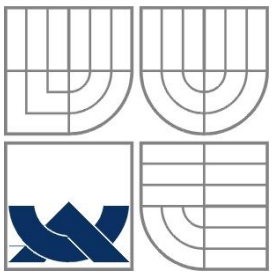
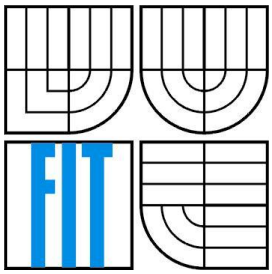


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

ZAŘÍZENÍ PRO VZDÁLENÝ WAKE-ON-LAN

REMOTE WAKE-ON-LAN

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. IVO PITNER

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. ZDENĚK VAŠÍČEK

BRNO 2011

Abstrakt

Tato práce se zabývá tvorbou vestavěného systému se vzdáleným přístupem, který ovládá počítače ve stejné síti s cílem snížit jejich spotřebu. K tomuto účelu využívá techniku WakeOnLAN. Zařízení dále využívá dva výkonové prvky typu SSR, využité ke spínání přívodu elektrické energie připojených spotřebičů. V práci je diskutována architektura TCP/IP, návrh vestavěného systému a na závěr je k danému systému implementován firmware.

Abstract

This work deals with construction of an embedded system with remote access, whitch controls computers in the same network in order to reduce their energy consumption. For this purpose is used the WakeOnLAN technique. The device also uses two power components called SSR and these are used for switching AC power to them attached appliances. The work discusses the TCP/IP architecture, system design and finally the firmware implementation.

Klíčová slova

WakeOnLAN, TCP/IP, Ethernet, spínání zátěže, SSR, MSP430F2617, ENC28J60, FT232RL, M41T81.

Keywords

WakeOnLAN, TCP/IP, Ethernet, power switching, SSR, MSP430F2617, ENC28J60, FT232RL, M41T81.

Bibliografická citace

Ivo Pitner: Zařízení pro vzdálený WakeOnLAN, diplomová práce, Brno, FIT VUT v Brně, 2011

Prohlášení

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

.....
Bc. Ivo Pitner
25. května 2011

Poděkování

Na tomto místě bych rád poděkoval panu Ing. Zdeňku Vašíčkovi, vedoucímu mé práce, za trpělivé vedení a především za jeho neocenitelnou pomoc při realizaci. Dále pak panu Ing. Václavu Šimkovi za jeho rady při konstrukci desky plošných spojů.

Obsah

1 Úvod

V posledních letech došlo ve světě k výraznému rozšíření elektrických či elektronických zařízení. Lidé kupují neustále nové televizory, počítače a další dnes oblíbené spotřebiče, ať už jsou určeny k užitečné činnosti nebo zábavě. Většinou neberou v úvahu fakt, že jsou neustále připojeny do elektrické sítě a permanentně odebírají elektrickou energii, i když nejsou zrovna používány. Proti tomu se zaměřily některé organizace a podnikly řadu tahů, omezující zbytečné plýtvání s touto v dnešní době nepostradatelnou komoditou. Jako příklad jmenuji zrušení obyčejných žárovek nad 60 Wattů a jejich nahrazení úspornými lampami. V konečném důsledku závisí na každém z nás, zda se rozhodne elektrickou energií šetřit nebo ne. Proto se nabízí možnost vytvoření nízkopříkonového systému, který by umožnil řízení ostatních spotřebičů a usnadnil tak běžnému uživateli ovládání spotřeby své domácnosti. I ve školách či jiných institucích by šetření energie takovým způsobem mohlo supořit byť jen minimální náklady.

Tato diplomová práce navazuje na semestrální projekt, v rámci kterého byla vypracována studie potřebných technologií k vytvoření vestavěného zařízení využívající techniku WakeOnLAN spolu s možností ovládat přívod energie několika spotřebičů za pomoci výkonových prvků. V rámci diplomové práce byla studie rozšířena a dovedena k praktické realizaci. Kromě teoretické teoretických částí zde může čtenář nalézt i popis implementace celého problému, zvláště pak firmware zařízení a výsledných vlastností vestavěného systému.

Dokument je členěn následovně. Po úvodu následuje kapitola, která specifikuje požadavky na funkčnost a vybavení celého systému. Ve třetí kapitole jsou probrány síťové protokoly a technologie využívané v lokálních sítích a Internetu. Čtvrtá kapitola popisuje různé typy zátěží, s kterými se lze v běžné praxi setkat. Dále pak s možnostmi jejich spínání. V páté kapitole je realizován návrh celého systému a jako výstup tak vzniklo schéma, podle něhož byl celý systém vyroben a následně sestaven. Šestá kapitola přiblíží čtenáři proces implementace firmware a rozdělení programu do několika knihoven. Předposlední, sedmá, kapitola demonstruje výsledné vlastnosti a nasazení prvku do infrastruktury počítačové sítě. V poslední kapitole jsou shrnuty dosažené výsledky a navrhnuto několik způsobů možných rozšíření systému.

2 Specifikace řešeného problému

Cílem práce je navrhnout vestavěné zařízení, které umožní spínat spotřebiče nebo probouzet počítače pomocí techniky WakeOnLAN. Navržené zařízení by mělo přispět ke správě napájení ostatních zařízení a tím tak šetřit elektrickou energii. Zároveň by mohlo být využito například k přípravě laboratoře a spustit tak měřící či jiné přístroje, které budou při příchodu uživatele zapnuty a připraveny k okamžitému použití.

2.1 Požadavky na funkčnost

Zařízení bude připojeno na počítačovou síť, kde naslouchá na příchozí požadavky. Jeho ovládání bude možné odkudkoliv přes lokální, podnikovou nebo internetovou síť. K ovládání přes Internet je nutné, aby systém vlastnil svou vlastní veřejnou IP adresu, nebo musí být povoleno přeposílání (tzv. forwarding) zvoleného portu přes bránu sítě. Pro snadné ovládání by mělo být přítomno jednoduché webové rozhraní, rozdělené na dvě části. Jedna část bude mít na starosti probouzení technikou WOL, druhá část pak spínání elektrické energie k jednotlivým zařízením.

Výhodou by také mohl být časovač, kterým si uživatel nastaví rozmezí nebo dobu, kdy zvolené zařízení sepnout či odpojit od elektrické sítě. Systém by tak měl mít povědomí o aktuálním denním čase. Toho lze dosáhnout zjišťováním času přes různé time servery (například český SNTP server time.nic.cz) nebo v případě offline režimu, kdy není možné získat čas prostřednictvím Internetu, je vhodné použít obvod reálného času. Nastavení času si uživatel provede sám přes webové rozhraní.

2.2 Požadavky na hardwarovou vybavenost

Jelikož jde o vestavěný systém, lze počítat s nízkou spotřebou a podle toho by měly být voleny komponenty zařízení. Výsledkem práce by tak měla být deska plošných spojů obsahující alespoň následující prvky:

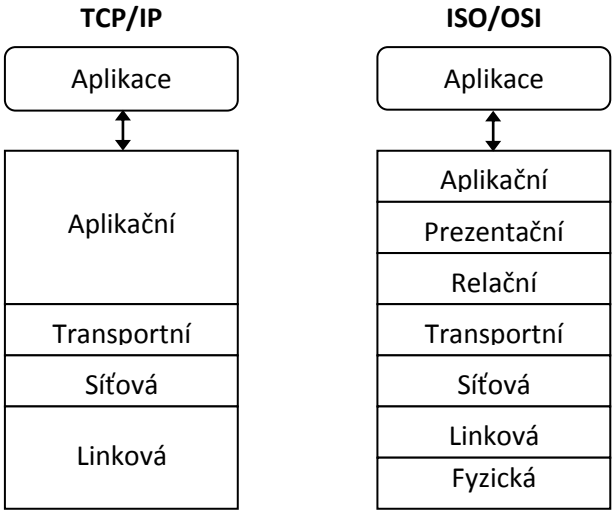
- **Mikrokontrolér (MCU)** Hlavní řídící prvek, obsluhující všechny ostatní části desky. MCU musí mít dostatek portů k připojení periferií, malou spotřebu a dostatečný výkon.
- **Obvod reálného času** Udržuje aktuální čas, i když je mikrokontrolér v režimu spánku nebo je zrovna programován. Umožňuje také plánování událostí.
- **USB rozhraní** Programování mikrokontroléru, napájení, přístup na zařízení.
- **Rozhraní Ethernet** Umožňuje komunikaci po síti a využití WOL.
- **Výkonové spínací prvky** Ovládání přívodu napájení ostatních zařízení.

3 Komunikace po síti

V této kapitole bude čtenář seznámen s existujícími síťovými protokoly definovaných v architektuře TCP/IP, na kterých staví dnešní moderní sítě. Předně jde o protokol rozhraní Ethernet sloužící ke komunikaci v lokální síti a protokol IP, s jehož pomocí lze směrovat odesílané pakety ve větších sítích. Dále jsou diskutovány transportní protokoly TCP a UDP, a jejich služby využívající aplikační protokoly HTTP, SNTP a DHCP. Aplikační protokoly jsou zaměřeny na realizaci webového serveru, zjišťování času prostřednictvím sítě a automatickou konfiguraci IP adresy zařízení. Nakonec je dopodrobna vysvětlena síťová technika probouzení přes síť – WakeOnLAN.

3.1 Protokolová architektura TCP/IP

Moderní počítačové sítě, ať už jde o síť Internet či menší lokální a podnikové sítě staví na architektuře TCP/IP. Právě ona byla u zrodu Internetu, jak jej známe dnes, jelikož umožnuje tvořit rozlehlé sítě bez nutnosti centrálního uzlu. Byla vyvinuta v rámci projektu ARPAnet v 70. letech minulého století a jako inspirace posloužil komunikační model ISO/OSI. Jejich porovnání je zobrazeno na obrázku 3.1.



Obrázek 3.1: Architektura TCP/IP a její vzor – model ISO/OSI.

Architektura definuje kompletní rodinu protokolů, za jejichž pomoci je realizována komunikace v různých typech sítí. Je složena ze čtyř abstraktních vrstev. První vrstva nese název aplikační a je umístěna na logickém vrcholu architektury. Obsahuje protokoly využívající konkrétních síťových služeb na koncovém prvku. Integruje aplikační, prezentační a relační vrstvy modelu ISO/OSI. Typickým zástupcem může být protokol HTTP, FTP nebo DNS. Protokoly v této vrstvě jsou přímo využívány jednotlivými službami na zdrojovém či cílovém zařízení. Většinou nemají žádný vliv na vlastní přenos přes síť, o ten se starají až další vrstvy. Úkolem druhé transportní vrstvy je zajistit přímé spojení mezi dvěma koncovými účastníky. Hlavními protokoly druhé vrstvy bývají v drtivé většině spolehlivý protokol TCP nebo nespolehlivý UDP. Směrování mezi jednotlivými sítěmi má na starosti síťová vrstva a protokol IP, kde slouží jako hlavní identifikátor IP adresa. V této vrstvě existují i další protokoly, kupříkladu protokol ICMP nebo ARP a doplňují funkčnost protokolu IP. Nejnižší vrstva

(linková) není v rámci architektury TCP/IP blíže specifikována, jelikož záleží na použité přenosové technologii. Může to být například Ethernet, Token Ring či Wi-Fi. Zde probíhá vlastní odesílání a přijímání skrze přenosové médium [2].

3.1.1 Ethernet

Mluví-li dnes někdo o lokální síti (LAN), má nejspíše na mysli Ethernet. Toto rozhraní vzniklo v roce 1980 pod názvem DIX Ethernet a vyvinuly ho známé firmy jako je Intel, Xerox a Digital Equipment Corporation. Poté byl standardizován pod názvem 802.3 a od svého vývoje prošel mnoha zdokonaleními. Dnes dosahuje rychlostí až 10 Gbitů za sekundu a mezi jeho nejvíce používané verze patří Fast Ethernet (100BaseT) s propustností 100Mbit/s a Gigabit Ethernet (1000BaseT) až s 1Gbit/s. Setkat se ale můžeme i s pomalejší verzí s označením 10BaseT, zvládající 10Mbit/s, která je vhodná pro zařízení s nízkými nároky na datovou propustnost [1].

Protokol Ethernet spadá do linkové vrstvy modelu TCP/IP a zapouzdřuje data, která přijímá ze síťové vrstvy. Zapouzdřeným datům se říká rámec a jde o sekvenci bitů, které jsou přenášeny po lince ostatním zařízením. Formát rámce je jednotný, aby byla zachována kompatibilita mezi různými typy rozhraní. Jeho složení je na obrázku 3.2 [1].

Preamble [7 bytes]	SFD [1 byte]	DST [6 bytes]	SRC [6 bytes]	Lenght/Type [2 bytes]	Data [46 -1500 bytes]	FCS [4 bytes]

Obrázek 3.2: Formát Ethernet rámce.

Význam jednotlivých polí je následující:

- **Preamble** Skládá se z různé kombinace nul a jedniček, slouží k synchronizaci hodinových signálu.
- **SFD (Start of Frame Delimiter)** Obsahuje 6 bitů kombinace nul a jedniček následovaných dvěma jedničkami (10101011). Značí začátek rámce.
- **DST (Destination Adress)** Adresa adaptéru, kterému je paket odesílán.
- **SRC (Source Adress)** Obsahuje adresu adaptéru, který paket vygeneroval.
- **Lenght/Type** Definuje délku pole Data (kromě výplně) nebo obsahuje typ přenášeného protokolu.
- **Data** Obsahuje odeslaná/přijatá data od síťové vrstvy. Jsou-li data příliš krátká (méně než 46 Bytů) adaptér doplní sekvenci bitů na minimální velikost.
- **FCS (Frame Check Sequence)** Obsahuje kontrolní hodnotu, využívanou ke zjištění, zda nedošlo k chybám v paketu.

V poli Length/Type se běžně vyskytují hodnoty 0x0806, 0x0800 nebo 0x86DD. První znamená, že je přenášen protokol ARP, druhá IPv4 datagram a třetí IPv6 datagram. Délka dat je pak určena z polí protokolů vyšších vrstev.

Cílová a zdrojová adresa udávají jedinečný síťový identifikátor nazývaný MAC adresa nebo někdy také fyzická adresa. Jde o šesti bajtovou hodnotu, zapsanou většinou v podobě hexadecimálních číslic oddělených dvojtečkou. První tři bajty udávají identifikátor výrobce zařízení a zbylé tři bajty jsou ponechány jeho číslování. Velikost adresy dává dostatečný prostor k tomu, aby nedocházelo ke konfliktům a každé zařízení mělo adresu jedinečnou v rámci celého

světa. Pokud se ale stane a jsou vyrobena dvě zařízení se stejnou adresou, je pravděpodobnost, že budou použita v rámci stejné sítě velmi malá. Speciálním případem adresy je tzv. broadcastová adresa tvořená samými jedničkami – zapsaná hexadecimálně jako FF:FF:FF:FF:FF:FF. Datagram odeslaný na broadcastovou adresu bude odeslán všem zařízením v rámci lokální sítě. [1].

3.1.2 Sada protokolů IP

Protokol IP je základní síťový protokol, který bývá většinou doprovázen dalšími podpůrnými protokoly, jako jsou například ARP, DHCP či protokol ICMP. Pracuje na principu hostitelů a sítí, kde hostitelem je jakékoliv zařízení s IP adresou. Skupina hostitelů se stejnou strukturou adres tvoří jednu síť. Pokud jsou v různých sítích, je nutné využít další podpůrná zařízení, jako jsou například směrovače. Adresování v síti spoléhá na to, že každý hostitel musí mít jedinečnou adresu, pomocí ní je nejprve nalezena síť a poté hostitel, se kterým je komunikováno. Adresa sestává ze 4 bytů (32bitů) vyjádřitelných jako 4 desítková čísla oddělená tečkami a bývá doplněna 32bitovou maskou, udávající podsíť [2].

Dnes existují dvě verze IP protokolu – IPv4 a IPv6. IPv4 je prozatím ve světě Internetu dominantní, ale kvůli krátké délce dresy dojde v řádu jednotek až desítek měsíců k vyčerpání jejího adresního prostoru. Z tohoto důvodu byla standardizována IPv6 jako náhrada, kde použití 128 bitové adresy zajišťuje téměř nemožné vyčerpání adres. IP protokol vytváří IP datagramy, které předá nižší vrstvě k následnému odeslání. Vzhled datagramu IPv4, je na obrázku 3.3.

B	1		2	3	4
0 - 3	Version	Header L.	Service Type	Datagram Length	
4 - 7	Identification			Flags	Offset
8 - 11	TTL		Protocol	Header CRC	
12 - 15	Source Adress				
16 - 19	Destination Adress				
20 - 23	Options				Padding
…	Data				

Obrázek 3.3: Formát IPv4 datagramu.

Popis jednotlivých polí IP datagramu:

- **Version** Určuje verzi IP protokolu zakódovanou ve čtyřech bitech (v4 nebo v6).
- **Header Lenght** Na 4 bitech nese hodnotu vyjadřující délku hlavičky ve slovech (minimální hodnota je 5, což představuje 5x32 = 160bitů neboli 20 bytů).
- **Service Type** Původně měla označovat charakter přepravní služby, nyní je používána hlavně pro mechanizmy QoS (Quality of Services).
- **Datagram Length** Udává celkovou délku datagramu včetně hlavičky a dat. Opět hodnota označuje délku ve slovech.
- **Identification** Při fragmentaci dat je využíváno k opětovnému sestavení datagramu.
- **Flags** Příznaky určující zda došlo ke fragmentaci a jakým způsobem.

- **Offset** Udává relativní pozici fragmetu vůči nefragmentovanému datagramu. Jednotka je 8 bajtový blok.
- **TTL** TimeToLive hodnota pomáhá, aby fragmenty nezůstaly „viset“ v síti. V podstatě jde o životnost paketu. Každý uzel v síti, kterým fragment projde, zmenší tuto hodnotu o jedna. Pokud TTL=0 znamená to, že paket došel ke svému příjemci, který jej zpracuje, nebo zahodí.
- **Protocol** Nese informaci jaký protokol vyšší vrstvy je přenášen. Většinou jde o protokoly TCP a UDP s hexa čísly 0x06 a 0x11.
- **Header CRC** Šestnácti bitový kontrolní součet využívaný ke kontrole hlavičky, přenášená data tak nejsou proti chybě zajištěna.
- **Source Adress** 32 bitová IP adresa odesílatele.
- **Destination Adress** 32 bitová IP adresa zařízení, jemuž je datagram určen.
- **Options** Umožňuje přidat další vlastnosti datagramu, ale položka se většinou nepoužívá.
- **Padding** Jde o zarovnání, které musí být přidáno, pokud pole Options nedosáhlo velikosti 4 Byty.

Adresování v síti probíhá na základě 32 bitových IP adres a jednotlivá zařízení lze adresovat jak individuálně, tak skupinově či všeobecně. Každé rozhraní musí mít svoji jedinečnou adresu, jinak by mohlo dojít ke konfliktu. Adresy jsou děleny do těchto pěti tříd [2]:

- **Třída A –** prvních 8 bitů je využito k adresaci dané sítě a zbylé číslo adresuje rozhraní. Tato třída poskytuje největší rozsah adres jednotlivých zařízení, ale může existovat maximálně 126 platných adres sítě. Speciálním případem jsou adresy v síti 127.0.0.0, zvláště pak adresa 127.0.0.1, která je využívána jako zpětná smyčka (loopback) a zařízení může adresovat samo sebe.
- **Třída B –** využívá prvních 16 bitů k adresaci sítě a zbylých 16 bitů na adresaci zařízení. Adresy třídy B využívají různé podniky nebo poskytovatelé přístupu k Internetu.
- **Třída C –** adresa sítě je určena prvními třemi osmibitovými čísly a poslední 8 bitové číslo určuje adresu zařízení. Adresy v této třídě bývají hojně využívány v lokálních sítích. Poskytují ovšem pouze 254 možných adres stanic (255 je všeobecná adresa v síti – broadcast).
- **Třída D –** je využívána ke skupinové adresaci. Existují speciální adresy jako například 224.0.0.1 sloužící k adresaci skupiny všech zařízení v lokální podsíti.
- **Třída E –** poskytuje nejmenší rozsah a je určena k experimentálním účelům.

Existují také adresy soukromých sítí. Tyto adresy byly nejprve určeny pro sítě nepřipojené k Internetu, ale z důvodu nedostatku adres jsou využívány pro sítě skryté za NAT. Jde o následující typy adres:

- **Třída A** 10.0.0.0 – 10.255.255.255.
- **Třída B** 172.16.0.0 – 172.31.255.255.
- **Třída C** 192.168.0.0 – 162.168.255.255.

V každé třídě jsou dále dva typy speciálních adres. První je adresa sítě a jde o adresu, která má nastaveny všechny bity určeny k adresaci zařízení na hodnotu „0“. Druhou je všeobecná adresa (broadcast), mající naopak všechny adresující bity nastaveny na „1“. Pomocí této adresy lze označit všechna zařízení v dané síti [2].

Samotné rozdělení do výše uvedených tříd způsobilo ohromné plýtvání adresami (např. třída A disponuje obrovským rozsahem adresovaných zařízení). Proto vznikly tzv. podsíťové adresy. Část pro adresaci jednotlivých zařízení v síti je tak možné rozdělit na více podsítí. Toho bylo dosáhnuto přidáním síťová masky. Maska má stejný formát jako IP adresa, kde jsou na pozici bitů značící adresu sítě samé jedničky. Zbylé bity jsou nastaveny na hodnotu 0. Maska pro plný rozsah ve třídě A má hodnotu 255.0.0.0, pro třídu B 255.255.0.0 a třídu C 255.255.255.0 [2].

