

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

SEMESTRÁLNÍ PRÁCE



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

SEMESTRÁLNÍ PRÁCE

SEMESTRAL THESIS

AUTOR PRÁCE

AUTHOR

Vojtěch Lukáš

VEDOUCÍ PRÁCE

SUPERVISOR

Ing. Ondřej Krajsa, Ph.D.

BRNO 2020

Semestrální 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. V rámci semestrální práce proveďte návrh zařízení a testování dílčích částí s využitím vývojového kitu.

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: 2.10.2020

Termín odevzdání: 11.12.2020

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

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

UPOZORNĚNÍ:

Autor semestrální práce nesmí při vytváření semestrální práce porušit autorská práva třetích osob, zejména nesmí zasahovat nedovoleným způsobem do cizích autorských práv osobnostních a musí si být plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č.40/2009 Sb.

PROHLÁŠENÍ

Prohlašuji, že svou semestrální práci na téma „MIDI po Ethernetu“ jsem vypracoval samostatně pod vedením vedoucího semestrální 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 semestrální práce dále prohlašuji, že v souvislosti s vytvořením této semestrální 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

Obsah

Úvod	7
1 Protokol MIDI	8
1.1 Hardwarová vrstva	8
1.1.1 MIDI výstup	8
1.1.2 MIDI vstup	9
1.2 Softwarová vrstva	10
1.2.1 Výjimky	11
2 Návrh přípravku	12
2.1 Hardware	12
2.2 Software	13
2.2.1 Databáze spojení	13
2.2.2 Formát zprávy	14
2.2.3 Síťová vrstva	14
3 Průběh komunikace	15
3.1 Připojení k lokální síti	15
3.2 Upozornění na vlastní přítomnost	15
3.3 Rozpoznání zařízení na síti	16
3.4 Přijetí MIDI zprávy	16
3.5 Přijetí UDP zprávy	16
4 Síťový ovladač	18
Závěr	20
Literatura	21
Seznam symbolů, veličin a zkratk	22
Seznam příloh	23
A Tabulky	24
A.1 MIDI status bajty	24
A.2 MoE značky	25
B Výpisy kódu	26
B.1 Výpis kódu pro příjem a zpracování MIDI zprávy	26

Seznam obrázků

1.1	Schéma MIDI výstupu [1].	9
1.2	Schéma MIDI vstupu [1].	9
1.3	Bity obou druhů MIDI bajtů [1]	10
1.4	Struktura zprávy <i>Změna kontroléru</i> [1].	10
2.1	Schéma prototypu [3]	12
2.2	Kódování zdrojového a cílového kanálu v bajtu <code>srcdstChannel</code>	14
4.1	Konzolová aplikace <i>MoE Matrix Editor</i> – úvodní obrazovka.	18
4.2	Konzolová aplikace <i>MoE Matrix Editor</i> – vytváření spojení.	19
4.3	Konzolová aplikace <i>MoE Matrix Editor</i> – využití vkládacího makra. .	19
C.1	Měření latence pro jeden aktivní záznam v databázi spojení.	27
C.2	Měření latence pro šestnáct aktivních záznamů v databázi spojení. . .	27

Úvod

Protokol MIDI¹ 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. Mimo elementární přesun MIDI zpráv by měl poskytnout i komplexní možnosti směřování jednotlivých vstupních a výstupních MIDI kanálů.

První kapitola je ve stručnosti věnována MIDI protokolu jako takovému. Ve druhé kapitole je nastíněna konstrukce přípravku a v obrysech popsán jeho program. Třetí kapitola nabízí bližší pohled na průběh komunikace a ve čtvrté kapitole je popsána funkčnost síťového editoru, jehož hlavní deviza tkví ve schopnosti ovládat všechny přípravek v síti z jednoho místa – řídicího PC.

¹Musical 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. [1] S nástupem moderních DAW¹ 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 parametrů řídicích konzolí.

