ZÁPADOČESKÁ UNIVERZITA V PLZNI
FAKULTA ELEKTROTECHNICKÁ

KATEDRA APLIKOVANÉ ELEKTRONIKY A TELEKOMUNIKACÍ

DIPLOMOVÁ PRÁCE

Distribuovaný zvukový modul

Bc. Ondřej Vavroch **2018/2019**

Originál (kopie) zadání BP/DP

Abstrakt

Diplomová práce je zaměřena na realizaci distribuovaného zvukového modulu na platformě STM32F429ZI, schopné přijmutí, uložení a následné přehrání zvukového souboru. Pro komunikaci je použita technologie Ethernet s implementovanými protokoly TCP a UDP. Jako úložné médium je použita mikroSD karta. Přenos zvuku do audio kodeku je realizován protokolem I^2S. V práci je představené kompletní zařízení s postupem realizace.

Klíčová slova

Mikrokontrolér, mikroSD karta, Ethernet, Serial Audio Interface, Inter-IC Sound,STM32F429

Abstract

Focus of this master's thesis is realisation of distributed audio module on STM32F429ZI platform capable of receiving, saving and playing audio files. Ethernet technology with implemented TCP and UDP protocols is used for comunication. For data storage microSD card is used. Audio is transmitted through I^2S protocol from microcontroller to audio codec. Complete work procedure is described in thesis.

Key words

Microcontroller, MicroSD card, Ethernet, Serial Audio Interface, Inter-IC Sound, STM32F429

Prohlášení

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

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

.....

podpis

V Plzni dne 27.05.2019 Bc. Ondřej Vavroch

Poděkování

Tímto bych rád poděkoval vedoucímu diplomové práce panu Ing. Petr Kristovi, Ph.D. za cenné profesionální rady, připomínky a metodické vedení práce.

Obsah

Úvod

Diplomová práce se zabývá realizací zařízení pro domácí použití pro jednoduché a komfortní přehrávání souborů, obsahující zvukový záznam. Volba komponentů tohoto zařízení je podřízena minimálním nárokům na změnu infrastruktury v používaných technologiích v domácnosti. Cílem této práce je vytvořit kompaktní modul, do kterého je možno přes lokální síť poslat a uložit zvukový soubor a ten následně kdykoliv přehrát. Jako základ tohoto zvukového modulu byl zvolen mikroprocesor STM32F429. Tento mikroprocesor umožňuje jak připojení ke standardu Ethernet, tak komunikaci se zvukovým kodekem CS4344-CZZ a ukládání přijatých souborů z lokální sítě na externí úložiště. Jako úložiště zvukových souborů byla vybrána flashová paměťová karta mikroSD pro její velkou kapacitu, přenosovou rychlost a přenositelnost mezi jinými systémy.

K práci je přiložen DVD disk, na kterém jsou doplňující materiály k diplomové práci.

Seznam symbolů a zkratek

ARP.....Address Resolution Protocol

ASCII.....American Standard Code for Information Interchange

CSMA.....Carrier Sense Multiple Access

CD.....Colision Detected

DPS..... Deska plošných spojů

DVD..... Digital Versatile Disc

DZM.....Distribuovaný Zvukový Modul

FATFS..... File Allocation Table File System

I^2S..... Inter-IC Sound

ICMP.....Internet Control Message Protocol

IGMP.....Internet Group Management Protocol

IP..... Internet Protocol

LPCM.....Linear Pulse Code Modulation

LRCK.....Left Right Clock

MAC..... Media Access Protocol

MCLK.....Master Clock

MCU..... Microcontroller Unit

MII.....Media Independent Interface

SAI..... Serial Audio Interface

SD..... Secure Digital

SDIO..... Secure Digital Input Output

SPI.....Serial Peripheral Interface

RMII.....Reduced Media Independent Interface

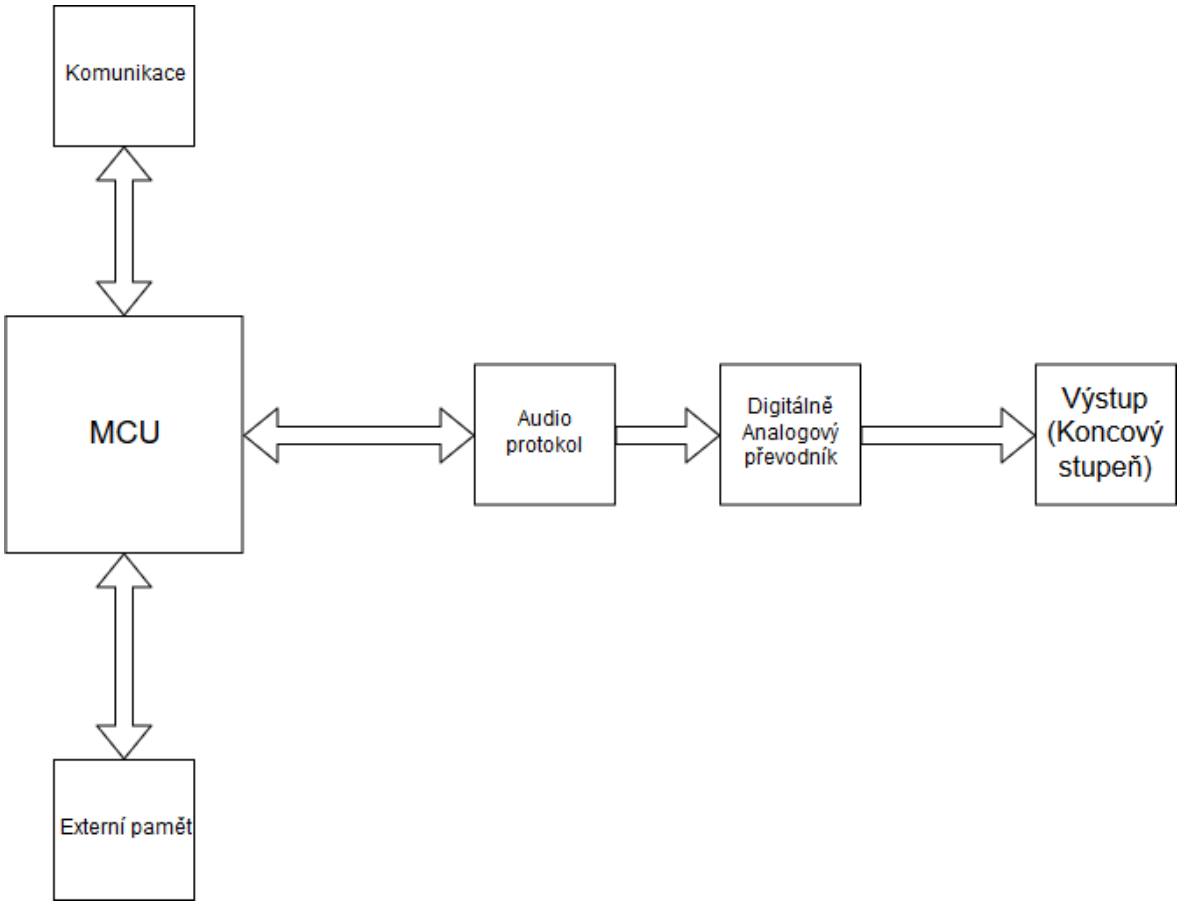
TCP..... Transmission Control Protocol

TDM.....Time Division Multiplexed

UDP..... User Datagram Protocol

1 Výběr komponent zařízení

Základní krok při návrhu zařízení je stanovení požadavků na jednotlivé komponenty daného zařízení. Pro funkci tohoto zařízení je nutná komunikace s poskytovatelem zvukových souborů, externím úložištěm pro ukládání přijatých souborů a komunikace s D/A převodníkem pro možnost přehrávání těchto souborů.



Obr. 1: Blokové schéma zpracování

Všechny tyto požadavky jsou promítnuty do zjednodušeného blokového schématu vyobrazeného na obrázku č.1. Následně je nutné tyto obecné bloky zkonkretizovat na reálné komponenty.

1.1 Výběr komunikace pro přenos dat a příkazů

Při výběru komunikace pro přenos dat a příkazů, jsem si stanovil několik podmínek:

- Z důvodu přenosu velkých souborů je nutná dostatečná přenosová rychlost.
- Při snaze o nízkou cenu finálního výrobku, nemělo by zařízení vyžadovat zavedení nové infrastruktury.
- Nutnost fyzické manipulace uživatele se zařízením, kromě počátečního zavedení, je nepřípustná.

Průmyslové sběrnice jako je CAN nebo RS485 jsou pro komunikaci nevhodné pro jejich minimální rozšíření v prostředí použití tohoto zařízení a jejich malé přenosové rychlosti. Jako vhodné se jeví použití USB nebo Ethernet technologií. Ethernet oproti USB nabízí několik výhod jako: možnosti přístupu z několika zařízení bez nutnosti přepojení, větší dosah a existence již hotového rozvodu ve většině domů. Díky těmto vlastnostem byla zvolena pro přenos dat a příkazů technologie Ethernet.

Technologie Ethernet zajišťuje pouze fyzickou a linkovou vrstvu. Běžně je vyšší, síťová vrstva, zprostředkovaná IP protokolem, se kterým pracují všechny moderní síťové prvky. Z tohoto důvodu považuji za vhodné držet se zavedených standardů a tento protokol implementovat spolu s protokoly TCP a UDP, používanými v protokolu IP.

1.2 Výběr protokolu pro přenos zvuku[13]

Při výběru protokolu pro přenos zvuku bylo nutné zvážit složitost, cenu komponentů a kompatibilitu mikrokontrolerů.

Byl vybrán protokol I^2S pro jeho jednoduchost ovládání, rozšířenost a kompatibilitu. Nebyla potřeba posílání dat do více audio kodeků současně. Protože jsou data posílány v LPCM formátu není potřeba dalšího konvertování dat. Formát LPCM je více popsán v kapitole 3.4.

Více informací o protokolech pro přenos zvuku je možné nalézt ve 4.kapitole.

1.3 Výběr externí paměti

Externí paměť je určena k trvalému ukládání dat, které se po vypnutí napájení neztrácí. Lze ji rozdělit na stálou a výměnnou. Pro dané zvukové zařízení byla zvolena paměťová karta mikroSD. Jde o velice rozšířenou flashovou paměťovou kartu, která je podporována v různých OS, mobilních zařízeních a přehrávačích hudby.

Výhody paměťové mikroSD karty:

- Velká kapacita dat
- Přijatelná cena
- Snadná implementace
- Rozšířená podpora ze strany mikrokontrolerů i PC s různými OS
- Možnost přímé interakce z počítače s pamětí
- Malý rozměr

S paměťovou kartu mikroSD se komunikuje pomocí rozhraní SDIO, které přímo podporuje zvolený mikrokontrolér.

