ZÁPADOČESKÁ UNIVERZITA V PLZNI

FAKULTA ELEKTROTECHNICKÁ

Katedra aplikované elektroniky a telekomunikací

DIPLOMOVÁ PRÁCE

Ovládání vybraných přístrojů v akustických laboratořích FEL pomocí webového rozhraní

Autor práce: Bc. Vladimír Smitka Plzeň 2012

Vedoucí práce: Ing. Oldřich Tureček, Ph.D.

ZÁPADOČESKÁ UNIVERZITA V PLZNI
Fakulta elektrotechnická
Akademický rok: **2011/2012**

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Vladimír SMITKA**
Osobní číslo:	**E09N0157P**
Studijní program:	**N2612 Elektrotechnika a informatika**
Studijní obor:	**Elektronika a aplikovaná informatika**
Název tématu:	**Ovládání vybraných přístrojů v akustických laboratořích FEL pomocí webového rozhraní**
Zadávající katedra:	**Katedra aplikované elektroniky a telekomunikací**

Zásady pro vypracování:

1. Prostudujte možnosti dálkového ovládání vybraných přístrojů v akustických laboratořích FEL.
2. Realizujte ovladač těchto přístrojů, který bude využívat webového rozhraní.
3. Při konstrukci ovladače berte v úvahu jeho možnou budoucí rozšiřitelnost.

Rozsah grafických prací:	**podle doporučení vedoucího**
Rozsah pracovní zprávy:	**30 - 40 stran**
Forma zpracování diplomové práce:	**tištěná/elektronická**
Seznam odborné literatury:	

Student si vhodnou literaturu vyhledá v dostupných pramenech podle doporučení vedoucího práce.

Vedoucí diplomové práce:	**Ing. Oldřich Tureček, Ph.D.** Katedra technologií a měření
Konzultant diplomové práce:	**Ing. Oldřich Tureček, Ph.D.** Katedra technologií a měření
Datum zadání diplomové práce:	**18. října 2010**
Termín odevzdání diplomové práce:	**11. května 2012**

Doc. Ing. Jiří Hammerbauer, Ph.D.
děkan

L.S.

Doc. Dr. Ing. Vjačeslav Georgiev
vedoucí katedry

V Plzni dne 17. října 2011

Abstrakt

Tato diplomová práce se zabývá návrhem a implementací modulárního zařízení sloužícího ke vzdálenému ovládání přístrojů převážně v akustických laboratořích FEL pomocí webového rozhraní. Zařízení je tvořeno základní deskou, která zprostředkovává veškerou síťovou komunikaci mezi webovým prohlížečem uživatele a rozšiřujícími deskami ovládajícími vybrané přístroje.

Zařízení je postaveno na platformě STM32 a využívá operačního systému reálného času FreeRTOS. Rozšiřující desky jsou k zařízení připojeny pomocí sběrnice I_2C. V systému je implementován http server nabízející uživateli komfortní webové rozhraní, které s hardwarem komunikuje pomocí technologie asynchronního javascriptu a protokolu založenému na datovém formátu JSON.

Součástí práce je návrh samotného zařízení včetně desky plošného spoje a také stručný teoretický popis použitých technologií s odkazy na podrobnější zdroje.

Klíčová slova

Embedded systém, mikrokontrolér, MCU, ARM Cortex-M3, STM32, FreeRTOS, Ethernet, TCP/IP, webové rozhraní, AJAX

Abstract

This diploma thesis describes the design and implementation of a modular device aimed to remote control of measuring instruments mainly in the acoustic laboratories of Faculty of Electrical Engineering (FEL). The remote control is driven by a web interface. The device consists of the mainboard which provides all network communication among user's web browser and the expansion boards that control selected measuring instruments.

The device is built on the STM32 platform and uses the FreeRTOS realtime operating system. The extension boards are connected with the device by I_2C bus. There is a HTTP server implemented in the system which provides a comfortable web interface to the user. The interface uses asynchronous javascript and a communication protocol based on the JSON format.

The diploma thesis contains the design of the device including the printed circuit board and a short theoretical description of used technologies with links to additional sources.

Key words

Embedded system, microcontroller, MCU, ARM Cortex-M3, STM32, FreeRTOS, Ethernet, TCP/IP, web interface, AJAX

Prohlášení

Předkládám tímto k posouzení a obhajobě diplomovou práci, zpracovanou na závěr studia na Fakultě elektrotechnické Západočeské univerzity v Plzni.

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně, s použitím odborné literatury a pramenů uvedených v seznamu, který je součástí této diplomové práce.

Dále prohlašuji, že veškerý software, použitý při řešení této diplomové práce, je legální.

V Plzni dne 28.4.2012 Bc. Vladimír Smitka

Poděkování

Tímto bych rád poděkoval vedoucímu diplomové práce Ing. Oldřichu Turečkovi, Ph.D. za možnost realizovat tuto práci. Zvláštní poděkování náleží Ing. Josefu Šístkovi za pomoc při výrobě prototypů. Dále děkuji své rodině a přátelům za podporu při tvorbě této práce.

Seznam použitých zkratek

ACK	Acknowledgement	Potvrzení příjmu
AD	Analog to Digital	Typicky převod z analogového signálu na číslicový
API	Application Programming Interface	Rozhraní pro programování aplikací
ARM	Advanced RISC Machine	Architektura procesoru
FIFO	First In First Out	Typ paměti
GPIO	General Purpose Input Output	Obecný vstupně/výstupní port
HTTP	Hypertext Transfer Protocol	Komunikační protokol používaný na internetu
LAN	Local Area Network	Místní síť
LDO	Low Dropout regulator	Regulátor s nízkým úbytkem napětí
LSB	Least Significant Bit	Nejméně významný bit
lwIP	Lightweight Internet Protocol	Konkrétní realizace TCP/IP stacku
MCU	Microcontroller Unit	Mikrokontrolér
MTU	Maximum Transmission Unit	Maximální přenosová jednotka
OS	Operating System	Operační systém
OTG	On-The-Go	Rozšíření USB, umožňující přepínat režim Host-Periferie za chodu
PCB	Printed Circuit Board	Deska plošného spoje (DPS)
PLL	Phase Locked Loopback	Fázový závěs
PoE	Power over Ethernet	Napájení po Ethernetu

Obsah

1 Úvod

Tato diplomová práce se zabývá možnostmi zprostředkování vzdáleného ovládání přístrojů pomocí webového rozhraní. Důvodem výběru právě tohoto tématu byla skutečnost, že se autor profesně zabývá sítěmi a návrh tohoto zařízení bude mít tedy praktické uplatnění i v tomto oboru.

Cílem této práce je vytvořit koncepci zařízení, které by umožňovalo vzdálené ovládání přístrojů používaných primárně v akustických laboratořích FEL ZČU, návrh jeho zapojení, jeho realizace na vývojové desce, napsání příslušného software k jeho ovládání a návrh finálního výrobku. Zařízení by mělo sloužit jako univerzální platforma, která komunikuje s dalšími moduly a poskytuje pro ně jednotné uživatelské webové rozhraní. Při vývoji musel být tedy kladen důraz na různé scénáře jeho použití.

Vybrané přístroje v akustických laboratořích je možné ovládat především pomocí sběrnice RS-232. Důležitou vlastností, kterou má navrhované zařízení splňovat, je jeho modularita a rozšiřitelnost pro ovládání dalších zařízení. Vzhledem k současnému vybavení laboratoří je tedy aktuálním požadavkem na zařízení možnost použití více konfigurovatelných RS-232 portů. Novější přístroje jsou stále častěji vybaveny rozhraním USB, proto je vhodné, aby toto rozhraní bylo také v budoucnu podporováno.

Zařízení bylo z důvodu možností rozšiřování navrženo jako hlavní základní deska zprostředkovávající síťovou komunikaci a obsluhu periférií s rozšiřujícími moduly pro konkrétní přístroje.

2 Rozbor problematiky a použité nástroje

2.1 Návrh řešení vzdáleného ovládání

Jak již bylo zmíněno, je zařízení složeno ze základní desky a rozšiřujících karet. Navržená základní deska je schopna pracovat i samostatně bez rozšiřujících modulů. Pro samostatné fungování je vybavena sériovým rozhraním RS-232, USB OTG, slotem pro SD kartu, GPIO piny a AD převodníkem. S rozšiřujícími kartami deska komunikuje pomocí sběrnice I_2C.

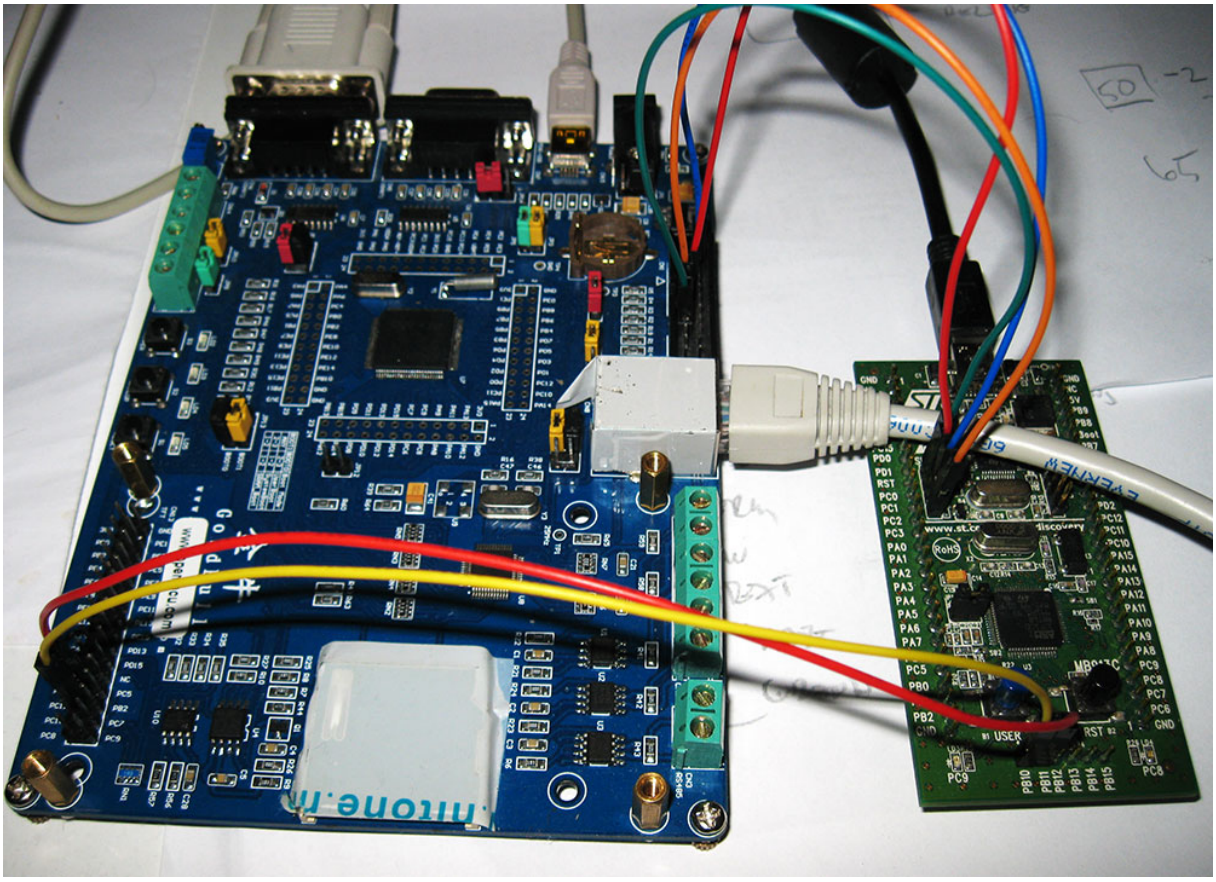
Ve firmware základní desky je naprogramován HTTP server, který poskytuje webové uživatelské rozhraní pro ovládání jednotlivých periférií. Pro snížení nároků na samotný HW je komunikace uživatelského rozhraní řešena pomocí technologie asynchronního javascriptu - AJAX. Při interakcích tak není přenášena celá www stránka, ale pouze informace z aktuálně komunikujících modulů. Pro výměnu zpráv se serverem je použit protokol založený na formátu JSON (JavaScript Object Notation). Podrobnější popis protokolu je možno nalézt dále v této práci.

V souvislosti s požadavkem na jednoduché nasazení zařízení v praxi je nutné se detailněji zabývat možnostmi jeho napájení. Vzhledem k připojení zařízení do sítě LAN se jako dobrá volba ukázala možnost napájení zařízení přes Ethernet - PoE (IEEE 802.3af). Řešení tohoto druhu napájení bude podrobněji popsáno v kapitole 4. Druhou možností je napájení externím zdrojem 10 - 40 V. Při napájení ze zdroje 12 V nepřekračuje odběr hodnotu 200 mA (odběr se liší převážně dle síťové aktivity, ale přes tuto hranici by se neměl, pokud zařízení funguje správně, nikdy dostat). Ve zvláštních případech by mohlo být vhodné i napájení přes sběrnici USB. Tato možnost byla při návrhu zařízení také brána v potaz.

2.2 Vývojové prostředky a další použitý software

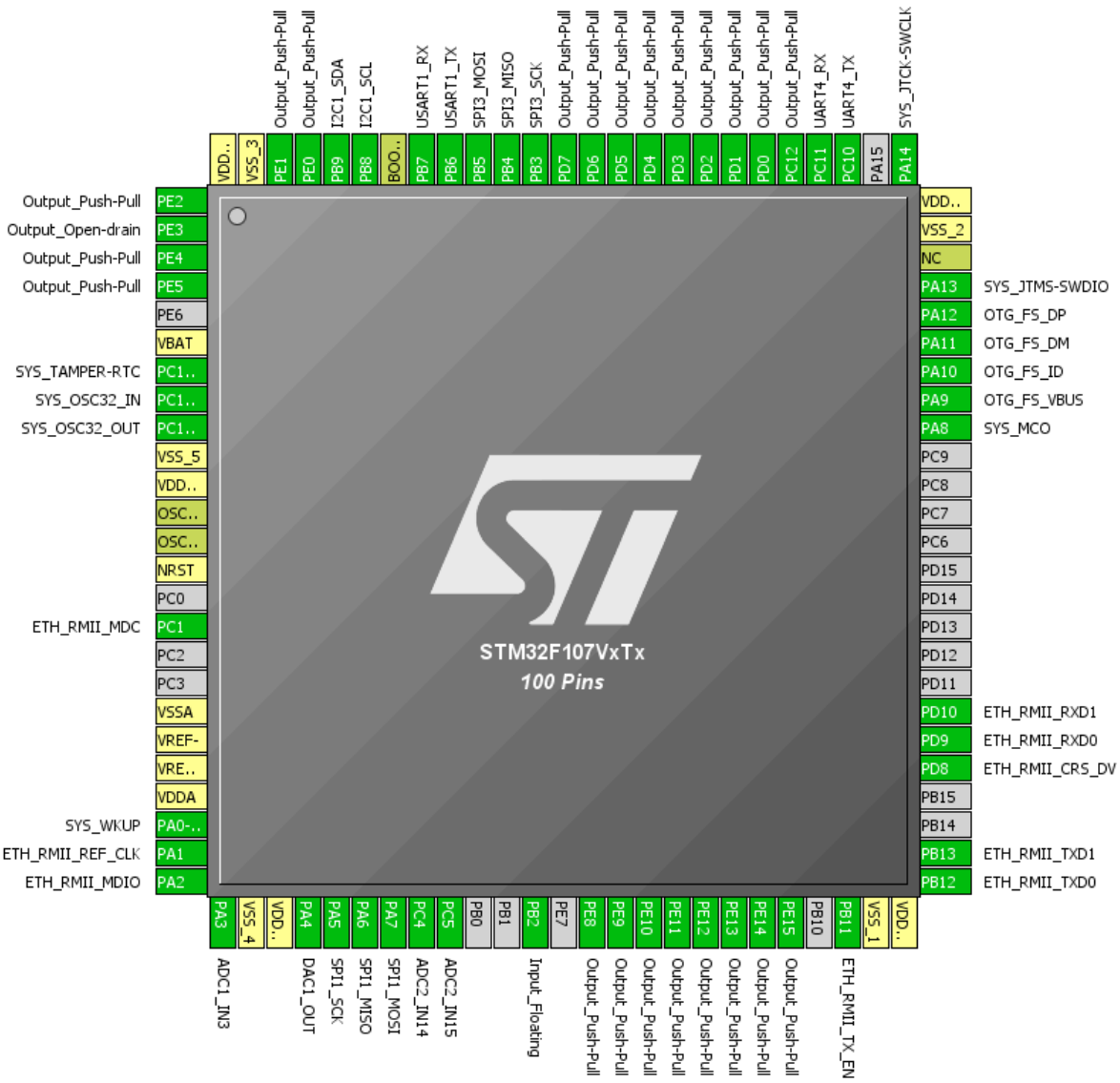
Vývoj zařízení probíhal na vývojové desce Taurus osazené 32-bitovým mikrokontrolérem ARM Cortex-M3. Jako vývojové prostředí byla použita bezplatná verze softwaru Atollic True Studio 2.6, která byla z důvodu změny licence nahrazena v koncových fázích vývoje zkušební verzí KEIL uVision. Komunikaci s přípravkem zajišťoval vývojový kit STM32-Discovery, který je možné využívat jako programátor přes sběrnici ST-Link.

Produkty firmy ST byly zvoleny právě díky dobré dostupnosti vývojových prostředků.



Obr 2.1: Vývojová deska s programátorem

Pro návrh využití pinů mikrokontroléru a jejich konfiguraci byl použit program MicroXplorer firmy ST.



Obr. 2.2: Využití pinů v programu MicroXplorer

K ladění byly využívány nástroje vývojového prostředí a nástroj STM Studio.

Návrh schématu a PCB byl prováděn v návrhovém prostředí Eagle. Jelikož prostředí Eagle neobsahuje knihovny pro zvolený typ procesoru a dalších použitých obvodů, musely být tyto knihovny vytvořeny. Schématické značky vychází z fyzického vzoru součástek, což umožňuje vytvořit si představu o výsledné podobě PCB již při návrhu jeho schématu.

