VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

BAKALÁŘSKÁ PRÁCE

Brno, 2021 Vojtěch Lukáš

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ
DEPARTMENT OF TELECOMMUNICATIONS

MIDI PO ETHERNETU
MIDI OVER ETHERNET

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

Vojtěch Lukáš

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. Ondřej Krajsa, Ph.D.

BRNO 2021

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní program **Audio inženýrství**
specializace Zvuková produkce a nahrávání
Ústav telekomunikací

Student: Vojtěch Lukáš ***ID:*** 211572
Ročník: 3 ***Akademický rok:*** 2020/21

NÁZEV TÉMATU:

MIDI po Ethernetu

POKYNY PRO VYPRACOVÁNÍ:

Navrhněte a realizujte přípravek pro přenos protokolu MIDI po Ethernetu. Přípravek bude mít dva vstupní porty a dva výstupní porty s možností přepnutí do režimu Thru. Data přenášená po Ethernetu pak budou v dalším přípravku odeslána na patřičné MIDI porty.

DOPORUČENÁ LITERATURA:

[1] LINSLEY HOOD, John. Audio Electronics. Kent: Elsevier Science, 1995. ISBN 9780750621816

[2] GUÉRIN, Robert. Velká kniha MIDI: standardy, hardware, software. Brno: Computer Press, 2004, 340 s. ISBN 80-7226-985-2

Termín zadání: 1.2.2021 ***Termín odevzdání:*** 31.5.2021

Vedoucí práce: Ing. Ondřej Krajsa, Ph.D.

doc. Ing. Jiří Schimmel, Ph.D.
předseda rady studijního programu

ABSTRAKT

V této práci je navržen přípravek sloužící k adaptaci protokolu MIDI pro použití v rámci počítačové sítě. Práce dále popisuje síťový ovladač, kterým lze tyto přípravky ovládat.

KLÍČOVÁ SLOVA

MIDI, Ethernet, AVR, C/C++, Python

ABSTRACT

This paper is about designing and building a device to adapt MIDI protocol for use on a standard computer network. The paper also includes a brief description of a network editor, which controls the whole system.

KEYWORDS

MIDI. Ethernet, AVR, C/C++, Python

LUKÁŠ, Vojtěch. *MIDI po Ethernetu*. Brno, 2021, 49 s. Bakalářská práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací. Vedoucí práce: Ing. Ondřej Krajsa, Ph.D.

PROHLÁŠENÍ

Prohlašuji, že svou bakalářskou práci na téma „MIDI po Ethernetu“ jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této bakalářské práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

podpis autora

PODĚKOVÁNÍ

Děkuji vedoucímu práce Ing. Ondřeji Krajsovi, Ph.D. za jeho rady, poskytnuté zázemí i trpělivost. Dík patří také mé rodině, přátelům i kolegům z divadla.

Obsah

Seznam obrázků

Úvod

Protokol MIDI[1] je již dlouhá léta zavedeným standardem nejen pro komunikaci mezi elektronickými hudebními nástroji, ale také pro řízení studiové nebo scénické techniky, časovou synchronizaci dvou a více zařízení a podobně. Cílem této práce je vyvinout hardwarový přípravek s vlastním softwarem, který adaptuje tento protokol pro použití v rámci počítačové sítě. Součástí projektu je taktéž síťový ovladač, s jehož pomocí lze nastavit jednotlivá zařízení i celkové směřování příkazů v rámci sítě.

První kapitola je ve stručnosti věnována MIDI protokolu jako takovému. Ve druhé kapitole je zmíněno několik elementárních pojmů z oblasti počítačových sítí. Třetí kapitola popisuje přípravek, jak z hardwarového, tak softwarového hlediska. Obsahem čtvrté kapitoly je popis síťového ovladače, jsou zmíněny dostupné příkazy a makra. Poslední kapitola navazuje na všechny předchozí a konkrétně vysvětluje chování přípravku za běhu programu.

[1] Musical Instrument Digital Interface – digitální rozhraní hudebního nástroje

1 Protokol MIDI

Tento protokol umožňuje přenos zejména hudebních informací mezi dvěma (i více) elektronickými hudebními nástroji, sekvencery, počítači a dalšími přístroji. Původně byl zamýšlen pro použití „v reálném čase“, tedy při živé produkci [7]. S nástupem moderních DAW[1] ale přišla možnost povely také nahrávat, upravovat a znovu přehrávat.

MIDI se však netýká výhradně hudby a hudebních dat. Umožňuje též přenos kontrolních povelů a synchronizačních značek. Přirozeně se tedy rozšířilo i do nahrávacích studií, divadel a dalších zařízení, kde umožňuje globální řízení jednotlivých přehrávačů nebo vzdálené ovládání řídících konzolí [4].

1.1 Hardwarová vrstva

Pro přenos MIDI dat se tradičně používá zásuvka a vidlice `DIN 5`[2]. Kabel mezi dvěma zařízeními by neměl být delší než 15 m a tvořit by jej měla stíněná kroucená dvojlinka. Toto stínění by mělo být připojeno k `pinu 2` na obou koncích.

Výměna informací je realizována pomocí 5mA proudové smyčky. Při protékání proudu je zaznamenána `logická 0`, v opačném případě `logická 1` [7].

Přesná specifika obvodů pro zpracování příchozích a odchozích MIDI signálů upravuje kapitola *Hardware* z dokumentu [7]. Tato pasáž byla v roce 2014 aktualizována normou [8], která adaptovala protokol i pro zařízení s 3,3V logikou a přidává další prvky pro zamezení zejména RF[3] interferencí. V následujících schématech bude však zobrazeno originální schéma z původní normy.

1.1.1 MIDI výstup

Výstupní port MIDI sběrnice je ve své podstatě jednoduchý. Z UART[4] čipu jsou přes napěťové sledovače nebo tranzistory vedeny logické impulzy na `pin 5`, zatímco na `pin 4` je přivedeno stálé napětí. `Pin 2` je v tomto případě uzemněn. [7]

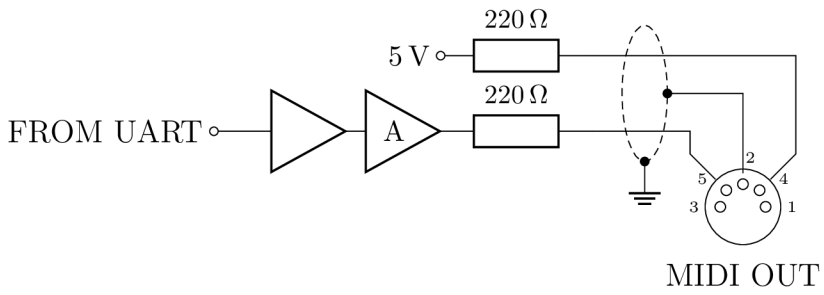
Podle aktualizační normy [8] je možné přidat za každý rezistor feritové jádro pro zamezení vysokofrekvenčních interferencí. V případě použití zařízení s 3,3V logikou jsou pak použity rezistory s menšími hodnotami odporů. Na obr. 1.1 je schéma k nahlédnutí.

[1] Digital Audio Workstation – digitální pracovní stanice pro náběr a úpravu vícestopého záznamu

[2] Dnes je však častější využití USB sběrnice nebo technologie Bluetooth pro obousměrný přenos.

[3] Radio Frequency

[4] Universal Asynchronous Receiver/Transmitter – univerzální asynchronní přijímač/vysílač



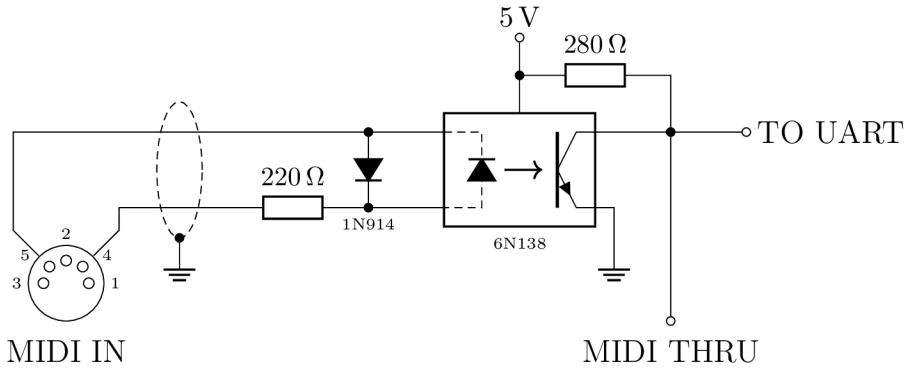
Obr. 1.1: Schéma MIDI výstupu [7]

1.1.2 MIDI vstup

Vstupní port MIDI sběrnice je mnohem složitější. Na obr. 1.2 je patrné, že s cílem eliminovat zemní smyčky, které jsou ve zvukové technice nepřípustné, je vstup každého MIDI zařízení skrz optočlen galvanicky oddělen. Z téhož důvodu je také `pin 2` – na rozdíl od výstupního portu – zapojen naprázdno (podle [7]).

Některá zařízení disponují také výstupem MIDI THRU. Ten „kopíruje“ data ze vstupu MIDI IN a tak umožňuje řetězení zařízení za sebou. Jeho schéma je totožné s tím na obr. 1.1.

Aktualizační norma [8] pak umožňuje přidání feritových jader za `piny 4` a `5`, přes kondenzátor nízké kapacity uzemnění `pinu 2` a při použití 3,3V logiky snížení odporu rezistoru na vstupu optočlenu.



Obr. 1.2: Schéma MIDI vstupu [7]

Bude-li do tohoto vstupu připojen výstup jiného zařízení, které vyšle `logickou 1`, objeví se na `pinech 4` a `5` stejné napětí, v obvodu tedy neprochází žádný proud – LED[5] v optočlenu nesvítí. Vstupní UART čip přijímá `logickou 1`. Vyšle-li jiné zařízení `logickou 0`, na `pinu 5` poklesne napětí oproti `pinu 4`, LED v optočlenu se rozsvítí, obvodem prochází proud a UART čip přijímá `logickou 1`.

[5]Light-Emiting Diode – dioda, která vyzařuje viditelné světlo

1.2 Softwarová vrstva

MIDI protokol přenáší informace (příkazy) po sériové lince přenosovou rychlostí 31 250 bps. Komunikace je jednosměrná, pro obousměrnou komunikaci je třeba využít dvou rozhraní (vstupu a výstupu) na všech zúčastněných zařízeních.

Příkazy se skládají z následujících dvou typů bajtů:

- `STATUS BAJT` v sobě kóduje druh příkazu (*Nota stlačena, Změna kontroléru . . .*) a cílový kanál (1–16).
- `DATA BAJT` vyjadřuje hodnotu s jakou je příkaz posílán (0–127).

STATUS BAJT							
1	a	a	a	b	b	b	b

(a) Bity `STATUS BAJTU`

DATA BAJT							
0	c	c	c	c	c	c	c

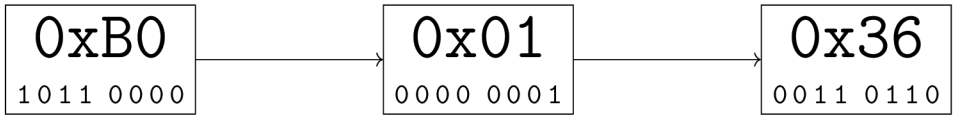
(b) Bity `DATA BAJTU`

a	bity pro kódování druhu zprávy
b	bity pro kódování kanálu
c	bity pro kódování hodnoty

Obr. 1.3: Bity obou druhů MIDI bajtů [7]

Na obrázku 1.3 je patrná struktura obou bajtů. Za pozornost stojí především jejich MSb[6]: `STATUS BAJT` má MSb vždy s hodnotou 1. Naproti tomu MSb `DATA BAJTU` je vždy nulový [4].

Pomineme-li výjimky, každý příkaz začíná jedním `STATUS BAJTEM`, za nímž následuje jeden, nebo dva `DATA BAJTY`. Na obr. 1.4 je uvedena struktura ukázkové zprávy.



Obr. 1.4: Struktura zprávy *Změna kontroléru* [7]

Z prvního bajtu zprávy z obr. 1.4 lze dekódovat druh a cílový kanál příkazu. Jedná se o *Control Change – Změna kontroléru* na *kanálu 1*[7] Druhý bajt představuje číslo kontroléru (2 – *Modulation*) a třetí jeho hodnotu (54).

Pro potřeby tohoto bakalářského projektu bude nutné, aby každý přípravek dokázal rozeznat kanál k němu příchozí MIDI zprávy, popř. jej pro další přenos pozměnil. `DATA BAJTY` budou jen „kopírovány“ a nebude do nich nijak zasahováno.

[6] Most Significant bit – nejvýznamnější bit (většinou v bajtu)

[7] $0000_2 \sim$ kanál 1, $0001_2 \sim$ kanál 2 . . . $1111_2 \sim$ kanál 16.