1.4 Výběr mikrokontroléru (MCU)

Pro optimální výběr mikrokontroléru byla stanovena tato kritéria.:

1. Předchozí zkušenosti s daným mikrokontrolérem.
2. Snadná přenositelnost kódu.
3. SDIO rozhraní.
4. Dostatečná frekvence jádra.
5. Možnost implementace protokolu I^2S pro přenos dat do kodeku.
6. Implementované rozhraní RMII(Reduced media-independent interface).
7. Dostupnost prototypového hardwaru.
8. Dostatečně velká RAM paměť pro zpracování zvukových souborů.

Pomocí těchto kritérii jsem vybral mikroprocesor STM32F429ZI[29].

Tento mikrokontrolér umožňuje nastavit frekvence jádra až 180MHz. Tato frekvence je dostatečná jak pro zpracování komunikace tak i pro potřebné úpravy signálu. Pro uložení dat na externí paměť, která je v tomto případě mikroSD karta, umožňuje použití rozhraní SDIO. Protokol I^2S je implementován pomocí SAI, který zaručuje přenositelnost kódu i do vyšších řad mikroprocesorů, jako je například řada F7.

Pro přenášení dat jsem zvolil technologii Ethernet, proto považuji za výhodu že kit od firmy STM32 s mikroprocesorem F429 již obsahuje zabudovaný LAN8742A[14] transceiver. Pomocí kitu lze vyzkoušet prototyp bez nutnosti výroby vlastní desky s mikrokontrolérem.

1.5 Výběr digitálně analogového převodníku

Byl vybrán jednoduchý audio kodek a to CS4344-CZZ[3] od firmy Cirrus Logic. Komunikace s kodekem je zprostředkována pomocí výše zmíněného protokolu I^2S. Kodek nedisponuje žádnými přídavnými funkcemi. Z důvodu možnosti bezproblémové výměny kodeku za jiný je snaha tyto funkce přesunout do mikrokontroléru.

Výhoda kodeku CS4344-CZZ je schopnost detekovat vzorkovací kmitočet audio signálu podle poměru signálů MCLK a LRCLK. Funkce těchto signálu je popsána při popisu sběrnice I^2S v kapitole 4.1.

2 Návrh prototypové desky plošných spojů

Vybrání konkrétních součástek umožňuje vytvoření prototypové desky plošných spojů, pro ověření funkčnosti konceptu. Deska plošných spojů byla navrhnuta v návrhovém softwaru Eagle. Výsledek návrhu lze najít v příloze A a B.

2.1 Koncept

Prototypová deska byla navržena jako rozšiřující modul pro komerčně dostupný vývojový kit Nucleo board od firmy STMicroelectronics[32]. Vývojový kit obsahuje jak základní věci jako vybraný mikrokontrolér STM32F429ZI, programátor ST-link pro jednoduché programování a potřebné napěťové regulátory pro bezproblémový chod. Vývojový kit obsahuje i pokročilé komponenty jako je ethernetový transciever s konektorem RJ45. Kitu schází audio kodek a slot pro mikroSD kartu, je proto nutné tyto periferie doplnit rozšiřujícím modulem.

Spojení vývojového kitu a rozšiřujícího modulu je realizováno dvěma páry konektorů. Tím je zajištěno jak elektrické, tak i mechanické spojení.

Pro lokální řízení byly navíc modulu přidány čtyři mikrospínače a pro indikaci čtyři LED diody. Modul je napájen z vývojového kitu, přes napěťový stabilizátor napájený z USB.

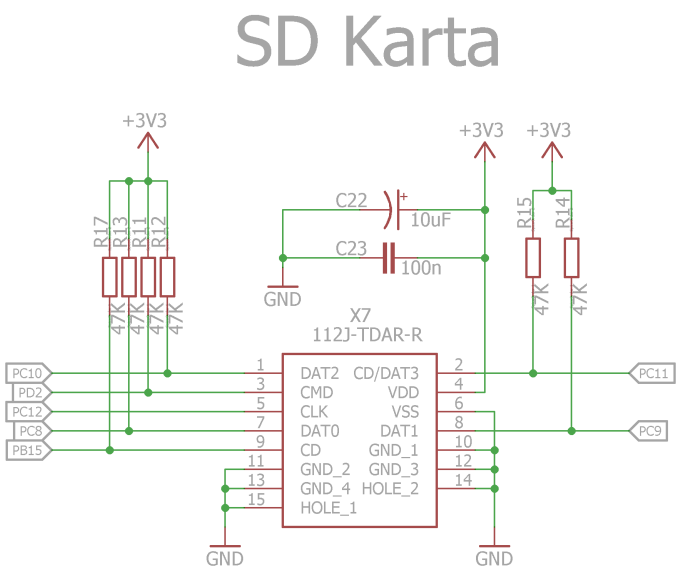
Při návrhu byl přidán i zesilovač TDA2822D[30] pro připojení sluchátek.

2.2 Realizace

Deska byla navrhnuta jako dvouvrstvá.

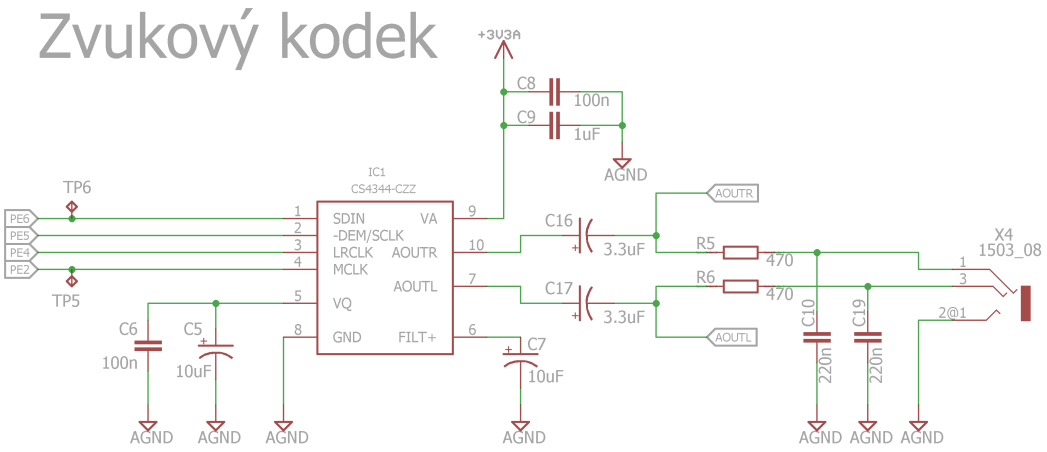
Jelikož se jedná o prototyp byla deska plošných spojů vyrobena pomocí obrábění CNC frézkou na katedře aplikované elektroniky a telekomunikací na Západočeské univerzitě v Plzni. Ačkoliv tato metoda výroby vykazuje jistá omezení, pro výrobu jednoho kusu je možné je zanedbat. Modul byl následně zapájen a oživen.

Na obrázku číslo 2 lze vidět schéma zapojení SD karty.



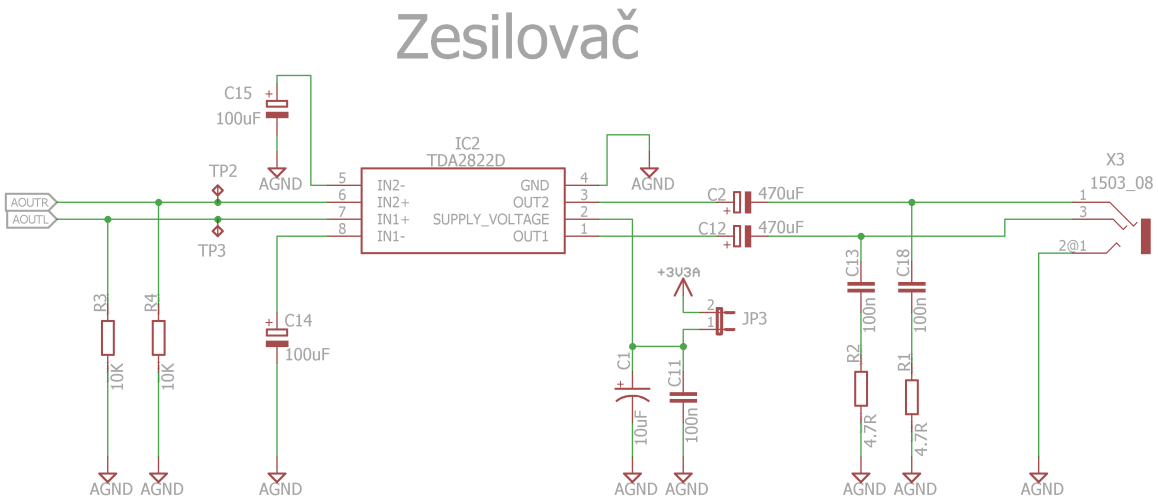
Obr. 2: Zapojení SD karty

Na obrázku číslo 3 lze vidět schéma zapojení zvukového kodeku CS4344-CZZ. Návrh vycházel z doporučeného zapojení v materiálech [3] na straně 11.



Obr. 3: Zapojení zvukového kodeku

Na obrázku číslo 4 je schéma zapojení zesilovače. Zapojení vychází z doporučeného zapojení od výrobce, které lze najít v materiálech [30] na straně 2.



Obr. 4: Zapojení zesilovače

3 Zvukové formáty[33]:

Zvuk je mechanické podélné vlnění v látkovém prostředí (vzduch, voda). Zvuk, který je schopen vyvolat zvukový vjem, označujeme slyšitelný zvuk. Jeho frekvence je přibližně od 16Hz až do 20kHz. Zvuk lze principiálně zaznamenat analogově nebo digitálně. U analogového záznamu se zvuk uloží do stopy (mechanické, magnetické, optické), která přímo určuje vlastnosti uloženého zvuku. U digitálního záznamu zvuku se spojitá funkce závislosti akustického tlaku na čase převede na posloupnost diskrétních vzorků. Kvalita takto zaznamenaného zvuku závisí na frekvenci vzorkování. Pro zmenšení velikosti zvukového souboru se často používá komprese signálu. Z hlediska komprimace dat rozlišujeme 3 základní typy formátů:

- formát bez komprese
- formát s bezeztrátovou kompresí
- formát se ztrátovou kompresí

3.1 Formát bez komprese[24]

Základním formátem, který nepoužívá žádnou kompresi, je formát LPCM . Ukládá se obvykle do souboru *.wav v OS Windows nebo *.aiff na MAC OS. Zpracování tohoto typu souboru je výpočetně nenáročné a vzhledem k jeho všeobecnému rozšíření se používá pro přenos zvukových souborů mezi různými systémy.

3.1.1 Soubor formátu aiff

*.aiff - formát souborů aiff se používá pro ukládání zvukových dat pro osobní počítače. Byl vyvinut společností Apple Inc. v roce 1988. Data jsou uložena v nekomprimované podobě. Z toho vyplývá jedna nevýhoda a to poměrně velké zvukové soubory. Na druhé straně však tento formát má vysokou kvalitu zvuku.