Úprava obrázků pro tuto práci probíhala v grafickém editoru Adobe Photoshop. Diagramy byly připravovány v programu Adobe Fireworks. Text práce byl psán v prostředí MS Word.

2.3 Využívané sběrnice a protokoly

Tato kapitola se bude zabývat charakteristikou sběrnic a protokolů použitých v navrhovaném zařízení. Konkrétně se jedná o protokoly Ethernet, RS-232, USB, SPI a I_2C. Jelikož je tato problematika dopodrobna probrána v mnoha publikacích, bude zde pouze nastíněn pouze stručný popis těchto sběrnic a protokolů a jejich zajímavé vlastnosti. Odkazy na doplňkovou literaturu jsou uvedené v konkrétních pasážích věnovaných danému tématu.

2.3.1 Ethernet a TCP/IP

Ethernet je soubor technologií, které se používají převážně v sítích LAN podle standardu IEEE 802.3 a pokrývají linkovou vrstvu v ISO/OSI modelu. Ethernet původně využíval sběrnicovou topologii s metodou řízení přístupu CSMA/CD (Collision Sense Multiple Access/Collision Detection). V dnešní podobě se však díky použití síťových switchů setkáváme výhradně s komunikací Point-to-Point, kde sběrnici využívají pouze 2 zařízení a ke kolizím proto nedochází. Topologie je tak stromová. Původně byl Ethernet také provozován po koaxiálním kabelu. Dnes se však používá kroucená dvoulinka, na které ani nelze původní sběrnicovou topologii provozovat. Parametry použitých vodičů jsou opět definovány standardy, nejrozšířenějším pro 100 Mbit/s sítě je dnes 100BaseTX na nestíněných UTP kabelech kategorie 5 a lepších. Impedance nestíněného UTP kabelu je 100 Ω a jeho maximální délka je 100 m. Na fyzické vrstvě jsou použita kódování 4B/5B a MLT-3. Standard 100BaseTX je podporovaný i navrženým zařízením. Další standardy určují, jakým způsobem provozovat Ethernet například na optických vláknech při rychlostech až 10 Gbit/s.

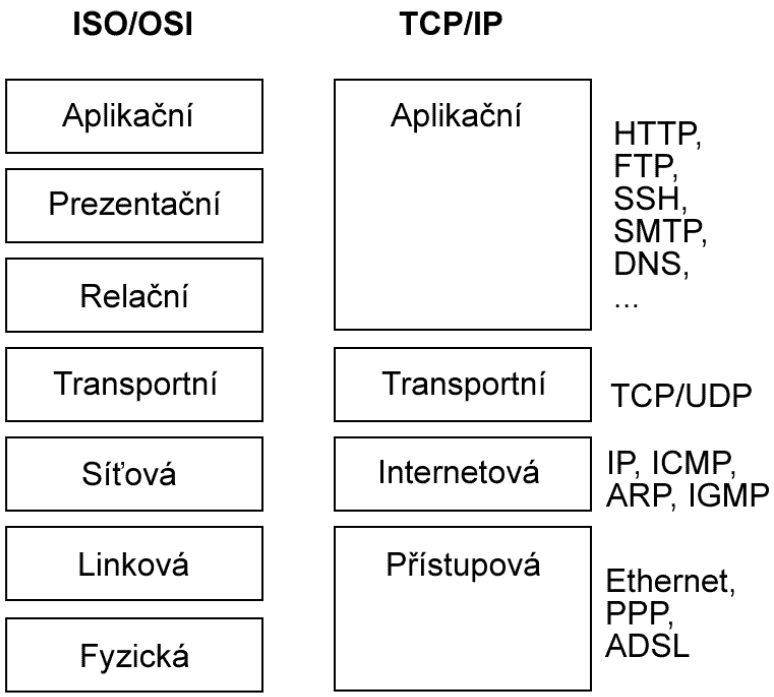
Standardy protokolu Ethernet taktéž definují použité datové rámce, které jsou pro všechny verze stejné.

1. Preambule - 7 oktetů - synchronizace hodin
2. SFD - 1 oktet - Start of Frame Delimiter - začátek rámce
3. MAC cíl - 6 oktetů - fyzická adresa cíle
4. MAC zdroj - 6 oktetů - fyzická adresa zdroje
5. Typ/délka - 2 oktety - typ vyššího protokolu (Ethernet 2) / délka dat (IEEE 802.3)
6. Data - 46-1500 oktetů - data pro vyšší vrstvu, doplněno na minimální délku
7. FCS - 4 oktety - Frame Control Check - kontrolní součet CRC32
8. Mezera mezi rámci - 12 oktetů

Obr. 2.3: Datový rámec protokolu Ethernet

Samotný význam jednotlivých polí rámce může být v některých aspektech odlišný v různých verzích, velikosti polí jsou však vždy zachovány. Tím je zaručena kompatibilita. Data jsou průchodem mezi jednotlivými vrstvami postupně zapouzdřována hlavičkami konkrétní vrstvy.

Protokoly z rodiny TCP/IP jsou nejčastěji používanými síťovým protokoly nad sítí Ethernet. Tyto protokoly mají vlastní komunikační vrstvy lišící se od standardu ISO/OSI, ze kterého však vychází.



Obr. 2.4: Model ISO/OSI a TCP/IP

Protokol IP zprostředkovává síťovou vrstvu a umožňuje komunikaci mezi jednotlivými oddělenými sítěmi. Také definuje podobu datového paketu. Na této vrstvě jsou dostupné i další síťové protokoly, např. ARP, ICMP nebo IGMP.

Nad protokolem IP naslouchá transportní vrstva, která podporuje dva typy přenosů: spojový TCP a nespojový UDP. Protokol TCP vytváří virtuální spojení mezi komunikujícími uzly a zajišťuje doručení paketu. Protokol UDP doručení nezajišťuje (o případné přeposlání chybějícího paketu se musí postarat vyšší vrstva), ale jeho nespornou výhodou je minimální režie vhodná např. pro přenos hlasu, kdy ztráta paketu nemá podstatný vliv na kvalitu hovoru. UDP transport reálně přidává do IP komunikace pouze číslo portu komunikující aplikace.

Podrobněji se této problematice věnuje například publikace *Velký průvodce protokoly TCP/IP a systémem DNS* [4].

2.3.2 RS-232 (UART/USART)

Rozhraní RS-232 umožňuje sériovou komunikaci 2 zařízení (Point-to-Point). Standard RS-232C definuje pořadí přenášených bitů od LSB a úrovně na sběrnici. Logickou nulu představuje hodnota napětí -5 V až -15 V a jedničku 5 V až 15 V. Komunikaci lze navázat pomocí 3 vodičů: příjem RxD, vysílání TxD a uzemnění. Dále lze použít několik dalších vodičů pro řízení přenosu.

RS-232 bývá realizováno komponentou UART (Universal Asynchronous Receiver/Transmitter). V tomto případě je komunikace asynchronní a každý vyslaný byte je třeba synchronizovat. Synchronizace probíhá podle sestupné hrany prvního bitu, tzv. start bitu. Dále je vyslána datová zpráva (nejčastěji o velikosti 1 byte) následovaná paritním bitem a stop bitem. Pokud je třeba z nějakého důvodu přenos pozastavit, posílá přijímací strana signál “logická nula” po dobu 100 - 600 ms.

Vzhledem k nutnosti synchronizovat komunikaci s každým zaslaným bytem není tento typ přenosu vhodný pro přenos většího množství dat.

Pro tyto účely je však možné použít synchronní variantu USART (Universal Synchronous/Asynchronous Receiver/Transmitter). V tomto případě je použit další vodič s hodinovým signálem, podle kterého se přenos synchronizuje.

Přenosové rychlosti se pohybují v rozmezí od 19,2 kbit/s na vzdálenost 15 m do 115,2 kbit/s na vzdálenost 3 m. Toto rozhraní je vytěsňováno sběrnicí USB i z důvodu potřeby generování poměrně vysokých napětí v kontrastu s ostatními napětími používaných v digitální technice. Dalším problémem je, že může dojít k potížím, pokud jsou komunikující zařízení napájena ze zdrojů s různým nulovým potenciálem.

RS-232 je základní rozhraní pro komunikaci s přístroji v akustických laboratořích. K tomuto účelu je navržena rozšiřující karta obsahující několik galvanicky oddělených sériových portů. Samotná základní deska obsahuje jedno klasické rozhraní RS-232, vyvedené na standardní konektor Canon 9, a jedno sériové rozhraní využitelné k ladění, které je vyvedené na pin header a pracuje s úrovněmi TTL.

Podrobnější specifikaci a použití této sběrnice lze nalézt v přehledu na serveru HW.CZ [34].

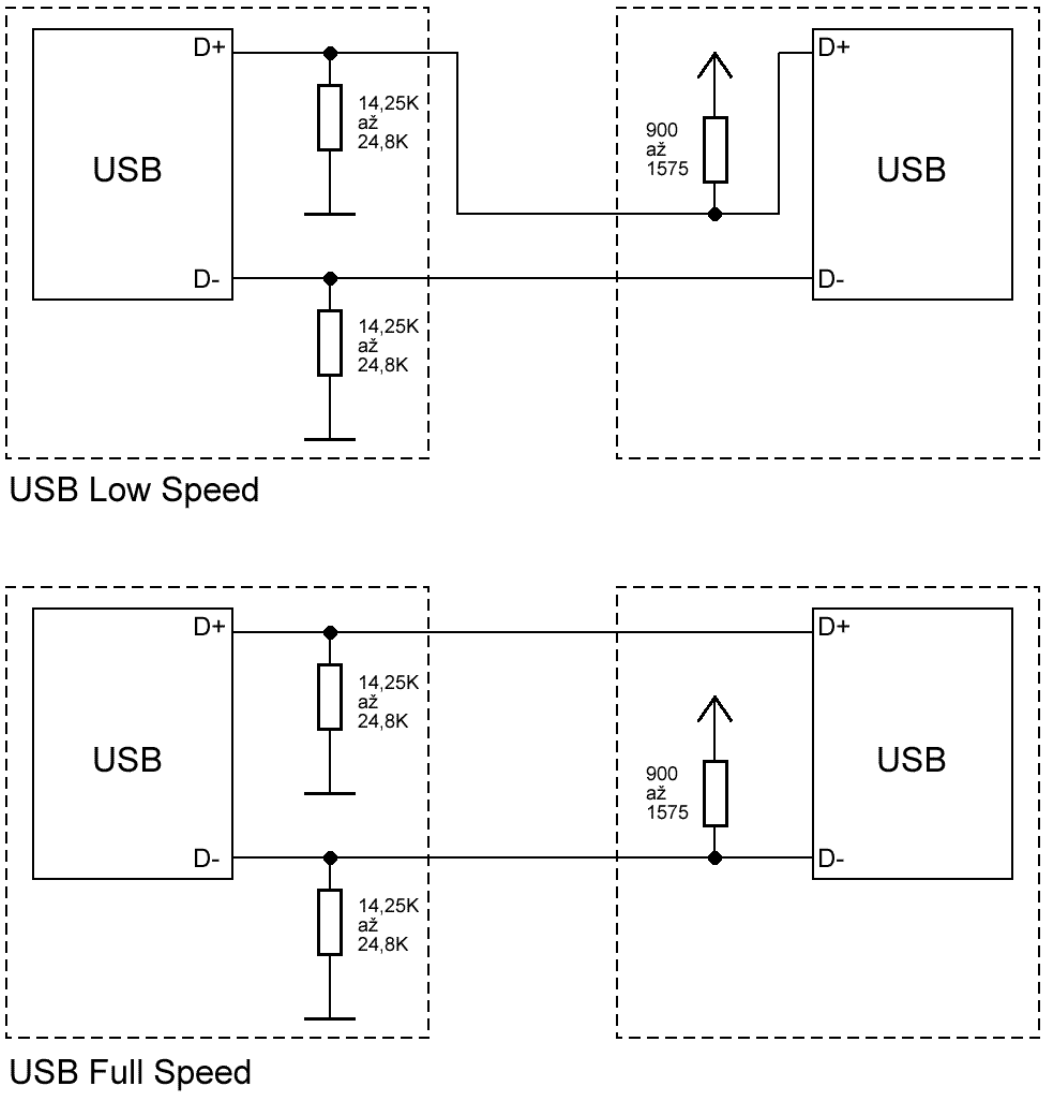
2.3.3 USB

USB (Universal Serial Bus) je sběrnice, jež na vzdálenost do 5 m umožnuje komunikovat rychlostí 1,5 až 480 Mbit/s (verze 2.0). Sběrnice je čtyřvodičová a skládá se z vlastního napájení 5 V, kroucené dvojice datových vodičů (DATA+, DATA-) a uzemnění. Nejmodernější standard USB 3.0 používá již čtyři datové vodiče a slibuje přenosovou rychlost až 5 Gbit/s. Následující popis se věnuje pouze verzím 1.1 a 2.0.

Na sběrnici je možné připojit až 127 zařízení. Po připojení zařízení dochází k inventarizaci sběrnice a následně k enumeraci, kdy připojené zařízení získá adresu. Sběrnice je řízena zařízením zvaným “host controller”. Každý přenos začíná “host controller” vysláním paketu s informacemi o typu a směru přenosu a příjemci - tzv. tokenu. Následuje přenos dat ukončený handshake paketem, který informuje o úspěšném příjmu.

Jsou definovány 3 základní rychlosti, které se v detailech liší i ve fyzické vrstvě:

- High Speed - 480Mbits/s (přidána v USB 2.0)
- Full Speed - 12Mbits/s
- Low Speed - 1.5Mbits/s



Obr. 2.5: Porovnání zapojení fyzické vrstvy u USB Low a Full Speed

Zařízení *High speed* je připojeno stejným způsobem jako *Full speed*, rychlost je vyjednána softwarově.

Navrhované zařízení podporuje sběrnici USB 2.0 včetně rozšíření protokolu OTG. USB tak může pracovat jak v režimu host, tak v režimu periférie. Mezi těmito typy komunikace lze případně za provozu přepínat a emulovat tak komunikaci Peer-to-Peer. Výměna rolí probíhá softwarově přes protokol HNP (Host Negotiation Protocol). Druhým použitým protokolem je SRP (Session Request Protocol), kdy periferní zařízení může poslat pulz po D+ vodiči k probuzení zařízení typu host.

Vyčerpávající informace o sběrnici USB lze nalézt v publikaci *USB Complete* [1].

2.3.4 SPI

SPI (Serial Peripheral Interface Bus) je jednoduchá sériová synchronní sběrnice, která umožňuje plně duplexní komunikaci jedné řídící master jednotky s dalšími podřízenými slave jednotkami po společné sběrnici.

Tato sběrnice využívá 4 typy signálů uvedené v následujícím přehledu:

- MOSI (Master Out Slave In) - alternativně označováno jako SI nebo SDI
- MISO (Master In Slave Out) - alternativně označováno jako SO nebo SDO
- SCLK (Serial Clock) - alternativně označováno jako SCK
- CS (Chip Select) - alternativně označován jako SSEL (Slave Select)

SPI periférie fungují na principu propojených posuvných registrů. To v praxi znamená, že řídící signál způsobí posun dat v jednotkách master i slave najednou - příjem i vysílání probíhá tedy zároveň. Samotná sběrnice využívá jednosměrné porty s TTL úrovněmi.

Díky své jednoduchosti lze na krátké vzdálenosti dosáhnout až rychlosti 140 Mbit/s. Nevýhodu použití signálu *Chip Select* (a tím i potřebu dalšího vodiče) lze odstranit zřetězením jednotlivých slave zařízení. K přenosu informace k cíli je tak potřebný počet taktů úměrný velikosti posuvného registru a počtu slave zařízení.

Sběnice SPI je v navrhovaném zařízení použita ke komunikaci s SD kartou, případně pro využití přídavné flash paměti.

2.3.5 I_2C

I_2C (Inter Integrated Circuit) je externí sběrnice, jež se v některých aspektech podobá SPI. Stejně jako druhá jmenovaná, i ona obsahuje hodinový signál. V případě I_2C však komunikace probíhá poloduplexně a díky tomu je potřeba pouze datový (SDA - Serial Data) a hodinový (SCK - Serial Clock) vodič. Nevýhodou tohoto přístupu jsou větší nároky na funkce hardwaru - piny musí být schopné přepínat svojí funkcionalitu mezi vstupním a výstupním režimem, což je dnes však již běžnou funkcí. Dále zde není možnost výběru slave zařízení pomocí signálu CS (Chip Select). Každému uzlu je tak přiřazena jedinečná adresa a na rozdíl od předchozích sběrnic je v tomto případě také standardem definován komunikační protokol včetně adresování. Právě díky možnosti softwarového adresování uzlů

byla tato sběrnice vybrána do navrhovaného zařízení pro zajištění komunikace s rozšiřujícími kartami. V případě použití SPI by bylo třeba pro každou kartu vyhradit zvláštní CS signál.

Sběrnice I_2C využívá principu otevřených kolektorů, v klidovém stavu je na sběrnici logická jedna. Je tedy třeba oba signály vybudit pomocí pull-up rezistorů. Veškerou komunikaci řídí zařízení master, které může být pouze jedno. Pro úplnost informací je třeba dodat, že existuje i tzv. režim multimaster, kdy je možné funkci master zařízení převést na jiné zařízení. Pokud chce zařízení typu slave komunikovat, musí k tomu být nejdříve vyzváno *master* zařízením. Běžně se využívá metoda neustálých dotazů k jednotlivým *slave*, zda nepotřebují vysílat. Tato metoda je označována jako polling. Podnět k vysílání může být případně předán jinou cestou (např. dalším vodičem generujícím přerušení).

Master komunikaci zahájí snížením úrovně na datovém vodiči po určitou dobu dle zvolené rychlosti - tzv. start bitem. Dále je vyslána 7-bitová adresa (případně 10-bit) a bit, který označuje, zda se jedná o příjem, či vysílání. Po odeslání adresy a informace o směru komunikace musí slave s cílovou adresou odpovědět (9. bit), že na sběrnici naslouchá. Následuje samotná datová zpráva ukončená stop bitem. Logická úroveň datového vodiče se mění pouze v nízké úrovni hodinového signálu. Data jsou čtena při náběžné hraně, kdy je stav na datovém vodiči ustálený. Výjimku tvoří vysílání start a stop bitů, které probíhá ve vysoké úrovni hodinového signálu.