K vyplnění pole CRC je využíváno 16 bitového jedničkového doplňku součtu všech 16 bitových slov v hlavičce. Při výpočtu je pole CRC prozatím vyplněno nulami. Výpočet jednoduchého kontrolního součtu ukáži v následujícím příkladu [19]:

Mějme hexadecimálně zapsanou hlavičku, kde poslední dva byty představují CRC:

01 01 F2 43 F4 F5 F5 F2 00 00,

Zformujeme 16 bitové slova:

0101 F243 F4F5 F5F2 0000

Spočteme sumu:

0101 + F243 + F4F5 + F5F2 + 0000 = 0002 DE2B (32 bitové slovo)

Sečteme opět 16 bitové slova:

0002 + DE2B = DE2D

Dostali jsme 16 bitové slovo, provedeme u něj jedničkový doplněk (negace):

¬DE2D = 21D2

Výsledek vložíme do pole CRC a dostaneme:

01 01 F2 43 F4 F5 F5 F2 21 D2

Kontrola probíhá sečtením:

0101 + F243 + F4F5 + F5F2 + 21D2 = 0002 FFFD

0002 + FFFD = FFFF

Výsledek FFFF značí, že je vše v pořádku.

Jelikož je kontrolní součet prováděn pouze nad prvky hlavičky, je nutné, aby si zabezpečení dat obstarala vyšší transportní nebo aplikační vrstva. V praxi se ovšem používá kontrolní součet na úrovni Ethernetového rámce v linkové vrstvě, protože není nutné počítat více kontrolních součtů. Také je potřeba dávat pozor na uložení dat v paměti, jelikož jsou po síti posílány a po přijetí musí být interpretovány ve formátu big endian (MSB) – nejvíce významný bajt posílán vždy jako první [1].

3.1.3 Adress Resolution Protocol

ARP patří stejně jako IP protokol do síťové vrstvy modelu ISO/OSI. Byl vytvořen za účelem převodu IPv4 adresy na fyzickou MAC adresu. Tato situace nastává v okamžiku, kdy chce jedno zařízení komunikovat s druhým ať už ve stejné nebo jiné síti. Odesílatel zná pouze adresu cíle, ovšem neví jaká je mezi nimi topologie. Proto musí každý prvek na cestě provést převod IP adresy na fyzickou, aby zjistil, kterým směrem dál data odeslat. Rozesílání paketů přes jednotlivé prvky (opakovače, routery, stanice) totiž probíhá právě na základě MAC adresy.

Protokol ARP pracuje na principu dotaz/odpověď. Zařízení, na které přijde paket s IP adresou cíle, odešle ARP dotaz v podobě linkového broadcastu (adresa FF:FF:FF:FF:FF:FF). Na ten mu odpoví uzel, který zná požadovanou IP adresu svou fyzickou adresou a dané zařízení pak ví, kam data odeslat. Aby každý síťový prvek nemusel pokaždé při odesílání dat neustále zjišťovat, na kterou MAC adresu je má odeslat, uchovává si již zjištěné dvojice „IP adresa, MAC adresa“ ve svojí paměti. Při dalším odesílání se pak pouze podívá a je-li záznam nalezen, zařízení ví, kam data odeslat. [2]. Hlavičku ARP dotazu zobrazuje obrázek 3.4.

B	1	2	3	4	5	6
0 - 3	HW type		Protocol Type			
4 - 7	HW Addr. Length	Prot. Address Length	Opcode			
8 - 13	Source hardware address					
14 - 17	Source protocol address					
18 - 23	Destination hardware address					
24 - 27	Destination protocol address					

Obrázek 3.4: Struktura ARP dotazu.

Význam polí ARP dotazu je následující:

- **HW type** Specifikuje, jaká linková vrstva se používá. Například hodnota 1 patří Ethernetu.
- **Protocol Type** Přenáší informaci o použitém protokolu (např IPv4).
- **HW Address Length** Udává délku hardwarové adresy. Pro Ethernet je to velikost 6 bytů.
- **Protocol Adress Length** Velikost adresy používaného protokolu. V případě IPv4 je 4 byty.
- **Opcode** Obsahuje číslo operace, která je odesílána. Dotaz má hodnotu 1, odpověď pak 2.
- **Source Hardware Address** MAC adresa zařízení odesílatele.
- **Source Protocol Adrdress** IP adresa odesílatele.
- **Destination Hardware Address** MAC adresa cílového zařízení.
- **Destination Protocol Address** IP adresa cíle.

3.1.4 Internet Control Message Protocol

Neméně důležitou částí architektury TCP/IP, na které závisí v podstatě celá síť internet, představuje kontrolní protokol ICMP (Internet Control Message Protocol). Využívá se zejména k hlášení kontrolních, řídících a chybových zpráv. Nepoužívá žádný protokol transportní vrstvy, bývá přenášen přímo v IP datagramech a obvyklým způsobem odesílán. ICMP zpráva zapouzdřuje data v jednom jediném IP datagramu a podobně jako UDP nezaručuje doručení.

B	1	2	3	4
0 - 3	Type	Code	CRC	

Obrázek 3.5: Hlavička protokolu ICMP.

Zpráva sestává z několika položek (viz obrázek 3.5). Nastavení políčka **Type** ovlivňuje jaký typ zprávy je odesílán. Její parametry jsou nastaveny v poli **Code**. Pro kontrolu chyb obsahuje hlavička pole **CRC**, kde je umístěn kontrolní součet spočtený nad celým datagramem [2].

Mezi nejběžnější zprávy patří zpráva typu 0 (Echo reply) a 8 (Echo request), které využívá například program ping. Mezi další patří typ 3 (Destination unreachable) nesoucí informaci, že cílová adresa není dostupná a onen důvod obsahují další části ICMP zprávy. Ostatní typy zpráv jsou uvedeny na [15].

3.1.5 Transmission Control Protocol

TCP je v počítačových sítích jedním ze dvou nejčastěji využívaných protokolů transportní vrstvy. Jeho hlavním úkol spočívá ve vytvoření virtuálního spoje mezi dvěma službami v síti. Protokol zajišťuje spolehlivou transportní službu, která doručí příjemci data tak, jak byly vyslány odesílatelem. Ke komunikaci mezi dvěma účastníky musí být nejprve vytvořeno spojení. To znamená, že než bude možné odesílat data, musí být spojení řádně navázáno a po vlastním přenosu následně ukončeno. Koncové body mohou komunikovat v režimu plného duplexu, tedy oba mohou vysílat zároveň [2]. Identifikace služby neboli přenášeného aplikačního protokolu se na daném zařízení provede na základě portu, uvedeného v hlavičce.

B	1		2	3	4
0 - 3	Src Port			Dst Port	
4 - 7	SEQ Number				
8 - 11	ACK Number				
12 - 15	Data Offset	Reserved	Flags	Window Size	
16 - 19	Checksum			Urgent Pointer	
20 - 23	Options				
…	Data				

Obrázek 3.6: Struktura TCP paketu.

Obrázek 3.6 zobrazjuje hlavičku TCP protokolu včetně popisu jednotlivých polí. Jejich význam je následující:

- **Src Port** Zdrojový port.
- **Dst Port** Cílový port.
- **SEQ Number** Číslování paketů – jeden z mechanizmů bezchybného doručování.
- **ACK Number** Číslo odpovědi, další část spolehlivého přenosu.
- **Data Offset** Specifikuje velikost TCP hlavičky v 32 bitových slovech. Minimum je 5 a maximum 15 slov, tedy 20 nebo 60 bytů.
- **Reserved** Pro budoucí účely, mělo by být vynulováno.
- **Flags** Příznaky pro řízení přenosu, každý bit má přidělen svůj význam. Pole obsahuje bity s názvem URG (signalizuje urgentní data), SYN (žádost o navázání spojení), ACK

(kladné potvrzení), RST (opětovné navázání spojení), PSH (okamžité doručení) a FIN (žádost o ukončení spojení).

- **Window Size** Udává velikost okna, které může příjemce maximálně přijmout.
- **Checksum** Pole pro umístění 16 bitového kontrolního součet pro hlavičku a data.
- **Urgent Pointer** Pokud je nastaven příznak URG v položce Flags má toto pole význam offsetu od sekvenčního čísla, identifikující poslední datový byte.
- **Options** Prostor pro dodatečné volby přenosu.

Odesílání dat s využitím TCP spojení rozlišuje tři fáze komunikace [2]:

- **Navázání spojení** Probíhá na principu tzv. „three way handshake“, kde je prvním krokem odeslání paketu SYN s nastaveným počátečním číslem (pole **SEQ Number**). Příjemce potvrdí opět zprávou SYN s vyplněným polem **ACK Number** (na číslo o jedna větší) a zároveň uvede svoje počáteční číslo v poli **SEQ Number**. Nakonec odesílatel potvrdí paketem typu ACK s oběma čísly zvýšenými o jedna.
- **Vlastní přenos dat** V tomto bodě jsou vyměňována data. Protokol přímá data jako bajtový proud a dělí je na jednotlivé pakety, ty opatřuje hlavičkou a sekvenčním číslem a pro každý odeslaný paket čeká kladné potvrzení. Pokud není přijato nebo je signalizováno, že byl paket porušen, musí být paket opětovně odeslán.
- **Ukončení spojení** Podobně jako v navázání spojení, probíhá i zde předem určená sekvence paketů. V případě ukončení je vyžadováno, aby konec spojení signalizovali oba komunikující, jinak může dojít ke ztrátě dat. K ukončení slouží speciální typ zprávy s názvem FIN.

3.1.6 User Datagram Protocol

Jde o druhý protokol transportní vrstvy a oproti TCP nezaručuje spolehlivý přenos. To přináší výhodu nižších latencí a menší režie s konstrukcí a zpracováním paketu. Protokol UDP nemá fáze zahajování a ukončování, data jsou odesílána okamžitě. UDP se využívá ve službách, kde nevadí ztráta paketu nebo změna pořadí jejich příjmu a hlavně tam, kde je kladen důraz na rychlé doručení. Mezi typické služby využívající nespolehlivý přenos patří protokol DNS (využívá i TCP), přenos audio-vizuálních dat nebo využití při komunikaci v síťových hrách. K určení služby opět slouží jako identifikátor port, uvedený v hlavičce (obrázek 3.7). Stejně jako je požadováno, aby MAC adresa či IP adresa byly unikátní, mělo by i číslo portu být přiřazeno pouze jedné službě. Port reprezentuje jedno šestnáctibitové číslo, kde hodnoty 0 – 1024 identifikují známé aplikační protokoly, větší čísla lze pak využít libovolně [2].

B	1	2	3	4
0 - 3	Src Port		Dst Port	
4 - 7	Lenght		Checksum	
…	Data			

Obrázek 3.7: Hlavička UDP datagramu.

Význam polí je obdobný jako v protokolu TCP, tedy **Src Port** představuje zdrojový port, **Dst Port** cílový port, **Length** označuje délku dat v násobcích 32 bitového slova, **CRC** kontrolní součet a položka **Data** obsahuje data aplikačního protokolu [16].

3.1.7 Dynamic Host Configuration Protocol

Dynamic Host Configuration Protocol alias DHCP patří do rodiny protokolů, usnadňujících automatickou konfiguraci zařízení. Dynamicky přiřazuje IP adresy pracovním stanicím, které byly poprvé nebo opětovně připojeny do sítě. Služba vznikla z nutnosti neustále změny IP adresy, kterou musel uživatel na svém počítači manuálně nastavovat při přechodu z jedné sítě do druhé. V dnešním mobilním světě je výhodnější, když si zařízení nakonfiguruje svou IP adresu samo v duchu „tradice“ Plug-and-Play. Odpadá tak nutnost uživatele pamatovat si adresu svého počítače v té které síti. Dynamické přiřazování lze také chápat jako šetření adresním prostorem IP, jelikož počítač, který není připojen/zapnut neblokuje zbytečně jednu IP adresu [1].

DHCP využívá ke svému přenosu protokol UDP a v jedné zprávě může být uvedeno více informací, jako například přiřazená IP adresa, adresa výchozí brány či maska podsítě. V protokolu existují tři typy přiřazování adres:

- **Dynamická konfigurace** Počítači je adresa přidělena na určitou dobu a tu využívá, dokud tato doba nevyprší.
- **Ruční konfigurace** Poskytuje správci sítě možnost přiřadit konkrétní adresu danému zařízení.
- **Automatická konfigurace** Zařízení je přidělena trvalá adresa při jeho prvním připojení.

Celý proces přiřazování sestává z šesti stavů. Jako první proběhne inicializace, do které zařízení vstoupí připojením na síť. Začne vysílat zprávu DHCPDISCOVER pro zjištění všech DHCP serverů na síti a tím přejde do druhého stavu. V tomto stavu čeká na jednu nebo více zpráv DHCPOFFER, z nichž si musí jednu vybrat. Třetí stav nese název vyjednávání. Klient odešle vybranému serveru zprávu DHCPREQUEST, server přijme a odpoví zprávou DHCPACK. Ve čtvrtém stavu již klient začne používat přidělenou IP adresu a může normálně komunikovat. Pokud chce klient z nějakého důvodu ukončit zápůjčku adresy, informuje DHCP server zprávou DHCPRELEASE a danou adresu již není možné používat. Pátý stav souvisí s vypršením doby zápůjčky a klient požádá server o její prodloužení zprávou DHCPREQUEST s aktuálně používanou adresou. Klient čeká, dokud mu server neodpoví. Jestliže odpoví kladně, je možné si adresu ponechat, pokud záporně klient přechází do stavu 1. Neodpoví-li server v požadovaném čase, přechází do šestého stavu. Zde může zkoušet opakované dotazy, nebo pokud zjistí, že je server nedostupný zkusí obeslat další servery o přidělení adresy. Obdržením nové adresy přechází do 4. stavu, pokud žádnou adresu neobdrží, přechází do stavu 1 [1].

3.1.8 Simple Network Time Protocol

SNTP vznikl na základě protokolu NTP, využívaného v síti internet k časové synchronizaci síťových zařízení. SNTP implementuje část z něj a je využíván tam, kde není třeba vysoká přesnost a spolehlivost plného NTP. SNTP je zástupcem aplikační vrstvy modelu TCP/IP a k přenosu využívá UDP datagramy. Aktuální specifikace obsahuje celkem 4 verze, z nichž tou nejrozšířenější je verze 3. Verze 4 přinesla pouze modifikovanou interpretaci

hlavičky kvůli přizpůsobení protokolu IPv6 a byla navržena tak, aby zůstala čitelná i pro aplikace běžící ve verzi 3 [21]. Hlavičku verze 3 zobrazuje obrázek 3.8.

B	1			2	3	4
0 - 3	LI	VN	Mode	Stratum	Poll	Presision
4 - 7	Root Delay					
8 - 11	Root Dispersion					
12 - 15	Reference Identifier					
16 - 19	Reference Timestamp					
20 - 23						
24 - 27	Originate Timestamp					
28 - 31						
32 - 35	Receive Timestamp					
36 - 39						
40 - 43	Transmit Timestamp					
44 - 47						
…	Optional					

Obrázek 3.8: SNTP hlavička a její části.

Význam jednotlivých polí v hlavičce:

- **LI** Dvoubitový identifikátor značící zda má být vložena nebo odebrána přestupná sekunda v minulé minutě.
- **VN** Číslo verze protokolu reprezentované třemi bity.
- **Mode** Určuje režim, ve kterém zařízení pracuje (3 bity). Klientská část má číslo 3, serverová číslo 4.
- **Stratum** Toto políčko udává, z jakého zdroje doručený čas pochází. Číslo 0 značí zdroj nespecifikován, 1 primární reference z přesného zdroje času, 2 – 15 sekundární reference z protokolu NTP nebo SNTP. Ostatní čísla jsou rezervována.
- **Poll** Interval, v kterém je čas zjišťován, vypočten jako 2^{Poll}. Standardní obsah je 6 (64 sekund) až 10 (1024 sekund).
- **Precision** Jedná se o číslo se znaménkem určující požadovanou přesnost. Číslo je reprezentováno jako $2^{Precision}$ sekund.
- **Root Delay** Zpoždění, které vzniklo cestou k primární referenci.
- **Root Dispersion** Relativní chyba v hodnotě Root Delay vzniklá cestou k primární referenci.
- **Reference Identifier** Identifikace zdroje času. Jde o čtyř-bytový řetězec.
- **Reference Timestamp** Čas kdy byly lokální hodiny naposled pozměněny.
- **Originate Timestamp** Čas, který má klient.
- **Receive Timestamp** Čas, kdy došel požadavek na server.
- **Transmit Timestamp** Odpověď od serveru. Toto je aktuální čas.

Pokud chce zařízení v síti znát aktuální čas, stačí nastavit hodnoty **LI**, **Mode** a **VN**. Ostatní mohou být nulové. Vyplněnou hlavičku odeslat UDP protokolem na časový server

(port 123) a čekat odpověď od serveru na stejném portu. Jestliže chce zařízení dosáhnout vyšší přesnosti, může odeslat další požadavek s nastavenou hodnotou **Originate Timestamp** rovnu přijatému času v prvním paketu a následně dopočítat zpoždění cesty po přijetí odpovědi [21].

Formát, v jakém je čas protokolem STNP přenášen, udává čas v sekundách od 1.1 roku 1900 a 0:00:00 hodin. Klient si proto musí z této hodnoty aktuální datum a čas vypočítat. Musí dávat pozor zejména na počet přestupných roků od výše uvedeného data. Přestupným rokem nazveme každý, je-li dělitelný čtyřmi s vyjímkou staletí, která jsou přestupná jen tehdy, když jsou dělitelná čtyřmi sty. Rok 1900 tedy přestupný nebyl, zatímco rok 2000 ano [20].

3.1.9 HyperText Transfer Protocol

HTTP bývá využíván k výměně hypertextových dokumentů ve formátu HTML. Využívá spolehlivý přenos TCP s číslem portu nastaveným na 80 a jde zřejmě o nejčastěji používaný aplikační protokol v moderním světě Internetu. Protokol pracuje na principu „dotaz-odpověď" a data se přenáší ve formě textu. Klient bývá většinou webový prohlížeč požadující webovou stránku a k jejímu získání kontaktuje webový server. K označení konkrétního zdroje dat využívá HTTP lokátoru URL (uniform resource locator) [23].

Vlastní komunikaci inicializuje klient odesláním požadavku GET. Požadavek má tvar ve formátu `GET / HTTP/1.1`, kde „/" označuje adresu na serveru a „HTTP/1.1" verzi protokolu. V dotazu může být také zmíněn typ a verze prohlížeče, kódování, jazyk, který má uživatel nastaven a další parametry. Příkladem budiž dotaz na server s www adresou www.fit.vutbr.cz:

```
GET / HTTP/1.1
Host: www.fit.vutbr.cz
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:2.0.1)
Accept: text/html
Accept-Language: cs,en-us;q=0.7,en;q=0.3
```

Server dotaz přijme, zpracuje a odpoví ve tvaru `HTTP/1.1 XXX`, kde „XXX" označuje kód odpovědi a udává, jak vyhodnocení dotazu dopadlo. HTTP verze 1.1 rozlišuje následující skupiny [22]:

- **2xx** Označuje, že klientův dotaz byl úspěšně přijat a porozuměn. Příkladem je kód `200 OK`, který označuje, že požadovaná data jsou uvedena ve zbytku dotazu.
- **3xx** Značí, že je vyžadováno přesměrování a ve zbytku odpovědi je uvedena adresa, kam byla stránka přesunuta.
- **4xx** Tato skupina informuje, že při vyřizování vznikla chyba. Např. kód `400 Bad Request` značí chybu v dotazu klienta a kód `404 Not Found` má význam, že stránka nebyla nenalezena.
- **5xx** Chyba na straně serveru. Většinou značí, že je server přetížen nebo že konkrétní dotaz není podporován.

Odpověď z předchozího příkladu přijatá od serveru www.fit.vutbr.cz vypadá takto:

```
HTTP/1.1 200 OK
Server: Apache/1.3.42 Ben-SSL/1.59 (Unix)
Connection: Close
Content-Type: text/html; charset=iso-8859-2
Content-Language: cs

<html>
     stránka ve formátu html
</html>
```

Jak je vidět, skládá se tělo odpovědi z hlavičky následované HTML kódem stránky. Hlavičku od dat odděluje znak „\n“, tedy nový řádek. V odpovědi chybí důležitý identifikátor s názvem `Content-Length`. Podle ní klient pozná, kolik dat má očekávat a kdy spojení ukončit. Existuje ovšem možnost, kdy server nemusí položku Content-Length uvést a klient pak pozná konec přenosu podle ukončení spojení ze strany serveru. Server dá tento způsob ukončení najevo nastavením položky `Connection` na hodnotu `Close` (jako v přikladu odpovědi výše) a hned jakmile odešle všechna data, ukončí TCP spojení. [22].

3.1.10 Technika WakeOnLAN

WOL slouží, jak již název napovídá, k probuzení počítače nebo nějakého jiného zařízení přes počítačovou síť. K jejímu využití je důležité, aby hardware probouzeného počítače WOL podporoval. Technika spoléhá na neustálé napájení síťové karty, a jakmile karta rozpozná speciální paket, nazývaný „Magic Packet“, síťová karta počítač probudí. IP adresa není při odesílání zapotřebí, jelikož paket putuje na poslední vrstvě architektury TCP/IP. K určení cílového zařízení tak WakeOnLAN využívá MAC adresu síťové karty. Strukturu magického paketu tvoří broadcastová adresa, složena ze šesti bajtů a šestnáctkrát se opakující MAC adresy zařízení, které chceme probudit [17].