3.2 Formát s bezeztrátovou kompresí[2]

Formáty s bezeztrátovou kompresí snižují velikost audio souboru bez ztráty kvality. Původní nekomprimovaný soubor lze z tohoto formátu znovu obnovit. Typickým kodekem je FLAC.

3.2.1 Soubor formátu flac

*.flac – Je zkratka pro označení otevřeného, zvukového kodeku Free Lossless Audio Codek. Jde o bezeztrátový formát, což znamená, že komprimovaný zvuk je bez ztráty kvality. Jeho princip lze přirovnat ke komprimačním programům jako je ZIP nebo RAR. Komprimace formátu FLAC je ale rychlejší, protože jeho kompresní algoritmus je optimalizován pro audio data. Tento formát nachází uplatnění především pro archivaci zvukových souborů, protože nedochází ke snížení kvality záznamu.

3.3 Formát se ztrátovou kompresí

Ztrátová komprese snižuje velikost audio souboru odstraněním určitých dat. Při kompresi se odstraňují taková data, aby výsledná ztráta kvality byla ze subjektivního poslechu člověkem byla co nejmenší. Nejznámějším představitelem tohoto kódování je formát MP3.

3.3.1 Soubor formátu mp3[6]

mp3 – formát mp3 se stal jeden z nejoblíbenějších formátů pro zpracování zvukových souborů. Jeho komprese je založena na psychoakustické analýze. Ze zvukového souboru se odeberou data, které člověk neslyší nebo si je neuvědomuje. Velikost zkomprimovaného soubor ve formátu mp3 se zmenší přibližně na desetinu původního souboru.

3.4 Vybraný formát wav[10]

Po zvážení všech výhod a nevýhod výše popsaných formátů, byl pro realizaci diplomové práce vybrán formát bez komprese LPCM. Tento formát se často používá ve *.wav souborech. Jedná se o digitalizaci zvuku, která je kódovaná pulzně kódovou modulací s lineárním kvantováním. Kvalita převodu z analogového na digitální signál je ovlivňována dvěma parametry – frekvence vzorkování a kvantování. Frekvence vzorkování se používá od 8 kHz u telefonních linek přes 44,1 kHz pro digitalizaci hudby v CD kvalitě až po vyšší hodnoty vzorkování pro profesionální záznamová zařízení. Kvantování určuje přesnost digitalizace v jednotlivých bodech průběhu analogového signálu. Používají se hodnoty 8 nebo 16 bitů. LPCM je bezeztrátový formát a jeho zpracování je snadné bez nutnosti použití kodeku. Hlavička souboru obsahuje délku souboru, vzorkovací frekvenci, počet kanálů, velikost vzorků a další informace, nutné pro reprodukci zvukového záznamu.

3.4.1 Hlavička souboru wav

Soubor formátu wav se skládá ze tří částí – RIFF hlavička, sekce fnt a datová sekce. RIFF hlavička je dlouhá 12 bytů a obsahuje identifikátor „RIFF“, velikost souboru a identifikátor „WAVE“. Sekce fmt popisuje parametry vzorků dat. Je dlouhá 24 bytů a obsahuje: identifikátor „fmt „ , formát dat, počet kanálů, počet vzorků za sekundu pro jeden kanál, velikost rámce a počet datových bitů ve vzorku. Následuje sekce dat, která obsahuje signaturu „data“, délku datového bloku a od pozice 44 bytů jsou uložená zvuková data. Data jsou uložena způsobem „little endian“ tedy nižší byte je uložen na nižší pozici v souboru.

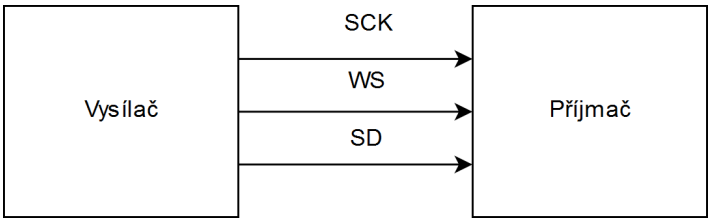
4 Protokoly pro přenos zvuku:

4.1 I²S sběrnice[11]:

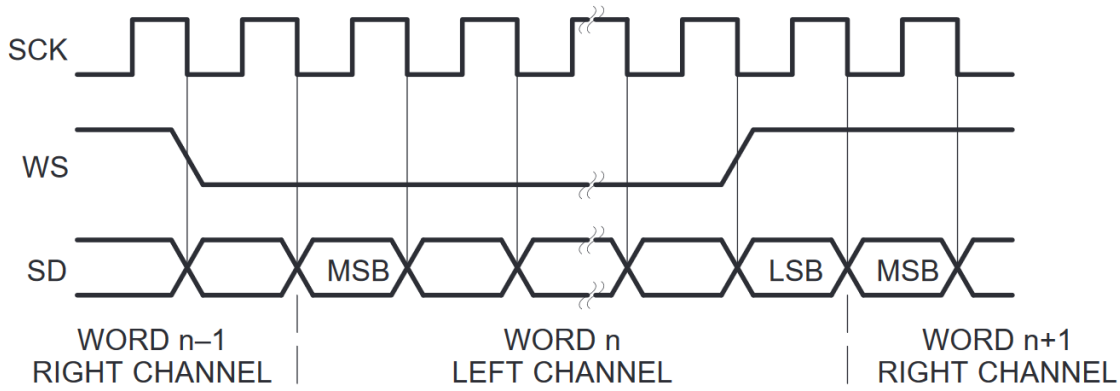
I²S sběrnice je určena pouze pro přenos audio signálu, zatímco řídící signály jsou přenášeny pomocí jiné sběrnice jako je I²C nebo SPI. Díky tomu sběrnici I²S stačí pouze 3 vodiče ke správné funkci. Data jsou přenášena sériově, pro oba kanály v jednom rámci po sobě.

Komunikace po sběrnici I²S je realizována třemi signály. SCK(continous serial clock), neboli nepřerušovaný sériový hodinový signál, WS(word select) sloužící k výběru levého nebo pravého kanálu a SD(serial data) sériová data pro přenos dat. Příklad zapojení signálů lze vidět na obrázku číslo 5. Ukázka komunikace na sběrnici je vidět na obrázku číslo 6.

V práci je použit ještě jeden signál který není součástí I²S standardu, ale je často součástí zapojení a to MCLK(master clock). Signál MCLK slouží pro synchronizaci vnitřních převodníků kodeku s mikrokontrolérem.



Obr. 5: Zapojení vysílače jako mastera [11,strana 1]



Obr. 6: Příklad komunikace na sběrnici[11, strana 1]

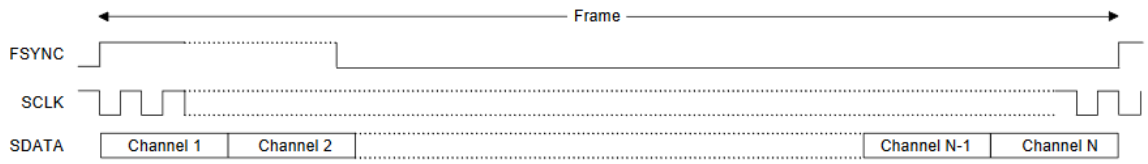
4.2 Time division multiplexed audio rozhraní[31]:

TDM audio rozhraní umožňuje posílání vícero kanálů po jedné digitální lince. TDM rozhraní nebylo standardizováno a proto může existovat vícero variant TDM formátu. Komunikace funguje na třech vodičích a to FSYNC (Frame synchronization pulse), SCLK(Serial clock) a SDATA(serial audio data line).

Signál FSYNC slouží k identifikaci začátku rámce. Začátek je indikován náběžnou hranou na tomto signálu. Většina implementací TDM ignoruje sestupnou hranu, lze ovšem najít výjimky které signál FSYNC používají pro indikaci šířky bloku kanálu.

Hodinový signál SCLK funguje k posunu dat ze zařízeních. Frekvence hodinového signálu je proporční k vzorkovacímu kmitočtu, počtu kanálů a šířce každého slova.

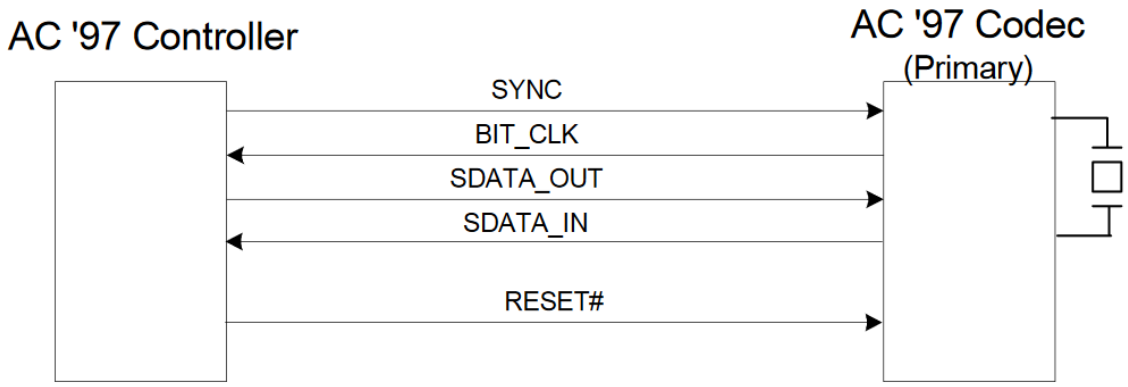
Typicky je přenášeno více kanálu, typicky 4,6 nebo 8 během jednoho rámce.



Obr. 7: Ukázka komunikace v TDM rozhraní[11, strana 2]

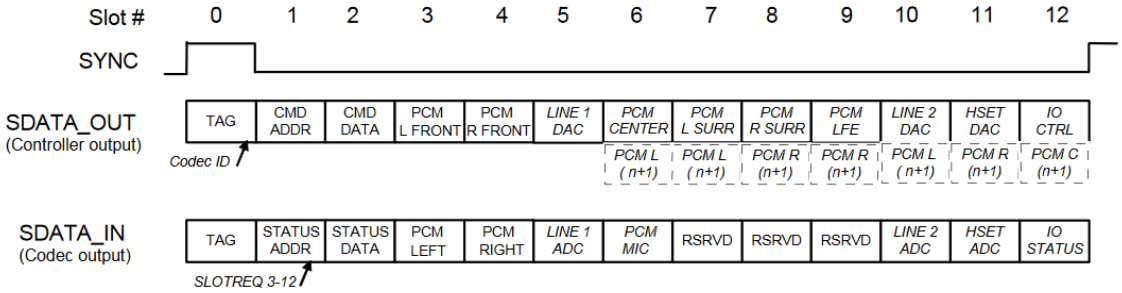
4.3 AC97 kodek[1]:

Nejjednodušší zapojení AC´97 je bodové spojení pomocí 5 vodičů. Zapojení lze vidět na obrázku číslo 8. Tyto vodiče jsou SYNC, BIT_CLK, SDATA_OUT,SDATA_IN a RESET.



