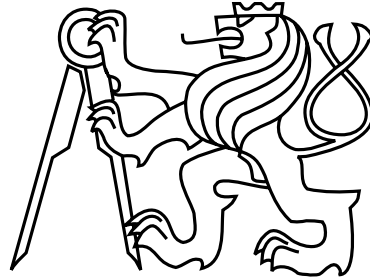


České vysoké učení technické v Praze
Fakulta elektrotechnická



Diplomová práce

Ethernetem řízený vstupně-výstupní modul
Ethernet-controlled input-output module

Michal Pešák

Vedoucí práce: Ing. Tomáš Vaňát

Studijní program: Elektrotechnika a informatika (magisterský)

Obor: Výpočetní technika

květen 2015

České vysoké učení technické v Praze
Fakulta elektrotechnická

katedra počítačů

ZADÁNÍ DIPLOMOVÉ PRÁCE

Student: **Bc. Michal Pešák**

Studijní program: Elektrotechnika a informatika (magisterský), strukturovaný
Obor: Výpočetní technika

Název tématu: **Ethernetem řízený vstupně-výstupní modul**

Pokyny pro vypracování:

Navrhňte a implementujte prototyp mikrokontrolérem řízeného hardwarového zařízení, které bude obsahovat alespoň dva digitální vstupní porty a alespoň dva digitální výstupní porty, všechny galvanicky oddělené. Pro čtení vstupů a ovládání výstupů bude použito rozhraní Ethernet s některým standardním protokolem nad TCP/IP (např. Telnet, SSH, HTTP). Jedno z možných použití zařízení bude vzdálené ovládání běžného PC, které nemá zabudovanou možnost vzdálené správy. To znamená, že zařízení bude připojeno místo čelního panelu počítačové skříně (nebo paralelně s ním), výstupní porty budou simulovat tlačítka Power a Reset a jeden ze vstupů bude snímat stav kontrolky Power. Zařízení by mělo umět simulovat krátký stisk tlačítka (cca 1 sec), dlouhý stisk tlačítka (cca 5 sec.) a rozlišit zda kontrolka svítí trvale, nebo bliká. Komunikační rozhraní implementujte tak, aby jej bylo možné ovládat jak strojově, tak manuálně.

Seznam odborné literatury:

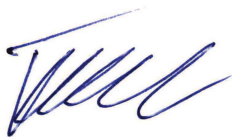
IEEE Std 802.3-2012 - IEEE Standard for Ethernet
<http://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=6419733>

Záhlava Vít, Návrh a konstrukce desek plošných spojů, BEN - technická literatura, 2010

RFC 791 - Internet Protocol
<http://tools.ietf.org/html/rfc791>

Vedoucí: Ing. Tomáš Vaňát

Platnost zadání: do konce letního semestru 2015/2016



doc. Ing. Filip Železný, Ph.D.
vedoucí katedry



prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 25. 3. 2015

Poděkování

Děkuji Ing. Ing. Tomáši Vaňátovi ,vedoucímu této diplomové práce, za jeho vstřícnou pomoc při vytváření zadání a tvorbě této práce.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

Nemám závažný důvod proti užití tohoto školního díla ve smyslu § 60 Zákona c. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon).

Praze dne 10.5.2015

.....

Abstrakt

Cílem této diplomové práce bylo vytvořit modul s galvanicky oddělenými vstupy a výstupy ovládaný přes ethernet pomocí některého ze standardních protokolů nad síťovým protokolem TPC/IP.

U této aplikace je největší důraz kladen na cenovou dostupnost, univerzálnost a možnost začlenit modul do stávající infrastruktury.

Abstract

The aim of project was to create module with . The main aim of this diploma thesis was to create a module with galvanically isolated inputs and outputs controlled via Ethernet using one of the standard protocols over a network protocol TPC/IP.

In this application in firts place was affordability, universality and the possibility of integrating the module into the existing infrastructure.

Obsah

Seznam obrázků	xv
Seznam tabulek	xvii
1 Úvod	1
2 Úvodní studie	2
2.1 Stávající stav a cíle	2
2.2 Funkční požadavky	2
2.3 Nefunkční požadavky	2
3 Návrh řešení	3
3.1 Komunikace s modulem	3
3.2 Napájení modulu	4
3.2.1 Centrální napájení	4
3.2.2 Napájení z počítače	4
3.3 Vstupy a výstupy	4
4 Návrh architektury modulu	5
4.1 Hlavní rysy vybrané architektury jednočipového počítače	5
4.2 Základní vlastnosti vybrané řady a typu jednočipového počítače pro tuto implementaci	6
4.3 Taktování jednočipového počítače	6
5 Návrh napájení modulu	7
5.1 Napájecí napětí použitá v modulu	7
5.2 Zdroje napájení 5V	7
5.2.1 Napájení přes POE	7
6 Návrh signalizačních LED diod modulu	8
7 Návrh zapojení oscilátoru	10
8 Návrh ovládacích tlačítek	11
8.1 Tlačítko reset	11
8.2 Tlačítko safemode	11
9 Návrh vstupů modulu	12
9.1 Zapojení emitoru optočlenu	12
9.2 Zapojení detektoru optočlenu	12
10 Návrh výstupů	14
11 Návrh zapojení ethernetu	15
12 Návrh schéma zapojení a desky tištěného spoje	16
12.1 Použitý software	16

12.2 Schéma zapojení	16
13 Popis základních prvků modulu	17
14 Návrh software - Architektura	19
15 Návrh ovládání modulu	20
16 Členění software - knihovny	21
16.1 TCP/IP stack	21
16.2 Ostatní knihovny programu	21
17 Úpravy TCP/IP stacku	22
18 Souborový systém MPFS	23
19 Základní nastavení TCP/IP stacku	24
19.1 MAC adresa	24
19.2 IP adresa	24
19.3 NETBIOS jméno	24
20 Implementace protokolu REST	25
20.1 Metody volání služeb	25
20.2 Aplikace pro tvorbu dat MPFS systému a definice typu souborů HTTP serveru	25
21 Implementace služeb a formátu pro výměnu dat JSON	26
21.1 Implementace služeb	26
21.2 Zpracování vstupu služeb	26
21.2.1 Identifikace metody služby	26
21.2.2 Zpracování bloku vstupních parametrů	26
21.2.2.1 Zpracování názvu parametru	27
21.2.2.2 Zpracování hodnoty parametru	27
21.2.3 Zpracování jednoho parametru	28
21.2.4 Kontrola parametrů a volání metod služby	28
21.3 Zpracování výstupu služeb	29
22 Zabezpečení aplikace	30
23 Popis služeb modulu	31
23.1 Služba Config.svc	31
23.1.1 Metoda Status	31
23.1.1.1 Vstupní parametry	31
23.1.1.2 Návrátové hodnoty	31
23.1.1.3 Příklad volání	32
23.1.2 Metoda WriteIP	32
23.1.2.1 Vstupní parametry	32
23.1.2.2 Návrátové hodnoty	32
23.1.2.3 Příklad volání	33

23.1.3	Metoda WritePC	33
23.1.3.1	Vstupní parametry	33
23.1.3.2	Návratové hodnoty	33
23.1.3.3	Příklad volání	33
23.1.4	Metoda Reset	34
23.1.4.1	Vstupní parametry	34
23.1.4.2	Návratové hodnoty	34
23.1.4.3	Příklad volání	34
23.2	Služba PC.svc	35
23.2.1	Metoda Get	35
23.2.1.1	Vstupní parametry	35
23.2.1.2	Návratové hodnoty	35
23.2.1.3	Příklad volání	35
23.2.2	Metoda Reset	35
23.2.2.1	Vstupní parametry	35
23.2.2.2	Návratové hodnoty	35
23.2.2.3	Příklad volání	35
23.2.3	Metoda PowerON	36
23.2.3.1	Vstupní parametry	36
23.2.3.2	Návratové hodnoty	36
23.2.3.3	Příklad volání	36
23.2.4	Metoda PowerOFF	36
23.2.4.1	Vstupní parametry	36
23.2.4.2	Návratové hodnoty	36
23.2.4.3	Příklad volání	36
23.3	Služba IO.svc	37
23.3.1	Metoda Get	37
23.3.1.1	Vstupní parametry	37
23.3.1.2	Návratové hodnoty	37
23.3.1.3	Příklad volání	37
23.3.2	Metoda Set	37
23.3.2.1	Vstupní parametry	37
23.3.2.2	Návratové hodnoty	37
23.3.2.3	Příklad volání	38
24	Základní nastavení a funkce modulu	39
24.1	TCP/IP protokol	39
24.2	Funkce módů IO a PC	39
24.2.1	Funkce modulu v nastavení PC	39
24.2.1.1	Vyhodnocení vstupů modulu	39
24.2.1.2	Ovládání výstupů PC	42
24.2.2	Funkce modulu v nastavení IO	42
25	Závěr	43
26	Seznam literatury	45
27	Seznam použitých zkratk	47

28	Uživatelská příručka	49
28.1	index.htm	49
28.2	Conf.htm	49
28.2.1	Konfigurace TCP/IP	50
28.2.2	Konfigurace parametrů pro mód PC	52
28.3	PC.htm	54
28.4	IO.htm	55
29	Obsah přiloženého CD	57
30	Schéma zapojení	59
31	Návrh tištěného spoje	63
32	Osazovací schéma	67

Seznam obrázků

6.1	Schéma zapojení led diod ethernetu	8
6.2	Schéma zapojení led diod signalizujících stav modulu	9
7.1	Schéma zapojení oscilátoru	10
8.1	Schéma tlačítka reset	11
8.2	Schéma tlačítka safemode	11
9.1	Schéma zapojení vstupů	13
10.1	Schéma zapojení výstupů	14
11.1	Schéma zapojení ethernetu	15
13.1	Schéma ovládacích prvků a konektorů modulu	17
24.1	Diagram detekce stavu PC "ON"	40
24.2	Diagram detekce stavu PC "OFF"	40
24.3	1. Diagram detekce stavu PC "Stanby"	41
24.4	2. Diagram detekce stavu PC "Stanby"	41

Seznam tabulek

9.1	Charakteristiky vstupů	12
20.1	HTTP server - přidané typy souborů	25

1 Úvod

S postupem stále častějšího nasazování osobních počítačů do všech lidských činností je kladen stále větší důraz na centrální správu těchto počítačů a minimalizaci potřeby lidské práce na tuto správu. Na trhu se základními deskami osobních počítačů většina těchto základních desek nedisponuje možností vzdálené správy. Chybějící funkcionalita pro vzdálenou správu neexistuje obzvláště u levných základních desek. Vzhledem k této situaci vynikla potřeba počítače doplnit o jednoduchý a cenově dostupný modul, který by umožňoval vzdálenou kontrolu stavu napájení počítače a ovládání jednotlivých tlačítek počítače. Tato diplomová práce si klade za cíl navrhnout hardware a k němu příslušný software modulu, který by takovouto vzdálenou správu počítače umožňoval. U navrhovaného modulu je kladen důraz na cenovou dostupnost, univerzálnost navrhovaného modulu a další kritéria, která umožní jednoduché začlenění do stávající infrastruktury a případné využití tohoto modulu pro další úlohy.

2 Úvodní studie

2.1 Stávající stav a cíle

Hlavním úkolem této diplomové práce řešení vzdáleného ovládání počítačové učebny, které by umožňovalo monitorování stavu a ovládání jednotlivých počítačů v počítačové učebně. Toto řešení má zejména sloužit pro řešení hromadných aktualizací vypínání všech spuštěných počítačů a další automatizované úlohy. Důležitým kritériem návrhu je začlenění tohoto ovládání do stávající infrastruktury bez nutnosti řešit další propojení s centrálním ovládáním.

Stávající infrastruktura je navržena pro rozhraní ethernet dle standardu ANSI/TIA/EIA-568-B.

Původní návrh řešení počítal s použitím některého průmyslového standardu sériové sběrnice jako je RS485 a implementace této komunikace pomocí nějakého jednočipového počítače a vývoje centrálního modulu pro ovládání jednotlivých koncových modulů.

2.2 Funkční požadavky

- Snadné začlenění jednotlivých koncových modulů do stávající infrastruktury
- Možnost komunikovat s jednotlivými moduly například pomocí jednoznačných přidělených adres.
- Možnost komunikovat s moduly i v okamžiku kdy je počítač vzpnutý.
- Možnost sledovat minimálně 2 vstupní signály
- Možnost ovládat alespoň 2 standardní tlačítka počítače
- Použití standardizovaných metod komunikace s modulem
- Univerzální zapojení a galvanické oddělení vstupů a výstupů aby bylo možné zapojení k libovolné základní desce počítače
- Možnost snadného ovládání modulů z budoucích aplikací
- Možnost ovládání modulů obsluhou za použití prostředků dostupných na běžném osobním počítači bez potřeby vývoje dalšího software

2.3 Nefunkční požadavky

- Nízká cena jednotlivých modulů
- Použití snadno dostupných vývojových prostředků
- Přehledný a snadno udržovatelný zdrojový kód software pro jednotlivé moduly

3 Návrh řešení

3.1 Komunikace s modulem

Vzhledem k tomu že je stávající infrastruktura navržena pro rozhraní ethernet jako hvězdicová je použití sériové komunikace v navrhované podobě velmi obtížně realizovatelné. Realizace hvězdicového zapojení sériového zapojení by musela v centrálním uzlu obsahovat nějaký rozbočovač nebo by každý propoj hvězdicového zapojení musel obsahovat dvě linky spojené v koncovém uzlu aby bylo možné tyto propoje řetězit. Řetězení jednotlivých propojů by mělo značné omezení kvůli maximální délce sběrnice RS485 a bylo by možné tímto způsobem propojit velice malé množství koncových uzlů na krátké vzdálenosti. Tyto problémy lze vyřešit použitím jiného způsobu komunikace než je původně navrhovaná standardní sériová sběrnice.

Automatické přidělování adres je na standardní sériové sběrnici také poměrně složitá úloha která by musela vycházet z odlišných inicializačních hodnot jednotlivých modulů, jako je unikátní sériové číslo, nebo řetězením těchto modulů. Implementace pomocí jedinečného sériového čísla by musela řešit přidělování těchto jedinečných sériových čísel při zápisu programu do jednotlivých modulů. Zřetězení jednotlivých by bylo velmi náchylné při poruše jednotlivých modulů abylo by tak velmi nespolehlivé.

Z uvedených důvodů bylo zavrženo řešení pomocí sériové sběrnice a bylo místo toho zvoleno rozhraní ethernet pro které je původní infrastruktura navržena. Při použití komunikace přes rozhraní ethernet také není nadále nutné realizovat samostatný centrální modul a je možné koncové uzly ovládat přímo z osobního počítače obsluhy.

3.2 Napájení modulu

Požadavek, komunikovat s jednotlivými moduly i v okamžiku kdy je počítač vypnutý, je možné realizovat pomocí centrálního napájení, nebo napájení ze zdroje počítače pokud se jedná o zdroj který je schopen dodávat napájecí napětí i v době kdy je počítač vypnutý.

3.2.1 Centrální napájení

Centrální napájení musí brát v potaz limity kabelů pro ethernet, které jsou počítány pro poměrně malý proud, a možnost že dojde k náhodnému připojení jiného zařízení s rozhraním ethernet na tento kabel. Při náhodném připojení jiného zařízení s rozhraním ethernet na tento kabel by nemělo dojít k poškození tohoto náhodně připojeného zařízení. Řešení centrálního napájení by mělo pokud možno respektovat standardy pro rozhraní ethernet. Pro tuto variantu byl zvolen standard Power over Ethernet dle normy IEEE802.3af[11].

3.2.2 Napájení z počítače

U osobních počítačů s napájením dle standardu ATX je k dispozici pohotovostní napájecí napětí 5V i v okamžiku kdy je počítač vypnutý pokud je tento zdroj napájen ze sítě. Napájení ze zdroje ATX znamená větší zásah do hardware počítače protože pohotovostní napájení je k dispozici pouze na konektoru pro napájení základní desky počítače. Oproti realizaci napájení dle dtandardu PoE je napájení ze zdroje ATX podtatně levnější varianta. Při návrhu hardware koncových modulů byly implementovány obě varianty napájení.