1.2.1 Výjimky

Je třeba alespoň okrajově zmínit výjimečné stavy, které v rámci MIDI protokolu mohou nastat.

Running Status

Pracuje-li MIDI vysílač v tomto stavu, neposílá `STATUS BAJT` v opakující se zprávě stejného druhu. Přijímač si tedy musí uložit poslední platný `STATUS BAJT` do paměti, kterou přemaže, obdrží-li zprávu s jiným, novým `STATUS BAJTEM`. Tento mód výrazně šetří přenesená data, zejména je-li použit při odesílání zprávy *Změna kontroléru* kontinuálních ovladačů [7].

Zprávy Real-Time

Tyto zprávy slouží pro synchronizaci MIDI zařízení, které pracují s časovou osou nebo časem obecně. Skládají se pouze z jediného bajtu a mají nejvyšší prioritu. Přijímač by měl být připraven i na to, že je obdrží uprostřed standardní zprávy nebo SysEx zprávy [7].

Zprávy SysEx

SysEx (System Exlusive) zprávy se skládají z většího a libovolného počtu bajtů. Jsou víceúčelové a univerzální, používají se například pro posun na časové ose (komplementárně ke zprávám Real-Time), MSC[8] a podobně [7].

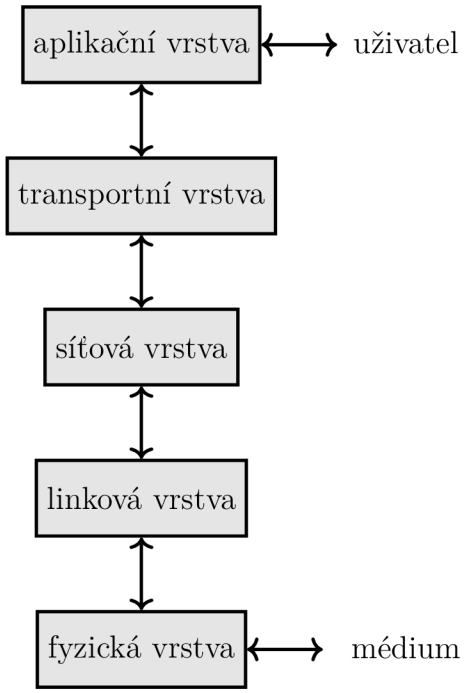
[8]MIDI Show Control – subprotokol pro ovládání scénické techniky, zejména pomocí příkazů GO, STOP, popř. RESUME. [7]

2 Teorie počítačových sítí

V rámci teoretické sekce této práce bude vhodné ve stručnosti zmínit elementární pojmy z teorie počítačových sítí.

2.1 TCP/IP model

Komunikace prostřednictvím technologie Ethernet probíhá „po vrstvách“. K vysvětlení je vhodné použít schéma ISO/OSI, resp. jeho jednodušší variantu – schéma komunikace protokolu TCP/IP na obr. 2.1. Vysílaná informace prochází od uživa-



Obr. 2.1: Komunikační model TCP/IP [10]

tele směrem dolů – při každém průchodu vrstvou je obohacena o její hlavičku. Tyto hlavičky slouží ke korektnímu směřování vysílané informace v rámci sítě. Dojde-li informace k příjemci, prochází opět tímto řetězcem, tentokrát směrem od média k uživateli [10].

2.2 MAC adresa

MAC adresa se skládá z 6 bajtů a v rámci sítě musí být unikátní. Ve většině případů ji má dané zařízení napevno přidělenou výrobcem [10]. V případě přípravku vyvinutého v rámci této bakalářské práce je MAC adresa nastavitelná programátorem, nikoli však uživatelem.

2.3 IP adresa

Tato adresa se skládá ze 4 bajtú[1]. V rámci lokální sítě musí být unikátní, je nastavována staticky (uživatelem) nebo přidělena dynamicky (DHCP serverem v dané síti).

S IP adresami úzce souvisí maska sítě. Tato sekvence 32 bitů (4 bajtů) určuje množství zařízení a podobu jejich adres v rámci jedné podsítě. Zjednodušeně řečeno, má-li síť masku 255.255.255.0, je možné v síti provozovat 254 zařízení (jedna adresa je vyhrazena pro globální adresu sítě, jedna adresa je vyhrazena pro broadcast vysílání, dohromady 256 adres) [10].

2.4 DHCP server

Úkolem tohoto serveru je automaticky konfigurovat zařízení v síti. Přiděluje jim IP adresu, sděluje masku podsítě a další údaje. Přidělení těchto údajů však má jistou dobu trvání a po uplynutí této doby může být stejnému zařízení přidělena IP adresa úplně nová, to představuje pro přípravek potenciální nebezpečí, více v kapitole 3.2.2 [10].

2.5 TCP & UDP

Tyto protokoly se vztahují k transportní vrstvě komunikačního modelu na obr. 2.1. Každý z nich představuje opačný přístup k danému problému multiplexování odchozích a příchozích dat od/k jednotlivým uživatelským aplikacím. TCP je bráno jako spolehlivý, ale pomalejší způsob. Pakety jsou potvrzovány a jejich pořadí je kontrolováno. Naproti tomu UDP představuje rychlejší, ale méně spolehlivý způsob dopravy. Pakety jsou vyslány a žádná služba již jejich doručení nekontroluje. Umožňují však variabilnější směřování i na broadcast adresu sítě – této vlastnosti je v rámci tohoto bakalářského projektu hojně využíváno [10].

2.6 Unicast & broadcast

Slovem „unicast“ popisujeme ty zprávy, které směřují od jednoho odesílatele jednomu příjemci. „Broadcast“ naopak označuje zprávy, které míří od jednoho odesílatele *všem* účastníkům sítě. Tohoto se docílí tak, že zařízení vyšle zprávu na broadcast adresu – adresu s nejvyšší danou hodnotou podle síťové masky [10].

[1] V době psaní této práce je protokol IPv4 stále zavedeným standardem

3 Návrh přípravku

Výstupem této bakalářské práce je autorsky navržený přípravek s vlastním softwarem. V souladu se zadáním zařízení disponuje dvěma MIDI vstupy, dvěma MIDI výstupy, napájecími konektory a přirozeně také konektorem RJ–45 pro přípojení do sítě Ethernet. Přípravek přijímá nejpoužívanější MIDI zprávy, zpracovává je a následně odesílá jinému přípravku nebo přípravkům, které se nachází ve stejné síti. Ty tuto přijatou zprávu odešlou na svůj MIDI výstup. Zařízení využívá všechny vrstvy TCP/IP modelu (obr. 2.1) a zmíněnou komunikaci realizuje prostřednictvím autorsky navrženého protokolu „MoE[1]“. Pro pohodlné ovládání MoE sítě byla dále vytvořena jednoduchá aplikace, která se spouští na počítači v této síti připojeném. Umožňuje maticové propojování napříč MoE přípravky a jejich kanály spolu s individuálním nastavením. Více v kapitole 4

Prototypování a první testy probíhaly na platformě Arduino UNO, která disponuje čipem ATmega 328P. Arduinu sekundoval Ethernet Shield V1, který je osazen čipem WizNet W5100. Okolní obvody byly pak zapojeny na nepájivém poli. Software byl vyvíjen v Arduino IDE[2] s využitím několika knihoven.

Finální zařízení je touto architekturou inspirováno, přímo z ní vychází – opět používá mikrokontrolér z rodiny ATmega, tentokrát ATmega3208. Síťovou komunikaci obstarává tentýž WizNet W5100. Obě tyto komponenty sedí na vlastní desce plošných spojů.

3.1 Hardware

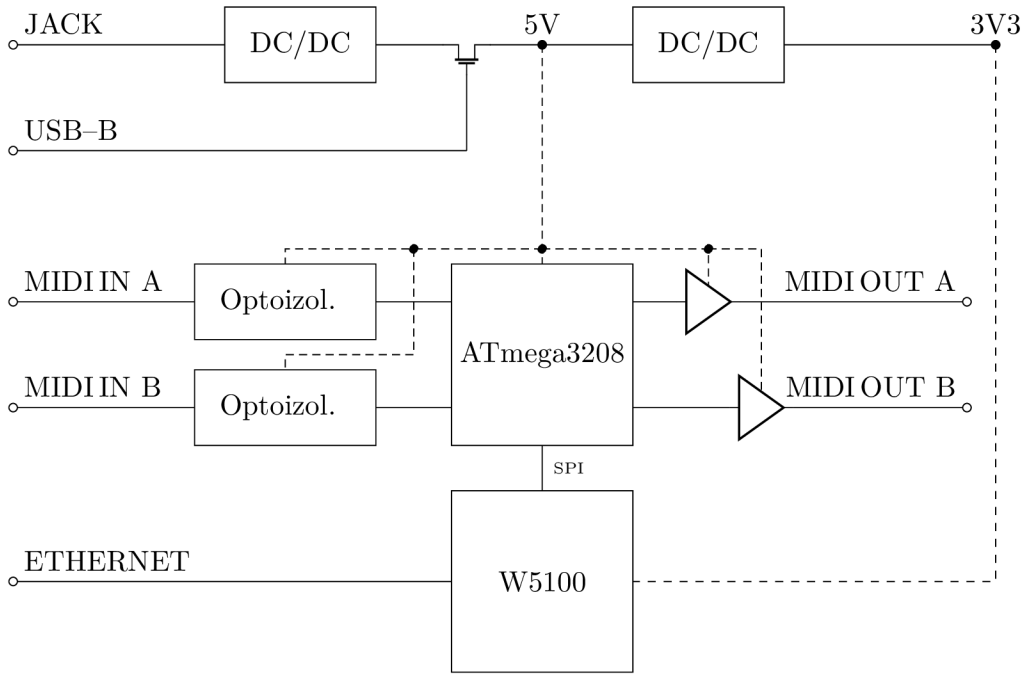
Vnitřní strukturu lze rozdělit na obvody napájecí, vstupní a výstupní, logické, programovací a obvody zprostředkovávající uživatelské rozhraní. Názorně to je patrné na blokovém schématu v obr. 3.1. Na následujících řádkách bude postupně každá sekce popsána.

3.1.1 Napájecí obvody

Při návrhu zařízení určeného pro generování nebo úpravu zvukového signálu je nutné věnovat speciální pozornost právě napájecím obvodům. Ty mohou totiž signál negativně ovlivnit, zkreslit či jinak znehodnotit. I když lze přípravek, který je předmětem této práce, zařadit do skupiny „audiozařízení“, sám o sobě žádný zvukový signál negeneruje ani neupravuje, na napájecí sekci tedy není kladen takový zřetel, jako by tomu bylo např. při návrhu zdroje pro syntezátor nebo zesilovač. Předpokládá se,

[1] MIDI over Ethernet – MIDI po Ethernetu

[2] Integrated Development Enviroment – integrované vývojové prostředí



Obr. 3.1: Blokové schéma přípravku MoE

že přípravek bude s takovými zařízeními propojován – ani v tomto případě však zkreslení signálu nehrozí díky implementaci optočlenu dle MIDI normy [7] na straně přijímače.

Návrh této sekce se opírá o soupisku doporučených napájecích napětí všech komponent na desce. Téměř všechny součástky přijímají napájecí napětí 5 V. Výjimku tvoří ethernetové rozhraní, tedy čip W5100 a s ním související konektor RJ–45 s LED. Z tohoto důvodu bylo nutné navrhnout dvě napájecí větve: 5 a 3,3 V.

V prvních fázích návrhu byly obě větve regulovány jednoduchými lineárními stabilizátory napětí 7805, respektive 7803. Z důvodu velké napěťové ztráty a nízké účinnosti těchto regulátorů byl však návrh přepracován tak, aby využíval DC/DC měniče (buck-boost konvertory), jenž pracují na principu PWM[3], tedy modulace střídy výstupního napětí. Jmenovitě se jedná o spínané regulátory MC34063 s doplňkovými obvody. Tyto integrované obvody pracují se vstupním napětím od 3 do 40 V (toto napětí dokáží na výstupu snížit – *step-down mode* – i zvýšit – *step-up mode*), vnitřní oscilátor je taktovaný na frekvenci 33 kHz a maximální spínaný proud, který dovedou dodat je 1,5 A. [12].