Obr. 8: Zapojení kontroleru a kodeku podle specifikace AC´97[1, strana 19]

Přítomnost SDATA_OUT a SDATA_IN naznačuje obousměrný přenos dat. Hodinový signál BIT_CLK slouží k synchronizaci bitů.



Obr. 9: Příklad obousměrné komunikace[1, strana 26]

Z příkladu komunikace na obrázku číslo 9 je patrné, že nejde o čistý tok dat zvukového souboru jako v případě předcházejících kodeků. To má za následek, že každý rámec podporuje až dvanáct dvacetibitových datových slotů. Signál SYNC indikuje začátek rámce a rozděluje rámec na „TAG“ část a na datovou část. V „TAG“ části se určuje které datové sloty jsou povolené.

5 Přenos zvukových souborů:

5.1 Serial Audio Interface(SAI)[26]:

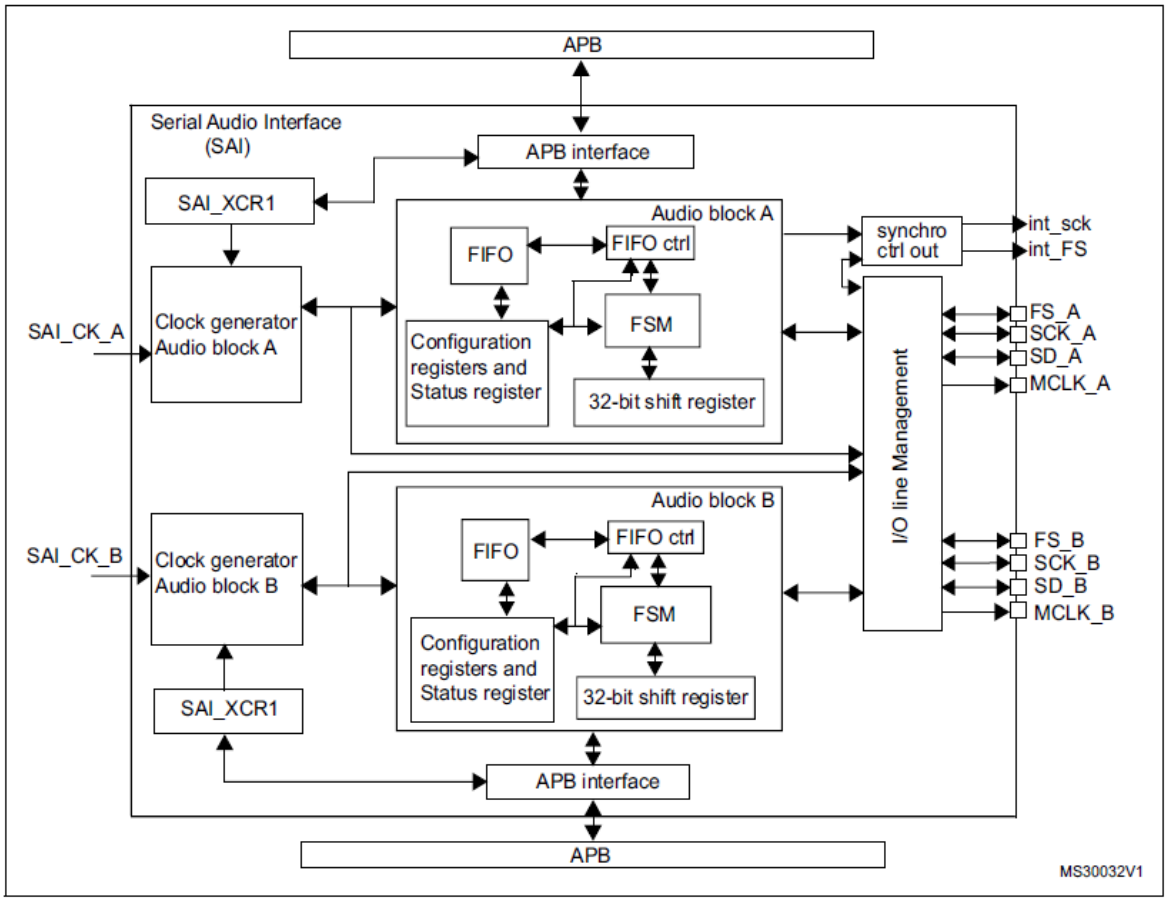
5.1.1 Úvod:

Sériové audio rozhraní (SAI) umožňuje mikrokontroleru komunikaci s pestrým výběrem audio zařízení. Rozhraní SAI může komunikovat s audio zařízeními pomocí různých protokolů:

- I^2S
- PCM
- AC97
- TDM

SAI rozhraní je schopno pracovat v slave nebo master módu. Pro větší flexibilitu obsahuje SAI rozhraní, integrované v mikrokontroleru STM32F429, dva bloky(Sub-Block A a Sub-Block B). Bloky mohou, nezávisle na sobě, zprostředkovat vysílání i přijímání. Oba bloky umožňují použití rozdílných MCLCK signálů.

5.1.2 Popis SAI rozhraní:



Obr. 10: Blokové schéma rozhraní SAI [4, strana 928]

Na blokovém schématu na obrázku č.10 je vidět že SAI periferie obsahuje dva audio bloky a to blok A a B. V této práci se využívá pouze blok A. Vstup SAI_CK_A je rozveden do bloku generování hodin pro audio blok A, který umožňuje měnit hodnotu hodin rozvedených do zbytku periferie. Tato funkcionalita je použita při změně MCLK vedoucích do audio kodeku, pro různé vzorkovací frekvence audio souborů. Tyto hodiny jsou také rozvedeny do Audio blocku A který obsahuje FIFO, FIFO ovladač,konfigurační a status registry.

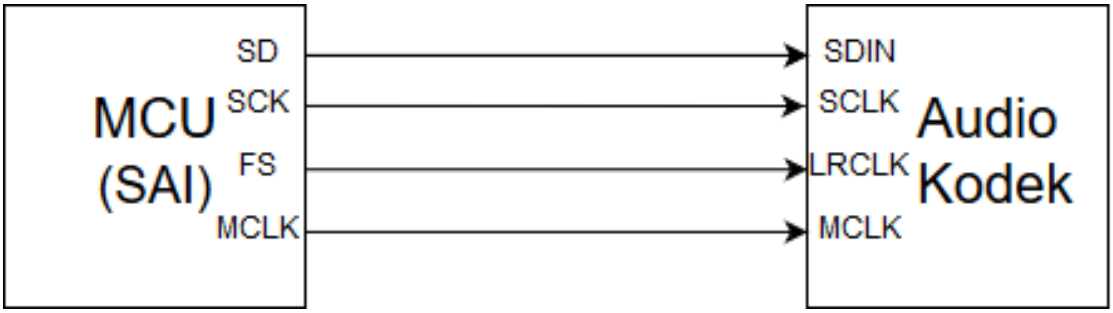
APB interface zajišťuje komunikaci s vnitřní APB sběrnicí. Registr SAI_xCR1, kde x je blok A nebo B, slouží jako konfigurační registr.

5.1.3 Důvod použití SAI:

Při realizaci komunikace s vybraným audio kodekem (CZ4344-CZZ), bylo nutné využít protokol I^2S po kterém zvolený kodek komunikuje.

Implementaci lze provést využitím SPI periferie se zapnutou možností I^2S a nebo pomocí SAI rozhraní. Vybráno bylo SAI rozhraní, z důvodu větší přenosnosti softwaru na jiný mikrokontroler nebo zvukový kodek, využívající jiného protokolu než je I^2S. Rozhraní SAI nabízí také několik vlastností

5.2 Realizace:



Obr. 11: Blokové schéma připojení audio kodeku k MCU

Na obrázku č.11 je zobrazeno připojení audio kodeku k MCU. Lze vidět připojení signálu FS na signál LRCLK, který při správném nastavení generuje signál určující, zda signál náleží levému nebo pravému kanálu. Zapojení ostatních signálů je již přímočaré. Ze zapojení je zřejmé že komunikace je pouze jednosměrná a to z mikrokontroléru do audio kodeku.

Inicializace modulu SAI je provedena ve funkci MX_SAI1_Init, vygenerovanou softwarem STM32Cube, kterou lze najít ve zdrojovém souboru DZM\Src\main.c. V této funkci je vidět struktura „hsai_BlockA1“ která je následně použita jako nastavení při inicializaci HALovou funkcí HAL_SAI_Init. Popisovaná struktura je vyobrazena na obrázku číslo 12.

```
hsai_BlockA1.Instance = SAI1_Block_A;
hsai_BlockA1.Init.Protocol = SAI_FREE_PROTOCOL;
hsai_BlockA1.Init.AudioMode = SAI_MODEMASTER_TX;
hsai_BlockA1.Init.DataSize = SAI_DATASIZE_16;
hsai_BlockA1.Init.FirstBit = SAI_FIRSTBIT_MSB;
hsai_BlockA1.Init.ClockStrobing = SAI_CLOCKSTROBING_FALLINGEDGE;
hsai_BlockA1.Init.Synchro = SAI_ASYNCHRONOUS;
hsai_BlockA1.Init.OutputDrive = SAI_OUTPUTDRIVE_DISABLE;
hsai_BlockA1.Init.NoDivider = SAI_MASTERDIVIDER_ENABLE;
hsai_BlockA1.Init.FIFOThreshold = SAI_FIFOTHRESHOLD_HF;
hsai_BlockA1.Init.ClockSource = SAI_CLKSOURCE_PLLSAI;
hsai_BlockA1.Init.AudioFrequency =  SAI_AUDIO_FREQUENCY_44K;
hsai_BlockA1.FrameInit.FrameLength = 32;
hsai_BlockA1.FrameInit.ActiveFrameLength = 16;
hsai_BlockA1.FrameInit.FSDefinition = SAI_FS_CHANNEL_IDENTIFICATION;
hsai_BlockA1.FrameInit.FSPolarity = SAI_FS_ACTIVE_LOW;
hsai_BlockA1.FrameInit.FSOffset = SAI_FS_BEFOREFIRSTBIT;
hsai_BlockA1.SlotInit.FirstBitOffset = 0;
hsai_BlockA1.SlotInit.SlotSize = SAI_SLOTSIZE_16B;
hsai_BlockA1.SlotInit.SlotNumber = 2;
hsai_BlockA1.SlotInit.SlotActive =SAI_SLOTACTIVE_ALL; //0x0000FFFF;
```

Obr. 12: Popisovaná struktura hsai_BlockA1

5.2.1 Nastavení rámce

V případě této práce je použit pouze jeden audio Sub-Block a to Sub-Block A. Je nastaven jako master pomocí bitu MODE[0] v registru SAI_xCR1 a zároveň jako vysílač(transmitter) pomocí bitu MODE[1]. Tento parametr je ve struktuře pojmenován jako „Init.AudioMode“ a je nastaven na hodnotu „SAI_MODEMASTER_TX“.

