akom sú definované vo vrstvovom modely TCP/IP. Pripojenie vstavaných systémov k internetu je ukázané na troch príkladoch. Do teoretického úvodu patrí aj porovnanie dostupných TCP/IP implementácií pre vstavané systémy. Druhá časť práce je návrh samotného VWS. Od vymedzenia vlastností a funkcií cez výber vhodných súčiastok po návrh blokovej schémy webového servera. Realizácia je rozdelená do dvoch častí, hardwarovú a softwarovú časť. V hardwarovej časti nájdeme návrh schémy zapojenia podľa požiadaviek návrhu, konštrukciu DPS, osadenie a oživenie dosky. Softwarové vybavenie je popisované v poslednej časti práce.

1 Architektúra TCP/IP

TCP/IP je označenie skupiny komunikačných protokolov. Názov je zavádzajúci, pretože TCP a IP sú iba dva z tuctu obsiahnutých protokolov. Vývoj TCP/IP odštartovalo v roku 1969 ministerstvo obrany USA za účelom umožniť komunikáciu medzi jednotlivými strediskami. Od toho roku sa sieť rozrastala do dnešnej podoby internetu.

TCP/IP protokoly sa mapujú do štvor–vrstvového konceptuálneho modelu niekedy označovaného ako DARPA model. Vrstvy modelu sú: aplikačná, transportná, internetová a vrstva sieťového rozhrania. Každá z vrstiev DARPA modelu korešponduje s jednou alebo viac vrstvami referenčného OSI modelu.



Obr. 1.1: Porovnanie OSI a TCP/IP modelu. Prevzaté z [7]

1.1 Vrstva sieťového rozhrania

Vrstva sieťového rozhrania je zodpovedná za umiestnenie paketov na médium a ich príjem. Protokoly TCP/IP sú navrhnuté tak, aby boli nezávislé na metóde prístupu na sieť, formáte rámca a druhu média. Preto môže byť TCP/IP použité na spojenie sietí rôzneho typu. To zahŕňa LAN technológie ako Ethernet alebo Token Ring a WAN technológie ako X.25 alebo Frame Relay. Na tejto vrstve je IP paket zapuzdrený do rámcu a vyslaný na sieť.

1.1.1 Ethernet

Ethernet je dnes najpoužívanejšou sieťovou technológiou pre svoju rýchlosť, nízku cenu a relatívne jednoduchú inštaláciu. Počiatky ethernetu siahajú do roku 1973, kedy bol po prvý krát definovaný Robertom Metcalfom, inžinierom spoločnosti Xerox. Prenosová rýchlosť prvej verzie bola 2.94

Mbit/s. Na ďalšom vývoji sa podieľala výskumná skupina *DIX*, (firmy Digital, Intel a Xerox), ktorá v roku 1980 vypracovala štandard 10 Mbit ethernetu. Treba podotknúť, že nešlo o otvorený štandard. Až o päť rokov neskôr, vydala spoločnosť *IEEE* otvorený štandard 802.3 postavený na myšlienke ethernetu.

Ethernet je striktne postavený na vrstvovom modely OSI. To umožňuje kombinovať metódu prístupu na médium – MAC s rôznymi fyzickými vrstvami. Ako MAC používa ethernet technológiu CSMA/CD. Stanica, ktorá chce vysielať sa pred tým musí uistiť, že nikto iný momentálne na zdieľané médium nevysiela. Pokiaľ je médium voľné určitý časový interval (9.6 µs), stanica môže zasielať dáta. Vysielanie nemôže začať skôr. Pretože ethernet používa zdieľane médium očakávajú sa kolízie. Kolízia vzniká ak dve alebo viac staníc čakajú na uvoľnenie média, súčasne detekujú, že kanál je voľný a začnú vysielať. Takáto udalosť znehodnotí dáta. Stanica aj počas vysielania neustále monitoruje kanál na prítomnosť kolízie. Ak stanica detekuje kolíziu počas vysielania, vysielanie sa okamžite zastaví a vyšle tzv. *jam* signál aby sa zaručilo, že všetky ostatné stanice kolíziu detekujú a odmietnu prijať znehodnotené dáta. Po určitej čakacej dobe si stanica, ktorá si priala dáta vysielať pokus zopakuje. Dobu, po ktorú bude stanica čakať určí náhodne algoritmus BEB. Obe stanice tak budú mať rozdielne štartovacie podmienky a vznik kolízia sa obmedzí.

Ethernet používa rámce dát na zaslanie aktuálnej informácie známej ako *payload*. Rámec môže mať rôznu veľkosť v závislosti na veľkosti prenášaných dát. Štruktúra ethernetového rámce je na obrázku 1.2.

Preamble	Start Frame Delimiter	Destination Address	Source Address	Length	LLC	Data	Pad	Frame Check Sequence
7 bytes	1 byte	6 bytes	6 bytes	2 bytes	Up to 1500 bytes			4 bytes

Obr. 1.2: IEEE ethernetový rámec. Prevzaté z [7]

Na začiatku rámca je preambula, ktorá synchronizuje prijímač s vysielaným rámcom. Je to sekvencia 01010101..., 7 bajtov dlhá. Nasleduje oddeľovacia značka (SFD) dlhá jeden bajt. *Destination address* označuje fyzickú adresu na akú sa má rámec doručiť. *Source address* je adresa označujúca odosielateľa. Dvoj–bajtové pole *length* špecifikuje počet bajtov v *LLC* a *Data*. *LLC* definuje štruktúru dát na linkovej vrstve. *Data* môžu byť dlhé do 1500 B. Minimálna veľkosť dát nie je definovaná, ale rámec musí mať minimálne 64 B. Linková vrstva preto môže pridať nejaké dáta navyše na udržanie minimálnej požadovanej veľkosti. Tieto dáta navyše predstavuje pole *Pad*. Na konci rámca je pole *FCS*, ktoré obsahuje kontrolný súčet (CRC) a zaručuje správnosť prenosu.

Spolu existujú štyri rôzne typy ethernetových rámcov. Rámec IEEE 802.3 popisovaný vyššie. Rámec IEEE 802.2 ktorý obsahuje naviac informácie o použitom protokole, polia *DSAP*, *SSPA* a *Control Field*. Rámec IEEE 802.2 SNAP, ktorý je totožný s 802.2, ale podporuje viac ako 256

typov protokolov zväčšením polí *DSAP* a *SSPA*. Posledným rámcom je rámec DIX Ethernet II. Neobsahuje žiadne *LLC* informácie pre vyššie vrstvy a minimálna veľkosť dát je 46 B.

S rozvojom ethernetu sa vyvíjali aj prenosové média. Od koaxiálnych káblov 10BASE5, cez krútené dvojlinky 10BASE–T až po optické vlákna 100BASE–T a 10BASE–FX. Práve krútená dvojlinka umožnila vznik plne duplexného ethernetu. Štyri dvojice káblov umožnili oddelenie vysielania a prijímania. Plne duplexný ethernet nepotrebuje algoritmus CSMA/CD pretože dáta sú prijímané a zasielané v rovnakom čase a nevznikajú kolízie. Plne duplexný ethernet môže byť použitý v troch situáciách:

- spojenie switch – host
- spojenie switch – switch
- spojenie host – host kríženým káblom

1.2 Internetová vrstva

Najznámejším protokolom na internetovej vrstve je Internet Protokol – IP, ktorý zabezpečuje základné paketové služby pre všetky TCP/IP siete. Okrem fyzických adries používaných na vrstve sieťového rozhrania, IP protokol implementuje systém logických IP adries. IP adresa je používaná internetovou vrstvou a všetkými vrstvami nad ňou na jednoznačnú identifikáciu zariadenia a smerovanie. IP protokol je využívaný každým protokolom na všetkých vrstvách.

1.2.1 IP Protokol

Je nespojovo orientovaný a nespoľahlivý protokol. Nespoľahlivý vo význame, že nie je zaručené úspešné doručenie dát. IP protokol definuje základnú jednotku dát, ktorá môže byť TCP/IP sieťou prenášaná na úrovni sieťovej vrstvy, tzv. IP datagram a jeho presný formát. Samotný IP protokol neposkytuje žiadne mechanizmy na detekciu alebo korekciu chýb, ale spolieha na služby vyšších vrstiev. Ako je datagram smerovaný cez rôzne siete, je niekedy nevyhnutné, aby bol rozdelený do menších častí podľa potrieb aktuálnej siete. Každý typ siete ma definovanú *Maximum Transmission Unit – MTU,* čo je najväčší paket aký môže sieť prenášať. Takéto delenie paketu sa volá *fragmentácia*. Každý fragment paketu má vlastnú IP hlavičku, ktorá obsahuje informácie:

- zdrojová IP adresa
- cieľová IP adresa
- identifikácia – jednoznačná identifikácia IP datagramu a v prípade fragmentácie identifikácia fragmentov patriacich do jedného datagramu
- protokol – aký protokol bol použitý vyššou vrstvou
- checksum – 32 bitové CRC

- TTL – životnosť paketu. Každý router, ktorý paket spracuje, dekrementuje TTL. Pri TTL = 0 je paket zahodený. Predchádza sa tak nekonečnému obiehaniu paketu po sieti.

1.2.2 ICMP

Internet Control Message Protocol, skrátene ICMP je určený k signalizácii problémov v TCP/IP sieťach. ICMP zaisťuje prenos cele rady dôležitých informácií medzi komponentmi IP siete ako koncová stanica alebo router. ICMP protokol používa na prenos IP datagram s číslom protokolu 1. Sám je teda prenášaný nespojovo a bez garancie úspešného prenosu.

Základnými položkami ICMP paketu sú polia *type* a *code*. Type udáva typ paketu ICMP a code rozširuje informáciu o type. Tab. 1.3 zobrazuje najčastejšie typy ICMP paketov.

Typ	Názov	Popis
0	Echo Reply	Odpoveď na paket Echo Request
3	Destination Unrechable	Používaný routrami na upozornenie, že cieľová IP adresa nie je dostupná
4	Source Quench	Používaný routrami na upozornenie, že pakety sú zasielané príliž rýchlo a nestíhaju byť spracované
8	Echo Request	Vytvorenie Echo paketu
11	Time Exceeded	Používaný routrami na upozornenie že vypršalo TTL

Tab. 1.3: Najčastejšie typy ICMP paketov

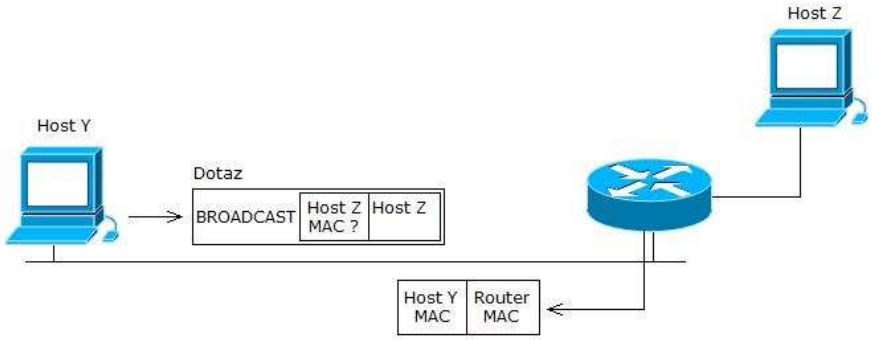
1.2.3 ARP

Stanice aj smerovače potrebujú vedieť fyzickú adresu svojich susedov pre prenos datagramu cez fyzický segment siete (ethernetové rámce nerozumejú IP adrese). Fyzická adresa je závislá na hardwari a nemá žiadny súvis s IP adresou. V rámci architektúry TCP/IP preto musí existovať mechanizmus prekladu fyzických a sieťový adries. Pre IPv4 sa mapovanie adries na LAN rieši dynamicky pomocou protokolu ARP (Address Resolution Protocol). Existujú architektúry ktoré riešia mapovanie staticky, začlenením fyzickej adresy priamo do sieťovej adresy. Statické mapovanie je pohodlnejšie a efektívnejšie. O to sa práve snaží IPv6.

ARP sa používa pri znalosti cieľovej IP adresy pre nájdenie príslušnej fyzickej adresy rozhrania. Ak chce stanica komunikovať so vzdialenou stanicou a pozná jej cieľovú IP adresu potrebuje zistiť jej fyzickú adresu na to, aby mohol byť datagram zapuzdrený do platného rámca. Ak stanica nenájde príslušnú adresu vo svojej ARP cache pamäti, musí vygenerovať ARP dotaz. Vyšle žiadosť protokolu ARP s informáciou o zdrojovej dvojici adries (fyzická, sieťová) a s cieľovou IP adresou, ktorú adresuje všetkým staniciam na sieti. Ak je cieľová adresa pripojená na rovnaký segment siete, reaguje odpoveďou zaslanou na adresu v ARP dotaze. Súčasne skontroluje obsah svojej ARP tabuľky, či ju nemôže doplniť o dvojicu prijatých adries z ARP dotazu. Ak cieľová

stanica nie je lokálna, potom na dotaz odpovie smerovač fyzickou adresou rozhrania ku ktorému je pripojená zdrojová stanica (Proxy ARP). ARP teoreticky nepatrí medzi sieťové protokoly, pretože sa jedná len o podporný protokol pre niektoré typy fyzických sieti. Nepoužíva IP pakety ale rámce spojovej vrstvy.

Aby sa minimalizovalo vysielanie ARP dotazov, má každá stanica nakonfigurovanú sieťovú masku podľa ktorej zistí, či cieľová stanica leží v rovnakej podsieti. Súčasne má nakonfigurovanú implicitnú adresu smerovača ktorému posiela všetky pakety určené mimo lokálnu sieť.



Obr. 1.4: Proxy ARP

1.3 Transportná vrstva

Tretia vrstva je niekedy označovaná aj ako TCP vrstva, lebo je najčastejšie realizovaná práve protokolom TCP. Hlavnou úlohou tejto vrstvy je zaistiť prenos medzi dvomi koncovými účastníkmi, ktorí sú v prípade TCP/IP priamo aplikačné programy. Podľa ich nárokov a požiadaviek môže transportná vrstva regulovať tok dát oboma smermi, zaisťovať spoľahlivosť prenosu prípadne meniť spojovaný charakter prenosu na nespojovaný.

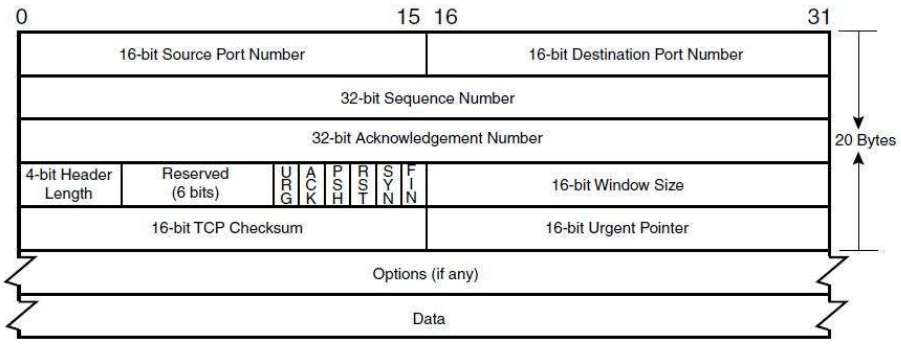
1.3.1 TCP

TCP je spojovo orientovaný transportný stavový protokol, ktorý garantuje integritu a sekvenčné doručovanie dát po tom, čo bolo naviazané spojenie medzi dvomi koncovými účastníkmi. TCP spojenie rozdeľujeme do troch fázy – ustanovenie spojenia, prenos dát a uzavretie.

Pred tým, ako začne výmena dát, je potrebné synchronizovať stavové automaty na oboch stranách spojenia. Úvodná fáza slúži okrem synchronizácie aj k vzájomnej výmene počiatočných sekvenčných čísel a počiatočnej veľkosti okien TCP spojenia. Toto zaisťuje tzv. *3–way handshaking*. Pri *3–way handshaking* si klient zvolí počiatočné sekvenčné číslo a pošle serveru TCP paket s nastavením príznakom *SYN*. Server po prijatí paketu zašle potvrdenie s príznakom *SYN/ACK*. Klient potvrdí príjem paketom *ACK*. Obe strany takto prejdú do stavu *estabilished*.

TCP hlavička nasleduje v pakete bezprostredne za IP hlavičkou. Dôležité sú položky:

- *flags* (URG, ACK atd.) – pole s príznakmi nesúce informáciu potrebnú pre detekciu stavu druhej strany. Nie sú dovolené všetky kombinácie.
- *sequence number* – 32–bitové číslo pre zaručenie sekvenčného doručovania paketov.
- *acknowledgment number* – 32–bitové číslo pre zaistenie spoľahlivosti doručenia. Úspešne prijatý paket je odosielateľovi potvrdený *ACK* paketom, ktorý má ako *acknowledgment number* *sequence number* posledného prijatého bajtu.
- *window size* – veľkosť okna, ktorá udáva maximálny počet dát ktoré môže druhá strana poslať.



Obr. 1.5: Hlavička TCP paketu. Prevzaté z [5]

1.3.2 UDP

Nevýhodou TCP je, že sa prenáša príliš veľa riadiacich informácií a veľká záťaž siete (musí sa potvrdzovať každý prijatý paket). Preto nám transportná vrstva ponúka možnosť využiť druhý z dvojice používaných protokolov – UDP.

UDP protokol nie je spojovo orientovaný a rovnako ako TCP používa číslo portu na identifikáciu programu na aplikačnej vrstve. Čísla portov TCP a UDP sú na sebe nezávislé. Pakety UDP nie sú potvrdzované, takže nie je možné zaručiť spoľahlivosť. Rovnako tak nie je zaručená správnosť dát, pretože UDP hlavička neobsahuje kontrolný súčet. Pakety môžu prísť do cieľa v rôznom poradí, prípadne môžu prísť duplikované pretože neexistuje sekvenčné číslo.

Využitie UDP sa naskytá v rôznych multimediálnych prenosoch, videokonferenciách apod., kde sa prenáša veľké množstvo dát ktoré nie je nutné potvrdzovať a výsledná informácia nie je závislá na každom doručenom pakete.

1.4 Aplikačná vrstva

1.4.1 DHCP

Dynamic Host Configuration Protocol alebo skrátene DHCP umožňuje nastaviť sieťové parametre sieťových zariadení na základe konfigurácie uloženej na DHCP serveri. Ku komunikácii protokolom

DHPC sa používajú UDP pakety ktoré klient vysiela z portu 68 na port 67 na ktorom počúva server. Komunikáciu zahajuje klient všesmerovým paketom. Ak sa DHCP server nachádza za routerom, je potrebné mať na routeri aktívne DHCP relay – predávanie DHCP správ medzi sieťami. DHCP protokol pozná osem správ.

Správa	Popis
DHCPDISCOVER	Všesmerové vysielanie klienta za účelom nájdenia servera.
DHCPOFFER	Odpoveď servera na dotaz klienta s ponukou konfiguračných parametrov.
DHCPREQUEST	1. Žiadosť o parametre ponúknuté DHCP serverom.
	2. Potvrdenie správnosti parametrov získaných skôr od DHCP serveru.
	3. Predĺženie doby platnosti parametrov získaných skôr od DHCP serveru.
DHCPACK	Server potvrdzuje pridelenie parametrov požadovaných v DHCPREQUEST.
DHCPNAK	Zamietnutie žiadosti o parametre alebo vypršanie doby platnosti parametrov.
DHCPDELINCE	Klient oznamuje serveru, že IP adresa je už použitá.
DHCPRELEASE	Klient sa vzdáva pridelenej IP adresy a ruší prenájom.
DHCPINFORM	Klient s nakonfigurovanou IP adresou sa dotazuje na dodatočné informácie

Tab. 1.6: Osem správ DHCP protokolu

Ako už bolo spomínané v tabuľke, DHCP server adresy prideľuje len na určitú časovú dobu. Po jej uplynutí je potrebné pôžičku obnoviť. Dole je uvedený záznam jeden pôžičky ako si ho udržiava DHCP server. *Starts* a *ends* vymedzujú dobu platnosti. *Tstp* určuje, kedy končí platnosť pôžičky ak sa použije *failover* protokol (pri použití dvoch DHCP serverov, automatické presmerovanie správ pri výpadku jedného servera). *Binding state active* znamená, že pôžička je aktívna. *Next binding state* definuje, do akého stavu sa dostane pôžička po vypršaní časového limitu.

```
lease 172.17.2.3 {
  starts 4 2009/07/16 11:54:39;
  ends 4 2009/07/16 12:54:39;
  tstp 4 2009/07/16 11:54:39;
  binding state active;
  next binding state free;
  hardware ethernet 00:50:56:c0:00:01;
  uid "\001\000PV\300\000\001";
}
```

1.4.2 DNS

Domain Name System je hierarchický systém doménových mien, ktorý je realizovaný servermi DNS a protokolom rovnakého mena. Jeho hlavnou úlohou a príčinou vzniku je vzájomný prevod doménových mien a IP adries.

Priestor doménových mien na internete je tvorený stromovou štruktúrou. Koreňom stromu je špeciálna doména nultého rádu, ktorá sa označuje bodkou. Názvy priamych potomkov každého uzlu musia byť rôzne. Celý doménový názov je potom tvorený cestou od koreňa k nejakému uzlu v strome, kde skok medzi uzlom a jeho nasledovníkom na ďalšej úrovni je označený bodkou.

Výsledné meno sa píše v opačnom smere, koreň je najviac vpravo. Doménam na druhej úrovni pod koreňom sa hovorí doména najvyššej úrovne, TLD (Top Level Domain). Každé doménové meno (uzol hierarchie systému) má pridelené tzv. *autoritatívne DNS servery*. Je to množina serverov, ktoré nesú informácie relevantné k doméne. Môžu obsahovať už konkrétne informácie (IP adresy, názvy mail serverov atd.) alebo názvy ďalších DNS serverov, ktoré sú autoritatívne pre subdomény. Množina autoritatívnych serverov pre konkrétnu doménu sa často delí na jeden *primárny* a jeden alebo viac *sekundárnych* DNS serverov. Primárny server je hlavným nositeľom informácie a miestom kde administrátor vykonáva úpravy a nastavenia. Akonáhle sa na tomto serveri vykonajú zmeny, sekundárny server si tieto zmeny synchronizuje zónovým presunom AXFR. Mimo autoritatívnych DNS serverov existujú aj tzv. *cache DNS* servery. Pretože DNS záznamy sa vyšších úrovniach menia len sporadicky, je zbytočné sa zasielať dotazy na záznam pre každú časť domény. Cache servery sprostredkujú celý mechanizmus prekladu IP adries a výsledky si priebežne ukladajú do pamäte. Výsledkom pre opakovaný dotaz je rýchlejší preklad a menšie zaťaženie siete.