1.1 Hardwarová vrstva

Pro přenos MIDI dat se tradičně používá zásuvka a vidlice DIN 5². 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. [1]

Přesná specifikace obvodů pro zpracování příchozích a odchozích MIDI signálů upravuje kapitola *Hardware* z dokumentu [1]. Tato pasáž byla v roce 2014 aktualizována normou [2], která adaptovala protokol i pro zařízení s 3,3V logikou a přidává další prvky pro zamezení zejména RF³ 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 UART⁴ 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. [1]

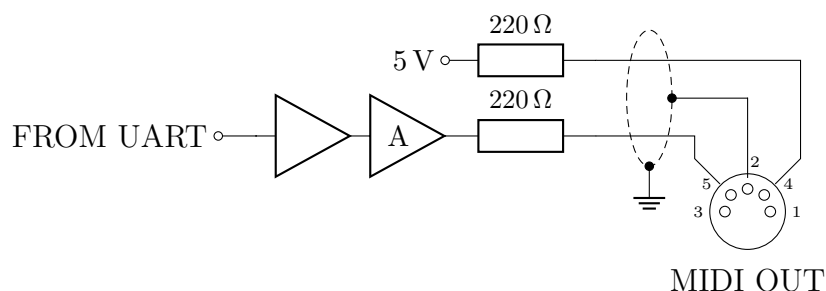
Podle aktualizací normy [2] je možné přidat za každý rezistor feritové jádro pro zamezení vysokofrekvenčních interferencí. V případě použití zařízení s 3,3V logikou jsou pak použity rezistory s menšími hodnotami odporů. Na obr. 1.1 je schéma k nahlédnutí.

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

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

³Radio Frequency

⁴Universal Asynchronous Receiver/Transmitter – univerzální asynchronní přijímač/vysílač.



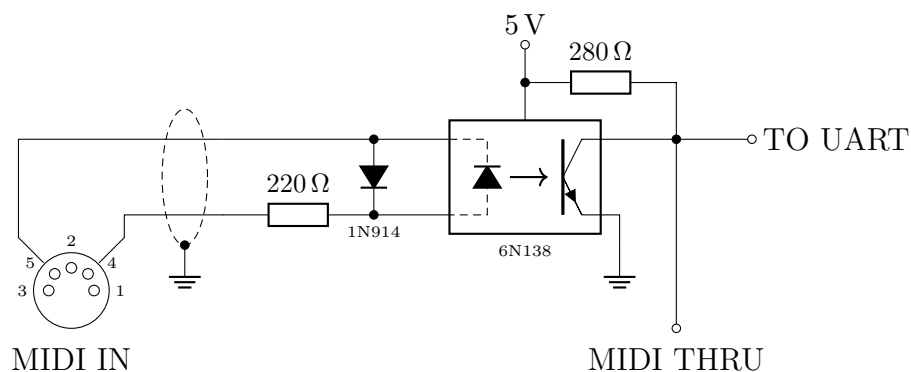
Obr. 1.1: Schéma MIDI výstupu [1].

1.1.2 MIDI vstup

Vstupní port MIDI sběrnice je mnohem složitější. Z obr. 1.2 je patrné, že s cílem eliminovat zemní smyčky, které jsou ve zvukové technice nepřijatelné, je vstup každého MIDI zařízení skrz optočlen galvanicky oddělen. Z téhož důvodu je také pin 2 – na rozdíl od výstupního portu – zapojen naprázdno (podle [1]).

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

Aktualizační norma [2] 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 [1].

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 – LED⁵ 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.

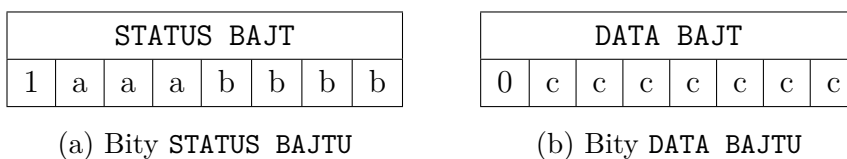
⁵Light-Emitting Diode – Dioda, která vyzařuje viditelné světlo.

1.2 Softwarová vrstva

MIDI protokol přenáší informace (příkazy) po sériové lince přenosovou rychlostí 31250 bps. Komunikace je jednosměrná, pro obousměrnou komunikaci je třeba využít dvou rozhraní (vstupu a výstupu) na všech zúčastněných zařízeních.

Příkazy se skládají z bajtů, jichž rozeznáváme dva typy:

- **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).

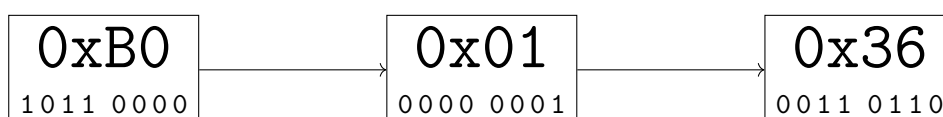


a	bits pro kódování druhu zprávy
b	bits pro kódování kanálu
c	bits pro kódování hodnoty

Obr. 1.3: Bity obou druhů MIDI bajtů [1]

Na obrázku 1.3 je patrná struktura obou bajtů. Za pozornost stojí především jejich MSB⁶: **STATUS BAJT** má MSB vždy s hodnotou 1. Naproti tomu MSB **DATA BAJTU** je vždy nulový.