Výhody WOL jsou jasné - umožňuje zapnout zařízení na dálku přes existující infrastrukturu počítačové sítě a není k tomu třeba žádného dalšího zařízení. Mezi nevýhody patří to, že probouzení funguje pouze v lokálních sítích, jelikož routery směrující pakety dovnitř sítě většinou nepodporují vysílání na broadcastové adrese. Pokud ale takový router broadcast podporuje, je možné magic packet odeslat libovolným transportním protokolem. WakeOnLAN většinou využívá protokol UDP s nastaveným portem na hodnotu 7 nebo 9. Obecně lze ale paket odeslat na jakýkoliv jiný port. Další nevýhoda techniky WakeOnLAN spočívá ve skutečnosti, že na paket není odpovězeno a uživatel se tak nedozví, zda opravdu došlo k probuzení počítače. Nakonec bych zmínil nutnost napájení síťové karty, zde se ale její spotřeba pohybuje zanedbatelných hodnotách menších než 1 Watt.

4 Spínání elektrického proudu

Kapitola se zaměřuje především na spínání přívodu elektřiny spotřebičů pracujících s nízkým síťovým napětím 230 V. V první části jsou diskutovány druhy elektrických zátěží (spotřebičů) a na základě jejich povahy a chování rozděleny do třech skupin. Druhá část pak na vlastnosti elektrických prvků navazuje a popisuje, jakým způsobem lze takovéto prvky spínat. Dále pak popisuje několik spínacích prvků a zmiňuje jejich výhody a nevýhody.

4.1 Typy zátěží

Ke spínání nízkého napětí je třeba znát základní charakter zátěže, kterou budeme na svorky připojovat. Spínáním různých druhů zátěže totiž vystavujeme spínač odlišným průběhům napětí a proudu. Mezi nejčastější typy patří odporová (rezistivní), induktivní a kapacitní zátěž. V praxi jde většinou o jejich kombinaci, ale vždy převažuje některá složka.

4.1.1 Odporová

Asi nejběžnější typ zátěže. Nepříznivou vlastností u odporových zátěží je jejich ztrátový výkon v podobě zahřívání jelikož mění elektrické parametry součástky a spínač tak nespíná konstantní výkon. U některých zařízení se toho s výhodou využívá a např. žárovka či topné těleso tak představují běžného zástupce odporové zátěže. U žárovek pak vyvstává další nepříznivá vlastnost a to že jejich odpor za studeného stavu může být až 18 krát menší, než v rozžhaveném stavu. Při spínání žárovky je tak třeba počítat s prudkým nárůstem proudu v obvodu [14].

4.1.2 Induktivní

Jde o jakoukoliv zátěž mající ve své výbavě cívku. Příkladem jsou motory, transformátory a různé zdroje. Nebezpečí induktivní zátěže spočívá v tom, že při skokové změně proudu dojde v cívce k indukci napětí, které působí proti protékajícímu proudu. Energie nahromaděná v cívce se tak při rozepnutí spínače musí vyzářit do okolí a tím může dojít k jeho poškození. Přidáním paralelně připojené diody v závěrném směru může být tento jev eliminován. Při rozpojení spínače se pak v cívce indukuje napětí opačné polarity, pro něž je dioda propustná, dojde k přechodovému ději a proud postupně zanikne [13].

4.1.3 Kapacitní

Tento typ zátěže nebyl dříve obvyklý, ale dnes je na ni s obrovským rozšířením spínaných zdrojů potřeba brát zřetel. V minulosti nebyl tento jev pozorován, protože většina zátěží měla vysokou hodnotu účiníku 0,6 – 0,8, který byl navíc induktivního charakteru. S tím, jak se většina návrhářů snaží dosáhnout hodnotu účiníku blízkou 1, vykazuje většina dnešních produktů, zejména při nižším zatížení, hodnotu účiníku 0,9 – 0,95 kapacitního charakteru. Kapacitní zátěž má při připojení vysokou strmost náběhu proudu, který může být až 40 násobný oproti běžnému proudu v obvodu. Jako účinná technika pro potlačení této vlastnosti se využívá spínání v nule [12].

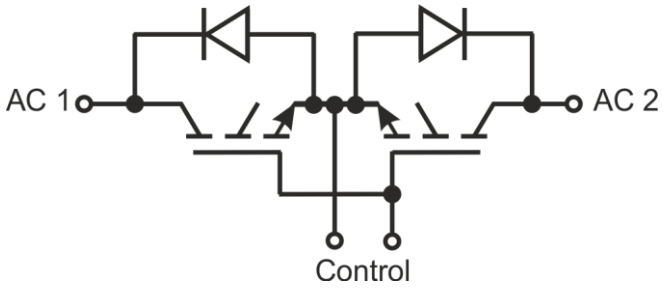
4.2 Spínací prvky

Existuje mnoho spínacích součástek, které lze rozdělit podle nespočetného množství parametrů. Základním rozdělením může být konstrukce, typ a velikost spínacího nebo spínaného napětí. Mým úkolem bude zaměřit se v této kapitole pouze na prvky schopné spínat střídavé napětí o velikosti 230 V s minimální proudovou propustností alespoň 8 A. Dále je nutné, aby se jejich ovládací napětí pohybovalo okolo 5 V, a aby jejich ovládání nevyžadovalo veliký příkon.

4.2.1 Tranzistor

Tranzistor je polovodičová součástka vynalezená v Bellových laboratořích v roce 1947. Za jeho objev byla jeho tvůrcům v roce 1956 udělena Nobelova cena za fyziku a šlo o převrat v oboru elektrotechniky. Hlavním přínosem byla míra miniaturizace, kterou polovodičové součástky poskytovaly, a tím jich mohlo být vloženo více na jednotku plochy. Tranzistor tvoří dva přechody PN a má vyvedeny tři elektrody, kterými je připojen do obvodu. Jsou děleny na unipolární MOSFET (N-FET a P-FET) a bipolární (NPN a PNP), podle způsobu jaké mají řízení [3].

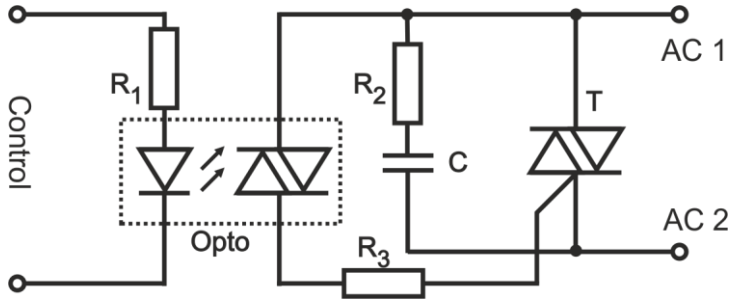
Ke spínání větších výkonů jsou primárně určeny tranzistory MOSFET nebo speciální druh – tranzistory IGBT. Tranzistory IGBT bývají využívány hlavně ke spínání stejnosměrného napětí ve výkonových obvodech vlaků, trolejbusů či tramvají, ale lze jimi spínat i střídavé stroje. Tranzistory MOSFET slouží ke spínání napětí v řádech stovek voltů. Snadno se s nimi ovládají i prvky s velkým nárazovým proudem. Jsou vhodné i ke spínání induktivních a kapacitních zátěží. Zatímco v případě kapacitních není třeba tranzistor nijak chránit, ke spínání induktivní zátěže je vhodné přiřadit k tranzistoru diodu [3]. Obrázek 4.1 ukazuje příklad obvodu vhodného ke spínání střídavého napětí. Porovnání tranzistorů MOSFET a IGBT, spolu s jejich dalšími vlastnostmi se zabývají [4][5][6].



Obrázek 4.1: Tranzistory MOSFET v zapojení pro spínání střídavého napětí.

4.2.2 Triak

Další z řady polovodičových prvků je triak. Jde o součástku s činností podobnou dvěma antiparalelně zapojeným tyristorům. Umožnuje spínání střídavého napětí o velikosti několika desítek kV a průchod proudu v řádu jednotek A. Jeho strukturu tvoří pět vrstev polovodičů v pořadí NPNPN a tři vývody – řídící elektroda a dvě anody. Obrázek 4.2 ukazuje jednoduchý spínací obvod s triakem [3].



Obrázek 4.2: Spínací obvod s triakem.

Vypínání triaků probíhá obdobně jako u tyristorů přirozenou komutací. Vzhledem k poměrně složité struktuře má triak oproti tyristoru menší odolnost proti rychlému nárůstu spínaného napětí a strmosti zmenšení proudu, ke kterému dochází při vypínání. Při odporové zátěži je proud procházející triakem ve fázi s napětím a při vypnutí prochází napětí nulou zároveň s proudem. Při indukční zátěži, kdy dochází k fázovému zpoždění proudu za napětím, vzniká nebezpečí samozapnutí. Dále může průchodem proudu nulou, kdy je napětí maximální, dojít ke zničení triaku, protože při jeho vypnutí v okamžiku nulového proudu dojde k prudké změně. Proto se jeví jako vhodné přidat k triaku při ovládání induktivní zátěže paralelně ochranný RC člen. Kondenzátor v něm pak snižuje strmost náběžné hrany spínaného napětí [3].

4.2.3 Relé

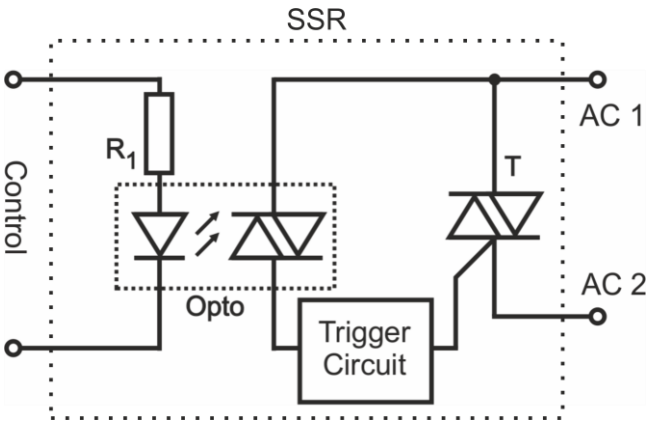
Před příchodem polovodičových součástek patřilo mezi nejčastěji využívané spínací prvky bezpochyby relé. Jde o elektro-mechanickou součástku využívající elektromagnet. Tvoří jej cívka s jádrem z magneticky měkké oceli, v jejichž blízkosti je kotva, taktéž z magneticky měkkého materiálu. Kotva ovlivňuje pružný kontakt, který plní funkci spínače. Jakmile začne elektromagnetem procházet proud, dojde k přitáhnutí kotvy, sepnutí kontaktů a uzavření elektrického obvodu. Spínací napětí, může být daleko menší než spínané napětí, navíc jsou od sebe galvanicky odděleny. Existuje několik druhů relé a dělí se dle mechanického principu (spínací, rozpínací, přepínací), typu a velikosti spouštěcího napětí, spínaných napětí/proudů a dalších různých vlastností.

Hlavní nevýhoda využití relé spočívá v jeho mechanické konstrukci. Součástky, které mají pohyblivé kontakty, bývají více náchylné na poruchy. Zvláště pak pokud jsou spínány větší výkony nebo induktivní zátěž bez ochranné diody, může dojít k rychlému opotřebování kontaktů a životnost relé rapidně klesá. Další vlastnost, se kterou musí konstruktér při využití relé počítat, je neustálý odběr elektrické energie, protože cívkou při sepnutí prochází proud, který nemusí být v některých případech zanedbatelný. Naproti tomu je relé využíváno v kritických aplikacích a plní spolehlivější funkci než odpovídající spínač konstruovaný z polovodičových prvků. Příkladem by mohla posloužit řídící logika v atomových elektrárnách, kde na prvky obvodu působí radiace a relé je zde odolnější. Ozáření polovodiče totiž způsobí jeho sepnutí [18].

4.2.4 Polovodičové relé

Solid State Relay (dále jen SSR) poskytuje stejnou funkci jako obyčejné relé s tím rozdílem, že ke spínání nejsou použity žádné mechanické součásti. Jako spínací prvek je využita některá z polovodičových součástek (triak, tyristor, tranzistor, jejich kombinace) většinou

doplněna o galvanické oddělení obou okruhů v podobě optočlenu nebo transformátoru. SSR lze použít jak pro stejnosměrné, tak i střídavé napětí. Co se spínání různých druhů zátěže týče, jsou SSR jedny z nejvíce odolných součástek. Všechna SSR jsou z výroby navržena ke spínání induktivních i kapacitních zátěží [14]. Jde v podstatě o integrovaný obvod a uživatel může vybírat z mnoha parametrů a provedení. Náhradní obvod SSR zobrazuje obrázek 4.3.



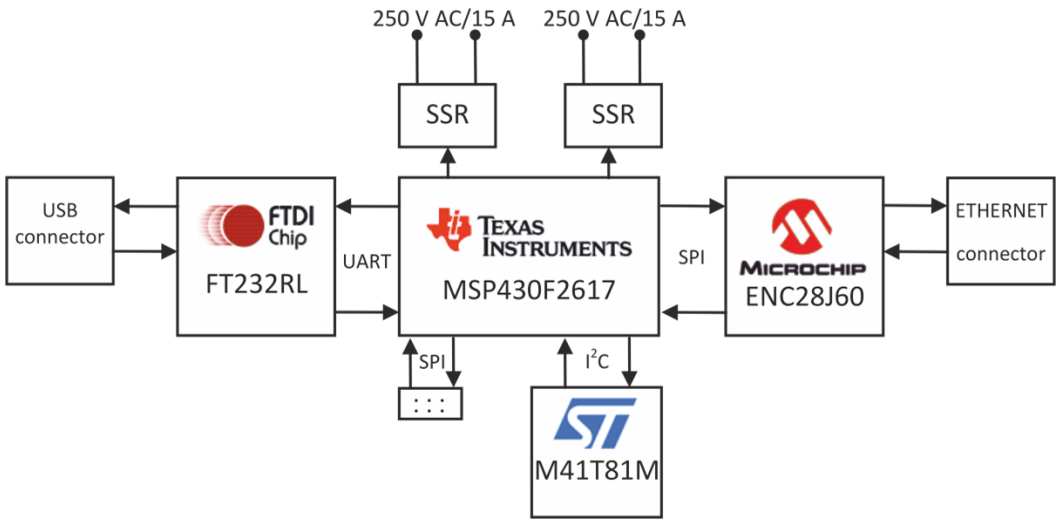
Obrázek 4.3: Náhradní obvod SSR.

Hlavní výhody oproti mechanickému relé jsou rychlejší sepnutí, žádné jiskření, menší spotřeba, menší elektrický šum, menší velikost, více odolné v určitém operačním prostředí (vlhkost, vibrace) a v neposlední řadě nulový hluk. Nevýhody plynou z polovodičových součástek, z nichž bývá dané SSR složeno. SSR většinou používá jako hlavní spínací prvek triak, doplněný o podpůrné obvody. Mezi ně patří startovací obvod triaku, různé ochranné obvody či obvod spínání v nule. Hlavní nevýhodou pak je zahřívání součástky při větších spínaných výkonech a nutnost jejího chlazení.

5 WakeOnLAN systém

Ke vzdálenému ovládání počítačů a spínání přívodu elektřiny jsem navrhl vestavěný systém, využívající mikroprocesor jako hlavní řídící člen. Ten byl zvolen ze stáje rodiny MSP430, kterou nabízí americký výrobce Texas Instruments. Konkrétně jde o typ F2617. K připojení systému do počítačové sítě slouží integrovaný obvod ENC28J60 firmy Microchip, realizující základní vrstvu rozhraní Ethernet. S mikroprocesorem jej spojuje sběrnice SPI. Sběrnice UART připojuje další integrovaný obvod s názvem FT232RL, jde o převodník na rozhraní USB a vyrábí jej firma FTDI. Poslední obvod pochází od výrobce ST Microelectronics a realizuje obvod reálného času s možností programovatelného alarmu. Ke spínání přívodu elektřiny byly zvoleny dva prvky SSR. Systém počítá s budoucím rozšířením v podobě dalších spínacích prvků nebo jiných přídavných obvodů a vyvádí dva konektory. Blokové schéma systému zobrazuje obrázek 5.1.

Propojovací schéma a deska plošných spojů, na níž jsou umístěny výše uvedené prvky, byly vytvořeny v návrhovém prostředí Eagle. Kompletní schéma zapojení jsem ve formátu A3 umístil do přílohy A. Navrženou desku plošných spojů pak do přílohy B a osazovací schéma a popis propojek do přílohy C.



Obrázek 5.1: Blokové schéma systému.

5.1 Mikrokontrolér – MSP430F2617

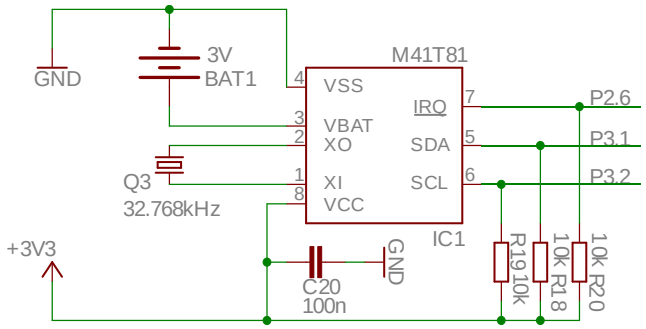
Mikrokontrolér řady MSP430 dodává firma Texas Instruments a jde o výkonný šestnáctibitový procesor s architekturou typu RISC vhodný k univerzálnímu využití. Jeho operační napětí se pohybuje v řádu 1,8 – 3,6 V a hlavním zaměřením cílí na nízkopříkonové aplikace. Spotřebuje pouze 330 µA při frekvenci hodin 1 MHz a napětí 2,2 V. Na čipu obsahuje podpůrné obvody jako integrovaný 12 bitový A/D převodník, dvojitý 12 bitový D/A převodník, tříkanálové DMA, dva šestnáctibitové čítače, komparátorem a hlavně připojení až čtyř univerzálních sériových linek. Konkrétně jde o sběrnice SPI, I^2C, UART a IrDA kodér/dekodér. Umožňuje také programovat mikrokontrolér přímo na čipu, nebo využívat interní paměť pro ukládání stálých dat, jelikož uchová informace i při výpadku napětí. Obrovskou výhodou je přítomnost Bootstrap Loaderu, který umožňuje programovat mikrokontrolér bez nutnosti

externího programátoru, postačí pouze sériová linka. Model F2617 disponuje 96 kB + 256 B interní Flash pamětí a 8 kB RAM [10].

5.2 Obvod reálného času

K udržení stálého času na desce byl zvolen integrovaný obvod M41T81 firmy ST microelectronics. Lze jej napájet jak centrálním napětím 2,7 V – 5,5 V, tak záložní baterií. Při výpadku hlavního napětí pak dochází k automatickému přepnutí do režimu udržování času s udávaným proudem 1 µA. Disponuje rozhraním I^2C s udávanou maximální rychlostí 400 kbit/s, programovatelným alarmem a dokáže generovat signál přerušení. Obsahuje čítače pro sekundy, minuty, hodiny, dny, měsíce, roky a bit signalizace století. Zvolena je varianta s pouzdrem SO-8, která neobsahuje vlastní krystal. Bylo tedy nutné dodat externí o frekvenci 32,768 kHz [9].

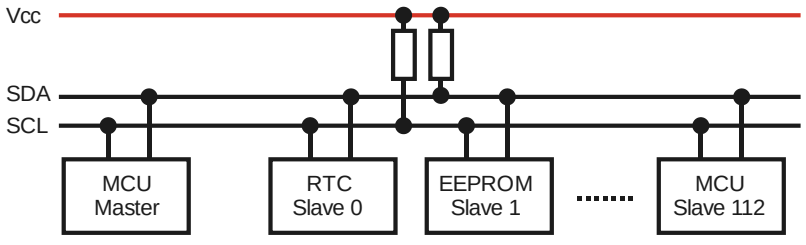
Zapojení obvodu a jeho připojení na porty mikrokontroléru ukazuje obrázek 5.2. Odpory R18, R19 a R20 slouží jako pull-up rezistory a jsou nutné pro správnou funkci sběrnice I^2C. C20 zastává roli blokovacího kondenzátoru.



Obrázek 5.2: Zapojení RTC M41T81M6E.

5.2.1 I^2C

Synchronní sběrnice I^2C firmy Philips (Inter-Integrated Circuit), někdy nazývaná „2-wire interface“, bývá často využívána k obsluze integrovaných obvodů, které nekladou velké nároky na rychlost komunikace. Využívá dva vodiče s otevřeným kolektorem SDA a SCL. Ke každému z nich musí být připojen pull-up rezistor, jinak není na sběrnici definován stav logické „1“. Referenční specifikace adresuje připojená zařízení sedmibitovou adresou, což po odečtení 16 rezervovaných adres znamená, že na jednu sběrnici může být připojeno až 112 zařízení, viz obrázek 5.3. Pokud by byl pro někoho tento počet malý, existuje rozšíření na 10 bitovou adresaci. Sběrnice pracuje na principu „master/slave“ [24].



Obrázek 5.3: Topologie I^2C.

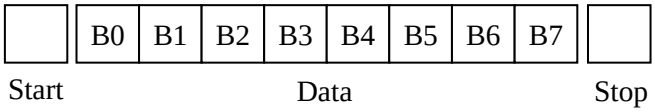
Vodič SCL slouží k přenosu hodinového signálu, data jsou pak odesílána vodičem SDA. Jelikož datový vodič využívají oba komunikující, musí existovat mechanizmus jak jej sdílet. I^2C řeší tento problém zavedením sekvencí Start, Stop, Ack a NoAck. Jde o kombinaci logických úrovní hodinového signálu a datového vodiče. Na počátku nastaví master oba vodiče na sekvenci Start, čímž dá najevo, že chce komunikovat. V dalším kroku vystaví na datový vodič 7bitovou adresu zařízení slave, následovanou R/W bitem, který značí, jakou operaci chce master realizovat, buď to bude čtení, nebo zápis. Následuje jeden datový bajt, udávající adresu v rámci slave prvku. Nyní proudí přes sběrnici sekvence čtecích/zapisovacích operací, každá potvrzená signálem Ack. V případě, že operace byla dokončena nebo nejsou k dispozici data, dojde k vygenerování sekvence NoAck. K ukončení a uvolnění sběrnice nakonec pošle master značku Stop. Sekvence Start a Stop generuje vždy master, slave pouze odpovídá signály Ack a NoAck.