3.3 Vstupy a výstupy

Navrhovaný modul by měl být co nejvíc univerzální a vhodný i pro jiné použití než je ovládání počítačů. Z toho důvodu jsem v návrhu počítal až s 8 vstupy a 8 výstupy modulu. Tento počet vstupů a výstupů byl zvolen s přihlédnutím k 8 bitové architektuře jednočipového počítače zvoleného pro realizaci koncových modulů. Výsledný modul lze pak osadit požadovaným počtem vstupů a výstupů podle zamýšleného použití.

4 Návrh architektury modulu

Vzhledem k zvolenému způsobu komunikace je možné realizovat modul s přijatelnými náklady pouze za použití jednočipového počítače. Návrh pomocí specializovaných obvodů vytvořených například pomocí hradlových polí by se návrh neúměrně prodražil. Vzhledem k požadavkům na cenu, galvanické oddělení vstupů a výstupů, použití rozhraní ethernet a standardu PoE jsem zavrhl použití hotových modulů s jednočipovými počítači, protože by cena těchto modulů byla několikanásobně vyšší, než je výsledná cena modulu navrženého přímo pro požadovaný účel.

Vzhledem k výborné podpoře ze strany výrobce, ceny a mým dlouholetým zkušenostem jsem pro realizaci zvolil jednočipové počítače PIC od firmy Microchip[8]. Pro realizaci tohoto modulu jsem zvolil typ jednočipového počítače PIC18F97J60[6] protože se jedná o nejlevnější typ tohoto výrobce disponující integrovaným rozhraním ethernet. Zvolený typ disponuje pouze rozhraním 10BASE-T ale většina současných síťových prvků tuto variantu stále ještě podporuje, takže by v tomto ohledu neměl být problém s kompatibilitou modulu s ostatními síťovými prvky.

4.1 Hlavní rysy vybrané architektury jednočipového počítače

- Jedná se procesor typu RISC
- 8 bitová architektura
- Harvardská architektura s oddělenou pamětí programu a dat
- Střadačová architektura s pracovním registrem pojmenovaným W
- Instrukční cyklus trvá 4 takty oscilátoru
- Většina instrukcí kromě skoků přeskoků instrukce a několika dalších případů trvá jeden instrukční cyklus
- Podpora programování paměti programu typu FLASH v aplikaci využitím 2 signálů pro data a taktovací signál a signálu reset pro programovací napětí
- Operace ALU probíhají mezi pracovním registrem W a registrem v paměti dat nebo mezi pracovním registrem W a konstantou obsaženou v instrukci
- Paměť programu je 12 nebo 16 bitová

4.2 Základní vlastnosti vybrané řady a typu jednočipového počítače pro tuto implementaci

- Taktovací frekvence v rozsahu 2.778MHz-41.667MHz při použití oscilátoru s frekvencí 25MHz
- Integrované hardwarové vrstvy MAC a PHY rozhraní Ethernet 10Base-T
- Napájecí napětí 3.3V
- Podpora proudů vybraných výstupů až do velikosti 25mA
- Podpora sběrnice I2C
- Programování paměti programu při programovacím napětí 3.3V inicializované pomocí speciální sekvence dat
- Paměť programu je 16 bitová o velikosti 128kiB (až 65532 instrukcí)
- Paměť programu je 3808 byte
- Bufer pro ethernet 8192 byte
- 70 vstupů/výstupů
- 100 pinové TQFP pouzdro

4.3 Taktování jednočipového počítače

Jako zdroj taktu procesoru byl použit krystal s frekvencí 25MHz dle katalogového zapojení[6]. Frekvence oscilátoru 25MHz je vyžadovaná pro správnou funkci ethernetové části jednočipového počítače. Vzhledem k relativně malé náročnosti prováděného programu nebyl použit integrovaný Fázový Závěs (PLL) a místo toho byl použita tato frekvence přímo jako takt procesoru. V této konfiguraci trvá instrukční cyklus 160ns a výkon je 3.125MIPS-6.25MIPS podle zastoupení jednotlivých typů instrukcí.

5 Návrh napájení modulu

5.1 Napájecí napětí použitá v modulu

Jako základní napájecí napětí modulu jsem zvolil napájení 5V. Použitý jednočipový počítač potřebuje také napětí 3.3V pro napájení periférií a dále napětí v rozsahu 2.35V-2.7V pro napájení jádra počítače.

Pro získání napětí 3.3V je na modulu osazen LDO stabilizátor 3.3V 100mA. Pro získání napájecího napětí pro napájení jádra byl použit interní napěťový regulátor tohoto jednočipového počítače.

5.2 Zdroje napájení 5V

Jako zdroj napětí 5V je zvolen přímý vstup pro napájení ze zdroje nebo vstup z modulu pro PoE. Jednotlivé zdroje napájení jsou odděleny použitím diod, takže lze modul provozovat i jako napájený z obou zdrojů fungujících jako vzájemná záloha napájení. Pro oddělení zdrojů napájení byla použity diody BAT60A které mají velmi malý úbytek napětí 0.2V takže pro napájení modulu zbývá napětí 4.8V což je dostatečné napětí pro všechna navrhovaná zapojení. Vstupy napájení jsou navíc opatřeny transily, které chrání tyto vstupy před napěťovými špičkami obzvláště před statickou elektrinou.

5.2.1 Napájení přes POE

Pro realizaci centrálního napájení přes PoE dle normy IEEE802.3af je nutné vyřešit následující požadavky:

- Počáteční signalizace že se jedná o zařízení způsobilé pro napájení po ethernetu
- Převod vstupního napětí v rozsahu 44V-57V na 5V pomocí DC-DC měniče
- Řízení připojení napájení po úspěšné počáteční signalizaci

Počáteční signalizaci a ovládání napájení lze řešit pomocí specializovaných obvodů, například obvodem TPS2370[15]. Velmi problematické jsou ale návrh a realizace DC-DC měniče kde by bylo zřejmě nutné spočítat a nechat vyrobit patričné transformátory. Problematické je také pro tato napětí koupit již hotový modul DC-DC měniče. Kvůli problémům s DC-DC měničem návrh vedoucí této diplomové práce použití již hotového modulu řešící celé napájení PoE.

Pro použití v tomto návrhu jsem nakonec vybral modul PEM1205[3] od firmy Infomart IT Solutions. Pro použití v tomto návrhu by byl vhodnější modul PEM1305 který je na rozdíl od PEM1205 osazen i diodovými můstky, u tohoto typu modulu však dodavatel uváděl poměrně dlouhou dobu dodání takže jsem nakonec použil typ PEM1205 doplněný externími diodovými můstky. Pro zapojení tohoto modulu v návrhu jsem použil katalogové zapojení udávané výrobcem modulu.

6 Návrh signalizačních LED diod modulu

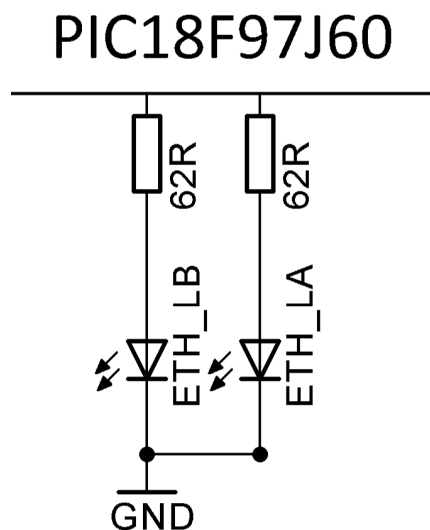
Pro signalizaci stavu a chyb jednočipového počítače jsem v návrhu použil celkem 6 led diod.

Dvě LED diody byly zapojeny na piny, které jsou přímo určeny pro signalizaci stavu ethernet adaptéru. Tyto led diody jsou zapojeny podle katalogového zapojení a jsou řízeny přímo na hardwarové úrovni ethernetovou částí procesoru. Minimální hodnota odporu předřadného rezistoru těchto LED diod byla vypočítán podle následujícího vztahu:

Úbytek napětí na vybraném typu LED diod: 2.1V Proud led diod: 20mA Výstupní napětí na pinech jednočipového počítače: 3.3V

$$R = \frac{3.3-2.1}{0.020} = 62\Omega$$

Jako nejbližší hodnota předřadných rezistorů byly použity rezistory s odporem 62Ω.



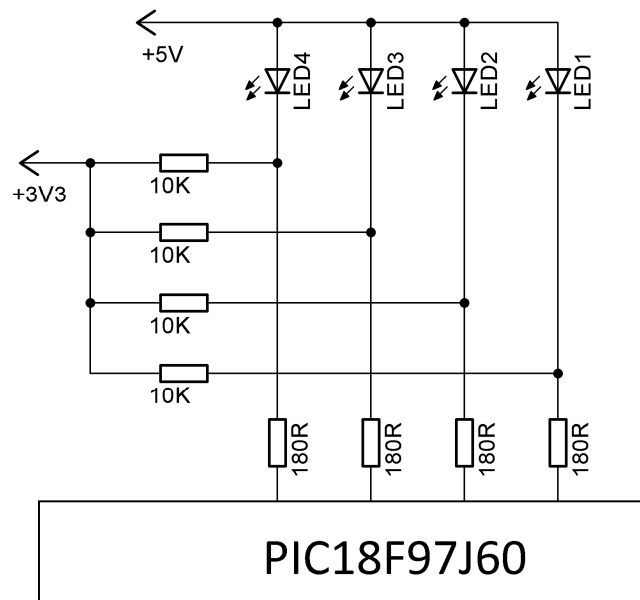
Obrázek 6.1: Schéma zapojení led diod ethernetu

Další 4 led diody jsou zapojeny na port B jednočipového počítače, který umožňuje odběr až 25mA na jednom pinu. Druhý vývod těchto LED diod byl zapojen na napájecí napětí 5V aby nedocházelo k dalšímu zatížení LDO 3.3V stabilizátoru. U zapojení těchto LED diod je využita skutečnost že mají úbytek napětí 2.1 V a při zapojení mezi napěťovými úrovněmi 5V a 3.3V na výstupním pinu nesvítí. Tyto LED diody začnou svítit teprve, když je na výstupním pinu napětí 0V. Minimální hodnota odporu předřadného rezistoru těchto LED diod byla vypočítán podle následujícího vztahu:

$$R = \frac{5-2.1}{0.020} = 145\Omega$$

Pro předřadné rezistory byla zvolena hodnota odporu 180Ω , která zahrnuje i určitou rezervu pro případ vyššího napájecího napětí.

K výstupním pinům pro tyto LED diody byly zapojeny ještě rezistory s odporem $10k\Omega$ na napájení $3.3V$. Tyto rezistory zajišťují aby se na vstupy ve stavu vysoké impedance nedostávalo napětí $5V$ vlivem svodu LED diod.



Obrázek 6.2: Schéma zapojení led diod signalizujících stav modulu

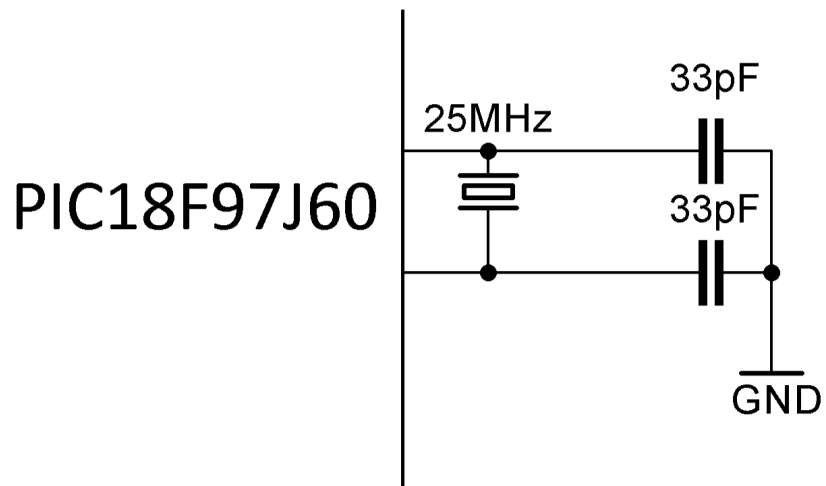
V návrhu modulu jsou ještě použity 3 LED diody signalizující stav napájení.

Dvě tyto LED diody signalizují přítomnost $5V$ z jednotlivých zdrojů napájení a jsou umístěny před diody zařazené do těchto napájení. Tyto LED diody mají předřadné rezistory s odporem 180Ω .

Třetí LED dioda signalizuje přítomnost napětí $3.3V$ a je zařazena za $3.3V$ stabilizátorem. Tato LED dioda je opatřena předřadným rezistorem s hodnotou 62Ω .

7 Návrh zapojení oscilátoru

Oscilátor je zapojen podle katalogového zapojení s použitím SMD krystalu doplněného keramickými kondenzátory s kapacitou 33pF dle doporučení výrobce. Odpor zapojený do série s krystalem nebyl v tomto případě použit (tento rezistor se používá pro krystaly s nízkým budícím výkonem).

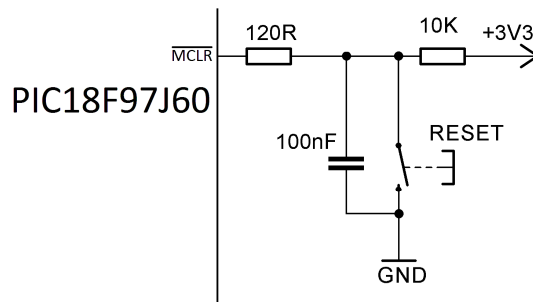


Obrázek 7.1: Schéma zapojení oscilátoru

8 Návrh ovládacích tlačítek

8.1 Tlačítko reset

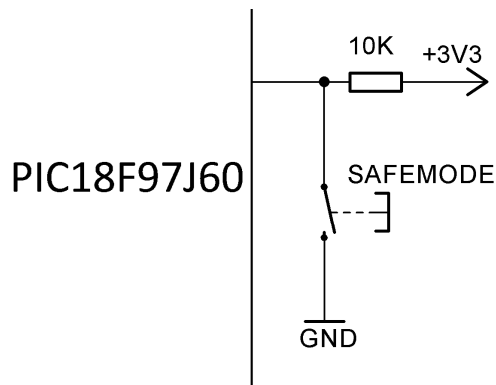
Tlačítko reset je zapojeno na signál \overline{MCLR} MCU a napájení 0V. Zapojení je doplněno zapojením doporučeným výrobcem skládající se z rezistoru s odporem $10\text{k}\Omega$ a kondenzátorem s kapacitou 100nF .



Obrázek 8.1: Schéma tlačítka reset

8.2 Tlačítko safemode

Tlačítko safemode je zapojeno na digitální vstup MCU a napájení 0V. Zapojení je doplněno rezistorem s odporem $10\text{k}\Omega$ zapojeným na napětí 3.3V .



Obrázek 8.2: Schéma tlačítka safemode

9 Návrh vstupů modulu

Jako jediná možná varianta pro realizaci vstupů při požadavku na galvanické oddělení je v tomto případě použití optočlenů. Pro tento modul jsem vybral optočleny H11L1M[4]. Tento optočlen jsem vybral s ohledem na nízkou cenu a velký rozsah spínacích proudů.

9.1 Zapojení emitoru optočlenu

Tento optočlen má dle specifikace výrobce rozsah vstupních proudů emitoru 1.6mA-30mA dále výrobce doporučuje o 10% větší proud než je minimální proud pro spolehlivé sepnutí. Maximální prahové napětí je 1.5V . Pro vstupy bylo požadováno vstupní napětí minimálně v rozsahu 3V-12V.

Na základě parametrů optočlenu a požadovaného vstupního napětí lze určit maximální velikost odporu předřadných rezistorů dle vztahu:

$$R = \frac{3-1.5}{0.0016*1.10} = 852\Omega$$

Minimální ztrátový výkon použitého rezistoru musí být při minimálním udávaném strátovém napětí emitoru 1.2V:

$$P = \frac{(12-1.2)^2}{852} = 0.137W$$

Tento ztrátový výkon přesahuje povolený ztrátový výkon použitých běžných SMD rezistorů, který je 0.125W. Pro vstupy jsem proto zvolil použití dvou předřadných rezistorů s odporem 390Ω. Parametry vstupů jsou pro tuto velikosti shrnuty v následující tabulce.