Jako další parametr je nutno nastavit velikost dat v registru SAI_xCR1 nastavením bitů DS[2:0]. Nastavitelné možnosti jsou 8,10,16,20,24 a 32 bitů. Tento parametr lze měnit a přizpůsobovat přijímaným datům. Protože vstupní data jsou 16 bitová, je i takto nastaven tento parametr. V inicializační struktuře lze tento parametr nastavit přiřazením hodnoty „SAI_DATASIZE_16“ členu struktury „Init.DataSize“.

Po nastavení těchto základních parametrů, je nutné přizpůsobit SAI používanému protokolu, v našem případě I^2S. Je nutné nastavit délku rámce nastavením bitů FRL[7:0] v registru. Při délce dat 16 bitů a použití protokolu I^2S, je potřeba délka rámce 32 bitů. Protože je reálná délka rámce vždy o jedna větší je nutno do registru zapsat číslo 31. Hodnota 32 v parametru „FrameInit.FrameLength“ v inicializační struktuře může být matoucí, ale lze dohledat ve funkci HAL_SAI_Init že při zápisu do registrů se zapíše hodnota „FrameInit.FrameLength – 1U“, neboli o jedna menší.

5.2.2 Nastavení signálu FS

Protokol I^2S potřebuje kromě sériových dat(SD) a hodinového signálu(SCK), také signál pro výběr slova(WS). WS signál určuje zda jsou data určeny pro levý nebo pravý kanál. V SAI rozhraní je pro tento účel určen správně nastavený frame synchronisation(FS) signál.

Nastavením bitu FSPOL v registru SAI_xFRCR se mění polarita signálu FS na začátku rámce. Protože jako první kanál chci přenést levý, nastavuji tento bit na 0. Opět je analogicky zapsána tato hodnota do inicializační struktury „FrameInit.FSPolarity = SAI_FS_ACTIVE_LOW“

Bez nastavení bitů FSALL[6:0] v SAI_xFRCR, by FS signál byl aktivní pouze po velikost 1 bitu. V protokolu I^2S je nutno aby byl signál FS aktivní polovinu rámce a proto je do registru zapsána hodnota 16. Tato hodnota je vypočtena jako maximální délka rámce dělena dvěma. „FrameInit.ActiveFrameLength = 16;“

Aby signál FS neukazoval pouze začátek rámce je nutné zapsat do bitu FSDEF v registru SAI_xFRCR hodnotu 1. V inicializační struktuře je tento krok proveden přiřazením hodnoty „SAI_FS_CHANNEL_IDENTIFICATION“ do parametru „FrameInit.FSDefinition“.

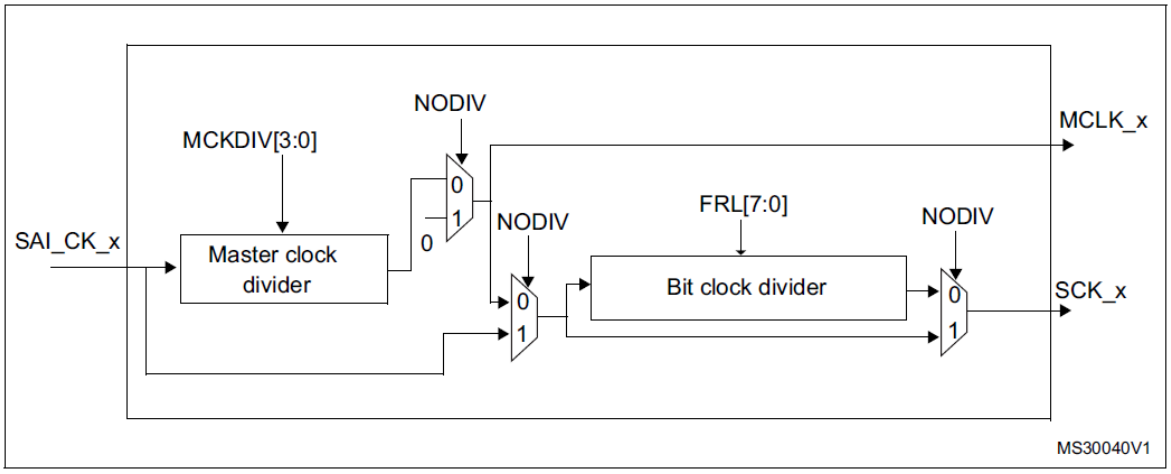
5.2.3 Nastavení slotů

Základní stavební prvek rámce se nazývá slot. Tyto sloty jdou libovolně povolovat a zakazovat v registru SAI_xSLOTR nastavením bitů SLOTEN[15:0]. Pro protokol I^2S tato možnost není využita a jsou povoleny sloty nastavením „SlotInit.SlotActive“ na hodnotu „SAI_SLOTACTIVE_ALL“ neboli číslo 0xFFFF.

Následně je nutné nastavit šířku slotů. Pokud je šířka slotu delší než délka dat doplní se nulami. Tento efekt je v tomto případě nežádaný a proto je nastavena šířka slotu na hodnotu 16 bitů. Lze nastavení provést jak zapsáním do registru SAI_xSLOTR a nastavením bitu SLOTZ[1:0], nebo zapsáním „SAI_SLOTSIZE_16B“ do parametru „SlotInit.SlotSize“.

Posledním krokem je nutné říci kolik slotů bude rámec obsahovat. Pro protokol I^2S o šířce dat 16 bitů a dvou kanálů je nutné nastavit velikost 2 slotů, pro každý kanál jeden. Nastavení se provede zapsáním hodnoty počtu slotů -1 do registru SAI_xSLOTR na bity NBSLOT[3:0]. Při nastavení pomocí struktury a knihovny HAL je zapsána hodnota 2 do „SlotInit.SlotNumber“. Funkce „HAL_SAI_Init“ již zabezpečí správné odečtení hodnoty a zinicializování.

5.2.4 Generování hodin



Obr. 13: Blokové schéma generování hodin audio bloku[4, strana 935]

Při nastavení audio bloku do role mastera, generátor hodin produkuje jak komunikační hodiny na výstup SCK_x tak i master hodiny do externích hodin kodeku na výstup MCLK_x. Hodnota MCLK musí být minimálně 256 až 1280(256*5)krát vyšší než je vzorkovací frekvence hudebního signálu. Lze proto bezpečně nastavit SAI_CK_x tak aby byla 44,1kHz*256 pro přehrání audio souborů vzorkované běžnou vzorkovací frekvencí 44,1kHz. Nastavení je provedeno v programu STMCube.

Pomocí MCKDIV [3:0], který slouží jako dělička frekvence, lze tento signál vyladit při změně vzorkovací frekvence audio souboru na Nx menší hodnotu. V programu je nastavení provedeno zápisem hodnot do struktury „Init.AudioFrequency“ a to buď „SAI_AUDIO_FREQUENCY_44K“, kdy se nastavení MCKDIV provede funkcí „“HAL_SAI_INIT“ dle nastavené požadované frekvence, a nebo zápisem „SAI_AUDIO_FREQUENCY_MCKDIV“ , který nechá nastavení MCKDIV na uživateli.

6 Implementace přehrávání zvukových souborů

Jádro přehrávání stojí na dvou bufferech o velikosti 22050 šestnáctibitových hodnot. Velikost bufferu vychází z vzorkovací frekvence, která v tomto případě činí 44,1KHz. Je tedy nutné audio kodeku poslat 44100 vzorků každou sekundu pro jeden kanál. Při stereo přehrávání je potřebný počet vzorků navýšen na hodnotu 88200. Jednoduchým dělením potřebných vzorků(88200) a počtu uložených(22050), lze odvodit že velikostně bude buffer stačit na $\frac{1}{4}$ záznamu. Časově lze vyjádřit $\frac{1}{4}$ záznamu jako 250milisekund.

Při příjmu příkazu pro začátek přehrávání je zavolána funkce „SAI_begin_transmission“, kterou lze najít v zdrojovém souboru Src/file_operations.c na řádce 148. V této funkci se nastaví příznak „Play“ na hodnotu jedna. Tím je indikováno nepřerušované přehrávání. Dále se nastaví proměnná „SAI_file_pos“ na hodnotu „WAV_offset“. Tato hodnota je definovaná v hlavičkovém souboru „file_operations.h“ jako číslo 44, tím je zaručeno přeskočení hlavičky souboru wav v dalším kroku. Hodnota WAV_offset je odvozena ze struktury hlavičky wav souboru. Struktura souboru wav je podrobně popsána v kapitole 3.4.1.

Následně se zavolá funkce „Fill_SAI_buffer“ s argumentem „SAI_Buffer_0“. Tato funkce je k nalezením ve stejném souboru na řádce 125. Hlavní role této funkce je, jak již název napovídá, naplnění bufferů pro přehrání. Vstupní argument této funkce je číslo bufferu, který je potřeba zaplnit daty. Protože buffer „Sai_buffer“ je dvouřádkové pole, je nejprve zkontrolováno zda je číslo v rozmezí a nezpůsobí se zápis do nesprávné oblasti.

V dalším kroku se otevře vybraný soubor na mikroSD kartě. Soubor je z důvodu zabezpečení otevřen čistě pro čtení. Nyní již přijde na řadu proměnná „SAI_file_pos“ ve které je hodnota začátku audio dat ve wav souboru. Provede se seek funkce, kterou se v souboru posune čtecí ukazatel na pozici 44. Buffer do kterého se data ukládají je 16 bitový. Funkce která čte z mikroSD karty má jako vstupní argument počet bytů k přečtení, je proto nutné přečíst dvakrát více bytů, aby se dosáhlo stejné velikosti jako má ukládací buffer.

Po správném přečtení je zkontrolováno zda se přečetlo tolik bytů kolik bylo zadáno a o tolik se zvětší proměnná „SAI_file_pos“ a zapíše se do proměnné „Bytes_read“. Tím se zajistí kontinuita čtení ze souboru.

Nyní lze již zavolat funkci „SAI_Play“, která začne přenos z prvního bufferu. Přenos je proveden pomocí funkce z knihovny HAL, které je jako vstupní argument dána struktura SAI, ukazatel na buffer a délka dat. Délka dat je v tomto případě polovina počet přečetných bytů, protože funkce již implicitně bere data jako šestnácti bitové.

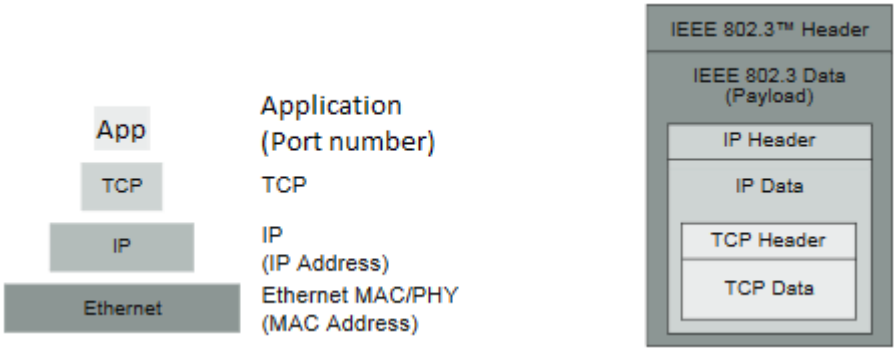
Nyní jsou již data přenášeny do rozhraní SAI a dále rozvedeny do zvukového kodeku. Pro plynulé přehrávání se naplní ještě druhý buffer a vybere se jako aktivní.