Při návrhu bylo respektováno výrobcem doporučené schéma zapojení v režimu *step-down*. Byly využity dva totožné integrované obvody: první pro regulaci vstupního napětí z externího zdroje na 5 V; druhý pro snížení tohoto napětí na 3,3 V. Nastavení kýženého výstupního napětí bylo docíleno výběrem hodnot odporů refe-

[3]Pulse Width Modulation – šířková modulace obdélníkového signálu

renčního napěťového děliče přidruženého danému regulátoru podle vztahu [12]

$$U_{\text{out}} = 1{,}25 \cdot \left(1 + \frac{R_2}{R_1}\right). \tag{3.1}$$

Úpravou tohoto vztahu dostaneme vzorec pro poměr $\frac{R_2}{R_1}$, tedy

$$\frac{R_2}{R_1} = \frac{U_{\text{out}}}{1{,}25} - 1. \tag{3.2}$$

Nyní zbývá dosadit za U_{out} požadované hodnoty napětí:

$$\frac{R_2}{R_1} = \frac{5\,\text{V}}{1{,}25} - 1 = 3,$$
$$\frac{R_2}{R_1} = \frac{3{,}3\,\text{V}}{1{,}25} - 1 = 1{,}64.$$

Hodnota rezistoru R_2 napěťového děliče přidruženého k 5V regulátoru musí být 3×větší než hodnota rezistoru R_1. V případě 3,3V regulátoru se jedná o poměr $R_2 = 1{,}64 \cdot R_1$.

Posledním krokem je volba daných hodnot výběrem z rezistorové řady. V tomto případě se jedná o citlivý napěťový dělič, i malá nepřesnost může způsobit výkyv výstupního napětí nad maximální úroveň únosnou pro mikrokontrolér. Z tohoto důvodu je vhodné vybírat hodnoty z řady, která dosahuje malých tolerancí, ideálně 1 %. Pro dělič prvního IO byly zvoleny hodnoty

$$R_2 = 3{,}6\,\text{k}\Omega$$
$$R_1 = 1{,}2\,\text{k}\Omega,$$

pro dělič druhého pak

$$R_2 = 3{,}3\,\text{k}\Omega$$
$$R_1 = 2{,}0\,\text{k}\Omega.$$

Všechny tyto hodnoty patří do řady E24, která je dostupná i v 1% toleranci.

3.1.2 Vstupní a výstupní obvody

Rozhraní MIDI

Zadání této bakalářské práce vyžaduje dva vstupní a dva výstupní MIDI porty, tedy dvě plná MIDI rozhraní. Implementace těchto rozhraní je pak v souladu s MIDI normou [7], jak je vidět v přiloženém schématu B.1.3 [3].

Jako DIN-5 konektory byly vybrány MAB5SH výrobce HIRSCHMANN. Vstupní optočleny typu 6N138 pak vyrobil ISOCOM. Aby MIDI výstup příliš nezatěžoval mikroprocesor, byl před oba výstupní MIDI konektory přidán operační zesilovač jako buffer. Jmenovitě se jedná o TEXAS INSTRUMENTS NE5532D. Jak optočleny, tak i dvoukanálový operační zesilovač přijmou napájecí napětí z 5V větve, jak je patrné ze schématu na obr. 3.1.

Rozhraní Ethernet

Z velkého množství dostupných konektorů RJ–45 byl vybrán BEL FUSE SI-60062-F. Patří spíše mezi dražší modely, svou cenu však obhajuje lehce dostupnou dokumentací a CAD modely, které jsou pro úspěšné projektování klíčové. Z technických parametrů konektoru lze uvést maximální přenosovou rychlost, která činí 100 Mbit/s, což je pro potřeby sítě MoE zcela dostačující.

Doplňkové obvody ethernetového rozhraní pak přímo vychází ze schématu Arduino Ethernet Shield 06.

3.1.3 Logické prvky

Hlavní mikroprocesor

Hlavní mikroprocesor je klíčový prvek celého zařízení. Vykonává program, komunikuje s periferiemi, zapisuje informace do paměti atp. Výběr správného modelu byl jeden z prvních kroků při návrhu vlastního hardwaru. Pro pohodlnou migraci programu a zapojení z prototypu, který byl předmětem semestrální práce, byly pro hledaný mikroprocesor stanoveny následující požadavky:

- rodina ATmega
- minimálně dvě UART sběrnice pro MIDI rozhraní
- SPI rozhraní pro komunikaci s mikroprocesorem W5100
- vestavěná EEPROM pro uložení MAC adresy a uživatelských dat
- pohodlně osaditelné pouzdro
- skladová dostupnost
- kompatibilita s Arduino IDE výhodou

Všechny tyto požadavky splňuje mikroprocesor ATmega3208: jedná se o čip se třemi UART, jedním SPI a jedním I^2C rozhraním, vestavěnou 256bajtovou EEPROM a 32kB flash pamětí, který je dostupný v 32pinovém pouzdře TQFP. Při využití knihovny MegaCoreX [5] jej lze také naprogramovat prostřednictvím Arduino IDE. Jako hlavní mikroprocesor tedy splňuje všechny požadavky.

Dle dokumentace [6] jsou pro UART rozhraní použitelné různé piny, často ještě s možností alternativní pozice. V tomto případě však bylo vhodné nekomplikovat situaci a využívat pozice standardní. MIDI A rozhraní pracuje na UART0 (pin 30 jako Tx – MIDI OUT; pin 31 jako Rx – MIDI IN), B používá UART1 (pin 6 jako Tx; pin 7 jako Rx). Komunikace mezi tímto mikroprocesorem a procesorem W5100 zajišťujícím přístup do počítačové sítě je uskutečněna pomocí protokolu SPI, který využívá piny 2, 3, 4 a 5.

V návrhu desky jsou nachystány i cesty pro externí 32,768kHz oscilátor, avšak v současné fázi vývjoje se s osazením nepočítá. Vnitřní oscilátor by měl být teoreticky

zcela dostačující. Jako poslední jsou k mikroprocesoru připojeny LED a tři tlačítka (sekce 3.1.5).

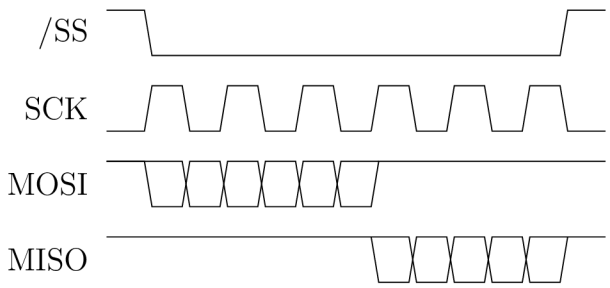
SPI

Dříve, než bude rozebrán druhý mikroprocesor nacházející se na zařízení, bude vhodné věnovat několik řádek protokolu, přes který oba mikroprocesory komunikují.

SPI (Serial Peripheral Interface) je rozhraní, které se používá na komunikaci jednoho MASTER zařízení s jedním (i více) SLAVE zařízením(i) – ve většině případů se jedná o mikroprocesory a periferie (paměti, AD/DA převodníky, senzory . . .) na desce plošných spojů. Délka jedné přenášené informace je typicky 8 bitů, přenosová rychlost je pak od 20 Mbit/s přibližně do 100 Mbit/s. Rozhraní využívá čtyři vodiče:

- SCK – synchronizační impulzy,
- MISO (Master In, Slave Out) – signálová cesta ze SLAVE zařízení do MASTER zařízení,
- MOSI (Master Out, Slave In) – signálová cesta z MASTER zařízení do jednoho nebo více SLAVE zařízení,
- SS (Slave Select) – pomocí tohoto vodiče může MASTER zařízení přepínat mezi jednotlivými SLAVE, se kterým chce komunikovat.

Pokud chce MASTER zapisovat data do SLAVE, nejprve jej aktivuje pomocí pinu SS, začne vysílat synchronizační signál SCK a přes cestu MOSI odešle instrukci *Write*, následně cílovou adresu registru a jako poslední hodnotu pro zapsání. Čtení ze SLAVE si MASTER zařízení vyžádá podobně, místo instrukce *Write* odešle instrukci *Read*, adresu registru a následně čeká na odpověď SLAVE zařízení [2]. Příklad průběhu takovéto komunikace je v hrubých obrysech znázorněn na obr. 3.2



Obr. 3.2: Orientační příklad zprávy protokolu SPI [14]

WizNet W5100

Hlavním úkolem tohoto mikrokontroléru je zprostředkovat hostujícímu zařízení přístup do počítačové sítě. Díky tomuto čipu se může přípravek chovat jako TCP host i server nebo přijímat a odesílat UDP datagramy.

Zatímco byl výběr hlavního mikrokontroleru relativně složitý a podmíněný spoustou kritérií, výběr ethernetového rozhraní byl zcela přímočarý. Byl využit stejný čip, kterým je osazen i Arduino Ethernet Shield, který byl součástí prototypu. To umožňuje využít programový kód, který byl vytvořen již v rámci semestrální práce.

WizNet W5100 pracuje v rámci MoE přípravku jako SPI SLAVE. K čipu je připojen RJ–45 konektor a přes SPI sběrnici pak hlavní mikroprocesor, tak jak lze vidět na obr. 3.1. W5100 je napájen větví 3,3 V [14].

Pro stabilizaci vnitřního oscilátoru bylo v tomto případě vhodné zapojit blízko k čipu oscilátor externí. Na desce se z tohoto důvodu nachází vývody pro 25MHz krystal. Oba tyto vývody jsou pak přes kondenzátory svedeny do země. Hodnota kapacity těchto kondenzátorů bylo vypočítána pomocí upravení vztahu [13]

$$c_{\mathrm{L}} = \frac{c_1 \cdot c_2}{c_1 + c_2} + c_{\mathrm{S}}, \tag{3.3}$$

kde je c_{L} zátěžová kapacita krystalu udávaná výrobcem, c_1 a c_2 jsou žádané kapacity kondenzátorů a c_{S} je kapacita samotných cest na desce plošných spojů. Nechť $c_1 = c_2 = c$, vztah lze potom přepsat na

$$c = 2 \cdot (c_{\mathrm{L}} - c_{\mathrm{S}}).$$

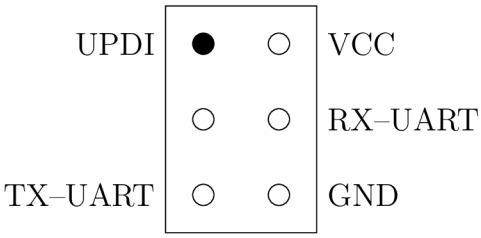
Dle dokumentace výrobce [11] zvoleného krystalu 25M–49SMD–SR je hodnota $c_{\mathrm{L}} = 20\,\mathrm{pF}$. Za hodnotu c_{S} bylo orientačně dosazeno 5 pF. Výsledná kapacita c je po dosazení

$$c = 2 \cdot 15\,\mathrm{pF} = 30\,\mathrm{pF}.$$

3.1.4 Programovací obvody

K naprogramování hlavního mikroprocesoru ATmega3208 je nutné využít jednopinové UPDI, tedy Unified Program Debug Interface – rozhraní pro programování a debugging. Jedná se o asynchronní spojení s poloviční duplexem, které dosahuje rychlosti až 0,9 Mbit/s. Mimo programování jej lze využít i pro debugging – uživatel má k dispozici neomezené množství softwarových breakpointů, dva hardwarové breakpointy, řízení běhu programu pomocí příkazů Go, Stop, Reset, Step into a další prostředky [6].

Tento UPDI pin je vyveden na programovací konektor na desce. Spolu s tímto rozhraním se na konektoru nachází také vývody rozhraní UART – pomocí těchto linek lze mikrokontrolér programovat rychleji za podmínky již nainstalovaného zavaděče (bootloaderu) Optiboot. Tento princip popisuje uživatel serveru GitHub „MCUdude“ ve svém projektu MegaCoreX [5]. Nastínil i doporučené rozložení programovacího konektoru (obr. 3.3).



Obr. 3.3: Rozložení programovacího konektoru podle [5]

3.1.5 Uživatelské rozhraní

Do této sekce lze započítat různé LED a tlačítka jimiž je zařízení vybaveno. Stavové LED jsou následující:

- `PWR` – kontrola přítomného napájecího napětí (větev 3,3 V)
- `LINK` – signalizace korektního připojení a aktivity ethernetového rozhraní
- `ACT` – signalizace aktivity mikrokontroléru
- `ERR` – LED signalizující chybu, tabulka chybových kódů je k dispozici jako příloha A.4

LED je přidružena i ke každému MIDI portu.

Přípravek je dále vybaven třemi tlačítky. Jedno slouží k restartu mikrokontroléru, druhé zapíná/vypíná funkci DHCP a třetí zapíná/vypíná THRU MODE (viz sekce 3.2).

3.2 Software

Programování mikrokontrolérů je do značné míry odlišné od programování „klasické“ desktopové nebo mobilní aplikace. Psaní kódu doprovází časté konzultace s dokumentací daného čipu z důvodu velkého počtu registrů do kterých je třeba zapisovat nebo z nich číst. Tuto nepříjemnost vyřešili v roce 2005 Massimo Banzi a David Cuartielles vytvořením platformy Arduino. Tato platforma obsahuje vývojovou desku a vývojové prostředí, obé uživatelsky přívětivé a rychle pochopitelné, přitom velice variabilní [1]. Vývoj prototypu i výsledného produktu je s touto platformou neodlučitelně spojen.