DNS servery komunikujú s ďalšími DNS servermi a klientmi pomocou protokolu TCP aj UDP, v oboch prípadoch na porte 53. Dotaz sa posiela v UDP pakete. Ak sa odpoveď zmestí do UDP paketu je odoslaná. Ak nie, je odoslaná iba jej časť a klient sa rozhodne, či mu čiastočná odpoveď stačí alebo sa má použiť TCP spojenie na prenos celej informácie.

Záznamy na autoritatívnom serveri sú rozdelené do *zónových súborov*. Každý takýto súbor obsahuje záznamy o konkrétnej doméne a môže obsahovať záznamy o svojich subdoménach ak nemajú vlastné zónové súbory. Každý riadok (záznam) zónového súboru obsahuje položky:

- názov domény
- TTL záznam
- trieda – rodina protokolov ku ktorej sa záznam vzťahuje
- typ záznamu (A, MX, SOA, NS, CNAME, PTR apd.)
- hodnota

SOA, Start Of Autority je typ záznamu, ktorý sa muší v zónovom súbore vyskytovať práve jeden krát. Ide o akúsi hlavičku. Záznam typu A je 32 bitová IPv4 adresa prislúchajúca doméne. Typ NS je zoznam názvov autoritatívnych serverov pre danú doménu. MX je názov mailserveru kam sa má doručovať elektronická pošta. Názov predchádza priorita ak sa použije viac mailserverov. CNAME záznam oznamuje, že daná doména je aliasom domény inej. Ak pre doménu existuje CNAME záznam, nesmie pre ňu existovať žiaden ďalší. Systém DNS neposkytuje len mechanizmus prekladu mien na IP adresy ale aj naopak. Na tento účel slúži typ záznamu PTR.

2 Pripojenie vstavaných systémov k internetu

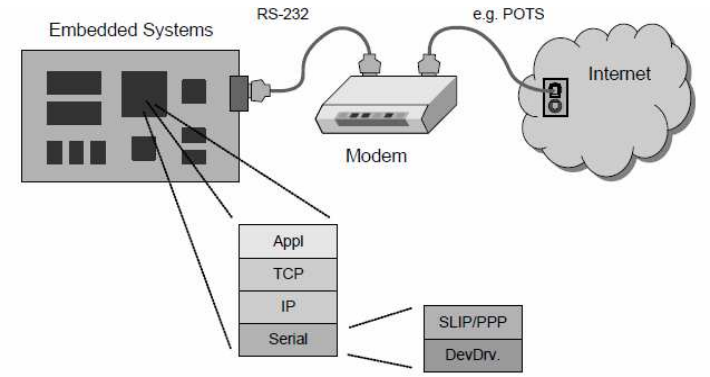
Prístup k internetu je dnes jednou z hlavných požiadaviek na vstavané systémy. Internetové pripojenie alebo obecne, pripojenie k sieti môže byť vnímané ako základná vlastnosť pre súčasné aj budúce aplikácie. Dokonca aj dnešné cenovo dostupné mikroprocesory sú schopné ovládať radič sieťového rozhrania LAN, komunikačné API a multivláknové aplikácie.

Existujú dva typy aplikácií, ktoré sa líšia potrebou prístupu na internet. Prvá kategória vyžaduje pripojenie na sieť ako svoju základnú vlastnosť aby sa dosiahla požadovaná funkčnosť. Toto predstavujú všetky mobilné zariadenia, zdravotnícke zariadenia, inteligentné domy alebo zabezpečovacie centrály. Pre druhú kategóriu aplikácií nie je pripojenie nevyhnutné, ale zvyšuje flexibilitu daného systému, napr. vzdialená údržba/správa, monitoring alebo aktualizácia softwaru.

V nasledujúcich podkapitolách sú diskutované tri možné spôsoby, ako pripojiť vstavané systémy k internetu.

2.1 Pripojenie pomocou modemu

Vstavaný systém je k modemu pripojený sériovým rozhraním (napr. RS232). Na druhej strane je modem pripojený k telefónnej, ISDN alebo xDSL linke. Všetku funkcionalitu spojenú s internetovým pripojením musí obstarávať vstavaný systém. Musí implementovať TCP/IP protokol a protokol linkovej vrstvy (PPP, SLIP).

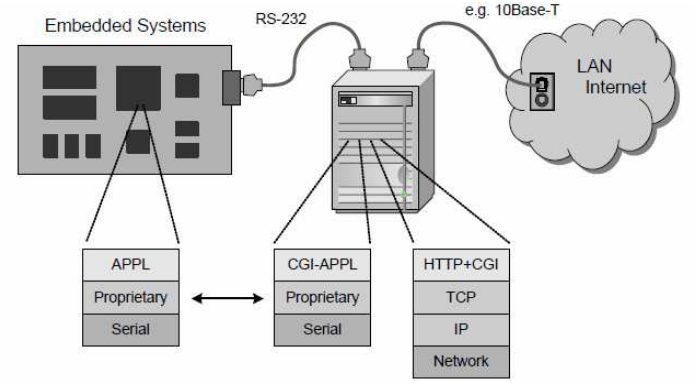


Obr. 2.1: Pripojenie pomocou modemu. Prevzaté z [1]

Takéto zapojenia sa používajú hlavne v domácnostiach a miestach, kde nie je priamy prístup do ethernetovej siete. Dáta preto musia byť modulované pre prenos po inom type linky.

2.2 Gateway počítač

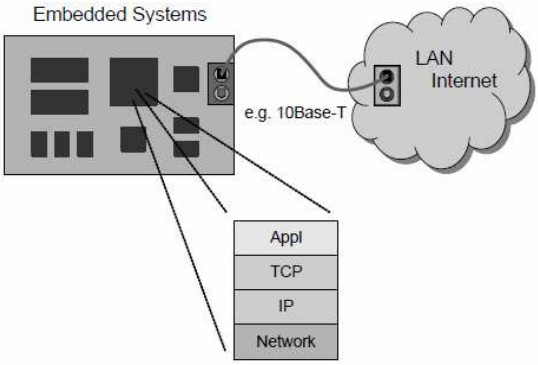
Vstavaný systém a *gateway* počítač sú spojené sériovým rozhraním. *Gateway* počítač je sám o sebe vybavený štandardným LAN/Internetovým rozhraním. Všetok software súvisiaci s internetovým pripojením implementuje *gateway* počítač (TCP/IP, Web server, CGI). Nie je potrebný žiadny štandardný protokol na spojenie vstavaného systému a *gateway* počítača.



Obr. 2.2: Pripojenie pomocou gateway počítača. Prevzaté z [1]

2.3 Priamym pripojením na LAN

Priamym pripojením na LAN je myslený vstavaný systém so zabudovaným radičom LAN rozhrania. Všetok ovládací software je implementovaný spolu s aplikáciou vo vstavanom systéme. Toto riešenie je v súčasnosti používané najviac pre dostupnosť a cenu potrebných komponentov. Nasledujúce podkapitoly ukazujú tri možnosti ako zabezpečiť priamu konektivitu vstavaného systému.



Obr. 2.3: Priame pripojenie. Prevzaté z [1]

Priame pripojenie a pripojenie pomocou *gateway* počítača potrebujú priamy prístup na ethernetovú sieť, čo nemusí byť vždy samozrejmé. Používajú sa však najviac.

2.3.1 LANTRONIX XPort

XPort je pripojený k sériovému rozhraniu mikroprocesora a dovoľuje prístup na internet alebo akúkoľvek inú sieť. Integruje v sebe kompletné TCP/IP programové rozhranie, RTOS bežiaci na výkonnom mikroprocesore DSTni–EX a 10/100 Mbit rozhranie Ethernet. Mechanicky je XPort integrovaný do konektoru RJ45. XPort obsahuje taktiež vlastný webový server používaný na vzdialenú konfiguráciu a odstraňovanie závad.

Vybrané vlastnosti:

- Vstavaný webový server
- Kompaktné rozmery v podobne konektoru RJ45
- E–mail
- 128, 192 alebo 256 bitové AES šifrovanie
- Výkonný, 22 MIPS procesor pri 88MHz
- Ochrana heslom

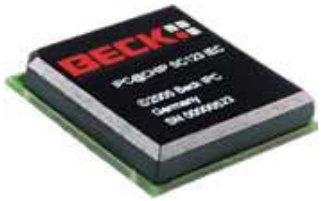
Obr. 2.4: XPort

2.3.2 IPC@CHIP

IPC@CHIP nemeckej firmy *Beck* je vstavaný WEB/LAN radič o rozmeroch integrovaného obvodu. Tvorí ho 16 bitový procesor 186, RAM, Flash, ethernetové rozhranie, watchdog a správa napájania. Predinštalovaný software zas obsahuje RTOS s podporou súborového systému, programové rozhranie TCP/IP, webový server, FTP, Telnet a vysoko úrovňové API pre programátora. Cieľom je vtesnať celú aplikáciu do jedného čipu.

Vybrané vlastnosti:

- 96 MHz CPU 186SX
- 8 MB RAM, 2MB Flash
- Jedno rozhranie 10/100 Mbit Ethernet + jedno rozhranie MII
- USB host a device
- CAN 2.0b



Obr. 2.5: IPC@CHIP

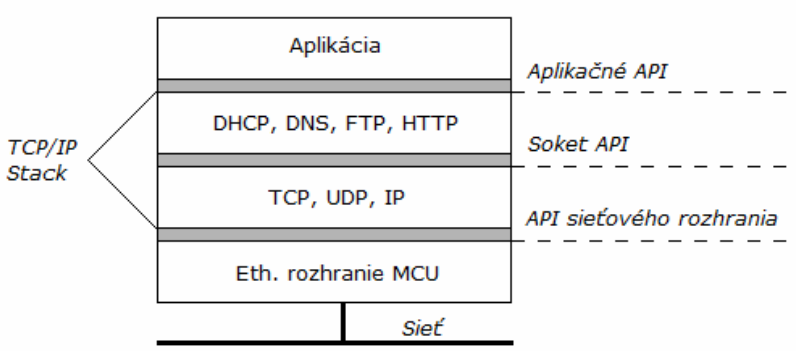
2.3.3 High – end a low – cost mikrokontroléry

Vstavané systémy založené na high – end 32 bitových mikrokontrolérech sú dnes zvyčajne vybavené ethernet rozhraním zabudovaným priamo na čipe (Freescale MCF5223x, Atmel AVR32 UC3A, Microchip PIC18F97J60, ...). Sú dostatočne výkonné na operačný systém ako *µCLinux*. Spolu s démonom *inet* tak vytvárajú plnú podporu pre pripojenie do internetu. Softwarové API tvoria obvykle BSD sokety alebo ich variácie.

Slabšie, low – cost, 8 a 16 bitové mikrokontroléry väčšinou ethernetové rozhranie nemajú. Majú však veľké množstvo sériových rozhraní a voľne využiteľných vstup/výstupných portov. Na dosiahnutie sieťovej konektivity je potrebné pridať špecializované hardwarové obvody. Jedným z takýchto obvodov je ENC28J60 od firmy *Microchip* a k hostiteľskému mikrokontroléru sa pripája rozhraním SPI. Integruje v sebe MAC a 10Base–T fyzickú vrstvu.

3 TCP/IP stack

Vstavané systémy majú obmedzenú výpočtovú kapacitu, ale hlavne obmedzenú pamäť a preto nemôže byť použitá implementácia TCP/IP z osobných počítačov. *TCP/IP stack* je odľahčená implementácia sady TCP/IP protokolov, ktorá poskytuje konektivitu pre takéto vstavané systémy. Pritom dodržiava všetky štandardy RFC.



Obr. 3.1: TCP/IP Stack tvorí vrstvu medzi aplikáciou a sieťovým rozhraním

Na obrázku 3.1 je vidno, aké miesto má TCP/IP stack v systéme. Užívateľskej aplikácii poskytuje softwarové API na aplikačné protokoly, s protokolmi transportnej vrstvy komunikuje klasicky cez sokety a ovláda ethernetové rozhranie, ktoré je súčasťou mikrokontroléru.

TCP/IP stack požaduje, aby sa mohlo v rovnakom čase vyskytnúť viac udalostí. Toto je dosiahnuté použitím jeden z dvoch metód: *loop* alebo *multitaskovým operačným systémom*. Pri *loop* beží program v nekonečnej slučke a každá udalosť predstavuje funkčné volanie a zdieľa sa spoločný zásobník. Druhou možnosťou je použiť multitaskový RTOS, kde je sa pre každú udalosť vytvorí nový proces a miesto na zásobníku.

3.1 OpenTCP

OpenTCP je robustná a prenositeľná implementácia rodiny TCP/IP protokolov pôvodne vyvinutá firmou *Viola Systems* ako open source software. Podporuje protokoly ARP, IPv4, ICMP, UDP, TCP, HTTP, BOOTP, TFTP, POP3 a SMNP. Obmedzená verzia OpenTCP pre 8 a 16 bitové mikroprocesory je kompatibilná s CodeWarrior IDE. Aby bolo možné použiť OpenTCP aj na systémoch s veľmi malou FLASH, ROM a RAM pamäťou, boli vykonané zmeny v implementácii protokolov, ktoré nie v súlade s RFC, ale udržiavajú si plnú funkčnosť.

- chýba podpora IEEE 802.3 paketov
- chýba podpora fragmentácie paketov
- ICMP podporuje iba echo pakety
- ignorujú sa polia *options* v TCP aj IP paketoch
- každý TCP paket musí byť potvrdený pred vyslaním nasledujúceho

OpenTCP v sebe integruje aj nízko úrovňový ovládač pre ethernet rozhranie mikroprocesoru (8 a 16 bitové procesory freescale s ethernetovým rozhraním). Je ale možné použiť ovládač dodávaný výrobcom čipu. OpenTCP je štruktúrovaný do ôsmich adresárov. Ich prehľad je v tabuľke 4.1.1.

Podadresár	Popis
`..\Doc`	Obsahuje užívateľský manuál
`..\OpenTCP`	OpenTCP zdrojové súbory
`..\NE64_OpenTCP`	Obsahuje OpenTCP súbory projektu v CodeWarrior
`..\NE64_OpenTCP\bin`	Preložené súbory projektu v CodeWarrior
`..\NE64_OpenTCP\prm`	Súbory prm projektu. Obsahujú informácie potrebné pre linker.
`..\NE64_OpenTCP\Ne64_drivers`	Zdrojové súbory nízko úrovnového ovládača eth. Rozhrania
`..\NE64_OpenTCP\sources`	Zdrojové súbory výslednej aplikácie, vrátane funkcie main().
`..\NE64_OpenTCP\web`	Súbory týkajúce sa výslednej webovej stránky.

Tab. 3.2: Adresárová štruktúra OpenTCP projektu

3.2 uIP

uIP je TCP/IP stack pôvodne vyvinutý na *Swedish Institute of Science*. Dnes je vyvíjaný celosvetovou komunitou ako open source software. Na rozdiel od OpenTCP, uIP už dodržiava všetky odporučenia RFC. Podporuje tak fragmentáciu alebo opätovné zasielanie paketov pri vypršaní časového limitu. Veľkou výhodou uIP sú tiež pamäťové nároky. Zdrojové kódy zaberajú menej ako 100 kB pamäte a stack zaberie v RAM pamäti len niekoľko desiatok kB. uIP poskytuje programátorovi dva typy aplikačného rozhrania:

- *protosockets* – API podobné BSD soketom
- *raw API* – API založené na udalostiach

Raw API používa udalosťami riadené rozhranie, kde je aplikácia vyzývaná k reakcii na určité udalosti. Aplikácia bežiaca nad uIP je implementovaná ako funkcia, ktorú uIP volá, aby reagovala na špecifické udalosti. Udalosťami môžu byť príjem dát, odoslanie dát, ukončenie alebo nadviazanie spojenia.

Špecifické je pre raw API aj opakované zasielanie pri strate paketu. Väčšina TCP/IP stack–ov si vyslané pakety ukladá do bufferu a v prípade žiadosti o opätovné zaslanie má dáta k dispozícii. uIP si však vysielané pakety neukladá a spolieha sa, že v prípade retransmisie je aplikácia schopná dáta znova vygenerovať. Takýmto spôsobom sa uIP snaží šetriť pamäť

Základom uIP sú *protothreads*. *Protothread* je odľahčený druh vlákna, ktorý nepotrebuje zásobník a poskytuje blokujúci kontext nad udalosťami riadeným systémom. Úlohou *protothreads* je implementovať sekvenčné riadenie toku dát bez nutnosti komplexných stavových automatov alebo plného multivláknového prostredia. *Protothreads* poskytujú podmienené blokovanie v C funkciách.

3.3 NicheLite TCP/IP stack

InterNiche Technologies, Inc. je americká firma, zaoberajúca sa vývojom sieťového softwaru pre vstavané zariadenia. Modularita jej produktov umožňuje programátorom vhodne si zvoliť protokoly a služby ktoré potrebujú. Mimo všetky štandardné protokoly patriace do rodiny TCP/IP podporujú protokoly ako PPP, PPPoE, RTP, RTCP, IPSec, IKE, SSL alebo IPv6. V ďalšej časti textu sa budeme zaoberať výhradne NicheLite TCP/IP stack–om, pretože priamo súvisí z povahou práce.

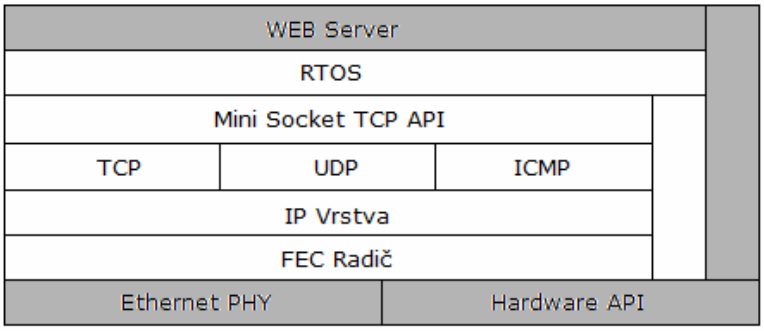
NicheLite stack môže pracovať v dvoch režimoch, *superloop* – nepoužíva sa RTOS a celý program tvorí jedna nekonečná slučka, alebo v režime multitaskového RTOS.

V *superloop* režime je každá úloha funkčným volaním a po ukončení sa musí vrátiť do hlavnej slučky. Všetky úlohy zdieľajú spoločný zásobník. Nevýhodou je, že úlohy sa nedajú vytvárať alebo rušiť dynamicky za behu programu a úloha sa nedá uspať.

V druhom režime, pri povolenom RTOS, sú úlohy vytvárané/rušené dynamicky. Pri vytvorení novej úlohy sa k nej vytvorí aj nový zásobník. Každá úloha má tak svoj vlastný zásobník a po jej ukončení sa pamäť vráti na hromadu. Veľkosť zásobníku je statická a určuje sa v dobe kompilácie. Každá úloha má pridelený kontrolný blok TCB (Task Control Block). TCB sú navzájom prepojené a plánovač jednoducho prechádza zoznamom a vykonáva úlohy na ktoré ukazujú TCB. Pretože RTOS je nepreemptivny, prepnutie úloh môže nastať iba ak sa bežiaca úloha sama vzdá prostriedkov.

Úloha sa môže nachádzať v dvoch stavoch, pozastavená alebo bežiaca. Pozastavená môže byť buď do vypršania časového limitu alebo do výskytu udalosti. Pozastavená úloha je plánovačom preskočená a šancu dostane znova až sa k nej plánovač vráti. Časovanie je vyriešené globálnou premennou, ktorej obsah sa periodicky zvyšuje. Ak je úloha pozastavená, do TCB sa uloží aktuálna hodnota premennej zvýšená o dobu po akú má byť úloha nečinná.

NicheLite podporuje tzv. zero–copy API. Sieťové servery posielajú značné množstvo dát zo súborov do siete. Zrýchlenie dosiahneme, ak budeme posielať dáta priamo z aplikácie do sieťového rozhrania pomocou DMA. Pri zero–copy API sa dáta zbytočne nekopírujú, ale do sieťového rozhrania sa presúvajú iba odkazy a DMA zaistí prenos dát.



Obr. 3.3: NicheLite TCP/IP stack

Rozhranie medzi RTOS a transportnou vrstvou tvoria mini sokety. Až na pár výnimiek je práca s nimi obdobná ako u BSD soketov. Mini sokety nepodporujú funkciu *accept()*. Namiesto toho implementujú *callback* funkciu, ktorá oznámi serveru, že je pripojený nový klient.

Na obrázku 3.3 je bielou NicheLite TCP/IP stack. K rozhraniu ethernet má prístup iba radič FEC. Stack má rozhranie s radičom v podobe IP vrstvy. Pretože RTOS nemá na starosti výlučne sieťovú komunikáciu, má prístup aj k ostatnému hardwaru mikrokontroléru.

4 Vstavaný webový server

Vzdialená správa alebo diagnostika vstavaných systémov má veľký význam pri mnohých zariadeniach a projektoch. Tradičné užívateľské rozhranie tvorené LCD displejom a klávesnicou sa používa čoraz menej a aj to len prípadoch, kedy ho nie je možné nahradiť.

Prístup k vstavanému zariadeniu cez internetový prehliadač zabezpečuje webový server bežiaci na zariadení. Web server si v pamäti spravuje a udržiava web stránky, na ktoré sa klient dotazuje pomocou HTTP protokolu.