Po dokončení odesílání je vyvoláno přerušení „HAL_SAI_TxCpltCallback“, které lze najít v zdrojovém souboru Src/main.c na řádce 502. Pokud je povoleno přehrávání, je opět zavolána funkce „SAI_Play“. A následně je nastaven příznak „SAI_Fill_flag“.

Při aktivním příznaku „SAI_Fill_flag“ je ve funkci main na řádce 229 povoleno naplnění neaktivního bufferu. Před samotným naplněním je zkontrolováno zda počet přečtených bytů odpovídá velikosti bufferu. Pokud se nerovnají, znamená to konec souboru a vypnutí přehrávání zapsáním hodnoty 0 do proměnné „Play“. Pokud je aktivován mód „Play_All“ vybere se další soubor jako aktivní a přehrávání pokračuje. Jestliže nenastal ani jeden z těchto příkladů jsou nahrány data do neaktivního bufferu a jeho vybrání jako aktivní.

7 Komunikace

Na obrázku číslo 14, lze vidět, které všechny vrstvy jsou v této práci zapotřebí pro vyslání dat. Vyplývá to ze zapouzdření jednotlivých vrstev. Na pozici aplikačních dat, si lze představit zvukový soubor, či příkaz zařízení. Tato data jsou následně zapouzdřena do protokolu vyšší vrstvy (TCP, UDP). V dalším kroku je provedeno zapouzdření do paketu Internet Protokolu. Na konec je nutné paket vyslat a proto je zapouzdřen do Ethernetového rámce, který je následně odeslán.



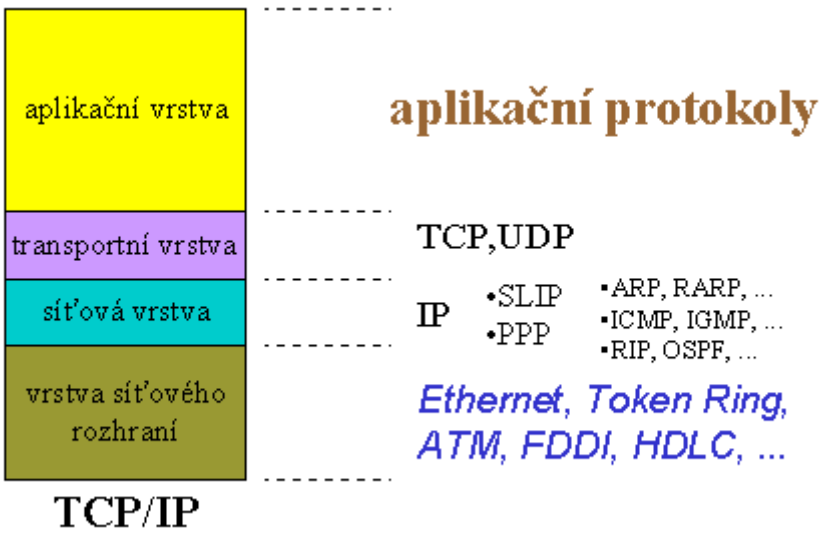
Obr. 14:Zapouzdření dat jednotlivými vrstvami [28,strana 4]

Pro správnou funkčnost komunikace je nutné implementovat všechny vrstvy z příkladu výše.

Pro komunikaci pomocí technologie Ethernet byl použit LAN8742A transceiver s konektorem RJ45. Tento transceiver komunikuje s mikrokontrolérem pomocí rozhraní RMII. Protokoly vyšších vrstev jsou již implementovány čistě softwarově.

7.1 Síťová architektura TCP/IP [22,23]

7.1.1 Popis jednotlivých vrstev TCP/IP [22]



Obr. 15: Příklady jednotlivých vrstev TCP/IP[22]

Aplikační vrstva je specifická vždy pro konkrétní použité aplikace. V případě zařízení DZM je to samotný program v mikrokontroléru.

V transportní vrstvě již figurují protokoly TCP a UDP. Tyto protokoly jsou více popsány v kapitolách 7.5 a 7.6. Jejich úkol je přenos aplikačních dat.

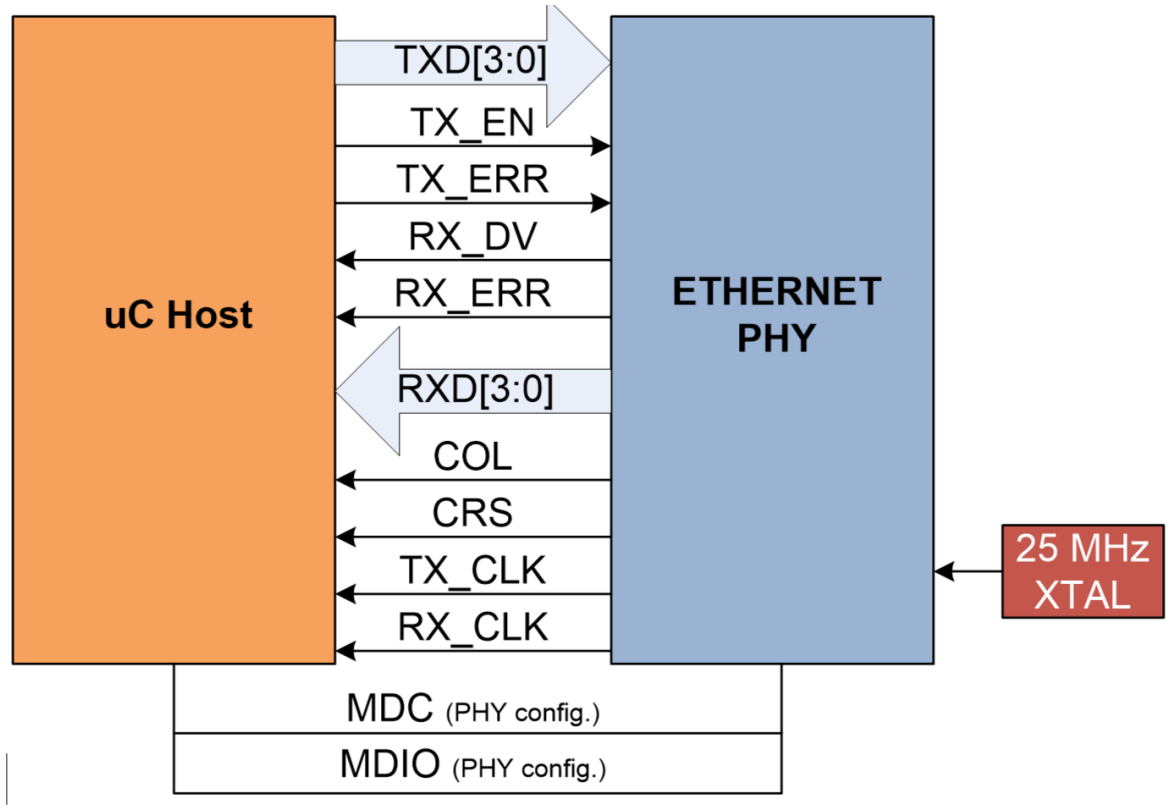
Mezi protokoly síťové vrstvy se řadí samotný Internet Protocol, více o protokolu v kapitole 7.4, a jeho podpůrné protokoly jako je ARP (kapitola 7.8),ICMP(kapitola 7.7) a IGMP. Protokol IP zajišťuje přenos datových paketů obsahující datagramy transportní vrstvy.

Vrstva síťového rozhraní již obsahuje fyzické přenosové médium. V případě této práce jde o technologii Ethernet (kapitola 7.3) přenášené pomocí kroucené dvoulinky.

7.2 Media Independent rozhraní[14]

Media Independent rozhraní slouží pro spojení MAC vrstvy Ethernet rozhraní v mikrokontroléru s Ethernet transcieverem. Rozhraní MII není závislé na fyzickém médiu pro přenášení dat. Komunikace s transcieverem, pomocí rozhraní MII, bude stejná ať se jedná o transciever pro kroucenou dvoulinku nebo pro optické vlákno.

MII rozhraní využívá celkem devatenáct vodičů pro komunikaci. Příklad zapojení lze vidět na obrázku číslo 16.



Obr. 16: Zapojení MII rozhraní [18]

7.2.1 Reduced Media Independent rozhraní[14]

RMII rozhraní se liší od MII rozhraní v několika bodech:

- Poloviční šířkou datových signálu RxD a TxD. Z 4 bitové šířky na 2 bitové šířky.
- Dvojnásobnou frekvencí hodin (50 MHz), přivedenou z externího zdroje.

Hlavní cíle RMII rozhraní bylo zmenšení počtu ovládacích pinů, zredukování hodin na jeden hodinový signál a počet pinů nezávislém na hustotě portů PHY vrstvy.

Právě zdvojnásobením frekvence bylo umožněno zredukování o 4 datové piny bez ohrožení stávajících vlastností. RMII specifické signály můžeme vidět v tabulce číslo 1.

Více informací o funkci RMII rozhraní lze najít v materiálech číslo [14].

Tab. 1: RMII specifické signály

Jméno signálu	Použití
REF_CLK	Synchronní hodinový signál
CRS_DV	Detekce nosné / Platnost přijatých dat
RXD[1:0]	Přijaté data
RX_ER	Chyba příjmu
TX_EN	Povolení vysílání
TXD[1:0]	Vysílaná data

7.3 Ethernet[19]

Technologie Ethernet podle referenčního modelu ISO/OSI patří do linkové a zároveň zasahuje do fyzické vrstvy. Při využití referenčního modelu TCP/IP lze zařadit technologii Ethernet čistě do síťového rozhraní.

Právě technologie Ethernet velmi dobře spolupracuje s protokoly TCP/IP. Pakety IP protokolu jsou nejčastěji zapouzdřeny do Ethernetových rámců.

Původně byl přístup stanice ke sběrnici na základě metody CSMA/CD. Metoda CSMA/CD je postavena na základě náhodného přístupu a detekci kolize. Při vysílání, stanice kontroluje sběrnici zda nedochází ke kolizi, pokud ano, vyšle se jam signál. Následně obě stanice přestanou vysílat, vyberou náhodnou čekací dobu a po uplynutí této doby začne stanice opět vysílat.

V dnešní době, převážně v domácnosti je jako přenosové médium použita hlavně kroucená dvoulinka. Kroucená dvoulinka, na rozdíl od koaxiálního kabelu umožňuje pouze dvoubodové spojení. Místo odboček bylo nutné zavedení prvků jako je hub nebo switch. Byl vyvinut standard pro použití kroucené dvoulinky a to 10BASE-T a jeho následovníci 100BASE-TX,1000BASE-T a 10GBASE-T. 10BASE-T značí rychlost 10Mbitů a písmeno T na konci znamená „twist“ od slova „twisted pair“(kroucená dvoulinka).