Program přípravku je napsán v jazyce `C/C++`, pro vývoj bylo použito primárně prostředí Arduino IDE, občasně pak Visual Studio Code s patřičnými doplňky. I přes nutnou strojově relativně nízkou úroveň jazyka bylo možné využít prvky vyšších stylů programování – objekty. Periferní zařízení lze pak pro pohodlnější manipulaci abstraktizovat do instance dané třídy. Jako příklad je zde uveden následující výpis:

```
//vytvoření instance abstraktizující ethernetové rozhraní
EthernetUDP eUDP;
```

```
//příkaz pro spuštění naslouchání na daném síťovém portu
eUDP.begin(PORT);

//kontrola přijetí zprávy a případný zápis do proměnné
if (eUDP.parsePacket()) {
    eUDP.read(message);
}
```

Pomocí několika málo řádek lze popsat relativně složitý proces otevření socketu pro příjem UDP zpráv. Díky prvkům OOP tedy není třeba přistupovat k jednotlivým registrům ani přímo ovládat komunikaci pomocí SPI.

Další zjednodušení představuje projekt uživatele GitHubu „MCUdude“ MegaCoreX [5]. Tento doplněk do vývojového prostředí rozšiřuje paletu programovatelných mikročipů. Díky tomu lze kód psaný pro Arduino UNO s procesorem ATmega328P nahrát na MoE zařízení s procesorem ATmega3208 pouze s minimálními úpravami. Výsledek samozřejmě nebude dokonale optimalizován, pro účely tohoto zařízení je to však dostačující.

Při vývoji softwaru bylo dbáno na maximalizaci variability směřování MIDI příkazů. Na funkční rovině je chování inspirováno systémem *Dante*, který se používá pro přenos zvukového signálu přes počítačovou síť. *Dante* umožňuje pokročilé směřování signálů napříč síťovými zařízeními a jejich kanály, ovládané pomocí přehledné matice na ovládacím PC. K tomuto byl směřován i protokol MoE. V tomto případě je však situace jednodušší v tom, že jedno fyzické MIDI rozhraní pracuje s 16 softwarovými kanály – směřování zpráv do jednotlivých kanálů tedy není řešeno fyzicky, ale softwarově, manipulací se `STATUS BAJTEM` dané zprávy.

V následujícím textu budou zmiňovány dva typy zpráv: Příchozí, odchozí MIDI zprávy, které se týkají hardwarové sběrnice, tedy přímo UART rozhraní – dále jen „MIDI zprávy“ – a příchozí, odchozí UDP zprávy, které se týkají pouze síťové komunikace – dále jen „UDP zprávy“.

3.2.1 Databáze spojení

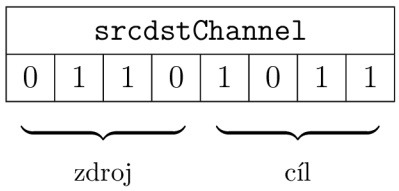
Základní stavební jednotkou programu je databáze (matice) spojení. Každý přípravek disponuje vlastní databází, kterou lze vzdáleně upravovat (záznamy je možno přidávat i mazat) a uložit do `EEPROM` paměti mikrokontroléru, takže po restartu přípravku bude znovu načtena. Její architektura by se dala připodobnit k relační databázi. Každý záznam spojuje kanál a port hostitelského vstupu s kanálem, portem a adresou cílového zařízení.

Ve zdrojovém kódu je představována polem struktur `subscriptions[]`:

```
struct subscription
{
    byte srcChannel;
    byte dstChannel;
    byte dstIPnib;
} _subscriptions[MAX_SUBS];
```

Bajt `srcChannel` představuje zdrojový MIDI kanál hostitelského zařízení, kterého se tento záznam spojení týká. MoE zařízení tedy na těchto kanálech naslouchá. Bajt `dstChannel` je výstupní MIDI kanál cílového zařízení. Pokud je přijata zpráva, jejíž kanál se týká daného záznamu, je před odesláním její kanál přepsán právě touto hodnotou. Přesnější průběh těchto událostí je rozebrán v sekci 5.3.1. Poslední bajt `dstIPnib` určuje čtvrtý nibble IP adresy cílového zařízení – komunikovat spolu tedy mohou zařízení pouze v síti s maskou 255.255.255.0.

Pozastavme se nyní nad paměťovou náročností tohoto pole struktur. Z kapitoly 1.2 vyplývá, že každé MIDI rozhraní umožňuje komunikovat po 16 kanálech, tedy kanálech 0–15. K uložení čísla kanálu do paměti nebo `EEPROM` jsou tedy potřeba pouze 4 bity – to umožňuje zapsat údaj o vstupním i výstupním kanálu spojení do jednoho bajtu `srcdstChannel` tak, jak je vidět na obr. 3.4. Tento princip byl využit



Obr. 3.4: Možné kódování zdrojového a cílového kanálu v bajtu `srcdstChannel`

v rámci semestrální práce, prototyp totiž obsahoval pouze jedno MIDI rozhraní. Bakalářský projekt však počítá se dvěma rozhraními, číslo kanálu a portu kóduje už 5 bitů[4] a výše popsaný šetrný způsob skládání informací již bohužel nejde využít. Po zvážení situace bylo implementováno programátorsky nejjednodušší řešení, tedy vyhradit označení jednoho kanálu a portu celý bajt[5]. Tři bity tohoto bajtu zůstanou v této fázi nevyužité (v případných dalších fázích vývoje se nabízí k popisů vlastností daného spojení nebo k zapínání a vypínání přenosu určitých typů MIDI zpráv). Jeden záznam tohoto pole tedy v paměti zabírá tři bajty, ze kterých je šest bitů nevyužito.

K manipulaci s obsahem této databáze dochází ve dvou případech. K prvnímu dojde tehdy, když zařízení dostane zprávu `beacon`, která značí, že se v síti objevilo

[4] 4 bity pro kanál + 1 bit pro port

[5] port A, kanály 1–16 = `0x00-0x0F`; port B, kanály 1–16 = `0x10-0x1F`

nový MoE přípravek. V tuto chvíli si vloží do tohoto pole nový záznam obsahující implicitní hodnoty; svůj MIDI kanál 1 portu A propojí s kanálem 1 portu A cílového zařízení. Druhý případ je vyvolán uživatelem, který přes *MoE Matrix Editor* vytvoří nové spojení.

3.2.2 Protokol MoE

Za účelem spolehlivé výměny MIDI příkazů byl vyvinut protokol, přes který zařízení navzájem komunikují. Tento protokol implementuje i síťový ovladač *MoE Matrix Editor* (kapitola 4). Architektura MoE protokolu je podobná architektuře MIDI. Jedna zpráva se skládá ze čtyř bajtů, z nichž první určuje povahu zprávy. Tento bajt je každým zařízením vždy kontrolován jako první, dle jeho hodnoty je pak se zprávou patřičně zacházeno. Tuto „MoE značku“ lze přirovnat ke `STATUS BAJTU` MIDI zprávy. Tabulku všech implementovaných MoE značek lze nalézt v příloze A.2. Další tři bajty pak představují tělo zprávy, hodnotu, se kterou se dále pracuje.

Pro příklad zpráva

```
0xA3, 0xE0, 0xFF, 0xFF
```

dle A.1 a A.2 znamená „zpráva zapouzdřující MIDI příkaz ohyb tónu úplně nahoru“. Zpráva

```
0x0F, 0x00, 0x73, 0x10
```

má už složitější význam. Vyjadřuje příkaz „zapsat nový záznam do databáze spojení: **z** vlastního kanálu 1 portu A **do** zařízení s IP adresou `x.x.x`.115, kanálu 1 portu B“ (viz sekce 3.2.1).

Je vhodné zmínit, že protokol svou *peer-to-peer* povahou implikuje použití UDP pro zapouzdření MoE zpráv. Komunikace probíhá na portu 50 000, zařízení jsou schopna se do lokální sítě připojit s adresou přidělenou DHCP serverem, nebo s adresou pevně nastavenou uživatelem. Jak již bylo řečeno, zařízení se navzájem rozeznávají pouze posledním nibblem IP adresy, v případě jejich nasazení do sítě s aktivním DHCP serverem musí mít tato síť již zmíněnou masku 255.255.255.0. V případě využití statických IP adres toto již není nutná podmínka, zařízení se však musí nacházet v též podsíti.

Použití DHCP serveru je potenciálně nebezpečné. Ze sekce 2.4 je zřejmé, že DHCP propůjčuje adresy pouze dočasně, po uplynutí jisté doby může být zařízení přidělena IP adresa nová. Toto představuje problém pro správné fungování databáze spojení (3.2.1), která rozlišuje zařízení podle posledního bajtu jejich IP adresy. Pokud je zařízení tato adresa změněna, již nebude přijímat zprávy pro něj určené. Z tohoto důvodu je vhodné DHCP server sítě s MoE zařízeními nastavit tak, aby daným zařízením (MAC adresám) přidělil vždy stejnou IP adresu.

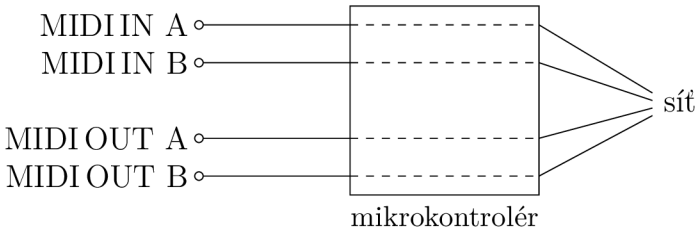
3.2.3 Spouštěcí direktivy

Pro korektní fungování a nastavování MoE zařízení byla na začátek běhu programu vložena sekce načítání „spouštěcích direktiv“. Tyto direktivy jsou binární povahy a jsou uložené v sedmém bajtu `EEPROM` paměti (viz příloha A.3). Udávají základní nastavení přípravku – zapnutí a vypnutí THRU MODE (viz sekce 3.2.4), nastavení IP adresy, apod. Uživateli jsou přístupná opět za pomoci aplikace *MoE Matrix Editor*.

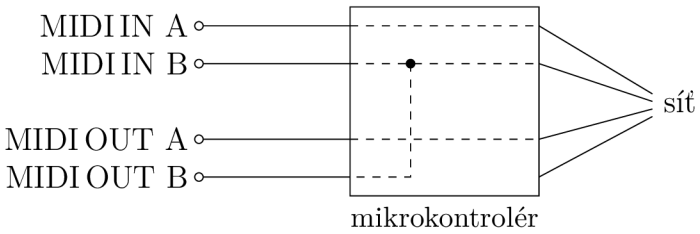
3.2.4 THRU mode

MIDI zařízení mají často vedle konektorú IN a OUT také THRU, ostatně s tímto konektorem počítá i norma [7]. Tato zařízení mají THRU konektor většinou napevno propojen se vstupem (za optočlenem) – obr. 1.2. Tato hardwarová implementace ovšem v případě MoE zařízení nebyla vhodná.

Zatímco obr. 3.5 zobrazuje standardní mód, obr. 3.6 zobrazuje chování přípravku ve stavu zapnutého THRU MODE. Implementace je tedy čistě softwarová a va-



Obr. 3.5: THRU MODE neaktivní



Obr. 3.6: THRU MODE aktivní

riabilní. V tomto stavu se nejvíce přibližuje normou [7] stanoveném schématu na obr. 1.2. Z obr. 3.6 je zřetelné, že informace pro MIDI výstup portu B do zařízení ze sítě stále přichází, to je však ignoruje a na výstupní MIDI sběrnici portu B posílá pouze vstup též sběrnice.

3.2.5 Funkce AUTODISCOVER

Tato funkce se opět zapíná a vypíná pomocí stejnojmenné spouštěcí direktivy. Ovlivňuje chování přípravku bezprostředně po připojení do sítě. Uživateli usnadňuje práci tím, že automaticky propojí nové zařízení se všemi zařízeními, které v síti již působí. Propojuje první kanál portu A hostitelského zařízení s prvními kanály portů B všech zařízení v síti.

Docílí toho tak, že odešle specifickou zprávu `_beacon` na broadcast adresu sítě, tedy všem jejím účastníkům. Všechny přípravky, které tuto zprávu obdrží (a které mají tuto funkci zapnutou), si vytvoří nový záznam spojení propojující jejich kanál 1 portu A s týmž kanálem téhož portu nového zařízení. Zároveň odešlou unicast odpověď novému zařízení – zprávu `_handshake`. Nové zařízení si za za každou přijatou zprávu `_handshake` vytvoří nový záznam do svého pole `_subscriptions`, který opět propojuje první kanál portu A hosta s prvním kanálem portu A odesílatele, tedy postupně všech přípravků v síti se zapnutou funkcí AUTODISCOVER.