4.1 HTTP protokol

HTTP je protokol typu dotaz – odpoveď. Klient sa dotazuje na web stránku uloženú sa serveri a server odpovedá zaslaním jej obsahu. HTTP protokolom sa neprenášajú iba web stránky ale obecne akékoľvek dáta. Klient žiada o dáta dvoma možnými spôsobmi, metódou GET alebo POST. Server odpovedá zaslaním HTTP hlavičky nasledovanú požadovaným obsahom. HTTP je bezstavový protokol a neobsahuje žiadny mechanizmus na zabezpečenie komunikácie. Spolieha sa na spoľahlivosť doručovania TCP.

```
GET /clanky/obsah.html HTTP/1.1
User-Agent: Mozilla/4.0 (compatible; MSIE 5.0; Windows NT)
Host: www.server.sk

HTTP/1.1 200 OK
Server: Microsoft-IIS/5.0
Date: Wed, 06 Dec 2009 13:37:40 GMT
Content-type: text/html
```

4.2 Dynamické HTML

Webový server síce k uloženým web stránkam prístup má, ale ak by ich mal modifikovať pri každom dotaze bolo by to veľmi nepraktické a náročné. Ak potrebujeme meniť obsah webovej stránky pri každom zobrazení máme dve možnosti. Použitie skriptovacieho jazyka alebo CGI.

Pri použití skriptovacieho jazyku je kód priamo súčasťou web stránky a interpret ho na strane serveru ešte pred zobrazením nahradí požadovaným obsahom. CGI, alebo *Common Gateway Interface* je rozhranie pre prepojenie externých aplikácii s webovým serverom. Server deleguje dáta externej aplikácii ktorá mu následne vráti statickú web stránku podľa požiadaviek.

Statické HTML stránky sú pre vstavané systémy nevhodné. U vstavaný systému nás zaujímajú predovšetkým vstupy a výstupy a tie sa menia dynamicky podľa okolností (hodnoty senzorov, výsledky AD prevodu, systémové premenné apod.). Vo väčšine prípadov sú to iba zmeny číselných

hodnôt ktoré nás zaujímajú a vzhľad stránky sa výrazne nemení. Preto by bolo použitie skriptovacieho jazyka zbytočne nevyužité.

NicheLite stack ponúka riešenie v podobe dynamických HTML tokenov. Dynamický token je umiestený niekde v HTML stránke a môže byť dynamicky nahradený obsahom premennej na ktorú ukazuje. Token môže ukazovať iba na jednu položku špeciálneho poľa. To ktorá premenná bude mapovaná na akú pozíciu poľa určuje programátor v dobe kompilácie. Veľkosť poľa sa môže ľubovoľne meniť. Token má tvar `~xxB;` , kde `xx` je index do poľa a `B` je použitá sústava.

```
<html><p>Data: ~05H;</p></html>
```

4.3 Súborový systém

Webové stránky alebo iné dáta podľa povahy serveru (HTTP server, FTP server apod.) musia byť uložené v pamäti. Pretože sa dáta na serveri často menia, potrebujeme pamäť s rýchlymi read/write operáciami a nezávislú na pripojenom napájacom napätí. Pri vstavaných systémoch máme isté obmedzenia. Môžeme použiť pamäť typu *EEPROM* alebo *flash EEPROM*. Kým prvá spomínaná je pomaly na ústupe, pamäť typu flash je masívne rozšírená.

Väčšina mikrokontrolérov štandardne obsahuje práve flash pamäť na uloženie kódu aplikácie. Typicky o veľkosti nepresahujúcej 512 kB. Jedným zo spôsobov kde uložiť dáta je vyčleniť časť internej flash pamäte pre dáta a časť pre program. Nevýhodou takého riešenia sú dve malé pamäťové časti. Zložitá aplikácia alebo RTOS obmedzujú veľkosť uložených dát a naopak.

Riešením je ponechať internú pamäť programu a dáta uložiť do pamäte externej. Externá flash pamäť môže byť pripojená sériovým rozhraním, alebo priamo na dátovú zbernicu, ak ju má mikrokontrolér vyvedenú na piny. Nevýhodou je pomalší prístup k dátam po sériovom rozhraní.

V internej flash pamäti dokáže vytvoriť dokonca dva rôzne typy súborových systémov, statický a dynamický. Statický FFS (Flash File System) je vytvorený súbor jazyka C v čase kompilácie. V textovom súbore sú uložené názvy všetkých súborov alebo adresárov ktoré chceme vo FSS mať. Aplikácia ich potom konvertuje do jedného C súboru, ktorý obsahuje niekoľko polí. Dáta z každého súboru sú konvertované do poľa s rovnakým názvom. Nasleduje pole mien súborov, veľkostí, ukazovateľov a pole formátu súborov. Na jednotlivé súbory sa server odkazuje cez tabuľku ukazovateľov.

Kým v statickom FSS je vhodné uchovávať dáta ktoré sa nemenia, dynamický FSS musí ponúka možnosť zápisu po ethernete. Aj keď je možné dáta v dynamickom FSS meniť, do flash pamäte sú nahrávané ako jeden obraz, v ktorom je uložená verzia, hlavička FAT, tabuľka FAT a dáta. Pri zmene dát sa obyčajne nepoužíva protokol FTP, ale klasická HTTP hlavička a port 80. Je to z dôvodu, že mnohé firewally blokujú FTP/TFTP prenos. Pri zmene sa mení celý obraz v pamäti.

```
/* Hlavička FAT */

Long[0] - FAT version
Long[1] - FAT id
Long[2] – Image size in bytes
Long[3] - Number of files in image
Long[4] - Pointer to the first file descriptor

/* Jeden záznam FAT tabluky */

typedef struct{
      unsigned longfile_pointer;
      unsigned longfile_size_in_bytes;
      unsigned longfile_type;
} FAT_FILE_DESCRIPTOR;
```

Externá flash pamäť býva spravidla väčšej kapacity a preto sa viac hodí na uloženie dát serveru. NichLite stack neobsahuje ovládače sériového rozhrania SPI ani ovládač pripojenej sériovej pamäte. Pri kapacitách externých pamatí je nutné použiť sofistikovaný súborový systém. Funkcie rozhrania potom podporujú blokové čítanie/zápis a mazanie. Je možné použiť napríklad komerčný súborový systém *smxFFS*.

V prípade, že chceme mať na serveri uložené značné množstvo stránok, alebo server používať ako sieťové uložisko dát, nestačí nám ani kapacita externej flash pamäte. Riešením je použitie veľkokapacitnej pamäťovej karty.

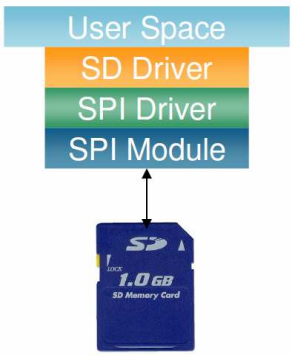
4.3.1 SD karta

Secure Digital (SD) karta je jedným z mnohých typov v súčasnosti používaných pamäťových kariet. Štandard SD kariet je dielom spolupráce firiem *SanDisk*, *Thosiba* a *MEI* a vychádza zo staršieho MultiMediaCard (MMC) štandardu. Rozmery, elektrické rozhranie a komunikačný protokol sú obsahom špecifikácie SD karty. SD karta podporuje tri komunikačné protokoly, z ktorých dva sa od seba kompletne odlišujú.

- *1 – bitový SD protokol* – je synchrónny sériový protokol s jednou dátovou, jednou riadiacou a jednou synchronizačnou linkou. Prenášajú sa bloky dát a explicitne podporuje zdieľanie zbernice.
- *4 – bitový SD protokol* – je totožný s predchádzajúcim, ale na prenos dát sa využívajú 4 paralelné linky. Dáta na linky rozdeľuje multiplexor a CRC sa počíta pre každú linku zvlášť.
- *SPI protokol* – je odlišný od oboch predchádzajúcich tým, že používa obecne známe sériové rozhranie. SPI je synchrónne sériové rozhranie využívané hlavne v oblasti mikrokontrolérov na pripojenie periférii. SPI režim podporuje iba podmnožinu protokolu SD.

Protokol SD karty je jednoduchý protokol typu príkaz – odpoveď. Všetky príkazy zasiela nadradené zariadenie. Pri dátovom prenose SD karta odpovedá definovaným rámcom za ktorým nasleduje dátový token indikujúci začiatok hromadného prenosu dát alebo chybový kód. Toto nie je

ale jediný spôsob prenosu dát. Pri druhom spôsobe sa prenášajú bloky dát definovanej veľkosti (typicky 512 bajtov). Každý blokový zápis musí byť potvrdený zo strany SD karty.



Obr. 4.1: Prístup na SD kartu cez SPI. Prevzaté z [8]

Pri blokovom zápise alebo čítaní je vždy adresovaný jeden blok pamäte o definovanej veľkosti. U hromadného prenosu je adresovaný prvý blok pamäte a adresa je postupne inkrementovaná radičom karty, pokiaľ je aktívny signál výberu karty (CS). Takýto prístup na kartu si vyžaduje dva ovládače. Jeden ovládač sériového rozhrania SPI a jeden ktorý ovláda SD protokol. Výhodou tohto prístupu je jednoduchosť. V najjednoduchšom prípade si vystačíme troma funkciami ovládača SD karty – *Init(), SD_Read_Block(), SD_Write_Block()*.

Nevýhody sa objavia ak potrebujeme adresovať pamäte veľkej kapacity. Identifikovať množstvo dát podľa adresy je nevyhovujúce. Navyše ak chceme, aby malo prístup k pamäťovej karte viac zariadení, musíme mať dáta uložené v dobre známom formáte. Všetko toto rieši použitie súborového systému. Najčastejšie *FAT16* alebo *FAT32* pre rozšírenú podporu, detailnú dokumentáciu a nenáročnú implementáciu.

Obr. 4.2: Prístup s použitím súborového systému. Prevzaté z [8]

FAT súborový systém prináša transparentnosť prístupu. Ovládač FAT má potom funkcie ako *Open_File(), Write_File()* alebo *Read_File()* rovnaké pre rôzny pripojený hardware.

4.4 Komunikácia s okolím

Okrem ethernetového rozhrania je vhodné, aby VWS komunikoval s okolím v ktorom sa nachádza aj iným spôsobom. Získaval informácie z rôznych senzorov, externých pamätí ale na druhú stranu mohol prostredie ovplyvňovať pomocou efektorov (spínače, relé, DC motor, apod.). Pretože väčšina periférii dnes komunikuje sériovo, v ďalšom výklade sa zameriame výlučne na sériové rozhrania, ich výhody a nevýhody.

4.4.1 I^2C

I^2C je zbernica typu multimaster. Na zbernici tak môže existovať viac ako jedno zariadenie typu master schopné zbernicu ovládať bez vzniku kolízií. Zbernicu tvoria iba dva vodiče, dátový SDA a hodinový SCK. Oba vodiče sú typu otvorený kolektor a ich maximálna dĺžka je daná ich maximálnou prípustnou kapacitou 400 pF. Takto sú zaručené časy nábežnej a zostupnej hrany hodinového signálu a rýchlosť 400 kbps.

Každé zariadenie na zbernici je identifikované 7 bitovou adresou. Po zachytení štartovacej podmienky porovnávajú všetky obvody svoju adresu s adresou na zbernici. Ak zistí niektorý z obvodov, že vysielanie je určené jemu, musí potvrdiť prijatie adresy bitom ACK. Potom prijíma alebo vysiela ďalšie dáta. Každému prenosu predchádza podmienka ŠTART. Nasleduje 7 bitová adresa a jeden bit R/W, ktorý indikuje požadovanú operáciu. Ďalší bit je ACK a za ním dáta, ktorých smer určil bit R/W. Po ukončení prenosu je vyslaná podmienka STOP.

	Štandardný režim	Rýchly režim	Veľmi rýchly režim
Rýchlosť [kbps]	100	400	3400
Max. kapacita vedenia [pF]	400	400	100
Nábežná doba (SCL) [ns]	1000	300	80

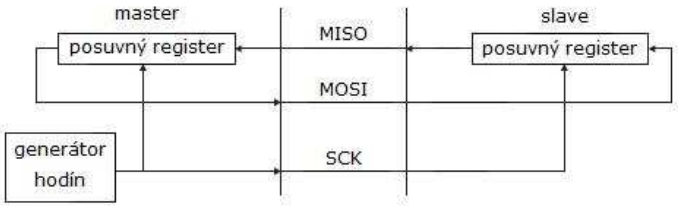
Tab. 4.3: Parametre zbernice I^2C

4.4.2 SPI

Rozhranie SPI je určené predovšetkým na pripojenie externých pamätí, A/D prevodníkov a rôznych senzorov. Niektoré mikrokontroléry využívajú SPI aj na programovanie vnútornej flash pamäte. SPI je trojvodičové rozhranie plus jeden výberový vodič – CS. Dátový výstup mastra – MOSI, vstup – MISO a hodiny – SCK.

Zariadenie s rozhraním SPI možno rovnako ako I^2C pripojiť na zbernicu. Výstupy každého zariadenia sú pripojené trojstavovým budičom, ktorého činnosť ovláda výberový signál. V prípade, že výberový signál nie je aktívny, sú výstupy v stave vysokej impedancie a zbernicu neovplyvňujú. Prenos na zbernici býva vždy medzi nadriadeným (master) a podriadeným (slave) zariadením.

Obvody obsahujú posuvné registre, ako je to na obrázku 4.4; master a slave teda v rovnakom čase môžu prijímať aj vysielať.



Obr. 4.4: Prepojenie master a slave obvodu

Začiatok prenosu iniciuje master. Nastaví generátor hodín a slave sa synchronizuje na použitú frekvenciu, spravidla v rozsahu 1 až 70 MHz. Konfigurácia hodín znamená nastavenie polarity (CPOL) a fáze (CPHA) hodinového signálu.

4.4.3 CAN

Zbernica CAN je rovnako ako I^2C typu multimaster. Nie je ale určená k pripájaniu periférií, ale iných mikrokontrolérov, inteligentných čidiel alebo iných akčných členov. Pôvodne bola určená pre automobilový priemysel (protišmykový asistent ABS), ale uplatnenie našla aj všade inde.

Všetky zariadenia pripojené na zbernicu vysielajú bez ohľadu na to, či má niekto o dáta záujem. Dátové rámce preto neobsahujú adresu, ale identifikátor aký druh dát rámec obsahuje. Rámec príjmu všetky zariadenia na zbernici, a podľa identifikátoru sa rozhodnú rámec akceptovať alebo nie. Existujú dve normy zbernice CAN, s 11 alebo 29 bitovým identifikátorom.

Zbernica je realizovaná jednou dátovou linkou na ktorej sa rozlišujú dve logické úrovne, dominantná a recesívna úroveň. Ak je na zbernici aspoň jedna stanica, ktorá vysiela dominantnú úroveň, má aj celá zbernica túto úroveň. V recesívnej úrovni je zbernica iba ak sú v rovnakej úrovni všetky zariadenia. Pre rozlíšenie úrovní je dôležitý rozdiel napätia, nie jeho hodnota. Takto je možné realizovať aj na veľmi veľké vzdialenosti. Pri dĺžke vedenia viac ako 1 km je ale prenosová rýchlosť len 50 kbps. Samotná norma CAN neobsahuje popis prenosového média preto sa v praxi používa viacero štandardov.

4.4.4 USB

Je univerzálna sériová zbernica (Universal Serial Bus). Predstavuje moderný spôsob pripojenia širokého spektra periférií k počítaču PC. Prenosové rýchlosti sa pohybujú v rozmedzí od 1 až do 480 Mb/s. Výhodou je možnosť napájať periférie priamo z portu USB. Rozhranie obsahuje 5 V napätie a umožňuje odber prúdu max. 100 mA. K jednému USB portu počítača je možné pripojiť až 127 rôznych zariadení. Zariadenia USB sa delia do troch kategórií: host, hub a device. USB host je

väčšinou počítač PC, riadi komunikáciu a prideľuje zbernicu. Hub rozširuje porty zbernice a umožňuje tak pripojenie tretej skupiny – USB device periférií.

I^2C	SPI	CAN	USB
• jednoduchosť • známy a overený protokol • plug & play • široké spektrum zariadení • nízka cena	• rýchlosť • nízka cena • široké spektrum zariadení • univerzálnosť	• bezpečnosť • rýchlosť na malé vzdialenosti	• rýchlosť • plug & play • jednoduchosť • nízka cena
• obmedzená rýchlosť	• chýba podpora plug & play • neexistuje štandard	• zložitosť • malé spektrum zariadení • komplikovaný Firmware	• komplikovaný USB host

Tab. 4.5: Výhody a nevýhody sériových rozhraní

5 Návrh

V predchádzajúcich kapitolách sme si objasnili čo je a ako funguje VWS, prečo sa oplatí pripájať vstavané zariadenia k internetu a rozobrali potrebné teoretické základy. Nasledujúca kapitola sa už zaoberá návrhom konkrétneho VWS na platforme Freescale. V návrhu sa postupne dostaneme od obecných častí ako návrh požadovaných funkcií a vlastností VWS cez blokovú schému až ku častiam konkrétnym ako výber súčiastok, návrh schémy zapojenia a DPS.

5.1 Vymedzenie požadovaných vlastností a funkcií

Na začiatok si musíme vymedziť funkcie aké po VWS požadujeme. Určiť v akom prostredí bude pracovať a akú úlohu plniť. Tomu podriadime celý návrh. Požiadavky na softwarové vybavenie VWS nie sú v tejto kapitole zahrnuté a budú diskutované neskôr.

Ideou je realizácia VWS, na ktorom bude možné v laboratóriách testovať a vyvíjať aplikácie pre vstavané systémy pripojené k internetu, no na druhej strane ho bude možné prakticky nasadiť a používať pri ovládaní zariadení na diaľku. Bude schopný zaznamenávať informácie z okolia v ktorom sa nachádza a bude schopný poskytovať vybrané sieťové služby. Napríklad sledovanie stavu siete, logovanie vybraných udalostí a odosielanie notifikačných emailov. Voľnú pamäť bude možné využiť ako úložisko dát dostupné cez HTTP alebo FTP. Žiadané vlastnosti VWS sú uvedené v nasledujúcich bodoch.

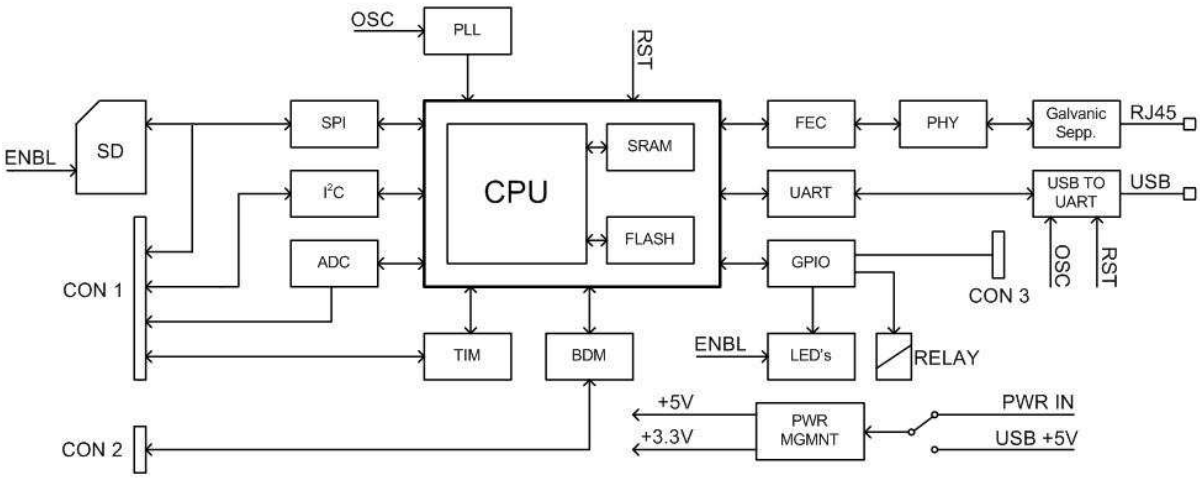
- *pripojenie na internet* – základná vlastnosť celého systému. VWS by mal obsahovať aspoň jedno rozhranie ethernet IEEE 802.3 kompatibilné.
- *veľká pamäť* – dostatočné veľká pamäť na uloženie väčšieho množstva web stránok a dát od užívateľov. Ideálne pamäťová karta o kapacite 1GB a viac.
- *univerzálne v/v rozhrania* – VWS musí mať možnosť pripojiť radu snímačov, senzorov a akčných prvkov. Periférie nebudú súčasťou VWS a signály rozhraní budú vyvedené do konektoru. Do konektoru nebudú vyvedené len signály rozhraní, ale aj nevyužité vstupy AD prevodníkov, vstupy/výstup čítačov/časovačov a obecné piny MCU. Niektoré periférie totiž nevyžadujú špeciálne rozhranie (servo, DC motor, relé apod.).
- *USB* – pri výpadku siete alebo zmene konfigurácie sa môže stať, že VWS nebude dostupný cez ethernetové rozhranie, preto musíme mať iný spôsob ako ho ovládať. VWS sa bude správať ako USB device.
- *programovacie a ladiace rozhranie* – pre pohodlné programovanie a ladenie programu priamo v mikrokontroléri. U procesorov Freescale to bude rozhranie BDM.
- *výkonný CPU* – procesor ktorý by zvládal v prípade potreby aj šifrovanú komunikáciu, zvládne obslúžiť väčšie množstvo pripojených klientov aj multivláknové RTOS.
- *nízka spotreba* – požadovaná vlastnosť všetkých vstavaných systémov. Procesor by mal funkciu správy napájania a spotrebu prispôsobiť aktuálnemu vyťaženiu. Znížiť frekvenciu hodín alebo sa úplne zastaviť ak nie je využitý. Funkčné bloky MCU, ktoré sa nevyužívajú by sa mali dať celkom odpojiť. Rovnako aj pamäťová karta napríklad cez MOSFET tranzistor. V prípade potreby sa bude môcť VWS napájať z batérie alebo USB.
- *malé rozmery* – kompaktné zariadenie, ktoré je možne bez problémov premiestniť podľa potreby.
- *SW podpora* – existujúci software (TCP/IP stack, RTOS, súborový systém, apod.), ktorý bude použiteľný s minimálnymi úpravami.