7.3.1 Ethernetový rámec[4]

Tab. 2: Ethernetový rámec[4]

Preambule	Oddělovač začátku rámce	Cílová MAC adresa	Zdrojová MAC adresa	EtherType	Data	Kontrolní posloupnost rámce
7 Bytů	1 Byte	6 Bytů	6 Bytů	2 Byty	46-1500 Bytů	4 byty

Ethernetový rámec nejprve začíná 7 bytovou preambulí, určenou pro příjmací systém k oznámení začátku rámce a synchronizaci. Preambule si lze představit jako 8x se opakující číslo 55_{16}. Následuje oddělovač začátku rámce, který tento vzor ruší a to číslem D_{16}.

Ihned po něm následuje cílová 6 bytová MAC adresa určující kam rámec směřuje. Zdrojová MAC adresa,

Pole EtherType definuje typ protokolu uvnitř rámce. Pro protokol IPv4 je obsah pole číslo 0800_{16} a pro ARP protokol je to číslo 0806_{16}.

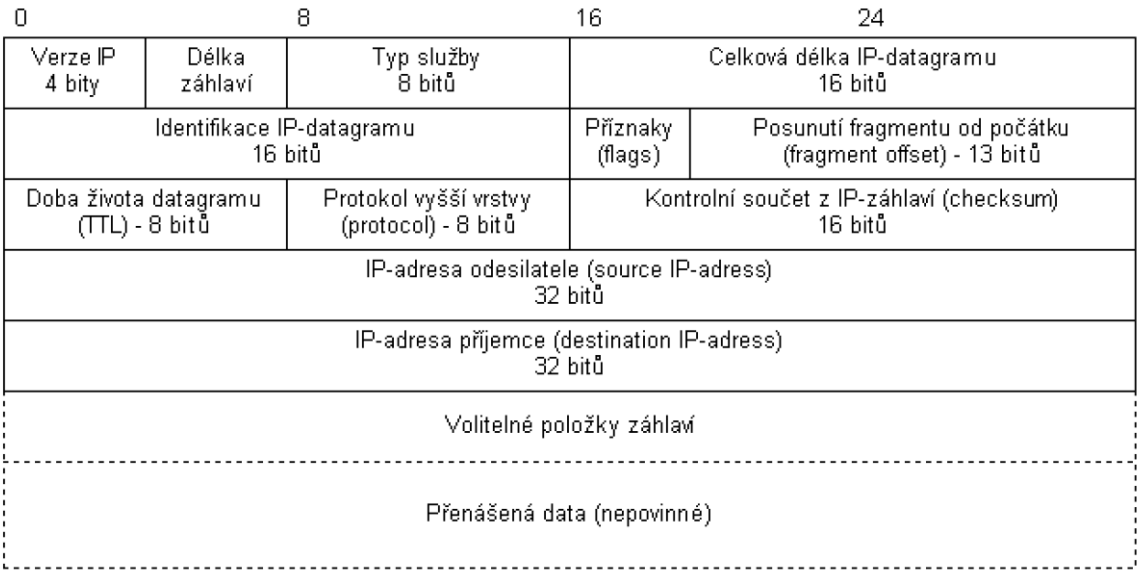
Položka Data obsahuje již vložené protokoly jako IPv4 a ARP. Pokud je výsledný rámec menší než je minimální délka Ethernetového rámce(46 bytů), je pole Data použito jako bytová vycpávka.

Na konci je pole obsahující kontrolní posloupnost rámce, použité ke zjištění chyb v rámci.

7.4 Internet Protokol[12]

Internet protokol je základní protokol, který se používá v počítačových sítí a na internetu. Data mezi dvěma libovolnými počítači se posílají po blocích nazývaných pakety. Internet protokol je zodpovědný za směrování jednotlivých paketů a jsou posílány zcela nezávisle. Každý paket obsahuje dva jednoznačné identifikátory tzv. IP adresy odesílatele a příjemce. IP adresa je dlouhá 4 bytů v případě protokolu IPv4 nebo 6 bytů pro verzi IPv6.

Internet Protokol není využíván pro přenos samotných dat, ale bývají na místě dat zapouzdřeny protokoly vyšších vrstev.



Obr. 17: IP-paket [12,strana 124]

Pole Verze obsahuje verzi protokolu IP. Protože se v práci používá IP protokol verze 4, je tato položka rovná číslu 4.

Položka délka záhlaví obsahuje délku IP záhlaví. Číslo uvnitř této položky je 4x menší než je délka bytů záhlaví.

Typ služby byl původně zamýšlen pro označení paketů jako urgentní. V dnešní době je pole využíváno k „Differentiated services“ definovanou v RFC2474.

Celková délka IP-paketu indikuje celkovou délku IP paketu v bytech. Při velikosti 16 bitů, lze vypozorovat že maximální délka IP paketu je 65535 bytů.

Identifikace IP paketu je spolu s příznaky a posunutím fragmentu využito pro fragmentaci paketů.

Doba života datagramu, známější spíše pod zkratkou TTL, určuje maximální počet přesměrování než je paket zahozen.

Protokol vyšší vrstvy identifikuje číselně který protokol vyšší vrstvy (TCP,UDP atd..) je použit v paketu.

Kontrolní součet z IP záhlaví je vypočten pouze se záhlaví a nikoliv z dat. Kontrolní součet je vypočten jako jedničkový doplňek součtu všech 16 bitových slov v záhlaví. Při počítání je nutno dosadit za položku kontrolního součtu nuly.

Zbývají pouze 2 povinná pole a to IP adresa odesílatele a IP adresa příjemce. Obě tyto hodnoty jsou 32 bitové.

Následují volitelné položky záhlaví a data.

7.5 User Datagram Protocol[12]

Tab. 3: UDP datagram[12]

Zdrojový port 16 bitů	Cílový port 16 bitů
Délka dat 16 bitů	Kontrolní součet(dobrovolný) 16 bitů
DATA	DATA

Protokol UDP je velmi jednoduchý protokol, protože se nestará o to zda UDP datagram příjemci dojde. O to se musí postarat aplikační protokol. Velmi malá režie paketů znamená, že jde o rychlý a datově lehký protokol a z tohoto důvodu je v této práci použit pro posílání příkazů do zařízení.

Jelikož se jedná o protokol vyšší vrstvy je UDP protokol, jak vychází z hlavičky, směřován na jednotlivé porty určující které aplikaci datagram náleží. Nelze si vybrat libovolný port, protože porty 0-1023 jsou vyhrazeny pro systémové porty a „dobře známé“ porty jako například protokol DNS, který používá port 53. 1024-49152 jsou registrované u společnosti IANA. Porty 49153-65535 jsou vyhrazeny pro soukromé použití. Pro posílání příkazu je v aplikaci použit port 50000.

Hlavní výhoda UDP protokolu je absence spojovaného připojení, čili je možné vysílat broadcast zprávy. Tato vlastnost je použita pro získání IP adresy zařízení při zapnutí PC aplikace pomocí implementace vlastní funkce DZMIP.

Výpočet kontrolního součtu je v tomto protokolu dobrovolný a pokud není použit, je pouze na aplikaci ověření správnosti dat.

7.6 Transmision Control Protocol[12]

Tab. 4: Hlavička TCP protokolu[12]

Zdrojový port 16 bitů		Cílový port 16 bitů	
Pořadové číslo odeslané sekvence bajtů 32 bitů			
Pořadové číslo potvrzené sekvence bajtů 32 bitů			
Offset dat 4 bity	Rezerva 6 bitů	Příznaky 6 bitů	Délka okna 16 bitů
Kontrolní součet 16 bitů		Ukazatel urgentních dat 16 bitů	
Možnosti (nepovinné)			
DATA			

Je patrné z hlavičky TCP protokolu že jde o složitější protokol než je výše zmiňovaný UDP. Protože jde o protokol vyšší vrstvy je opět směřován na porty v cílovém zařízení jako protokol UDP, avšak sady portů pro oba protokoly jsou různá. Lze tedy využít stejné číslo portu pro různé aplikace pokud se s aplikacemi komunikuje rozdílnými protokoly.

Následuje Sequence number a Acknowledgment number, které jsou nutnou součástí protokolu pro zaručení správného doručení dat. Sequence number určuje pořadí odeslaných dat, zatímco Acknowledgment number potvrzuje počet přijatých bytů.

Dalším důležitým polem jsou příznaky které mohou být:

- URG Urgent pointer značí že položka ukazatel urgentních dat je platná a ukazuje na data,
- ACK Ackownledge příznak značí platnost položky Acknowledgment number v TCP segmentu.
- PSH Push příznak indikuje že segment obsahuje aplikační data .
- RST Reset příznak značí zrušení spojení a navrácení do výchozího nastavení.
- SYN Synchronization příznak je použit při zahajování spojení a zahájení nové sekvence číslování sequence number od náhodného čísla.
- FIN Final značí ukončení odesílání dat.

Stejně tak délka okna, která indikuje protějšku maximální počet bytů kolik může zařízení přijmout. Protější strana musí tuto hodnotu respektovat a neposlat více bytů než je povoleno. Pomocí této hodnoty lze tak regulovat tok dat podle volného místa v paměti.

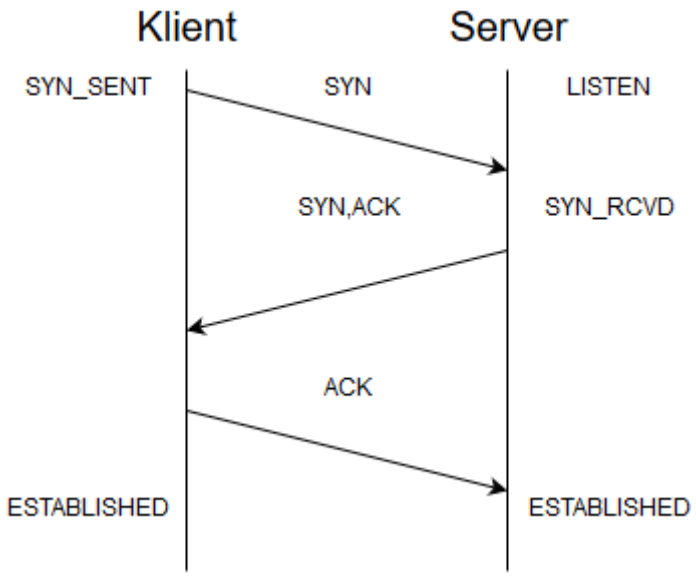
Na rozdíl od protokolu UDP, kdy kontrolní součet je čistě dobrovolný, je nedílnou součástí protokolu TCP a musí být spočten a zkontrolován jak při příjmu tak při vysílání.

Následuje pole možností, které je využito při navazování spojení při aktivním příznaku SYN. V poslední části segmentu jsou již poslaná data.

Pro výše zmíněné vlastnosti je protokol TCP použit v práci čistě pro přenášení zvukových souborů.