Každá vysílající stanice hlídá, zda stav sběrnice odpovídá právě vysílanému bitu. Jinými slovy, vysílající stanice kontroluje, zda na sběrnici nedošlo ke kolizi. Pokud se stav odlišuje, je vysílání zastaveno.

7-bitové adresy umožňují v teoretické rovině adresovat 127 zařízení. Prakticky je jich však o něco méně, jelikož část adres je rezervována pro jiné použití. Pro rozsáhlejší sítě je možné využít 10-bitové adresování, kdy se adresa přenáší v prvních 2 bytech. V případě, že samotná přenášená data jsou krátká, může toto adresování degradovat výkon sběrnice.

Specifikace I_2C definuje i standardní přenosové rychlosti. Jejich přehled ukazuje následující tabulka:

Přenosová rychlost	Označení módu
10 kbit/s	low speed mode
100 kbit/s	standard mode
400 kbit/s	fast mode
1 Mbit/s	fast mode +
3,4 Mbit/s	high speed mode

Rozsáhlejší informace o posledních dvou sběrnicích je možné nalézt v článku K. Dudáčka [36].

3 Návrh zařízení

Tato krátká kapitola představí hlavní vlastnosti a použití navrhovaného zařízení.

3.1 Základní vlastnosti S-02x

Navržené zařízení bylo pojmenováno kódem S-02x, kde x značí konkrétní hardwarovou výbavu a konfiguraci. Předmětem této diplomové práce je zařízení označené jako S-02c, které má následující vlastnosti:

- Architektura ARM s jádrem Cortex-M3 realizovaná procesorem STM32F107 od výrobce ST Microelectronics
- Počítáno i s možností využití procesorů vyšší řady STM32F2
- Firmware programován v jazyce C
- Operační systém FreeRTOS
- Podpora protokolů HTTP, ICMP, DHCP
- Napájení 10 - 40 V DC, 200 mA / PoE / USB
- Komunikace s rozšiřujícími moduly přes sběrnici I_2C
- 3× analogový vstup 0 - 3.3 V, 12-bit (vyvedeno na pin header pro další využití)
- 1× analogový výstup (vyvedeno na pin header pro další využití)
- 16× GPIO (2× 8 pinů) - jednu bránu lze přemapovat na CAN
- 1× RS-232
- 1× UART (TTL úrovně)

Prototyp zařízení vyvinutý na vývojové desce je označen kódem S-02a.

3.2 Zapojení do infrastruktury

Primární způsob zařazení S-02 do infrastruktury je jeho zapojení jako plnohodnotného síťového zařízení. Zařízení je obsluhováno vzdáleně z klientského PC pomocí webového prohlížeče, jenž ovládá rozšiřující kartu/y. Zmíněné využití je předmětem této diplomové práce.

3.2.1 Požadavky na prohlížeč

Webové rozhraní má na prohlížeč následující požadavky:

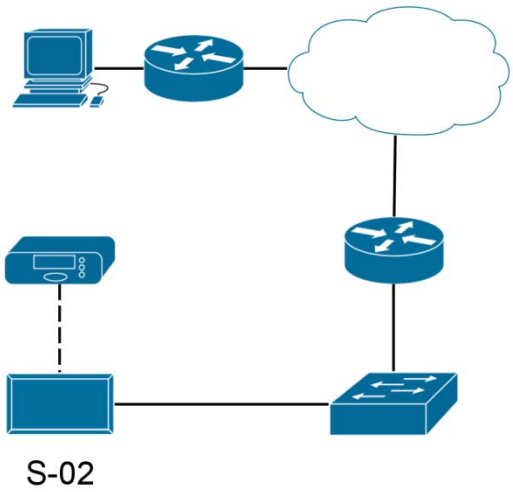
- zapnutá podpora javascriptu
- dostupné připojení k internetu, odkud se stahují externí knihovny (lze je ale zabudovat i přímo do firmware)

3.2.2 Podporované prohlížeče:

Funkčnost uživatelského rozhraní je deklarována v následujících prohlížečích:

- Internet Explorer 7+
- Mozilla Firefox 2+
- Opera 10+
- Google Chrome

Uživatelské rozhraní je funkční i v IE 6 a Firefox 1, nelze však zaručit veškerou jeho plánovanou funkcionalitu.



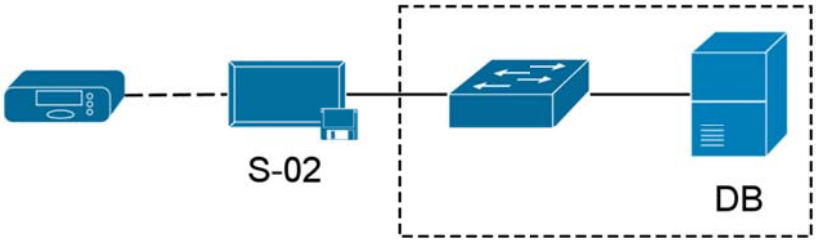
Obr 3.1: Předpokládané zapojení v infrastruktuře

Druhou možností je provozovat pouze samotnou základní desku bez rozšiřujících karet. Toto je možné v případech, kdy pro danou aplikaci postačuje základní výbava desky.

3.3 Další varianty použití S-02 v infrastruktuře

3.3.1 Autonomní sběrač dat

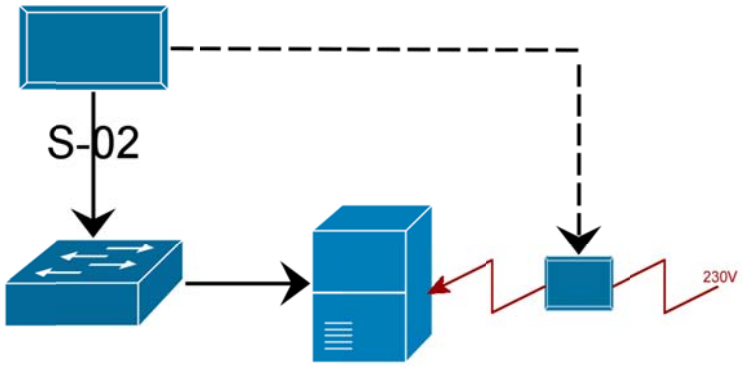
S-02 může fungovat, jako autonomní sběrač dat. V tomto případě S-02 zaznamenává měřená data, která ukládá buď na SD kartu, anebo je případně přeposílá na databázový server.



Obr. 3.2: Zapojení S-02 jako sběrače dat

3.3.2 Ethernetový watchdog[1]

Další využití, které se nabízí, je použití S-02 jako ethernetového watchdogu, kdy zařízení hlídá pomocí protokolu ICMP dostupnost určeného stroje v počítačové síti a v případě nedostupnosti provede jeho restart.



Obr. 3.3: S-02 jako ethernetový watchdog

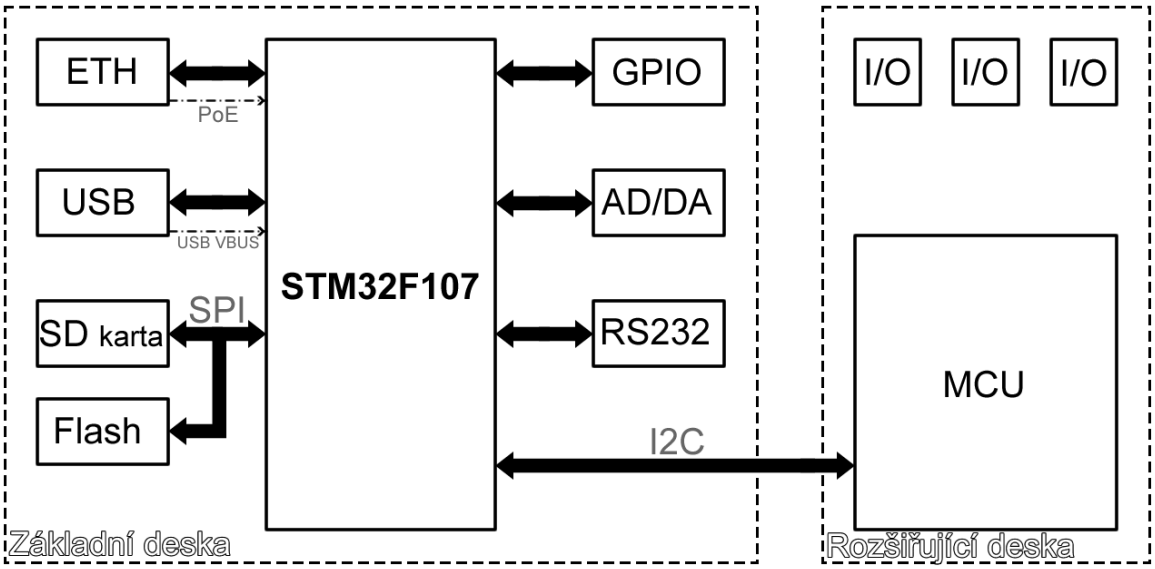
[1] Upravená varianta S-02b je použita v několika technologických místnostech společnosti Sloane Park Property Trust, a.s. (nyní součást UPC Česká republika, s.r.o.), kde hlídá teplotu a umožňuje vzdálený restart síťových zařízení.

4 Hardware

Tato kapitola se věnuje konkrétní hardwarové realizaci navrhovaného zařízení S-02c. Větší prostor je vyhrazen jeho napájení a v souvislosti s tím je podrobněji probrána možnost napájení po Ethernetu.

4.1 Blokové zapojení

Následující obrázek demonstruje základní navržené principy v blokovém zapojení.



Obr. 4.1: Blokové zapojení S-02

4.2 MCU

Tato sekce podrobněji probere konfiguraci mikrokontroléru vybraného pro realizaci zařízení.

Základní součástí zařízení S-02 je 32-bitový mikrokontrolér STM32F107VC s jádrem ARM Cortex-M3. Mikrokontrolér je dodáván v pouzdře TQFP100. K dispozici je i varianta v menším pouzdře TQFP64, která by byla při omezení počtu GPIO pinů pro použití v této práci dostačující. Důvodem pro použití většího pouzdra je jeho kompatibilita s mikrokontroléry vyšší řady STM32F2, jež se v 64-pinové variantě s ethernetovou periférií nevyrábí. Tento model MCU byl vybrán z důvodu své bohaté výbavy, která plní veškeré důležité požadované funkce. Dalšími důvody pro její výběr byla dobrá dostupnost vývojových prostředků a rovněž předchozí kladné zkušenosti autora práce s touto řadou mikrokontrolérů.

Vlastnosti vybraného MCU:

Jádro:

- maximální frekvence jádra 72 MHz (1.25 DMIPS/MHz)
- HW dělení
- násobení v jednom cyklu

Paměť:

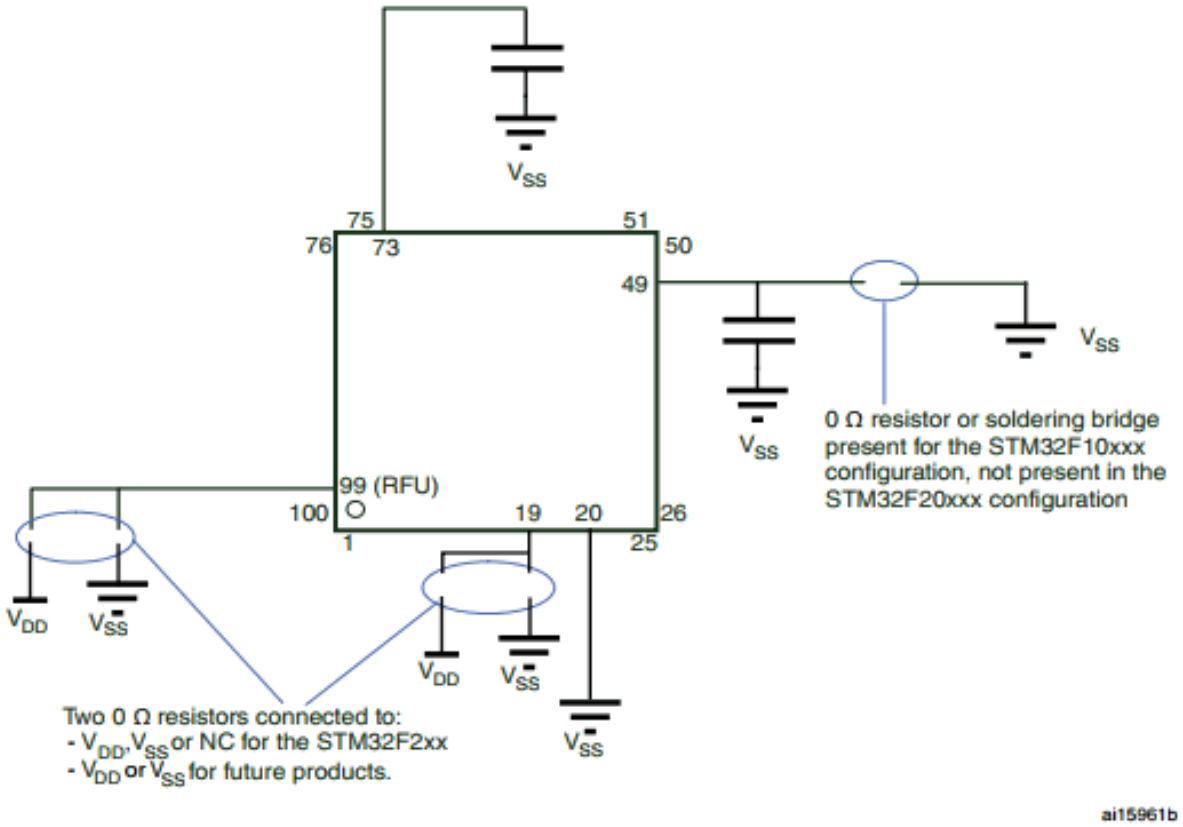
- 64 kB SRAM
- 256 kB Flash

Periférie:

- 2× 12-bitový A/D převodník, 2 Ms/s, 16 kanálů (1× interní teplotní senzor)
- 2× 12-bitový D/A převodník
- 3× USART
- 2× UART
- 3× SPI (z toho 2 s podporou I2S)
- 2× I_2C
- 10/100 Ethernet MAC (4 kB vlastní SRAM)
- USB OTG (1,25 kB vlastní SRAM)
- 2× CAN (512 B vlastní SRAM)
- 7× časovač/čítač
- 1× systémový časovač pro OS
- 2× Watchdog
- 80 I/O portů
- CRC jednotka

Možnosti ladění:

- JTAG
- SWD (Serial Wire Debug)



Obr. 4.2: Návrh kompatibilního zapojení s novou řadou MCU (obrázek převzat z dokumentace výrobce – AN3427 [18])

4.3 Ethernet MAC

S-02 využívá pro přístup k fyzické vrstvě nízkoodběrový phyter DP83848C od výrobce National Semiconductor. Tento phyter podporuje redukované MII rozhraní (RMII), které poskytuje veškerou funkcionalitu standartu IEEE 802.3u pro MII. RMII snižuje náročnost zapojení omezením šířky datového rozhraní ze 4-bitové (nibble) sběrnice (4× 25 MHz) na 2-bitovou (2× 50 MHz). Zároveň je zjednodušeno i ovládací rozhraní.

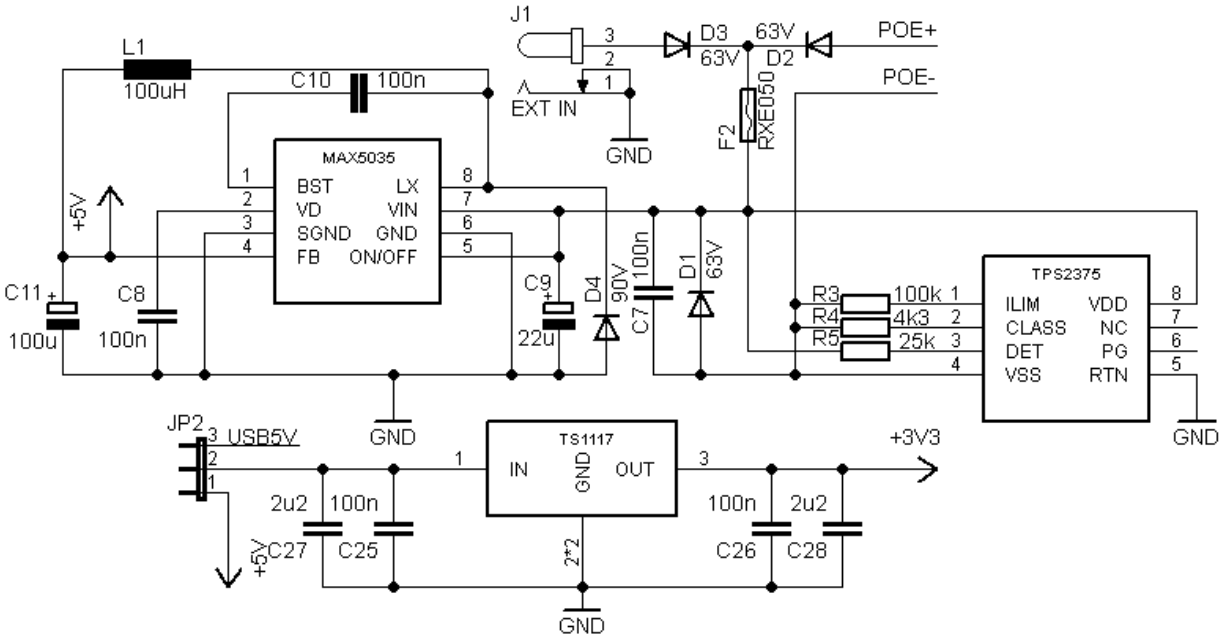
Důvodem pro tyto úpravy byla snaha zjednodušit víceportové aplikace (např. síťové switche), kde redukováním počtu potřebných pinů dochází ke značnému zjednodušení zapojení a k finančním úsporám. Vzhledem k tomu, že není složité připravit si požadovaný 50 MHz kmitočet pomocí PLL v MCU, a k možnosti zjednodušení zapojení, bylo zvoleno použití rozhraní RMII.