5.2 Bloková schéma a výber súčiastok

Bloková schéma vychádza z požiadaviek, ktoré sme si určili v predchádzajúcej kapitole. Základom je mikrokontrolér s výkonným jadrom, ethernetovým rozhraním a perifériami na čipe.

OSC je referenčný signál pre PLL a skutočná taktovacia frekvencia je vyššia. RST je externý signál reset. Pamäťová karta je pripojená k MCU cez SPI rozhranie. Signál ENBL umožňuje kartu celkom odpojiť od napájacieho napätia. Pretože SPI dovoľuje zbernicové zapojenie, sú signály SPI vyvedené aj do konektoru v prípade budúceho rozšírenia. Do konektoru sú vyvedené aj signály zbernice I^2C, vstupy AD prevodníkov, externé vstupy/výstupy čítačov a výstupy PWM modulácie. Programovacie rozhranie BDM je vyvedené to samostatného konektoru. Obecné vstup/výstupy ktoré ostanú nevyužité sú rovnako vyvedené do konektoru. Časť z nich je použitá na ovládanie LED

a jeden obecný výstup ovláda relé. LED sú zapojené cez trojstavový budič, takže sa dajú signálom ENBL od procesoru odpojiť. Len niekoľko málo mikroprocesorov má integrované USB rozhranie, zato skoro všetky majú asynchrónne sériové rozhranie UART. Ideálnym je preto prevodník USB – UART. Ethernetové rozhranie má dve časti, samotný radič ethernetu (Fast Ethernet Controller) a fyzickú vrstvu. Pretože ethernet využíva diferenciálny prenos, štandard IEEE 802.3 vyžaduje, aby bol ethernetový port elektricky oddelený od zvyšku obvodu. Galvanické oddelenie je preto nevyhnutné. Oddeľovacie transformátory sú integrované v konektore RJ45. VWS môže byť napájaný zo zdroja, alebo USB. V oboch prípadoch sa musí napätie upraviť na dve hodnoty, kvôli rôznym požiadavkám obvodov.



Obr. 5.1: Bloková schéma VWS

5.2.1 Výber súčiastok

Srdce celého systému je mikrokontrolér. Jeho vlastnosti udávajú vlastnosti celého VWS a preto je jeho výber kľúčový. Ako vyplýva z názvu práce, MCU musí byť od firmy Freescale a musí mať minimálne jedno ethernetové rozhranie. Freescale má v ponuke radu procesorov s možnosťou pripojenia k internetu, mnohé sú však industriálne mikroprocesory bez integrovanej flash a RAM pamäte. Ak má byť VWS čo najlacnejší a najjednoduchší, takéto mikroprocesory neuvažujeme. V nasledujúcom prehľade je zoznam vyhovujúcich mikrokontrolérov.

- *MC9S12NE64* – jediný 16 bitový mikrokontrolér s integrovaným rozhraním ethernet. Základom je jadro HCS12, 64 kB flash pamäte a 8 kB RAM. Mikrokontrolér ďalej ponúka 8 kanálový AD prevodník s 10 bitovým rozlíšením, 4 kanálový časovač s PWM výstupom a možnosťou reagovať na nábežnú alebo zostupnú hranu externého signálu, sériové rozhrania UART, SPI a I^2C, watchdog alebo integrovaným napäťovým regulátorom s funkciou Low Voltage a Power On restet.

- *MCF5223x* – 32 bitový MCU s jadrom ColdFire V2, 256 kB flash a 32 kB RAM. Maximálna frekvencia zbernice je 60 MHz. Samozrejmosťou je integrovaný radič rýchleho ethernetu 10/100 Mbit/s a príslušná fyzická vrstva, I^2C, SPI, UART, CAN alebo 4 kanálový DMA radič. Podľa konkrétneho typu, môže mať *MCF5223x* kryptografickú jednotku s hardwarovým generátorom náhodných čísel.
- *MCF5225x* – je 32 bitový MCU s jadrom ColdFire V2 taktovaným frekvenciou od 66 až do 80 MHz. Na rozdiel od *MCF5223x* poskytuje 512 kB flash pamäte a 64 kB RAM, rozhranie USB a Mini FlexBus. Každý typ má štandardne kryptografickú jednotku. Veľkou výhodou je plná kompatibilita softwarových produktov *Freescale MQX*.
- *MCF528x* – 32 bitový ColdFire V2 MCU, s 512 kB flash, 64 kB RAM a 2 kB inštrukčnej cache pamäte. Oproti *MCF5225x* má vylepšené AD prevodníky so vstupnou frontou. Procesor má vyvedené signály zbernice, preto je možné podľa potreby rozšíriť flash aj RAM pamäť.
- *MCF548x* – 32 bitový MCU na jadre ColdFire V4. S frekvenciou 200 MHz a výkonom 308 MIPS patrí k najvýkonnejším v kategórii. Nemá integrovanú flash pamäť, iba 32 kB RAM, 32 kB inštrukčnú a 32 kB dátovú cache. Signály zbernice sú samozrejme vyvedené na piny. Integrovaný radič pamäte (MMU) umožňuje pripojiť DDR alebo SDR DRAM. Podporuje hardwarovú akceleráciu šifrovacích algoritmov (DES, 3DES, AES, MD5, RC4) a má hardwarovú jednotku pre prácu s desatinnými číslami.

Všetky mikrokontroléry spĺňajú požiadavky uvedené v kapitole 5.2.1 Majú potrebné rozhrania, výkon a režimy nízkej spotreby. Okrem *MC9S12NE64* a *MCF5223x* disponujú aj rozhraním USB. Pri použití prevodníku to ale nie je veľký problém. Pre ethernetové aplikácie je výhodnejšia 32 bitová architektúra. Prichádzajúce aj odchádzajúce dáta prechádzajú skrz TCP/IP stack realizovaný ako zásobník. Prijaté dáta sa zhromažďujú na spodku zásobníku a postupne prechádzajú nahor, kde si ich prevezme aplikácia. Odosielané dáta sa postupne presúvajú smerom nadol. Namiesto časovo náročného kopírovania dát na zásobníku sa používajú ukazovatele (zero–copy API). Aritmetika ukazovateľov je efektívnejšia na 32 bitových procesoroch.

Procesory *MCF528x a MCF548x* síce požiadavky spĺňajú, ale sú na navrhovaný systém až príliš výkonné a väčšina vlastností by ani nebola využitá. Navyše sú oba dostupne iba v puzdre MAPBGA 256 resp. PBGA 388, ktoré je náročné osadiť v amatérskych podmienkach.

Pri ďalšom rozhodovaní zohrala rolu dostupnosť a kvalita dokumentácie, aplikačné poznámky, ukážkové aplikácie a rozšírenosť daného typu MCU. U návrhárov je viac rozšírená rodina *MCF5223x*, preto k nej existuje viac ukážkových príkladov a dokumentov. Na webových stránkach Freescale je dostupných niekoľko návodov ako pracovať s ethernetovým rozhraním, AD prevodníkmi a dokumentácia k InterNiche TCP/IP Lite stacku. Výhodou je aj dostupnosť MCU v puzdre LQFP 80. U rodiny *MCF5225x* takéto veci dostupné nie sú, pretože je všetko ponechané na dokumentáciu k *Freescale MQX*.

5.2.2 MCF52233

V predošlej kapitole sme si vytvorili zoznam vhodných kandidátov pre navrhovaný VWS. V tejto kapitole si povieme niečo viac o konkrétne použitom mikrokontroléri *MCF52233*.

Mikrokontrolér je postavený na základe 32 bitovej RISC architektúry ColdFire verzia 2. Jadro je taktované na 60 MHz a poskytuje výkon 56 MIPS (Dhrystone 2.1 Benchmark). Súčasťou jadra je EMAC (Enhanced MAC) jednotka podobná tým v DSP procesoroch a 16 obecne použiteľných registrov. EMAC poskytuje sadu DSP inštrukcií pričom podporuje inštrukcie násobenia inštrukčnej sady ColdFire. Podporované sú znamienkové aj bezznamienkové celé čísla. EMAC implementuje plne zreťazené 4-stupňové spracovanie s 32 bitovými vstupnými operandami a 48 bitovým akumulátorom. Ďalšie vlastnosti sú uvedené v zozname:

- *BDM* – programovacie a ladiace rozhranie vhodné pre real–time ladenie programu priamo v hardwari. Podporuje až 6 hardwarových *breakpointov*.
- *32 kB SRAM* – dvojportová SRAM podporuje DMA prenosy aj prenosy jadra.
- *256 kB Flash EEPROM*
- *Správa napájania* – možnosť zastavenia chodu celého procesoru alebo zastavenia hodinového signálu nepoužívaným perifériám. Veľmi rýchla odozva na prerušenia keď je procesor v úspornom režime.
- *FEC* – radič 10/100 BaseT/TX ethernetu, prenos half/full duplex mód. Vstupná a výstupná FIFO fronta na čipe. Zabudovaný DMA radič.
- *EPHY* – fyzická vrstva ethernetu.
- *UART* – asynchrónny sériový prenos. Súčasťou je 16 bitová delička hodinového signálu, logika prerušenia a podpora DMA prenosov. Sú dostupné riadiace signály RTS a CTS.
- I^2C
- *QSPI* – alebo Queued SPI. SPI rozhranie s frontou. Do fronty je možné uložiť 16 prenosov. Sú dostupné 4 výberové signály (CS) a MCU môže pracovať iba v režime master.
- *ADC* – 8 kanálový AD prevodník s 12 bitovým rozlíšením. Je možné vzorkovať dva kanály súčasne a jednorazový alebo kontinuálny prevod. Programovateľné prerušenia.
- *Časovače* – Procesor má na čipe štyri 32 bitové DMA časovače (schopné iniciovať DMA prenos), jeden obecne použiteľný čítač/časovač a jeden časovač generujúci PWM signál pre osem samostatných výstupov.
- *Obvod reálnych hodín* – môže pracovať vo funkcii stopiek alebo alarmu.
- *Periodic Interrupt Timer* – PIT, časovač generujúci pravidelné prerušenia.
- *Watchdog*
- *DMA radič*

5.2.3 SD karta

Nie všetky typy SD kariet sú vzájomne kompatibilné. Nové typy SD kariet (SDHC, SDXC) s kapacitou nad 2 GB majú mierne odlišný protokol. Rozmery karty sa nelíšia, ani módy v ktorých môže karta pracovať (SD alebo SPI). SDHC karty majú iný spôsob adresovania. Bytové adresovanie sa dá použiť iba do kapacity 4 GB. Pre väčšie kapacity sa používa adresovanie sektorov, ktoré nie je kompatibilné. SDHC karty dosahujú kapacít do 32 GB a čítačky SD kariet nie sú schopné pracovať s SDHC.

Najnovší štandard SDXC iba oficiálne rozšíril kapacitu SDHC kariet na 2 TB. Túto kapacitu už teoreticky mohli dosiahnuť SDHC, ale boli limitované SD 2.0 špecifikáciou. Novinkou je podpora súborového systému Microsoft *exFAT*.

Kapacita 2 GB je pre navrhovaný VWS dostačujúca, preto budú podporované iba karty SD špecifikácie SD 1.1.

5.2.4 FT232RL

FT232RL je USB – UART prevodník od firmy FTDI Chip. Podporuje protokol USB 1.1 aj USB 2.0 full speed. Obvod môže byť napájaný z USB alebo externe od 3.3 po 5.25 V. Dátové vstupy/výstupy sú CMOS/TTL kompatibilné od 1.8 do 5V. Pre zvýšenie rýchlosti majú vysielač aj prijímač vyrovnávacie pamäte. UART má riadiace signály pre RS232/RS485. Obvod má možnosť výstupu interných hodín 6, 12, 24 alebo 48 MHz pre použitie v aplikácii.

Obvod má integrovanú EEPROM, zdroj hodinového signálu a USB rezistory, čo je veľká výhoda. K iným prevodníkom treba tieto súčiastky pripojiť externe, napr. *FT232BM*. Ovládače pre operačné systémy MS Windows 2000/XP/Vista/7 Mac OS a Linux sú k dispozícii zdarma.

6 Realizácia

Nasledujúca kapitola sa zaoberá konkrétnou realizáciou VWS podľa návrhu s predchádzajúcej kapitoly. Kapitola je rozdelená do dvoch hlavných podkapitol, a to na hardwarovú a softwarovú časť. Hardwarová časť sa venuje použitým súčiastkam, schéme zapojenia, návrhu plošného spoja, osadeniu a oživeniu dosky VWS. Softwarová časť zas použitým TCP/IP stackom, súborovým systémom ako aj vlastnou implementáciou webového serveru a ukážkových aplikácií.

6.1 Hardwarová časť

6.1.1 Schéma zapojenia

Hlavnou súčiastkou je samozrejme procesor ColdFire *MCF52233CAF60*. Označenie *CAF60* označuje použité puzdro – LQFP80. Procesor sa dodáva iba v dvoch vyhotoveniach, druhou možnosťou je puzdro LQFP112. Použitie menšieho puzdra síce obmedzuje funkcionalitu, ale toto obmedzenie nie je výrazné. Nie je dostupné tretie rozhranie UART, tri výberové signály (Chip Select) SPI rozhrania, štyri výstupy PWM modulátora a jedenásť vstupov externého prerušenia. Vysoká miera funkcionality k pomerne nízkemu počtu I/O pinov je zaručená niekedy až štyrmi rôznymi funkciami na jednom pine.

Procesor má tri možnosti taktovania. Použiť interný obvod fázového závesu (PLL) s externým zdrojom referenčných hodín alebo ako zdroj referenčného signálu pre PLL použiť kryštál. Pri použití kryštálu je v procesore dostupný zosilňovač signálu. Poslednou možnosťou je PLL obvod zablokovať a celý procesor taktovať z externých hodín. V našom prípade je použitá druhá možnosť s PPL obvodom a externým kryštálom. Aby mohol PPL obvod generovať maximálny možný hodinový signál 60 MHz pre jadro procesoru, je pripojený kryštál s hodnotou 25 MHz. Rovnaký PLL obvod generuje hodinový signál aj pre fyzickú vrstvu ethernetového rozhrania EPHY.

Napájacie napätie procesoru je 3.3 V. Pretože procesor obsahuje obvody citlivé na šum v napájacom napätí (PLL, ADC prevodník) je napájacie napätie 3.3 V rozdelené na digitálne a analógové. Vzájomne sú odrušené feritovým jadrom. Cievka relé a USB prevodník si vyžiadali prítomnosť aj napájacieho napätia 5 V. Celý obvod je primárne napájaný zo zdroja, no v prípade potreby, ak odber nepresahuje 200 mA, je možné použiť aj napájanie z USB. *JP3* slúži na výber zdroja napájacieho napätia. Všetky dostupné úrovne a druhy napätí sú vyvedené do konektoru *POWER* alebo *ANALOG*.

Štandard procesorov Freescale je BDM/JTAG rozhranie. MCF52233 ponúka 26 pinové programovacie rozhranie spoločné pre BDM a JTAG. Voľba medzi použitým rozhraním sa deje privedením logickej „0“ alebo „1“ na pin JTAG_EN procesoru. Signály rozhraní BDM a JTAG

zdieľajú spoločné vývody procesoru a podľa voľby *JTAG/BDM* sa po reštarte procesora nastavia. Jediný rozdiel v zapojení signálov BDM a JTAG rozhrania do konektoru je v hodinovom signálne. Kým BMD má výstup hodín na pine 24, JTAG očakáva z toho istého vývodu procesoru vstup synchronizačných impulzov na pine 6. Toto je vyriešené voľbou *JP1*.

Pretože procesor nemá zabudované USB rozhranie, musel byť použitý prevodník. Konkrétne obvod FT232RL s minimálnym počtom externých súčiastok. Obvod má zbudovaný interný oscilátor alebo je možné použiť externý zdroj hodín. Pri použití interného oscilátora musí byť napájacie napätie minimálne 4 V, nie menej. Obvod je zapojený so samostatným napájaním (druhá možnosť je použiť napájanie priamo z USB) a je použitý interný oscilátor. Pri takomto zapojení je hlavnou požiadavkou, aby obvod nezaťažoval USB zbernicu žiadnym prúdom ak je USB host vypnutý. Preto je na vstupe RESET# pripojený napäťový delič napájaný z USB zbernice. Ak je USB host vypnutý, na vstupe RESET# je logická „0“ a obvod je nečinný. Sériové rozhranie UART má vlastný vstup napájacieho napätia – VCCIO. Takto je možné si ľubovoľne zvoliť napäťové úrovne s ktorými bude UART pracovať. K prevodníku sú zapojené dve LED diódy indikujúce priebeh R/W operácií.

6.1.1.1 Fyzická vrstva ethernetu

Fyzická vrstva ethernetu – EPHY, je kompatibilná so štandardom IEEE 802.3 10BASE-T / 100BASE-TX a podporuje rozhranie nezávislé na médiu (Medium Independent Interface – MII). Pre svoju funkciu potrebuje 25 MHz kryštál pripojený k PLL obvodu. Podpora 100 Mbps ethernetu vyžaduje internú zbernicu taktovanú na 25 MHz. U 10 Mbps je to len 2.5 MHz. Napájacie napätie 2.5 V dodáva interný regulátor napätia. Externé piny EPHY zaberajú väčšinu dostupných pinov procesoru. V tabuľke 6.1 je prehľad.

Funkcia	Názov	Popis
Napájanie	PHY_VDDA	Napájanie pre analógové obvody EPHY
	PHY_VDDA	
	PHY_VDDRX	Napájanie prijímača EPHY
	PHY_VSSRX	
	PHY_VDDTX	Napájanie vysielača EPHY
	PHY_VSSTX	
Signál	PHY_TXP	Krútená dvojlinka Tx a Rx
	PHY_TXN	
	PHY_RXP	
	PHY_RXN	
Bias	PHY_BIAS	Referenčný odpor
Indikácia	COLLED	Indikuje kolíziu v half duplex móde
	DUPLED	Indikuje mód. Half / full duplex
	SPDLED	Indikátor rýchlosti. 10 / 100 Mbps
	LNKLED	Indikátor aktívneho spojenia
	ACTLED	Indikátor prijatia dát

Tab. 6.1: Výstupy a vstupy fyzickej vrstvy

Pretože napájacie napätie dodáva interný regulátor, všetky napájacie piny PHY_VDD a PHY_VSS musia byť bez statickej záťaže a slúžia na pripojenie odrušovacích kondenzátorov. Interný napäťový regulátor je neaktívny, ak je pin VDDR pripojený na spoločnú zem. Referenčný odpor PHY_RBIAS musí mať pevnú hodnotu 12k4.

Ak použijeme integrovanú EPHY a MAC procesoru, pripojenie sieťového konektoru nie je zložité. Vysokorýchlostné magnetické oddelenie procesoru od siete je nevyhnutné. MCF52233 vyžaduje použitie transformátora s pomerom 1:1 k magnetickému oddeleniu prijímača a vysielača. Transformátor môže byť vyhotovený ako samostatná diskrétna súčiastka alebo integrovaný v konektore RJ45. Je lepšie použiť integrovaný, pretože znižuje počet potrebných komponent, zjednodušuje návrh a hlavne skracuje vodiče potrebné na prepojenie na minimum. Takto nevznikajú parazitné kapacity. Požiadavky na transformátor sú uvedené v tabuľke 6.2.

Parameter	Hodnota	Jednotka	Poznámka
Transf. pomer Tx / Rx	1:1 CT / 1:1	---	---
Indukčnosť	350	mH (min)	---
Priechodzí útlm	1.1	dB (max)	---
Spätný útlm	-18	dB (min)	1 až 30 MHz
	-14	dB (min)	30 až 60 MHz
	-12	dB (min)	60 až 80 MHz
Transformátor	1500	V	---

Tab. 6.2: Požiadavky na magnetické oddelenie

Freescale síce nedoporučuje žiadny konkrétny typ alebo značku oddeľovacieho modulu ktorý sa má použiť, ale má uvedený zoznam úspešne testovaných.

Typ	Výrobca	Model
Diskrétne	Pulse	H1102
	Midcom	6241-37R
Integrované	Pulse	J10-0026
	Midcom	MIC24xxx-1010
	Bel Fuse	0810-1X1T-06
	Halo	HFJ11-2450E

Tab. 6.3: Testované súčiastky

Zapojenie indikačných LED je voliteľné. V našom prípade sú zapojené iba dve, ACDLED a LNKLED. Obe LED diódy sú integrované v použitom konektore Midcom MIC24211-0101.

Nie je nevyhnutné použiť integrovanú fyzickú vrstvu EPHY procesoru. Radič ethernetu má totiž vyvedené všetky potrebné vstupy a výstupy na externé piny a preto je možné pripojiť akúkoľvek EPHY, ktorá podporuje rozhranie MII. Signály radiča sú dostupné vždy ako terciárna funkcia a pri

ich použití je výrazne obmedzená celková funkčnosť procesoru. Sú zablokované rozhrania UART0, UART1 a všetky piny obecného čítača/časovača GPT.

Aby nedochádzalo k odrazom signálu na konci vedenia, musí byť ethernetová linka správne ukončená. Charakteristická impedancia ethernetového káblu je 50 Ω, čo je presne hodnota rezistoru aká sa má použiť. Rezistor na konci vedenia predstavuje záťaž pre signál.

Obr. 6.4: Ukončenie ethernetovej linky

6.1.1.2 Pripojenie SD karty

Pretože protokol SD nie je otvoreným protokolom, jedinou možnosťou ako pripojiť SD kartu je rozhraním SPI. MCF52233 má jedno upravené SPI rozhranie s frontou – QSPI. Takto je možné uložiť do fronty až 16 transakcií naraz a vykonať ich bez zásahu programátora alebo CPU. QSPI má k dispozícii 80 B statickej RAM pamäte rozdelenej do troch častí. Pamäť je organizovaná tak, aby jedna transakcia zaberala presne 5 B. Jeden bajt pre príkaz, dva bajty pre vysielané dáta a dva pre prijímané dáta.