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* [1].

Z prvního bajtu zprávy z obr. 1.4 lze dekodovat 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 semestrálního projektu bude nutné, aby každý přípravek dokázal rozeznat kanál k němu přichází MIDI zprávy, popř. jej pro další přenos pozměnil. **DATA BAJTY** budou jen „kopírovány“ a nebude se do nich nijak zasahováno.

⁶Most Significant Bit – Nejvýznamnější bit (většinou v bajtu).

⁷0000₂ ~ kanál 1 ... 1111₂ ~ kanál 16.

1.2.1 Výjimky

Je třeba alespoň okrajově zmínit výjimečné stavy, které v rámci MIDI protokolu mohou nastat. V této fázi semestrální práce tyto výjimky zatím nejsou ošetřeny.

Running Status

Pracuje-li MIDI vysílač v tomto stavu, neposílá **STATUS BAJT** v opakující se zprávě stejného druhu. Příjímač 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 kontroly* kontinuálních ovladačů. [1]

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. Příjímač by měl být připraven i na to, že je obdrží uprostřed standardní zprávy nebo SysEx zprávy. [1]

Zprávy SysEx

SysEx (*System Exclusive*) 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**), MSC⁸ apod. [1]

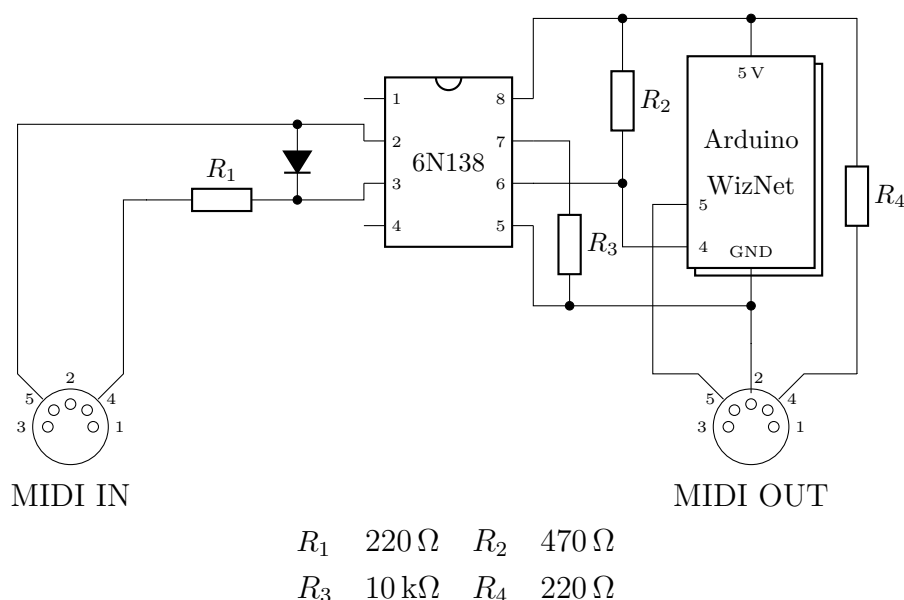
⁸MIDI Show Control – subprotokol pro ovládání scénické techniky, zejména pomocí příkazů GO, STOP, popř. RESUME. [1]

2 Návrh přípravku

V rámci semestrální práce byl vytvořen prototyp zařízení na platformě Arduino UNO, která disponuje čipem ATMEL ATmega 328P. Arduino sekunduje Ethernet Shield V1, který je osazen čipem WizNet W5100.

Tento přípravek umožňuje výměnu MIDI příkazů přes lokální síť, využívá první čtyři vrstvy ISO/OSI modelu. Zároveň, pomocí síťového editoru (kapitola 4), poskytuje pokročilé možnosti směrování jednotlivých MIDI kanálů napříč dalšími přípravky v síti.

2.1 Hardware



Obr. 2.1: Schéma prototypu [3]

Na obr. 2.1 je patrné schéma přípravku. Je využito standardní zapojení MIDI vstupu a výstupu v souladu se schématy na obr. 1.1 a 1.2. Jak již bylo řečeno v kapitole 1, MIDI využívá pro komunikaci sériovou linku. V tomto případě je vstupní linka zapojena k Arduino na pinu 4 a výstupní k pinu 5¹.