Velikost odporu předřadných rezistorů	780Ω
Minimální hodnota proudu pro 3V	1.9mA
Typická hodnota proudu pro 12V	13.9mA
Min. hodnota vst. napětí při dodržení min. vst. proudu +10%	2.75V
Max. velikost vst. napětí při dodržení max. ztrát. výkonu rezistorů	15V

Tabulka 9.1: Charakteristiky vstupů

U vstupů je také možné zkratovací propojkou rezistory vyřadit a použít emitor bez těchto rezistorů. Varianta bez předřadných rezistorů je určena zejména pro zapojení do série s kontrolkami čelního panelu počítače.

Dále byla antiparalelně s LED diodou emitoru zapojena dioda, která chrání LED diodu před zničením, pokud by bylo na vstup přivedeno napětí s opačnou polaritou.

9.2 Zapojení detektoru optočlenu

Detektor optočlenu byl zapojen podle katalogového zapojení kdy je optočlen napájen 5V a výstup s otevřeným kolektorem je propojen přes rezistor s odporem 270Ω na napájecí napětí 5V. Vybraný typ jednočipového počítače má 5V-tollerant digitální vstupy ale

netýká se to všech vstupů a také záleží na nastavení konkrétního vstupu. Kvůli potencionálním problémům s 5V signály bylo původní zapojení doplněno ještě o rezistor s odporem 390Ω zapojený na 0V. Doplněný rezistor 390Ω tvoří s rezistorem 270Ω napěťový dělič, který přizpůsobuje výstupní napětí pro vstup jednočipového počítače. Kvůli velmi omezenému výstupnímu proudu LDO stabilizátoru pro 3.3V byl použit dělič napětí místo zapojení rezistoru výstupu na napájení 3.3V. Výstupní napětí děliče je dáno vztahem:

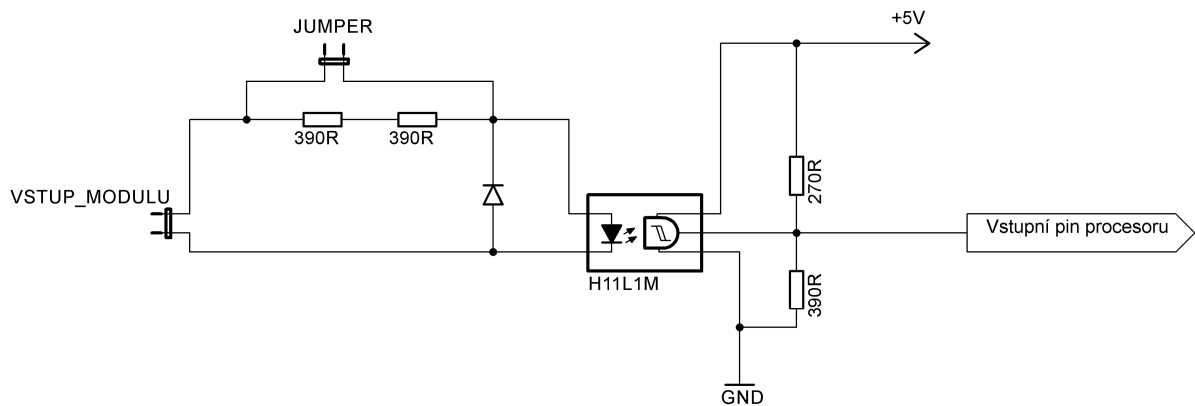
$$U = \frac{4.8 \cdot 390}{270 + 390} = 2.84V$$

Klidový proud děličů všech vstupů při rozepnutých vstupech je:

$$I = \frac{4.8}{270 + 390} * 8 = 58mA$$

Tento výsledný proud není pro klidový odběr modulu nijak kritický.

Kvůli poměrně malému maximálnímu proudu detektoru optočlenu nebyly pro vstupy použity signalizační LED diody.



Obrázek 9.1: Schéma zapojení vstupů

10 Návrh výstupů

Pro výstupy byly, s ohledem na maximální kompatibilitu pro simulaci stisknutí tlačítek, v návrhu použita mechanická relé.

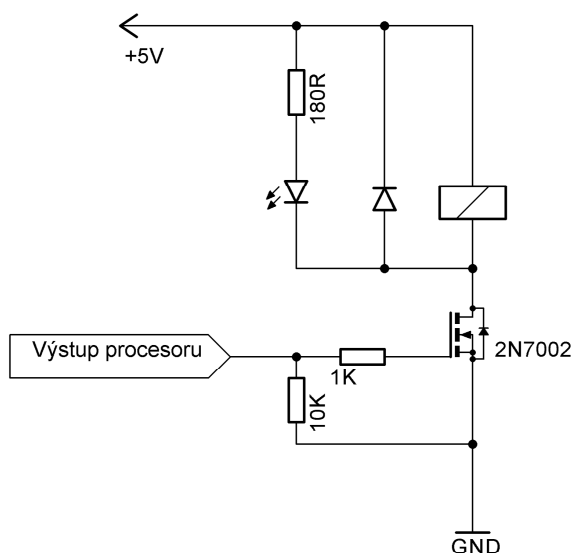
Pro prvních 6 výstupů byla zvolena možnost osadit jazýčková relé do plošného spoje s jedním spínacím kontaktem typu RELES1A050000.

Kvůli možnosti použít hardwarové vypnutí počítačem přerušení signálu PS_ON na pinu 16 napájecího ATX konektoru základní desky byly poslední 2 pozice pro relé navrženy pro relé s přepínacím kontaktem typu N4100CHS3DC05V.

Oba typy použitých relé mají cívku určenou pro stejnosměrné napětí 5V. Pro oddělení cívek relé od jednočipového počítače byly použity budící tranzistory N-Channel MOS-FET typu 2N7002. Tranzistory byly doplněny o rezistor s odporem $1\text{k}\Omega$ zapojeným mezi výstupní pin procesoru a hradlo tranzistoru. Tento rezistor limituje špičkový proud tekoucí do hradla tranzistoru při náběžné nebo sestupné hraně řídicího signálu způsobenou kapacitou hradla. Dále byl použit rezistor $10\text{k}\Omega$ připojený mezi výstupní pin jednočipového počítače a napájecí napětí 0V. Druhý použitý rezistor zajišťuje vypnutí tranzistoru, pokud je výstupní pin jednočipového počítače ve stavu vysoké impedance (například po resetu procesoru). Hodnoty obou použitých rezistorů byly určeny empiricky.

Paralelně s cívkou relé byly zapojeny ochranné diody, které zabraňují tomu, aby se napěťové špičky indukované po rozeptnutí budícího tranzistoru přenášely do napájecího napětí (budící tranzistory disponují vlastní ochranou diodou).

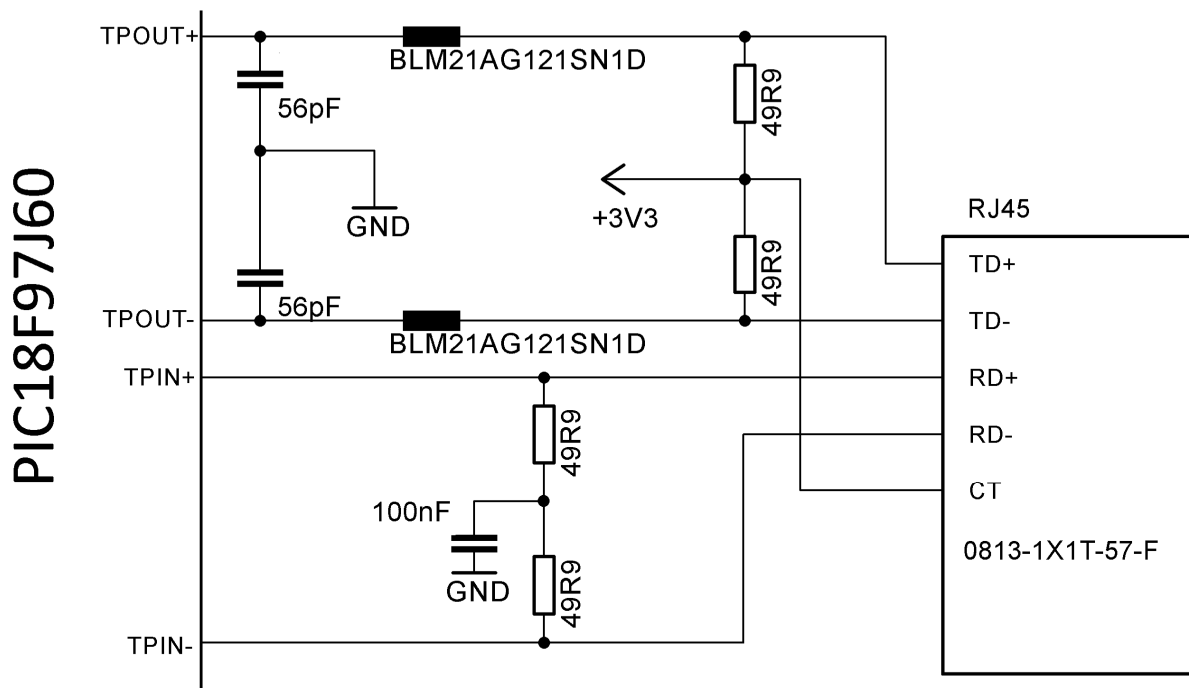
Dále jsou paralelně s cívkou zapojeny signalizační LED diody s předřadným rezistorem s odporem 180Ω , aby bylo možné poznat dle těchto LED diod stav jednotlivých výstupů.



Obrázek 10.1: Schéma zapojení výstupů

11 Návrh zapojení ethernetu

Pro ethernetovou část jsem použil katalogové zapojení ethernetu uváděné v dokumentaci použitého jednočipového počítače výrobcem. Pro externí připojení jsem použil zásuvku RJ45 s integrovanými transformátory typu 0813-1X1T-57-F[1] od výrobce Bel Fuse Inc. . Tento konektor jako jediný typ u vybraného dodavatele obsahoval kombinaci integrovaných transformátorů s vývody pro Power over Ethernet. Zapojení tohoto konektoru se katalogového zapojení obsaženého v dokumentaci jednočipového počítače liší tím, že má propojené středy oddělovacích transformátorů na jednom výstupním pinu. Konstrukce tohoto konektoru naštěstí umožňuje relativně snadnou úpravu zapojení což mi umožnilo odpojit od výstupního pinu střed cívky vstupního transformátoru a tím dodržet katalogové zapojení doporučené výrobcem jednočipového počítače.



Obrázek 11.1: Schéma zapojení ethernetu

12 Návrh schéma zapojení a desky tištěného spoje

12.1 Použitý software

Pro návrh schéma zapojení a desky tištěného spoje byla použita freewarová verze programu pro návrh DPS Eagle firmy CADSoft[2]. Desku tištěných spojů jsem navrhoval v této aplikaci ručně bez přímého použití dříve navrženého schéma a autorouteru. Tento postup jsem zvolil na základě dřívějších špatných zkušeností s autorouterem této aplikace a také kvůli specifickým nárokům na stínění, odrušení a dalších specifických požadavků tohoto návrhu.

12.2 Schéma zapojení

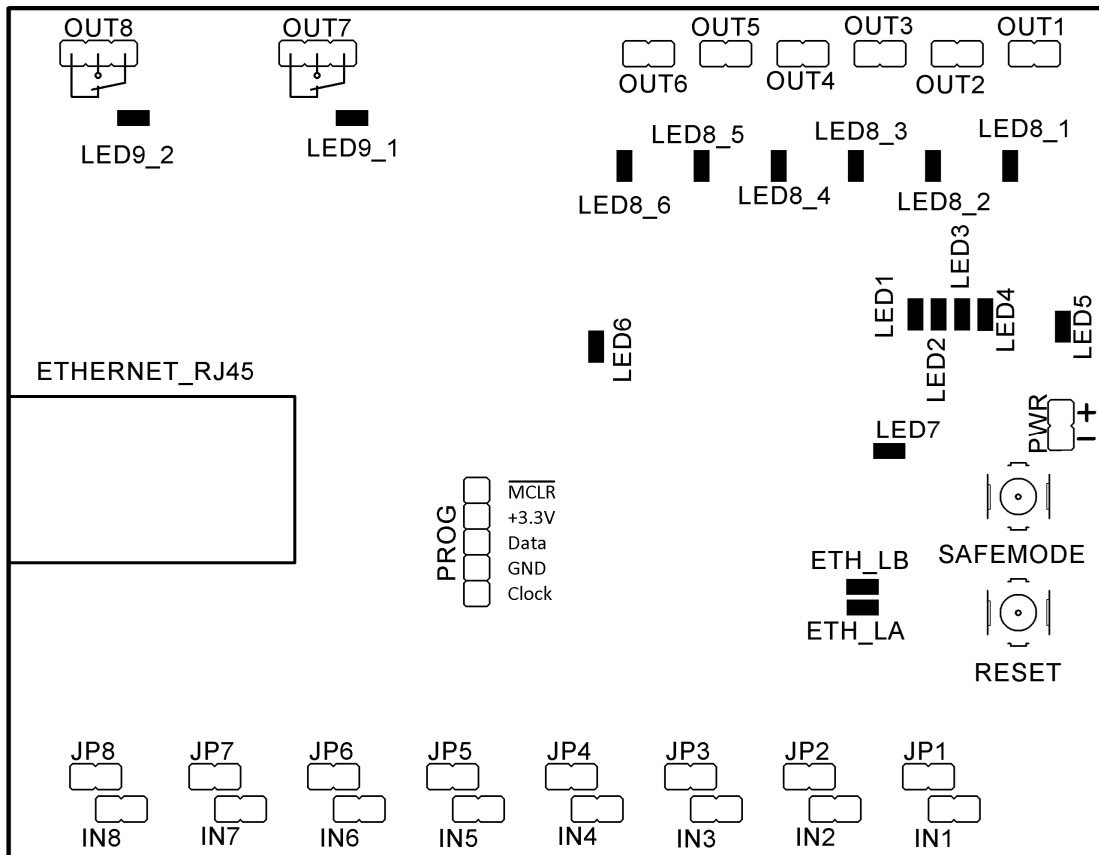
Návrh schéma spojení byl pro větší přehlednost rozčleněn na 3 části:

- `napajeni.sch`: Obsahuje schéma zapojení napájení modulu
- `procesor.sch`: Obsahuje zapojení procesoru, připojení ethernetu, signalizačních LED diod a dalších podpůrných obvodů MCU.
- `periferie.sch`: Obsahuje schéma zapojení vstupů a výstupů modulu.

Při návrhu tištěného spoje jsem se snažil většinu SMD součástek umístit na spodní stranu tištěného spoje ostatní součástky na stranu vrchní. V příložených datových souborech návrhu tištěného spoje jsou kvůli snadnějšímu návrhu, při použití jednočipového počítače jako ústřední součástky návrhu, použity jednotlivé vrstvy opačně aby nebyl v době návrhu jednočipový počítač zrcadlově převrácený.

Jako dost značný problém se při tomto návrhu ukázalo omezení freewarové verze programu, která umožňuje rozmístit součástky na ploše do velikosti 100x80mm. V popisu aplikace je uvedeno, že se jedná o limit velikosti tištěného spoje. Toto tvrzení není zcela přesné, spoje mezi součástkami a další čáry lze umístit i mimo tuto oblast ale jednotlivé součástky musí být umístěny v této vymezené oblasti. Omezení velikosti tištěného spoje je ve výsledném návrhu vykoupeno větším počtem SMD součástek na vrchní straně tištěného spoje a větším počtem propojů mezi spodní a vrchní stranou tištěného spoje. Výsledný návrh tištěného spoje jsem použil pro výrobu tištěného spoje fotocestou.