5.3 USB subsystém

Připojení k počítači je požné za pomoci převodníku FT232RL firmy FTDI. Jde o USB – UART převodník a po připojení k počítači realizuje jednu sériovou linku. Obvod pracuje při napájecím rozsahu 3,3 – 5,25 V, tedy včetně napájení přímo z USB rozhraní. Integruje vlastní převodník 3,3 V a lze jej využít k napájení ostatních prvků. Disponuje integrovanou 1024 bitovou EEPROM pamětí, 256 B vstupním a 128 B výstupním bufferem. Není zapotřebí externího krystalu, protože je generování signálu hodin přítomno přímo na čipu. Zajišťuje plnou kompatibilitu s USB 2.0 [11].

5.3.1 UART

UART, celým jménem Universal Asynchronous Receiver/Transmitter, je využíván zejména ke komunikaci počítače s periferními obvody. Toho bývá s výhodou využito a drtivá většina dnešních mikrokontrolérů tento komunikační způsob podporuje. Standard pracuje na principu rozhraní RS-232 (rozdíl je v podstatě pouze v úrovni signálů) a je složen ze dvou vodičů. Jeden slouží pro příjem dat a nese název RXD, druhý pak k jejich odesílání a bývá nazýván TXD. Data jsou po vodičích odesílána jako sekvence bitů a jelikož jsou oba na sobě nezávislé, lze komunikovat buď v režimu full-duplex nebo half-duplex. Komunikace musí mít určenou konfiguraci jinak by došlo ke špatné interpretaci dat. Skládá se z několika bitů, které obalují data tak, aby příjemce poznal jejich začátek a konec, popřípadě detekoval možnou chybu. Začátek přenosu dává odesílatel najevo Start bitem, po němž následují datové bity, kterých může být 5 až 8. Volitelně lze odeslat paritní bit, tím je zaručena detekce chyb v přenášených datech, ovšem jak je všeobecně známo, parita dokáže detekovat pouze lichý počet chyb. Konec přenosu odesílatel signalizuje odesláním jednoho nebo dvou Stop [25]. Posloupnost bitů při využívanou v implementaci ukazuje obrázek 5.4.



Obrázek 5.4: Formát přenosu dat na sběrnici UART

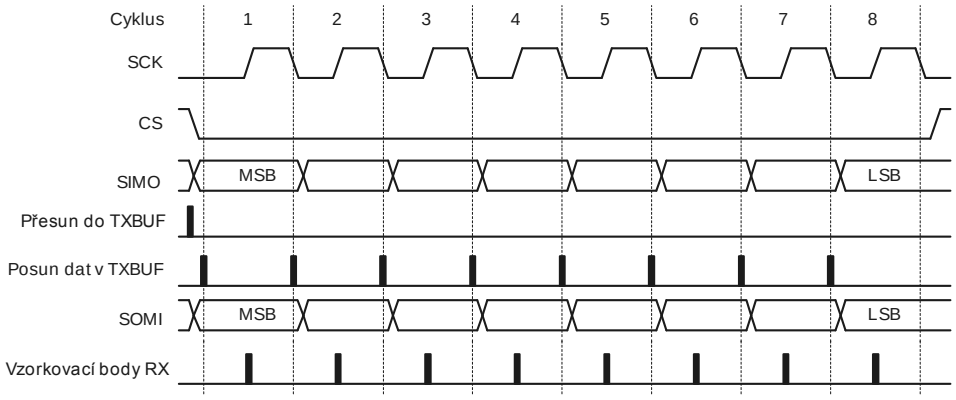
5.4 Rozhraní Ethernet

Společnost Microchip obsahuje ve svém portfoliu integrovaný obvod ENC28J60, který plně integruje rozhraní Ethernet a to jak na fyzické, tak částečně linkové vrstvě. Řadič podporuje jeden 10Base-T polo nebo plně duplexní port s automatickou detekcí polarity a je kompatibilní s 10/100/1000Base-T sítěmi. S mikrokontrolérem je spojen synchronní sběrnicí SPI o komunikační frekvenci až 20 MHz. Má programovatelné automatické opakování odesílání při kolizi na médiu, hardwarově asistované generování kontrolního součtu a zahazování chybových paketů. MAC vrstva podporuje všechny typy odesílání a přijímání paketů (unicast, broadcast, multicast) a lze vygenerovat přerušení po přijmutí Magického paketu techniky WOL. Další vlastnosti zařízení zahrnují až 6 zdrojů přerušení, operační napětí 3,1 až 3,6 V a nutnost připojení externích hodin v podobě krystalu o kmitočtu 25 MHz [8].

ENC28J60 automaticky generuje pouze pole Preamble a Start of Frame Delimiter, také zarovnává rámec na požadovanou délku a nakonec spočte CRC. Z toho vyplývá, že zbytek rámce musí vytvořit mikrokontrolér a nahrát do paměti řadiče k odeslání.

5.4.1 SPI

Rozhraní SPI bylo vyvinuto ke komunikaci dvou a více integrovaných obvodů na velmi malé vzdálenosti (většinou na stejné desce). Jedná se o synchronní, plně duplexní rozhraní vyvinuté firmou Motorola, kde zařízení komunikují v režimu „master/slave“. Ke komunikaci slouží čtyři vodiče. První přenáší hodinový signál a bývá nazýván SCK. Druhý a třetí slouží k přenosu dat směrem k a od zařízení, jejich jména jsou SIMO a SOMI. Poslední signál, nazývaný CS, pak určuje, se kterým prvkem typu slave chce master komunikovat. Průběh signálů při komunikaci zachycuje obrázek 5.5. Zakresluje také vzorkování vstupního signálu (RX) a přesun a posouvání dat ve výstupním bufferu (TXBUF) tak, jak tyto operace provádí mikrokontrolér. SPI má tři možnosti nastavení. První je délka slova v bitech, druhá nastavení fáze a polarity hodinového signálu (na kterou hranu/úroveň se vzorkuje) a třetí který bit má být poslán nejdříve (MSB/LSB) [26].

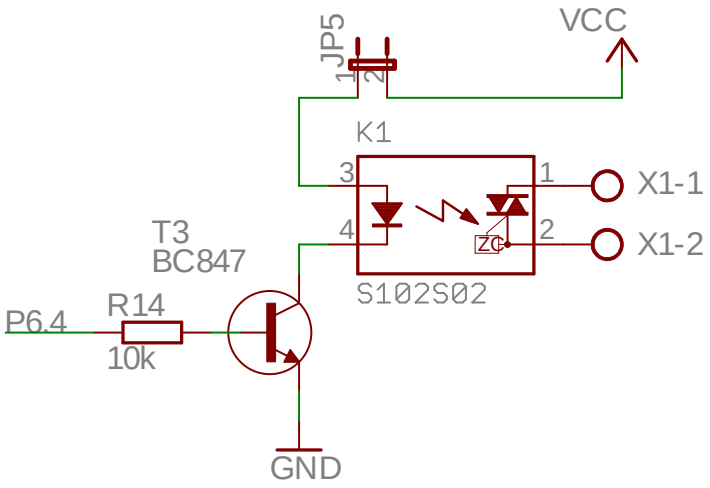


Obrázek 5.5: Časování na sběrnici SPI.

5.5 Spínací subsystém

Jako spínací prvek jsem zvolil Solid State Relé, jelikož jde o tzv. „All-in-One“ součástku s dobrou spolehlivostí a optimalizacemi pro všechny typy zátěže. Obrázek 5.6 ukazuje zapojení spínacího obvodu. Použiji solid state relé KSD215AC3 firmy COSMO. SSR

dokáže na výstupu spínat až 250~ V / 15 A zatímco je řízeno vstupním napětím 5 – 12 V DC s maximálním proudem 5 – 35 mA. Umožňuje spínání v nule, obsahuje optočlen, led diodu značící sepnutí a galvanicky odděluje obě části až do průrazného napětí 4000 V [7]. Řízení mikrokontrolérem probíhá za pomoci tranzistoru BC847, jenž je dimenzován pro proudy až 300 mA. K manuálnímu rozpojení nebo měření napětí a proudu je do obvodu vložen jumper.



Obrázek 5.6: Spínací obvod s prvkem SSR.

5.6 Rozšiřující sběrnice

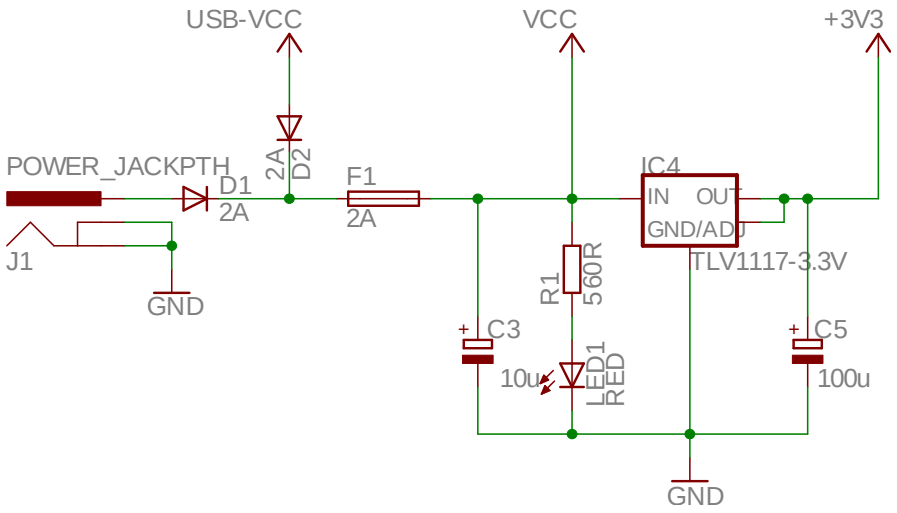
Počet spínacích prvků je možné rozšířit vytvořením přídavných modulů a jejich připojením na vyvedené sběrnice. Deska obsahuje celkem dva konektory, z nichž jeden má vyvedenu sběrnici SPI a na druhém je 16 portů mikroprocesoru. Oba porty mají vyhrazeny piny pro hlavní a 3,3 V větev spolu s dvěma piny GND. Lze tak snadno připojit další spínací prvky nebo další moduly. Díky vyvedení sběrnice SPI může být připojen i SPI expandér, za jehož pomoci pak bude možné adresovat větší množství spínacích prvků.

5.7 Napájení

Při návrhu bylo počítáno se dvěma možnými způsoby napájení. První je napájení prostřednictvím rozhraní USB, druhý způsob umožňuje připojení adaptéru konektorem power jack 2.1 mm. Oba zdroje oddělují diody D1 a D2, jinak by mohlo dojít ke zkratu a případnému poškození nejen komponent na desce, ale i napájecích zdrojů v případě kdy jsou oba zdroje napájení připojeny současně. Protože jsou na desce obvody a prvky, které vyžadují různé operační napětí, bylo nutné použít napěťový regulátor. Zvolil jsem obvod TLV1117 firmy Texas Instruments, konkrétně typ 33CDCY snižující vstupní napětí v rozsahu 4,7 – 15 V na hodnotu výstupního napětí 3,3 V s maximálním výstupním proudem 0,8 A. Provozní teploty regulátoru jsou od 0 °C po 125 °C. 3,3 V větví je napájen mikroprocesor, Ethernet kontrolér a obvod reálného času. Obvod FT232RL je napájen pouze sběrnicí USB a pokud není připojena, není aktivní. Navržené schéma je zobrazeno na obrázku 5.7.

Jelikož deska obsahuje spínací prvky SSR, u kterých je udáváno minimální napětí 5 V, je právě toto napětí operačním minimem. Operačním maximem je pak 15 voltů, což je limit napěťového regulátoru. Je ovšem nutné počítat s úbytkem napětí na určitých prvcích

a parametry napájecího zdroje. Cesty hlavní napájecí větve jsem navrhl na maximální velikost proudu 2 A. Větev 3,3 V omezuje regulátor maximálním proudem 0,8 A.



Obrázek 5.7: Schéma napájecího obvodu.

6 Návrh a implementace firmware

Výsledný program tvoří řada knihoven, z nichž nejdůležitější budou popsány v této kapitole. Ke komunikaci přes počítačovou síť byla využita volně šiřitelná knihovna z projektu FITkit [27], kde bylo třeba některé části drobně upravit pro potřeby této práce. Za její pomoci byl implementován webový server, realizována probouzecí technika WakeOnLAN a také vytvořena knihovna protokolu SNTP.

Při implementaci firmware jsem využil možnosti programovat v jazyce C. K překladu byl použit balík MSPGCC a obsahuje upravenou verzi známého překladače GCC pro rodinu procesorů MSP430. Balík dále obsahuje řadu dalších podpůrných nástrojů, jako například linker asembleru, debugger či programovací nástroj bsl. MSPGCC lze stáhnout z adresy mspgcc.sourceforge.net a to jak pro systémy Windows, tak Linux a BSD. K překladu všech zdrojových souborů slouží dávkový soubor make.bat. Naprogramování mikroprocesoru lze provést po překladu souborem load.bat, který využívá nástroj bsl.

6.1 Knihovna komunikace s periferiemi

Aby mohl mikroprocesor využívat k němu připojené obvody, bylo nutné vytvořit knihovnu realizující komunikaci mezi nimi. Jelikož jsou v návrhu využity 3 komunikační sběrnice, byla pro každou z nich vytvořena implementace obsluhy a umístěna do zdrojového souboru USCI.c. Přidružený hlavičkový soubor USCI.h pak obsahuje definice nejdůležitějších konstant. Implementace využívá vnitřních obvodů mikroprocesoru s názvem USCI (Universal Serial Communication Interface). Přístup ke konfiguraci těchto obvodů je jednotný a řídí se těmito body:

- Nastavení resetovacího bitu UCSWRST v registru UCxxCTL1*.
- Inicializace registrů UCxxCTL0*, UCxxCTL1* a dalších na požadované parametry přenosu (bit UCSWRST musí zůstat nastavený).
- Konfigurace příslušných portů.
- Nulování resetovacího bitu UCSWRST registru UCxxCTL1*.
- Případné povolení přerušení.

*poznámka: místo xx je doplněno jméno konkrétního USCI rozhraní

6.1.1 Komunikace prostřednictvím rozhraní UART

Toto rozhraní posloužilo v rámci implementace hlavně jako dobrý ladící prostředek, jelikož lze skrze něj komunikovat s osobním počítačem. Pro možnost využití v budoucích rozšířeních byla jeho implementace v souboru ponechána.

Implementace využívá rozhraní mikrokontroléru s názvem UCA0. K inicializaci rozhraní slouží funkce *void UART_init()*, která nastaví rozhraní na konfiguraci s 8 data bity, jedním stop bitem a žádnou paritou. Rychlost je zvolena nastavením dělících registrů UCA0BR0 (spodní část) a UCA0BR1 (horní část) na hodnotu 0x0080. To odpovídá 115200 Baudům, při frekvenci hodin SMCLK rovné 14,7456 MHz.

K použití jsou připraveny vysílací funkce *void UART_writeLongInt (unsigned long int i)*, *void UART_writeInt(int i)* a *void UART_writeChar(unsigned char c)*, kde každá vyšle odpovídající datový typ. Odeslání řetězce znaků realizuje funkce *void UART_writeString(const*

*char *str, unsigned int length).* Příjem může být realizován buď za pomoci přerušení nebo využitím funkce *char UART_getChar(unsigned char c),* u které se musí počítat s implicitním blokováním.

6.1.2 SPI a komunikace s ENC28J60

Mikroprocesor komunikuje s řadičem Ethernetu skrze rozhraní UCB1, jemuž jsou přidruženy porty P5.1 – P5.3. Port P4.7 sloužící jako signál CS není součástí rozhraní a musí tak být ovládán programově. Mikroprocesor ovládá i resetovací vodič řadiče, připojený na portu P4.6. Parametry komunikace jsou nastaveny na přenosovou rychlost 14,7456 Mb/s (dělič kmitočtu SMCLK je nastaven na 1). MCU vystupuje v roli master a ENC28J60 jako slave. Přenáší se 8 datových bitů, nejdříve MSB a polarita hodin nastavena na neaktivní v logické hodnotě 0. Funkce obsluhující SPI jsou psány tak, aby poskytly univerzální přístup ke komunikaci s jakýkoliv zařízením.

unsigned char SPI_write_wait_read(unsigned char data) Funkce nejprve odešle bajt na sběrnici SPI, poté počká, dokud odeslání neskončí a očekává jeden bajt jako odpověď. Jde o základní komunikační funkci.

void SPI_write_wait(unsigned char data) Vyšle jeden bajt na sběrnici a čeká na konec přenosu.

void SPI_set_CS(unsigned char value) Dle zadané hodnoty nastaví signál CS. Signál v logické úrovni „0“ aktivuje řadič Ethernetu, logická „1“ pak řadič opět deaktivuje (pouze pro přenos).

6.1.3 I^2C a obvod reálného času

Obvod reálného času je vodičem SCL a SDA připojen na porty P3.2 a P3.1. Tyto porty obsluhuje v režimu I^2C komunikační rozhraní s názvem UCB0. Přerušení generované při hlášení alarmu bylo vyvedeno na port P2.6, musí tak být nastaven do režimu sledování signálu. Následující komunikační rutiny implementují inicializaci a komunikaci na sběrnici I^2C:

- **void I2C_init()** Inicializuje port P2.6 do režimu generování přerušení. Dále nastaví reakci na sestupnou hranu (díky pull-up rezistoru je klidový stav signálu v logické úrovni „1“).
- **void I2C_receiveInit()** Tato rutina je volána před každým příjmem dat. Nastaví rozhraní do režimu master, rychlost komunikace 105 kilobitů za sekundu a do registru UCB0I2CSA zapíše adresu obvodu reálného času 0xD0. Zároveň povolí přerušení při příjmu nastavením bitu UCB0RXIE v registru IE2.
- **void I2C_transmitInit()** Podobně jako v předchozí funkci. Nastaví rychlost 105 kb/s, zapíše adresu zařízení a nakonec povolí přerušení při volném odesílacím bufferu nastavením bitu UCB0TXIE v registru IE2.
- **void I2C_receive(unsigned char byteCount, unsigned char *field)** Funkce pro příjem. Jako parametr bere počet bajtů, které mají být přijaty a ukazatel na pole bajtů do kterého je uloží. Její činnost je rozdělena do dvou větví podle toho, jestli se odesílá jeden nebo více bajtů. V obou větvích nejprve na sběrnici odešle Start podmínku s R/W bitem

nastaveným na čtení. Při odesílání více bajtů funkce končí, v případě odesílání jednoho bajtu čeká, dokud mikroprocesor nevynuluje bit UCTXSTT v registru UCB0CTL1. Vzápětí generuje podmínku Stop a funkce končí. Data od RTC budou v obou případech přijata v přerušovací rutině USCIAB0TX_ISR().

- **void I2C_transmit(unsigned char byteCount, unsigned char *field)** Činnost odesílací funkce spočívá v nastavení globálních proměnných *transmit_field* a *byteCtr* na ukazatel a hodnotu v parametru. Nakonec vygeneruje podmínku start s bitem R/W v logické hodnotě 1 (zápis). Veškeré odesílání je pak realizováno v přerušení USCIAB0TX_ISR().
- **unsigned char I2C_notReady()** Vrací stav bitu UCBBUSY v registru UCB0STAT. Je-li návratová hodnota „1“, sběrnice je zaneprázdněna, v opačném případě je volná. Funkce slouží při testování, že byla všechna data odeslána/přijata.
- **unsigned char I2C_slavePresent(unsigned char addr)** Otestuje, zda je zařízení s adresou předanou v parametru připojeno na sběrnici.
- **void I2C_writeBytes(…)** Zapíše data na konkrétní adresu v RTC. Bere jako parametr pole s hodnotami, počet zapisovaných hodnot a adresu, kam data přijdou. Pro každou adresu volá funkci *I2C_transmit()* následovanou čekáním na volný stav sběrnice.
- **void I2C_readBytes(…)** Očekává v parametrech ukazatel na pole bajtů, počet čtených hodnot a adresu. Nejprve odešle, z jaké adresy chce číst a čeká, dokud není sběrnice uvolněna. Následně čte data funkcí *I2C_receive()* a ukládá je do pole, na nějž ukazuje pointer předaný v parametrech. Opět čeká na uvolnění sběrnice.
- **Přerušovací rutina USCIAB0TX_ISR()** Realizuje jak odesílání, tak příjem dat. Výběr mezi nimi uskuteční na základě bitu UCB0RXIFG v registru IE2. Je-li nastaven na „1“, obsluhuje čtení, jinak obsluhuje zápis. V případě čtení zkopíruje přijatý bajt do příjímacího pole, následně inkrementuje ukazatel a sníží proměnnou *byteCtr* o 1. Takto je přerušovací rutina volána dokola, dokud není hodnota proměnné *byteCtr* rovna nule. Je-li *byteCtr* rovno 0, odešle se ukončující podmínka Stop a obvod reálného času ví, že má přestat vysílat. Obsluha odesílání pracuje podobně. Na sběrnici posílá data, dokud není *byteCtr* rovno 0. Pak odešle podmínku Stop jako signalizaci, že je přenos u konce.
- **Přerušovací rutina USCIAB0RX_ISR()** K vyvolání dojde při příjmu dat. Zkontroluje, jestli byla přijata podmínka NoAck a v případě že ano, vysílá se podmínka Stop. Tím rutina končí.