WizNet Shield zprostředkovává přípravku standardní ethernetové rozhraní v podobě zásuvky typu RJ45, ovládacího mikrokontroléru a také připravených softwarových knihoven.

¹Arduino umožňuje softwarové přepnutí pinů z režimu výstupu na režim vstup a naopak.

Přípravek je v této fázi vývoje napájen prostřednictvím USB sběrnice Arduina napětím 5 V. Všechny součástky jsou zatím vsazeny do nepájivého kontaktního pole. Fotografie prototypu jsou přiloženy k nahlédnutí v **TODO!**

2.2 Software

Software přípravku byl vyvíjen za účelem maximalizace variability směřování MIDI příkazů. Na funkční rovině je inspirován systémem *Dante*, který se používá pro přenos zvukového signálu přes počítačovou síť. *Dante* umožňuje pokročilé směřování signálů napříč síťovými zařízeními, ovládané pomocí přehledné matice na ovládacím PC. Do této fáze by měl být vyvinut i systém MoE².

Přípravek pracuje s dvěma typy zpráv: Příchozí, odchozí MIDI zprávy, které se týkají hardwarové sběrnice (tedy přímo pinů 4 a 5 z obr. 2.1) – dále jen „MIDI zprávy“ – a příchozí, odchozí UDP zprávy, které se týkají pouze ethernetového rozhraní zprostředkovaném WizNet Shieldem – dále jen „UDP zprávy“.

2.2.1 Databáze spojení

S cílem docílit již zmiňované velké variability ve směřování zpráv napříč přípravku byla do zdrojového kódu implementována tzv. databáze spojení `subscriptions[]`:

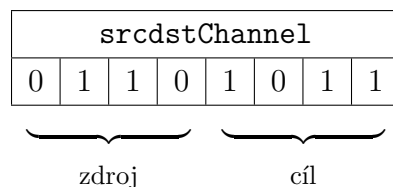
```
typedef struct subscription
{
    byte srcdstChannel;
    byte dstIPnib;
} _subscriptions[MAX_SUBS];
```

Toto pole struktur `subscription` je pro MoE protokol zásadní. Jedno „spojení“, tedy jeden prvek v tohoto pole, propojuje jeden MIDI kanál lokálního zařízení s jedním MIDI kanálem cílového zařízení. Toto pole si lze představit jako virtuální patchbay. Jeden patch-kabel odpovídá jednomu prvku v poli (záznamu v databázi).

Každý záznam je tvořen dvěma bajty:

1. `srcdstChannel` v sobě kóduje zdrojový MIDI kanál, který se týká lokálního zařízení, a cílový MIDI kanál – ten se týká zařízení cílového. „Kódování“ je uskutečněno způsobem vyobrazeným na obr. 2.2. Jeden kanál je vyjádřen pomocí čtyř bitů (viz kapitolu 1.2), jeden bajt tedy „pojme“ vyjádření přesně dvou kanálů.
2. `dstIPnib` vyjadřuje poslední bajt IP adresy cílového zařízení (viz též sekci 2.2.3)

²MIDI over Ethernet – MIDI po Ethernetu



Obr. 2.2: Kódování zdrojového a cílového kanálu v bajtu `srcdstChannel`

Tato databáze je při každém přijetí MIDI zprávy procházena. Pro přesný průběh programu viz 3.4 resp. B.1

2.2.2 Formát zprávy

Pro optimální komunikaci zařízení mezi sebou bylo nutné zapouzdřit MIDI příkazy do MoE datagramu. Tento datagram se skládá ze čtyř bajtů, z čehož první je vyhrazen MoE značce a zbylé tři samotné MIDI zprávě nebo jiným datům. Tabulku všech dosavadně implementovaných MoE značek lze nalézt v příloze A.2. Podle hodnoty této značky se přípravek k dané MoE zprávě zachová. Ku příkladu příchozí zpráva

0xA3, 0x90, 0x45, 0x7F,

bude interpretována jako okamžité odeslání MIDI zprávy na sériovou sběrnici. Na zprávu

0x08, 0x08, 0x08, 0x08,

bude přípravek zase reagovat zasláním své databáze spojení na adresu odesílatele atp.