13 Popis základních prvků modulu



Obrázek 13.1: Schéma ovládacích prvků a konektorů modulu

- **ETH_LA:** červená LED dioda sloužící k indikaci příchozích komunikace RX ethernetového modulu
- **ETH_LB:** zelená LED dioda sloužící k indikaci odchozí komunikace TX ethernetového modulu
- **LED1:** zelená LED dioda indikující ukončení základní inicializace hardware modulu
- **LED2:** zelená LED dioda indikující ukončení inicializace ethernetového modulu - v safe módu tato LED dioda bliká
- **LED3:** červená LED dioda indikující chybový stav modulu
- **LED2:** žlutá LED dioda indikující stav linky ethernetového modulu - tato LED je ošetřena softwarově v hlavním cyklu programu

- **RESET:** tlačítko kterým lze provést hardwarový reset procesoru modulu
- **SAFEMODE:** Tlačítko kterým se vynutí, aby modul použil výchozí nastavení IP protokolu (slouží pro nastavení IP adresy při neznámé aktuální adrese)

- **IN1-IN8:** Opticky oddělené vstupy modulu
- **JP1-JP8:** Zkratovací propojky pro vyřazení vstupních rezistorů
- **OUT1-OUT8:** Výstupy modulu - kontakty výstupních relé
- **LED8_1-LED8_6,LED9_1,LED9_2:** zelené LED diody signalizující sepnutí jednotlivých výstupů

- **PWR:** 5V externí napájení modulu

- **LED5:** červená LED indikující externí napájení 5V
- **LED6:** červená LED indikující napájení přes ethernet pomocí PoE
- **LED7:** červená LED indikující napájecí napětí 3.3V

- **ETHERNET_RJ45:** Konektor pro připojení ethernetu
- **PROG:** Konektor sloužící pro nahrání programu do MCU

14 Návrh software - Architektura

Veškerý software pro návrh modulu spočívá ve vytvoření programu umístěného v jednočipovém počítači. Pro tento návrh nebyly navrženy žádné další aplikace umístěné mimo modul.

Pro návrh software jsem použil integrované vývojové prostředí MPLAB [9] a překladač jazyka C XC8 [10] od firmy Microchip. Oba tyto nástroje jsou dodávány v základní variantě zdarma.

Jako základ pro vývoj software posloužil TCP/IP stack obsažený v "Microchip Libraries for Applications" dostupný na stránkách firmy Microchip [7] ve verzi v2014-07-22. Ve zvolených knihovnách chybí knihovna nezbytná pro použití TCP/IP stacku s vybraným MCU. Chybějící knihovnu ETH97J60 jsem byl nucen doplnit ze starší legacy verze TCP/IP stacku v2013-06-15.

15 Návrh ovládání modulu

Požadavky na software modulu:

- použití standardního protokolu nad TCP/IP protokolem
- možnost ovládat modul automaticky
- možnost ovládat modul ručně
- při návrhu reflektovat velmi omezenou velikost paměti programu a paměti dat MCU

Při návrhu ovládání modulu jsem vycházel ze skutečnosti, že součástí TCP/IP stacku je HTTP webový server.

Na základě toho že součástí požadavků na realizaci modulu je i možnost ovládat tento modul automaticky jsem zavrhl možnost ovládat modul přímo přes webové stránky. Ovládání přes webové stránky se velmi špatně volá z jiné aplikace. Pro ovládání modulu jsem za těchto podmínek považoval implementovat nějaký standardní RPC (Remote Procedure Call) protokol. Implementace binárních protokolů jako je CORBA nebo Java RMI by neumožňovala použít HTTP server a bylo by nutné implementovat tyto protokoly celé. Pro snadnější implementaci s využitím HTTP serveru jsem se snažil vybírat některý z RPC protokolů které jsou textové a mohou být implementovány na tomto serveru. Zpracování protokolů pracujících s XML jako je například SOAP nebo XMLRPC vyžaduje pro svojí implementaci kvůli nutnosti zpracovat XML poměrně dost datové paměti (zejména při kontrole validity vstupního XML). Jako nejvhodnější jsem nakonec zvolil protokol REST[14]. Protokol REST není dle definice přímo RPC protokol ale prakticky je možné tento protokol pro RPC model použít. Pro jednoduchost zpracování vstupních dat, úspornost formátu a jednoduché volání z jazyka javascript jsem nakonec vybral formát pro výměnu dat JSON[12]. Aby bylo možné obsluhovat modul přímo bez nutnosti dalšího software tak je modul doplněn o statické stránky ze kterých jsou volány jednotlivé služby z javascriptu použitím knihovny jQuery[5].

16 Členění software - knihovny

16.1 TCP/IP stack

Knih

- **MAC vrstva:** Tato vrstva zajišťuje základní komunikaci po rozhraní ethernet. Knihovny: ETH97J60.h, ETH97J60.c, mac.h, mac.c, delay.h, delay.c, tick.c, tick.h, stack_task.h, stack_task.c, helpers.c, helpers.h
- **IP vrstva s podporou protokolu ARP:** Tato vrstva řeší základní komunikaci IP protokolu. Knihovny: arp.h, arp.c, ip.h, ip.c, tcpip.h, tcpip_types.h
- **UDP protokol:** Tato vrstva slouží k UDP komunikaci, kterou využívají protokoly NETBIOS, DHCP a ICMP. Knihovny: udp.h, udp.c
- **TCP protokol:** Tato vrstva slouží k vlastní komunikaci se zařízením převážně pro komunikaci s HTTP serverem. Knihovny: tcp.h, tcp.c
- **NETBIOS:** Tato vrstva umožňuje zobrazení názvu zařízení v síti. Knihovny: nbns.h, nbns.c
- **DHCP client:** Tato vrstva umožňuje dynamické získání IP adresy pomocí DHCP protokolu. Knihovny: dhcp.h, dhcp_client.c
- **ICMP protokol:** Tato vrstva umožňuje, aby modul odpovídal na ICMP pakety jako je PING. Knihovny: icmp.h, icmp.c
- **HTTP webový server:** Tato vrstva implementuje HTTP webový server, který slouží k vlastní komunikaci s modulem a jeho ovládání. HTTP server obsahuje jednoduchý souborový systém pro ukládání jednotlivých stránek a souborů nazvaný "Microchip File System" (MPFS) verze 2. Knihovny: http2.h, http2.c, mpfs2.h, mpfs2.c

Všechny uvedené knihovny jsou umístěny v podadresáři TcpIp projektu sloužícího pro vývoj software pod IDE MPLAB.

16.2 Ostatní knihovny programu

- **system_config.h, system_config.c:** zde jsou umístěny definice hardwarové konfigurace modulu a inicializace hardware MCU
- **utils.h, utils.c:** tato knihovna je určena pro pomocné metody využívané v rámci aplikace (v tuto chvíli obsahuje metody pro ovládání LED diod)
- **JSON.h, JSON.c, JSONParams.h, JSONParams.c:** knihovny implementující formát pro výměnu dat s modulem JSON
- **http_print.h:** soubor vygenerovaný generátorem MPFS systému implementující zpětné volání metod na základě obsahu souborů HTTP serveru (volaná metoda je jedna a je umístěná v knihovně JSON.c)

- **MPFSImg.c:** soubor vygenerovaný generátorem MPFS systému (po vygenerování je potřeba vždy upravit blok definic jak bylo uvedeno dříve)
- **ServiceMethods.c:** knihovna obsahující implementaci funkcionality metod služeb JSON
- **Main.c:** knihovna obsahující hlavní část programu ze které se volají všechny další funkce knihoven.

17 Úpravy TCP/IP stacku

Aktuální verze TCP/IP stacku na stránkách firmy Microchip není v tuto chvíli správně přizpůsobená k použití s překladačem XC8. Pro zprovoznění TCP/IP stacku bylo nutné doplnit veškeré vzájemné importy knihoven, protože ty v dodané verzi zcela chyběly a při překladu s implicitním nastavením docházelo k velkému množství chyb způsobených nevyřešenými referencemi.

Velké množství chyb a problémů bylo dále obsaženo v definici maker převážně při použití konstrukce jazyka sizeof. V dokumentaci překladače XC8 je uvedeno, že v době překladu lze použít sizeof pouze na jednoduché datové typy. Na větším počtu míst byla v knihovnách použita konstrukce sizeof na složené typy a tím docházelo k špatným výpočtům základních adres a velikostí bufferů. Také dodané knihovny obsahovaly některé nesprávné nebo chybně umístěné definice maker nahrazující některé knihovní funkce. Definice maker také obsahovaly chybné definice některých konstant pro nastavení velikostí bufferů, počtů připojení a podobných definic.

18 Souborový systém MPFS

Data souborů použitých webovým serverem jsou ukládána do jednoduchého souborového systému MPFS verze 2 navržený firmou Microchip který lze umístit přímo v paměti programu MCU. Tento souborový systém obsahuje jednoduchou tabulku názvů a umístění souborů, která se prochází sekvenčně na úplnou shodu názvu souboru. Za tabulkou názvů jsou umístěny vlastní data souborů. Soubory, které nejsou přímo používány programem, jsou v tomto souborovém systému uloženy jako komprimované metodou GZIP. Knihovna pro čtení souborů z tohoto souborového systému nemá implementovanou dekompresi komprimovaných souborů. Z tohoto důvodu musí dekompresi provádět přímo prohlížeč. Většina současných běžných prohlížečů tuto dekompresi podporuje. Data souborového systému jsou generovány aplikací v jazyce JAVA dodávanou firmou Microchip.

Během vývoje aplikace jsem zjistil, že definice vygenerovaných dat obsahuje závažnou chybu. Po vygenerování knihovny byla při překladač použita sekce určená pro překladač XC8. Bohužel při překladač došlo k tomu, že se jednotlivé bloky, na které byla definice rozdělena, neumístili v paměti ve správném pořadí. Tato chyba způsobila, že při čtení dat umístěných dále než v prvním bloku docházelo ke čtení nesprávných dat. Jednou z možností bylo upravit MPFS knihovnu tak aby načítala data po jednotlivých blocích a adresu každého bloku načítala zvlášť. Variantu s úpravou knihovny jsem nakonec zavrhl, protože by byla poměrně složitá a výsledný program by byl o dost pomalejší. Tento problém jsem nakonec vyřešil tak že po vygenerování knihovny jsem doplnil definice o pevné adresy bloků, čímž jsem zajistil to, že jsou v paměti programu ve správném pořadí. Dále bylo nutné řádkovým parametrem překladače ROM zajistit, aby linker nepoužil tuto pevně definovanou oblast pro jiná data.

19 Základní nastavení TCP/IP stacku

Většina základních nastavení TCP/IP stacku je obsažena v definičním souboru TCPIP-Config.h . Tento soubor byl převzat z TCP/IP stacku a upraven podle potřeb aby konfigurace odpovídala aktuálním potřebám pro implementaci software modulu.

19.1 MAC adresa

Jeden ze základních problémů bylo získání pokud možno jedinečné hardwarové MAC adresy pro jednotlivé moduly. Jednočipový počítač PIC18F97J60 obsahuje hardwarovou MAC vrstvu bohužel, ale nemá od výrobce přidělenou jedinečnou MAC adresu. Zkoumal jsem jednotlivé možnosti pro registraci a získání bloku MAC adres ale tyto registrace byly dosti složité a finančně náročné. Nakonec jsem zjistil, že firma Microchip vyrábí také EEPROM paměti s readonly částí paměti která obsahuje jedinečnou MAC adresu. Pro získání jedinečné 48 bitové MAC adresy jsem nakonec použil zakoupenou paměť 24AA02E48. Tuto variantu jsem objevil až po kompletním návrhu a osazení modulu paměť tudíž už není přímo osazená na tomto modulu, ale posloužila pouze jako zdroj MAC adresy. K tomu aby byla paměť přímo součástí modulu, by bylo možné použít úpravu zapojení a připojit tuto paměť na nepoužité piny 87 a 88 jednočipového počítače. Na těchto pinech je hardwarová implementace sběrnice I2C. Další možností by bylo připojit u stávajícího zapojení modulu na konektor sloužící pro zápis programu do jednočipového počítače. Na programovacím konektoru jsou dostupné 2 datové piny s vnitřními PULL-UP rezistory a napájení 3.3V. Na tento konektor lze tudíž připojit paměť osazenou na samostatném modulu. Pro připojení paměti přes programovací konektor by bylo ale nutné doplnit program o softwarovou implementaci I2C sběrnice (datové piny pro programování MCU nemají hardwarovou implementaci I2C sběrnice).

19.2 IP adresa

Jako základní výchozí IP adresu jsem použil adresu 192.168.1.100 s maskou 255.255.255.255 a bránou 192.168.1.1 . Vybraná IP adresa je součástí jednoho z adresních prostorů určených pro vnitřní síť. Možnosti nastavení jiné IP adresy budou uvedeny dále v popisu jednotlivých funkcí pro ovládání modulu. DHCP je v základním nastavení vypnuto.

19.3 NETBIOS jméno

Jako jméno modulu jsem zvolil jméno "ETH_PC_CN_" doplněné o poslední 2 byte MAC adresy v hexadecimálním tvaru.

20 Implementace protokolu REST

20.1 Metody volání služeb

Pro komunikaci protokolem REST jsou ve stávající implementaci implementovány pouze volání GET a POST. Pro tyto metody volání nebylo nutné upravovat stávající implementaci HTTP serveru a pro implementaci služeb jsou zcela dostačující.

20.2 Aplikace pro tvorbu dat MPFS systému a definice typu souborů HTTP serveru

Pro implementaci služeb byly doplněny do definic serveru a konfigurace generátoru MPFS knihovny soubory následujícího typu:

Přípona	Typ v software	Mime typ	Popis
svc	HTTP_SVC	application/json	Soubory obsahující vlastní definici služeb
wsp	HTTP_WSP	application/json	Definice rozhraní služeb ve formátu WSP
wsdl	HTTP_WSDL	text/xml	Definice rozhraní služeb ve formátu WSDL

Tabulka 20.1: HTTP server - přidané typy souborů

Pro implementaci protokolu REST bylo nutné upravit stávající zpracování URL adresy tak aby bylo možné poslední část adresy použít jako jméno metody služby a jako název souboru se zpracovala URL adresa bez této části. Této funkcionality bylo dosaženo tak že se v URL adrese vyhledá řetězec ".svc/" a pokud je nalezen tak se jako jméno souboru uvažuje bez nalezeného znaku "/" a všech dalších znaků následující za tímto znakem. Sekvence následující po nalezeném znaku "/" se zpracuje samostatně a je chápána jako název metody služby. Zpracování části URL adresy chápané jako název souboru v MPFS systému je ponecháno původní a je case-sensitive. Zpracování názvu metody a bloku vstupních hodnot je už naprogramováno tak že na velikosti písmen nezáleží.

Jako další úprava zpracování URL adresy bylo doplněno že nelze externě zobrazit soubory začínající znakem ".". Soubory začínající tečkou jsou chápány jako skryté soubory pro interní použití. Dále bylo doplněna funkcionality že pokud má soubor příponu ".svc" a následuje parametr "?wsp" tak se název souboru (za posledním znakem "/") doplní na začátku znakem "." a změní se přípona na ".wsp". Doplněním těchto 2 funkcí je umožněno zobrazení definic služeb ve formátu WSP[13]. Definice jsou v software modulu vloženy jako statické soubory. WSP definice jsou vytvořeny ručně. Součástí aplikace měly být i definice ve formátu WSDL které byly vygenerovány pomocnou aplikací napsanou .NET Framework sloužící pro návrh rozhraní a snadnějšímu ladění použitých webových stránek. WSDL byly ve finální implementaci vynechány z důvodu příliš velké velikosti souborového systému MPFS. Vlastní implementace je v programu obsažena ale je z důvodu absence WSDL souborů zakomentována (http.c řádky 550-552).

21 Implementace služeb a formátu pro výměnu dat JSON

21.1 Implementace služeb

Implementace vlastního zpracování služeb je řešena přes parametry uzavřené mezi znaky "~". Takto uvedené parametry převede MPFS generátor na volání metod HTTPPrint. Všechny soubory pro služby obsahují pouze řetězec "~json(x)~" kde x je číslo služby předávané metodě pro zpracování volání s názvem HTTPPrint_json. Tato metoda je implementována v knihovně JSON.c a obsahuje vlastní zpracování volání.