Kvôli šetreniu miestom je použitý slot pre *microSD* kartu. Vývody slotu sú pripojené k pinom QSPI rozhrania cez konektor z radových kolíkov. Takéto riešenie má hneď dva dôvody. SD kartu je možné fyzicky úplne odpojiť alebo namiesto nej pripojiť iné SPI slave zariadenia. Druhým dôvodom sú ďalšie funkcie pinov QSPI. Sekundárna funkcia je napríklad rozhranie I^2C. Práve kvôli sekundárnej funkcii sú do schémy pridané miesta pre pull–up rezistory, ktoré inak osadzovať netreba.

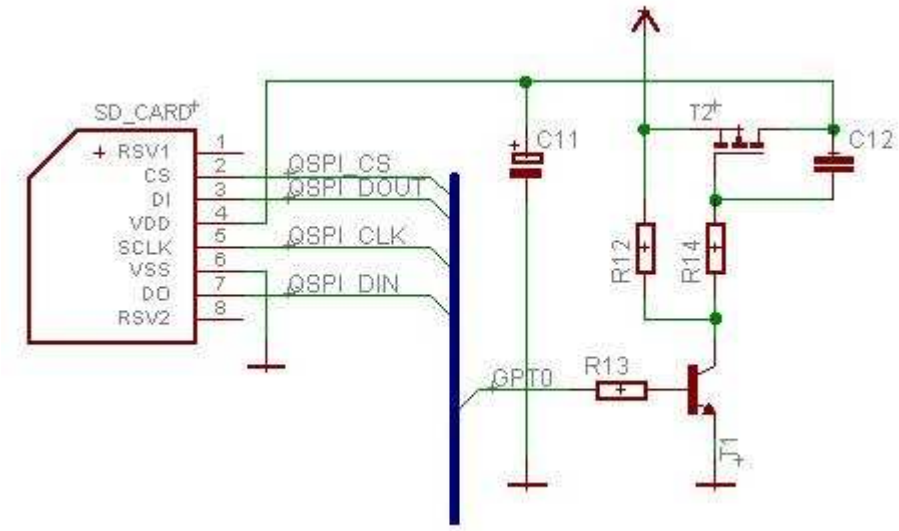
SPI zbernica pull–up rezistory nepotrebuje, ak je na zbernici vždy definovaná hodnota. Pri aktívnom výberovom signáli CS, sú aj výstupy SD karty aktívne a majú vždy definovanú úroveň. V opačnom prípade, pri CS = log „0“, sú výstupy v stave vysoká impedancia a na zbernici definovaná úroveň nie je. To ale nevadí, ak je k QSPI pripojená len SD karta. Ak je pripojených viac zariadení, je vhodné rezistory osadiť.

Použitý procesor má dostupný len jeden signál CS ovládaný priamo QSPI rozhraním. Ten je možné podľa potreby konfigurovať a potom je ovládaný automaticky radičom rozhrania. Ak by bolo potrebné na zbernicu pripojiť viac zariadení, je ako CS signál nutné použiť obecný výstup a funkciu implementovať v riadiacom softwari.

Za určitých okolností sa môže karta dostať do neidentifikovateľného stavu, v ktorom nemusí správne reagovať na príkazy z procesoru. Preto musíme mať možnosť ako kartu zresetovať bez

fyzickej prítomnosti pri VWS. Práve z toho dôvodu je napájanie karty riešené cez MOSFET spínací tranzistor, ktorý je ovládaný procesorom. Napájanie je tak možné kedykoľvek odpojiť a pripojiť a kartu resetovať. Ďalším dôvodom tohto zapojenia je spotreba. Ak karta nie je používaná, napájanie sa môže odpojiť. Z jedného výstupu procesoru je možné odoberať len veľmi malý prúd – 2 mA. Výstup je preto posilnený jedným NPN tranzistorom.

Kvôli obmedzeniu prúdového nárazu po zopnutí MOSFET spínača, je pripojený tzv. „soft start circuit“. Ide o sériové zapojenie kondenzátoru a rezistoru. Po zopnutí NPN tranzistoru, sa na hradle MOSFET spínača objaví napätie, ktoré s konštantou $\tau = R14 \cdot C12$ klesá k nule. Pre hodnoty súčiastok R14 = 1 kΩ a C12 = 100 nF sa napätie ustáli po 0.1 ms.



Obr. 6.5: Pripojenie SD karty

6.1.1.3 Relé

Aby bolo možné používať VWS „out of the box“, t.j. bez ďalších úprav a rozšírení, je osadený jedným bistabilným relé s dvomi prepínacími kontaktmi. Nie je preto problém pripojiť a ovládať iné externé zariadenia a spotrebiče.

Použité relé *AL–D5W–K* má dve 5 V ovládacie cievky, SET a RESET. Použité relé je polarizované bistabilné, čo znamená, že záleží na priloženej polarite napätia na cievke a cievka ostáva zopnutá vďaka permanentným magnetom aj bez prítomnosti napätia. Na ovládanie sa používajú krátke impulzy. K cievkam relé je pripojená ochranná dióda. Pri odpojení napätia dochádza k spätnej indukcii a dióda slúži ako skratovací obvod na ochranu tranzistoru. Relé je v puzdre DIL10 a oba prepínacie kontakty sú vyvedené do konektoru.

Maximálne spínané napätie je výrobcom stanovené na 250 VAC a maximálny spínaný prúd na 2 A. Výkon by však nemal prekročiť 30 W.

K dvom so štyroch vstupov externého prerušenia sú pripojené signalizačné LED diódy. Ak už bolo spomínané, jeden výstup procesoru môže byť zaťažený maximálne 2 mA. Preto sú LED pripojené cez PNP tranzistor. K druhým dvom IRQ vstupom sú pripojenú pull-up rezistory aby nespôsobovali náhodné vyvolanie prerušenia a sú vyvedené na externé piny. Všetky nevyužité vstupy/výstupy procesoru sú vyvedené do konektoru pre prípadné ďalšie rozšírenia.

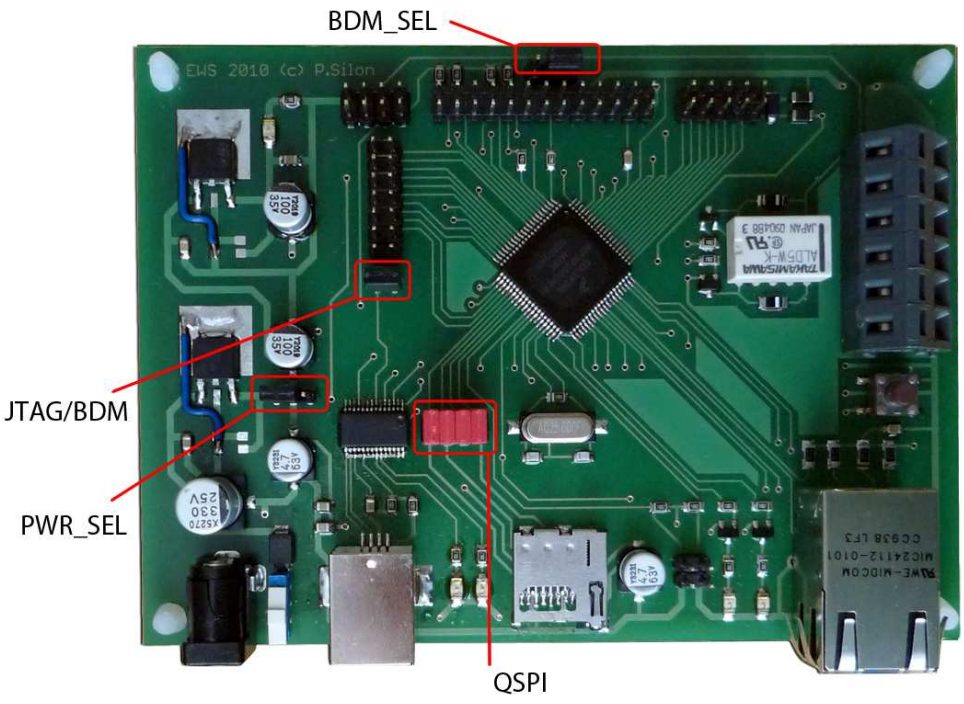
6.1.2 Konfigurácia

Pre správnu funkciu VWS je nutné ho pred zapnutím nastaviť. Nastavenie spočíva v osadení skratovacích prepojek (jumperov) na určité piny. Takto je možné konfigurovať zdroj napájacieho napätia, použité programovacie rozhranie a pripojenie/odpojenie SPI periférií.

Zdroj napájacieho napätia je možné voliť medzi externým adaptérom alebo napájaním z USB. Na to slúžia piny *PWR_SEL*. Osadením prepojky medzi piny 1 a 2 je zvolené externé napájanie, prepojka medzi 2 a 3 znamená napájanie z USB.

SD karta je pripojená cez konektor z radových kolíkov. Preto je možné ju fyzicky odpojiť a v prípade potreby pripojiť iné periférie. Osadením skratovacích prepojek na piny *QSPI* je SD karta pripojená.

V poslednom rade je nutné zvoliť, akým spôsobom budeme VWS programovať a ladiť. Máme na výber z dvoch možností, rozhranie BDM alebo JTAG. Osadením prepojky medzi piny *JTAG/BDM* zvolíme rozhranie BDM, neosadená prepojka znamená JTAG. To ale nestačí. V prípade BDM je ešte nutné osadiť prepojku medzi piny 1 a 2 *BDM_SEL*. Prepojka medzi pinamy 2 a 3 znamená naopak JTAG.



Obr. 6.6: Konfiguračné piny

6.1.3 Osadenie a oživenie

Výber všetkých súčiastok bol volený tak, aby ich bolo možné osadiť v amatérskych podmienkach. Pasívne súčiastky veľkosti SMD 805 alebo 1206, tranzistory SOT23. Elektrolytické kondenzátory sú dostupné aj v menších prevedeniach ako použité, ale na rozmery a funkčnosť VWS to zásadný vplyv nemá.

Súčiastky je doporučené osadzovať v poradí od komplexnejších po jednoduché. Je dobré začať slotom na SD kartu, procesorom a potom USB prevodníkom. Najkomplikovanejšie bolo osadenie práve slotu na SD kartu, pretože má veľmi zlý prístup v vývodom.

Puzdrá procesoru a prevodníku boli kreslené s presnosťou na stotinu milimetru, preto verne kopírujú vývody obvodov. Spájkovacie plôšky procesoru sú o niečo dlhšie ako je doporučené v dokumentácii pre bezproblémové osadenie.

Pri prvom pripojením napájacieho napätia by nemala byť osadená prepojka na pinoch *PWR_SEL*. Ľahko tak skontrolujeme či máme správne polaritu napätia a funkčnosť vstupného napäťového stabilizátora bez rizika poškodenia iných obvodov. Správnosť osadenia sa najlepšie overí ak máme k dispozícii zdroj s regulovateľným výstupným prúdom. Pri prvom pripojení by nemala mať doska dober väčší ako 70 mA. V opačnom prípade ide pravdepodobne o skrat. Po naprogramovaní procesoru sa odber odlišuje v závislosti od použitých periférii na čipe. V žiadnom prípade by ale celkový odber nemal prekročiť 400 mA bez SD karty.

6.1.4 DPS

VWS je realizovaný na obojstrannej doske plošného spoja o rozmeroch 120 x 90 mm. Kvôli zmenšeniu rozmerov sú použité súčiastky pre povrchovú montáž umiestnené na oboch stranách dosky. Všetky otvory sú prekovené a obe strany dosky sú ošetrené nespájkovacou maskou. Jej úlohou je pri spájkovaní zabrániť nežiaducemu skratovaniu signálov.

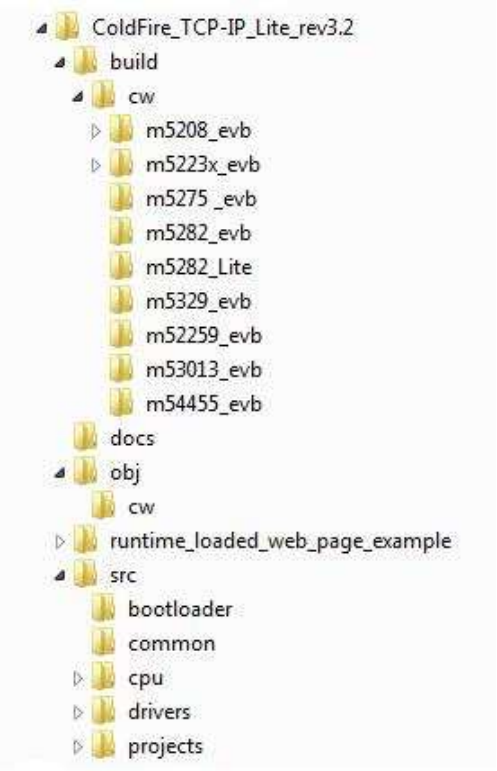
Pri návrhu bola snaha umiestniť napájací konektor, USB konektor, ethernetový konektor a slot pre SD kartu na jednu stranu dosky. BDM konektor je umiestený na opačnej strane dosky pre ľahký prístup programátora. Zoznam použitých súčiastok a puzdier je v prílohách. Pre návrh schémy zapojenia a DPS bol použitý program *Eagle 4.16*.

6.2 Softwarová časť

Softwarová časť sa venuje implementácii VWS v použitom procesore ColdFire. Celý VWS je postavený na NicheLite TCP/IP stacku upraveného pre použitie v procesoroch ColdFire. Preto ak nebude uvedené inak, nasledujúce podkapitoly sa týkajú práve spomínanej TCP/IP implementácii.

6.2.1 Adresárová štruktúra

Obrázok 6.7 zobrazuje adresárovú štruktúru NicheLite TCP/IP stacku verzie 3.2. Na rozdiel od predchádzajúcej verzie 2.2 je pridaná podpora pre CodeWarrior 7.2 a podpora pre nové procesory MCF52259, M53013 a MCF54455. Pre každý z procesorov existuje od Freescale príslušná vývojová doska (napr.: M52259EVB, M52259DEMO). Ich podpora je vo verzii 3.2 samozrejmá.



Obr. 6.7: Adresárová štruktúra verzie 3.2

Všetky CodeWarrior projekty sú uložené v adresári `ColdFire_Lite -> build -> cw`. Každý projekt je uložený v dvoch verziách, `6x.mpc` a `7x.mcp`. V závislosti na použitej verzii CodeWarrior stačí otvoriť jeden z projektov a je pripravený na použitie. Kód závislý na procesore je uložený v `ColdFire_Lite -> src -> cpu`. Všetky ostatné adresáre obsahujú kód, ktorý je nezávislý na procesore. Samotná implementácia TCP/IP stacku je uložená v `ColdFire_Lite -> src -> projects -> NicheLite`. Predpripravené projekty a súbory závislé na type procesoru sú

nakonfigurované pre použitie s Freescale vývojovými doskami. Predpokladajú preto zapojenie vývodov procesoru podľa dokumentácie a prítomnosť rôznych periférií na doske. Pre použitie s inými doskami, je nutné ich upraviť. Editujú sa súbory v adresári `cpu`.

6.2.2 Kompatibilita s CW 7.2

Hoci je ColdFire Lite TCP/IP stack 3.2 podľa oficiálnej dokumentácie od Freescale plne kompatibilný s CodeWarrior 7.2 nie je to celkom pravda. Bezplatná inštalácia CW 7.2 podporuje predávanie parametrov do funkcií len cez registre (`Target Settings -> Code Generation -> ColdFire Processor -> Parameter Passing = Register`). Nízko úrovňové rutiny TCP/IP stacku ale predpokladajú parametre na zásobníku. Zdrojový kód tak treba mierne upraviť. Týka sa to zdrojových súborov `mcf5xxx_lo.s`, `cksum.s` a `tk_util.s`. Napríklad:

```
mcf5xxx_wr_sr:
_mcf5xxx_wr_sr:
     move.l  4(SP),D0         /* Načítanie premennej zo zásobníku */
     move.w  D0,SR
     rts
```

Červený riadok je v upravenej verzii zmazaný a parameter je vložený do registru D0 pred volaním rutiny. Upravené zdrojové súbory sú súčasťou príloh.

6.2.3 RTOS

TCP/IP stack vyžaduje, aby sa mohli dve a viac udalostí vyskytnúť naraz, preto potrebuje implementovať nejakú správu procesov. Súčasťou TCP/IP implementácie je tak aj jednoduchý real–time operačný systém (RTOS).

Použitie RTOS ale nie je nevyhnutné. Druhou možnosťou je tzv. *superloop* režim, v ktorom predstavuje každá úloha funkčné volanie. Všetky funkcie zdieľajú spoločný zásobník. S pohľadu využitia RAM je *superloop* režim efektívnejší. Nevýhodou je, že je statický. Úlohy nemôžu byť pridané alebo odobrané za behu aplikácie. Tak isto nie je možné úlohu uspať. Každá funkcia sa musí po ukončení vrátiť do hlavnej slučky.

S RTOS môžu byť úlohy vytvárané a rušené za behu programu. Každá nová úloha má vlastný zásobník vytvorený na hromade. Veľkosť zásobníku je statická a určená v dobe kompilácie. Po zrušení úlohy je alokovaná pamäť vrátená naspäť na hromadu. U dynamických aplikácií tak môže byť tento režim pamäťovo efektívnejší. Veľkosť zásobníku musí byť dostatočná pre úlohu samotnú plus prerušenia definované v systéme.

Každá úloha v RTOS má vlastný kontrolný blok TCB (Task Control Block) uložený v pamäti RAM. Kontrolné bloky sú spojené do lineárneho zoznamu. Je implementovaný jednoduchý plánovač

ktorý nepodporuje priority. Plánovač preto len postupne prechádza zoznamom od začiatku po koniec a vykonáva úlohy kontrolného bloku TCB na ktorý práve ukazuje. Úloha je vykonaná, len ak je na to pripravená. Pretože RTOS je nepreemptivny, prepnutie úloh nastane až keď sa aktuálna úloha sama vzdá procesora. Na to má dve možnosti, zavolaním funkcie `tk_block()` alebo `tk_sleep()`. Štruktúra TCB ako je definovaná v súbore `task.h`:

```
struct task {
     struct task *  tk_next;        /* ukazovateľ na ďalší TCB v zozname */
     stack_t *   tk_fp;             /* typ dát na zásobníku úlohy */
     char *      tk_name;           /* meno úlohy */
     int         tk_flags;          /* 1 ak je plánovaná, inak 0 */
     ulong       tk_count;          /* počet spustení */
     stack_t *   tk_guard;          /* ukazovateľ na „guardword“ */
     unsigned    tk_size;           /* veľkosť zásobníku */
     stack_t *   tk_stack;          /* bázová adresa zásobníku úlohy */
     void *      tk_event;          /* udalosť na ktorú sa čaká */
     ulong       tk_waketick;       /* počet cticks do zobudenia úlohy */
};
```

Guardword je špeciálne 32 bitové číslo, ktorým je vyplnený celý zásobník úlohy po jeho vytvorení. Normálnou činnosťou ho úloha na zásobníku prepisuje. Ak sa stane, že zásobník nie je dostatočne veľký úloha prepíše *guardword* na poslednej adrese (`tk_stack + STACK_SIZE`) a vygeneruje sa výnimka. Veľkosť zásobníku treba voliť s ohľadom na požiadavky aplikácie. V prípade prerušenia je celý kontext uložený na zásobník práve vykonávanej úlohy.

Úloha môže byť uspaná na základe času alebo čakania na udalosť. V prípade, že úloha je uspaná na základe času, zmení svoj stav po uplynutí tejto doby. Doba sa meria v relatívnych jednotkách *cticks*.

Časovanie je kritické na sledovanie udalostí a dodržiavanie oneskorení. Pretože RTOS je nepreemptivny, prepnutie úloh je asynchrónne, bez systémového časovača. Periodický časovač inkrementuje premennú *cticks* definovanú užívateľom v súbore `main.c`. Časovanie celého TCP/IP stacku sa odkazuje na túto premennú. *Cticks* musí byť typu `unsigned long` a nesmie dôjsť k jej pretečeniu. Inkrementovaná musí byť každých 5 ms a konkrétna implementácia je ponechaná na programátora. Funkcia sleep tak berie ako parameter počet cticks. Časť implementácie `tk_sleep`:

```
tk_cur -> tk_waketick = cticks + 10;     /* tk_sleep(10) */
tk_block();
tk_wakeups++;
tk_cur -> tk_count++;                    /* dalšia úloha */
```

Vytvorenie a pridanie novej úlohy do operačného systému je jednoduché. Sú definované tri makrá, ktoré zapuzdrujú všetky potrebné funkčné volania. Makro `TK_OBJECT()` vytvorí TCB štruktúru, `TK_ENTRY()` definuje *callback* funkciu a makro `TK_NEWTASK()` pridá kontrolný blok do zoznamu. `TK_NEWTASK` zapuzdruje volanie funkcie `tk_new()`. Tá očakáva ako parameter ukazovateľ na *callback* funkciu, meno úlohy a ukazovateľ na kontrolný blok. Nový TCB bude zaradený do zoznamu za túto položku. Toto je jediná možnosť ako aspoň malým spôsobom ovplyvňovať prioritu úloh. Programátor tak môže vložiť blok na akékoľvek miesto do zoznamu. Ak nie je definovaný predchádzajúci blok, je nová úloha naplánovaná za aktuálne bežiacu. API RTOS tvorí okrem `tk_new()` ďalších sedem funkcií.

- `tk_block(void)` – funkcia najprv skontroluje *guardword* na poslednej adrese zásobníku. Ak nie je poškodené, vyhľadá v zozname najbližšiu úlohu ktorá je pripravená a vynúti prepnutie úloh.
- `tk_exit(void)` – po dobehnutí, je úloha odstránená so zoznamu.
- `tk_kill(task *tk_die)` – nečaká sa na dobehnutie úlohy.
- `tk_wake(task *tk)` – úloha je označená ako pripravená. Spustí sa až sa k nej dostane plánovač.
- `tk_sleep(ulong cticks)` – uspí úlohu na určitý čas. Súčasťou volania `tk_sleep()` je `tk_block()`.
- `tk_ev_block(void *ev)` – uspí úlohu do výskytu určitej udalosti. Udalosť je 32 bitové číslo.
- `tk_ev_wake(void *ev)` – prejde celým zoznamom TCB a porovná parameter `ev` s hodnotou udalosti na ktorú úloha čaká. Ak nájde zhodu, úloha je pripravená sa spustenie.