Výše uvedené funkce jsou univerzální a lze je využít ke komunikaci s jakýmkoliv zařízením na sběrnici I^2C. Každé zařízení má ovšem jiný přístup ke správě dat a proto byla v souboru USCI.c implementována další sada funkcí zaměřených přímo na obvod M41T81. K povolení nebo zakázání čítání času jsou využívány funkce *RTC_run()* a *RTC_stop()*. K zápisu času byla vytvořena rutina *RTC_setTime(unsigned char *time)*, která zapíše do obvodu čas uložený na adrese ukazatele time. Funkce *RTC_readTime(unsigned char *time)* přečte aktuální čas. Obě funkce přepokládají pole o velikosti 8 bajtů. Funkce *RTC_alarmEnable()* a *RTC_alarmDisable()* slouží k povolení/zakázání alarmu. Nastavení alarmu na konkrétní čas realizuje funkce *RTC_setAlarm(unsigned char *alarm)*, kde předaná adresa alarm ukazuje na pole s časem o velikosti 5 bajtů. Čtení aktuální doby alarmu realizuje *RTC_readAlarm (unsigned char *alarm)* do pole velikosti 5. Jako poslední jmenuji rutinu *RTC_isAlarmSet()*, která v návratové hodnotě vrací informaci o nastaveném/nenastaveném alarmu. Adresy

jednotlivých registrů a jejich bitů jsou definovány v souboru USCI.h. Na stejném místě je uložena i adresa obvodu reálného času.

6.2 Obsluha ethernetového řadiče

K realizaci potřebných protokolů nutných ke komunikaci přes počítačovou síť jsem využil část knihovny nacházející se v repositáři FITkit. Konkrétně jde o knihovnu pro komunikaci s řadičem ENC28J60 a implementaci nejdůležitějších protokolů architektury TCP/IP. Knihovna obsahuje implementaci protokolů Ethernet, IP, UDP, TCP, ARP a DHCP. Dále obsahuje implementaci inicializace bufferu řadiče ethernetu a odesílání/příjímání řadičem ethernetu. Výhodou knihovny je centralizovaný způsob jejího používání. Stačí totiž využívat jednu skupinu odesílacích, přijímacích a nastavovacích funkcí a o nutné doplnění důležitých hlaviček jednotlivých vrstev se knihovna postará sama. Všechny zdrojové soubory pocházející z této knihovny začínají názvem „enc28j60“. Knihovna je koncipována jako open-source a její zdrojové soubory jsem stáhnul ze stránky projektu [27].

6.2.1 Ovládání řadiče ethernetu

Obvod ENC28J60 obsahuje vlastní instrukční sadu, na jejímž základě jsou nad ním prováděny veškeré operace. Sada obsahuje sedm instrukcí, odesílaných jako jeden bajt. První tři bity označují, o jakou instrukci jde, zbylých pět pak argument instrukce. Za instrukcí může následovat jeden nebo více datových bajtů.

Jméno a zkratka instrukce	Bajt 0		Bajt 1 a další
	Kód	Argument	
Read Control Register (RCR)	0 0 0	a a a a a	N/A
Read Buffer Memory (RBM)	0 0 1	1 1 0 1 0	N/A
Write Control Register (WCR)	0 1 0	a a a a a	d d d d d d d d
Write Buffer Memory (WBM)	0 1 1	1 1 0 1 0	d d d d d d d d
Bit Field Set (BFS)	1 0 0	a a a a a	d d d d d d d d
Bit Field Clear (BFC)	1 0 1	a a a a a	d d d d d d d d
System Reset Command (SRC)	1 1 1	1 1 1 1 1	N/A

Tabulka 6.1: Instukční sada obvodu ENC28J60. a – adresa registru, d – data.

Všechny instrukce k sobě mají přidruženu jednu nebo více funkcí, umístěných v souboru enc28j60_spi.c. Jejich realizace staví na univerzálních funkcích popsaných v kapitole 6.1.2 SPI. Adresy všech důležitých registrů a jejich bity jsou definovány jako konstanty v hlavičkovém souboru enc28j60_spi.h.

Instrukce BFC slouží k vynulování bitů v registru na adrese předané v argumentu. Vynulují se pouze ty bity, odpovídající pozicím logické „1“ v dalším odeslaném bajtu. Instrukci implementuje funkce **void bf_clear(unsigned char addr, unsigned char mask).** Opakem je instrukce BFS, která bity mající na pozici logickou „1“ nastaví. K tomu slouží funkce **void bf_set(unsigned char addr, unsigned char mask).** Instrukci SRC implementuje funkce **void enc_reset()** a vyvolá tak softwarový reset. Ostatní instrukce souvisí s pamětí řadiče a jsou popsány v následující podkapitole.

6.2.2 Paměťový prostor řadiče

Paměťový prostor se v obvodu ENC28J60 dělí na tři skupiny. První je společný přijímací a odesílací buffer o velikosti 8 kB. Druhá jsou kontrolní registry, které lze adresovat přímo instrukcemi RCR a WRC. Poslední skupinou jsou tzv. PHY registry. Velikosti pro přijímací a odesílací část v bufferu zařízení jsou programovatelné a přístup k bufferu obsluhují pouze instrukce RBM a WBM. Kontrolní registry slouží ke konfiguraci a kontrole celého řadiče, jejich změna se projeví okamžitě. PHY registry nastavují modul fyzického rozhraní a přístup k nim je realizován prostřednictvím speciálních kontrolních registrů a rozhraní nazývaného MIIM (Media Independent Interface Management). Změna těchto registrů se provede až po interním překopírování z kontrolních registrů. Popis celé paměťové struktury lze nalézt v datasheetu k řadiči ENC28J60 [8]. Implementace všech operací souvisejících se čtením a zápisem obsahuje zdrojový soubor enc28j60_spi.c.

Pro adresaci kontrolních registrů slouží v řadiči celkem 4 banky. Přepínání mezi nimi je realizováno zápisem čísla banky do nejnižších dvou bitů registru ECON1. Tento a čtyři další registry jsou společné pro všechny banky, mají přiřazeny adresy v rozsahu 0x1B – 0x1F. Ke změně banky slouží funkce *select_bank(unsigned char bank)*. K zápisu do, a čtení z registrů slouží funkce:

- **void write_reg8(unsigned char addr, unsigned char data),**
- **void write_reg16(unsigned char addr, unsigned int data),**
- **unsigned char read_reg8(unsigned char addr, unsigned char read_dummy),**
- **unsigned int read_reg16(unsigned char addr, unsigned char read_dummy).**

Nastavení fyzického modulu obsluhují registry umístěné v bankách 3 a 4, jejichž název začíná vždy na „MI“. Jestliže chce uživatel číst obsah těchto registrů, musí nejprve zapsat jejich adresu do registru MIREGADR. Nastavit bit MIIRD v registru MICMD, čímž dojde k nastavení bitu BUSY v registru MISTAT. Až bude bit BUSY opět vynulován, jsou data k dispozici v registrech MIRDL (dolních 8 bitů) a MIRDH (horních 8 bitů). Zápis má postup opačný. Nejprve je zapsána adresa registru do MIREGADR, poté musí být zapsáno spodních 8 bitů do registru MIRDL a nakonec zápis horních 8 bitů do registru MIRDH. Po zapsání horního slova dojde automaticky k nastavení bitu BUSY v registru MISTAT a až má hodnotu 0, došlo k překopírování dat. Obsluhu registrů fyzického modulu realizují funkce:

- **int read_phy_reg16(unsigned char addr)** a
- **void write_phy_reg16(unsigned char addr, unsigned int data).**

K nastavení velikosti a umístění přijímacího bufferu slouží dvojice párových registrů ERXSTH:ERXSTL a ERXNDH:ERXNDL. Tyto registry mohou být upravovány pouze v případě, že není povolen příjem dat ze sítě. V opačném případě by mohlo dojít ke ztrátě dat, nebo k nežádoucímu přemazání ostatních částí bufferu. Párové registry ERXWRPTH: ERXWRPTL označují místo, kam budou zapisována přijatá data. Tyto registry slouží pouze ke čtení a lze pomocí nich snadno zjistit kolik volného místa v přijímacím bufferu zbývá. Čtení z přijímacího bufferu realizuje instrukce RBM implementovaná ve funkcích *char spi_getc()*, int spi_read(void* ptr, unsigned int length) a *int spi_getline(void* ptr, unsigned int max_length).* První přečte z bufferu jeden bajt, druhá uloží do předaného pole požadovaný počet bajtů a třetí

čte, dokud nenarazí na znak ‚\n' nebo do maximálního počtu znaků specifikovaného parametrem *max_length.*

Všechno volné místo, které není nastaveno jako příjímací buffer zůstává přiděleno odesílací části. Stačí vyplnit buffer rámcem, který má být odeslán a nastavit jeho začátek a konec do registrů ETXST a ETXND. Při odeslání řadič automaticky spočte CRC a ethernetový rámec vyšle na síť. K zápisu do odesílacího bufferu využívám funkce *void spi_putc(unsigned char c), void spi_write(void* ptr, unsigned int length)* a *void spi_fill(unsigned char value, unsigned int length)*. Všechny jsou založeny na volání instrukce WBM a zapisují jeden nebo více bajtů.

6.2.3 Síťové protokoly

V této podkapitole budou popsány některé důležité procedury, které jsem využil při realizaci webového serveru, WakeOnLAN techniky a protokolu SNTP. Jsou rozděleny dle souborů, ve kterých jsou umístěny a obecně platí, co soubor to jedna implementace protokolu. Jednotlivé implementace jsou navzájem provázány a přesně kopírují vrstvovou strukturu TCP/IP. Protokol vyšší vrstvy vždy opatří odesílaný paket svou částí a předá ji vrstvě nižší. Jakmile je dosažen konec řetězce dojde k odeslání paketu.

enc28j60_eth.c Realizace linkové vrstvy a protokolu Ethernet. Jednou z důležitých funkcí je *void ENC28J60_idle(),* která by měla být volána z hlavní smyčky. Kontroluje, zda obvod ENC28J60 přijal data a jestliže ano, volá další funkci s prototypem *void eth_recv()*. Za její pomoci jsou data přečtena, zjistí se, jaký protokol vyšší vrstvy rámec přenáší a podle toho zavolá určenou obslužnou rutinu. Při odeslání volají vyšší vrstvy funkci *char eth_send(unsigned char* dest_addr,unsigned int type)*. Ta do bufferu přidá hlavičku ethernetu, čímž vznikne rámec, který odešle. Nakonec funkce *inline char set_mac(unsigned char* mac_ptr)* nastavující mac adresu fyzické vrstvy zařízení.

enc28j60_arp.c Kompletní implementace síťového protokolu ARP. Procedury *void arp_recv()* a *char arp_send(unsigned long dest_ip)* přijmou a odešlou paket z nebo do ethernetové části implementace. Aby systém nemusel neustále překládat stejné IP adresy na fyzické MAC adresy, obsahuje implementace ARP tabulku. V ní jsou uchovány poslední přeložené adresy a operace nad ní realizují funkce *void arp_table_clear(), void arp_table_add(int ip,unsigned char* mac), unsigned char* arp_table_find(int ip)*.

enc28j60_ip.c Mezi základní funkce protokolu IP patří opět ty přijímací void ip_recv(void) a odesílací *char ip_send(unsigned long dest_addr,unsigned char protocol)*. Dále pak procedura *unsigned int count_checksum(void* pointer,int count),* která vypočítá 16-bitový kontrolní součet nad hlavičkou protokolu. Další rutiny slouží k nastavení nebo čtení adres ip, masky, brány a výchozího DNS serveru.

enc28j60_icmp.c Díky této implementaci odpovídá knihovna na dotazy známých programů ping či tracert. Obsahuje pouze dvě funkce *void icmp_recv(struct ip_h* ip_header)* a *char icmp_send(...)*, které doplňují registrační a deregistrační rutiny *void icmp_bind(void* handler)* a *void icmp_unbind()*. Registrační proto, jelikož při volání přiřadí/registruje ukazatel na funkci, která bude volána v případě příjmu paketů protokolu ICMP. Deregistrační rutina ukazatel opět odregistruje. Přijímací a odesílací funkce přímo volají služby protokolu IP.

enc28j60_udp.c Součástí zdrojového kódu protokolu UDP jsou opět dvě funkce ke čtení a modifikaci odesílacího bufferu. Jde o *void udp_recv(struct ip_h* ip_header)* a *char udp_send(unsigned int dest_port,unsigned int src_port, unsigned long ip)*. Nezbytná kalkulace kontrolního součtu je přítomna v rutině *unsigned int udp_checksum(unsigned long dest_ip,int length)*, registrace ukazatele na obslužnou funkci pak řeší *unsigned int udp_bind(unsigned int port,void* handler)* a *void udp_unbind(unsigned int port)*.

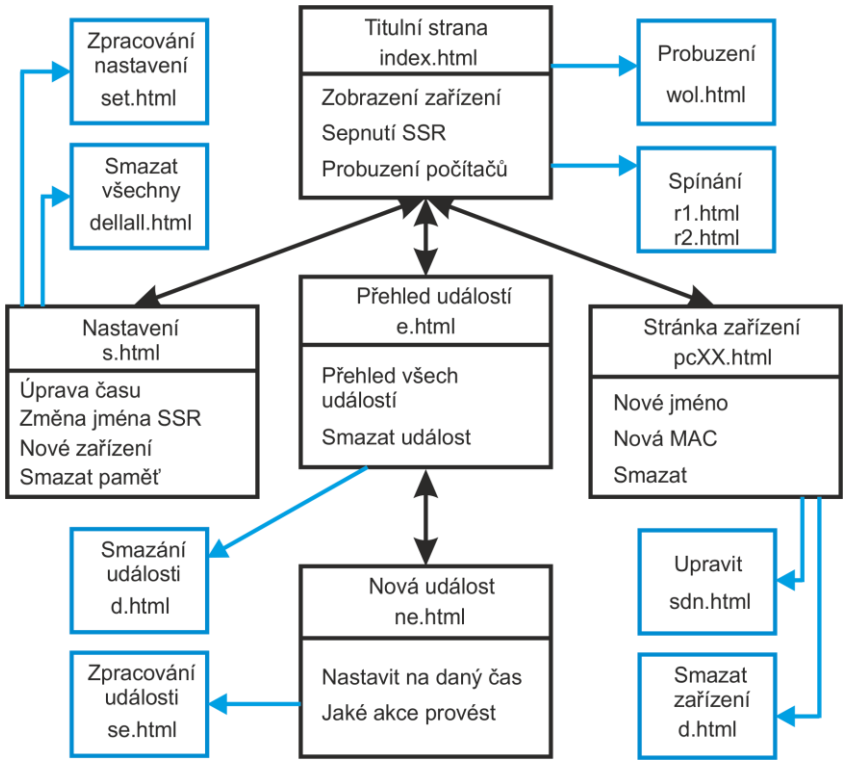
enc28j60_tcp.c a enc28j60_tcp_fsm.c Tyto dva zdrojové soubory v sobě plně integrují veškeré operace spojené se spolehlivým protokolem TCP. V prvním souboru jsou funkce sloužící k odeslání a přijímání TCP paketů a funkce využívané při spojení a odpojení od vzdáleného počítače, a registrační/deregistrační funkce. Nakonec jsou v souboru obslužné operace nastavující stavový automat implementovaný v druhém souboru. Stavový automat slouží ke kontrole stavu linky, udržuje spojení a hlavně opětovně odesílá nepotvrzené pakety a potvrzuje přijaté. Jeho práce se dělí podle toho, zda je nějaký paket odesílán nebo přijímán. Obsluha stavového automatu musí být volána buď v hlavní smyčce, nebo alespoň jedenkrát za krátký časový úsek.

enc28j60_dhcp.c Protokol automatické konfigurace IP adresy alias DHCP. K jeho inicializaci slouží funkce *char dhcp_init()* a k resetování funkce *char dhcp_reset()*. Protokol si registruje ukazatel na obslužnou rutinu protokolu UDP, ve které je implementován automat, mající na starosti zpracování všech DHCP dotazů. K nastartování konfiguračního procesu slouží funkce *char dhcp_send_discover()* a k odesílání DHCP žádostí funkce *char dhcp_send_request()*. Funkce *void dhcp_timer()* obsahuje další automat, který zjišťuje v jakém stavu se DHCP žádost nachází. Funkce by měla být volána z hlavní smyčky.

enc28j60_spi_rx.c a enc28j60_spi_tx.c Implementace v těchto souborech nesouvisí ani tak se síťovými protokoly jako spíše s přípravou bufferu (a práci s ním) pro každý z nich. Kvůli jejich vzájemnému provázání ji zařazuji do této kapitoly. Funkce *char tx_init(unsigned char protocol)*, inicializuje buffer pro konkrétní protokol. Tato funkce by měla být volána při odesílání dat vždy jako první. Funkce *unsigned int tx_write(void* ptr, unsigned int length*) a makro *tx_write_str* slouží pro zápis do bufferu. Po inicializaci a naplnění bufferu výše uvedenými funkcemi musí být buffer uzavřen. O to se stará funkce *inline void tx_close()*. Nyní může být naplněný rámec odeslán operací *char tx_send()*. K příjmu pak slouží funkce *unsigned int rx_read(void* ptr, int length)*, která z bufferu přečte zadaný počet dat. Implementace obsahuje spoustu důležitých rutin, ty ale nejsou využity.

6.3 Webový server

Jedním ze základních požadavků bylo, aby mohl uživatel zařízení vzdáleně ovládat. K tomu se výborně hodí prezentace zasíláním hypertextových dokumentů, jelikož kombinují dostatečné uživatelské rozhraní a možnost interakce uživatele se serverem pomocí webových formulářů. Veškeré operace, které výsledný systém poskytuje, jsou centralizovány právě do implementace webového serveru. Ten tak neplní jen funkci odesílání dat, ale i spínání výkonových prvků, probouzení počítačů, ukládání dat do paměti mikrokontroléru a plánování událostí. Zdrojový kód je umístěn v souboru web.c, definice důležitých konstant pak v souboru web.h.



Obrázek 6.1: Činnosti jednotlivých webových stránek a jejich návaznost.

Server poskytuje několik webových stránek a ještě než bude vysvětlen jejich princip, bude vhodné seznámit čtenáře s jejich funkcí a vzájemným propojením. Vše zachycuje diagram na obrázku 6.1. Černé bloky představují pětici hlavních webových stránek. Černé šipky webové odkazy, takže lze mezi jednotlivými stránkami přecházet tam a zpět. Modré bloky a šipky znamenají přechod na stránky, kde jsou data z vyplněných formulářů zpracována. Podle zpracovaných dat jsou pak realizovány dodatečné operace. Data z formulářů jsou do webového serveru odesílána metodou GET (vložení dat z vyplněných formulářů do adresy stránky). Generování stránek probíhá za pomoci funkcí *tx_write_str()* a *tx_write()*, které zapisují do odesílacího bufferu obvodu ENC28J60.

6.3.1 Důležité globální proměnné

Všechny části webového rozhraní pracují se společnými proměnnými. Jsou to v prvé řadě seznam uložených zařízení deklarovaný jako *struct device devices[MAX_DEVICES]* a seznam naplánovaných událostí *struct event events[MAX_EVENTS]*. Oba seznamy jsou realizovány jako pole struktur. K uchování záznamu o jednom zařízení využívám strukturu o dvou prvcích deklarovanou jako:

```
struct device{
    char name[DEVICE_NAME_LENGTH];
    unsigned char mac[MAC_LEN];
};
```

Položka *name* obsahuje jméno zařízení omezené na 4 znaky, pole *mac* pak ukládá MAC adresu zařízení a jeho délka je 6 znaků. Celková velikost struktury je tedy 10 bajtů.

Pro uchování záznamu jedné události pak slouží struktura s velikostí 9 bajtů:

```
struct event{
    unsigned char ssr:4        //vypinani/zapinani SSR
    unsigned char mo:4;        //mesic
    unsigned char d;           //den
    unsigned char h;           //hodin
    unsigned char m;           //minut
    unsigned char s;           //sekund
    unsigned long wol;         //az 32 zarizeni
};
```

Proměnná *ssr* zabírá horní čtyři bity prvního bajtu a uchovává informace o spínání SSR ve formátu daném tabulkou 6.2.

nastavený bit číslo	akce
0	SSR2 Vypnout
1	SSR2 Zapnout
2	SSR1 Vypnout
3	SSR1 Zapnout

Tabulka 6.2: Význam bitů v proměnné SSR.

Bity 3 a 2 slouží k nastavení SSR1 a bity 1 a 0 ovládají SSR2. Nikdy nemohou být nastaveny oba bity patřící jednomu SSR zároveň. Kombinace bitů „00“ znamená, že stav SSR nebude měněn. Člen struktury s názvem *wol* udává, které stroje mají být probuzeny. Každý k-tý bit nastavený na logickou „1“ značí, že je třeba zaslat magický paket k-tému počítači. Maximální počet probouzených počítačů je tak omezen na 32 (unsigned long má 32 bitů). Ostatní prvky struktury určují čas, kdy je třeba událost provést.