21.2 Zpracování vstupu služeb

21.2.1 Identifikace metody služby

Ve funkci se nejdříve zpracuje název metody a převede se na celé číslo jednoznačně identifikující název metody. Pro identifikaci názvu metody se používá kombinace metod rekurzivního sestupu a porovnání zbylé části řetězce po jednotlivých znacích, tento postup byl zvolen kvůli efektivnímu zpracování, minimalizaci potřebné paměti a řešení převodu velikosti písmen. Pokud není název metody identifikován jako jeden z existujících názvů metod, tak se vypíše do výstupu chybová zpráva a ukončí zpracování.

21.2.2 Zpracování bloku vstupních parametrů

Po identifikaci názvu metody se provede zpracování bloku vstupních parametrů. Zpracování bloku parametrů se provádí najednou a implementace nepočítá s načítáním další části parametrů. Vstupní buffer sloužící pro ukládání přijatých parametrů je volen tak aby i nejdelší přípustná kombinace vstupních parametrů se s určitou rezervou vešla do tohoto bufferu. Vlastní zpracování vstupních parametrů je implementováno v knihovně JSONParams.c. Zpracování parametrů je navrženo tak aby bylo co nejvíce robustní a zpracuje i některé vstupy které jsou v rozporu s původní definicí. Zpracování parametrů je implementováno v nezbytné míře pro použití v této aplikaci a neimplementuje vlastnosti, které nejsou pro tuto aplikaci potřeba.

Vlastní zpracování bloku vstupních parametrů je implementováno jako zpracování jednorozměrného pole začínající a končící složenou závorkou kde jsou jednotlivé položky oddělené čárkou. Každá položka je chápána jako dvojice hodnot oddělená dvojtečkou. Zpracování jednotlivých částí probíhá podle následujícího postupu:

1. Vynechají se všechny počáteční mezery
2. Zkontroluje se první znak následující sekvence:
 - (a) Znak je apostrof nebo uvozovka: přeskočí se všechny mezery za prvním znakem a zpracuje se další řetězec, dokud není nalezen stejný znak jako na začátku
 - (b) Je nalezen jiný znak: zpracuje se další řetězec, dokud není nalezen jeden z očekávaných oddělovačů (bílý znak, čárka, dvojtečka, pravá složená závorka)

3. Při zpracování se data kopírují do pomocného bufferu a probíhají převody některých znaků, jako jsou escape sekvence formátu JSON a převod velkých písmen na malá
4. Nakonec se oříznou ve výstupním bufferu všechny mezery z konce řetězce

Metoda knihovny JSONParams.c s názvem JSONParseParam se snaží vždy načíst jeden parametr jako dvojici oddělenou dvojtečkou. Na každou ze dvou částí zavolá dříve popsané zpracování částí.

21.2.2.1 Zpracování názvu parametru

Na první část obsahující název parametru po jeho předzpracování zavolá metodu pro identifikaci parametru. Metoda pro identifikaci parametru převádí název parametru na prvek výčtového pole jednoznačně identifikující parametr. Pokud není název identifikován metoda JSONParseParam vrátí hodnotu signalizující nalezení neexistujícího parametru. Metoda pro identifikaci parametru používá kombinace metod rekurzivního sestupu a porovnání zbylé části řetězce pomocí metody pro porovnání. V tomto případě jsem nepoužil porovnávání metod po znacích protože názvy parametrů některých parametrů jsou delší než názvy metod a nebylo potřeba při tomto porovnání řečit velikost písmen (převod velikosti písmen probíhá v rámci předzpracování).

21.2.2.2 Zpracování hodnoty parametru

Na druhou část obsahující hodnotu parametru po jeho předzpracování se obsah zpracuje v závislosti na dříve identifikovaném názvu parametru. Názvy parametrů jsou až na parametr Mask jednoznačné a je tudíž možné zpracovat jejich obsah jen na základě jejich názvu. Pro parametr Mask používá metoda JSONParseParam vstupní parametr který na základě dříve identifikovaného názvu metody určuje, zda se jedná o masku podsítě IP adresy nebo o masku nastavení výstupů v IO módu. Zpracování hodnot parametrů probíhá jako zpracování jednoho z následujících typu hodnoty:

1. Hodnota typu boolean : akceptuje hodnoty [0,n,no,ne,f,false] pro hodnotu false a hodnoty [1,y,a,yes,ano,t,true] pro hodnotu true
2. Číslo: akceptuje kladná čísla v jedno z následujících formátů (záporná čísla se nikde nepoužívají):
 - (a) V desítkové soustavě: číslo obsahuje číslice 0-9
 - (b) V šestnáctkové soustavě: číslo začíná sekvencí 0x a po této sekvenci obsahuje číslice 0-9 a znaky a-f
 - (c) V dvojkové soustavě: číslo začíná sekvencí 0b a po této sekvenci obsahuje číslice 0 a 1.

Všechna čísla musí být bez mezer nebo jiných než uvedených znaků (počáteční a koncové mezery byly ořezány v předchozím zpracování). Délka jednotlivých čísel není kontrolována, ale je hlídáno jejich přetečení (v tomto kroku je použit rozsah čísla 32 bitů). Rozšíření o zápis šestnáctkové a dvojkové číselné soustavy je rozšíření původní definice formátu JSON a tato čísla by tudíž měla být posílána v souladu s definicí jako string uzavřený v uvozovkách (z hlediska programu to není nezbytné, program tuto skutečnost nekontroluje)

3. IP adresa ve formátu A.B.C.D kde jednotlivé části musí být čísla v rozsahu 0-255 a maximálně 3 znaky dlouhé. IP adresa by neměla obsahovat mezery ale zpracování je napsáno tak že mezery před tečkou nebo za tečkou nevyhodnotí jako chybu ale řetězec korektně zpracuje. Zpracování IP adresy počítá pouze s decimálním zápisem a jiný zápis čísla nepřipouští.
4. Výčtový typ: metoda služby pro zápis konfigurace ovládání PC (bude popsána dále) používá 2 parametry s dvouznakovými výčtovými typy, tyto parametry se zpracovávají samostatně přímo v metodě JSONParseParam.

Jiné než uvedené formáty parametrů se nezpracovávají, protože jiné vstupní hodnoty nejsou v současné době implementovány.

21.2.3 Zpracování jednoho parametru

Metoda pro zpracování parametrů vrací JSONParseParam vrací jako návratovou hodnotu typ parametru. Pokud by parametr obsahoval název, ale chyběla by hodnota tak metoda vrátí hodnotu signalizující prázdný parametr (na základě této skutečnosti se později signalizuje chyba). Pomocí odkazů na proměnou metoda JSONParseParam předá také získanou hodnotu parametru. Hodnota parametru je převedena vždy na kladné 32 bitové číslo. Po návratu z funkce JSONParseParam se zkontroluje návratová hodnota funkce. Pokud se zpracování parametru nezdaří z důvodu neznámého názvu, neplatné hodnoty nebo prázdné hodnoty tak se vypíše do výstupu chybová zpráva a ukončí zpracování. Po té se zkontroluje, že se jedná o první výskyt daného parametru v bloku parametrů. Dále se pro každý parametr zkontroluje povolený rozsah a nastaví se hodnota proměnné určené pro tento parametr (proměnné pro hodnoty parametrů jsou už požadovaného typu pro další zpracování). Signalizace vyplněného parametru a skutečnosti že je v povoleném rozsahu se zapisuje do lokálního jednorozměrného pole kde má každý typ/název parametru jednu 8 bitovou proměnnou. Podle tohoto pole se v tomto kroku zpracování kontroluje, že nejde o opakovaný výskyt parametru se stejným názvem. Při každém zápisu parametru se také inkrementuje čítač počtu parametrů. Metoda JSONParseParam se volá, dokud nevrátí hodnotu signalizující konec bloku parametrů (při nalezení pravé složené závorky) nebo některou z chyb zpracování parametrů.

21.2.4 Kontrola parametrů a volání metod služby

Po dokončení zpracování parametrů se na základě hodnoty vzniklé kombinací id služby s id metody vyhodnotí jakou vnitřní metodu služby volat. Před voláním jednotlivých metod se zkontroluje, že jsou k dispozici všechny požadované parametry. Kontrola parametrů se provádí na základě počtu nalezených parametrů obsaženém v čítači a kontrolou hodnot pole vyplněnosti parametrů. Tento postup zaručuje, že byly použity správné parametry pro volanou metodu. Pokud by byly použity některé jiné parametry tak by nesouhlasil počet nebo by nebyly nastaveny kontrolované hodnoty pole. Stejně důsledky mají i parametry jejichž hodnoty jsou mimo povolený rozsah (i v tomto případě kontrola hodnot pole vrací chybu). Pokud jsou parametry v pořádku, dojde k zavolání příslušné metody v opačném případě je do výstupu zapsána chyba a zpracování se ukončí.

21.3 Zpracování výstupu služeb

Pro výstup dat ve formátu JSON je použit zápis jednotlivých hodnot přímo do výstupu nebo v případě chyb zpracování vstupu přímo zápis konstantních řetězců obsahující celou odpověď. Pro potřeby výstupu není nutné nějaké další složitější zpracování. Pro zjednodušení zápisu výstupu jsou navrženy metody zapisující vždy kombinaci název a hodnota a zajišťující oddělení parametrů čárkami což je pro tento účel plně postačující. Tyto metody zjednodušují zápis parametrů do výstupu a zajišťují jejich syntaktickou správnost. Ještě jsou definovány další 2 metody pro začátek a konec zápisu bloku výstupních parametrů. Metoda pro začátek zapíše do výstupu levou složenou závorku a vynuluje globální čítač výstupních parametrů. Globální čítač parametrů slouží ke kontrole, zda se má před parametrem zapsat do výstupu čárka (pokud je čítač větší než 0). Metoda pro ukončení zápisu bloku parametrů pouze zapíše pravou složenou závorku.

22 Zabezpečení aplikace

V návrhu aplikace nebyl kladen důraz na zabezpečení navrhovaných modulů, protože je počítáno s tím, že budou provozovány na samostatné části sítě (VLAN). Aktuální implementace HTTP serveru má implementován způsob autentizace BASIC. Stávající implementace bohužel také nepodporuje pro řadu PCT18F zabezpečenou komunikaci SSL/TSL. Autorizaci BASIC lze zapnout implementací metod HTTPNeedsAuth a HTTPCheckAuth. Obě tyto metody jsou obsaženy v hlavní části programu, která je v knihovně Main.c .

Metoda HTTPNeedsAuth na základě informace o který se jedná soubor, určí zda vyžaduje autentizaci.

A metoda HTTPCheckAuth na základě parametrů se jménem a heslem ověří uživatele.

23 Popis služeb modulu

Funkce pro ovládání modulu byly rozděleny do následujících 3 služeb:

- **Config.svc:** Služba sloužící k zjištění aktuální konfigurace modulu s nastavení nové konfigurace
- **PC.svc:** Služba sloužící k ovládání modulu pokud je přepnut v módu určeném pro ovládání PC
- **IO.svc:** Služba sloužící k ovládání modulu pokud je přepnut v módu určeném pro přímé ovládání vstupů a výstupů

23.1 Služba Config.svc

Služba slouží pro získání údajů o aktuálním stavu a nastavení modulu. Dále služba umožňuje nastavit parametry konfigurace. Poslední funkcí této služby je možnost softwarově restartovat procesor modulu. Soubor Config.svc obsahuje řetězec "`~json(1)`". Služba obsahuje následující metody:

23.1.1 Metoda Status

Metoda slouží k získání aktuálních hodnot konfiguračních parametrů.

URL: Config.svc/Status

23.1.1.1 Vstupní parametry

Tato metoda nemá žádné vstupní parametry.

23.1.1.2 Návrátové hodnoty

- **IP_IP:** hodnota typu string obsahující IP adresu ve tvaru A.B.C.D
- **IP_Mask:** hodnota typu string obsahující masku sítě TCP/IP protokolu ve tvaru A.B.C.D
- **IP_Gateway:** hodnota typu string obsahující bránu IP protokolu ve tvaru A.B.C.D
- **IP_DHCP:** číselná hodnota 0 nebo 1 indikující zda je zapnuto automatické přidělení adres modulu pomocí DHCP
- **PC_Mode:** hodnota typu string obsahující hodnoty "PC" nebo "IO" podle nastavení modulu
- **PC_StanbyMaxOff:** číselná hodnota udávající maximální počet vteřin po které kontrolka napájení PC zhasne pokud bliká ve stanby módu
- **PC_StanbyMaxOn:** číselná hodnota udávající maximální počet vteřin po které kontrolka napájení PC rozsvícená pokud bliká ve stanby módu
- **PC_PowerOffType:** hodnota typu string obsahující způsob vynuceného vypnutí PC může obsahovat hodnoty "HW" nebo "SW"

- **PC_ResetTime:** číselná hodnota udávající počet desetin vteřiny pro stisknutí PC tlačítka reset
- **PC_PowerSwTime:** číselná hodnota udávající počet desetin vteřin pro stisknutí PC tlačítka pro zapnutí nebo vypnutí
- **PC_ForceOffTime:** číselná hodnota udávající počet desetin vteřiny pro stisknutí PC tlačítka pro vypnutí o při vynuceném vypnutí
- **Version:** string obsahující aktuální verzi software
- **UpTimeSec:** číselná hodnota udávající maximální počet vteřin od posledního zapnutí nebo resetu
- **Error:** string obsahující chybovou zprávu v případě nepovedeného volání metody služby.

23.1.1.3 Příklad volání

- **Vstup:** Tato metoda nemá žádné vstupní parametry
- **Výstup:** {"IP_IP": "192.168.1.100", "IP_Mask": "255.255.255.0", "IP_Gateway": "192.168.1.1", "IP_DHCP": 0, "PC_Mode": "PC", "PC_StanbyMaxOff": 5, "PC_StanbyMaxOn": 5, "PC_PowerOffType": "SW", "PC_ResetTime": 8, "PC_PowerSwTime": 8, "PC_ForceOffTime": 100, "Version": "1.00", "UpTimeSec": 12345}

23.1.2 Metoda WriteIP

Metoda slouží k zápisu konfigurace TCP/IP protokolu do jednočipového počítače.

URL: Config.svc/WriteIP

23.1.2.1 Vstupní parametry

- **IP:** hodnota typu string obsahující statickou IP adresu ve tvaru A.B.C.D (například 192.168.1.100)
- **Mask:** hodnota typu string obsahující statickou masku sítě TCP/IP protokolu ve tvaru A.B.C.D (například 255.255.255.0) nebo celé číslo udávající počet bitů od začátku masky nastavených na 1 v rozsahu 8-32.
- **Gateway:** hodnota typu string obsahující statickou bránu IP protokolu ve tvaru A.B.C.D
- **DHCP:** číselná hodnota 0 nebo 1 indikující zda je zapnuto automatické přidělení adres modulu pomocí DHCP (pokud je hodnota 1 je DHCP zapnuto).

23.1.2.2 Návrátové hodnoty

- **Result:** obsahuje řetězec "OK" v případě úspěšného volání metody služby.
- **Error:** string obsahující chybovou zprávu v případě nepovedeného volání metody služby.

23.1.2.3 Příklad volání

- **Vstup:** {"IP": "192.168.1.100", "Mask": "255.255.255.0", "Gateway": "192.168.1", "DHCP": 0}
- **Výstup:** {"Result": "OK"}

23.1.3 Metoda WritePC

Metoda slouží k zápisu konfigurace pro práci se vstupy a výstupy.

URL: Config.svc/WritePC

23.1.3.1 Vstupní parametry

- **Mode:** hodnota typu string obsahující hodnoty "PC" nebo "IO". Tento mód se aplikuje na způsob ovládání výstupů modulu, vstupy se chovají vždy stejně bez ohledu na toto nastavení
- **StanbyMaxOff:** číselná hodnota udávající maximální počet vteřin po které kontrolka napájení PC zhasnutá pokud bliká ve stanby módu
- **StanbyMaxOn:** číselná hodnota udávající maximální počet vteřin po které kontrolka napájení PC rozsvícená pokud bliká ve stanby módu
- **PowerOffType:** hodnota typu string obsahující způsob vynuceného vypnutí PC může obsahovat hodnoty "HW" nebo "SW"
- **ResetTime:** číselná hodnota udávající počet desetin vteřiny pro stisknutí PC tlačítka reset
- **PowerSwTime:** číselná hodnota udávající počet desetin vteřin pro stisknutí PC tlačítka pro zapnutí nebo vypnutí
- **ForceOffTime:** číselná hodnota udávající počet desetin vteřiny pro stisknutí PC tlačítka pro vypnutí o při vynuceném vypnutí

23.1.3.2 Návrátové hodnoty

- **Result:** obsahuje řetězec "OK" v případě úspěšného volání metody služby.
- **Error:** string obsahující chybovou zprávu v případě nepovedeného volání metody služby.