Samotné fyzické připojení obstarává ethernetový konektor RJ-45 s integrovanými transformátory a podporou PoE.

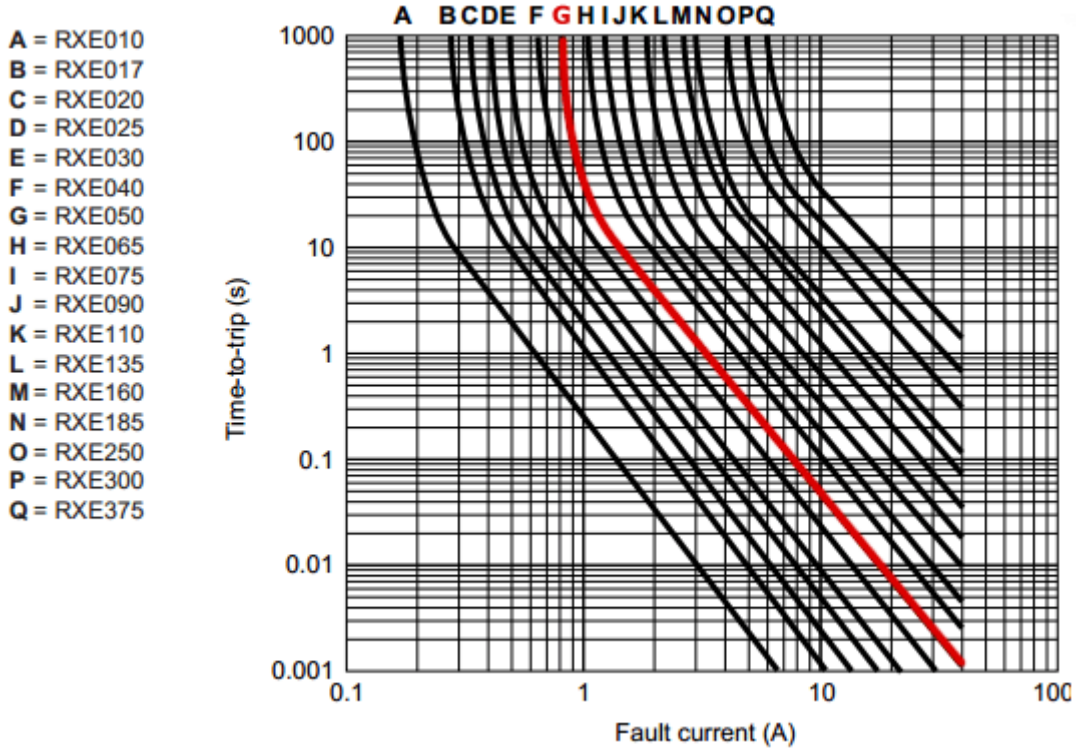
4.4 Napájení

S-02 interně používá dvě hodnoty napětí: 3,3 V a 5 V (pro USB). Abychom získali napětí 5 V, byl použit step-down DC/DC měnič MAX5035B. Z tohoto napětí je pomocí LDO regulátoru TS1117 získáno napětí 3,3 V. Prostřednictvím jumperu lze přepnout vstup LDO regulátoru na napětí 5 V z USB periférie.

Jako zdroj napětí pro step-down DC/DC měnič je použit externí zdroj 10 - 40 V (typicky 12 V), případně PoE. Princip PoE bude podrobněji popsán v následující sekci. Vytěsnění napájení z Ethernetu dle standardu IEEE 802.3af má na starosti obvod TPS2375. Vstup z těchto zdrojů je chráněn polyswitchem RXE050.



Obr. 4.3: Schéma napájecího obvodu S-02



Obr. 4.4: Křivky aktivity vybraného polyswitche (podklady pro obrázek byly převzaty z dokumentace výrobce a upraveny)

4.5 Standard IEEE 802.3af

Standard IEEE 803.af (dále jen standard) definuje možnost napájet spotřebiče (PD - Powered Device) pomocí stávajících datových vodičů. Před tímto standardem bylo využíváno napájení přes volné vodiče v UTP kabelu - tzv. pasivní PoE. Tento postup však nelze použít v gigabitovém Ethernetu, jelikož zde jsou všechny vodiče využívány pro přenos dat.

Standard definuje velikost napájecího napětí na 44 - 57 V, maximální proud 550 mA a typický proud 10 - 350 mA. Při odběrech nad 350 mA je detekováno přetížení. Řídící zdroj napětí (PSE - Power Sourcing Equipment) detekuje podporu PoE a získává od něj identifikaci třídy spotřeby zařízení. Obě zařízení spolu musí komunikovat, proto se tento typ PoE někdy označuje jako aktivní.

V klidovém stavu je napětí na PSE 0 - 2,8 V, při připojení PD je detekován proudový odběr a nastává fáze detekce.

Detekce probíhá měřením terminačního odporu PD, který by měl mít u zařízení podporujících standard hodnotu 25 kΩ. PSE nastaví úroveň měřícího stejnosměrného napětí na 2,8 - 10 V; pokud je na straně PD detekován odpor 24,1 - 26 kΩ, vyhovuje zařízení standardu. Délka měření nepřekračuje 500 ms.

Pokud úspěšně proběhla fáze detekce, je třeba identifikovat třídu spotřeby PD. Identifikace probíhá měřením výstupního proudu při napětí 15,5 - 20 V. Měření probíhá po dobu maximálně 75 ms. Vstupní obvody PD pro identifikaci by proto neměly obsahovat filtry s velkou kapacitou, aby časová konstanta nepřesáhla měřící interval. Třídy spotřeby zařízení ukazuje následující tabulka:

Třída	Proud pro identifikaci	Příkon PD	Popis třídy
0	0 – 4 mA	0,44 – 12,9 W	základní třída
1	9 – 12 mA	0.44 – 3,84 W	velmi nízký výkon
2	17 – 20 mA	3,84 – 6,49 W	nízký výkon
3	26 – 30 mA	6,49 – 12,95 W	střední výkon
4 *	36 – 44 mA	12,95 – 25,50 W	vysoký výkon

* Třída 4 je definována až ve standardu 802.3at, pro 802.3af nelze použít.

Po určení třídy PD dochází k jeho aktivaci. Během aktivace, která by měla trvat do 50 ms, je zvýšeno napětí na 30 - 44 V. Tato hodnota je označována jako UVLO (Under Voltage Lock Out). V této fázi by měl zdroj PD aktivovat samotné zařízení. Během této etapy zároveň dochází k nabíjení filtračních kondenzátorů PD, čímž je omezena proudová špička při zapnutí zařízení, která by mohla překročit dané limity.

Po aktivaci je již PD napájeno plným napětím. V tuto chvíli PSE hlídá minimální odběr, který by neměl klesnout pod 10 mA. V případě poklesu odběru pod uvedenou hodnotu je PD vyhodnoceno jako odpojené. Pokud aplikace vyžaduje menší příkon, je možné napájení udržet pomocí krátkodobých odběrů (MPS - Maintain Power Signature). V tomto případě je po dobu alespoň 75 ms odebírán proud 10 mA a následně je možné po dobu až 250 ms pracovat ve snížené spotřebě.

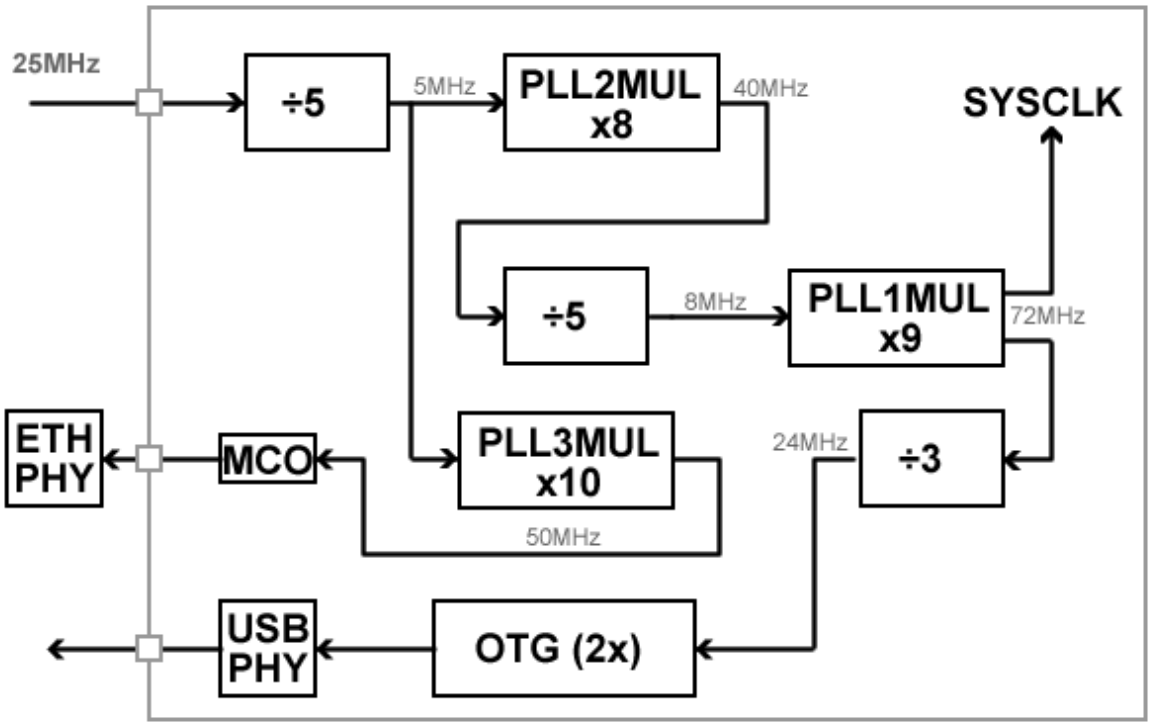
S-02 používá pro detekci třídy odběru odpor 4,4 kΩ, náleží tedy do základní třídy 0.

4.6 Časování

Jelikož v předchozích sekcích byly probrány technologie spojené s rozhraním Ethernet, je nyní vhodné objasnit realizaci zdrojů hodinového signálu, které RMII i další periférie potřebují ke svému fungování.

Základním zdrojem hodinového signálu je 25 MHz krystal. Potřebné hodinové signály mají frekvenci 72 MHz pro jádro, 50 MHz pro Ethernet (v RMII módu) a 48 MHz pro případné využití USB. Všechny jmenované frekvence jsou odvozeny od základních 25MHz pomocí integrovaných PLL.

Signál 25 MHz je prostřednictvím děličky PREDIV2 vydělen 5. Výsledný signál je následně za použití PLL3MUL vynásoben 10×. Získaný signál 50 MHz je vyveden na MCO pin a je použit jako zdroj časové základny pro phyter. Výstup z PREDIV2 je dále vynásoben 8× (PLL2MUL) a opětovně vydělen 5 (PREDIV1). Výsledný signál 8MHz je vynásoben 9× v násobičce PLLMUL a použit jako zdroj hodin pro jádro (72 MHz). USB periférie sama interně násobí frekvenci vstupního signálu 2× a konfigurací předděličky této periférie je frekvence vydělena 3 na požadovaných 48 MHz.



Obr. 4.5: Přehled generování hodinových signálů v S-02

S-02 je osazen i zdrojem hodinového signálu pro periférii reálného času RTC. Jako zdroj je zde použit krystal o frekvenci 32,768 kHz.

Mikrokontroléry STM32 dále obsahují interní 8 MHz RC oscilátor, který je však, vzhledem k jeho závislosti na teplotě, vhodný jen pro nenáročné aplikace.

4.7 Restartování a bootování

S-02 disponuje dvěma tlačítky, která umožňují restartování zařízení a zavedení bootloaderu. Pomocí tohoto bootloaderu lze přehrát firmware.

Restartovací tlačíko je navržené jako bezzákmitové s časovou konstantou 1 ms. Výrobce MCU udává, že pro restart je třeba na pinu NRST logická nula alespoň po dobu 20 μs, pro phyter je tato doba 1 μs. Pokud je během startu drženo tlačítko BOOT, je zaveden bootloader od výrobce, který umožnuje do MCU nahrát program pomocí periférií USART1, USART2, USB OTG nebo CAN2.

4.8 PCB

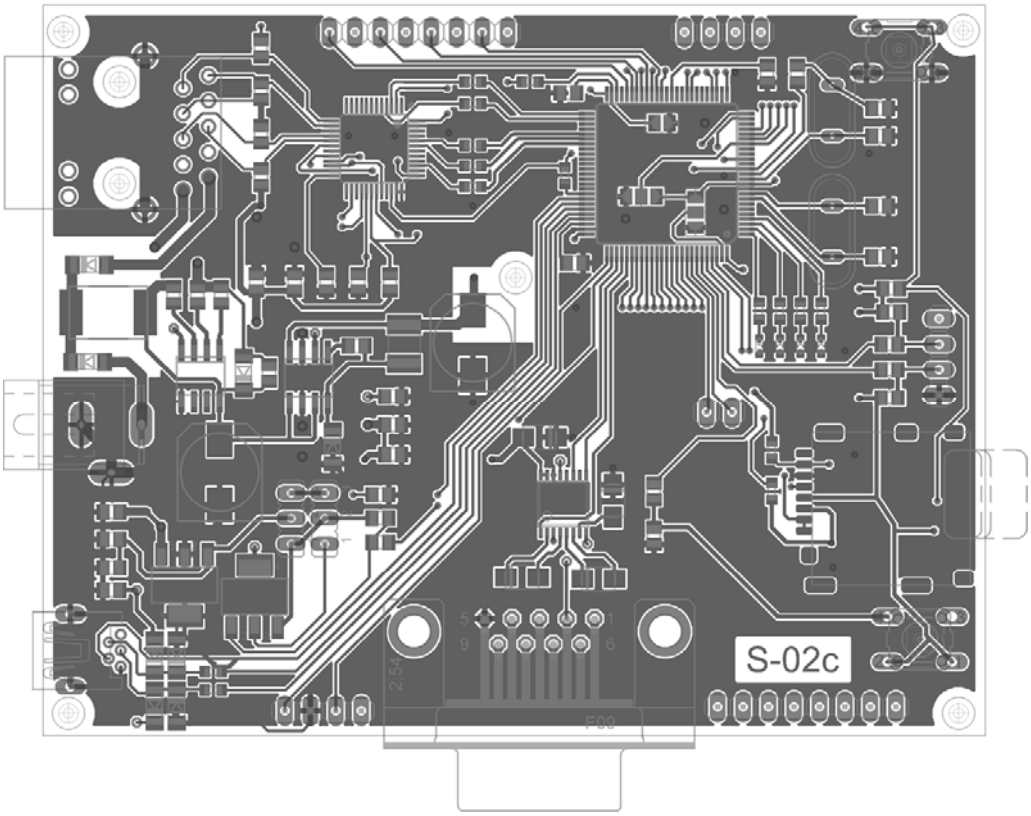
Tato pasáž pojednává o konkrétním návrhu desky plošného spoje pro variantu S-02c.

Varianta S-02c je navržena na dvouvrstvé desce o rozměrech 7,3 × 9,4 cm s pěti montážními otvory. Na desku A4 je možné umístit až 8 těchto PCB. Pro návrh byl použit návrhový systém Eagle Freeware společnosti CadSoft. Použitá neplacená verze je omezena na návrh desky plošného spoje o maximálních rozměrech 100 × 80 mm maximálně o 2 vrstvách, což pro potřeby tohoto projektu dostačuje. Toto návrhové prostředí bylo zvoleno z důvodu autorových dlouhodobých zkušeností s tímto návrhovým prostředím.

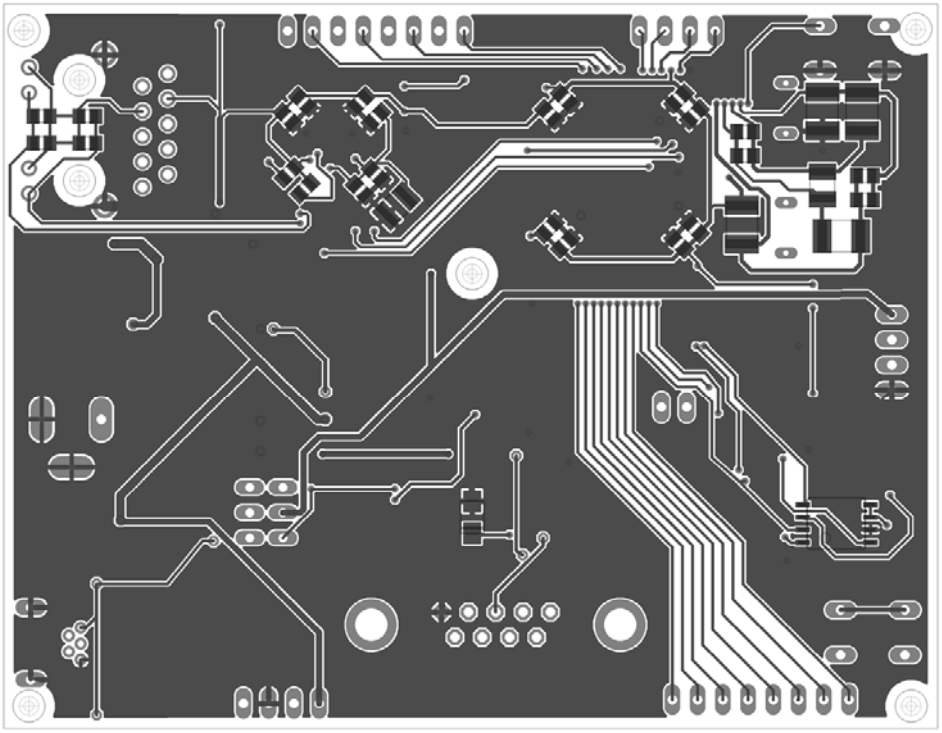
Klíčové součástky jsou umístěny na vrchní straně spoje, kde se zároveň nachází i většina důležitých signálových vodičů. Zvláštní pozornost byla věnována propojení MCU a phyteru, kdy byly při návrhu upřednostněny datové vodiče pracující na vyšších frekvencích. Tyto spoje jsou realizovány v co nejkratší délce a s minimem prokovených otvorů, na rozdíl od nenáročných logických signálů typu Chip Select, Reset a dalších. Pod MCU jsou umístěny propojky pro zvolení typu mikrokontroléru (řada STM32F1 nebo STM32F2) před osazením.