Webový server obsahuje několik globálních proměnných, které lze číst z ostatních částí programu. Jde o čítače *device_counter* a *event_counter* uchovávající kolik zařízení nebo událostí má systém uložen v paměti a zároveň tak ukazují do seznamu, kam přijde nový záznam. Další proměnnou je *elements_sent*, udávající počet odeslaných elementů (zařízení nebo událostí) a využívá se hlavně ve funkcích, jejichž odesílání je rozděleno na více paketů. Server tak podle této proměnné ví, co již bylo odesláno a co ne. Globální proměnná *alarm* udává, že došlo k přerušení obvodem reálného času a systém tak ví, že má obsloužit naplánovanou událost. Poslední z důležitých proměnných uchovávají jména obou relé a jsou to pole *relay1_name* a *relay2_name* s omezenou velikostí na 4 znaky.

6.3.2 Společná část funkcí webového rozhraní

Odesílání všech stran webového rozhraní začíná stejným způsobem. Do bufferu se vyplní záhlaví „HTTP/1.0 200 OK“. Pouze pokud došlo k chybě, generuje se kód „HTTP/1.0 404 ERR“, spolu s chybovou hláškou uloženou dále ve zprávě. V hlavičce jsou dále vyplněny položky „Conection: Close“, „Server: WOL System“ a „Content-Type: text/html“. Následuje identifikátor délky odesílané zprávy „Content-Length“. Jelikož většina stránek obsahuje dynamicky se měnící obsah, generovaný na základě prvků a událostí uložených v paměti, musí být délka odesílaných dat vypočtena předem. To by bez normalizovaných délek všech proměnných nebylo možné, proto jsou všechny omezeny na pevné velikosti definované v souboru web.h. Finta zmíněná v kapitole 3.1.9 HTTP, že položka typu Content-Length nemusí být přítomna, pokud je nastaveno Connection na typ Close, nelze použít. Klient totiž pozná

konec spojení pouze tehdy, je-li po odeslání všech dat ukončeno TCP spojení. Tedy že dojde k úplnému odpojení uživatele od serveru. Při další interakci s webovým rozhraním musí být spojení TCP opět navázáno, což znamená častou inicializaci TCP automatu a právě tato operace stojí na mikrokontroléru ohromné množství procesorového času. Rychlost reakce rozhraní tak rapidně klesá. Při testování s uzavíráním spojení se reakce serveru pohybovala v řádu jednotek sekund. Při uvedené položce délky zprávy a TCP spojení ponechaném jako otevřené klesla reakce serveru v průměru pod 1 sekundu. Ukončení spojení tak závisí pouze na správném odeslání celkové délky dat.

Odesílání většiny stránek probíhá odesláním několika paketů, protože odesílané informace omezuje ethernetový rámec velikostí 1500 bajtů. Odečtením všech hlaviček protokolů dostaneme maximální velikost rovnu 1380 znaků. Knihovna enc28j60 neumožnuje jednoduše poslat více paketů za sebou, protože po odeslání každého paketu musí proběhnout několika kroky automat TCP spojení. Pouhým skládáním paketů za sebe dojde vždy k zablokování systému. Tato situace byla vyřešena odesláním prvního paketu a zbylé části jsou odeslány až opětovným voláním funkce z hlavní smyčky, kdy je automat TCP obsloužen.

Nakonec po naplnění každého paketu volá každá funkce rutinu *tx_close()*, čímž uzavře buffer a následně paket odešle rutinou *tcp_send()*.

6.3.3 Titulní strana

Odesílání hlavní stránky uživatelského rozhraní proběhne voláním funkce *void write_index_page(unsigned char packet)*. Odeslání musí být rozděleno do třech částí a každá část vždy odešle kousek výsledné webové stránky. První část zapisuje do bufferu HTML kód značící začátek stránky. Dále zapíše aktuální čas uložený v zařízení a dva odkazy na stránku „Nastavení“ a „Výpis událostí“. Následuje zjištění stavu obou SSR přečtením bitů 3 a 4 v registru P6OUT a zjištěný stav se zapíše do bufferu. Oba spínací prvky mají na hlavní straně tlačítko, kterým lze ovládat jejich stav, do bufferu je tedy zapsán jejich HTML kód. Tím je ukončeno naplnění prvního paketu a může být odeslán. Nyní musí být obsloužen TCP automat a až se tak stane musí být funkce *write_index_page()* volána podruhé. Budou v ní totiž na hlavní stránku vypsány všechny zařízení v síti, které má systém v paměti a které lze probudit technikou WakeOnLAN. Na stránku se vypisují pouze jejich jména, doplněná o zatrhávací políčko „checkbox“ a odkaz na stránku s informacemi o jednom síťovém zařízení. Do jednoho paketu lze uložit HTML kód až šestnácti zařízení, znamená to tedy, že po jeho naplnění musí být paket odeslán. Funkce musí být opětovně volána, dokud nejsou na hlavní stránku odeslána všechna zařízení a jakmile se tak stane, bude funkce volána naposledy. V posledním volání je na stránku přidáno tlačítko spolu s ukončením HTML kódu stránky, paket je odeslán a odesílání stránky dokončeno.

K hlavní straně jsou přidruženy stránky k probuzení zvolených zařízení nebo sepnutí spínacích prvků. Jde o stránky r1.html, r2.html a wol.html volané při kliknutí na tlačítka ve stránce. Stránky r1.html a r2.html zajišťují ovládání spínacích prvků a ovládání probíhá voláním funkce *void set_ssr(unsigned char ssr)*. Po provedení akce server zobrazí stránku s výsledkem. Stránka wol.html zajišťuje odesílání magického maketu všem vybraným počítačům. K jejímu vyvolání si uživatel vybere zařízení, které chce probudit a klikne na tlačítko „Probudit“. Dojde k přesměrování na webovou stránku a všechna zařízení, které uživatel vybral budou uvedena v adrese.Adresa bude mít následující formát:

```
wol.html?00=on&01=on&02=on&end=1
```

Jde o příjem formulářovou metodou GET a čísla uvedená v adrese stránky odpovídají číslům jednotlivých prvků v poli *devices*. Zařízení, které nemají být probuzena, se do adresy nepromítnou. Aby server poznal, že došel ve zpracování adresy nakonec, ukončuje adresu identifikátor „&end=1“, který byl do formuláře přidán při odesílání stránky. V průběhu zpracování bude každému zařízení odeslán magický paket generovaný funkcí *void wake_up_devices(char *buffer, unsigned char offset)*. Nakonec je uživateli vypsána stránka s informací jak probouzení dopadlo.

6.3.4 Stránka nastavení

Na této stránce probíhá generování stránky k nastavení celého systému WakeOnLAN voláním funkce *void write_setup_page(unsigned char packet)***.** Součástí stránky jsou dva formuláře. První obsahuje 10 textových polí a umožňuje uživateli nastavit názvy obou spínacích prvků, přidat nové zařízení a změnit aktuální čas systému. Při kliknutí na tlačítko odeslat dojde k přesměrování na stránku set.hmtl, kde budou informace zpracovány. Druhý formulář obsahuje pouze jedno tlačítko a plní funkci smazání všech zařízení. Aby nedošlo k omylu, obsahuje toto tlačítko jednoduchý skript, který se uživatele zeptá, zda chce všechna uložená zařízení opravdu smazat. Stránka je generována ve dvou paketech.

Zpracování dat realizuje funkce *void setup_system(char *buffer, unsigned char offset)*, kde slouží parametr *buffer* jako ukazatel na přijatý buffer a parametr *offset* udává první prvek, od kterého má být čteno. Formát přijatých dat dostane funkce ve tvaru:

```
set.html?S1=R1&S2=R2&N=PC&M=AABBCCDDEEFF& … &y=2011&end=1
```

Celý řetězec musí být rozdělen na jednotlivé části. Identifikátory „S1“ a „S2“ určují nové názvy obou SSR. „N“ a „M“ značí nové jméno zařízení a jeho MAC adresu, zbytek polí udává nový čas. Pokud některé pole uživatel nezadá, nedojde ke změně hodnoty. Nové zařízení bude do seznamu *devices* přidáno pouze v případě, že nebylo dosáhnuto maximálního limitu uložených zařízení. Přijatý čas se zapíše do obvodu reálného času a podle něj se zjistí nebližší uložená událost. Ke zjišťování slouží funkce *get_first_event()*, které zohledňuje při výběru aktuální čas. To znamená, že bude vybrána nejmenší následující událost od aktuálního času. Události plánované před aktuálním časem budou provedeny až za další rok.

Klikne-li uživatel na tlačítko „smazat všechny zařízení“ dojde k přesměrování na stránku delall.html a tím vyvolání její obslužné funkce s názvem *void delete_all_devices()*. Funkce vynuluje čítač všech zařízení a nebude tak možné k nim přistupovat.

6.3.5 Stránka přehledu o naplánovaných událostech

Další stránka s dynamicky se měnícím obsahem rozdělená do více paketů. Uživateli je nutné zobrazit všechny naplánované události. Velikost každé z nich může být různá, protože každá událost může mít nastaven jiný počet spínaných a probouzených zařízení. V rámci vkládání jedné události na webovou stránku dojde prvně k vložení jejího naplánovaného času. Potom je třeba projít záznam po záznamu v poli *events* a na základě stavu proměnných *ssr* a *wol* vypsat všechny prvky, které jsou v události naplánovány. U každé události je pak přítomný webový odkaz, který přesměruje uživatele na stránku smazání události. Velikost jednoho záznamu o naplánované události se pohybuje v rozmezí od 100 až po zhruba 400 znaků (včetně prvků jazyka html). To znamená, že v jednom paketu lze odeslat maximálně 4 záznamy

o události. Výpis stránky s událostmi realizuje funkce *void write_event_page(unsigned char packet)*.

Po kliknutí na odkaz smazání události, bude vyvolána funkce *void delete_event(char *buffer, unsigned char offset)*, která událost smaže a pokud není seznam událostí prázdný, volá funkci *get_first_event()* k naplánování nejbližší události.

6.3.6 Stránka „Nová událost“

Na této stránce může uživatel naplánovat novou událost. Stránka je zobrazena voláním funkce *void write_new_event(unsigned char packet)* a obsahuje formulář s pěti textovými poli k zadání času události. Ve zbytku stránky uživatel nastaví, které SSR chce sepnout a která zařízení probudit. Nastavení SSR realizuje HTML prvek „radio button“ a k výběru zařízení slouží „checkbox“. Uživatel může vybrat jen ty zařízení, které má systém uloženy v paměti. Formulář uzavírá odesílací tlačítko s přesměrováním na stránku zpracování se.html.

Obslužná funkce *void set_new_event(char *buffer, unsigned char offset)* zpracuje veškeré předané hodnoty v adrese ve formátu:

```
se.html?h=11&m=11&s=11&d=22&mo=22&&S1=n&S2=n&01=on&02=on&end=1
```

První identifikátory udávají čas události. „S1“, „S2“ a „01“ označují které SSR sepnout a které zařízení v síti probudit. Zpracováním dat funkce vytvoří novou událost, uloží ji do seznamu *events* a nakonec projde všechny události a vybere tu nejbližší, kterou uloží do obvodu reálného času. Stránka kontroluje chybně zadaný čas, ale nekontroluje čas v návaznosti na aktuální čas. Znamená to, že pokud si uživatel naplánuje událost na čas, který již byl, provede se až za rok. Jestliže je seznam událostí plný, uživateli bude zobrazena stránka s chybovou zprávou.

6.3.7 Stránka zařízení

Stránka věnovaná jednomu konkrétnímu zařízení. Odeslání provádí funkce *void write_device_page(char *id_str)* a stránka slouží jak k informativnímu, tak k nastavovacímu charakteru. Informativní proto, poněvadž je to jediné místo, kde uživatel vidí, jakou MAC adresu zařízení nastavil. Nastavovací z toho důvodu, že lze jednoduše změnit jak název zařízení název, tak jeho adresa. Prostřednictvím této stránky lze zařízení také smazat.

Zpracování informací mají na starosti funkce *void set_device_name(char *buffer, unsigned char offset)* a *void delete_device(char *buffer, unsigned char offset)*. První upraví aktuální záznam na předané hodnoty. Druhá pak zařízení smaže. Při mazání musí být dané zařízení vymazáno i ze všech naplánovaných událostí, proto jsou postupně procházeny a existuje-li v některé záznam, bude odebrán.

6.3.8 Doplňující funkce

Webový server poskytuje ostatním částem programu jednotné rozhraní, pomocí kterého si mohou zjistit jeho stav, popřípadě nastavit některou z jeho vlastností.

void event_interrupt() Obsluha přerušení generované obvodem reálného času. Je umístěno ve webovém serveru, jelikož jedině on má přístup k polím se zařízeními a událostmi. Funkce zjistí, jestli je naplánovaná nějaká událost a pokud ano zkontroluje, zda souhlasí její čas s aktuálním.

Při kladném výsledku obslouží všechny nastavené akce. Nakonec událost smaže ze seznamu a naplánuje další, pokud existuje.

void load_data_from_memory() Nahraje data uložená v paměti Flash do paměti RAM. Zde jsou využívány funkce knihovny práce s pamětí popsané v další kapitole.

unsigned char get_device_counter() Vrátí hodnotu globální proměnné *device_counter*.

unsigned char get_event_counter() Vrátí hodnotu globální proměnné *event_counter*.

unsigned char get_elements_sent() Vrátí hodnotu globální proměnné *elements_sent*.

void clear_alarm() Vynuluje globální proměnnou *alarm*.

void set_alarm() Nastaví globální proměnnou *alarm*.

unsigned char is_alarm() Vrací stav globální proměnné *alarm*.

6.4 Knihovna pro práci s pamětí

Mikroprocesor pracuje převážně s volatilní pamětí RAM, kde má uloženy veškeré uživatelem nastavené vlastnosti. To nemusí být v některých případech vhodné, zvláště pak při ztrátě napájení by to znamenalo, že budou veškerá data ztracena. Uživatel by pak po každém spuštění musel znovu ukládat zařízení v síti a naplánované události. Tato situace se jevila jako nežádoucí a vznikla tak knihovna pro práci s vnitřní pamětí mikrokontroléru. Její zdrojové kódy jsou uloženy v souboru memory.c.

Procesor MSP430 disponuje sadou registrů, které slouží k ovládání paměťového řadiče, umístěného na čipu. V řadě případů to pak znamená, že pro ukládání trvalých dat nemusí být dodatečně přítomna externí paměť EEPROM. Využití vnitřní nevolatilní paměti s sebou ale nese řadu obtížností.

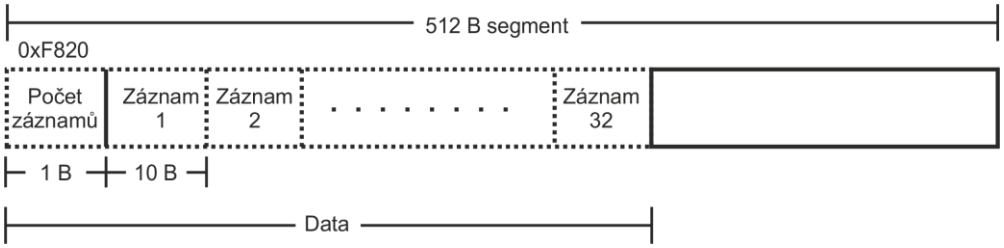
Obrázek 6.2: Struktura paměti mikroprocesoru MSP430.

Obrázek 6.2 zachycuje organizaci paměti v mikroprocesoru MSP430F2617. K trvalému ukládání lze využít pouze blok hlavní paměti a tzv. informační paměť. Obě tyto části jsou rozděleny do segmentů, kde jde v případě informační oblasti o segmenty velikosti 64 bajtů, hlavní paměť pak dělí segmenty velikosti 512 bajtů. Flash paměť umožňuje zapisovat po bitech, bajtech nebo celých slovech, ale mazat lze pouze celý segment najednou. Informační paměť by k uložení veškerých dat nestačila, proto byla zvolena k ukládání paměť hlavní.

K zápisu do paměti slouží funkce *void write_devices_to_memory(struct device *read_ptr, unsigned char count)* a *void write_events_to_memory(struct event *read_ptr, unsigned char count)*. Podle druhu funkce obdrží každá z nich v parametru pole se všemi záznamy o uložených zařízeních/událostech, spolu s velikostí tohoto pole. Jejich postup při nahrávání do paměti se shoduje s postupem udávaným v uživatelské příručce a je následující:

- Inicializuje hodinový signál řadiče nastavením registru FCTL2 na hodnotu SMCLK / 23 (~640 kb/s).
- Vytvoří ukazatel do paměti (jsou používány segmenty na adrese 0xFBAA a 0xFDAA).
- Odemkne paměť nastavením registru FCTL3 a nastaví režim mazání v registru FCTL1.
- Zapíše do paměti „dummy“ bajt, čímž dojde ke smazání segmentu.
- Nastaví režim zápisu ve FCTL1.
- Kopírování dat do paměti.
- Vynuluje bit zápisu v registru FCTL1 a zamkne paměť proti zápisu.

Při zápisu do paměti je tak nejprve vymazán celý 512 bajtový segment a následně přepsán novými daty. Například pole *devices* bude pak v paměti uloženo tak, jak ukazuje obrázek 6.3.



Obrázek 6.3: Uložení dat v paměti.

K načítání dat z paměti slouží funkce:

- **unsigned char read_device_count_from_memory()** Přečte z paměti všechna uložená zařízení.
- **unsigned char read_events_from_memory (struct event *write_ptr)** Přečte z paměti všechny uložené události.
- **unsigned char read_device_count_from_memory()** Přečte z paměti první znak a zjistí, kolik zařízení je uloženo.
- **unsigned char read_event_count_from_memory()** Přečte z paměti první znak a zjistí kolik událostí je uloženo.

6.5 Probouzení počítače pomocí WOL

Technika WakeOnLAN je implementována v jedné funkci ve zdrojovém souboru wol.c. Jde o funkci *void send_wake_on_lan(unsigned char *mac_adress)* a k přenosu magického paketu využívá protokol UDP. Nejprve je inicializován buffer pro přenos UDP, poté je zjištěna broadcastová adresa v síti. Ta je spočtena jako:

```
Adresa_Bcast = Adresa_zařízení OR NOT(maska_sítě)
```

Na tuto adresu adresu je následně odeslán magický paket. Pro odesílání musela být dále upravena knihovna enc28j60_ip, kde bylo nutné zvolit broadcastovou MAC adresu vždy, když je IP adresa ve formátu IP broadcastu. Jinak by služba WakeOnLAN nefungovala.

6.6 Synchronizace času

Ke zjišťování času z internetu byl použit protokol SNTP. Jeho implementaci obsahuje soubor sntp.c a sntp.h. Obsluha je volána z funkce *void send_sntp_request()*, která zaregistruje „handler“ na port 123 transportního protokolu UDP, nastaví pole hlavičky, zapíše ji do odesílacího bufferu a odešle na adresu 217.31.205.226 (český server time.nic.cz).

Z odpovědi musí být vypočítáno aktuální datum a čas. To realizuje obslužná funkce SNTP, která přijaté sekundy od časového údaje 00:00 1.1.1900 přepočítá na sekundy k časovému údaji 00:00 1.1.2004. Tím se ušetří spousta kroků ve smyčce, kde jsou od přijatého čísla postupně odečítány časové úseky odpovídající rokům, měsícům, dnům, hodinám a minutám. Pro každý časový údaj existuje proměnná, která je inkrementována vždy, když se tento časový údaj podaří od celkové hodnoty sekund odečíst. Hodnota, která po odečítání zbyde, představuje aktuální počet sekund. Vypočtený čas se uloží do obvodu reálného času. Zjišťování času probíhá pouze jednou při startu systému, aktuální čas po zbytek běhu zařízení tak udržuje pouze obvod reálného času.

6.7 Hlavní program

Při připojení napájení začíná hlavní program mikrokontroléru ve funkci *int main()*, implementované v souboru main.c. Při startu systému dojde k jeho inicializaci voláním všech inicializačních funkcí. První z nich je *void BC_init()* umístěná v souboru BasicClock.c, která má na starosti inicializaci hodin mikroprocesoru. V prvé řadě povolí použití krystalu XT2 o frekvenci 14,7456 MHz a nastaví jej jako zdroj hodinového signálu s názvem CLK. Dále zvolí kmitočet hodin SMCLK roven kmitočtu CLK a kmitočet MCLK roven hodnotě CLK / 2. Spustí oscilátor a čeká ve smyčce, dokud nedojde k jeho ustálení. Nakonec nastaví 16 bitový časovač B na kmitočet tiku SMCLK / 4. Tento časovač využívá pouze funkce *void delay_ms(unsigned long msecs)*, určená pro generování zpoždění v milisekundách. Další inicializační funkce jsou *I2C_init()* a *SPI_init()* a nastavují komunikaci s obvodem reálného času a řadičem ethernetu ENC28J60. Následně jsou nulovány oba porty obsluhující spínací prvky SSR, nedojde tak k nechtěnému zapnutí spotřebiče. Inicializuje se i řadič ethernetu funkcí ENC28J60_init() a nastaví časovač A do režimu generování přerušení každých 500 milisekund. V poslední části inicializace se do RAM nahrají všechna data uložená v paměti flash, spustí inicializace DHCP (nebo nastaví pevná adresa) a nakonec je zaregistrován ukazatel na funkci obsluhující příchozím TCP spojení s portem číslo 80 (HTTP).