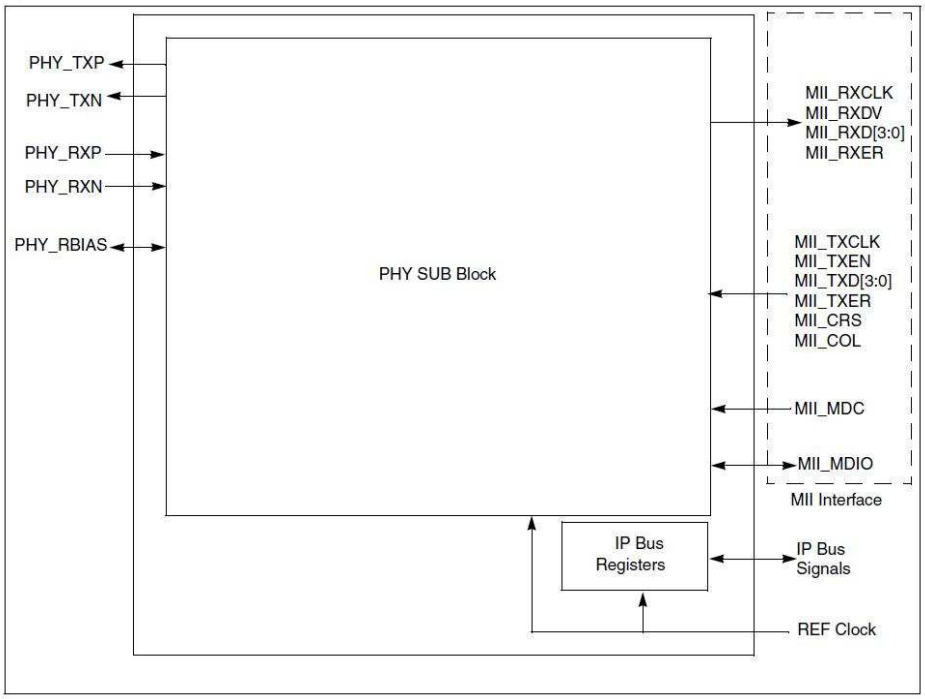
RTOS a stack vyžadujú aby sa rutina `timer_isr()` volala periodicky a každých 5 ms inkrementovala premennú *cticks*. Implementáciu je ponechaná na programátora. U procesoru MCF52233 môžeme využiť jeden z dvoch časovačov PIT (Programmable Interrupt Timer). PIT je schopný generovať prerušenia v presných časových intervaloch bez zásahu CPU. Keď časovač dosiahne hodnotu $0x0000$, hodnota uložená v modulo registri (PMR) sa načíta do registru časovača, vygeneruje sa prerušenie a časovač začne nový cyklus. Perióda sa vypočíta podľa vzorca:

$$T = PRE \times (PMR + 1) \times (f_{sys} \div 2)$$

PRE je hodnota 4 – bitovej preddeličky. f_{sys} je rýchlosť systémovej zbernice, typicky 60 MHz. Všetky periférie na čipe sú taktované polovičnou rýchlosťou – 30 MHz. Pre periódu 1 ms je preddelička vypnutá a hodnota modulo registru $PMR = 0x7530$.

6.2.4 MAC a PHY

Štruktúra stacku presne kopíruje vrstvový model TCP/IP. Funkcie sú rozdelené do zdrojových súborov podľa toho, na akej vrstve modelu operujú. Najnižšie je fyzická vrstva. Štandard IEEE 802.3 sa skladá z dvoch častí, vrstvy riadiacej prístup na médium – MAC a fyzickej vrstvy – PHY. Existuje veľa typov fyzických vrstiev, no ColdFire podporuje dve základné 10 Mbps (IEEE 802.3a–t) a 100 Mbps (IEEE 802.3u). Napriek tomu, že sú fyzické vrstvy rozdielne, sú implementovaná ako celok. Každá fyzická nie je nič iné ako elektrický modulátor. PHY komunikuje s MAC cez rozhranie Media Independent Interface (MII). Súčasťou MII je sériové rozhranie cez ktoré sú dostupné konfiguračné registre PHY. U procesorov ColdFire je to Media Management Interface (MMI).



Obr. 6.8: Blokový diagram EPHY (Ethernet PHY) procesorov ColdFire. Prevzaté z [9]

Čiastočne je možné EPHY konfigurovať pomocou troch kontrolných registrov ktoré sú mapované do RAM pamäte. Nie je tak ale možné nastaviť všetko. Interné registre sú dostupné len cez MII rozhranie. Tak nastavuje parametre aj ColdFire Lite stack. Rutiny na zápis a čítanie interných MII registrov sú uložené v `mii.c`. Aplikačné rozhranie fyzickej vrstvy tak tvoria len dve funkcie:

```
int fec_mii_write (int phy_addr, int reg_addr, int data);
int fec_mii_read (int phy_addr, int reg_addr, uint16* data);
```

EPHY môže byť v jednom z piatich stavov, inicializácia, 10BASE–T, 100BASE–TX, auto–negotiate a stav nízkej spotreby. Východzie nastavenie stacku je auto–negotiate, kedy si komunikačný partneri navzájom dohodnú rýchlosť na ktorej budú komunikovať.

Adresa	Použitie	Právo
0x00	Riadiaci register 1	R/W
0x01	Stavový register 1	R
0x02	PHY Identifikačný register 1	R
0x03	PHY Identifikačný register 2	R
0x04	Auto-negotiation	R/W
0x05	Auto-negotiation	R
0x06	Auto-negotiation	R
0x07	Auto-negotiation	R/W
0x10	Register Prerušení	R/W
0x11	Stavový register 2	R
0x12	Riadiaci register 2	R/W

Tab. 6.9: Interné registre EPHY dostupné cez MII rozhranie

Nad fyzickou vrstvou sa nachádza linková. Tú predstavuje MAC, u procesorov ColdFire nazvaná Fast Ethernet Controller (FEC). FEC ovláda signály MII rozhrania v prípade, že je PHY v režime 100BASE–TX. V režime 10BASE–T je rozhranie s PHY tvorené siedmim signálmi EMAC. FEC pracuje v kombinácii s integrovanou PHY. Je možné pripojiť externú fyzickú vrstvu pretože signály MII sú vyvedené na piny.

Základná funkcia FEC je kontrola toku dát. Môže pracovať v dvoch režimoch, *full duplex* alebo *half duplex*. V plnom duplexe je priepustnosť dát 200 Mbps, v polovičnom o polovicu menej. FEC je navrhnutý tak, aby pracoval s čo najmenším počtom zásahov riadiaceho softwaru. Po umiestnení dát do výstupnej FIFO fronty, je automaticky spočítaný kontrolný súčet a dáta sú odoslané. Ak je detekovaná kolízia, FEC sa pokúsi dáta znova odoslať až do vyčerpania pokusov. Pre príjem rámcov zo siete máme rozšírené možnosti. Okrem automatickej kontroly CRC a veľkosti rámca umožňuje FEC filtrovanie paketov na základe cieľovej adresy ethernetového rámca. Sú rozlišované tri druhy adries, unicastové, multicastové a broadcastové. Broadcastové adresy sú filtrované priamo FEC, unicastové a multicastové adresy musí filtrovať procesor. Procesor filtruje adresy hashovaciou funkciou s úspešnosťou menšou ako 100%. Pakety je preto treba ešte kontrolovať na vyšších vrstvách. ColdFire TCP/IP stack nemá plne implementovanú podporu filtrovania multicastových adries.

Aplikačné dáta medzi RAM a FEC prenáša DMA radič. Dáta sú uložené v jednom alebo viacerých bufferoch. Ku každému bufferu je asociovaný tzv. *buffer descriptor* (BD), ktorý obsahuje ukazovateľ na počiatok bufferu, dĺžku a riadiace a stavové bity. Správa bufferov a BD je plne v moci riadiaceho softwaru. Buffre a BD pre príjem a vysielanie sú separátne. Obe skupiny BD sú usporiadané do kruhovej štruktúry. Zápis do riadiaceho registru FEC odštartuje DMA prenos. Radič

DMA postupne prechádza zoznamom BD a kontroluje, ktoré buffre sú pripravené na prenos. Dáta pre jeden paket môžu byť rozložené do viacerých častí. Počiatočné adresy BD sú uložené v registroch FEC.

Obr. 6.10: Buffer Descriptor. Prevzaté z [9]

Prakticky je *Data Pointer* implementovaný len ako 8 – bitový ukazovateľ, pretože nie je potrebné adresovať tak veľkú pamäť.

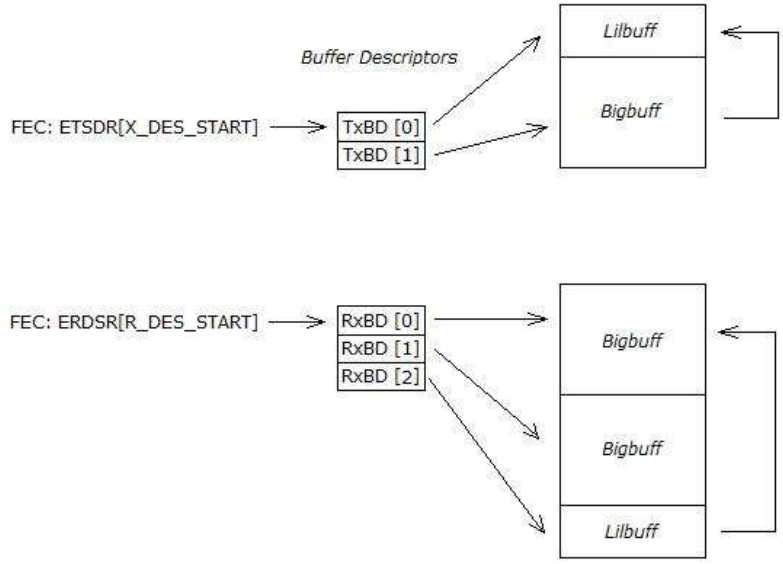
```
typedef struct BufferDescriptor {
     volatile unshort  bd_cstatus;
     volatile unshort  bd_length;
     volatile u_char * bd_addr;
} BD;
```

TCP/IP stack implementuje dva typy bufferov, veľký – *bigbuff* a malý – *lilbuff*. Veľký buffer by mal byť veľký ako najväčší možný rámec prijatý FEC (rovnaká hodnota je uložená aj v riadiacom registri EMBR, ktorý hardwarovo obmedzuje maximálnu veľkosť príjemného rámca) a používa sa pre každý prijímaný rámec a rámce zasielané na sieť pre ktoré je *lilbuff* malý. Malý buffer je hlavne pre ARP dotazy, ICMP a ACK pakety. Minimálna veľkosť by mala byť väčšia ako ACK pakety – 60 bajtov. To aký buffer bude alokovaný, je definované parametrom `len` vo volaní funckie PACKET `pk_alloc (unsigned len)`.

Pointer na buffer je zabalený spolu s informáciou o celkovej veľkosti, veľkosti pre dáta protokolu (TCP/UDP) , IP adresou a typom protokolu do štruktúry `PACKET`. Alokáciu a uvoľnenie pamäte pre paket nemá na starosti spáva pamäte RTOS ale fronta do ktorej sú pri inicializácii priradené. Počet paketov je rovný počtu *lilbuff* a *bigbuff* dohromady. Pamäť je alokovaná z hromady a nie je nikdy uvoľnená. Volanie `pk_alloc()` vráti prvý paket vo fronte s bufferom alokovaným na správnu veľkosť. Veľkosť už nie je možné behom programu meniť.

Počet bufferov je akýmsi kompromisom medzi výkonom a spotrebovanou RAM a musí byť volený s ohľadom na prostredie v akom bude VWS pracovať. Na sieť s vysokým tokom dát je potrebných bufferov viac. Pri vypnutom filtrovaní rámcov je potrebné mať bufferov čo najviac,

pretože Broadcastové pakety môžu aplikáciu úplne zahltiť. Každú zmenu počtu bufferov je dobré otestovať.



Obr. 6.11: Organizácia BD a bufferov

6.2.5 Implementácia TCP/IP

RTOS je len podporným prostriedkom pre implementáciu rodiny protokol TCP/IP. Konkrétne nájdeme v stacku podporu pre tieto protokoly:

- Internet Protocol (IP) – RFC791
- Internet Control Message Protocol (ICMP) – RFC792
- Transmission Control Protocol (TCP) – RFC793
- User Datagram Protocol (UDP) – RFC768
- Address Resolution Protocol (ARP) – RFC826
- Domain Name System (DNS) – RFC1035
- Dynamic Host Configuration Protocol (DHCP) – RFC2131

Každý paket prijatý zo siete je uložený do bufferu a vložený do fronty. O fyzický príjem paketov sa stará FEC a úspešne prijatý paket oznámi vyvolaním prerušenia. Obslužná rutina prerušenia umiestni prijatý paket do bufferu a zavolá funkciu `input_ippkt()`. Tá vyplní potrebné údaje v štruktúre `PACKET` (dôležitý je hlavne typ paketu) a `PACKET` vloží do fronty. ColdFire Stack používa maskovanie všetkých prerušení namiesto semafórov na ochranu kritických sekcií akou je

napríklad vloženie paketu do fronty. Operácie nad frontou sú tak atomické. Makro `SignalPktDemux()` odštartuje spracovanie prijatého paketu. Nie je to nič iné ako volanie `tk_wake()` na úlohu *netmain* – hlavnú úlohu TCP/IP stacku. Hlavná úloha periodicky kontroluje veľkosť fronty. Ak je fronta neprázdna zavolá funkciu `pktdemux()`. Tá vyberá pakety z fronty a na základe ich typu ich posiela na ďalšie spracovanie. Typ paketu je extrahovaný z hlavičky ethernetového rámca. Na tejto úrovni rozlišujeme iba dva typy, IP a ARP pakety. IP pakety sú posielané na spracovanie funkciou `ip_rcv()`. Tá filtruje pakety na úrovni IP vrstvy. Najprv skontroluje typ IP paketu (IPv6 pakety nie sú podporované). Ak sedí kontrolný súčet a cieľová IP adresa, je paket zaslaný na spracovanie vyššou vrstvou podľa typu použitého protokolu (TCP, UDP, ICMP).

ARP pakety spracováva funkcia `arprcv()`. Tá udržiava a aktualizuje ARP tabuľku a rozlišuje či ide o ARP dotaz alebo odpoveď. ARP tabuľka má nasledujúcu štruktúru:

```
struct arptabent  {
     unsigned long  t_pro_addr;       /* IP  adresa */
     unsigned char  t_phy_addr[6];    /* MAC adresa */
     struct net *net;                 /* Informácie o rozhraní */
     PACKET   pending;                /* pakety čakajúce na preklad */
     u_long   createtime;             /* vytvorenie záznamu (cticks) */
     u_long   lasttime;               /* posledné použitie záznamu */
     unshort  flags;
};
```

Veľkosť ARP tabuľky určuje programátor a záleží od prostredia v ktorom sa VWS bude používať. U LAN s veľkým počtom komunikujúcich ušetrí veľká ARP tabuľka množstvo dotazov. Pri PPP spojeniach alebo na sieťach s minimálnym počtom zariadení nám postačí malý počet záznamov.

Na internetovej vrstve potrebujeme zabezpečiť smerovanie. Pretože VWS má iba jedno sieťové rozhranie, nie je potrebné si udržiavať žiadnu smerovaciu tabuľku. Implementovaný smerovací protokol je triviálny. Vymaskuje cieľovú IP adresu maskou podsiete a zistí, či sa cieľ nachádza na rovnakej podsieti ako VWS. Ak áno, ako *next hop* bude označená IP adresa cieľového systému. Ak je cieľová IP adresa všesmerová, ako *next hop* je označená broadcastova doména podsiete. To že ide o všesmerové vysielanie je poznačené aj do hlavičky ethernetového rámcu. Posledným prípadom je cieľová IP adresa ležiaca mimo našu lokálnu sieť. Vtedy je do *next hop* skopírovaná IP adresa *default gateway* (predvolený smerovač našej siete). Funkcia `ip_write()` potom vyplní všetky položky IP paketu (vezia, flags, dĺžka, kontrolný súčet, tos, ...) a pošle IP paket funkcii `send_via_arp()`. Ako už naznačuje názov funkcie, pre každý odosielaný paket je prehľadávaná ARP tabuľka. Vo väčšine prípadov nie je prístup do tabuľky vôbec potrebný. ARP cache si udržuje posledný použitý záznam.

Jednoduchou úpravou kódu je možné cache rozšíriť v prípade, že sa na server pripája mnoho klientov ktorých potrebujeme obsluhovať súčasne.

Rozhranie medzi aplikáciou a IP protokolom tvorí transportná vrstva. ColdFire stack implementuje dva transportné protokoly, TCP a UDP. Aby sa aplikačné rozhranie čo najviac podobalo na zaužívaný spôsob programovania sieťových aplikácií je implementované pomocou mini socketov. Mini sockety vychádzajú z BSD socketov až na pár odlišností. Sled BSD volaní `bind() -> listen() -> accept()` je u mini socketov nahradený volaním jeden funkcie `m_listen()`. Namiesto blokujúceho volania funkcie `select()` je použitá *callback* funkcia. API tvorí sedem funkcií:

- `m_socket()` – alokuje pamäť pre nový socket a vráti ukazovateľ. Nový socket je blokovací a po vytvorení je vložený do fronty.
- `m_connect()` – snaží sa pripojiť na server s IP adresou a portom uvedenými v štruktúre `sockaddr_in`. Ak je socket nastavený ako neblokujúci, funkcia okamžite skončí a vráti `EINPROGRESS`. Ak je socket blokujúci a spojenie sa nenadviaže do času definovaného užívateľom, spojenie zlyhá.
- `m_listen()` – funkcia vytvorí a vráti nový socket, ktorý čaká na zadanom porte na príchodzie spojenia. Ako parameter očakáva štruktúru `sockaddr_in` a ukazovateľ na *callback* funkciu. Je možné nastaviť konkrétnu IP adresu klienta ktorá bude akceptovaná. Klienti s inými adresami budú ignorovaní. Ak nie je IP adresa definovaná, sú akceptované všetky spojenia. Volanie `m_listen()` skončí okamžite a každé nové spojenie spôsobí volanie *callback* funkcie. Formát funkcie je predefinovaný.
  `int server_callback (int code, M_SOCK socket, void * data).`
- `m_send()` – odošle dáta do socketu.
- `m_recv()` – prijme dáta zo socketu.
- `m_ioctl()` – Nastavuje parametre socketu. Chovanie funkcií m_send() a m_recv() ide ovplyvniť rôznym nastavením socketu.
- `m_close()` – ukončí všetky spojenia a zavrie socket.

Informácie o každom TCP spojení sú uložené v štruktúre `tcpcb` – TCP control block. RCF793 doporučuje, aby malo každé spojenie vlastný kontrolný blok (existujú riešenia s centrálnym zdieľaným). ColdFire stack udržiava pre každé spojenie jednu `tcpcp` štruktúru. Inými slovami, každé volanie `m_listen()` a `m_connect()` spôsobí vytvorenie TCP kontrolného bloku a jeho priradenie k socketu. Blok aj socket sú na seba navzájom odkazujú. Informácie v kontrolnom bloku sú vyplnené až po nadviazaní spojenia a prijatí paketu. `tcpcb` si udržiava informácie o sekvenčných číslach, paketoch čakajúcich na potvrdenie, MTU, veľkosti vysielacieho okna alebo timeoutoch.

Súčasťou transportnej vrstvy sú tri časovače – TCPT REXMT, TCPT PERSIST a TCPT KEEP. REXMT slúži na vynútenie opakovaného prenosu paketu. Časovač je vynulovaný zakaždým keď je odoslaný paket a keď je prijatý príslušný ACK paket. V prípade pretečenia časovača, je paket odoslaný znovu. PERSIST udržiava správnu veľkosť vysielacieho okna. Informácia o aktuálnej veľkosti okna je zaslaná pri pretečení časovača. Posledný časovač KEEP udržuje spojenie. Ak je spojenie nečinné po dobu `TCPT_KEEP_IDLE`, odošle VWS paket so sekvenčným číslom mimo poradia a vynúti si tak odpoveď od komunikačného partnera v podobe ACK paketu. Aby sa udržiavalo spojenie, musí byť povolená voľba `keepalive` v nastaveniach socketu. Časovače sú implementované v `tcp_timr.c`. Programátor má prístup k nastaveniu periód časovačov a API funkciám socketov. Všetky ostatné dátové štruktúry a funkcie by sa nemali volať explicitne a modifikovať.

Pretože pri UDP nepotrebujeme udržiavať informácie o stave spojenia alebo čakať na potvrdenie o úspešnom prijatí paketu, je práca o niečo jednoduchšia. UDP hlavička má len štyri položky a informácií o spojení je tiež nie je veľa.

```
struct udp_header {
        unshort  ud_srcp;          /* zdrojový port */
        unshort  ud_dstp;          /* cieľový port */
        unshort  ud_len;           /* dĺžka paketu */
        unshort  ud_cksum;         /* UDP checksum */
};
```

Pre každé spojenie (číslo portu) je v pamäti uložená štruktúra popisujúca toto spojenie. Obsahuje zdrojové a cieľové IP adresy, čísla portov a dva odkazy na dve *callback* funkcie. Funkcia `udp_open()` vytvorí štruktúru UDP spojenia a vloží ju do zoznamu. Funkcia `udpdemux()` tak po prijatí UDP paketu z nižšej vrstvy prejde celý zoznam a porovnáva čísla portov. Ak nájde zhodu, zavolá *callback* funkciu z príslušnej štruktúry a predá jej aplikačné dáta z paketu. UDP pakety odosiela funkcia `udp_send()`. Ako parameter prijíma číslo zdrojového a cieľového portu a `PACKET` alokovaný na správnu veľkosť. Funkcia len pripojí UDP hlavičku, spočíta kontrolný súčet a pošle paket IP vrstve.