Spodní strana spoje slouží převážně pro rozvod napájení a například u MCU a phyteru je využita k umístění blokovacích kondenzátorů do těsné blízkosti integrovaného obvodu.

Na obou stranách je použita rozlitá měď pro zvýšení odolnosti signálů vůči rušení.

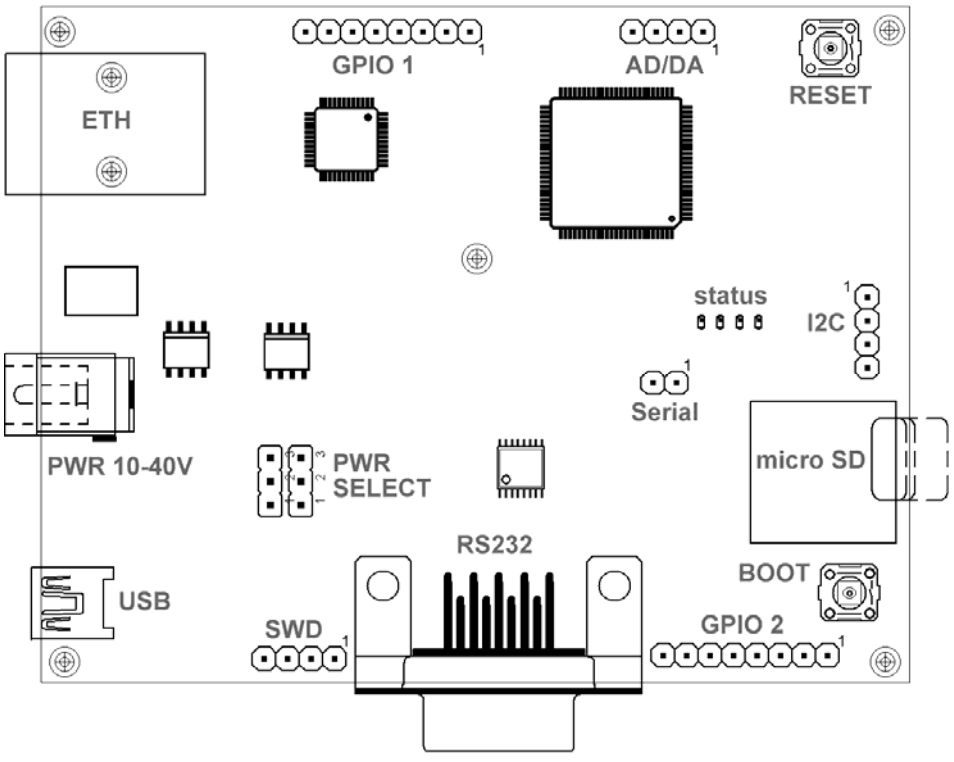


Obr. 4.6: Vrchní strana PCB



Obr. 4.7: Spodní strana PCB

4.8.1 Zapojení konektorů a propojek



Obr. 4.8: Rozmístění konektorů

Následující tabulky popisují zapojení jednotlivých konektorů:

GPIO 1		**GPIO 2**	
1	PE8	1	PD0
…	…	…	…
8	PE15	8	PD7

AD/DA		**I2C**		**SWD**	
1	ADC1_3	1	+5V	1	+3V3
2	DAC1	2	I2C1_SDA	2	SWDCLK
3	ADC2_14	3	I2C1_SCL	3	GND
4	ADC2_15	4	GND	4	SWDIO

Serial		PWR_SELECT (nastavení pro obě propojky)	
1	USART1_RX	1-2	USB
2	USART1_TX	2-3	PoE/externí zdroj

5 Operační systém

Jelikož má být zařízení, kterým se zabývá tato diplomová práce, univerzální, je třeba zvážit použití operačního systému a stanovit jeho výhody a nevýhody.

Nevýhodou použití operačního systému je vyšší výpočetní náročnost. Tento handicap je však v našem případě převýšen výhodami oproti běžnému použití jedné nekonečné smyčky programu.

Rozdělení programu na více paralelních smyček přináší výhodu v abstrakci řešeného problému. Problém je rozdělen do několika nezávislých procesů, což umožňuje jednodušeji provádět úpravy v jednotlivých procesech nebo přidávat další, aniž by to mělo vliv na ostatní procesy.

Operační systém se musí primárně starat o plánování procesů, řízení přístupů ke zdrojům a obsluhu přerušení. Pro řídící aplikace by měl pracovat v reálném čase.

K dispozici je velké množství realtime operačních systémů (RTOS), které jsou vhodné pro embedded aplikace. Operační systémy se liší svými funkcemi, náročností - overheadem, podporou platforem a licencí.

Pro použití v S-02 byl preferován jednoduchý RTOS portovaný na platformu Cortex-M3 s otevřenou licencí.

V následující části bude stručně probráno několik možných kandidátů pro použití v zařízení S-02.

5.1 Přehled možných OS pro použití v S-02

5.1.1 Linux

Operační systém Linux je synonymem pro otevřený systém podporovaný mnoha platformami.

Vzhledem k použitému procesoru je však pro zařízení S-02 nevhodný, protože vyžaduje MMU (Memory Management Unit). MMU jednotka se stará o ochranu paměti a především o mapování virtuální paměti na skutečnou. Upravené linuxové jádro je možné spustit i na

procesorech bez MMU (např. projekt uClinux), nicméně i takto upravený systém je stále poměrně náročný na operační paměť RAM - reálně vyžaduje alespoň 4 MB.

Použití v S-02 by vyžadovalo přídavnou SRAM a kvůli absenci MMU by nebyly využity všechny benefity, které Linux nabízí.

5.1.2 ARMexe

ARMexe je jednoduchý a kompaktní preemptivní RTOS určený pro použití s Luminary Micro Stellaris Cortex-M3 a Keil uVision, které nabízí i jeho konfigurátor. Portace na platformu STM32 by měla být možná. Bohužel při jeho studiu byly zjištěny nejasnosti v licencování. Dle licence je totiž jeho komerční použití možné pouze s písemným svolením autora. Z tohoto důvodu nebyl tento systém podrobněji zkoumán.

5.1.3 eCOS

Dalším otevřeným RTOS je eCOS. Jedná se o plnohodnotný operační systém kompatibilní se standardem POSIX, s vrstvou HAL a ovladači pro množství běžných periférií. Jádro systému potřebuje minimálně 64 kB operační paměti. Jelikož S-02 obsahuje právě 64 kB operační paměti, nelze zaručit optimální výkon tohoto systému, neboť nedostatek operační paměti má velký dopad na výkon TCP/IP stacku. Z tohoto důvodu nebyl vybrán.

5.1.4 FreeRTOS

FreeRTOS není operačním systémem v pravém slova smyslu. Je pouze jádrem s plánovačem úloh, správou paměti a nástroji pro synchronizaci vláken. V určitých případech to však může být výhodou, protože systém zůstává minimalistický. Tento RTOS byl právě kvůli nízkým nárokům vybrán pro S-02 a bude v této práci podrobněji popsán v následující kapitole.

6 FreeRTOS

Jak již bylo výše uvedeno, tato kapitola se bude věnovat operačnímu systému FreeRTOS. Vzhledem k tomu, že se jedná o základní stavební kámen firmwaru navrhovaného zařízení, budou podrobněji popsány i teoretické základy fungování operačních systémů a konkrétní řešení ve FreeRTOS.

FreeRTOS je minimalistický operační systém pracující v reálném čase. Je portovaný na více než 30 platforem a jeho licence umožňuje i publikaci s uzavřeným kódem. Podporuje kooperativní i preemptivní multitasking. Při kooperativním multitaskingu je pro vlákno vyhrazen veškerý procesorový čas, který je uvolněn až po dokončení jeho činnosti. Tímto způsobem může dojít ke kolapsu celého systému v případě, že pracující proces havaruje. Také zde existuje určitá latence mezi skončením procesu a předáním procesorového času dalšímu procesu. Výhodnější proto je preemptivní multitasking, kdy je procesorový čas plně pod kontrolou operačního systému, jenž jej sám přiděluje a odebírá procesům. Na tomto místě by mělo být ozřejměno, že v koncepci tohoto RTOS označuje proces jedno běžící vlákno - úlohu. Reálný proces je soubor vláken pracujících v odděleném paměťovém prostoru, zde však z důvodu jednoduchosti tohoto RTOS žádný oddělený paměťový prostor není využíván.

Pro úplnost je na místě zmínit, že kromě FreeRTOS existují ještě další dvě jeho komerční distribuce:

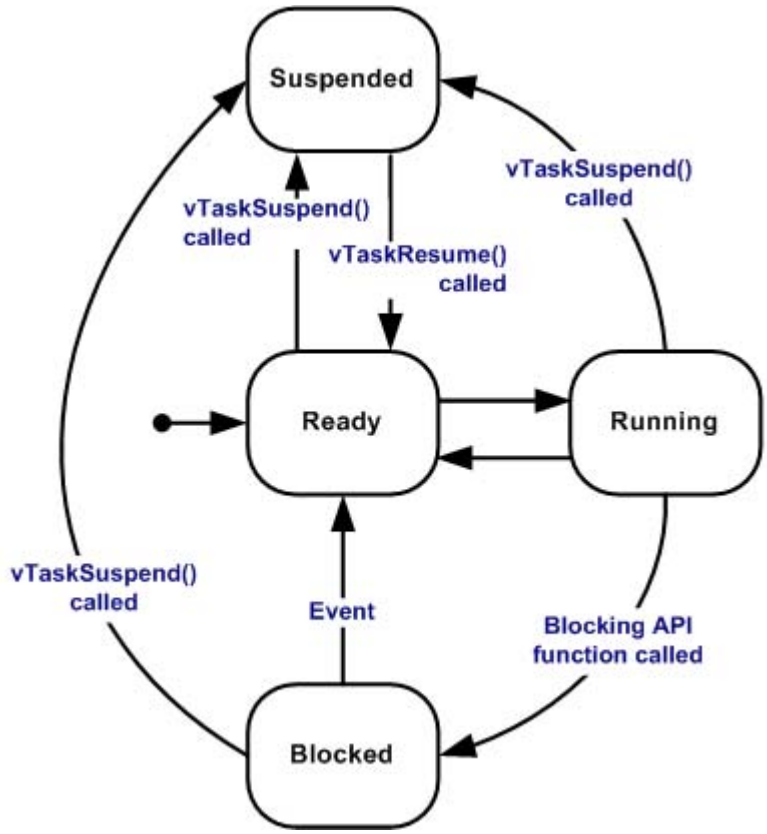
- OpenRTOS - obsahuje ovladače pro různé periferie a protokoly (TCP/IP, USB)
- SafeRTOS - verze certifikovaná pro bezpečnostní aplikace

Obr. 6.1: Oficiální logo systému FreeRTOS

V následujících odstavcích bude probrána teorie jednotlivých funkcionalit operačního systému FreeRTOS.

6.1 Plánovač úloh

Úloha je ve FreeRTOS definována několika charakteristikami: svým názvem, prioritou, velikostí stacku, vstupními parametry, ukazatelem na prováděnou funkci a samotným výkonným kódem. Úlohy se mohou nacházet ve stavech zobrazených na následujícím obrázku, který je převzatý z dokumentace FreeRTOS.



Obr. 6.2: Stavový diagram plánovače ve FreeRTOS (obrázek převzat z dokumentace FreeRTOS [28])

Vytvořená úloha se nachází ve stavu *Ready*. Po přidělení procesorového času plánovačem přejde do stavu *Running*, ze kterého se může dostat do stavu *Blocked* v případě, že musí čekat na nějakou událost. Pokud událost nastane, opět se navrací do stavu *Ready* a čeká na přidělení prostředků plánovačem. Události mohou být časové (to znamená zablokování vlákna na určitý čas - `vTaskDelay`) a synchronizační (čeká se na přístup k nějakému zdroji). Posledním stavem je stav označovaný jako *Suspended* (úloha je neaktivní), který je ovládán API funkcemi z programu.

Preemptivita spočívá v přerušování úloh s nižší prioritou, pokud existuje ve stavu *Ready* úloha s vyšší prioritou. Počet priorit není ve FreeRTOS omezen a rozsah jejich hodnot se pohybuje od nejnižší priority 0 do nejvyšší priority definované konstantou `configMAX_PRIORITIES`. Maximální počet priorit by měl odpovídat reálným požadavkům, jelikož se vzrůstajícím počtem priorit roste i spotřeba operační paměti. Mezi procesy se stejnou prioritou je procesorový čas přiřazován metodou *Round Robin*.

FreeRTOS automaticky vytváří úlohu *IdleTask* s nejnižší prioritou (`tskIDLE_PRIORITY`). Tato úloha je zodpovědná za čištění paměti, pokud v aplikaci dynamicky vytváříme a odstraňujeme úlohy. Její funkcionalitu lze rozšířit `funkcí vApplicationIdleHook`, můžeme např. nastavit úsporný režim.

Komunikaci mezi úlohami zajišťují fronty. Fronta je zásobník typu FIFO a je definována datovým typem a hloubkou. Úloha může být při čtení z fronty blokována v případě, kdy ve frontě žádná data nejsou, nebo pokud se z ní pokouší číst úloha s vyšší prioritou. Při zápise může dojít k blokování, jestliže je fronta již zaplněna, nebo pokud do ní zapisuje úloha s vyšší prioritou. Pro operace s frontou lze specifikovat maximální čas blokování úlohy, který je možné nastavit i jako nekonečný. Přístup k frontám je atomický.

Práci s frontou zajišťují funkce `xQueueSend` a `xQueueReceive`.

6.2 Řízení přístupu ke zdrojům

Další funkcí, kterou operační systém musí zajišťovat, je řízení přístupu ke zdrojům. Problém exkluzivního přístupu ke zdrojům lze v FreeRTOS řešit několika způsoby. Základní metody jsou metoda uspání plánovače, binární semafor, mutex a řízení přístupu pomocí tzv. Gatekeeper úlohy. Tyto postupy jsou podrobněji popsány níže.

6.2.1 Uspání plánovače

Uspání plánovače je nejjednodušším způsobem řízení přístupu ke zdrojům. Pokud úloha vstupuje do sekce, ke které je vyžadován exkluzivní přístup, je plánovač pozastaven a veškerý procesorový čas je přidělen úloze. Po dokončení práce je funkce plánovače opět obnovena. Při tomto přístupu může dojít k zablokování celé aplikace.

6.2.2 Binární semafor

Pokud chce úloha přistupovat ke zdroji, musí požádat o token (semafor). Jestliže jej dostane, může úloha vykonat požadované operace a vrátit token zpět. V opačném případě úloha přejde do stavu *Blocked*. Při použití semaforu může dojít k tzv. inverzi priorit. Inverze priorit nastane, pokud úloha s nízkou prioritou přistupuje ke zdroji a dostane token, díky němuž pak ke zdroji nemůže přistupovat úloha s vysokou prioritou. Tento postup je správný, ale mohou se objevit problémy, jestliže začne pracovat úloha se střední prioritou, která ke zdroji nepřistupuje. V tomto případě je pak úloha s nízkou prioritou blokována a úloha s vysokou prioritou tak musí čekat na dokončení činnosti úlohy se střední prioritou a zároveň i na uvolnění zdroje úlohy s nízkou prioritou. Může však nastat ještě problematičtější situace v případě, že dvě úlohy potřebují exkluzivně přistupovat ke dvěma zdrojům. Pokud každá vlastní token na jeden ze zdrojů, tak se vzájemně blokují před dokončením činnosti a dochází k zablokování systému - tzv. deadlocku.

Operace s tokeny mají na starosti funkce `xSemaphoreGive` a `xSemaphoreTake`.

6.2.3 Mutex

Problém s inverzí priorit je ve FreeRTOS řešen pomocí mutexu. Rozdíl mezi mutexem a semaforem je v tom, že mutex musí vrátit ta samá úloha, která si jej vzala. Tímto způsobem je ošetřena inverze priorit. Zároveň je tak vyloučeno použití mutexu při obsluze přerušení. Pokud úloha s nízkou prioritou získá token - mutex a úloha s vysokou prioritou se pokouší o přístup ke zdroji, zdědí úloha s nízkou prioritou prioritu druhé úlohy, aby nemohlo dojít k jejímu zablokování úlohami s nižší prioritou. Stále však může dojít k problémům při pokusu o přístup ke dvěma zdrojům ze dvou úloh.

6.2.4 Gatekeeper úloha

Nejbezpečnější řešení exkluzivního přístupu představuje tzv. Gatekeeper úloha. Zdroj přestává být globálním a je k němu umožněn přístup pouze a jedině z této úlohy. Ostatní úlohy ke zdroji přistupují skrze FIFO frontu, kterou Gatekeeper úloha postupně zpracovává. Toto řešení vyžaduje tedy další úlohu a je proto i náročnější na prostředky.

6.3 Časování OS

V této a následující části práce budou podané informace již prakticky zaměřené na FreeRTOS a jeho konfiguraci.

Operace plánovače probíhají v intervalech systémového časovače. Při spuštění plánovače začíná zvyšování proměnné `xTickCount` od nuly. Hodnota této proměnné tedy určuje systémový čas. Jako generátor systémového času je u procesorů řady STM32 použita periférie SysTick (Cortex System Timer).

Frekvence systémového časovače (tzv. jiffy) je nastavitelná konstantou `configTICK_RATE_HZ` v nastavení FreeRTOS.

Na každý tik systémového časovače můžeme zareagovat pomocí funkce `vApplicationTickHook`. Můžeme zde například provést kontroly běžících aplikací.