2.2.3 Síťová vrstva

Tento MoE datagram je následně zapouzdřen do UDP³ datagramu. Pro síťovou komunikaci byl vybrán protokol UDP z důvodu jeho „peer2peer“ architektury, která je k tomuto projektu mnohem vhodnější než „client-server“, kterou poskytuje protokol TCP. Komunikace probíhá na portu 50 000. MoE zařízení jsou schopna připojit se do lokální sítě s pevně nastavenou IP adresou⁴, nebo požádat o přidělení IP adresy DHCP server. Tato síť však musí mít masku 255.255.255.0 – zařízení se z úsporných důvodů navzájem rozlišují *pouze* posledním bajtem IP adresy (jak je popsáno v sekci 2.2.1).

³User Datagram Protocol – jednoduchý síťový protokol, který umožňuje výměnu zpráv „host-to-host“ [4].

⁴Tato IP adresa je zatím nastavitelná pouze ve zdrojovém kódu.

3 Průběh komunikace

V této kapitole 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.

3.1 Připojení k lokální síti

Každému přípravku byla do prvních šesti bajtů EEPROM paměti uložena unikátní MAC adresa. Při zapnutí a následné inicializaci je adresa načtena a přiřazena. Nejprve proběhne pokus o získání IP adresy pomocí DHCP serveru sítě. V případě neúspěchu použije přípravek napevno nastavenou IP adresu.

Díky knihovně `EthernetUdp.h` jsou výše zmíněné operace otázkou pouze několika málo řádků:

```
if (Ethernet.begin())
{
    //DHCP server zdárně přidělil zařízení IP adresu
    Serial.println(Ethernet.localIP());
}
else
{
    Ethernet.begin(_myMac, _userIP);
    //Přidělení IP adresy napevno
    Serial.println(Ethernet.localIP());
}
```

3.2 Upozornění na vlastní přítomnost

Okamžitě po zdárném připojení k lokální síti je vyslána zpráva `beacon`, jejíž účel je upozornit všechna zařízení v lokální síti na vlastní přítomnost. Zpráva se skládá ze čtyř bajtů:

`0xFF, 0xFF, 0xFF, 0xFF`

a je poslána na broadcast adresu sítě.

```
_broadcastIP = Ethernet.localIP();
_broadcastIP[3] = 255;
EthernetUdp eUDP;
eUDP.begin(MOE_PORT);
```



```
eUDP.beginPacket(_broadcastIP, MOE_PORT);  
eUDP.write(_beacon, sizeof(_beacon));  
eUDP.endPacket();
```

3.3 Rozpoznání zařízení na síti

V případě přijetí zprávy **beacon** je rozpoznána IP adresa odesílatele a vytvořen nový záznam do databáze spojení **subscriptions**. V této fázi projektu přípravek napevno přiřazuje kanál 1 lokální příchozí MIDI sběrnice kanálu 1 výstupní MIDI sběrnice cílového zařízení. Unikátnost této zprávy v rámci běhu programu má však velice negativní efekt: Zařízení, které je do sítě připojeno jako poslední nepřijme žádnou zprávu **beacon**, do své databáze si tedy žádné spojení nepřidá. Z toho důvodu byl vyvinut síťový editor, který umožňuje přidávat a mazat záznamy do databází spojení všech přípravků kdykoliv za běhu programu.

3.4 Přijetí MIDI zprávy

Při přijímání MIDI zprávy je kontrolován vstupní buffer sériové sběrnice **pinu 5**. Pokud jsou v bufferu přesně tři bajty, jsou postupně přečteny a uloženy do paměti. V dalším kroku je nutné dekodovat kanál této MIDI zprávy z **DATA BAJTU** – tuto „extrakci“ lze vyřešit pomocí bitového maskování. Takto získaný bajt je pak porovnáván s prvním nibblem bajtu **srcdstChannel** každého záznamu databáze **subscriptions**. Dojde-li ke shodě, začíná konstrukce odchozí zprávy podle dalších informací v odpovídajícím záznamu databáze spojení. V tomto bodě je důležité zmínit, že druhý nibble **DATA BAJTU** – kanál – je změněn, aby bylo zaručeno, že z MIDI sběrnice cílového zařízení budou proudit zprávy na kanálech odpovídajících databázi spojení. Výpis kódu je k dispozici jako příloha B.1

3.5 Přijetí UDP zprávy

Ve okamžiku příjmu UDP zprávy je nejprve nutné rozlišit její první bajt – ten určuje účel zprávy. Toto se uskutečňuje pomocí jednoduchého větvení **switch**.

```
if (eUDP.parsePacket())  
{  
    eUDP.readByte(_incomingUDP. 4);  
    switch (_incomingUDP[0]) {...}  
}
```