6.2.6 DHCP klient

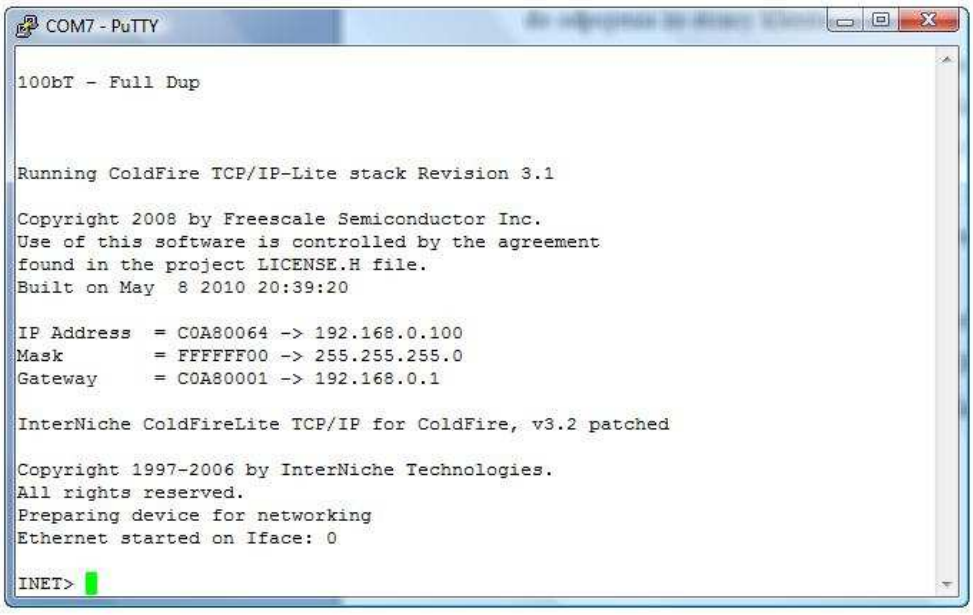
RCF2132 definuje množinu informácií ktoré môže DHCP klient požadovať od serveru. Komunikácia funguje obojsmerne, a klient môže zasielať stavové informácie DHCP serveru. Množina príkazov ktoré DHCP klient podporuje sú uložené v `dhcpclnt.h`. Iba príkazy `DHOP_NAME` a `DHOP_DOMAIN` posielajú informácie v smere od klienta na server. Je to názov a doména hostiteľského systému. DHCP server ich preposiela na DNS server.

IP adresa sa dá nastaviť manuálne, alebo automaticky získať od DHCP serveru. V oboch prípadoch je však vložená do rovnakej štruktúry – `netstatic`. Odtiaľ si ju vyberie IP vrstva a uloží do svojich interných záznamov. Pred prvým dotazom na DHCP server musí byť preto `netstatic` prázdna, inak sa klient pokúsi predĺžiť pôžičku uloženej IP adresu namiesto získania novej. Po získaní novej IP musíme periodicky žiadať o predĺženie jej pôžičky. Na to slúži funkcia `dhc_second()`, ktorá je volaná každú sekundu. DHCP klienta aktivujeme definíciou v súbore `processor.h`

```
#define DHCP      0     /* 0 = DHCP OFF, 1 = DHCP ON */
```

Pri vypnutom DHCP klientovi, je použitá IP adresa uložená do `netstatic` v súbore `main.c`.

```
netstatic[0].n_ipaddr = (0xC0A80064);     /* 0xC0A80064 = 192.168.0.100 */
netstatic[0].n_defgw  = (0xC0A80001);     /* 0xC0A80101 = 192.168.0.1   */
netstatic[0].snmask   = (0xffffff00);     /* 0xffffff00 = 255.255.255.0 */
```



Obr. 6.12: IP konfigurácia

6.2.7 DNS klient

Pre správnu funkciu DNS klienta je potrebné mať nastavené IP adresy DNS serverov. Pri povolenom DHCP protokole sú vyplnené automaticky, inak ich musíme zadať ručne do poľa `dns_servers[]`. Počet serverov na ktoré budeme smerovať DNS dotazy je ľubovoľný. Nevyužité položky musia byť vyplnené nulami. DNS klient si udržiava lineárny zoznam platných doménových mien a príslušných IP adries – DNS cache. Každý nový dotaz a odpoveď sú automaticky vložené do zoznamu. Aby bol zoznam aktuálny, je potrebné periodicky volať funkciu `dns_check()`. Tá skontroluje platnosť všetkých položiek a pokúsi sa obnoviť tie, ktorým platnosť vypršala.

Pri prvom preklade je nutné volať funkciu `dns_query()` resp. `dns_query_type()`. Obe vytvoria novú položku v DNS cache. Pre ďalšie dotazy stačí prehľadávať DNS cache funkciou `dns_lookup()`. DNS klienta aktivujeme definíciou v súbore `ipport.h`.

```
#define DNS_CLIENT      1     /* DNS Klient ON  */
//undef DNS_CLIENT            /* DNS Klient OFF */
```

6.2.8 Ovládač SD karty

V pôvodom ColdFire TCP/IP stacku podpora pre SD kartu chýba a nie je prítomný ani ovládač rozhrania QSPI cez ktoré je karta pripojená. Bolo ich preto potrebné doprogramovať a vložiť medzi zdrojové súbory. Rozhranie SPI má u procesorov ColdFire označenie QSPI – *Queued SPI*, alebo SPI z frontou. Takto umožňuje naskladať do fronty až 16 operácií (dáta + príkaz) a vykonať ich naraz bez zásahu programátora alebo CPU. QSPI môže pracovať len v master móde.

Fronta má v RAM pamäti vyhradených 80 bajtov rozdelených do troch častí. 32 bajtov pre príjem, 32 pre vysielanie a 16 bajtov na príkazy. Z toho vyplýva, že sa môžu prenášať slová až dva bajty dlhé a prenos každej položky riady jeden ovládací bajt. Položky riadiacej RAM sú mapované na 16 registrov QCR0 – QCR15. Tie majú v skutočnosti dva bajty každý, no prístupná je len vrchná polovica.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CONT	BITSE	DT	DSCK	QSPI_CS				0	0	0	0	0	0	0	0

Obr. 6.13: Riadiaci register OCR. Prevzaté z [9]

Každému prenosu je možné individuálne nastaviť oneskorenie, počet prenášaných bitov, závislosť signálu CS na hodinovom signály CLK alebo chovanie CS po skončení prenosu. Je možné ovládať až 16 externých signálov CS. Použité puzdro ma však dostupný len jeden – CS0.

Práca s QSPI je jednoduchá. Stačí nakopírovať dáta do vysielacej fronty, nastaviť ukazovateľ na poslednú položku a v riadiacom registri rozhrania povoliť štartovací bit. Koľko položiek sme umiestnili do vysielacej fronty, toľko ich bolo prijatých. Ak chceme dáta prijímať, proces je rovnaký. Musíme sa ale uistiť, že dáta ktoré budú vysielané nebudú chybne interpretované. U SD karty je pri prijímaní vysielaná hodnota `0xFF`. Pre prácu s QSPI nám stačia tri funkcie, `write_to_qspi_ram()`, `read_from_qspi_ram()` a `start_qspi_transfer()`.

SD protokol je jednoduchý protokol typu dotaz odpoveď. Všetky dotazy iniciuje master. SD karta odpovedá špeciálnym rámcom za ktorým môžu podľa typu dotazu nasledovať dáta. SD dotazy majú tvar „CMDXX“ alebo „ACMDXX“, kde „CMD“ odkazuje na obecný dotaz, „ACMD“ aplikačne špecifický dotaz a „XX“ je číslo dotazu. Dotazy sú SD karte zasielané v 6 – bajtovej štruktúre v tvare číslo príkazu, argument, CRC.

Obr. 6.14: Štruktúra SD dorazu

Na každý dotaz odpovedá karta špeciálnym odpovedným rámcom, podľa typu dotazu. V SPI móde je to jeden z troch možných: R1, R2 alebo R3. Podľa hodnoty odpovede vieme potvrdiť správnosť interpretácie dotazu kartou prípadne identifikovať chybu. Najčastejšia je odpoveď typu R1. Pri prenose dát je formát dotazu rovnaký. V argumente je adresa bloku ktorý chceme čítať/zapisovať, CRC bajt je posledný. Na príkazy blokového čítania/zápisu odpovedá karta rámcom typu R1. Za ním v prípade čítania nasleduje dátový paket do veľkosti 2kB. Hlavička paketu sa líši podľa toho či ide o zápis jedného alebo viacerých blokov. Za dátovým paketom už nenasledujú žiadne dáta. Pri zápise na SD kartu posiela dátový paket procesor. O tom, či boli dáta úspešne prijaté nás informuje karta zaslaním odpovede *Data Response* (DR).

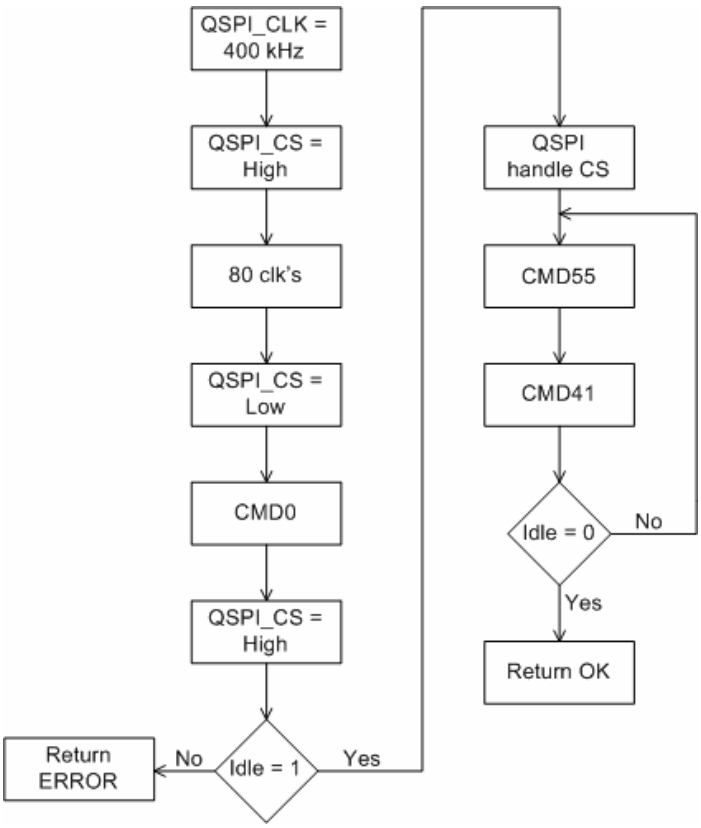
Hoci je možné komunikáciu s SD kartou zabezpečiť kontrolným súčtom CRC, v SPI móde je táto funkcia predvolene vypnutá. CRC sa kontroluje len pri prvom príkaze CMD0, kedy ešte karta nie je v SPI režime. Kontrolný súčet je pre CMD0 uložený v zdrojovom kóde ako konštanta.

Aby pracovala SD karta v SPI režime, vyžaduje špecifickú inicializačnú sekvenciu. Graf na obrázku 6.15 zobrazuje inicializáciu ako je implementovaná vo funkcii `sd_init()`. Kmitočet hodín je pri inicializácii z dôvodu zachovania kompatibility s MMC kartami znížený na 400 kHz. Po inicializácii je kmitočet zvýšený na 15 Mhz. QSPI rozhranie dokáže automaticky ovládať výberový signál CS. Na začiatku inicializácie je chovanie signálu CS odlišné ako pri normálnej činnosti, preto je ovládaný programovo. Sled príkazov CMD55 a CMD41 tvorí dohromady príkaz ACMD41 a odštartuje samotnú inicializáciu karty. Inicializácia môže trvať stovky milisekúnd a bez nej nie je možné kartu použiť. Cyklus preto končí až pri úspešnom pokuse.

Funkcia `sd_send_cmd()` zašle príkaz karte. Ako parameter očakáva číslo dotazu, typ očakávanej odpovede a počet cyklov ktoré má na odpoveď čakať. Funkcia tak sama čaká na odpoveď a skontroluje ju na prípadné chyby. V prípade chyby alebo vypršania časového limitu vráti chybu. Argument príkazu je vložený do globálnej premennej typu `Targument`.

```
typedef union {
     uint8  bytes[4];
     uint32 lword;
} Targument;
```

Pre R/W operácie sú implementované len dve funkcie blokové čítania/zápisu – `sd_read_block()` a `sd_write_block()`. Veľkosť bloku je možné zmeniť príkazom CMD16. Prednastavená veľkosť je 512 B.



Obr. 6.15: Implementovaná inicializácia SD karty

Popisovaný kód bol testovaný na micro SD karte *Kingston* 2 GB. SPI mód na SD kartách nie je štandardizovaný, preto je niekedy funkčnosť závislá na type použitej karty.

6.3 Freescale webový server

Freescale pridáva k existujúcemu TCP/IP stacku implementáciu webového servera. Web server je kompatibilný s HTTP 1.0, podporuje trvalé spojenia (HTTP je bezstavový protokol a pri každom dotaze sa inak musí vytvoriť nové spojenie) a dokáže obslúžiť niekoľko klientov súčasne. Web server má vlastný statický súborový systém a na uloženie dát používa internú flash pamäť mikrokontroléru.

Implementovaný súborový systém nie je nič iné, ako statické dáta prevedené do formy poľa jazyka C. Program *emg_static_fss.exe* berie ako argument zoznam súborov ktoré chceme aby súborový systém obsahoval. Výstup je zdrojový súbor jazyka C v ktorom sú prevedené všetky požadované súbory. Ako príklad je uvedená štruktúra súborového systému jeden z ukážkových aplikácií.

```
const unsigned char index_htm[] = {
      0x20,0x31,0x65,0x6D,0x20,0x31,0x65,0x6D,0x3B,0x0D,
      ...
const unsigned char style_css[] = {
      0x48,0x54,0x54,0x50,0x2F,0x31,0x2E,0x31,0x20,0x32,
      ...
const unsigned char freescale_logo_jpg[] = {
      0xFF,0xDB,0x00,0x43,0x00,0x0A,0x07,0x07,0x08,0x07,
      ...
const char *emg_static_ffs_filenames[] = {
      "index.htm", "style.css", "freescale_logo.jpg"
};
const unsigned char *emg_static_ffs_ptrs[] = {
      index_htm, style_css, freescale_logo_jpg
};
const unsigned short emg_static_ffs_len[] = {
      2108, 672, 5785
};
const unsigned long emg_static_ffs_type[] = {
      0x68746d6c, 0x68746d6c, 0x6a706567
};
const unsigned char emg_static_ffs_nof = 3;
```

Počet uložených súborov udáva premenná `emg_static_ffs_nof`. Pri vyhľadávaní súboru algoritmus prechádza pole s názvami súborov `emg_static_ffs_filenames` a hľadá zhodu. Ak nájde zhodu, použije rovnaký index na pole ukazovateľov `emg_static_ffs_ptrs`. Algoritmus tak pozná počiatočnú adresu na ktorej je súbor uložený. Aký veľký blok pamäte má prečítať vie z poľa

`emg_static_ffs_len`. Výhoda použitého statické súborového systému je jeho jednoduchosť a prehľadná implementácia. Nevýhoda je, že ho pri každej zmene musíme znovu kompilovať a nahrávať nový obraz do procesoru.

Web server podporuje len tri príkazy, GET, POST a špeciálny príkaz EMG pre nahrávanie dát na server. Server očakáva príchodzie spojenie na porte 80. Po nadviazaní spojenie, je komunikácia presunutá na iný, náhodne zvolený port a port 80 je znova voľný. Každé HTTP spojenie je uložené do štruktúry `EMG_HTTP_SESSION`. Maximálny počet spojení je potom definovaný konštantou `MAX_NUMBER_OF_SESSIONS`. V štruktúre sú uložené informácie o stave spojenia, type metódy (GET/POST/EMG), type súboru a odkaz na aktívny socket. Obsluhu všetkých pripojených klientov zvláda jedna RTOS úloha. V cykle prechádza zoznam spojení a podľa stavu spojenia rozhodne o obsluhe. Nasledujúci klient je obslúžený až po dokončení predchádzajúceho. Po obsluhe každého klienta sa úloha sama vzdá procesoru, aby dlho neblokovala celý systém.

6.4 Ukážková aplikácia

Ukážková aplikácia je rozdelená do dvoch častí. Prvá časť je TCP/IP server implementovaný pomocou mini – socketov. Server dokáže naraz obslúžiť niekoľko klientov a spojenie si udržiava až do odpojenia zo strany klienta.

Server je vytvorený ako jedna úloha RTOS, ktorá pravidelne kontroluje frontu na príchodzie spojenia do klientov. *Callback* funkcia hlavného socketu spojenia prijíma a do fronty ich vkladá. Klient môže po pripojení jednoduchými príkazmi ovládať LED diódy na doske alebo čítať/zapisovať na SD kartu. „SET LED1 ON“ rozsvieti LED1, „SET LED1 OFF“ ju zhasne. Príkaz „SDREAD addr32“ vráti blok pamäte uložený na adrese *addr32* a príkaz „SDWRITE dataBlock addr32“ zapíše blok dát na uvedenú adresu.

Druhá časť aplikácie je Freescale webový server prispôsobený na ovládanie periférií na VWS. Základ webového serveru je upravený TCP/IP server z prvej časti ukážkovej aplikácie. Na uloženej web stránke je formulár s vloženými tlačidlami ovládajúcimi dve LED diódy na doske. Stlačenie tlačidla „LED2 ON“ vygeneruje nasledujúcu HTTP hlavičku.

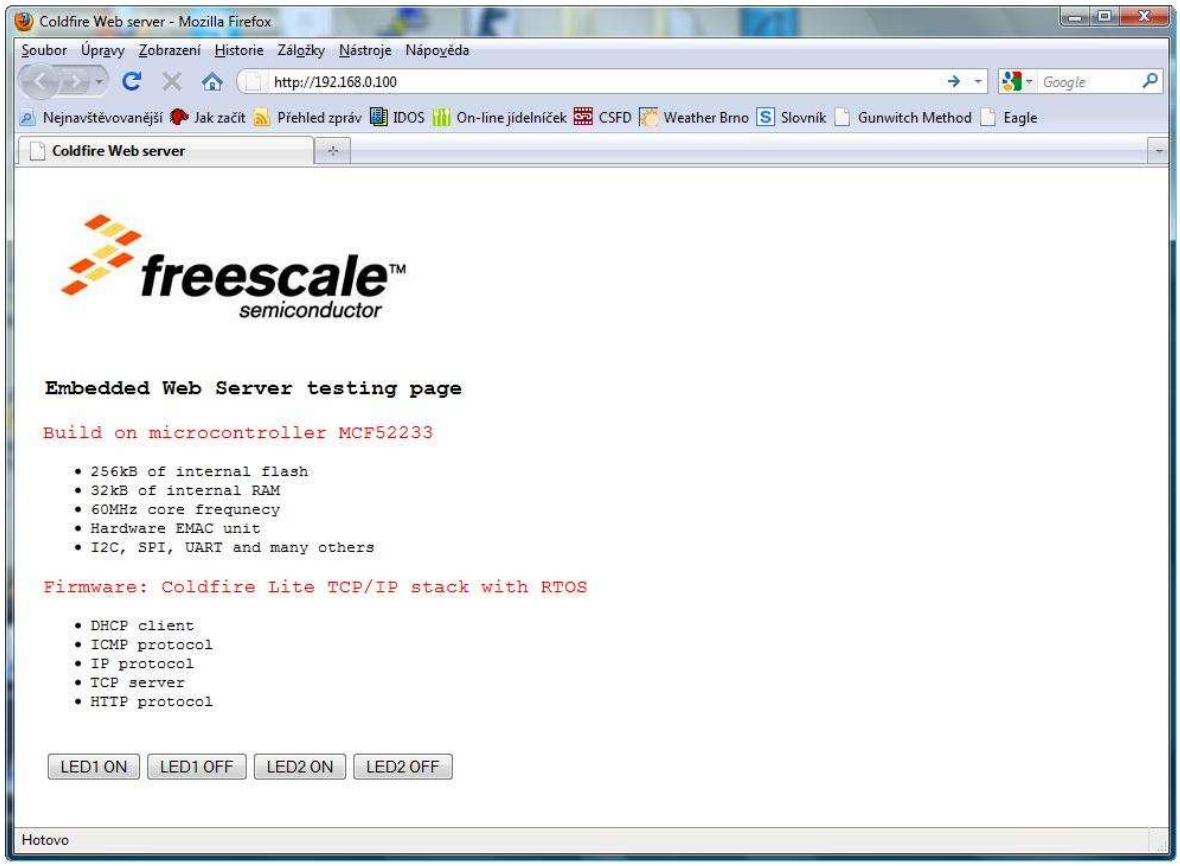
```
GET /index.htm?led=LED2+ON HTTP/1.1
Host: 192.168.0.100
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 6.0; cs; rv:1.9.2.3)
Gecko/20100401 Firefox/3.6.3
...
```

Web server podľa znaku „?“ za názvom súboru pozná, že nasledujú dáta z formuláru a posunie ich funkcii `process_form()` na spracovanie. Pre každú možnú položku formuláru na web stránke si

musíme vytvoriť záznam typu `FORM_STRUCTURE` v kóde web serveru. Funkcia `process_form()` tak podľa názvu položky zavolá príslušnú obslužnú funkciu, ktorá zapne alebo vypne LED diódu. Rovnakým spôsobom je možné ovládať akúkoľvek pripojenú periférium.

```
typedef struct {
     unsigned char form_name[8];    /* Názov položky */
     void (*func)(char *);          /* Pointer na obslužnú funkciu */
} FORM_STRUCTURE;

FORM_STRUCTURE forms[] = {
      {“led”,   form_led_function},
      {“relay”, form_relay_function}
};
```



Obr. 6.16: Ukážková aplikácia

Web stránka, ktorá sa má zobraziť ako prvá pri prístupe na server musí byť uvedená ako prvá v zozname súborového systému. Meno súboru stránky nie je dôležité. Zdrojové súbory k aplikácii sú v prílohách na CD.