6.4 Základní konfigurace FreeRTOS

V této části je ukázána konkrétní konfigurace z S-02. FreeRTOS se konfiguruje pomocí hlavičkového souboru *FreeRTOSConfig.h*.

```
/* Zapíná preemptivní plánovač */
#define configUSE_PREEMPTION       1
/* Umožňuje použití funkce vApplicationIdleHook */
#define configUSE_IDLE_HOOK        0
* Umožňuje použití funkce vApplicationTickHook */
#define configUSE_TICK_HOOK        0
/* Definuje takt procesrou pro další výpočty */
#define configCPU_CLOCK_HZ         ( ( unsigned portLONG ) 72000000 )
/* Definuje frekvenci systémového časovače */
#define configTICK_RATE_HZ         ( ( portTickType ) 100 )
/* Maximální počet priorit úloh */
#define configMAX_PRIORITIES       ( ( unsigned portBASE_TYPE ) 5 )
/* Velikost zásobníku Idle úlohy */
#define configMINIMAL_STACK_SIZE   ( ( unsigned portSHORT ) 128 )
/* Definuje velikost použité operační paměti - lwIP stack používá vlastní
*/
#define configTOTAL_HEAP_SIZE      ( ( size_t ) ( 18 * 1024 ) )
/* Maximální délka jména úlohy */
#define configMAX_TASK_NAME_LEN          ( 16 )
/* Umožňuje sledovat chod systému */
#define configUSE_TRACE_FACILITY   0
```

```
/* Definuje 16-bitový systémový časovač místo 32-bit = vyšší výkon na 8 a
16bit MCU */
#define configUSE_16_BIT_TICKS           0
/* Upravuje chování idle úlohy, pokud jsou další úlohy se stejnou prioritou
- aby se idle úloha mohla vzdát funkce, musí začít pracovat, čímž ubírá
procesorový čas další úloze */
#define configIDLE_SHOULD_YIELD          1

/* Typ detekce přetečení zásobníku
1 = jádro kontroluje, zda při zápisu do stacku bude nejhlubší prvek stále
ve vyhrazeném rozsahu
2 = při vytvoření je zásobník naplněn známou hodnotou a je kontrolován
obsah posledních 16 bytů */
#define configCHECK_FOR_STACK_OVERFLOW   2

/* Nastavení vlastností plánovače */

/* Umožňuje používat  korutiny */
#define configUSE_CO_ROUTINES            0
#define configMAX_CO_ROUTINE_PRIORITIES ( 2 )

/* Použití Mutexů */
#define configUSE_MUTEXES                1
/* Semafory mohou být vícenásobné (nejen hodnoty 0/1) */
#define configUSE_COUNTING_SEMAPHORES    0
#define configUSE_ALTERNATIVE_API        0
#define configUSE_RECURSIVE_MUTEXES      1
#define configQUEUE_REGISTRY_SIZE        0
#define configGENERATE_RUN_TIME_STATS    1

/* Aktivace API funkcí */
#define INCLUDE_vTaskPrioritySet         1
#define INCLUDE_uxTaskPriorityGet        1
#define INCLUDE_vTaskDelete              1
#define INCLUDE_vTaskCleanUpResources    0
#define INCLUDE_vTaskSuspend             1
#define INCLUDE_vTaskDelayUntil          0
#define INCLUDE_vTaskDelay               1
```

Pro použití FreeRTOS v projektu stačí načíst následující hlavičkové soubory. Můžeme případně načíst také další hlavičkové soubory s funkcemi RTOS, které chceme využívat.

```
#include "FreeRTOS.h"
#include "task.h"
#include "queue.h"
```

Dále je třeba do projektu přidat jeden ze souborů *heap_1.c*, *heap_2.c*, nebo *heap_3.c*.

Tyto soubory určují, jakým způsobem bude FreeRTOS alokovat paměť. Varianta *1* je nejjednodušší a hodí se pro případy, kdy se dynamicky nevytváří nová vlákna, protože neumožňuje alokovanou paměť uvolnit. Pro případ dynamicky vytvářených vláken je vhodná varianta *2*, která využívá algoritmus *best-fit*. Varianta *3* využívá standardních funkcí `malloc` a `free` a je vhodná pro architektury řady x86. Jelikož jsou v S-02 dynamicky vytvářena vlákna webserveru, byla pro něj použita varianta *2*.

6.5 FreeRTOS v S-02

V S-02 je použita aktuální verze FreeRTOS 7. V základní verzi firmware jsou v systému vytvořeny 4 základní úlohy:

- ovládání stavových led
- posílání ladících informací po sériové lince (tato úloha je v produkční verzi nepoužitá)
- komunikace po I_2C
- úloha vytvořená TCP/IP stackem

Použitý lwIP stack si vytváří autonomně další vlákna pro aplikace, které v rámci něj běží.

V případě použití dalších periférií na základní desce je velmi jednoduché přidat další úlohy pomocí funkce `xTaskCreate`.

7 TCP/IP stack

Tato kapitola pojednává o výběru vhodného TCP/IP stacku pro S-02. Dále obsahuje základní informace o vybraném lwIP a jeho konfiguraci.

Podstatnou část kódu firmware tvoří implementace HTTP serveru a příslušného TCP/IP stacku.

Volba TCP/IP stacku probíhala podle tří hlavních kritérií: dobré podpory TCP, nízkých paměťových nároků a otevřené licence.

Těmto kategoriím nejlépe odpovídají TCP/IP stacky uIP (µIP) a lwIP. Oba jmenované stacky pocházejí od stejného autora, Adama Dunklese.

7.1 uIP

uIP je minimalistický TCP/IP stack, jehož použití s FreeRTOS je velmi dobře zdokumentováno. Je portován pro množství architektur, včetně 8-bitových. Jeho implementace zabere cca 10 kB programové paměti a 2 kB operační paměti. Právě díky jednoduchosti tohoto stacku je jeho implementace velmi snadná.

uIP však má také několik nevýhod. Pro jednoduchost používá pouze jeden buffer o velikosti MTU, a proto je schopen odeslat najednou pouze 1 paket. Z toho vyplývá, že velikost TCP okna tak nemůže být větší než hodnota MTU (1500 B v případě protokolu Ethernet).

Tento problém je navíc posílen použitím Nagle algoritmu v sítích (RFC 896), který se snaží snížit počet malých segmentů v komunikaci pro zvýšení jejího výkonu. Při odeslání jednoho malého segmentu není poslána odpověď okamžitě, ale až za 200 ms. Tato časová prodleva je využita na případné poskládání více malých segmentů do jednoho většího. Tímto způsobem je snížena četnost ACK potvrzení a tím pádem klesá vytížení sítě. V případě použití uIP stacku tyto vlastnosti degradují přenosovou rychlost na jednotky kbit/s.

7.2 lwIP

Tento TCP/IP stack využívá dynamickou alokaci paměti, proto je schopen vyslat více paketů najednou. Tím dramaticky roste jeho výkon a přenosová rychlost dosahuje až 5 Mbit/s.

Tento stack zatím nemá tak podrobnou dokumentaci jako uIP, jeho implementace je složitější a má větší nároky na paměť (vyžaduje alespoň 40 kB programové paměti a 10 kB operační paměti). Přesto byl díky svému výkonu vybrán pro S-02 právě on.

Tento stack poskytuje 3 druhy API rozhraní:

- **RAW** - přímá práce se stackem, složitější na programování, nevyžaduje OS
- **NetConn** - vyžaduje vlákna a tedy potřebuje i operační systém, pracuje na vyšší úrovni a práce s ním je tudíž mnohem pohodlnější
- **Socket** - toto rozhraní je nadstavba výše zmíněného, oproti kterému přináší navíc kompatibilitu s posix/BSD sockety a kód v něm napsaný je možné použít i na jiných TCP/IP stacích, ovšem za cenu sníženého výkonu

Volba rozhraní se provádí v nastavení lwIP stacku - v hlavičkovém souboru *lwipopts.h*, kde se nastavují i jeho další vlastnosti.

Pro tento projekt bylo vzhledem k použití operačního systému zvoleno rozhraní NetConn.

```
#define LWIP_RAW        0
#define LWIP_NETCONN    1
#define LWIP_SOCKET     0
```

Rozhraní NetConn obsahuje následující API funkce:

netconn_new	Vytvoří nové spojení
netconn_delete	Odstraní spojení
netconn_bind	Nastaví spojení na lokální adresu a port
netconn_connect	Připojí se na vzdálenou adresu a port
netconn_send	Odešle data vzdálené straně (UDP)
netconn_recv	Přijme data
netconn_listen	Nastaví naslouchání TCP
netconn_accept	Přijme příchozí spojení TCP
netconn_write	Odešle data vzdálené straně (TCP)
netconn_close	Uzavře TCP spojení

API funkce pro ostatní druhy rozhraní a jejich popis naleznete v dokumentaci lwIP stacku.

Pro přístup k fyzické vrstvě je použit ovladač dodaný firmou ST.

Jak lwIP, tak použitý phyter DP83848 podporují MII i RMII rozhraní. Redukované MII rozhraní (RMII) má výhodu v menším počtu vodičů, ale požaduje dvojnásobnou frekvenci hodin. Použití phyteru DP83848 je výhodné z důvodu shody parametrů hodinového signálu 50 MHz generovaného MCU s kvalitativními požadavky phyteru (viz podkapitola Časování). Tento phyter doporučuje také výrobce MCU.

Pro použití lwIP je třeba připojit potřebné hlavičkové soubory:

```
#include "net_launch_srv.h"

#include "lwip/sys.h"
#include "lwip/api.h"
#include "lwip/tcpip.h"
#include "lwip/memp.h"
#include "lwip/stats.h"
#include "netif/loopif.h"
#include "lwip/dhcp.h"

//aplikace webserveru
#include "webserver/httpd.h"
```

Samotné spouštění TCP/IP stacku provádí následující funkce:

```
void vStartEthernetTasks( unsigned portBASE_TYPE uxPriority, void
*pvParameters )
{
  // inicializace lwip a EMAC rozhraní
  tcpip_init( prvEthernetConfigureInterface, NULL );

  // vytvoření nového vlákna pro http server
  sys_thread_new( "httpd", httpd, ( void * ) NULL, HTTPD_STACK_SIZE,
uxPriority );

}
```

8 HTTP server

Klíčovým prvkem firmware S-02 je HTTP server, o kterém pojednává tato kapitola.

Jako základ pro webserver byla použita ukázková aplikace dodávaná s lwIP stackem. Samotná data uživatelského rozhraní jsou uložena v jednoduchém souborovém systému na bázi jednosměrného seznamu. Při požadavku uložení všech dat v S-02 je možné využít přídavné flash paměti, která komunikuje přes sběrnici SPI.

Server při požadavku zadaném HTTP metodou GET vrací jako odpověď stránku ze souborového systému, kde jsou umístěna data pro sestavení uživatelského prostředí. Data poslaná HTTP metodou POST jsou předávána funkci pro zpracování asynchronních požadavků. Tato funkce z hodnoty "id" zjistí, pro jakou komponentu jsou data určena, a přepošle jí požadavek.

Ukázka komunikace metodou GET:

```
GET /index.htm HTTP/1.1
Host: test.dev
Connection: keep-alive
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/536.8 (KHTML,
like Gecko) Chrome/20.0.1105.2 Safari/536.8
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Encoding: gzip,deflate,sdch
Accept-Language: cs-CZ,cs;q=0.8
Accept-Charset: windows-1250,utf-8;q=0.7,*;q=0.3
```

Dle parametru GET je vrácen příslušný soubor ze souborového systému.

Ukázka komunikace metodou POST:

```
POST / HTTP/1.1
Host: test.dev
Connection: keep-alive
Content-Length: 29
Origin: null
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/536.8 (KHTML,
like Gecko) Chrome/20.0.1105.2 Safari/536.8
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
Accept: */*
Accept-Encoding: gzip,deflate,sdch
Accept-Language: cs-CZ,cs;q=0.8
Accept-Charset: windows-1250,utf-8;q=0.7,*;q=0.3

{"id": "253","data": "4,5"}
```

Data přijatá v požadavku POST jsou předána ke zpracování:

```
switch (json-get(data,"id")){
...
case "253": reply=component253(data); break;
      ...
}
```

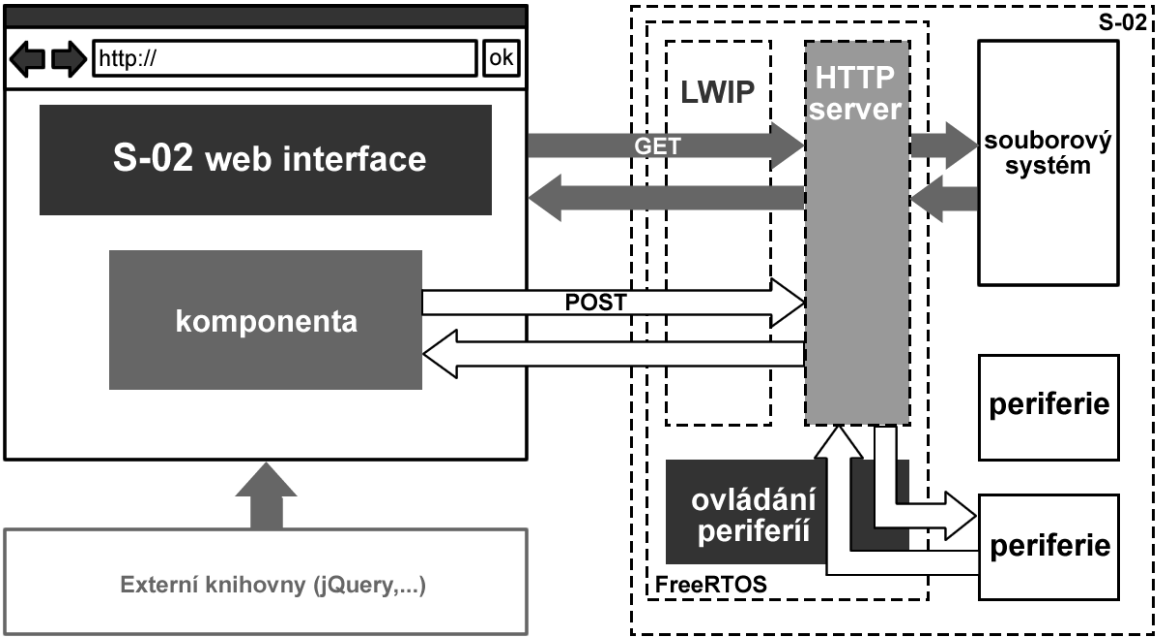
Funkce `json-get` nalezne v řetězci formátu JSON požadovaný klíč a vrátí jeho hodnotu.

Obslužná funkce komponenty vrátí výsledek opět v tomto formátu.

9 Uživatelské rozhraní

Signifikantním znakem zařízení S-02 je jeho uživatelské webové rozhraní, jehož možnosti představí tato kapitola.

Webové uživatelské rozhraní je generováno částečně ze statických souborů umístěných ve firmware (samotná definice uživatelského rozhraní) a částečně z knihoven dostupných v síti internet. Je však možné umístit veškerý kód do firmware. Umístění mimo samotné zařízení má výhodu v možnosti upravovat funkcionalitu a samotné rozhraní bez zásahů do firmware, což podstatným způsobem zkracuje reakční dobu nutnou pro případné úpravy. Zároveň se zde nabízí možnost rozhraní integrovat do stávajících externích systémů a s S-02 komunikovat pouze jeho komunikačním protokolem.



Obr. 9.1: Princip uživatelského rozhraní

Komunikace s HTTP serverem je inicializována z rozhraní. Po načtení úvodní stránky probíhá veškerá komunikace asynchronně pomocí javascriptové technologie AJAX. Pro přenos zprávy je použit jednoduchý protokol JSON, který přenáší standardně 2 položky - identifikátor komponenty a data. Server odpovídá stejným způsobem. Pokud nastane chyba, přidává server do komunikace položku *error* s textovým popisem chyby, na kterou rozhraní reaguje jejím zobrazením. Uživatelské rozhraní může dále poslat položku *config*, která je

určena k nastavení periférie (např. k nastavení parametrů sériového portu), a položku *auth* k autentifikaci uživatele, pokud ji periférie vyžaduje.

Ukázka komunikace s testovací matematickou komponentou *id=253*, která sčítá dvě čísla:

Požadavek z rozhraní:

```
{"id": "253","data": "4,5"}
```

Odpověď od serveru:

```
{"id": "253","data": "9"}
```

Pro tvorbu rozhraní a zprostředkování komunikace je použit javascriptový framework jQuery, pro vizualizaci komponent pak CSS framework Twitter Bootstrap.

Webové rozhraní je složeno z komponent a je zde připraveno několik základních komponent pro běžné použití:

9.1 Základní komponenty uživatelského rozhraní

Button

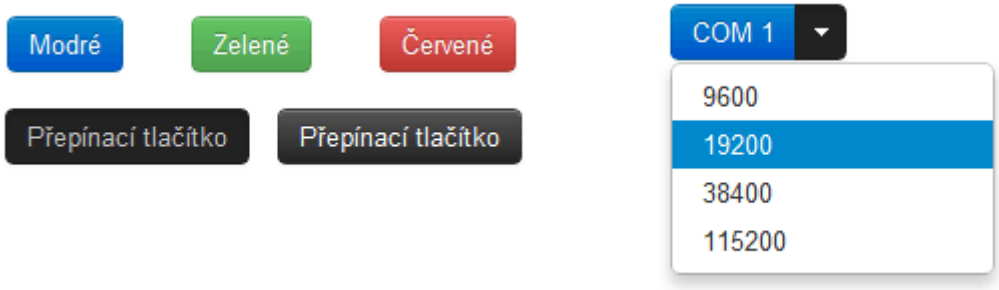
Vstupní komponenta, vyvolává jednorázový dotaz na server. Je připravena i modifikace této komponenty umožňující vybrat akci ze seznamu.

Toggle Button

Vstupní komponenta, rozšíření komponenty Button o možnost zapamatování si logického stavu.

Select Button

Vstupní komponenta, opět se jedná o rozšíření základní komponenty Button. Nabízí možnost výběru z několika voleb.



Obr. 9.2: Typy tlačítek v uživatelském rozhraní

Terminal

Vstupně výstupní komponenta, emulátor terminálu vhodný pro sériovou komunikaci. Tato komponenta je postavena na doplňku jQuery Terminal Emulator.