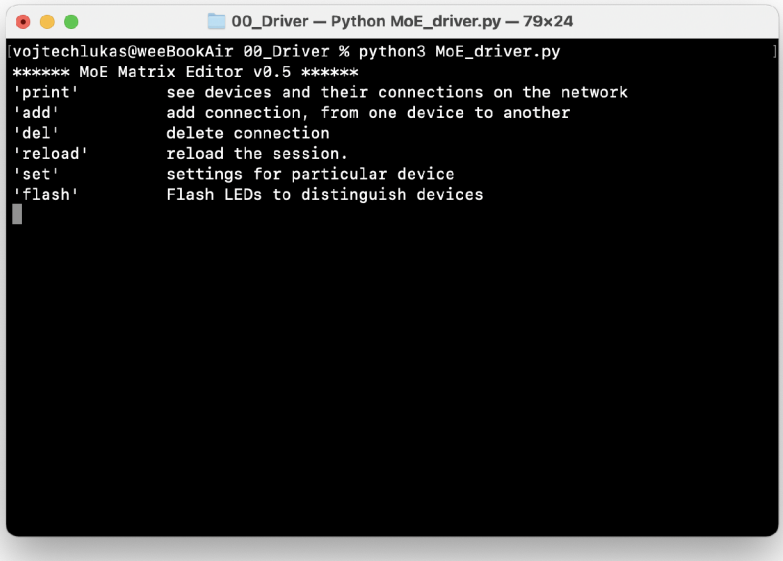
3.2.6 Chyby

Každý program by měl být připraven na chyby, které za jeho běhu mohou vzniknout. Reakce je relativní podle závažnosti chyby – program se může pokusit o automatickou nápravu nebo zřetelnou signalizaci uživateli, popřípadě obojí.

Software tohoto přípravku rozeznává několik chyb (jejich kompletní seznam je uveden v příloze A.4). Některé z nich vedou k restartu zařízení, jiným stačí k vyřešení změna nastavení pomocí *MoE Matrix Editoru*.

4 Síťový editor

Pro centrální dálkové ovládání databází spojení na jednotlivých přípravcích v síti byl vytvořen jednoduchý program v jazyce Python, který je nutné spustit na PC, jenž je připojen ve stejné síti jako všechny přípravky (připojení lze realizovat i bezdrátově pomocí Wi–Fi).

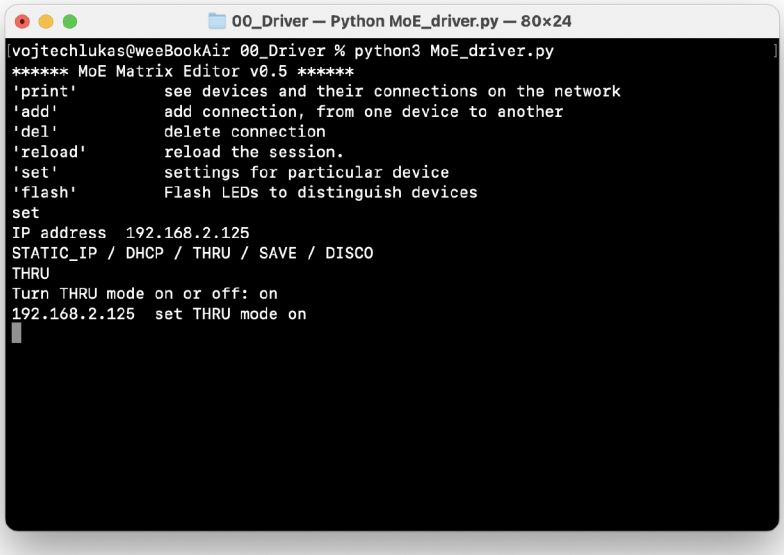


Obr. 4.1: Konzolová aplikace *MoE Matrix Editor* – úvodní obrazovka

Za běhu programu má uživatel na výběr z šesti hlavních příkazů.

`print`	Na konzoli vytiskne všechna zařízení v síti a jejich aktuální spojení.
`add`	Přidá vybranému zařízení záznam do databáze spojení `subscriptions`.
`del`	Vymaže vybranému zařízení záznam z databáze spojení.
`reload`	Vyžádá si po všech zařízeních na síti aktualizaci jejich databází spojení.
`set`	Slouží k nastavení jednotlivých zařízení.
`flash`	Rozsvítí LED na přípravku pro lokalizaci v místnosti, případně rozlišení přípravků od sebe.

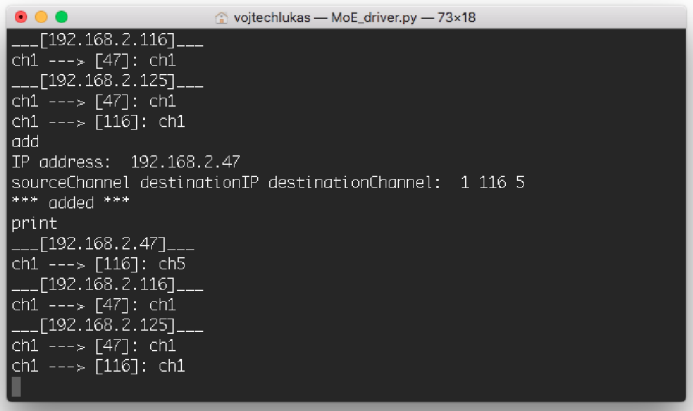
Příkaz `set` nabízí po výběru zařízení další možnosti



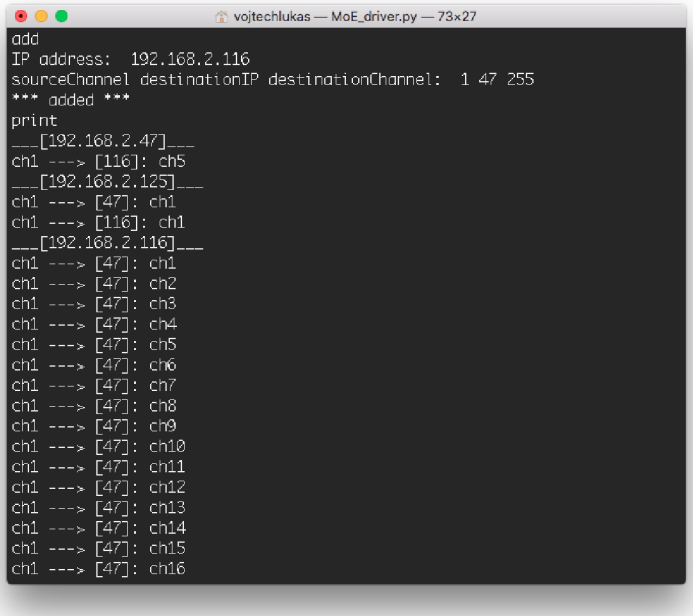
Obr. 4.2: Konzolová aplikace *MoE Matrix Editor* – nastavení jednotlivých zařízení

`STATIC_IP`	Nastaví statickou IP adresu zařízení k využití při vypnuté funkci DHCP.
`DHCP`	Zapíná/vypíná DHCP.
`THRU`	Zapíná/vypíná THRU MODE
`SAVE`	Příkaz pro uložení pole `_subscriptions` do `EEPROM` daného zařízení
`DISCO`	Zapíná/vypíná funkci AUTODISCOVER (3.2.5)

Při zápisu do databáze spojení (příkaz `add`) bylo pro zjednodušení přidáno „makro“, které se spustí zadáním hodnoty `255` do pole pro cílový kanál zařízení. V tomto případě bude namísto jediného spojení vytvořeno šestnáct unikátních spojení tak, aby jeden MIDI kanál zdrojového zařízení přijímalo všech šestnáct kanálů zařízení cílového. Je třeba dodat, že toto makro reálně využitelné není, velice však usnadní práci při testovacích a měřících aktivitách. Analogicky k tomuto makru funguje i příkaz `del` s parametrem `255` u hodnoty `destinationChannel`.



Obr. 4.3: Konzolová aplikace *MoE Matrix Editor* – vytváření spojení



Obr. 4.4: Konzolová aplikace *MoE Matrix Editor* – využití vkládacího makra

5 Průběh komunikace

Tato kapitola spojuje veškeré informace poskytnuté v předešlých sekcích. Jejím cílem je shrnout kompletní fungování vyvinutého přípravku prostřednictvím modelové situace.

V dalším textu bude popsán průběh připojení přípravku k síti, automatické rozpoznání a zacházení s přijatými MIDI a UDP zprávami, spolu s reakcemi na příkazy odeslané síťovým editorem.

5.1 Úvodní inicializace

Ihned po zapnutí zařízení je z `EEPROM` paměti přečteno prvních šest bajtů, které představují MAC adresu přípravku. Následně je načten sedmý bajt, který představuje spouštěcí direktivy (3.2.3), od kterých se odvíjí další chování přípravku. V případně zapnuté direktivy `STC_DHCP_ENABLE` je z `EEPROM` načtena i uživatelsky nastavitelná statická IP adresa.

5.2 Připojení k lokální síti

Bezprostředně po inicializaci se zařízení pokusí připojit do sítě. Podle stavu direktivy `STC_DHCP_ENABLE` se o to pokusí buď zažádáním DHCP serveru o přidělení IP adresy, nebo si zařízení nastaví svou IP adresu napevno zadanou uživatelem.

Pokud se připojení pomocí DHCP nepodaří, zařízení přejde do chybového stavu a bude čekat na restart.

```
if (stcRead(STC_DHCP_ENABLE)) {
    if (!Ethernet.begin(_myMac)) {
        error(ERR_DHCP_FAIL);
    }
}
else {
    Ethernet.begin(_myMac, _staticIP);
}
```

Pokud se toto připojení podaří, podle přidělené IP adresy se vyplní pomocné proměnné. Dále se v této chvíli otevírá UDP socket a UART porty.

Jako poslední inicializační krok se provede funkce AUTODISCOVER (3.2.5), pokud je zapnutá příslušná direktiva.

5.3 Běh programu

Po těchto inicializačních krocích se program dostává do nekonečné smyčky. Aktivně naslouchá na dvou UART rozhraních a UDP socketu. Jakmile je na jednom z těchto zařízení přítomna nějaká zpráva, je rozklíčována a dále zpracovávána.

5.3.1 Přijetí MIDI zprávy

Vstupní UART sběrnice, tedy MIDI porty, jsou v každé iteraci programu kontrolovány na přítomnost „nepřečtených“ bajtů. Pokud je tato přítomnost detekována, je nejprve zjištěno o jaký bajt se jedná a podle toho je uložen do patřičné proměnné. Program je v této fázi relativně složitý z důvodu ošetření módu Running Status (1.2.1). Díky této komplexnosti je však možné, aby obě MIDI sběrnice úspěšně přijímaly a zpracovávaly jak standardní zprávy, tak i zprávy v módu Running Status.

Jakmile jsou postupně zaznamenány všechny bajty tvořící platnou MIDI zprávu, je volána funkce pro její odeslání přes UDP (`sendUDP()`). Tuto funkci sdílí oba porty, v první fázi tedy rozlišuje, který port ji volá. Následně je extrahován kanál MIDI zprávy, který je poté v cyklu porovnáván se `_subscriptions[i].sourceChannel`, tedy s příslušnou proměnou v každém záznamu databáze spojení. Dojde-li ke shodě, začne konstrukce UDP zprávy podle dalších informací v odpovídajícím záznamu. Jako první je do datagramu zapsána příslušná MoE značka. Dále, jedná-li se o tříbajtovou MIDI zprávu[1], je ve `STATUS BAJTU` přepsán její kanál hodnotou proměnné `_subscriptions[i].destinationChannel`. Tento `STATUS BAJT` je následně vepsán do datagramu, spolu se zbývajícími `DATA BAJTY`. Tento složitě působící proces zajistí, že do cílového zařízení dorazí příslušná zpráva se správným kanálem, který bude v souladu s databází spojení odesílatele. Tento děj je názorně patrný na výpisu 5.1.

Výpis 5.1: funkce `sendUDP()`

```
void Controller::sendUDP(bool port, byte data0, byte data1,
                         byte data2) {
  //zjištění portu, který volá tuto funkci
  if (!port) _srcChA = (data0 & 0x0F);
  else _srcChB = (data0 & 0x0F) + 16;

  //procházení databáze spojení
  for (byte i = 0; i < _numSubs; i++)
  {
    //dojde-li ke shodě...
```

[1] Dvoubajtové zprávy, tedy zprávy módu Running Status, vyvolávají přetíženou funkci, která má v argumentu pouze `port`, `data1` a `data2`.

```
    if (((!port) && (_subscriptions[i].sourceChannel ==
    _srcChA)) ||
    ((port) && (_subscriptions[i].sourceChannel ==
    _srcChB)))
    {
      //...začíná konstrukce UDP zprávy
      //z DATA BAJTU je smazán MIDI kanál
      data0 &= 0xF0;
      //Za vstupní MIDI kanál je dosazen kýžený výstupní
      data0 |= _subscriptions[i].destinationChannel;
      //formátování IP adresy příjemce
      _destinationIP[3] = _subscriptions[i].dstIPnib;
      //začátek datagramu
      eUDP.beginPacket(_destinationIP, MOE_PORT);
      //nastavení MoE značky  a výst. kanálu
      if (_subscriptions[i].destinationChannel > 15) {
        data0 |= (_subscriptions[i].destinationChannel - 16);
        eUDP.write(0xB3);
      }
      else {
        data0 |= (_subscriptions[i].destinationChannel);
        eUDP.write(0xA3);
      }
      //kontrukce datagramu
      eUDP.write(data0);
      eUDP.write(data1);
      eUDP.write(data2);
      //konec a odeslání datagramu
      eUDP.endPacket();
    }
  }
}
```

5.3.2 Přijetí UDP zprávy

Stejně jako UART sběrnice i UDP socket je v každé programové iteraci kontrolován. Pokud je v něm dostupná zpráva, je načtena do pole bajtů `_incomingUDP[4]`. Podle prvního (nultého) bajtu tohoto pole je se zprávou dále nakládáno.