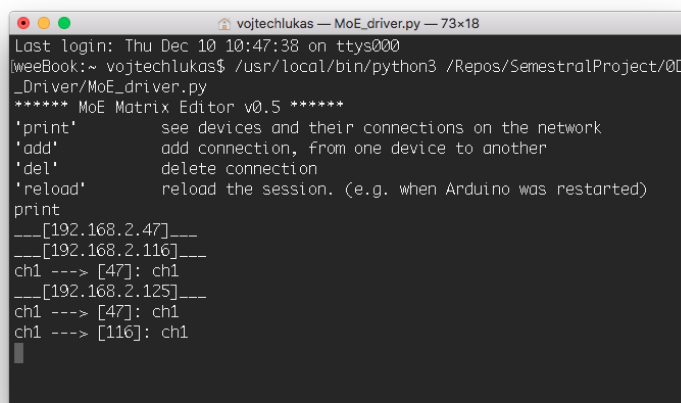
Pro příklad jsou zde uvedeny dvě reakce na dané druhy příchozích zpráv v souladu s tabulkou A.2. Tento kód je obsahem větvení **switch** z předchozího odstavce.

```
//...
case 0x08:           //dotaz řídicího PC na záznamy
                    //v databázi spojení
    sendSubs(eUDP.remoteIP());
    break;

case 0xA3:           //Příjem třibajtové MIDI zprávy
                    //a zápis na sběrnici
    midiSerial.write(_incomingUDP[1]);
    midiSerial.write(_incomingUDP[2]);
    midiSerial.write(_incomingUDP[3]);
    break;
//...
```

4 Síťový ovladač

Pro centrální dálkové ovládání databází spojení na jednotlivých přípravcích v síti byl vytvořen jednoduchý program v jazyce Python, který je nutné spustit na PC, který je připojen ve stejné síti jako všechny přípravky. V době odevzdávání této semestrální práce existuje ovladač pouze jako konzolový skript, v plánu je však vytvořit i uživatelsky mnohem přívětivější aplikaci s grafickým rozhraním.



```
vojtechlukas - MoE_driver.py - 73x18
Last login: Thu Dec 10 10:47:38 on ttys000
weeBook:~ vojtechlukas$ /usr/local/bin/python3 /Repos/SemestralProject/001/
_Driver/MoE_driver.py
***** MoE Matrix Editor v0.5 *****
'print'      see devices and their connections on the network
'add'        add connection, from one device to another
'del'        delete connection
'reload'     reload the session. (e.g. when Arduino was restarted)
print
---[192.168.2.47]---
---[192.168.2.116]---
ch1 ---> [47]: ch1
---[192.168.2.125]---
ch1 ---> [47]: ch1
ch1 ---> [116]: ch1
```

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

Za běhu programu má uživatel na výběr ze čtyř příkazů.

<code>print</code>	Na konzoli vytiskne všechna zařízení v síti a jejich aktuální spojení.
<code>add</code>	Přidá vybranému zařízení záznam do databáze spojení <code>subscriptions</code> .
<code>del</code>	Vymaže vybranému zařízení záznam z databáze spojení.
<code>reload</code>	Vyžádá si po všech zařízeních na síti aktualizaci jejich databází spojení.

Pro zjednodušení je přidáno „makro“, které se spouští 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 přijímal všech šestnáct kanálů zařízení cílového. Je třeba dodat, že toto makro reálně využitelné není, velice však usnadní práci při testovacích a měřicích aktivitách. Analogicky k tomuto makru funduje i příkaz `del` s parametrem 255 u hodnoty `destinationChannel`.

```
vojtechlukas — MoE_driver.py — 73x18
___[192.168.2.116]___
ch1 ----> [47]: ch1
___[192.168.2.125]___
ch1 ----> [47]: ch1
ch1 ----> [116]: ch1
add
IP address: 192.168.2.47
sourceChannel destinationIP destinationChannel: 1 116 5
*** added ***
print
___[192.168.2.47]___
ch1 ----> [116]: ch5
___[192.168.2.116]___
ch1 ----> [47]: ch1
___[192.168.2.125]___
ch1 ----> [47]: ch1
ch1 ----> [116]: ch1
```