23.1.3.3 Příklad volání

- **Vstup:** {"Mode": "PC", "StanbyMaxOff": 2, "StanbyMaxOn": 2, "PowerOffType": "SW", "ResetTime": 8, "PowerSwTime": 8, "ForceOffTime": 100}
- **Výstup:** {"Result": "OK"}

23.1.4 Metoda Reset

Metoda slouží k zaslání požadavku softwarový restart modulu. Reset se provede se zpožděním 1s aby se správně odeslala odpověď metody.

URL: Config.svc/Reset

23.1.4.1 Vstupní parametry

Tato metoda nemá žádné vstupní parametry.

23.1.4.2 Návrátové hodnoty

- **Result:** obsahuje řetězec "OK" v případě úspěšného volání metody služby.
- **Error:** string obsahující chybovou zprávu v případě nepovedeného volání metody služby.

23.1.4.3 Příklad volání

- **Vstup:** Tato metoda nemá žádné vstupní parametry
- **Výstup:** {"Result": "OK"}

23.2 Služba PC.svc

Služba slouží pro ovládání a získání údajů o aktuálním stavu připojeného PC. Soubor PC.svc obsahuje řetězec "~json(2)~". Služba obsahuje následující metody:

23.2.1 Metoda Get

Metoda slouží k získání informace o stavu připojeného PC a aktuálním módu modulu.

URL: PC.svc/Get

23.2.1.1 Vstupní parametry

Tato metoda nemá žádné vstupní parametry.

23.2.1.2 Návrátové hodnoty

- **Mode:** hodnota typu string obsahující hodnoty "PC" nebo "IO" podle nastavení modulu
- **Status:** hodnota typu string obsahující aktuální stav připojeného PC. Hodnoty mohou být: "ON", "OFF", "TurnOFF", "Stanby", "Unknown"

23.2.1.3 Příklad volání

- **Vstup:** Tato metoda nemá žádné vstupní parametry
- **Výstup:** {"Mode":"PC","Status":"ON" }

23.2.2 Metoda Reset

Metoda slouží k zaslání požadavku restart připojeného PC.

URL: PC.svc/Reset

23.2.2.1 Vstupní parametry

Tato metoda nemá žádné vstupní parametry.

23.2.2.2 Návrátové hodnoty

- **Result:** obsahuje řetězec "OK" v případě úspěšného volání metody služby.
- **Error:** string obsahující chybovou zprávu v případě nepovedeného volání metody služby.

23.2.2.3 Příklad volání

- **Vstup:** Tato metoda nemá žádné vstupní parametry
- **Výstup:** {"Result":"OK" }

23.2.3 Metoda PowerON

Metoda slouží k zaslání požadavku na zapnutí připojeného PC.

URL: PC.svc/PowerON

23.2.3.1 Vstupní parametry

Tato metoda nemá žádné vstupní parametry.

23.2.3.2 Návrátové hodnoty

- **Result:** obsahuje řetězec "OK" v případě úspěšného volání metody služby.
- **Error:** string obsahující chybovou zprávu v případě nepovedeného volání metody služby.

23.2.3.3 Příklad volání

- **Vstup:** Tato metoda nemá žádné vstupní parametry
- **Výstup:** {"Result": "OK" }

23.2.4 Metoda PowerOFF

Metoda slouží k zaslání požadavku na vypnutí připojeného PC.

URL: PC.svc/PowerOFF

23.2.4.1 Vstupní parametry

- **Force:** číselná hodnota 0 nebo 1 indikující že jde o nucené vypnutí (vynucené vypnutí je při 1)

23.2.4.2 Návrátové hodnoty

- **Result:** obsahuje řetězec "OK" v případě úspěšného volání metody služby.
- **Error:** string obsahující chybovou zprávu v případě nepovedeného volání metody služby.

23.2.4.3 Příklad volání

- **Vstup:** {"Force": 0}
- **Výstup:** {"Result": "OK" }

23.3 Služba IO.svc

Služba slouží pro ovládání a získání údajů o aktuálním stavu vstupů a výstupů modulu. Soubor IO.svc obsahuje řetězec "~json(3)~". Služba obsahuje následující metody:

23.3.1 Metoda Get

Metoda slouží k získání informace o stavu vstupů, výstupů a aktuálním módu modulu
URL: IO.svc/Get

23.3.1.1 Vstupní parametry

Tato metoda nemá žádné vstupní parametry.

23.3.1.2 Návrátové hodnoty

- **Mode:** hodnota typu string obsahující hodnoty "PC" nebo "IO" podle nastavení modulu
- **DataIn:** číselná hodnota obsahující informaci o aktuálním stavu vstupů, 1. vstup odpovídá bitu 0
- **DataOut:** číselná hodnota obsahující informaci o aktuálním stavu výstupů, 1. výstup odpovídá bitu 0

23.3.1.3 Příklad volání

- **Vstup:** Tato metoda nemá žádné vstupní parametry
- **Výstup:** {"Mode": "IO", "DataIn": 123, "DataOut": 234}

23.3.2 Metoda Set

Metoda slouží k nastavení výstupů modulu.

URL: IO.svc/Set

23.3.2.1 Vstupní parametry

- **Value:** číselná hodnota k nastavení výstupů 1. výstup odpovídá bitu 0
- **Mask:** bitová maska pro nastavení výstupů 1. výstup odpovídá bitu 0 (budou nastaveny pouze výstupy u kterých je bit masky v 1). Mask je nepovinný parametr, pokud se neuvede, dojde k nastavení všech výstupů.

23.3.2.2 Návrátové hodnoty

- **Result:** obsahuje řetězec "OK" v případě úspěšného volání metody služby.
- **Error:** string obsahující chybovou zprávu v případě nepovedeného volání metody služby.

23.3.2.3 Příklad volání

- **Vstup:** {"Value": "0b11001100", "Mask": "0b00001111"}
(binární čísla by měla být v uvozovkách, protože nejde o standardní konstrukci formátu JSON)
- **Výstup:** {"Result": "OK"}

24 Základní nastavení a funkce modulu

24.1 TCP/IP protokol

Modul implementuje pouze IP protokol verze 4, verze protokolu 6 není v současné době v TPC/IP stacku podporována. Nastavení parametrů jako IP adresa maska a brána probíhá pouze při startu modulu. Modul může používat statickou adresu na základě nastavených parametrů nebo adresu získanou pomocí protokolu DHCP. Pokud je zapnuto DHCP, modul se při startu pokusí získat IP adresu, masku podsítě a bránu pomocí DHCP protokolu z DHCP serveru. Pokud se získání adresy pomocí DHCP nezdaří, použijí se hodnoty pro statické nastavení. Modul nepotřebuje pro svojí funkci DNS. Nastavení DNS tudíž není součástí nastavení modulu a modul implementaci DNS neobsahuje.

24.2 Funkce módů IO a PC

Nastavení módu určuje, jakým způsobem se budou ovládat výstupy modulu. Chování vstupů modulu a vyhodnocování jejich stavu funguje vždy stejně a není ovlivněno nastaveným módem.

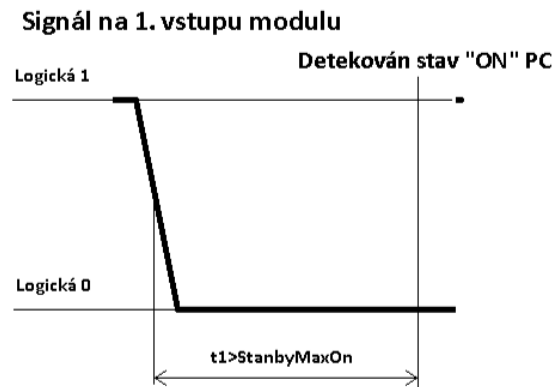
24.2.1 Funkce modulu v nastavení PC

24.2.1.1 Vyhodnocení vstupů modulu

Služba pro vyhodnocení stavu v tuto chvíli používá pouze první vstup modulu. Vstup modulu je pro vyhodnocení stavu PC programově ošetřen proti záskmitům. Detekce stavu vstupu závisí na nastavení parametrů `StanbyMaxOn` a `StanbyMaxOff` (hodnoty parametrů jsou v sekundách). Parametry `StanbyMaxOn` a `StanbyMaxOff` slouží pro detekci blikání kontrolky napájení. Příliš vysoká hodnota těchto parametrů způsobí delší dobu, za kterou je detekováno vypnutí nebo zapnutí počítače. Příliš nízké hodnoty těchto parametrů způsobí, že blikání nebude detekováno správně. Stanby mód je detekován vždy při nalezení jedné periody rozsvícení a zhasnutí kontrolky dle nastavených parametrů.

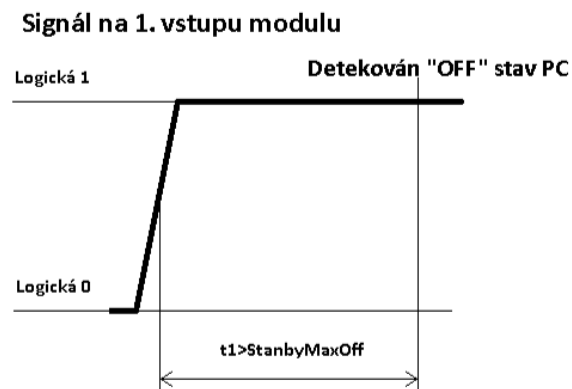
Modul rozlišuje následující stavy PC:

ON: vstup je sepnut déle, než udává hodnota StanbyMaxOn



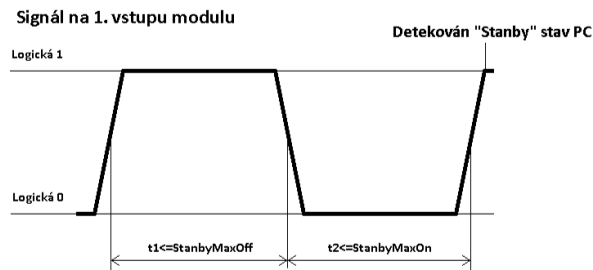
Obrázek 24.1: Diagram detekce stavu PC "ON"

OFF: vstup je rozepnut déle, než udává StanbyMaxOff

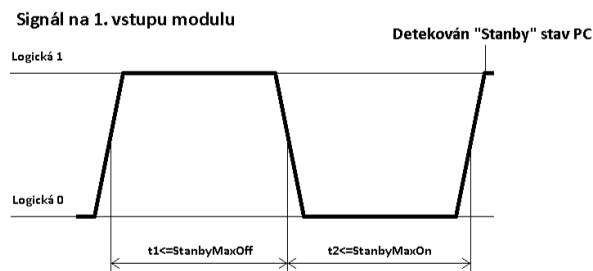


Obrázek 24.2: Diagram detekce stavu PC "OFF"

Stanby: je detekováno spínání a rozpínání vstupu (blikání kontrolky) v mezích parametrů StanbyMaxOn a StanbyMaxOff



Obrázek 24.3: 1. Diagram detekce stavu PC "Stanby"



Obrázek 24.4: 2. Diagram detekce stavu PC "Stanby"

TurnOFF: modul úspěšně zpracoval požadavek na vypnutí počítače (Force=0) ale je stále signalizováno zapnutí počítače. Stav ukončen při:

1. detekci vypnutí, stav přejde do stavu OFF
2. detekci módu stanby, stav přejde do stavu Stanby
3. uplynutí doby delší než 5 minut (tato hodnota je nastavena jako konstanta v software), stav přejde do stavu ON
4. při úspěšném zpracování požadavku na restart počítače, stav přejde do stavu ON

Unknown: Tento stav je signalizován po zapnutí modulu do doby detekce jednoho ze stavů. Maximální doba trvání tohoto stavu je $\text{StanbyMaxOn} + \text{StanbyMaxOff}$.

24.2.1.2 Ovládání výstupů PC

Ovládání používá první 2 výstupy modulu případně 7. výstup modulu.

Výstupy jsou přiřazeny následovně:

1. výstup: Tlačítko pro zapnutí/vypnutí počítače
2. výstup: Tlačítko pro reset počítače
7. výstup: Vynucené vypnutí počítače, rozpínací kontakt relé se vřadí do signálu PS_ON na pinu 16 napájecího ATX konektoru základní desky

Pro použití následujících funkcí musí být modul přepnut do PC módu a nesmí být právě vykonávána jiná funkce.

- **Funkce PC.svc/Reset:** Sepne 2. výstup na dobu $\text{ResetTime} * 0.1\text{s}$. Provede se pouze, pokud je PC ve stavu "ON" nebo "TurnOFF".
- **Funkce PC.svc/PowerON:** Sepne 1. výstup na dobu $\text{PowerSwTime} * 0.1\text{s}$. Provede se pouze, pokud je PC ve stavu "OFF" nebo "Stanby".
- **Funkce PC.svc/PowerOFF:**
 1. **Force=0:** Sepne 1. výstup na dobu $\text{PowerSwTime} * 0.1\text{s}$. Provede se pouze, pokud je PC ve stavu "ON".
 2. **Force=1**
 - (a) **Nastavení PowerOffType="SW":** Sepne 1. výstup na dobu $\text{ForceOffTime} * 0.1\text{s}$. Provede se pouze, pokud je PC ve stavu "ON", "TurnOFF" nebo "Stanby".
 - (b) **Nastavení PowerOffType="HW":** Přepne 7. výstup na dobu $\text{ForceOffTime} * 0.1\text{s}$. Provede se pouze, pokud je PC ve stavu "ON", "TurnOFF" nebo "Stanby".

24.2.2 Funkce modulu v nastavení IO

Služba Get čte vždy aktuální stav vstupů a výstupů. Přečtená hodnota vstupů se před odesláním invertuje, takže přečtená hodnota odpovídá přímo stavu vstupů před optočleny. Pokud je modul přepnut do stavu "IO" lze metodou Set přímo ovládat výstupy modulu.

25 Závěr

Tato kapitola je věnována shrnutí a zhodnocení provedené práce vzhledem k zadání a stanoveným cílům.

Základní cíle práce a jejich řešení:

- **Navrhnout hardware modulu alespoň se 2 vstupy a výstupy:** Podařilo se mi úspěšně navrhnout a otestovat modul v osazení 8 vstupů a výstupů.
- **Navrhnout software komunikující pomocí některého ze standardních protokolů nad TCP/IP:** Úspěšně jsem implementoval komunikaci pomocí rozšíření HTTP protokolu - protokolu REST s využitím formátu pro výměnu dat JSON.
- **Implementovat možnost strojového ovládání:** Strojové ovládání je zajištěné implementací služeb nad protokolem REST. Tyto služby lze velmi snadno volat z jiných aplikací.
- **Implementovat možnost manuálního ovládání:** Manuální ovládání modulu je zajištěno doplněním statických webových stránek, které umožňují snadné ovládání jednotlivých funkcí modulu.
- **Implementovat možnost ovládat připojené PC:** Ovládání připojeného PC je zajištěno službou PC.svc, která zajišťuje veškerou logiku potřebnou k tomuto ovládání.

Myslím, že se mi podařilo úspěšně splnit vytyčené cíle práce a navrhnout funkční modul s potřebným software.

Pro reálné nasazení navrženého modulu by bylo vhodné ještě zapnout funkci jednočipového počítače WDT. Jedná se o časovač, který zajišťuje že v prostředí s velkým rušením se MCU restartuje, pokud by došlo k tomu, že by procesor vlivem rušení skočil na neplatnou adresu. Pro správnou funkci by bylo nutné doplnit program o reset tohoto časovače (pravděpodobně by stačilo reset v hlavní smyčce programu, pokud by byl nastavený čas dostatečně dlouhý).