Nejjednodušší reakce jsou na zprávy se značkou `0xA2` a `0xA3`. Jejich obsah je ihned zapisován na výstup UART sběrnic, tedy do výstupních MIDI portů. Podobná

situace nastane v případě zpráv se značkou `0xB2` a `0xB3`, zde však dojde k zapisování na fyzický výstup pouze tehdy, je-li vypnutý THRU MODE.

V kódu je toto větvení uskutečněno pomocí podmínky `switch`. Ve výpisu 5.2 je zobrazeno větvení a reakce na určité MoE značky.

Výpis 5.2: Vybrané reakce na určité příchozí UDP zprávy

```
if (eUDP.parsePacket()) {
  eUDP.readByte(_incomingUDP. 4);
  switch (_incomingUDP[0]) {
    //...
    case 0xA2:
      //příchozí zpráva určená na výstupní MIDI port A
      Serial.write(_incomingUDP[1]);
      Serial.write(_incomingUDP[2]);
      break;
    case 0xB2:
      //příchozí zpráva určená na výstupní MIDI port B
      //nejprve proběhne kontrola direktivy THRU MODE
      if (!stcRead(STC_THRU_MODE)) {
        Serial1.write(_incomingUDP[1]);
        Serial1.write(_incomingUDP[2]);
      }
      break;
    case 0x0F:
      //zpráva od MoE Matrix Editoru vkládající nový
      //záznam do databáze spojení
      addSubscription(_incomingUDP[1], _incomingUDP[2],
      _incomingUDP[3]);
      break;
    case 0xD0:
      //příkaz od MoE Matrix Editoru vypínající DHCP
      stcSet(STC_DHCP_ENABLE, false);
      break;
    //...
  }
}
```

Kromě těchto dvou hlavních funkcí je v hlavní programové smyčce už pouze funkce `maintain()`, která slouží pro aktualizaci adresy poskytované DHCP serverem. Tato adresa by měla být v době používání MoE sítě neměnná, to však záleží na nastavení DHCP serveru (2.4).

Závěr

Výstupem této bakalářské práce je přípravek, který disponuje ethernetovým rozhraním, dvěma vstupními a výstupními MIDI porty napájený buď prostřednictvím USB, nebo DC jack konektoru. Přípravek se automaticky připojí do počítačové sítě a dále funguje již autonomně. Dokáže přijmout MIDI zprávu, přeformátovat ji v souladu s daným záznamem databáze spojení a s využitím UDP protokolu poslat správnému příjemci. Databáze spojení je taktéž editovatelná pomocí programu *MoE Matrix Editor*. Tento program dále umožňuje zapínat specifické funkce jednotlivých přípravků, ku příkladu THRU MODE nebo AUTODISCOVER.

Přípravek dokáže přijímat tříbajtové (nebo dvoubajtové Running Status) MIDI zprávy, což jsou nejčastější přenášené zprávy užívané MIDI klaviaturami nebo kontroléry. Nedokáže však reagovat na všechny druhy MIDI příkazů – pro další vývoj se nabízí implementace SysEx, MTC nebo Real Time zpráv.

Jako uspokojivé lze vnímat dosažené hodnoty latence mezi přijetím MIDI zprávy na sériové sběrnici jednoho zařízení a odesláním téže zprávy na sériovou sběrnici zařízení druhého. Při tomto měření probíhala komunikace mezi samotnými zařízeními pochopitelně prostřednictvím počítačové sítě. Dle přílohy C.1 vykazuje dolní hranice celkové latence MoE řešení 2,18 ms v případě jediného záznamu v databázi spojení. S každým dalším záznamem pak latence přirozeně roste. Maximální naměřená hodnota latence (mezi první přijatou a poslední odeslanou zprávou) pak dosahovala 20,92 ms. Při živé hudební produkci s využitím MIDI klávesového nástroje by neměla celková latence přesáhnout 5 ms. S touto premisou lze prohlásit, že MoE řešení by mohlo být za určitých podmínek použitelné i pro časově náročnější podmínky.

Literatura

[1] Arduino. *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2021-5-26]. Dostupné z: <https://cs.wikipedia.org/wiki/Arduino>.

[2] FRENZEL, Louis. *Handbook of serial communications interfaces: A comprehensive compendium of serial digital input/output (I/O) standards*. Oxford: Elsevier, 2016. ISBN 978-0-12-800629-0.

[3] GHASSAEI, Amanda. *Send and Receive MIDI With Arduino* [online]. [cit. 1. 12. 2020]. Dostupné z URL: <https://www.instructables.com/Send-and-Receive-MIDI-with-Arduino/>.

[4] GUÉRIN, Robert. *Velká kniha MIDI: standardy, hardware, software*. Brno: Computer Press, 2004. 328 s. ISBN 80-722-6985-2.

[5] „MCUDUDE“. *MegaCoreX* [online]. Dostupné z: <https://github.com/MCUdude/MegaCoreX>

[6] MICROCHIP TECHNOLOGY. *ATmega3208/3209 Data Sheet* [online]. [cit. 2021-5-25]. 2021, 552 s. Dostupné z <https://ww1.microchip.com/downloads/en/DeviceDoc/ATmega3208-09-DataSheet-DS40002174C.pdf>.

[7] MIDI MANUFACTURERS ASSOCIATION, The. *MIDI 1.0 Detailed Specification.* Document Version 4.2.1. Los Angeles, CA: MMA, 1996. 58 s. Dostupné také z: <https://www.midi.org/specifications/m1-v4-2-1-midi-1-0-detailed-specification-96-1-4/download>.

[8] MMA TECHNICAL STANDARDS BOARD/AMEI MIDI COMMITTEE. *MIDI 1.0 Electrical Specification Update.* 2014. s. 1–3. Dostupné také z: <https://www.midi.org/downloads?task=callelement&format=raw&item_id=100&element=f85c494b-2b32-4109-b8c1-083cca2b7db6&method=download>.

[9] MNEIMNEH, Saad. *Computer Networks UDP and TCP*. Hunter College of CUNY. New York. 2008. Dostupné také z: <http://www.cs.hunter.cuny.edu/~saad/courses/networks/notes/note7.pdf>.

[10] MOLNÁR, Karol. *Praktikum informačních sítí*. 1. Brno: Vysoké učení technické v Brně, 2013.

[11] SRPASSIVES. *Resistance Welded HC-49/S Surface Mount Package* [online]. [cit. 2021-5-27]. Dostupné z <https://www.tme.eu/Document/00238472573cc96d704246f7ceaf8322/HC-49.pdf>.

[12] ST MICROELECTRONICS. *MC3406xx DC-DC converter control circuits* [online]. 2013 [cit. 2021-5-26]. Dostupné z <https://www.tme.eu/Document/822ff90970b8de68f08f609d00979f57/MC34063xx.pdf>.

[13] STMICROELECTRONICS. *Oscillator design guide for STM8AF/AL/S, STM32 MCUs and MPUs* [online]. 2020 [cit. 2021-5-26]. Dostupné z: <https://www.st.com/resource/en/application_note/cd00221665-oscillator-design-guide-for-stm8af-al-s-stm32-mcus-and-mpus-stmicroelectronics.pdf>.

[14] WIZNET. *W5100 Datasheet* [online]. 2008, 70 s. Dostupné z <https://www.sparkfun.com/datasheets/DevTools/Arduino/W5100_Datasheet_v1_1_6.pdf>.

Seznam symbolů, veličin a zkratek

DAW Digital Audio Workstation – digitální pracovní stanice pro náběr a úpravu vícestopého záznamu

DHCP Dynamic Host Configuration Protocol

IDE Integrated Development Enviroment – integrované vývojové prostředí

IO Integrovaný obvod

ISO/OSI ISO: Open Systems Interconnection – model navrhující protokol komunikace mezi zařízeními

LED Light-Emiting Diode – dioda, která vyzařuje viditelné světlo

MIDI Musical Instrument Digital Interface – digitální rozhraní hudebního nástroje

MoE MIDI over Ethernet – MIDI po Ethernetu

MSb Most Significant bit – nejvýznamnější bit (většinou v bajtu)

MSC MIDI Show Control – subprotokol pro ovládání scénické techniky, zejména pomocí příkazů GO, STOP, popř. RESUME. [7]

OOP Objektově Orientované Programování

PWM Pulse Width Modulation – šířková modulace obdélníkového signálu

RF Radio Frequency

SPI Serial Perpiheral Interface – Sériové rozhraní pro periferie

UART Universal Asynchronous Receiver/Transmitter – univerzální asynchronní přijímač/vysílač

UDP User Datagram Protocol – jednoduchý síťový protokol, který umožňuje výměnu zpráv „host-to-host“ [9]

UPDI Unified Program Debug Interface

USB Universal Serial Bus – univerzální sériová sběrnice pro komunikace s periferními zařízeními.

Seznam příloh

A Tabulky

A.1 MIDI status bajty

Tab. A.1: Tabulka MIDI `STATUS BAJTŮ` [7]

Název		Hex. hodnota	Bin. hodnota
Note-Off	Nota vypnuta	`0x8n`	`1000 nnnn`
Note-On	Nota zapnuta	`0x9n`	`1001 nnnn`
Poly Key Pressure	Polyfonický tlakový ovladač	`0xAn`	`1010 nnnn`
Control Change	Změna kontroléru	`0xBn`	`1011 nnnn`
Program Change	Změna programu	`0xCn`	`1100 nnnn`
Channel Pressure	Monofonický tlakový ovladač	`0xDn`	`1101 nnnn`
Pitch Bend	Ohyb výšky tónu	`0xEn`	`1110 nnnn`

A.2 MoE značky

Bajt	Popis
⋮	
`0x08`	Broadcast zpráva od řídícího PC, která sonduje účastníky sítě
⋮	
`0x0E`	Unicast zpráva od řídícího PC, která zařízení maže záznam z databáze `subscriptions`.
`0x0F`	Unicast zpráva od řídícího PC, která zařízení vkládá nový záznam do databáze `subscriptions`, který je obsahem datagramu
⋮	
`0x80`	Unicast odpověď na sondáž PC, součástí datagramu je i jeden záznam databáze `subscriptions`
⋮	
`0xA2`	Dvoubajtová MIDI zpráva pro port A
`0xA3`	Tříbajtová MIDI zpráva pro port A
`0xB2`	Dvoubajtová MIDI zpráva pro port B
`0xB3`	Tříbajtová MIDI zpráva pro port B

⋮	
0xD0	Zpráva pro úpravu spouštěcí direktivy: DHCP vypnuto
0xD1	Zpráva pro úpravu spouštěcí direktivy: DHCP zapnuto
⋮	
0xD3	Příkaz pro uložení_subscriptions do EEPROM
0xD4	Zpráva pro úpravu spouštěcí direktivy: AUTODISCOVER vypnuto
0xD5	Zpráva pro úpravu spouštěcí direktivy: AUTODISCOVER zapnuto
0xD6	Příkaz pro rozsvícení LED pro lokalizaci daného zařízení v místnosti
0xD7	Zpráva pro úpravu spouštěcí direktivy: THRU vypnuto
0xD8	Zpráva pro úpravu spouštěcí direktivy: THRU zapnuto
⋮	
0xC1	Zpráva s prvními dvěma bajty statické IP adresy pro zaspaní do EEPROM
0xC2	Zpráva se zbylými dvěma bajty statické IP adresy pro zapsání do EEPROM
⋮	
0xEE	Zpráva handshake posílaná jako odpověď na přijatou zprávu beacon
⋮	
0xFF	Zpráva beacon sloužící pro upozornění na vlastní přítomnost

A.3 Mapa EEPROM paměti přípravku

Tab. A.3: Mapa bajtů paměti EEPROM přípravku

Adresa bajtu	Popis
0-5	MAC adresa zařízení
6	STC: Bajt spouštěcích direktiv
7-10	Statická IP adresa
⋮	
62	_numSubs: počet uložených záznamů databáze spojení _subscriptions
63-255	Vyhrazené místo pro databázi spojení _subscriptions