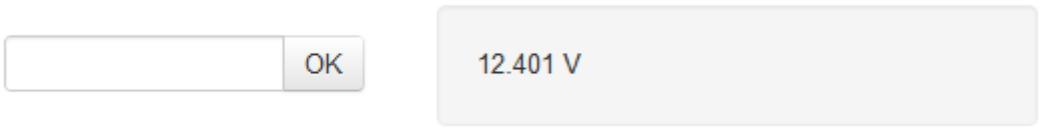
Obr. 9.3: Komponenta „emulátor terminálu“

Input

Vstupní komponenta, odeslání jednorázového dotazu na server s uživatelem vloženými daty.

Text

Výstupní komponenta, vizualizuje data odpovědi, možno nastavit interval obnovování.



Obr. 9.4.: Komponenty pro textový vstup a výstup

State

Výstupní komponenta, vizualizuje logickou hodnotu odpovědi, možno nastavit interval obnovování.

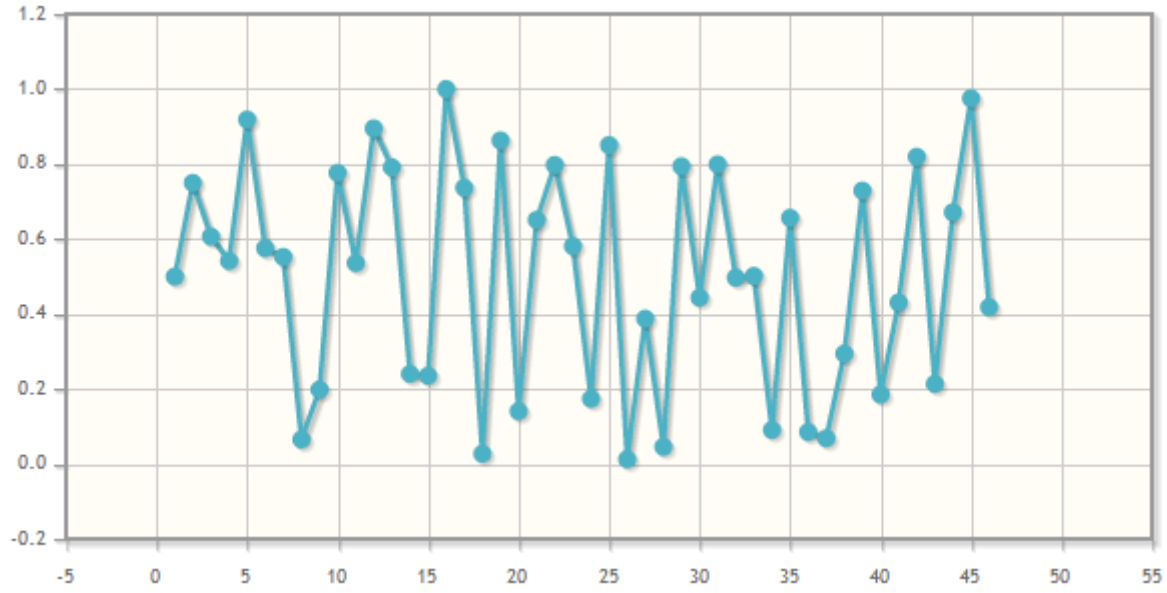
Bar

Výstupní komponenta, vizualizuje hodnotu 0 - 100% v podobě proužkového ukazatele.

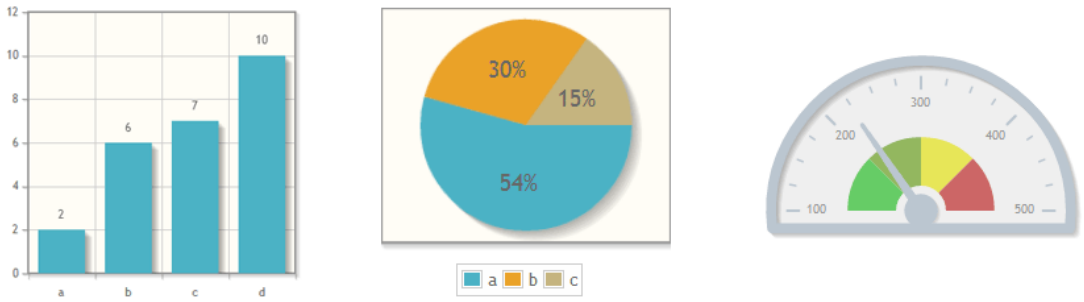
Obr. 9.5: Komponenty pro vizualizaci stavů

Graph

Experimentální výstupní komponenta, vizualizuje přijatá data do různých typů grafu. Využívá opensource javascriptovou komponentu jqPlot. Knihovna využívá HTML 5, které je v prohlížeči Internet Explorer plně implementované až od verze 9. Starší verze tohoto prohlížeče jsou podporovány pomocí externí knihovny explorerCanvas.

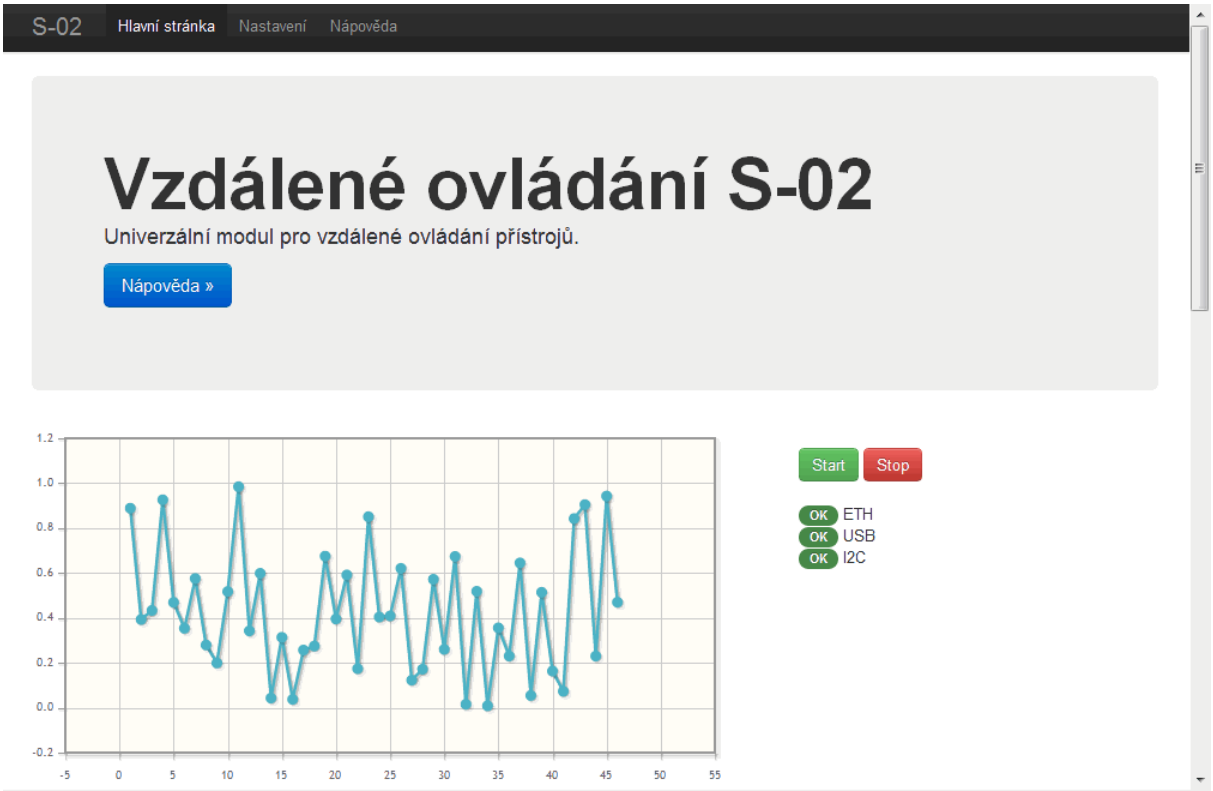


Obr. 9.6.: Komponenta pro vykreslování grafů



Obr. 9.7: Další možnosti grafické knihovny

Statická ukázka webového rozhraní s jednotlivými komponentami se nachází na adrese http://s02.smitka.org .



Obr. 9.8: Ukázka stránky s webovým rozhraním

9.2 Objektový model komponent

Pro vytváření konkrétních komponent je využit objektový model, kde jsou základní funkce děděny ze základní abstraktní komponenty. Při tvorbě nových komponent je tedy možné se věnovat pouze návrhu konkrétní funkcionality a nestarat se o samotný proces komunikace.

Následující příklad ukazuje zjednodušené použití objektového modelu při tvorbě vlastní komponenty, která neobsahuje žádné grafické prvky, pouze zobrazuje dialogové okno s přijatými hodnotami.

```
//vytvoření jmenného prostoru
if (!('s02' in this)) this.s02 = {};

//konstruktor základní komponenty
s02.BaseComponent = function (timeout) {
      //timeout v násobcích 0.1s, vytvořen pokdu je nastaven
      if (timeout) {
            this.interval = setInterval(this.send, timeout * 100);
      }
};

//úložiště dat pro odeslání, ve formátu JSON
s02.BaseComponent.prototype.data = "";

//vytvoření JSON řetězce z dodaného objektu
s02.BaseComponent.prototype.setData = function (data) {
      this.data = JSON.stringify(data);
};
//testovací funkce
s02.BaseComponent.prototype.test = function () {
      alert("OK");
};
//zpracování odpovědi ze serveru
s02.BaseComponent.prototype.receive = function (json) {
      this.data=json;

};
//odeslání požadavku na server
s02.BaseComponent.prototype.send = function () {
      $.postJSON("http://s02.dev/", this.data, this.receive);
};

//ukázka testovací komponenty
s02.TestComponent = function (timeout, info) {
      //ID preriferie
      var id = "253";
      //datový objekt dle protokolu
      var data = new Object();
      data['id'] = id;
      data['data'] = info;
      //zdědění základní komponenty
      var cmp = new s02.BaseComponent(timeout);
      //nová vlastnost info
      cmp.info = data;
      //vlastní zpracování přijatých dat
```

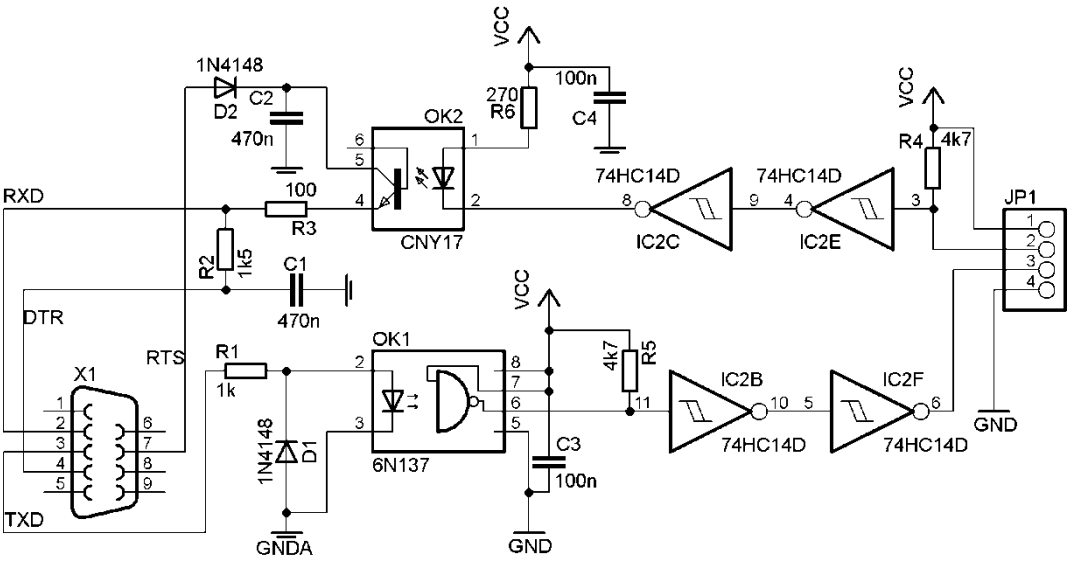
```
      cmp.receive = function (json) {
            alert("Přijatá data:" + json.data)
      };
      return cmp;
};

//příklad použití
//vyčítání dat z periférie po 1s
var komponenta = new s02.TestComponent(10,"test");
komponenta.setData(komponenta.info);
komponenta.send();
```

10 Rozšiřující karta

V závěrečné kapitole jsou uvažovány možnosti vytvoření rozšiřující karty pro použití v akustických laboratořích FEL. Tato deska obsahuje galvanicky oddělené porty sběrnice RS-232.

Rozšiřující karta byla navržena za využití mikrokontroléru STM32F100RC. Tento mikrokontrolér může pracovat na maximální frekvenci 24 MHz, obsahuje 256 kbit flash a 24 kbit SRAM. Kromě pokročilých periférií jako jsou Ethernet, USB nebo CAN podporuje všechny periférie běžné v řadě STM32. Mikrokontrolér nabízí 5 UART rozhraní, které je v rozšiřující kartě možné využít.

Sériové porty je vhodné galvanicky oddělit. První varianta ukazuje zapojení pro galvanicky oddělenou RS-232 s napájením pomocí signálů DTR a RTS ze vzdáleného přístroje. Galvanické oddělení je realizováno optrony. Pro udržení hodnoty logického signálu jsou použity Schmittovy klopné obvody. Napájení lze nahradit i externím zdrojem. Pokud však není sériový port ve vzdáleném přístroji galvanicky oddělen, hrozí v tomto případě jeho poškození.



Obr. 10.1: Schéma galvanicky odděleného sériového portu

V případech, kdy lze předpokládat, že rozdíl nulových potenciálů nebude větší než 50 V, lze s úspěchem pro galvanické oddělení použít obvod MAX3250. Podrobný popis jeho zapojení lze najít v příslušné technické dokumentaci [16].

11 Závěr

V rámci práce byly prozkoumány potřeby vzdáleného ovládání přístrojů akustických laboratoří. Výsledkem zkoumání bylo zjištění, že přístroje jsou velmi různorodé a nejčastěji využívají rozhraní RS-232, ale je možné se setkat i s rozhraními USB nebo Ethernet. Na základě toho byl navržen princip komunikace nezávislý na konkrétním rozhraní přístroje, jednotné rozhraní a komunikační protokol zajišťující potřebné úkony.

Tento návrh umožňuje realizovat konkrétní protokoly jednotlivých přístrojů a navrhnout pro ně komfortní uživatelské webové prostředí.

Jako konkrétní řešení problému byl zvolen koncept univerzální základní desky a rozšiřujících karet komunikujících po sběrnici I_2C. Základní deska zprostředkovává síťovou komunikaci a obsahuje operační systém reálného času, který obsahuje HTTP server.

Aby byly minimalizovány nároky na zařízení, komunikuje uživatelské rozhraní se zařízením pomocí asynchronního javascriptu, u něhož není nutné přenášet při aktualizaci hodnot celé uživatelské rozhraní, ale pouze konkrétní data vystavená periferiemi. Dalším zjednodušením uživatelského rozhraní je použití externích knihoven stahovaných ze sítě internet. Pokud by tento princip nevyhovoval potřebám dané aplikace, je možné zařízení rozšířit o flash paměť a knihovny umístit do ní.

Pro jednoduchost nasazení v praxi lze pro napájení zařízení využít technologii Power over Ethernet.

Implementace a ověření správnosti návrhu proběhlo na univerzální vývojové desce. Na základě zkušeností s vývojovým prototypem bylo navrženo finální zařízení včetně návrhu dvouvrstvé PCB.

Hlavní přínos této práce spočívá v návrhu koncepce univerzálního ovládání nejrůznějších přístrojů jednotným komunikačním protokolem a vytvoření komfortního uživatelského rozhraní, které je díky svému objektovému modelu velmi dobře modifikovatelné a rozšiřitelné o další funkce a podporu dalších ovládaných přístrojů. Přes svou komplexnost klade uživatelské rozhraní minimální nároky na obsluhující hardware, což je dáno přenesením operací pro zobrazení prostředí z webového serveru na klientský prohlížeč.

Vzhledem ke své modularitě a rozšiřitelnosti najde tento koncept využití i v řadě dalších praktických aplikací.

12 Seznam literatury

[1] Axelson, Jan. USB Complete - Everything You Need to Develop Custom USB Peripherals Third Edition. Lakeview Research LLC, 2005. ISBN: 978-1-931448-03-1.

[2] Barry, Richard. Using the FreeRTOS™ Real Time Kernel - A Practical Guide. FreeRTOS.org, 2010. ISBN: 978-1446169148.

[3] Camenzind, Hans. Designing Analog Chips. Dostupné z: <http://www.designinganalogchips.com/>. ISBN-13: 978-1589397187.

[4] Dostálek, Libor; Kabelová, Alena. Velký průvodce protokoly TCP/IP a systémem DNS. Computer Press, 2008. ISBN: 978-80-251-2236.

[5] Kainka, Burkhard. USB - měření, řízení a regulace pomocí sběrnice USB. Praha: BEN - technická literatura, 2002. ISBN: 80-7300-073-3.

[6] Pinker, Jiří. Mikroprocesory a mikropočítače. Praha: BEN - technická literatura, 2004. ISBN: 80-7300-110-1.

[7] Pinker, Jiří; Poupa, Martin. Číslicové systémy a jazyk VHDL. Praha: BEN - technická literatura, 2001. ISBN: 80-7300-198-5.

[8] Váňa, Vladimír. ARM pro začátečníky. Praha: BEN - technická literatura, 2009. ISBN: 978-80-7300-246-6.