Dále jsem během ladění software modulu zjistil, že použitý LDO stabilizátor 3.3V 100mA se poměrně dosti zahřívá. Odběr modulu by neměl dle výpočtů za žádných okolností překročit 100mA ale přesto by asi bylo vhodné počít v reálně nasazených modulech stabilizátor pro větší výstupní proud.

26 Seznam literatury

- [1] Bel Fuse Inc.
 - . 10/100 MagJack® PoE 0813-1X1T-57-F PATENTED
 - , 2011
 - . <http://belfuse.com/pdfs/0813-1X1T-57-F.pdf>
 - .

- [2] CadSoft
 - . CadSoft EAGLE PCB Design Software
 - , 2011
 - . <http://www.cadsoftusa.com/>
 - .

- [3] Devesh R. Agarwal, Infomart Pvt. Ltd.
 - . PEM1200 SERIES
 - , 2010
 - . http://www.poweredethernet.com/download/PEM1200_Rev12I_R1-1.pdf
 - .

- [4] Fairchild Semiconductor
 - . H11L1M, H11L2M, H11L3M 6-Pin DIP Schmitt Trigger Output Optocoupler
 - , 2014
 - . <https://www.fairchildsemi.com/datasheets/H1/H11L1M.pdf>
 - .

- [5] jQuery Foundation
 - . jQuery
 - , 2015
 - . <https://jquery.com/>
 - .

- [6] Microchip Technology Inc.
 - . PIC18F97J60 Family Data Sheet
 - , 2011
 - . <http://ww1.microchip.com/downloads/en/DeviceDoc/39762f.pdf>
 - .

- [7] Microchip Technology Inc.
 - . Microchip Libraries for Applications
 - , 2014
 - . <http://www.microchip.com/pagehandler/en-us/devtools/mla/home.html>
 - .

- [8] Microchip Technology Inc.
 - . Microchip MCUs, Microchip Technology Inc.
 - , 2014
 - . <http://www.microchip.com/pagehandler/en-us/products/picmicrocontrollers>
 - .

- [9] Microchip Technology Inc.
. MPLAB® Integrated Development Environment Integrated Development Environment
, 2014
. <http://www.microchip.com/pagehandler/en-us/devtools/mplab/home.html>
.
- [10] Microchip Technology Inc.
. MPLAB® XC Compilers
, 2014
. <http://www.microchip.com/pagehandler/en-us/devtools/mplabxc/>
.
- [11] Morty Eisen, Marcum Technology
. Introduction to PoE and the IEEE802.3af and 802.3at Standards
, 2009
. http://www.ieee.li/pdf/viewgraphs/introduction_to_poe_802.3af_802.3-at.pdf
.
- [12] Neznámý autor - ECMA International
. Úvod do JSON
, 2015
. <http://www.json.org/json-cz.html>
.
- [13] Neznámý autor - Wikimedia Foundation Inc.
. JSON-WSP
, 2015
. <http://en.wikipedia.org/wiki/JSON-WSP>
.
- [14] Neznámý autor - Wikimedia Foundation Inc.
. Representational State Transfer
, 2015
. http://cs.wikipedia.org/wiki/Representational_State_Transfer/
.
- [15] Texas Instruments, Incorporated
. IEEE 802.3 af for Power Over Ethernet (PoE) Powered Devices (Rev. C)
, 2004
. <http://www.ti.com/lit/ds/symlink/tps2370.pdf>
.

27 Seznam použitých zkratek

MCU Microcontroller unit

REST Representational State Transfer

JSON JavaScript Object Notation

PC Personal computer

LAN Local Area Network

VLAN Virtual Local Area Network

SSL Secure Sockets Layer

TSL Transport Layer Security

DNS Domain Name System

WDT Watch Dog Timer

28 Uživatelská příručka

Modul obsahuje celkem 4 webové stránky určené pro ovládání modulu:

28.1 index.htm

Tato stránka slouží jako úvodní stránka a obsahuje menu s odkazy na další stránky. Tato stránka se zobrazí i v případě že není uveden název stránky.

28.2 Conf.htm

Tato stránka slouží k zobrazení aktuální konfigurace a k nastavení nových hodnot konfigurace.

Ethernet PC controller verze 1.00

Konfigurace	Obnovit
Ovládání PC	
Přímé ovládání výstupů	

IP adresa :	192.168.1.100	Změnit konfiguraci
Maska IP adresy :	255.255.255.0	
Brána :	192.168.1.1	
DHCP :	<input type="checkbox"/>	
Mód :	Ovládání PC ▾	Změnit konfiguraci
Maximální délka zhasnutí kontrolky ve stanby režimu (1-60) [s] :	5	
Maximální délka rozsvícení kontrolky ve stanby režimu (1-60) [s] :	5	
Typ vypnutí :	Hardwarové ▾	
Délka stisku tlačítka reset [0.1s] (5-100) :	8	
Délka stisku tlačítka pro vypnutí nebo zapnutí PC [0.1s] (5-100) :	8	
Délka stisku tlačítka pro vynucené softwarové vypnutí PC [0.1s] (5-600) :	100	
UpTime :	4 dny 07:19:25	

Restartovat modul

Stránka umožňuje změnu 2 bloků konfigurace:

28.2.1 Konfigurace TCP/IP

Ethernet PC controller verze 1.00

Konfigurace Ovládání PC Přímé ovládání výstupů	Obnovit		
	IP adresa :	192.168.1.100	Uložit
	Maska IP adresy :	255.255.255.0	
	Brána :	192.168.1.1	
	DHCP :	<input type="checkbox"/>	Storno
	Mód :	Ovládání PC	Změnit konfiguraci
	Maximální délka zhasnutí kontrolky ve stanby režimu (1-60) [s] :	5	
	Maximální délka rozsvícení kontrolky ve stanby režimu (1-60) [s] :	5	
	Typ vypnutí :	Hardwarové	
	Délka stisku tlačítka reset [0.1s] (5-100) :	8	
	Délka stisku tlačítka pro vypnutí nebo zapnutí PC [0.1s] (5-100) :	8	
	Délka stisku tlačítka pro vynucené softwarové vypnutí PC [0.1s] (5-600) :	100	
	UpTime :	4 dny 07:19:25	
	Restartovat modul		

Před uložením se zobrazí dialog pro potvrzení zápisu:

Ethernet PC controller verze 1.00

Konfigurace Ovládání PC Přímé ovládání výstupů	Obnovit		
	IP adresa :	192.168.1.100	Uložit
	Maska IP adresy :	255.255.255.0	
	Brána :	192.168.1.1	
	DHCP :	<input type="checkbox"/>	Storno
	Mód :	Ovládání PC	Změnit konfiguraci
	Maximální délka zhasnutí kontrolky ve stanby režimu (1-60) [s] :	5	
	Maximální délka rozsvícení kontrolky ve stanby režimu (1-60) [s] :	5	
	Typ vypnutí :	Hardwarové	
	Délka stisku tlačítka reset [0.1s] (5-100) :	8	
	Délka stisku tlačítka pro vypnutí nebo zapnutí PC [0.1s] (5-100) :	8	
	Délka stisku tlačítka pro vynucené softwarové vypnutí PC [0.1s] (5-600) :	100	
	UpTime :	4 dny 07:19:25	
	Restartovat modul		

Zpráva z webové stránky

?

Opravdu chcete zapsat nastavení TCP/IP?

OK Storno

Po úspěšném uložení se zobrazí zpráva o úspěšném uložení:

Ethernet PC controller verze 1.00


Konfigurace
Ovládání PC
Přímé ovládání výstupů

Obnovit

IP adresa :		Změnit konfiguraci
Maska IP adresy :		
Brána :		
DHCP :		
Mód :		
Maximální délka zhasnutí kontrolky ve stanby režimu (1-60) [s] :	5	Změnit konfiguraci
Typ vypnutí :	Hardwarově ▾	
Délka stisku tlačítka reset [0.1s] (5-100) :	8	
Délka stisku tlačítka pro vypnutí nebo zapnutí PC [0.1s] (5-100) :	8	
Délka stisku tlačítka pro vynucené softwarové vypnutí PC [0.1s] (5-600) :	100	
UpTime :	4 dny 07:25:50	

Restartovat modul

Zpráva z webové stránky

 Nastavení TCP/IP bylo úspěšně zapsáno.
Změny se projeví až po resetu modulu!

OK

28.2.2 Konfigurace parametrů pro mód PC

Ethernet PC controller verze 1.00

Konfigurace Ovládání PC Přímé ovládání výstupů		Obnovit
IP adresa :	192.168.1.100	Změnit konfiguraci
Maska IP adresy :	255.255.255.0	
Brána :	192.168.1.1	
DHCP :	<input type="checkbox"/>	
Mód :	Ovládání PC	Uložit Storno
Maximální délka zhasnutí kontrolky ve stanby režimu (1-60) [s] :	5	
Maximální délka rozsvícení kontrolky ve stanby režimu (1-60) [s] :	5	
Typ vypnutí :	Hardwarové	
Délka stisku tlačítka reset [0.1s] (5-100) :	8	
Délka stisku tlačítka pro vypnutí nebo zapnutí PC [0.1s] (5-100) :	8	
Délka stisku tlačítka pro vynucené softwarové vypnutí PC [0.1s] (5-600) :	100	
UpTime :	4 dny 07:22:01	Restartovat modul

Před uložením se zobrazí dialog pro potvrzení zápisu:

Ethernet PC controller verze 1.00

Konfigurace Ovládání PC Přímé ovládání výstupů		Obnovit
IP adresa :	192.168.1.100	Změnit konfiguraci
Maska IP adresy :	255.255.255.0	
Brána :	192.168.1.1	
DHCP :	<input type="checkbox"/>	
Mód :	Ovládání PC	Uložit Storno
Maximální délka zhasnutí kontrolky ve stanby režimu (1-60) [s] :	5	
Maximální délka rozsvícení kontrolky ve stanby režimu (1-60) [s] :	5	
Typ vypnutí :	Hardwarové	
Délka stisku tlačítka reset [0.1s] (5-100) :	8	
Délka stisku tlačítka pro vypnutí nebo zapnutí PC [0.1s] (5-100) :	8	
Délka stisku tlačítka pro vynucené softwarové vypnutí PC [0.1s] (5-600) :	100	
UpTime :	4 dny 07:22:01	Restartovat modul

Zpráva z webové stránky

Opravdu chcete zapsat nastavení ovládání počítače?

Po úspěšném uložení se zobrazí zpráva o úspěšném uložení:

Ethernet PC controller verze 1.00

Konfigurace
Ovládání PC
Přímé ovládání výstupů

Obnovit


IP adresa :	
Maska IP adresy :	
Brána :	
DHCP :	
Mód :	
Maximální délka zhasnutí kontrolky v režimu :	
Maximální délka rozsvícení kontrolky ve stanby režimu (1-60) [s] :	5
Typ vypnutí :	Hardwarové ▾
Délka stisku tlačítka reset [0.1s] (5-100) :	8
Délka stisku tlačítka pro vypnutí nebo zapnutí PC [0.1s] (5-100) :	8
Délka stisku tlačítka pro vynucené softwarové vypnutí PC [0.1s] (5-600) :	100
UpTime :	4 dny 07:25:07

Změnit konfiguraci

Změnit konfiguraci

Restartovat modul

Zpráva z webové stránky

 Nastavení ovládání počítače bylo úspěšně zapsáno.

OK

28.3 PC.htm

Tato stránka slouží k zobrazení aktuálního stavu připojeného PC a umožňuje zasílat příkazy na ovládání tohoto PC. Stav se automaticky načítá s periodou 5s.

Ethernet PC controller verze 1.00

Konfigurace	Mód :	Ovládání PC
Ovládání PC	Stav počítače :	Zapnut
Přímé ovládání výstupů		

Restartovat PC Zapnout PC Vypnout PC Vynucené vypnutí PC

Tlačítka pro ovládání jsou aktivní, pouze pokud je modul v módu PC. Každý příkaz zobrazí dialog pro potvrzení a zprávu o úspěšném provedení stejně jako stránka pro konfiguraci.

28.4 IO.htm

Tato stránka slouží k zobrazení aktuálního stavu připojeného PC a umožňuje zasílat příkazy na ovládání tohoto PC. Stav se automaticky načítá s periodou 2s.

Ethernet PC controller verze 1.00

Konfigurace	Obnovit
Ovládání PC	
Přímé ovládání výstupů	
Mód :	Přímé ovládání
Vstupy :	<input type="checkbox"/> IN0 <input checked="" type="checkbox"/> IN1 <input checked="" type="checkbox"/> IN2 <input type="checkbox"/> IN3 <input type="checkbox"/> IN4 <input checked="" type="checkbox"/> IN5 <input type="checkbox"/> IN6 <input checked="" type="checkbox"/> IN7
	<input type="checkbox"/> OUT0 <input checked="" type="checkbox"/> OUT1 <input type="checkbox"/> OUT2 <input checked="" type="checkbox"/> OUT3 <input type="checkbox"/> OUT4 <input checked="" type="checkbox"/> OUT5 <input checked="" type="checkbox"/> OUT6 <input type="checkbox"/> OUT7
Vstupy :	<input type="checkbox"/> Zapsat změny automaticky <input type="button" value="Zapsat"/>

Tlačítko a checkboxy pro ovládání výstupů jsou aktivní, pouze pokud je modul v módu IO. Nastavení výstupů se nijak nepotvrzuje. Tlačítko obnovit načte kromě vstupů i aktuální stavy výstupů (pokud je modul v PC módu stavy výstupů se obnovují automaticky).

29 Obsah příloženého CD

Datasheets : Adresář obsahující katalogové listy použitých součástek

DPS.zip :Archiv obsahující schéma a podklady pro výrobu tištěného spoje ve formátu CadSoft Eagle.