A.4 Chyby a chybové signalizace

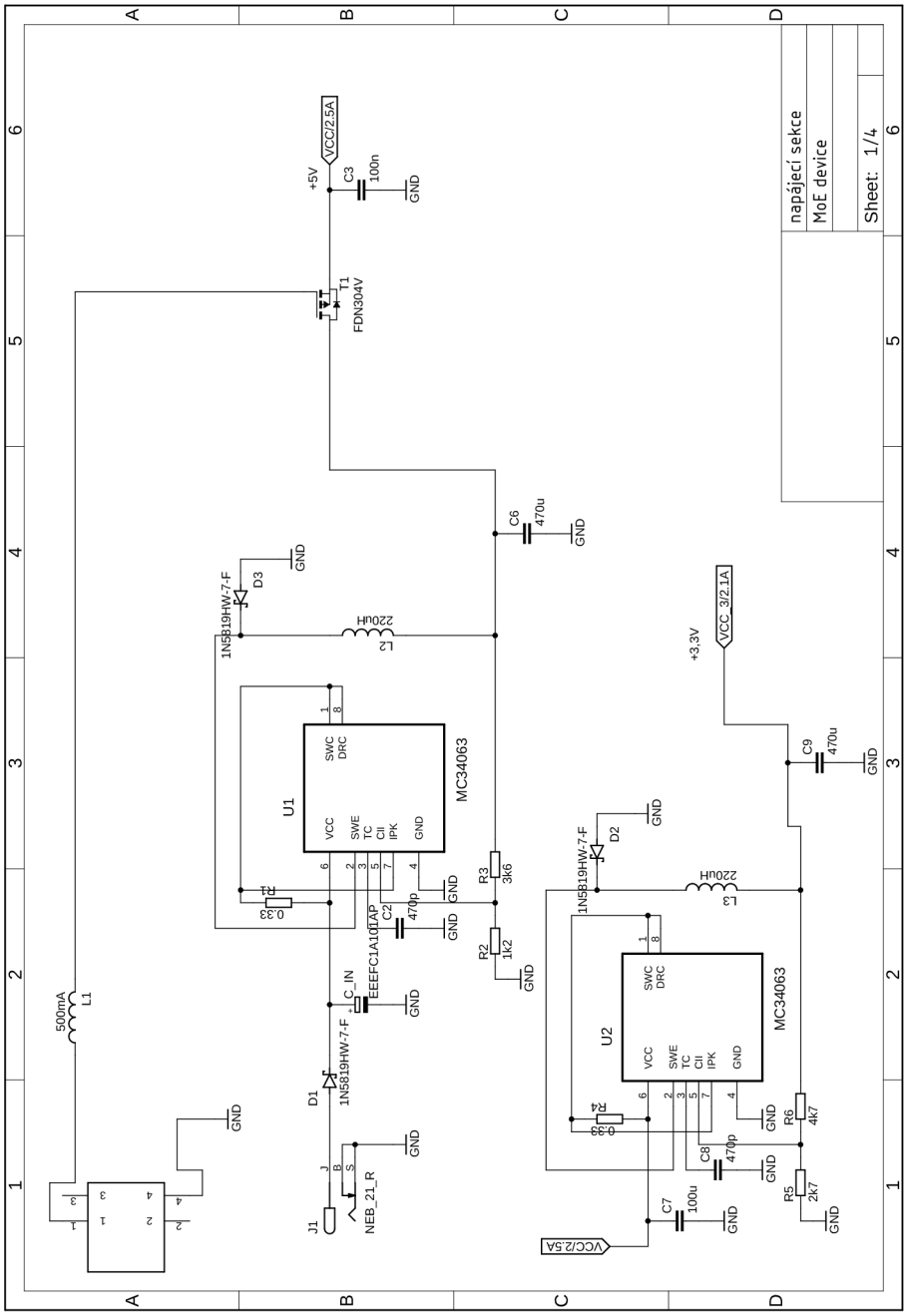
Tab. A.4: Kódy chyb, jejich význam a signalizace

Kód chyby	Význam chyby	Signalizace
ERR_DHCP_FAIL	Pokus o připojení do sítě pomocí DHCP byl neúspěšný	LED ERR svítí
ERR_FULL_SUBS	Plná databáze spojení _subscriptions	LED ERR čtyřikrát zabliká
ERR_WRNG_CHNLS	Pokus o nesmyslný zápis do databáze spojení _subscriptions	LED ERR dvakrát zabliká