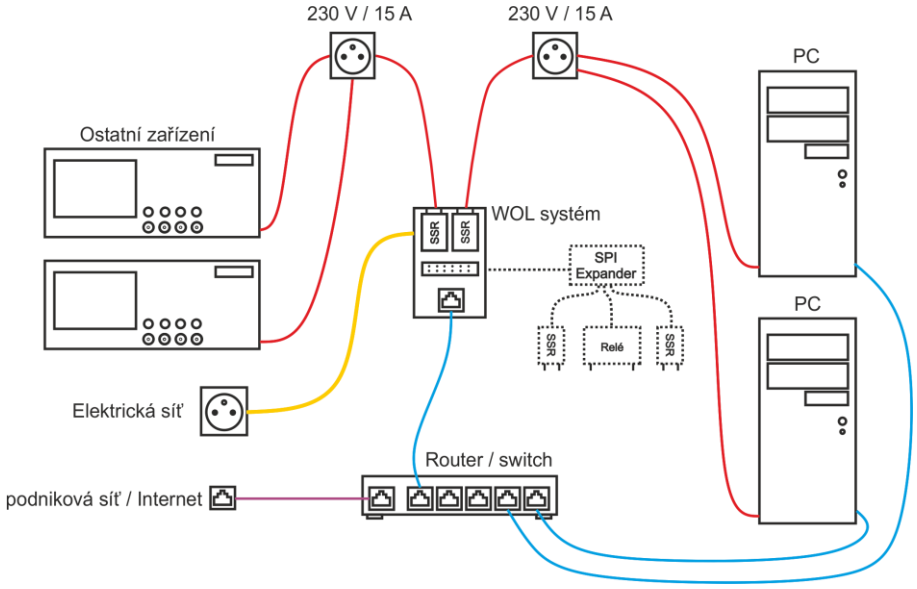
6.7.1 Hlavní smyčka a přerušovací rutiny

Hlavní smyčka obsluhuje automat spolehlivého přenosu TCP a odesílá tak nepotvrzené pakety nebo vysílá potvrzení pro přijaté pakety. Dále zjišťuje, jestli nebylo vyvoláno přerušení obvodem reálného času a pokud ano, vyvolá přidruženou obslužnou funkci, ve které se provedou všechny naplánované operace.

V programu využívám pouze dvojice přerušení. První pochází od časovače A, které se vyvolá každých 500 ms. Přerušení obslouží stavový automat protokolu DHCP a stavový automat TCP spojení. Druhé přerušení generuje obvod reálného času a dochází zde k nastavení globální proměnné *alarm* funkcí *set_alarm()*, čímž dojde při dalším cyklu hlavní smyčky k obsloužení uložené události.

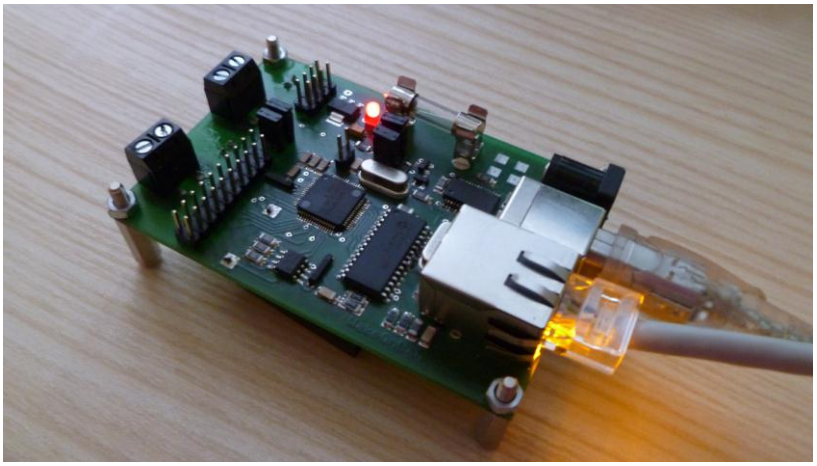
7 Výsledný systém

K testování systému jsem připojil výsledné zařízení do domácí sítě s přístupem na Internet. Jako centrální prvek sítě vystupuje router, na kterém jsem nastavil přesměrování veškeré komunikace s příchozím portem 80 na vytvořené zařízení. Prvek jsem tak mohl ovládat jak zadáním jeho lokální adresy, tak zadáním veřejné adresy, kterou mi přidělil poskytovatel internetu. Systém lze snadno připojit do jakékoliv jiné sítě a za jeho pomoci tak ovládat spotřebiče či probouzet počítače v okolí. Typickou modelovou situaci, kde bude systém využíván, zobrazuje obrázek 7.1.



Obrázek 7.1: Aplikační schéma systému.

Výhodou systému je jeho snadné rozšíření pomocí přídavných modulů. Lze tak mít například pro každý spotřebič v laboratoři vlastní spínací prvek, jenž bude připojen k WakeOnLAN systému. Toho lze dosáhnout například využitím SPI expandéru, který umožní adresovat několik desítek dalších spínacích prvků. Dalším rozšířením by mohl být modul teploměru, který by zjišťoval aktuální teplotu v místnosti a systém na jejím základě spínal topení v místnosti.

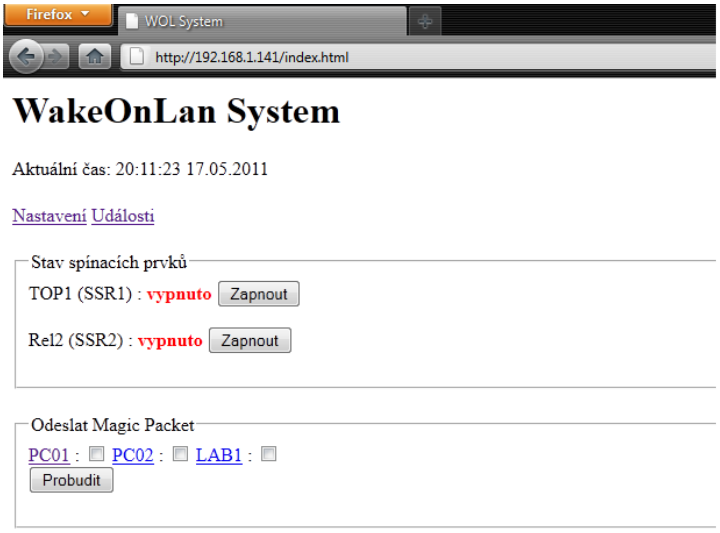


Obrázek 7.2: Výsledné zařízení - systém WakeOnLAN.

7.1 Vlastnosti webového rozhraní

Výkon mikrokontroléru není dostatečný na to, aby obsluhoval více uživatelů. Proto bylo snahou realizovat webové rozhraní s co nejrychlejší odezvou. To se podařilo, ovšem za cenu jeho jednoduchosti. Nepředpokládá se ale, že by uživatel trávil ovládáním zařízení delší dobu, spíše zařízení během krátkého okamžiku navštíví, probudí požadovaný počítač, popřípadě nastaví novou událost na konkrétní čas a tím jeho komunikace končí.

Nevýhodou je, že přístup k rozhraní nijak není chráněn. Znamená to tedy, že je vhodné omezit přístup na zařízení obsloužením pouze uživatelů s konkrétní IP adresou nebo skupinou adres. K tomu je v obslužné funkci TCP spojení připravena omezující podmínka.



Obrázek 7.3: Ukázka hlavní strany uživatelského rozhraní.

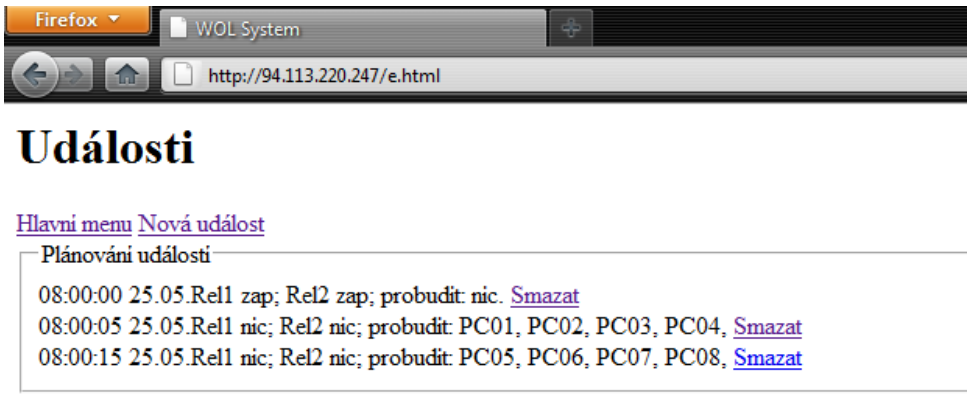
7.2 Plánování událostí a WakeOnLAN

Technika WakeOnLAN byla otestována jak teoreticky sledováním sítě programem Wireshark, tak prakticky probouzením reálných počítačů. Zařízení dokáže uložit až 32 počítačů a stejný počet lze probudit. Probouzet všechny počítače najednou ale není rozumné, jelikož zapnutím třiceti počítačů vznikne v síti proudová špička a některé prvky (pojistky, jističe) by tak mohly zareagovat a vypnout proud v celé učebně nebo budově. K probouzení více počítačů tak lze s výhodou využít plánovač, kde muže být v rozmezí několika sekund až minut vždy zapnuta některá část učebny. Například obrázek 7.4 demonstruje situaci, ve které nejprve dojde k sepnutí napájení a poté v rozmezí deseti sekund k probuzení dvou menších skupin počítačů.

Při vytváření nové události se zadává měsíc, den, hodina, minuta a sekuda, to znamená, že je možné plánovat s minimálním rozlišením jedné sekundy. Jediné na co by si měl uživatel při plánování dávat pozor je návaznost na aktuální čas (událost s časem menším než je aktuální čas se provede až za rok) a fakt, že obvod reálného času dokáže uložit pouze jednu událost (nelze mít více událostí na stejný čas).

Maximální limit uložených událostí byl nastaven na hodnotu 10. V případě nutnosti je možno tento limit upravit nastavením konstanty MAX_DEVICES v souboru web.h. Maximální limit uložených zařízení lze také zvětšit, ovšem je třeba počítat s jejich uložením v jedné události, kde má každé zařízení přiřazen jeden bit v proměnné *wol* typu unsigned long (kapitola 6.3.1). K probouzení většího počtu počítačů než je 32 by tak muselo dojít k úpravě struktury,

nebo by musel být vymyšlen jiný způsob uložení. Můj názor je, že toto číslo bohatě stačí i pro veliké učebny, natož domácnosti, kde jsou většinou jeden až dva počítače.



Obrázek 7.4: Stránka událostí s naplánovaným postupným spouštěním počítačů.

7.3 Spotřeba

Spotřeba systému byla zjišťována v šesti měřeních. V prvním byl systém v běžném provozu, kdy řadič ethernetu naslouchá na lince a mikroprocesor zjišťuje jeho stav. Napájení bylo realizováno transformátorovým zdrojem s udávaným výstupem 7 V / 2 A. Druhé měření probíhalo se stejným zdrojem, ale obvod ENC28J60 byl uspán. Všechny ostatní měření byly provedeny s využitím USB rozhraní jako hlavní napájecí zdroj. V třetím měření byl systém opět v normálním provozu. Ve čtvrtém měření byl uspán řadič ethernetu. Páté měření ukazuje spotřebu zařízení při uspaném procesoru v režimu LPM0 a uspaném ENC28J60. V posledním měření jsem zkoumal spotřebu celého systému s oběma SSR sepnutými. Výsledky měření shrnuje tabulka 7.1.

Měření číslo	Napájení	Popis	Napětí	Proud	Příkon
			V	mA	mW
1	Trafo 7 V	běžný provoz	7.18	174	1249.3
2	Trafo 7 V	ENC28J60 uspán	8.75	24.1	210.9
3	USB 5 V	běžný provoz	4.9	169.5	830.6
4	USB 5 V	ENC28J60 uspán	4.97	16.6	82.5
5	USB 5 V	MCU a ENC28J60 uspány	4.98	12.5	62.3
6	USB 5 V	běžný provoz, obě SSR sepnuty	4.89	186.2	910.5

Tabulka 7.1: Spotřeba systému v různých režimech.

Jak je vidět, rozdíl mezi některými stavy je značný. Zařízení má při běžném provozu několikanásobně větší spotřebu než s uspaným ethernetovým řadičem. V měření číslo 3, 4, 5 a 6 je systém napájen rozhraním USB, je tedy připojen i převodník FT232RL. Jeho spotřeba se ale nijak výrazně neprojeví. Spolu s obvodem reálného času spotřebovávají zhruba takové množství energie, jako ukazuje měření číslo 5. Z měření číslo 3 a 6 lze zjistit spotřebu jednoho SSR, která je přibližně 40 mW.

Největší konzument elektrické energie je tak právě obvod ENC28J60. Jeho veliká spotřeba se projevuje i značným tepelným vyzařováním.

7.4 Spínací prvky

Spínání pomocí prvků SSR bylo otestováno na několika spotřebičích. Hodnoty spínaných výkonů se pohybovaly v řádu od 7 W po 1900 W, při běžném síťovém napětí. K ovládání relé posloužilo jak napětí USB rozhraní s naměřenou hodnotou 4,9 V, tak běžný síťový zdroj s naměřenou hodnotou 7,1 V. Ačkoliv je napětí USB nižší než minimální ovládací napětí, které udává výrobce SSR (5 V), nebyl pozorován žádný rozdíl oproti silnějšímu 7 V zdroji, ani žádný úbytek napětí na svorkách relé vlivem nedostatečného „otevření“ spínacího prvku uvnitř SSR.

S rostoucím výkonem, roste také teplota zahřívání celého SSR. Zhruba na rozmezí 1 kW je teplota relé neúnosná na dotek člověka. Je možné, že by při delší době sepnutí teplota relé dosáhla maximální udávané teploty 100 °C, proto se při spínaní větších výkonů po delší časový úsek jeví jako vhodné použít chladič.

7.5 Časomíra

SNTP server poskytuje čas v zimním formátu. To znamená, že v době kdy je čas v letním formátu, bude čas zjištěný prostřednictvím SNTP serveru o hodinu zpožděn oproti aktuálnímu času.

Nevýhoda použitého RTC je, že při ztrátě napájení přechází obvod reálného času do režimu udržování aktuálních hodnot registrů a napájení přepne na záložní baterii. Znamená to, že při dalším spuštění systému bude v obvodu čas, kdy došlo k výpadku elektřiny. Pokud systém při startu nezíská přístup na síť Internet a nebude tak moci aktualizovat čas na základě SNTP požadavku, zůstává čas nenastaven. Tento stav je pak zobrazen uživateli na hlavní straně webového rozhraní v podobě upozornění a nastavení času musí uživatel provést sám.

7.6 Ovládací skript

Pokud by bylo zařízení nasazeno v laboratořích či učebnách, nebylo by příliš vhodné zpřístupnit veškeré jeho funkce neoprávněným osobám, například studentům. Z tohoto důvodu jsem vytvořil jednoduchý skript v jazyce Python, který je zaměřen právě na tyto osoby a umožní jim alespoň probudit počítač, na kterém chtějí pracovat. Skript využívá protokol TCP s cílovým portem nastaveným na hodnotu 80 a pro komunikaci se zařízením používá vlastní protokol.

Protokol obsahuje dva příkazy, z nichž jeden je tvaru `WOL LIST`, po jehož odeslání odpoví zařízení výpisem všech uložených zařízení. Druhý pak tvaru `WOL W NAME`, kde „NAME“ je jméno některého z vypsaných zařízení. Jakmile zařízení tento příkaz přijme, probudí žádaný počítač. Skript je uložen na přiloženém CD a pro použití v něm musí být nastavena IP adresa WakeOnLAN systému.

8 Závěr

Cílem diplomové práce bylo vytvořit vestavěné zařízení, které umožní vzdáleně ovládat přívod elektrické energie několika spotřebičů a probouzet uspané počítače technikou WakeOnLAN. K tomu bylo třeba nastudovat protokoly využívané v počítačové síti, hlavně protokoly architektury TCP/IP. Systém byl navrhnut a jako hlavní řídící člen využívá nízkopříkonový mikrokontrolér firmy Texas Instruments MSP430F2617. Ke komunikaci přes síť je připojen sběrnicí SPI řadič ethernetu ENC28J60. Zařízení obsahuje další dva integrované obvody. FT232RL je využíván k programování mikrokontroléru skrze rozhraní USB a obvod reálného času M41T81, který udržuje v systému aktuální čas spolu s možností nastavení alarmu. Pro implementaci síťových protokolů TCP, UDP, ICMP, IP, ARP a DHCP jsem využil část knihovny z open-source projektu FITkit, zaměřenou také na ovládání použitého řadiče ethernetu. Následně jsem se zaměřil hlavně na realizaci webového rozhraní, jehož prostřednictvím je systém ovládán, na implementaci techniky WakeOnLAN, tvorbou knihovny pro komunikaci s obvodem reálného času a v neposlední řadě realizaci protokolu SNTP.

Výsledkem je tak univerzální systém, který může být použit jak pro správu elektrické energie, tak například jako malý vývojový kit, na kterém lze díky své modularitě realizovat další aplikace. Rozšíření systému tak vidím hlavně v implementaci minimálních autentizačních prvků k použití ve veřejné síti. Dále v implementaci protokolu SSDP firmy Microsoft, který zjišťuje, jaké služby síťové zařízení poskytuje. Mohl by být připojen teploměr a na základě zjištěné teploty regulovat prostředí v místnostech. Nakonec by bylo v některých situacích výhodné použití wi-fi modulu, který by omezil počet potřebných kabelů na jeden napájecí.

Během testovací fáze bylo zjištěno, že použití prvků SSR není vhodné v případě spínání větších výkonů po delší časový úsek z důvodu jejich zahřívání. Jako náhradu bych doporučil obyčejné elektromagnetické relé, které sice při sepnutí spotřebovává více energie, ale nedochází k vyzařování ohromného množství tepla do okolí.

9 Použitá literatura

[1] BIGELOW, S. J-; MATĚJŮ, P. *Mistrovství v počítačových sítích: správa, konfigurace, diagnostika a řešení problémů*. Vyd. 1. Brno : Computer Press, 2004. 990 s. ISBN 80-251-0178-9.

[2] PUŽMANOVÁ, R. *Moderní komunikační sítě od A do Z*. 2. aktualiz. vyd. Brno : Computer Press, 2006. 430 s. ISBN 80-251-1278-0.

[3] DOLEČEK, J. *Moderní učebnice elektroniky*. Praha : BEN - technická literatura, 2005. 206 s. ISBN 80-730-0161-6.

[4] BLAKE, Carl; BULL, Chris *IGBT or MOSFET: Choose Wisely*. [online] [cit. 2011-01-09].
URL: http://www.irf.com/technical-info/whitepaper/choosewisely.pdf

[5] DAPKUS, D. *Using MOS-Gated power transistor in AC switch application*. [online]. April 4, 2003 [cit. 2011-01-09].
URL: http://www.irf.com/technical-info/designtp/dt94-5.pdf

[6] DODGE, J. *Latest Technology PT IGBTs vs. Power MOSFETs*. [online]. Duben 4, 2003 [cit. 2011-01-09].
URL: http://www.microsemi.com/micnotes/APT0302.pdf

[7] Cosmo *KSD215AC3 Datasheet*. [online]. 3 listopadu 2004 [cit. 2011-01-09].
URL: http://www.cosmo-ic.com/object/products/KSD215AC3.pdf

[8] Microchip *ENC28J60 Datasheet*. [online]. 2008 [cit. 2011-01-09].
URL: http://ww1.microchip.com/downloads/en/DeviceDoc/39662c.pdf

[9] ST Microelectronics *M41T81S Datasheet*. [online]. Prosinec 2004 [cit. 2011-01-09].
URL: http://www.datasheetcatalog.org/datasheet2/4/090pu1oe8z21kyl765k66p2fj0wy.pdf

[10] Texas Instruments *MSP430F241x, MSP430F261x*. [online]. Červen 2007, Prosinec 2009 [cit. 2011-01-09].
URL: http://focus.ti.com/lit/ds/symlink/msp430f2617.pdf

[11] FTDI Chip *FT232R Datasheet*. In [online]. 2010 [cit. 2011-01-09].
URL: http://www.ftdichip.com/Support/Documents/DataSheets/ICs/DS_FT232R.pdf

[12] Eaton Corporation. *Úvahy o vlivu kapacitní zátěže na činnost UPS.* [online]. 13 Červenec 2009 [cit. 2011-01-09].
URL: http://www.etm.cz/rubriky/energetika/404-uvahy-o-vlivu-kapacitni-zateze-na-cinnost-ups

[13] KOPECKÝ, L. *Spínání induktivní zátěže*. [online]. 2005 [cit. 2011-01-09].
URL: http://kopecky.rtyne.net/teorie/RL-switch.pdf

[14] VACKA, Z. *Polovodičové stykače - Solid State Relay*. online]. 2008 [cit. 2011-01-09].
URL: http://www.elproz.cz/Ssr.htm

[15] POSTEL, J. *Internet Control Message Protocol : Darpa Internet Program Protocol SPECIFICATION*. [online]. Říjen 1981 [cit. 2011-01-09].
URL: http://tools.ietf.org/html/rfc792

[16] POSTEL, J. *User Datagram Protocol*. [online]. Srpen 1980 [cit. 2011-05-17].
URL: http://www.faqs.org/rfcs/rfc768.html

[17] Wikipedia : the free encyclopedia *Wake on lan.* [online]. 4 března 2006, naposledy upraveno 30 prosince 2010 [cit. 2011-01-09].
URL: http://en.wikipedia.org/wiki/Wake_on_lan

[18] Wikipedia : the free encyclopedia. *Relay*. [online]. 24 května 2005, naposledy upraveno 5 ledna 2011 [cit. 2011-01-09].
URL: http://en.wikipedia.org/wiki/Relay

[19] BRADEN, R.; BORMAN, D.; PARTRIDGE, C. *Computing the Internet Checksum*. [online]. Září 1988 [cit. 2011-01-09].
URL: http://tools.ietf.org/html/rfc1071

[20] Wikipedia : the free encyclopedia *Přestupný rok*. [online]. 8 června 2004, naposledy upraveno 18 března 2011 [cit. 2011-05-06].
URL: http://cs.wikipedia.org/wiki/P%C5%99estupn%C3%BD_rok