12.1 Technikcé manuály

[9] DP83848 Single 10/100 Mb/s Ethernet Transceiver - Reduced Media Independent Interface™ (RMII™) Mode, AN-1405, Application Note, September 2005. National Semiconductor

[10] DP83848C PHYTER Commercial Temperature Single Port 10/100 Mb/s Ethernet Physical Layer Transceiver, Datasheet, May 2008. National Semiconductor

[11] Getting started with uClinux™ for STM32F10x high-density devices, AN3012, Application note, revision 2. ST Microelectronics

[12] Getting started with STM32F10xxx hardware development, AN2586, Application note, revision 7. ST Microelectronics

[13] LM1117 800mA Low-Dropout Linear Regulator, Datasheet, February 2000. National Semiconductor

[14] LwIP TCP/IP stack demonstration for STM32F107xx connectivity line microcontrollers, AN3102, Application note, revision 1. ST Microelectronics

[15] LwIP TCP/IP stack demonstration for STM32F2x7xx microcontrollers, AN3384, Application note, revision 2. ST Microelectronics

[16] MAX3250 ±50V Isolated, 3.0V to 5.5V, 250kbps, 2 Tx/2 Rx, RS-232 Transceiver, Datasheet, revision 3. Maxim Integrated Products

[17] MAX5035 1A, 76V, High-Efficiency MAXPower Step-Down DC-DC Converter, Datasheet, revision 5. Maxim Integrated Products

[18] Migrating a microcontroller application from STM32F1 to STM32F2 series, AN3427, Application note, revision 1. ST Microelectronics

[19] RXE Selection Guide and Product Data. Datasheet, March 2006. TE Connectivity

[20] STM32F101xx, STM32F102xx, STM32F103xx, STM32F105xx and STM32F107xx advanced ARM-based 32-bit MCUs, RM0008, Reference manual, revision 11. ST Microelectronics

[21] STM32F101xx, STM32F102xx and STM32F103xx low-power modes, AN2629, Application note, revision 2. ST Microelectronics

[22] STM32F105xx, STM32F107xx Connectivity line, ARM-based 32-bit MCU with 64/256 KB Flash, USB OTG, Ethernet, 10 timers, 2 CANs, 2 ADCs, 14 communication interfaces, Datasheet, revision 5. ST Microelectronics

[23] STM32F10x in-application programming using the USART, AN2557, Application note, revision 8. ST Microelectronics

[24] STM32F10xxx - USB-FS-Device development kit, User manual, UM0424, Datasheet, revision 10. ST Microelectronics

[25] STM32™ microcontroller system memory boot mode, AN2606, Application note, revision 13. ST Microelectronics

[26] TPS237x: IEEE 802.3af PoE Powered Device Controllers, Datasheet, revision B. Texas Instruments

12.2 Internetové zdroje

[27] lwIP Wiki [online] [22.1.2011] dostupné z <http://lwip.wikia.com>

[28] FreeRTOS - Quick start guide [online] [27.12.2010] dostupné z <http://www.freertos.org/FreeRTOS-quick-start-guide.html>

[29] FreeRTOS API categories [online] [27.12.2010] dostupné z <http://www.freertos.org/a00106.html>

[30] IEEE-SA -IEEE Get 802 Program - 802.3: Ethernet [online] [7.2.2012] dostupné z <http://standards.ieee.org/about/get/802/802.3.html>

[31] Power over Ethernet - Wikipedia, the free encyclopedia [online] [7.2.2012] dostupné z <http://en.wikipedia.org/wiki/Power_over_Ethernet>

[32] Ethernet Technologies - DocWiki [online] [8.11.2010] dostupné z <http://docwiki.cisco.com/wiki/Ethernet_Technologies>

[33] Internet Protocols - DocWiki [online] [8.11.2010] dostupné z <http://docwiki.cisco.com/wiki/Internet_Protocols>

[34] HW server představuje - Sériová linka RS-232 | HW.cz [online] [3.6.2010] dostupné z <http://www.hw.cz/rozhrani/hw-server-predstavuje-seriova-linka-rs-232.html>

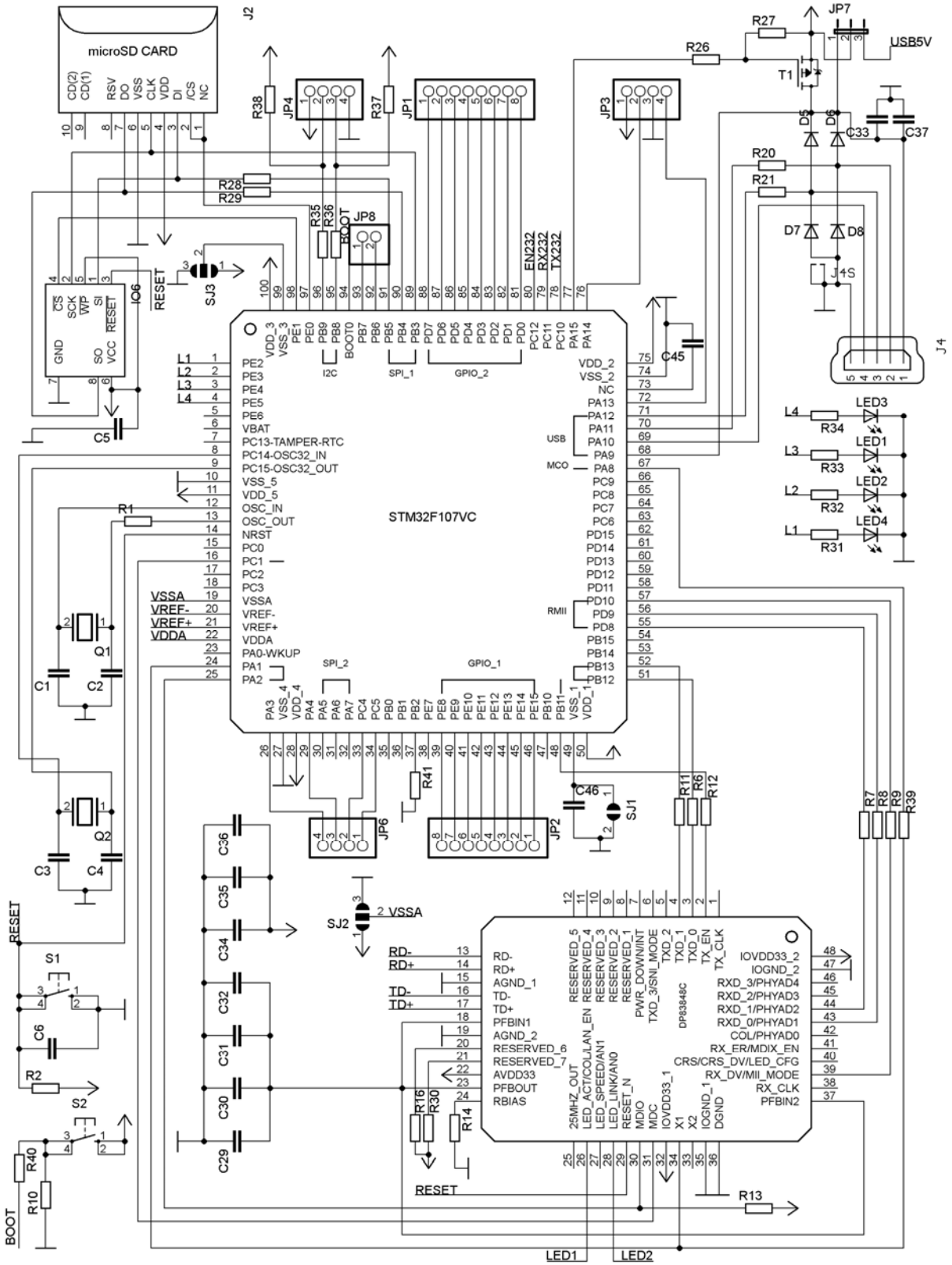
12.3 Další zdroje

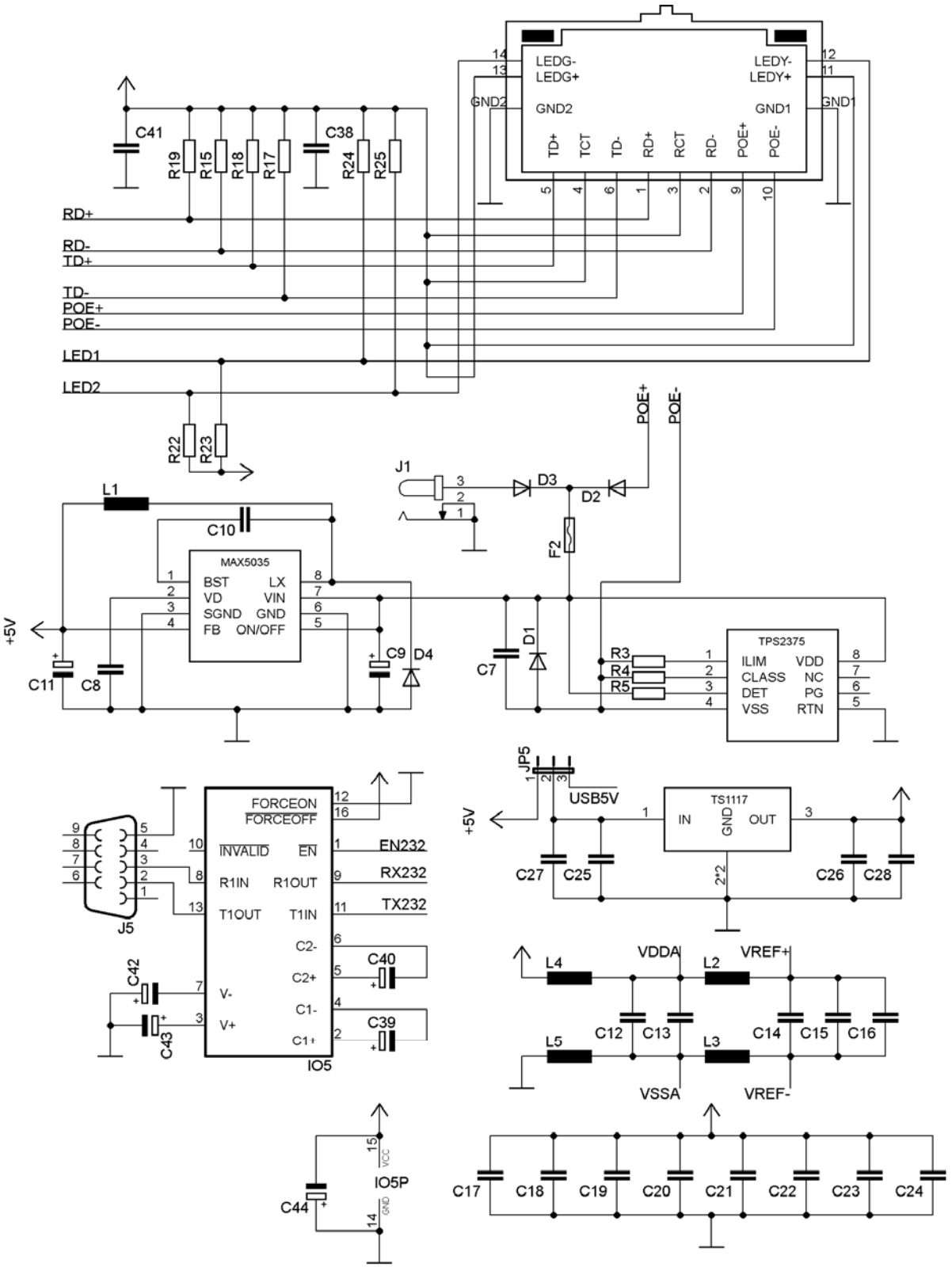
[35] Studijní materiály Cisco CCNA

[36] Dudáček, Karel. Sériová rozhraní SPI, Microwire, I2C a CAN. 2002. Dostupné z:

<http://home.zcu.cz/~dudacek/NMS/Seriova_rozhrani.pdf>

13 Příloha A – Schéma zapojení



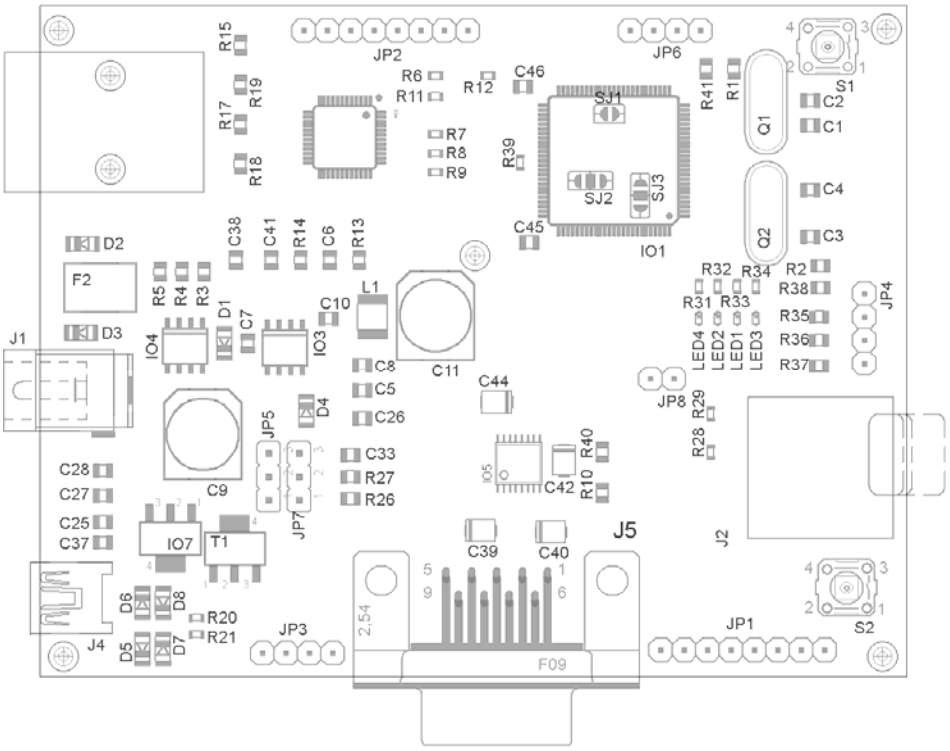

LEDG-
LEDG+
LEDY-
LEDY+
GND2
GND1
TD+
TCT
TD-
RD+
RCT
RD-
POE+
POE-
C41
C38
R19
R15
R18
R17
R24
R25
RD+
RD-
TD+
TD-
POE+
POE-
LED1
LED2
R22
R23
J1
D3
D2
F2
L1
C10
MAX5035
BST
LX
VD
VIN
SGND
GND
FB
ON/OFF
+5V
C11
C8
C9
D4
C7
D1
R3
R4
R5
TPS2375
ILIM
VDD
CLASS
NC
DET
PG
VSS
RTN
JP5
USB5V
TS1117
IN
GND
OUT
C27
C25
C26
C28
2*2
FORCEON
FORCEOFF
INVALID
EN
R1IN
R1OUT
T1OUT
T1IN
C2-
C2+
V-
V+
C1-
C1+
EN232
RX232
TX232
J5
C42
C43
C40
C39
IO5
L4
L2
L5
L3
VDDA
VREF+
VSSA
VREF-
C12
C13
C14
C15
C16
C17
C18
C19
C20
C21
C22
C23
C24
C44
IO5P
VCC
GND

14 Příloha B – Seznam součástek

C1	20pF, C0805	**C45**	2u2, C0805	**R2**	10k, R0805
C2	20pF, C0805	**C46**	100n, C0805	**R3**	100k, R0805
C3	10pF, C0805	**D1**	63V, SOD80C	**R4**	4k3, R0805
C4	10pF, C0805	**D2**	63V, SOD80C	**R5**	25k, R0805
C5	100n, C0805	**D3**	63V, SOD80C	**R6**	33, R0603
C6	100nF, C0805	**D4**	90V, SOD80C	**R7**	33, R0603
C7	100n, C0805	**D5**	63V, SOD80C	**R8**	33, R0603
C8	100n, C0805	**D6**	63V, SOD80C	**R9**	33, R0603
C9	22u	**D7**	63V, SOD80C	**R10**	10k, R0805
C10	100n, C0805	**D8**	63V, SOD80C	**R11**	33, R0603
C11	100u	**F2**	RXE050	**R12**	33, R0603
C12	100n, C0805	**IO1**	STM32F107V, TQFP100	**R13**	1k5, R0805
C13	100n, C0805	**IO2**	DP83848C, LQFP48	**R14**	4k8, R0805
C14	100n, C0805	**IO3**	MAX5035, SO8	**R15**	50, R0805
C15	100n, C0805	**IO4**	TPS2375, SO8	**R16**	2k2, R0805
C16	10u, C1210	**IO5**	MAX3221CUE, TSSOP16	**R17**	50, R0805
C17	100n, C0805	**IO6**	AT45DB161D	**R18**	50, R0805
C18	100n, C0805	**IO7**	TS1117	**R19**	50, R0805
C19	100n, C0805	**J1**	DC POWER JACK 2mm	**R20**	22, R0603
C20	100n, C0805	**J2**	MICROSD	**R21**	22, R0603
C21	100n, C0805	**J3**	RJ45-TRAFO-POE	**R22**	2k2, R0805
C22	100n, C0805	**J4**	MINI-USB_S	**R23**	2k2, R0805
C23	100n, C0805	**J5**	Canon 9	**R24**	220, R0805
C24	100n, C0805	**JP1**	1x8	**R25**	220, R0805
C25	100n, C0805	**JP2**	1x8	**R26**	470, R0805
C26	100n, C0805	**JP3**	1x4	**R27**	10k, R0805
C27	2u2, C0805	**JP4**	1x4	**R28**	22, R0603
C28	2u2, C0805	**JP5**	1x3	**R29**	22, R0603
C29	100n, C0805	**JP6**	1x4	**R30**	2k2, R0805
C30	100n, C0805	**JP7**	1x3	**R31**	470, R0603
C31	100n, C0805	**JP8**	1x2	**R32**	470, R0603
C32	10u, C1210	**L1**	100uH, L1812	**R33**	470, R0603
C33	10u, C0805	**L2**	10uH, L1812	**R34**	470, R0603
C34	100n, C0805	**L3**	10uH, L1812	**R35**	100, R0805
C35	100n, C0805	**L4**	10uH, L1812	**R36**	100, R0805
C36	100n, C0805	**L5**	10uH, L1812	**R37**	4k7, R0805
C37	100n, C0805	**LED1**	RED, 0603	**R38**	4k7, R0805
C38	100n, C0805	**LED1**	RED, 0603	**R39**	33, R0603
C39	0.1u	**LED3**	RED, 0603	**R40**	10k, R0805
C40	0.1u	**LED4**	RED, 0603	**R41**	10k, R0805
C41	100n, C0805	**Q1**	25MHz	**S1**	SW1
C42	0.1u	**Q2**	32.768KHz	**S2**	SW1
C43	0.1u	**T1**	TSM2301		
C44	0.1u	**R1**	220, R0805		

15 Příloha C – Rozmístění součástek

Vrchní strana



Spodní strana