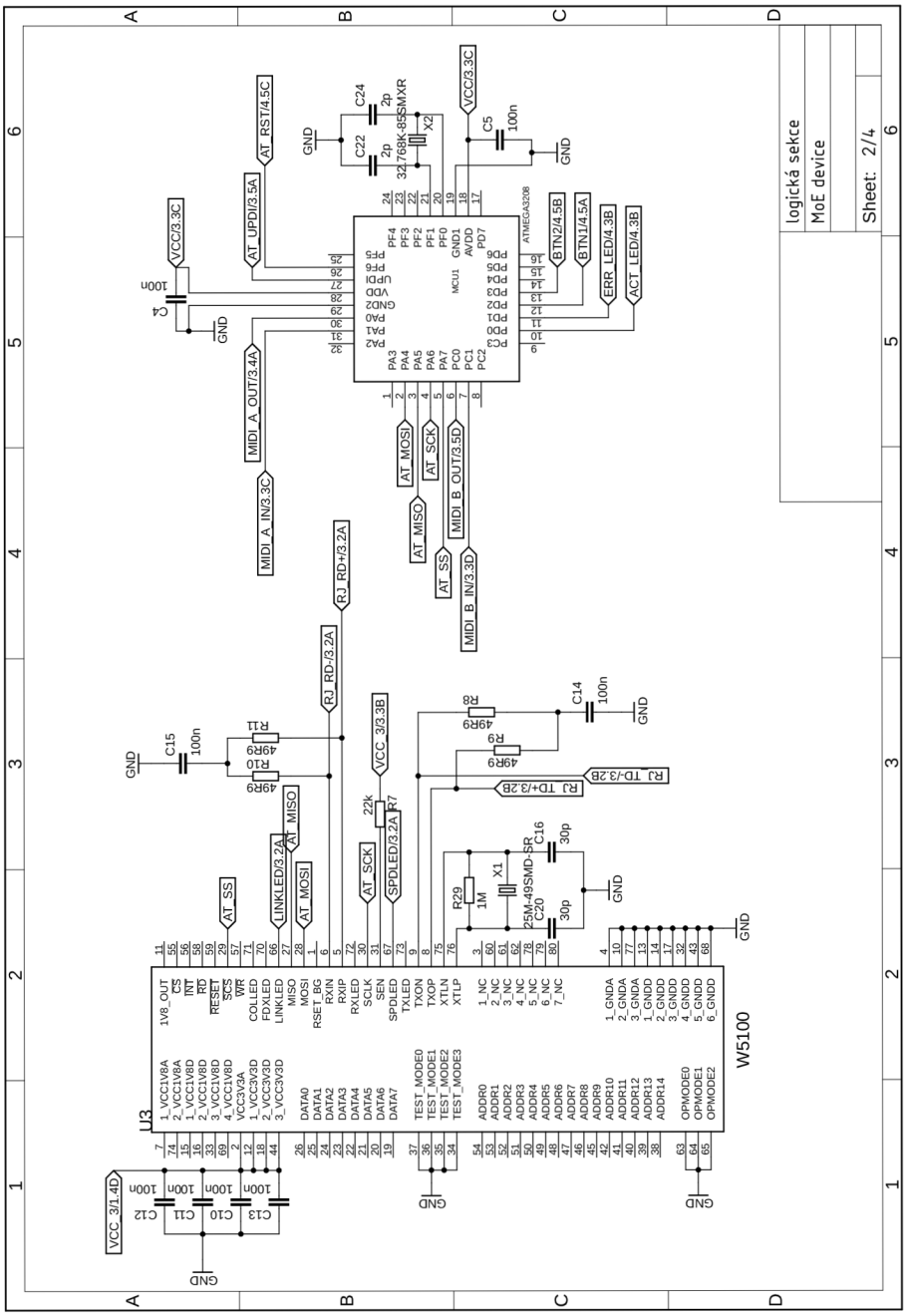
B Hardware přípravku

B.1 Schémata

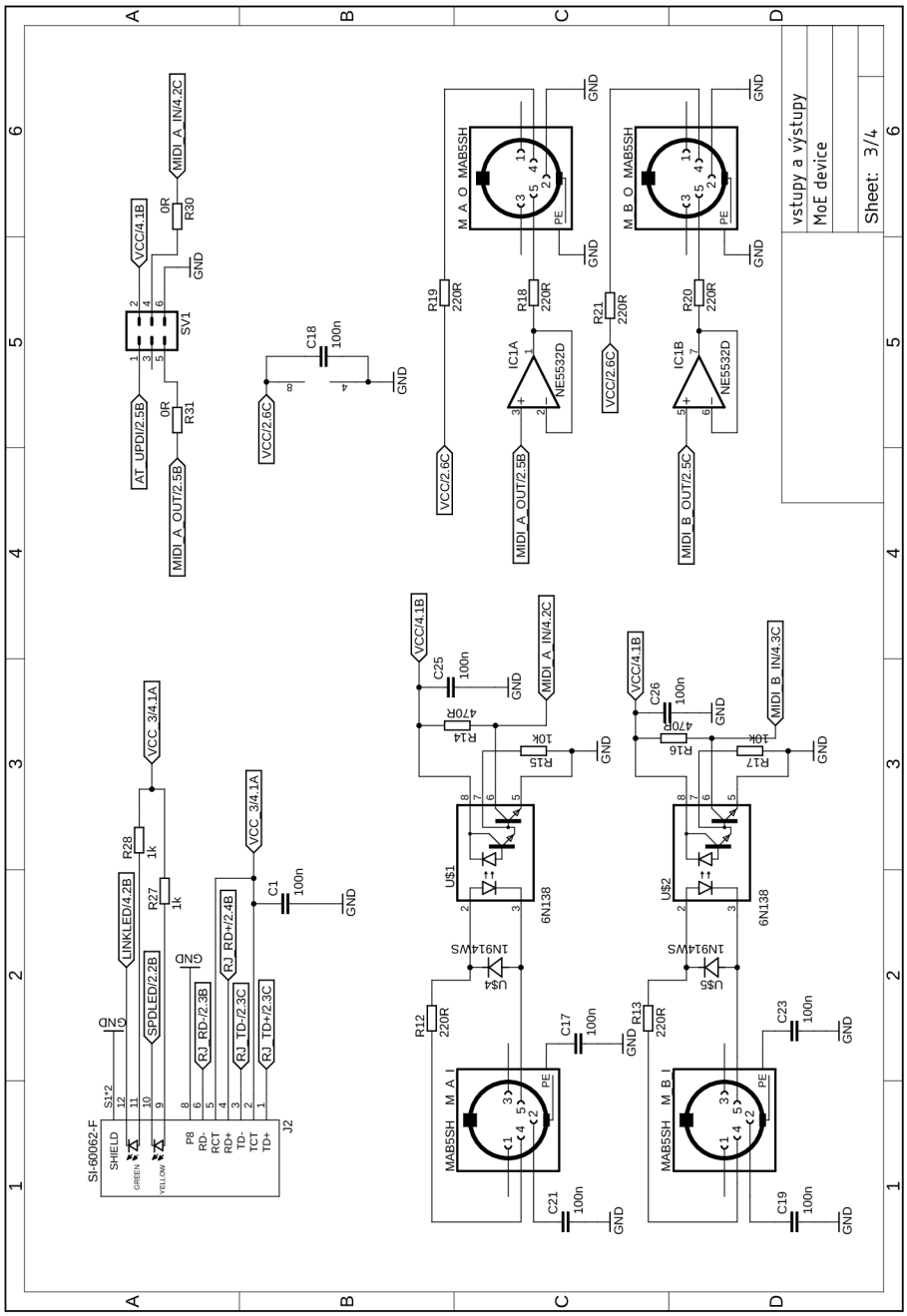
B.1.1 Schéma napájecích obvodů



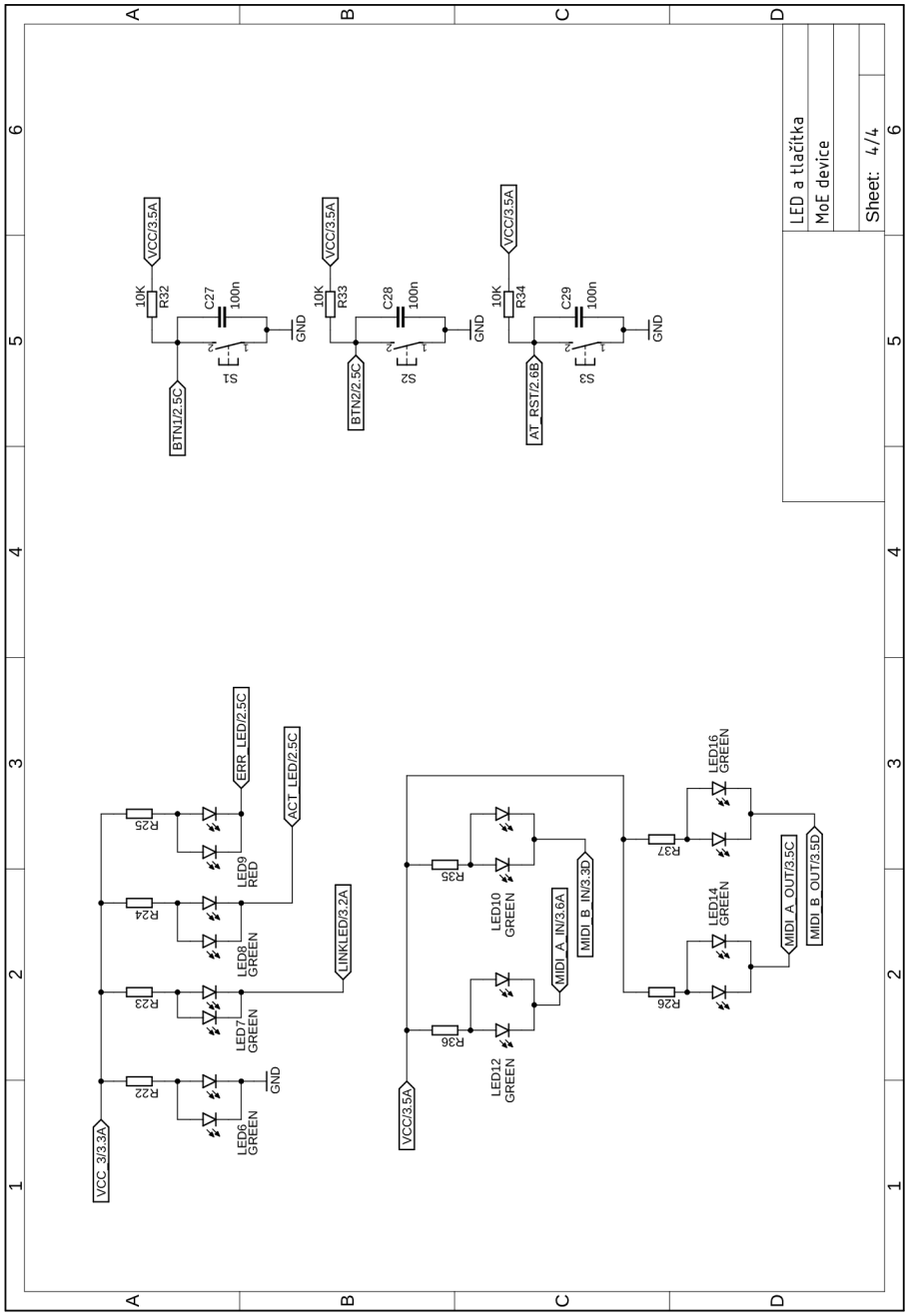
B.1.2 Schéma logické sekce



B.1.3 Schéma vstupů a výstupů

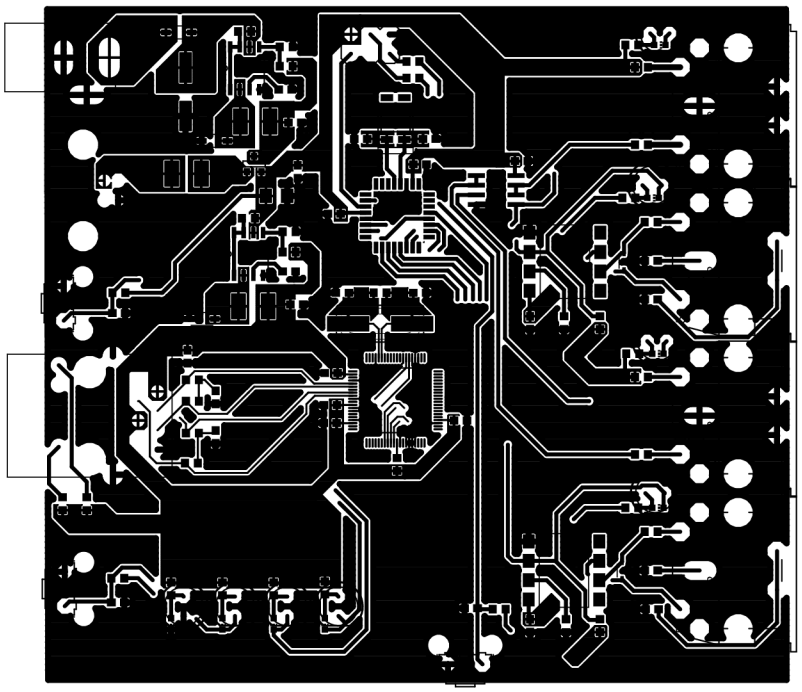


B.1.4 Schéma LED a tlačítek

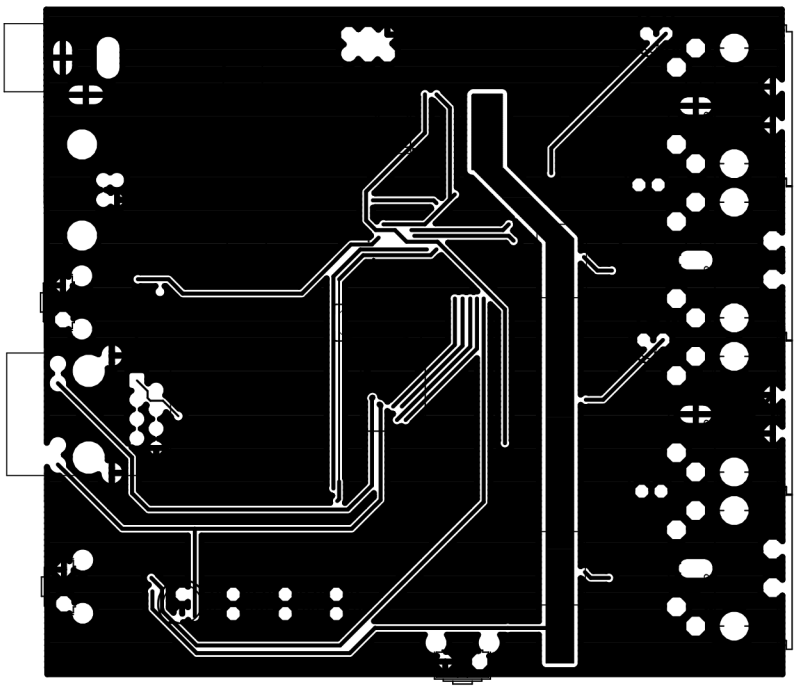


B.2 Deska plošných spojů

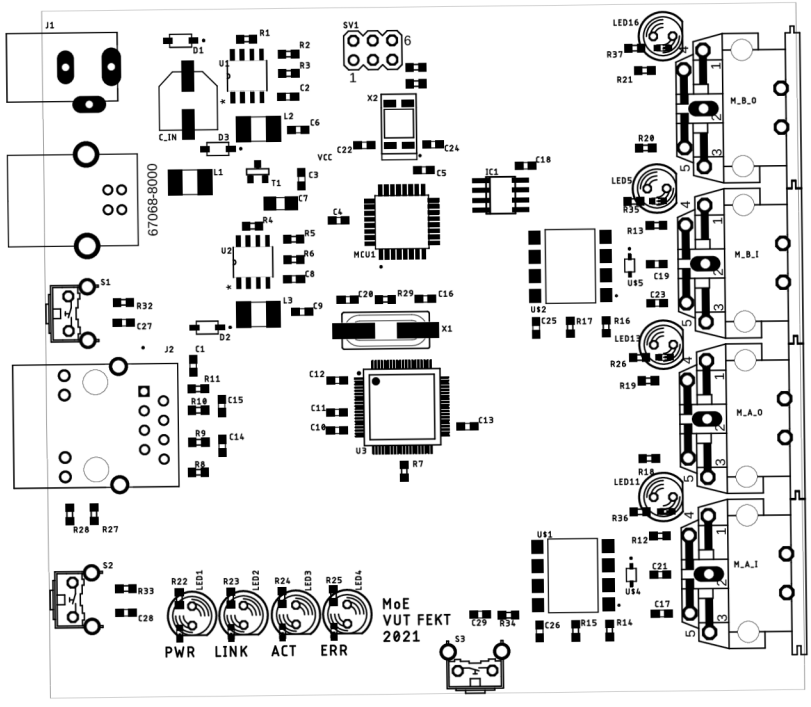
B.2.1 Horní strana DPS



B.2.2 Dolní strana DPS

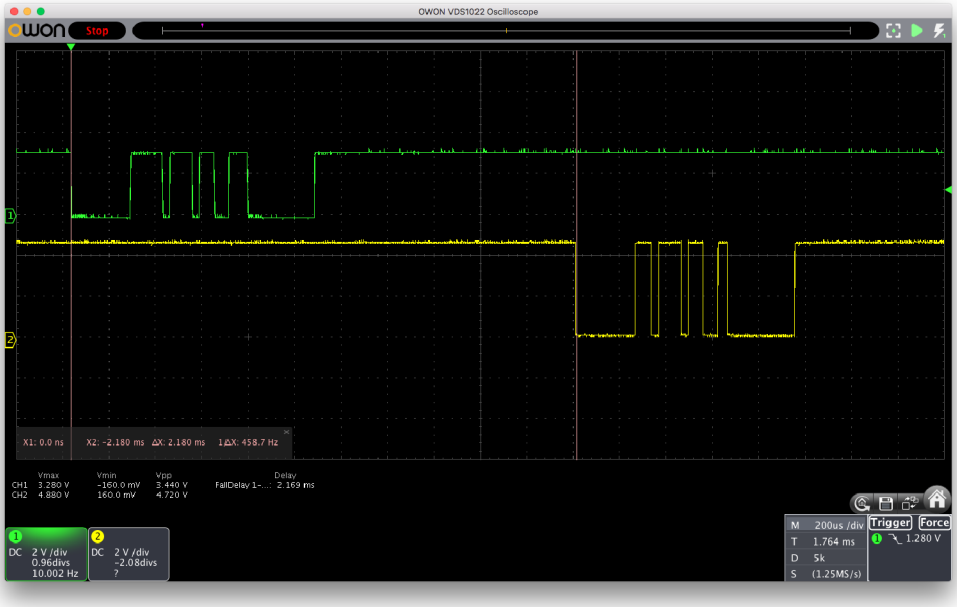


B.2.3 Rozmístění součástek

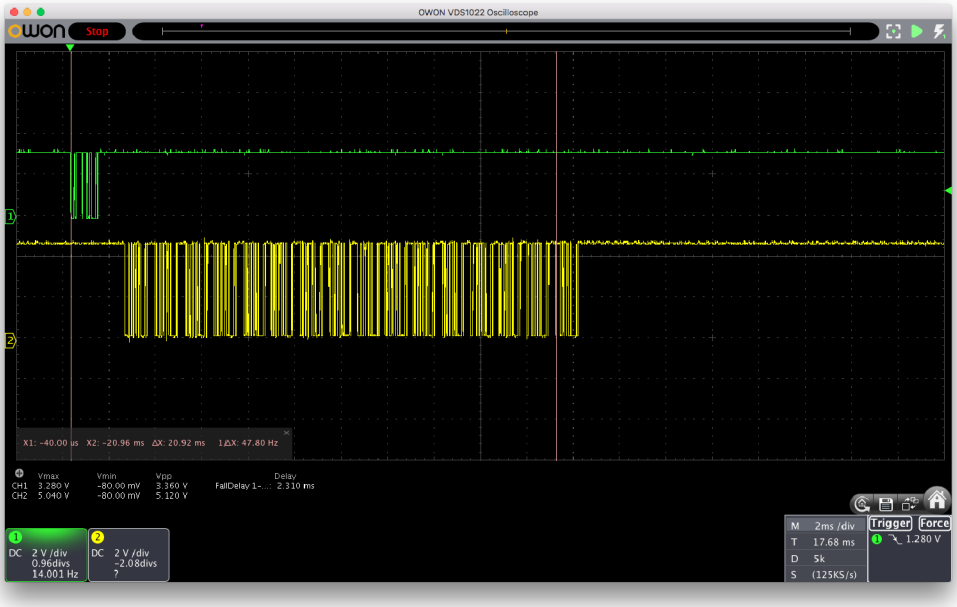


C Záznamy z měření prototypu

C.1 Měření latence



Obr. C.1: Měření latence pro jeden aktivní záznam v databázi spojení.



Obr. C.2: Měření latence pro šestnáct aktivních záznamů v databázi spojení.