Obr. 4.2: Konzolová aplikace *MoE Matrix Editor* – vytváření spojení.

```
vojtechlukas — MoE_driver.py — 73x27
add
IP address: 192.168.2.116
sourceChannel destinationIP destinationChannel: 1 47 255
*** added ***
print
___[192.168.2.47]___
ch1 ----> [116]: ch5
___[192.168.2.125]___
ch1 ----> [47]: ch1
ch1 ----> [116]: ch1
___[192.168.2.116]___
ch1 ----> [47]: ch1
ch1 ----> [47]: ch2
ch1 ----> [47]: ch3
ch1 ----> [47]: ch4
ch1 ----> [47]: ch5
ch1 ----> [47]: ch6
ch1 ----> [47]: ch7
ch1 ----> [47]: ch8
ch1 ----> [47]: ch9
ch1 ----> [47]: ch10
ch1 ----> [47]: ch11
ch1 ----> [47]: ch12
ch1 ----> [47]: ch13
ch1 ----> [47]: ch14
ch1 ----> [47]: ch15
ch1 ----> [47]: ch16
```

Obr. 4.3: Konzolová aplikace *MoE Matrix Editor* – využití vkládacího makra.

Závěr

Výstupem této semestrální práce je funkční přípravek, který disponuje ethernetovým a vstupním i výstupním MIDI rozhraním. 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 na dálku pomocí programu *MoE Matrix Editor*.

Stále však existuje mnoho prostoru pro optimalizaci. Pokud přípravek obdrží více než dvě MIDI zprávy těsně za sebou, přestane přijímat jakékoliv další bajty a jeho funkčnost je do restartu přerušena. Absence implementace ošetření výjimek (dle kapitoly 1.2.1) zase prozatím brání využití přípravků tam, kde je příjem zpráv v módu Running Status nebo SysEx standardem.

Jako uspokojivé lze však vnímat dosažené hodnoty latence mezi přijetím MIDI zprávy na sériové sběrnici jednoho zařízení a odeslání též zprávy na sériovou sběrnici zařízení druhého. Komunikace mezi samotnými zařízeními samozřejmě probíhala 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] THE MIDI MANUFACTURERS ASSOCIATION. *MIDI 1.0 Detailed Specification*. Document Version 4.2.1. Los Angeles, CA: MMA, 1996.
- [2] MMA TECHNICAL STANDARDS BOARD/AMEI MIDI COMMITTEE. *MIDI 1.0 Electrical Specification Update*. 2014.
- [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] MNEIMNEH, Saad. *Computer Networks UDP and TCP*. Hunter College of CUNY. New York. 2008.

Seznam symbolů, veličin a zkratek

MIDI	Musical Instrument Digital Interface – digitální rozhraní hudebního nástroje
DAW	Digital Audio Workstation – digitální pracovní stanice pro náběr a úpravu vícestopého záznamu.
USB	Universal Serial Bus – univerzální sériová sběrnice pro komunikace s periferními zařízeními.
RF	Radio Frequency
UART	Universal Asynchronous Receiver/Transmitter – univerzální asynchronní přijímač/vysílač.
LED	Light-Emitting Diode – Dioda, která vyzařuje viditelné světlo.
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. [1]
MoE	MIDI over Ethernet – MIDI po Ethernetu
UDP	User Datagram Protocol – jednoduchý síťový protokol, který umožňuje výměnu zpráv „host-to-host“ [4].

Seznam příloh

A	Tabulky	24
A.1	MIDI status bajty	24
A.2	MoE značky	25
B	Výpisy kódu	26
B.1	Výpis kódu pro příjem a zpracování MIDI zprávy	26
C	Záznamy z měření latence	27

A Tabulky

A.1 MIDI status bajty

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

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	0xA n	1010 nnnn
Control Change	Změna kontroléru	0xB n	1011 nnnn
Program Change	Změna programu	0xC n	1100 nnnn
Channel Pressure	Monofonický tlakový ovladač	0xD n	1101 nnnn
Pitch Bend	Ohyb výšky tónu	0xE n	1110 nnnn

A.2 MoE značky

Tab. A.2: Tabulka prvních bajtů MoE zpráv.

Bajt	Popis
⋮	
0x08	Broadcast zpráva od řídicího PC, která sonduje účastníky sítě
⋮	
0x0E	Unicast zpráva od řídicího PC, která zařízení maže záznam z databáze subscriptions .
0x0F	Unicast zpráva od řídicího 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
⋮	
0xA3	Třibajtová MIDI zpráva
⋮	
0xFF	Zpráva beacon sloužící pro upozornění na vlastní přítomnost