7 Záver

Cieľom diplomovej práce bolo navrhnúť a zrealizovať vstavaný webový server z prostriedkov od firmy Freescale. Po teoretickom úvode do problematiky sieťovej komunikácie vstavaných systémov nasleduje návrh a výber vhodných komponent. Procesor je najdôležitejšou súčiastkou a jeho výberu bola venovaná aj na patričná pozornosť. Použitý procesor *MCF52233* má dostatočný výkon pre použitie s operačným systémom a integrovaný radič ethernetu zvláda komunikáciu rýchlosťou až 100 Mbps. Rozhrania procesora určujú, aké periférie je možné pripojiť. Podľa zadania má VWS slot pre micro SD kartu pripojenú SPI rozhraním. Kartu je možné od procesoru fyzicky opojiť a cez SPI pripojiť iné periférie. VWS má jeden USB port. Jeho hlavnou úlohou je možnosť pripojiť konzolu. VWS je tak možné ovládať aj pri strate sieťovej konektivity. Pretože VWS nemá žiaden zobrazovací prvok, všetky stavové informácie sa zobrazujú do konzoly. Stav je možné signalizovať aj na dvoch pripojených LED diódach. Pre zvýšenie funkčnosti VWS je na doske osadené jedno bistabilné relé. Oba spínacie/rozpínacie kontakty sú vyvedené do konektoru a je možné spínať výkony do 30 W. Do konektoru sú vyvedené aj všetky nevyužité signály: vstupy externého prerušenia, vstupy A/D prevodníku, signály I^2C rozhrania, vstupy a výstupy DMA časovačov a signály sériového portu UART.

K procesorom ColdFire je od Freescale dostupná implementácia rodiny protokolov TCP/IP pod názvom ColdFire Lite TCP/IP stack. Je to odľahčená verzia komerčne dostupnej verzie od firmy InterNiche. Stack presne kopíruje vrstvový model internetu a aplikačné rozhranie je odvodené od BSD socketov. Na odoslanie alebo príjem dát stačí jedna, resp. dve funkcie a implementovať sieťovú komunikáciu preto nie je problém. Po úpravách kódu bol TCP/IP stack prispôsobený vyvíjanému VWS. Aké protokoly bude výsledná aplikácia používať si môžeme definíciami v kóde navoliť. Na TCP/IP stacku je postavená ukážková aplikácia webového serveru. Po načítaní web stránky z pamäte procesora, môžeme ovládať periférie na doske. Pôvodný návrh počítal s web stránkami uloženými na SD karte a súborovým systémom FAT16. To sa však nepodarilo implementovať a preto sú web stránky uložené vo flash pamäti procesoru a pri každej zmene ich treba nahrať spolu s ovládacím softwarom. Ukážková aplikácia web serveru zaberá 96 kB a teoreticky tak máme priestor o veľkosti 160 kB na uloženie web stránok. S rozšírením v podobe dát uložených na SD karte sa do budúcna počíta a v prílohách môžme nájsť ovládač SPI rozhrania spolu s ovládačom SD karty.

Komerčné výrobky z kapitoly 2.3.1 a 2.3.2 majú síce minimálne rozmery ale mimo ethernetového rozhrania nemajú žiadne iné spojenie s okolím. To je ponechané na systém ku ktorému sú pripojené. Realizovaný VWS spája dohromady prístup na internet a interakcie s okolím v ktorom sa nachádza. Využitie tak nenájde len v oblasti riadenia alebo monitoringu na diaľku, ale napríklad aj ako vývojová doska na testovanie aplikácii vstavaného internetu.

Literatúra

[1] Balog, P.: *Network and Internet Integration of Embedded Systems*. University of Applied Sciences, Viedeň, 2002.
URL: http://embsys.technikum-wien.at/staff/balog/documents/conference-papers/ew05_inet-embsys_paper.pdf

[2] Torres, S.: *Web Server Development with MC9S12NE64 and OpenTCP*. Freescale Semiconductors Inc., 2004
URL: http://www.freescale.com/files/microcontrollers/doc/app_note/AN2836.pdf

[3] Gregori, E.: *ColdFire Lite HTTP Server*. Freescale Semiconductors Inc., 2007
URL: http://www.freescale.com/files/microcontrollers/doc/app_note/AN3455.pdf

[4] Gregori, E.: *ColdFire TCP/UDP/IP Stack and RTOS*. Freescale Semiconductors Inc., 2007
URL: http://www.freescale.com/files/microcontrollers/doc/app_note/AN3470.pdf

[5] Atmel Application Note.: *AVR460: Embedded Web Server*. Data Book, USA, 2005
URL: http://www.atmel.com/dyn/resources/prod_documents/doc2396.pdf

[6] Dunkels, A: *The uIP Embedded TCP/IP Stack*. Swedish Institute of Computer Science, 2006
URL: http://www.sics.se/~adam/download/uip-1.0-refman.pdf

[7] Riley, S., Breyer, R.: *Switched, Fast, and Gigabit Ethernet 3rd Edition*. New Riders Publishing, Indianapolis, 1999. ISBN 15-787-0073-6

[8] Designer Reference Manual.: *SD Card Reader Using the M9S08JM60 Series*. Freescale Semiconductors Inc., 2008
URL: http://www.freescale.com/files/microcontrollers/doc/ref_manual/DRM104.pdf

[9] *MCF52235 ColdFire Integrated Microcontroller Reference Manual*. Freescale Semiconductors Inc., 2007
URL: http://www.freescale.com/files/32bit/doc/ref_manual/MCF52235RM.pdf

[10] *FT232R USB UART IC Datasheet Version 2.01*. FTDI International Ltd., 2008
URL: http://www.ftdichip.com/Documents/DataSheets/DS_FT232R.pdf

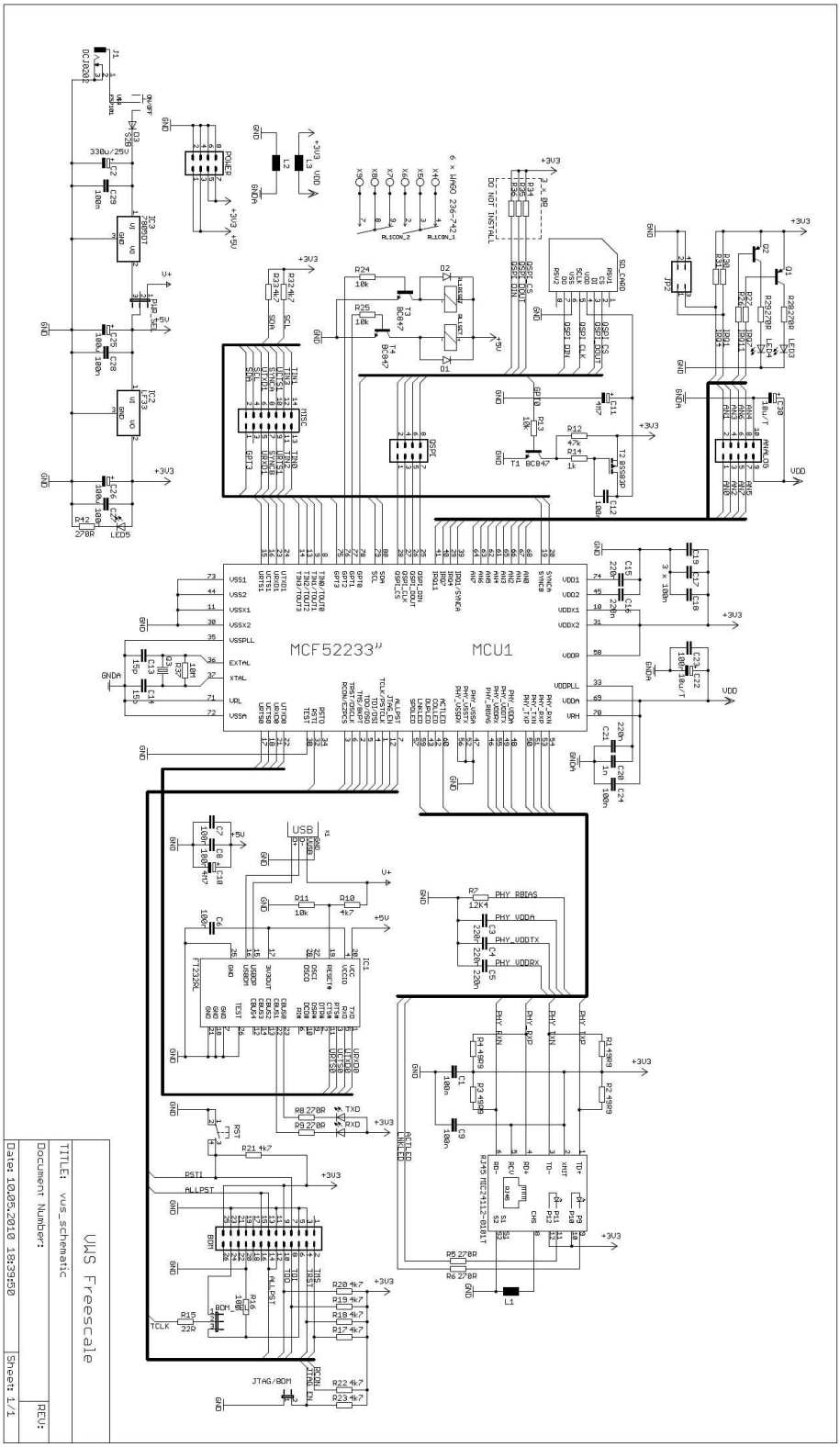
[11] Foust, F: *Secure Digital Card Interface for the MSP430*. Michigan State University, 2004
URL: http://www.cs.ucr.edu/~amitra/sdcard/Additional/sdcard_appnote_foust.pdf

[12] SOS Electronic: *Online katalóg elektronických súčiastok*. SOS Electronic, 2010
URL: http://www.sos.cz

[13] *Ethernet Tutorial*. Fujitsu Network Communications Inc, 2006
URL: http://www.fujitsu.com/downloads/TEL/fnc/pdfservices/ethernet-prerequisite.pdf

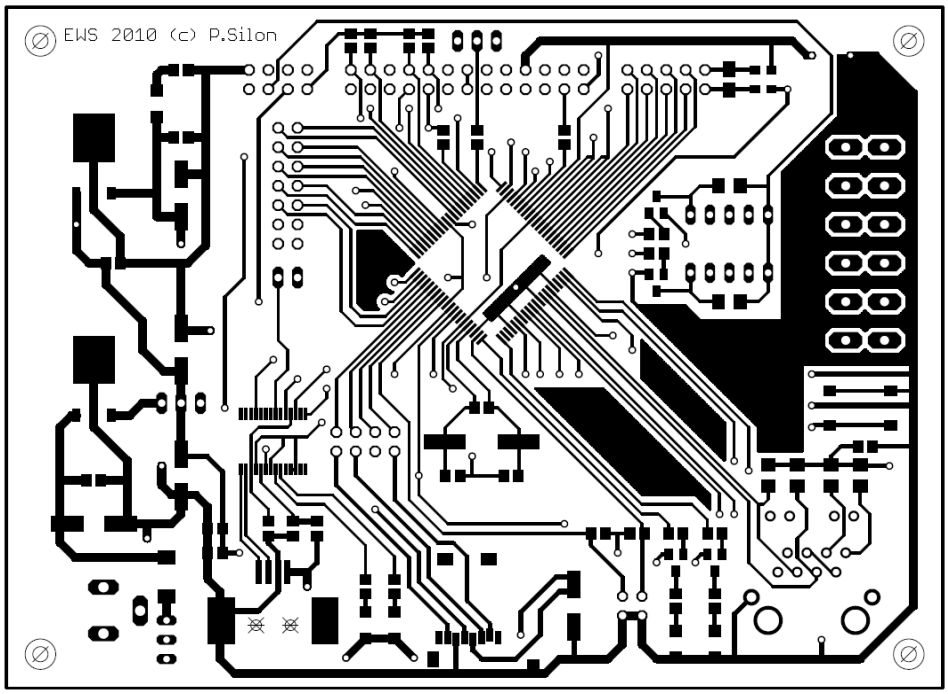
Príloha A

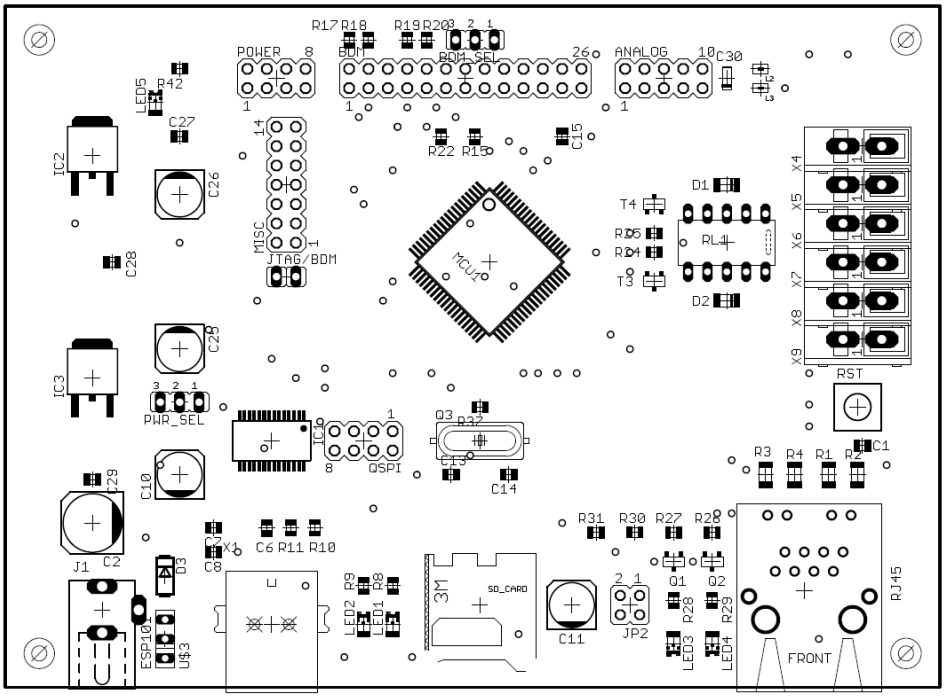
Schéma zapojenia



Príloha B

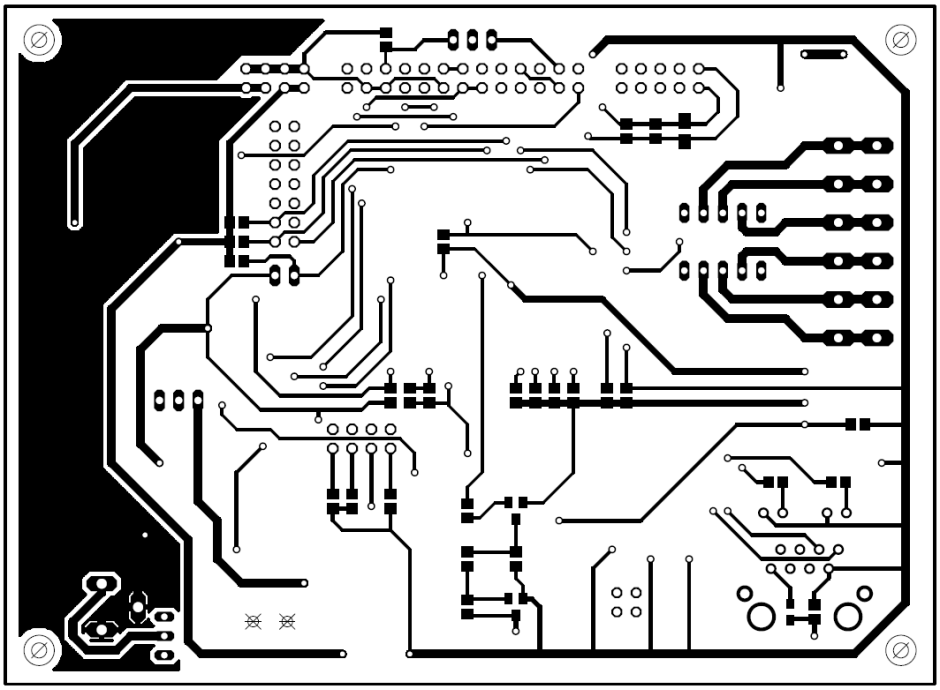
DPS Vrstva Top

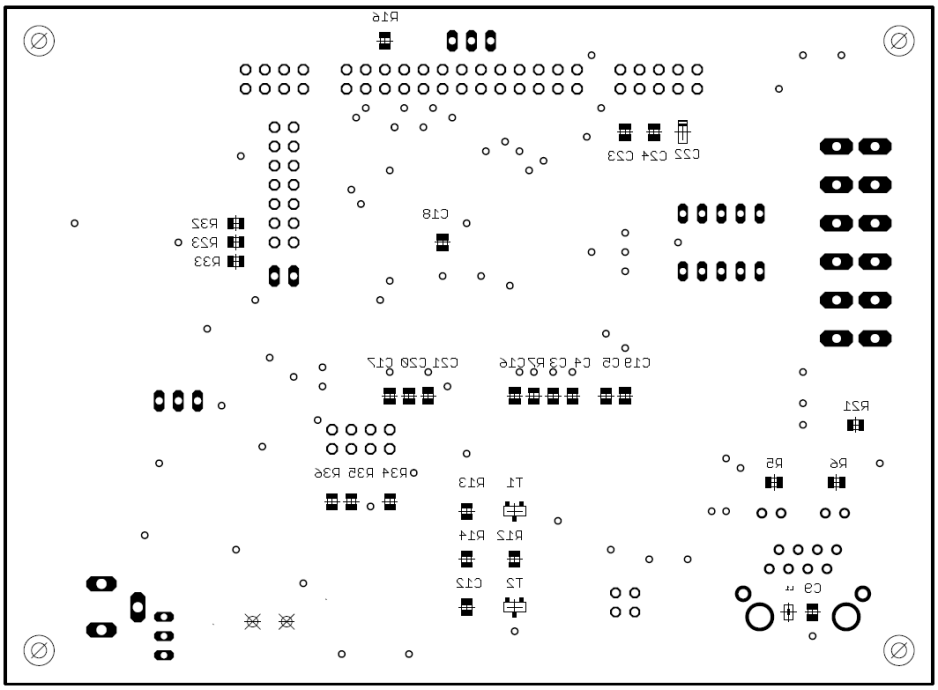




Príloha C

DPS Vrstva Bottom





Príloha D

Zoznam súčiastok

Part	Value	Device	Package
ANALOG		MA05-2	MA05-2
BDM		MA13-2	MA13-2
BDM_SEL		JP2E	JP2
C1,6,7,8,9,12,17,18	100n	C-EUC0805	C0805
C19,23,24,27,28,29	100n	C-EUC0805	C0805
C2	330u/25V	CPOL-SMD_E7	SMD_E7
C3,4,5,15,16,21	220n	C-EUC0805	C0805
C10,11	4M7	CPOL-SMD_C6	SMD_C6
C13,14	15p	C-EUC0805	C0805
C20	1n	C-EUC0805	C0805
C22,30	10u/T	CPOL-EUA	SMD_A
C25,26	100u	CPOL-SMD_C6	SMD_C6
D1,D2	1N4148	1N4148	R1206
D3	S2B	DIODE-DO214AA	DO214AA
IC1	FT232RL	FT232RL	SSOP28
IC2	LF33	7806DT	TO252
IC3	7805DT	7805DT	TO252
J1	DCJ0202	DCJ0202	DCJ0202
JP2		JP2Q	JP2Q
JTAG/BDM		JP1E	JP1
L1,L2,L3		WE-CBF_0805	0805
LED1,2,3,4,5		LEDCHIPLED_1206	CHIPLED_1206
MCU1	MCF52233	MCF52233	LQFP-80
MISC		MA07-2	MA07-2
POWER		MA04-2	MA04-2
PWR_SEL		JP2E	JP2
Q1,Q2	BC857BSMD	BC857BSMD	SOT23
Q3	25.000MHz	CRYSTALHC49UP	HC49UP
QSPI		MA04-2	MA04-2
R1,2,3,4	49R9	R-EU_R1206	R1206
R5,6,8,9,28,29,42	270R	R-EU_R0805	R0805
R7	12K4	R-EU_R0805	R0805
R10,17,18,19,20	4k7	R-EU_R0805	R0805
R21,22,23,32,33	4k7	R-EU_R0805	R0805
R11,13,16,24	10k	R-EU_R0805	R0805
R25,26,27,30,31	10k	R-EU_R0805	R0805
R12	47k	R-EU_R0805	R0805
R14	1k	R-EU_R0805	R0805
R15	22R	R-EU_R0805	R0805
R34,35,36	0R	R-EU_R0805	R0805
R37	10M	R-EU_R0805	R0805
RJ45	MIC24112-0101T	MIC24112-0101T	RJ45

Part	Value	Device	Package
RL1	ALD5WK	ALD5WK	DIL10
RST	DTSM-6	DTSM-6	DTSM-6
SD_CARD	2908-05WB-MGSMT	2908-05WB-MGSMT	3M_SMT
T1,T3,T4	BC847	BC847	SOT23
T2	BSS83P	BSS83P	SOT23
X1	USB B	USB-B-S	USB-B-SMT
X4..X9	WAGO236	W236-1	WAGO236

Príloha E

CD s prílohami v elektronickej podobe

Obsah CD

`ColdFire_Stack` – ColdFire (InterNiche) TCP/IP stack upravený pre realizovaný VWS. Pridané sú aj zdrojové súbory pre ovládanie periférií a SD karty.

`Eagle_Files` – Schéma zapojenia a doska plošného spoja vo formáte programu Eagle verzie 4.16.

`docs` – Schéma zapojenia a doska plošného spoja vo formáte *pdf*, elektronická verzia diplomovej práce vo formáte *pdf* a *doc*.

`misc` – dokumentácia k použitému mikrokontroléru a súčiastkam, obrázky použité v diplomovej práci a obrázky VWS.