[21] MILLIS, D. *Simple Network Time Protocol. (SNTP) Version4*. [online]. Říjen 1988 [cit. 2011-05-01].
URL: http://tools.ietf.org/html/rfc1071

[22] FIELDING, R. a další. *Hypertext Transfer Protocol – HTTP/1.1*. [online]. Červen 1999 [cit. 2011-05-01].
URL: http://www.ietf.org/rfc/rfc2616.txt

[23] Wikipedia : the free encyclopedia. *Hypertext Transfer Protocol*. [online]. 27 června 2004, naposledy upraveno 14 května 2011 [cit. 2011-05-01].
URL: http://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol

[24] Wikipedia : the free encyclopedia. *I^2C*. [online]. 9 října 2005, naposledy upraveno 19 května 2011 [cit. 2011-05-08].
URL: http://en.wikipedia.org/wiki/I%C2%B2C

[25] Wikipedia : the free encyclopedia. *Universal asynchronous receiver/transmitter*. [online]. 7 listopadu 2005, naposledy upraveno 24 května 2011 [cit. 2011-05-08].
URL: http://en.wikipedia.org/wiki/Universal_asynchronous_receiver/transmitter

[26] KALINSKY, D., KALINSKY, R. *Introducing to Serial Peripheral Interface*. [online]. 1 Února 2002, [cit. 2011-05-08]
URL: http://www.eetimes.com/discussion/beginner-s-corner/4023908/Introduction-to-Serial-Peripheral-Interface

[27] VAŠÍČEK Z., Fakulta informačních technologií VUT *Stránka projetktu FITkit*
URL: http://merlin.fit.vutbr.cz/FITkit/

Seznam použitých zkratek

ARP – Address Resolution Protocol

DHCP – Dynamic Host Configuration Protocol

HTML – HyperText Markup Language

HTTP – HyperText Transfer Protocol

ICMP – Internet Control Message Protocol

IP – Internet Protocol

MAC – Media Access Control

MCU – Micro Control Unit

RTC – Real Time Circuit

SNTP – Simple Network Time Protocol

SSR – Solid State Relay

TCP – Transtport Control Protocol

UDP – User Datagram Protocol

URL – Uniform Resource Locator

WOL – Wake On LAN

Seznam příloh

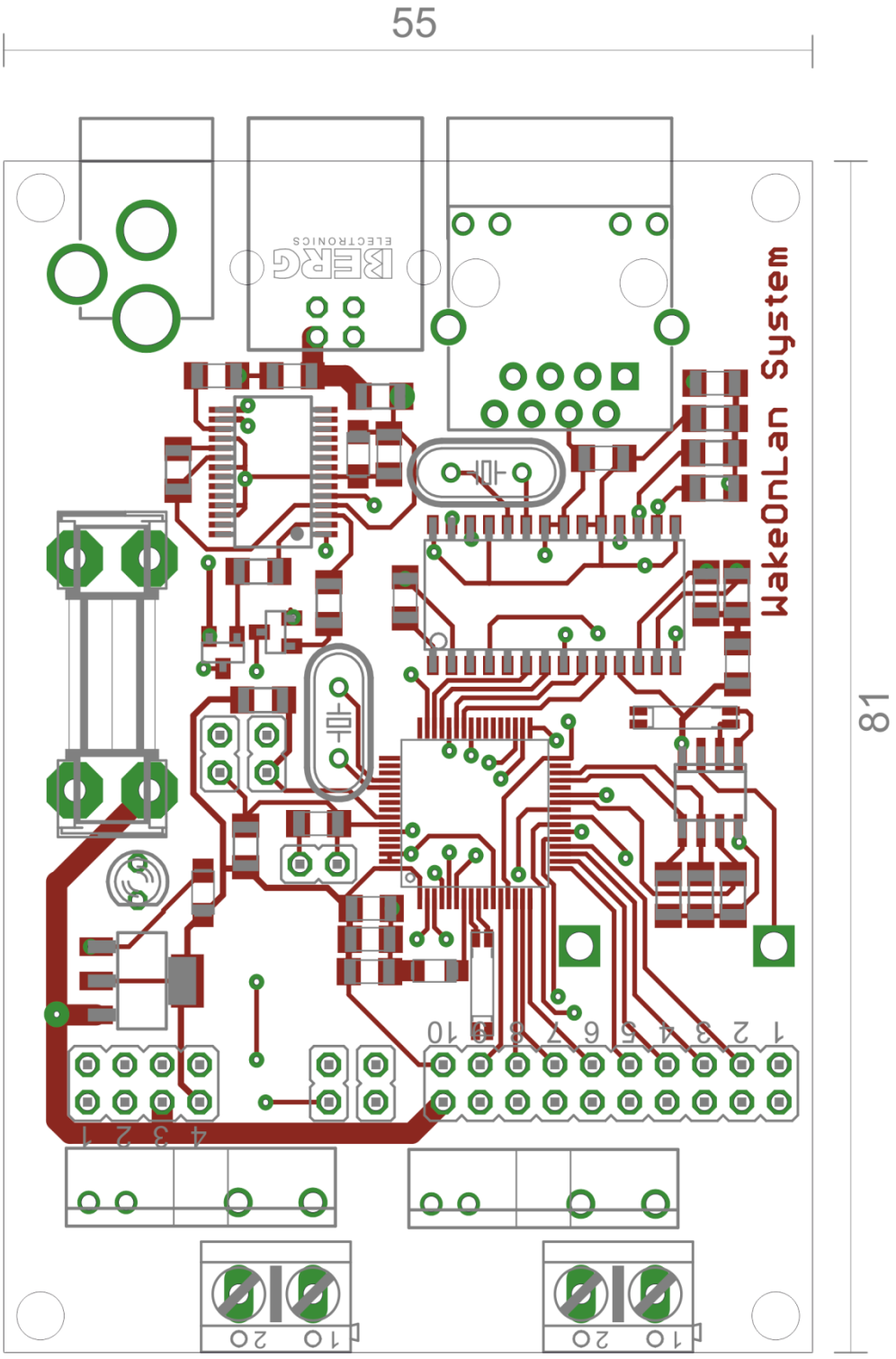
Příloha A

Navržené schéma zapojení

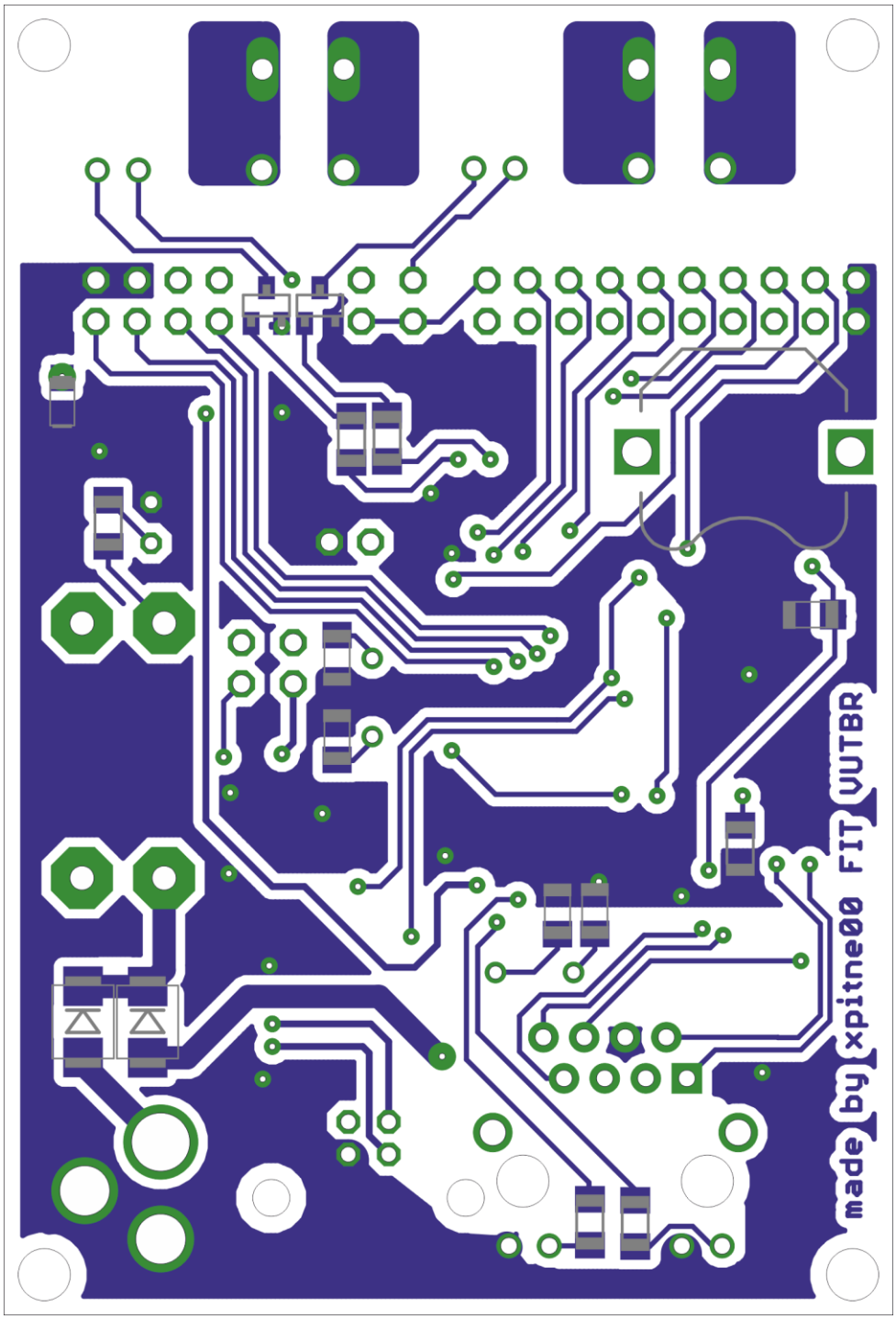
Schéma je přiloženo ve formátu A3 jako externí příloha

Příloha B

Navržená deska plošných spojů



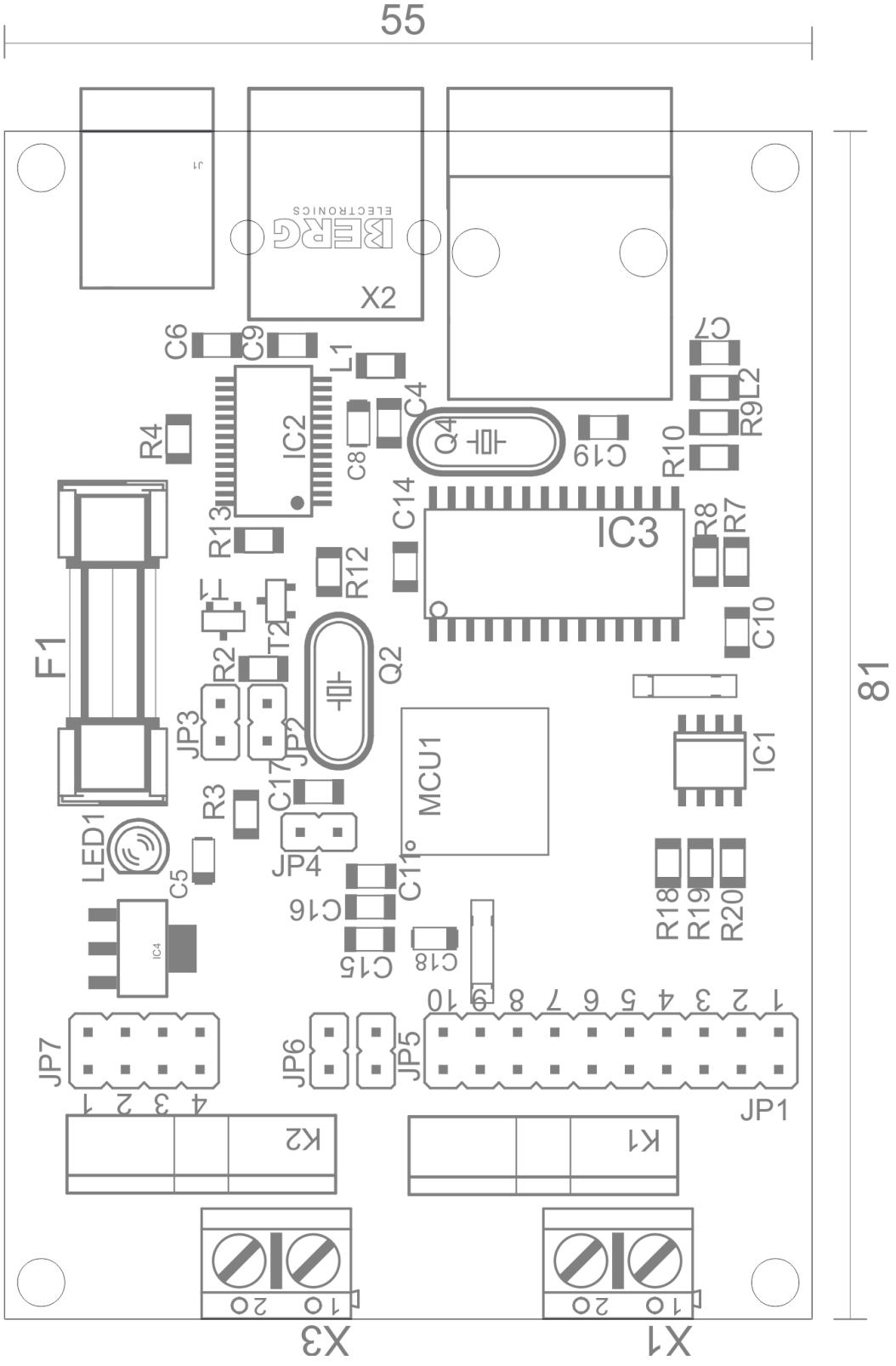
Obrázek B.1: Horní strana desky plošných spojů.



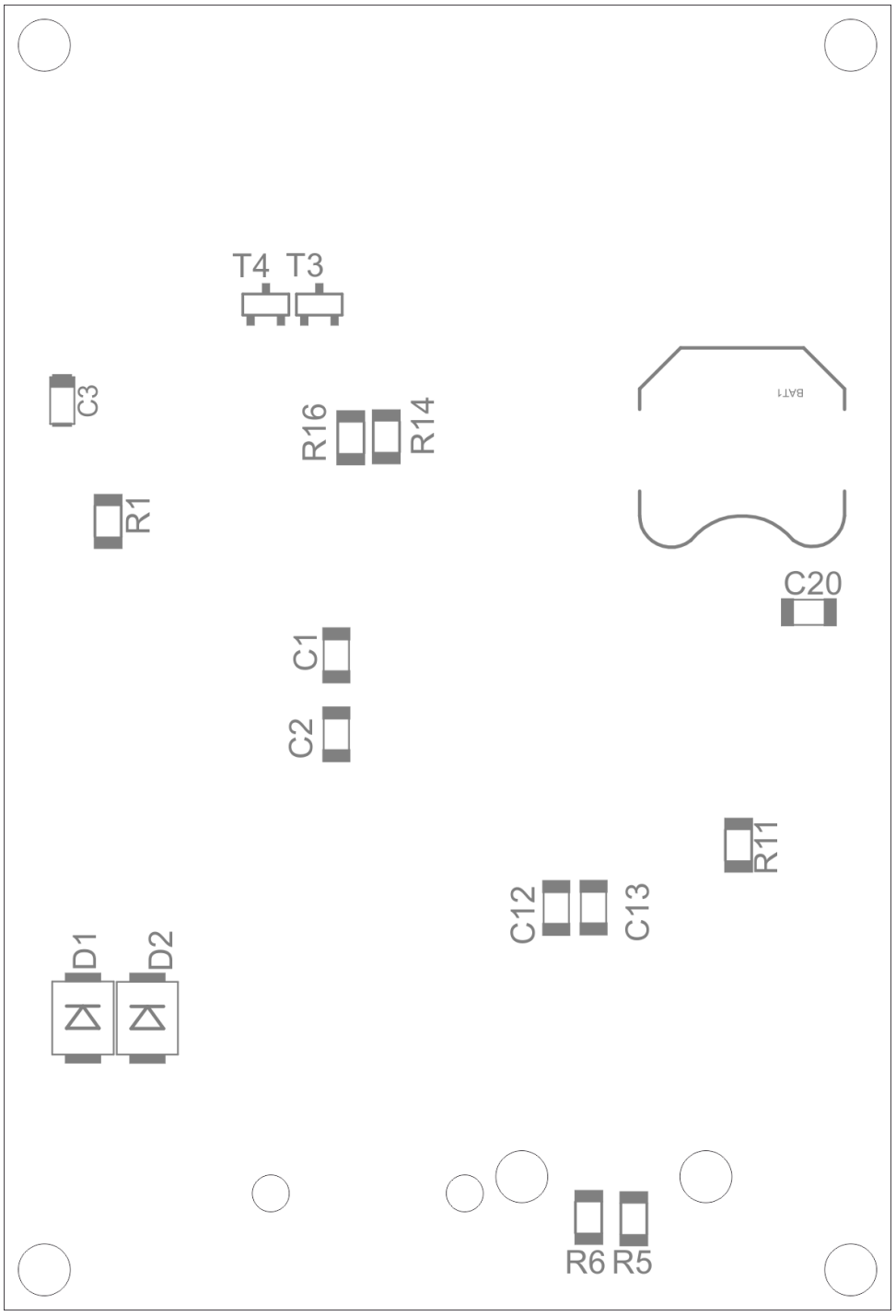
Obrázek B.2: Spodní strana desky plošných spojů.

Příloha C

Osazovací schéma a popis propojek



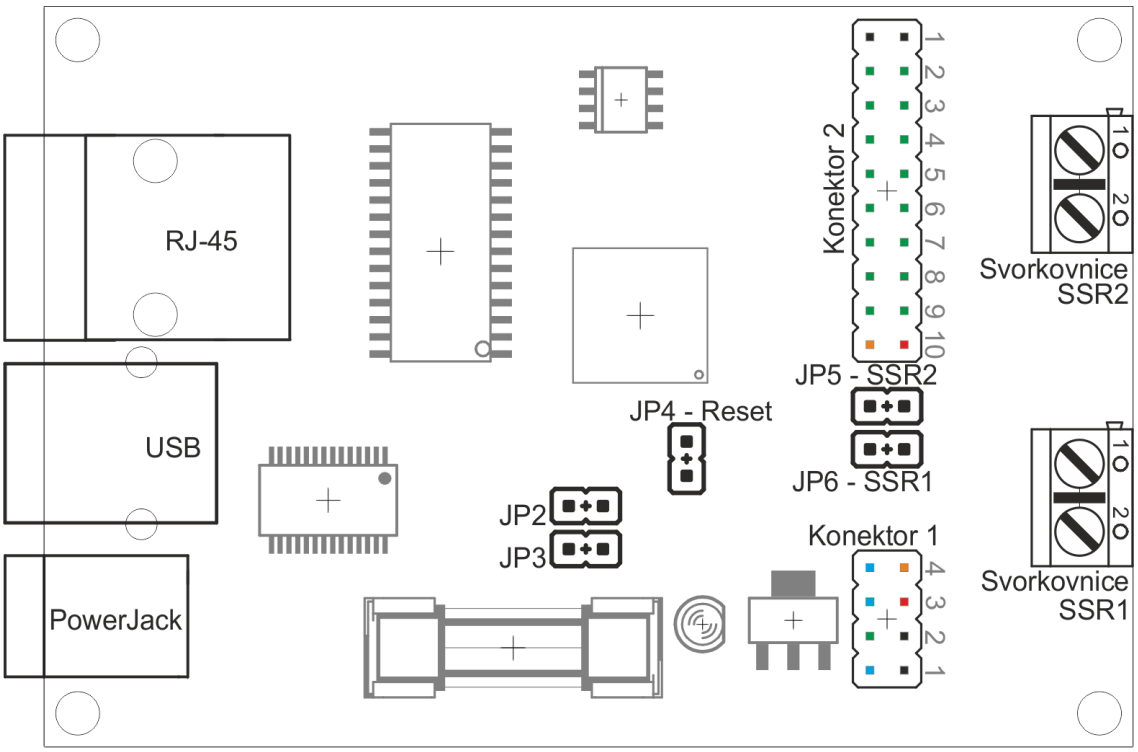
Obrázek C.1: Horní strana osazení součástek.



Obrázek C.2: Spodní strana osazení součástek.

Příloha D

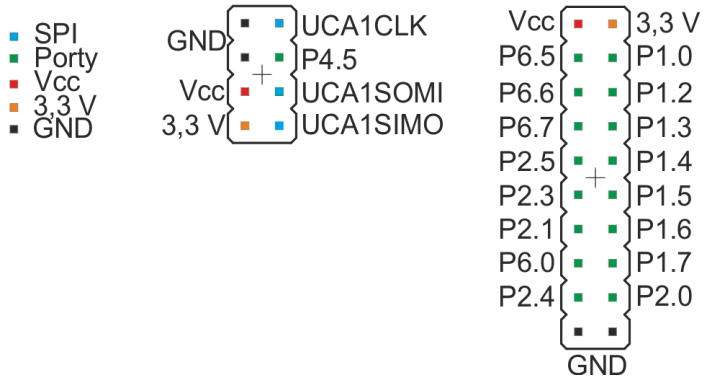
Popis propojek a konektorů



Propojky:

JP2, JP3 - programovací propojky,
při programování musí být propojeny
JP5, JP6 - manuální rozpínání relé,
při ovládání musí být propojeny
JP4 - reset, spojením dojde k resetu MCU

Konektory:



Obrázek D.1: Popis propojek a konektorů.