B Výpisy kódu

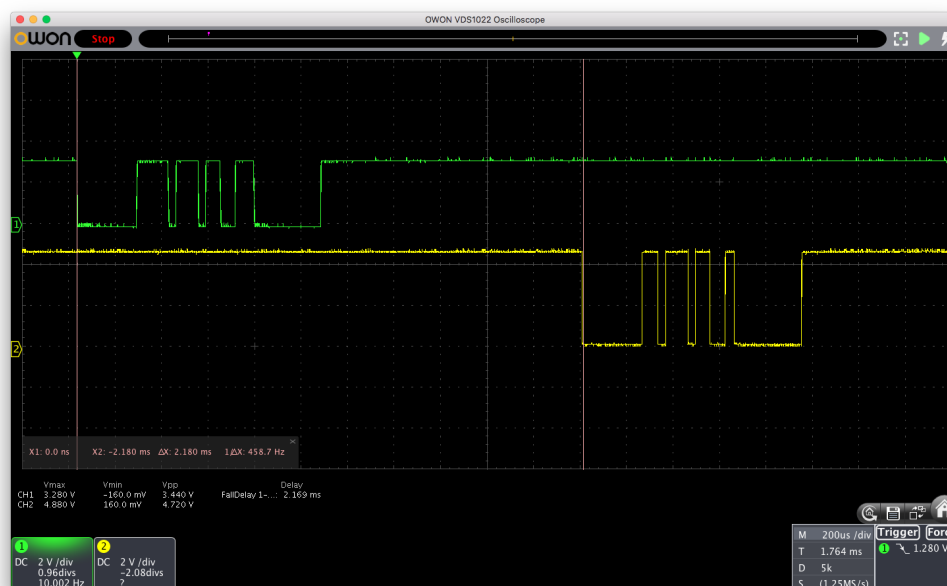
B.1 Výpis kódu pro příjem a zpracování MIDI zprávy

Výpis B.1: Výpis kódu pro příjem a zpracování MIDI zprávy

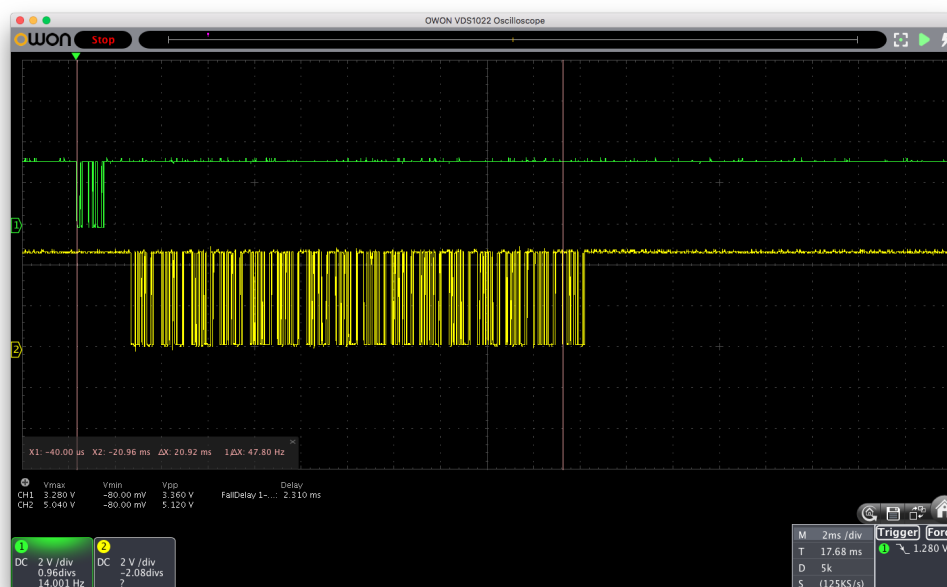
```
void handleMIDI()
{
    if (midiSerial.available() == 3)
    {
        _data0 = midiSerial.read();
        _data1 = midiSerial.read();
        _data2 = midiSerial.read();
        sendUDP(_data0, _data1, _data2);
    }
}

void sendUDP(byte data0, byte data1, byte data2)
{
    byte srcCh = data0 & 0x0F;
    for (byte i = 0; i < MAX_SUBS; i++)
    {
        if (srcCh == (_subscriptions[i].srcdstChannel
                    & 0xF0) >> 4)
        {
            data0 = data0 & 0xF0;
            data0 = data0 | (_subscriptions[i].srcdstChannel
                            & 0x0F);
            IPAddress destinationIP = Ethernet.localIP();
            destinationIP[3] = _subscriptions[i].dstIPnib;
            eUDP.beginPacket(destinationIP, MOE_PORT);
            eUDP.write(0xA3);
            eUDP.write(data0);
            eUDP.write(data1);
            eUDP.write(data2);
            eUDP.endPacket();
        }
    }
}
```

C Záznamy z 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.