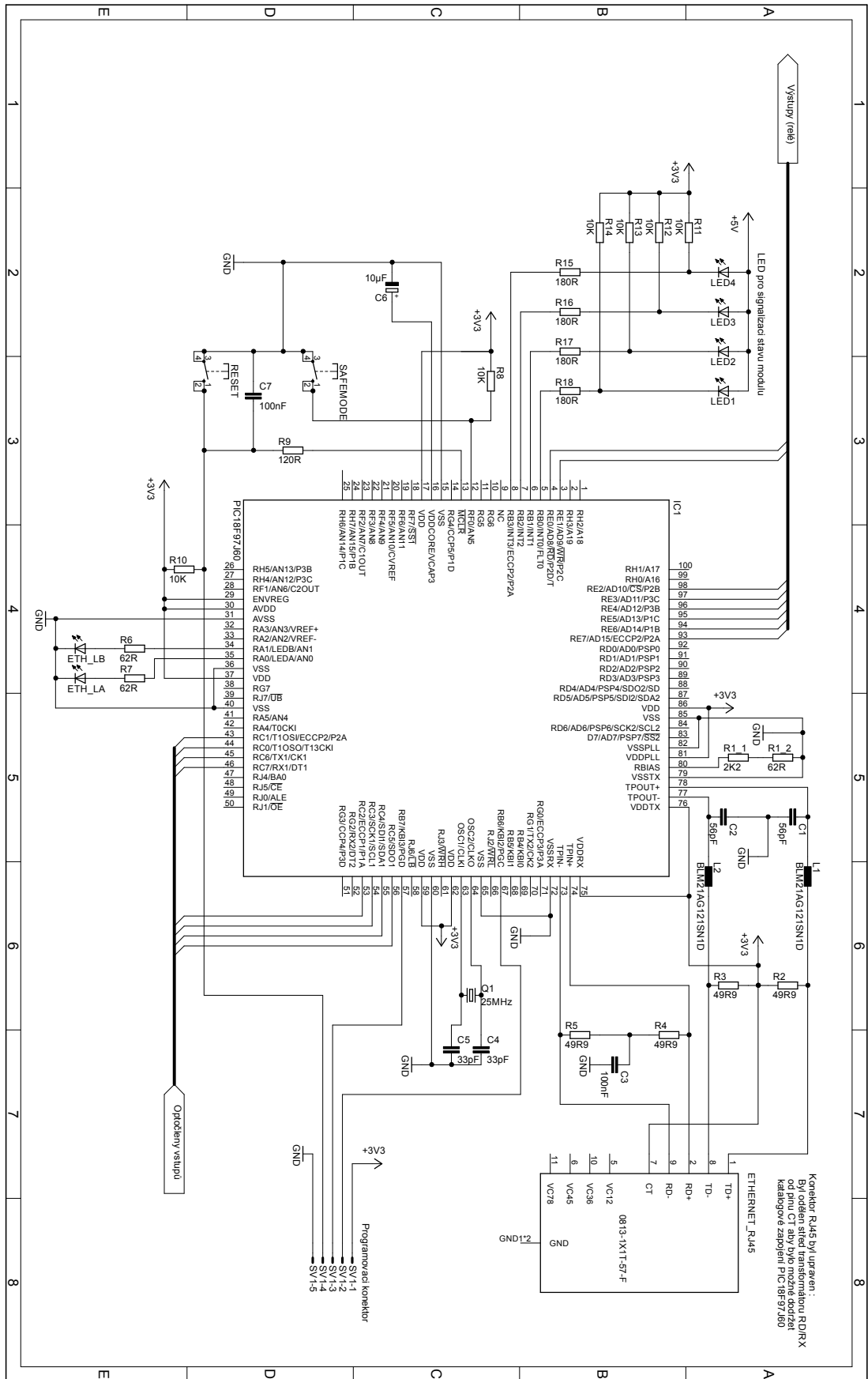
SeznamSoucastek.xls :Soupiska součástek ve formátu Microsoft Excel

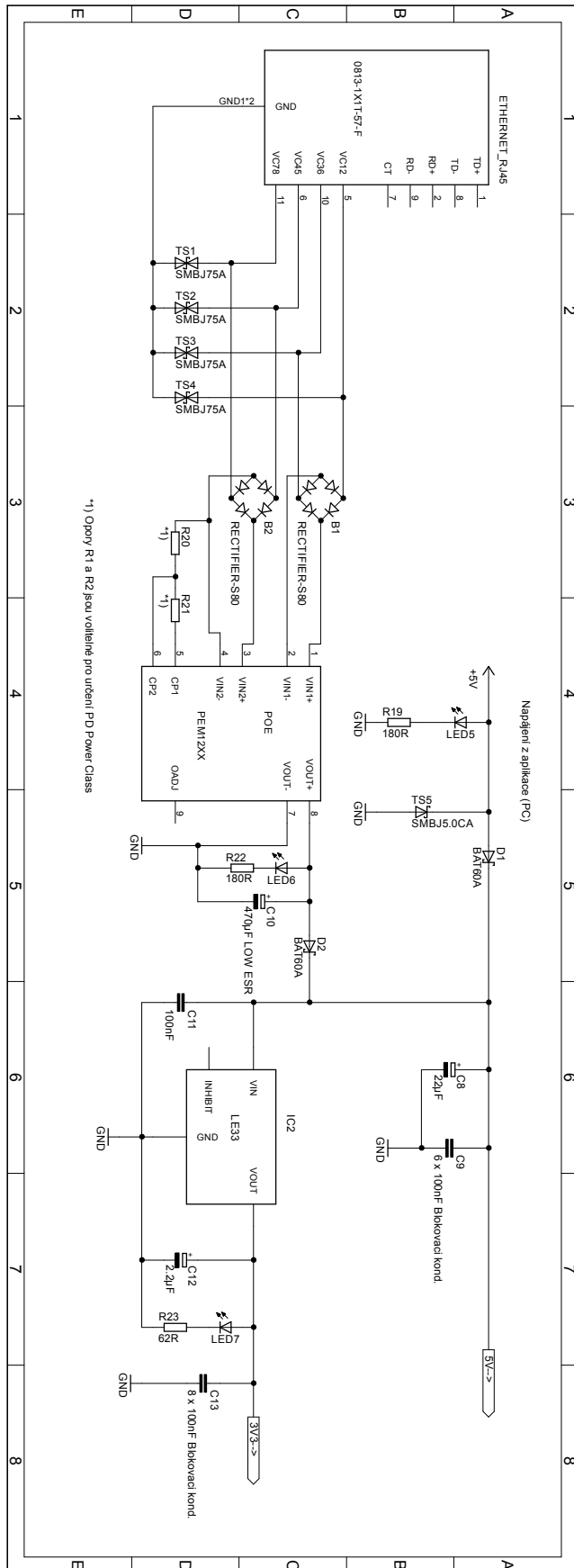
DiplomovaPrace.X.zip :Projekt MPLAB X IDE v2.35 obsahující software modulu.

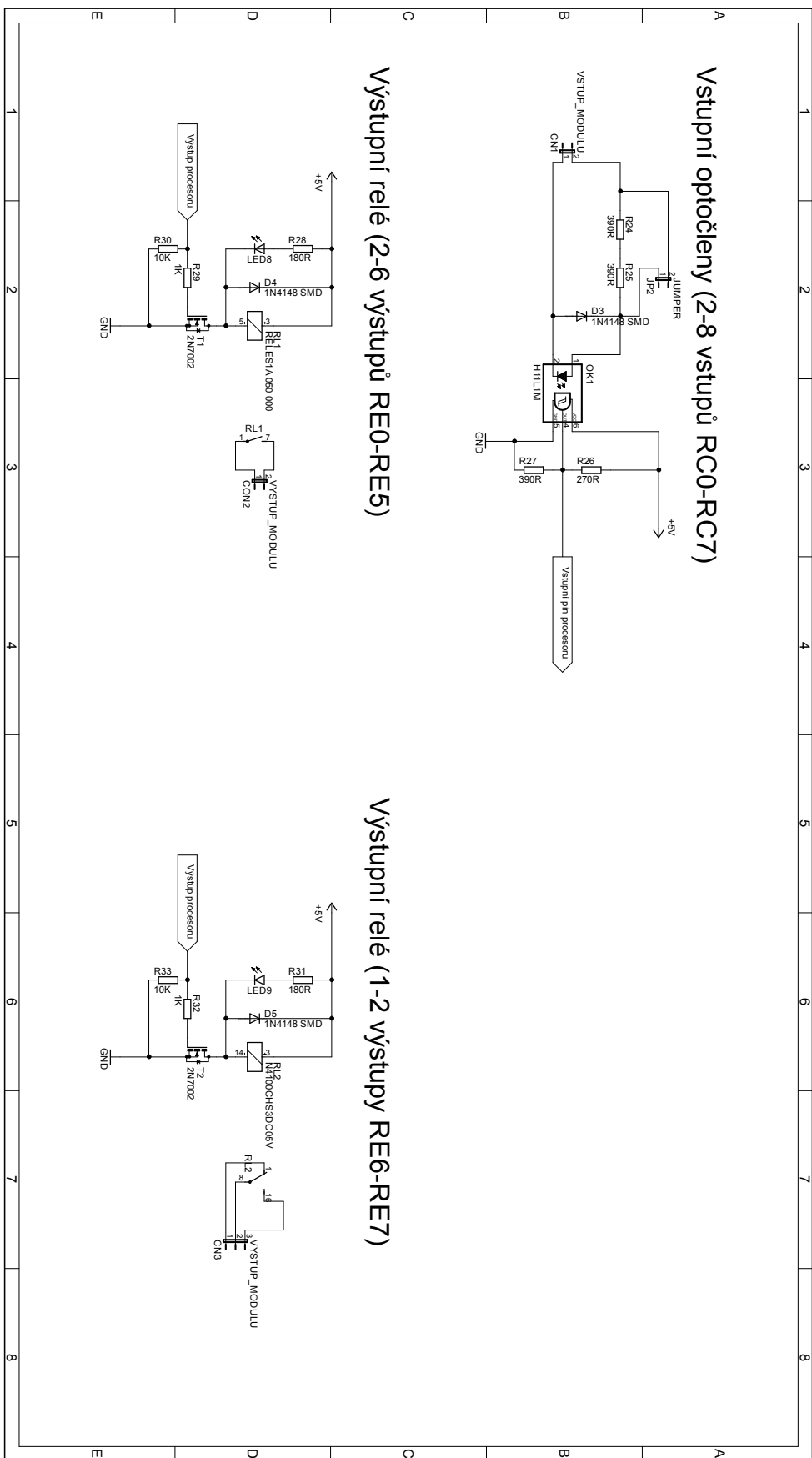
DP_JSON.zip :Pomocný projekt ve Visual Studiu 2013 sloužící pro návrh rozhraní služeb a ladění webových stránek (také obsahuje pomocnou stránku pro generování definice MPFS systému)

ETH_PC_CON.pdf :Elektronická kopie zprávy.

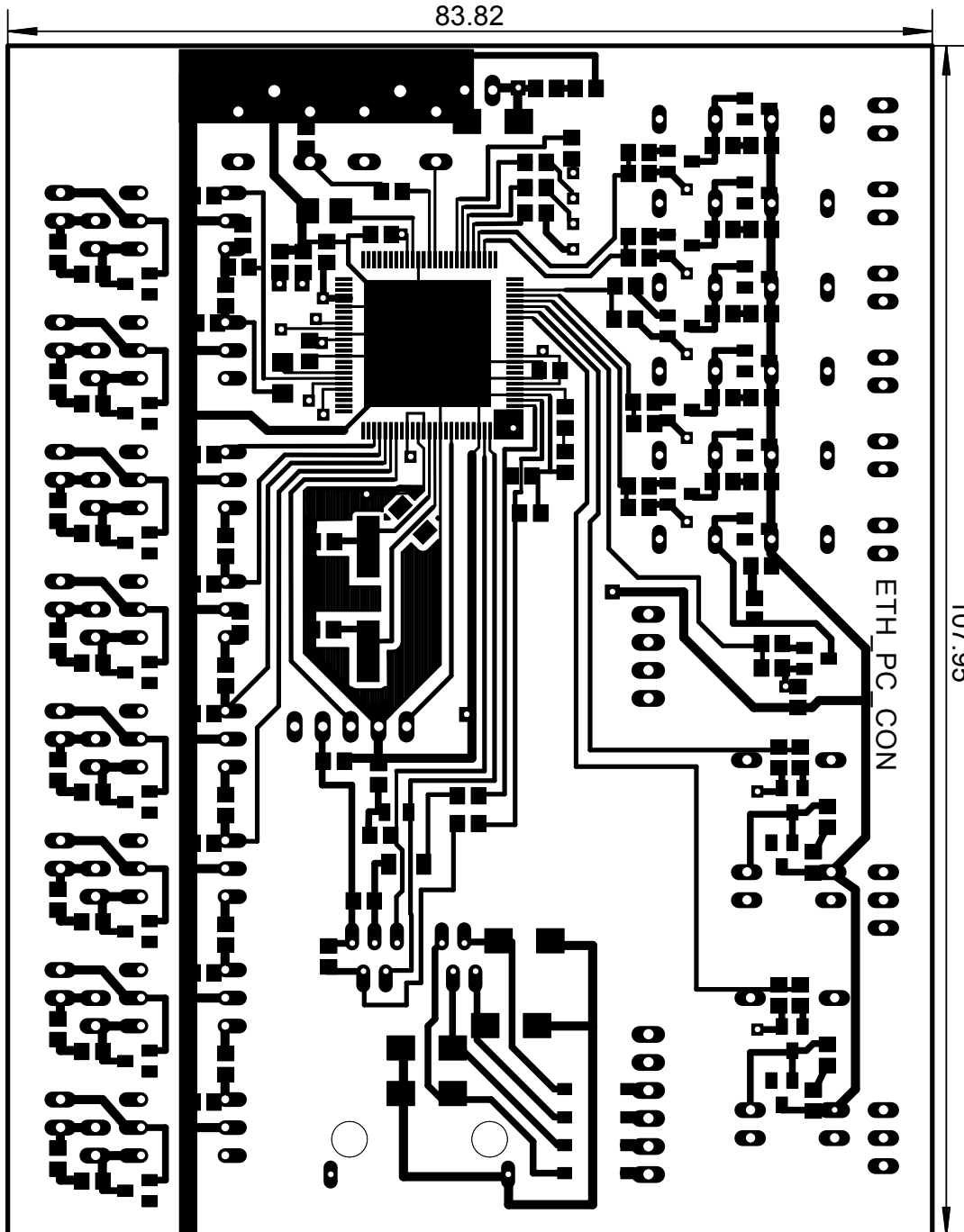
30 Schéma zapojení

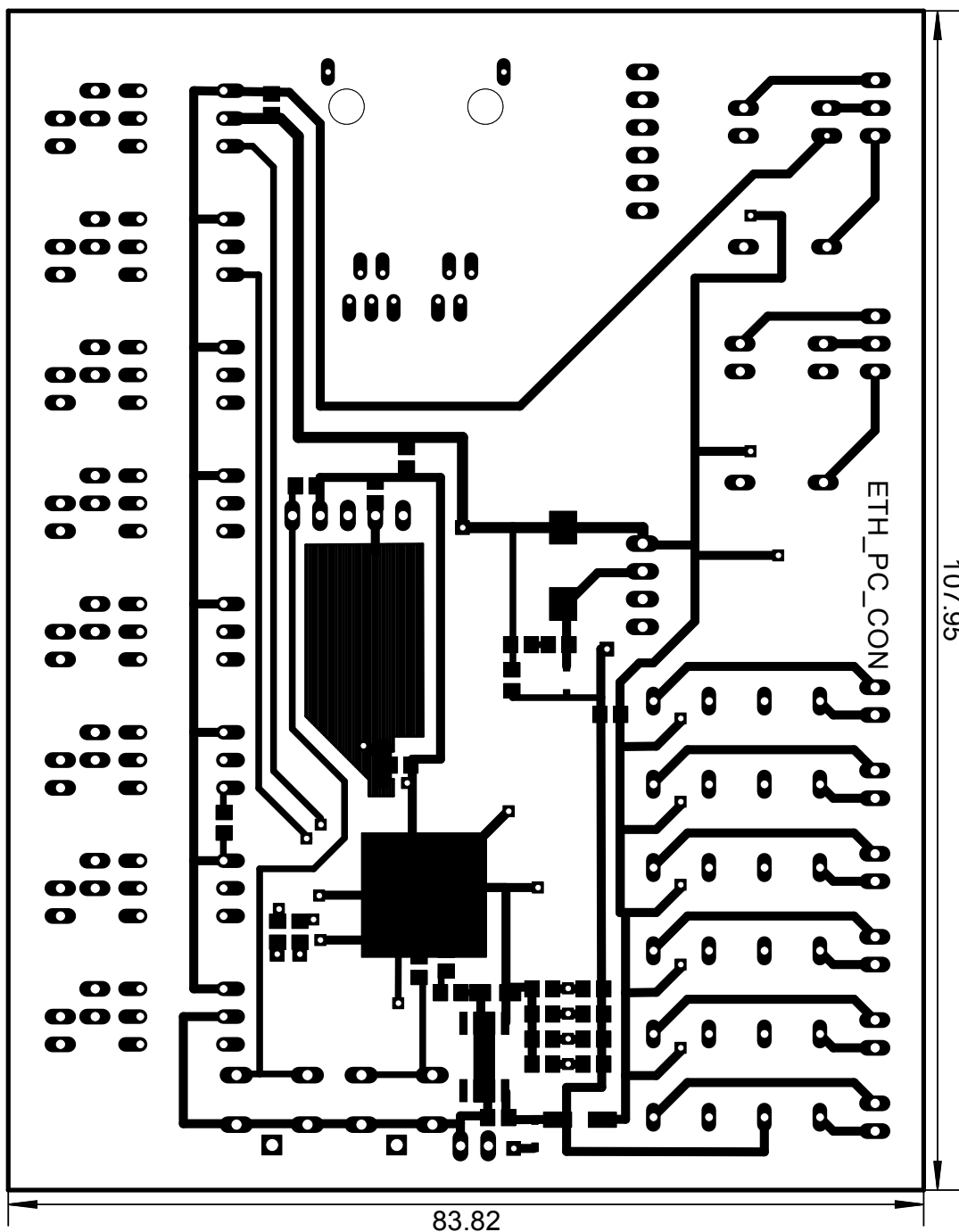






31 Návrh tištěného spoje





32 Osazovací schéma