7.6.1 Navázání spojení

Jeden z hlavních rozdílů je, že jde o spojovaný protokol. Při pokusu o navázání spojení se musí nejprve provést tří fázový handshake. Jeden z účastníků se chová jako server a druhý jako klient. Jak je vidět na obrázku č. 18 spojení zahajuje klient posláním paketu s nastaveným příznakem SYN a popřípadě nastavením v poli „možnosti“.



Obr. 18: Třífázový handshake protokolu TCP [12]

Server odpoví jeho nastavením a příznakem SYN. Již ale potvrzuje předchozí zprávu proto je nutné zvětšit Ackonwledgment number a nahodit příznak ACK.

Poslední část handshaku je potvrzení předchozího paketu klientem a tím je stav jak serveru tak i klienta ve stavu „Established“ neboli navázáno.

V tomto stavu je již možno plně komunikovat v obou směrech.

Tab. 5: Implementace třífázového handshaku(Snímáno programem wireshark)

No.	Time	Source	Destination	Protocol	Length	Info
93	14.984124	192.168.1.102	192.168.1.103	TCP	66	49974 > 50001 [SYN] Seq=0 Win=17520 Len=0 MSS=1460 WS=256 SACK_PERM=1
94	14.984391	192.168.1.103	192.168.1.102	TCP	62	50001 > 49974 [SYN, ACK] Seq=0 Ack=1 Win=1456 Len=0 MSS=1456
95	14.984441	192.168.1.102	192.168.1.103	TCP	54	49974 > 50001 [ACK] Seq=1 Ack=1 Win=65520 Len=0

IP adresa 192.168.1.102 náleží počítači ze kterého byla komunikace inicializována a IP adresa 192.168.1.103 je již adresa zařízení. Hodnoty Seq a Ack jsou relativní k začátku relace. Z ukázky je vidět v hranatých závorkách v kolonce info jak předání nastavení druhé straně tak i změnu příznaku v pořadí SYN;SYN,ACK a ACK.

7.6.2 Datový přenos

Po navázání komunikace, již probíhá přenos dat. Lze vidět aktivní příznak PSH, který signalizuje platnost dat. Přenos dat probíhá tak, že klient vyšle tolik bytů kolik mu dovolí přenosové médium a tolik aby nepřesáhl velikost okna. Velikost okna zařízení je v tomto případě 1456 bytů, tím je navíc zaručeno že odesílatel nepošle více dat, než se vejde do jednoho paketu. Přijímač potvrzuje přijaté byty posláním paketu s ACK příznakem a zvětšeným Acknowledgment číslem o počet správně přijatých bytů. Velikost okna souhlasí se zvětšováním Acknowledgment čísla po 1456 bytech.

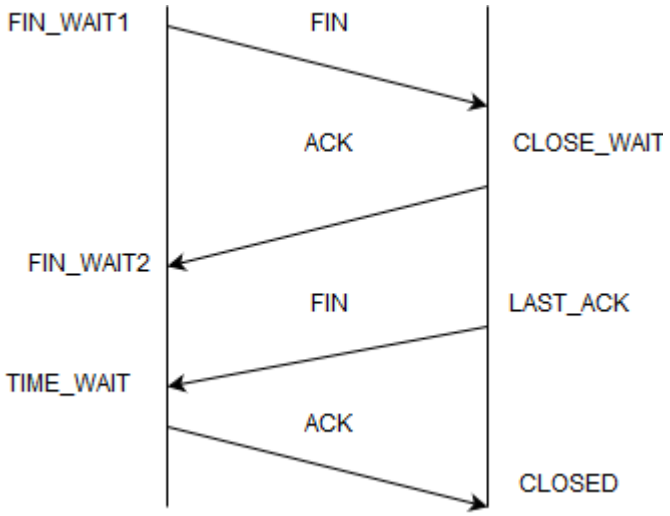
Tab. 6: Ukázka části přenosu 500KB zvukového souboru(Snímáno programem wireshark)

No.	Time	Source	Destination	Protocol	Length	Info
96	14.987984	192.168.1.102	192.168.1.103	TCP	1510	[TCP Window Full] 49974 > 50001 [PSH, ACK] Seq=1 Ack=1 Win=65520 Len=1456
97	14.991119	192.168.1.103	192.168.1.102	TCP	60	50001 > 49974 [ACK] Seq=1 Ack=1457 Win=1456 Len=0
98	14.991140	192.168.1.102	192.168.1.103	TCP	1510	[TCP Window Full] 49974 > 50001 [PSH, ACK] Seq=1457 Ack=1 Win=65520 Len=1456
99	14.993478	192.168.1.103	192.168.1.102	TCP	60	50001 > 49974 [ACK] Seq=1 Ack=2913 Win=1456 Len=0
100	14.993496	192.168.1.102	192.168.1.103	TCP	1510	[TCP Window Full] 49974 > 50001 [PSH, ACK] Seq=2913 Ack=1 Win=65520 Len=1456
101	14.996374	192.168.1.103	192.168.1.102	TCP	60	50001 > 49974 [ACK] Seq=1 Ack=4369 Win=1456 Len=0

7.6.3 Ukončení spojení

Spojení je možné ukončit dvěma způsoby, RST příznakem v paketu a nebo čtyř fázovým handshakem. Při přijmutí příznaku RST je vnitřní stav TCP serveru vrácen do výchozího nastavení před příjmem prvního handshaku a to do stavu LISTEN.

Druhou možností je čtyř fázový handshake, který je vidět na obrázku č.19. a jeho reálná implementace v zařízení je vidět v tabulce č.7. Jak je vidět z reálného příkladu, příznak FIN může být součástí segmentu nesoucího data a tím značit že jde o poslední data, která byla odeslána.



Obr. 19: Čtyř fázový handshake pro ukončení TCP spojení[12]

Tab. 7: Ukázka implementovaného čtyř fázového handshaku(Snímáno programem wireshark)

No.	Time	Source	Destination	Protocol	Length	Info
744	17.877848	192.168.1.102	192.168.1.103	TCP	1388	49974 > 50001 [FIN, PSH, ACK] Seq=439713 Ack=1 Win=65520 Len=1334
745	17.879996	192.168.1.103	192.168.1.102	TCP	60	50001 > 49974 [ACK] Seq=1 Ack=441047 Win=1456 Len=0
746	17.900087	192.168.1.103	192.168.1.102	TCP	60	50001 > 49974 [FIN] Seq=1 Ack=441048 Win=1456 Len=0
747	17.900130	192.168.1.102	192.168.1.103	TCP	54	49974 > 50001 [ACK] Seq=441048 Ack=2 Win=65520 Len=0

Lze vidět v tabulce č. 7 posloupnost příznaku FIN;ACK;FIN;ACK v kolonce info. Po dokončení handshaku je již server připraven na další spojení.

7.7 Internet Control Message Protocol[12]

Protokol ICMP lze považovat za služební protokol, který je součástí IP-protokolu. V této práci je z protokolu ICMP použita hlavně zpráva pro Echo a Echo request. Tyto dvě zprávy jsou použity společně pro otestování dosažitelnosti zařízení v síti. Žadatel, v tomto případě PC, vyšle Echo request zprávu cílovému uzlu, který na ní odpoví zprávou Echo.

```
Pinging 192.168.1.103 with 32 bytes of data:
Reply from 192.168.1.103: bytes=32 time<1ms TTL=128
Reply from 192.168.1.103: bytes=32 time<1ms TTL=128
Reply from 192.168.1.103: bytes=32 time=1ms TTL=128
Reply from 192.168.1.103: bytes=32 time<1ms TTL=128

Ping statistics for 192.168.1.103:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 1ms, Average = 0ms
```

Obr. 20: Ukázka příkazu ping na zařízení DZM z operačního systému Windows

Příkazem ping, který je běžnou součástí operačních systémů jako je Windows, lze zjistit dostupnost zařízení. Na obrázku č.20 je vidět odpověď zařízení DZM na příkaz ping vyslaného z počítače.

Ačkoliv není implementace zprávy Echo povinna, je to velmi vhodný nástroj při zjišťování problémů. Z dříve zmíněného důvodu byla tato funkcionalita implementována do zařízení.

7.8 Address Resolution Protocol[12,5]

Ačkoliv je známá IP adresa jednotlivých účastníků komunikace a je možno sestavit IP-paket, musí být zabalen do linkového rámce, v našem případě ethernetového rámce. Tento rámec ovšem využívá adresy linkové (MAC adresa) které lze zjistit přes ARP protokol. ARP protokol je z tohoto důvodu nutný doplněk Internet protokolu, avšak není jeho součástí.

Pokud zařízení chce navázat komunikaci s jiným a zná pouze IP adresu je vyslán ARP ARP request který slouží jako dotaz na MAC adresu. Tento paket je vyslán na broadcast adresu (FF:FF:FF:FF:FF:FF) do celé sítě. Často se ARP request přirovnává k otázce „Kdo je X.X.X.X řekni Y.Y.Y.Y“ Kde X a Y jsou IP adresy tázacího a tázaného.

Po otázce přijde odpověď ARP reply „X.X.X.X is hh:hh:hh:hh:hh:hh“ kde hh je MAC adresa. Tento paket již není broadcast ale je poslán čistě tázateli Y.Y.Y.Y. Následně je do překladové ARP tabulky zapsán vztah MAC← →IP adresa.

8 Implementace IP stacku

Nejprve je nutné zmínit co vlastně IP stack znamená. Stack lze vyložit jako sada protokolů které spolupracují. Termín stack lze také vyložit jako software zpracovávající tyto protokoly.[21]

Existuje velká řada implementací IP stacků. Lze uvést open source varianty například microIP nebo lwIPstack, ovšem existují i komerční varianty a to například NicheStack TCP/IPv4 a jiné. LwIP stack lze nalézt i jako součástí middleware softwaru v softwaru STM32CubeMX.

Hotové IP stacky obsahují spousty funkcí, které nemusí být všechny využity v tomto projektu. Již napsaný kód nemusí být přehledný a může trvat delší dobu se v programu zorientovat, než implementovat vlastní. Proto jsem se rozhodl pro implementování vlastního IP stacku s vybranými funkcemi potřebné pro chod zařízení.

8.1 Inicializace

Pro inicializaci periferie Ethernet byla použita část knihovny Standard Peripheral Libraries od firmy STMicroelectronics a to v podobě souboru stm32f4xx_eth.c. Důvodem byla kladná předchozí zkušenost s tímto zdrojovým kódem a snadnější uživatelská interakce, než v případě knihovny HAL.

Inicializace se provede při zapnutí mikrokontroléru zavoláním funkce „Ethernet_init“ ve funkci main. Funkci „Ethernet_init“ lze najít v souboru Src/main.c na řádce 505. V této funkci se následně volají další součásti inicializace. Nejprve se zavolá funkce „ENET_RMII_PinInit“, která nejprve zkonfiguruje rozhraní MII aby fungovala v režimu RMII a následně inicializuje piny do potřebného stavu.

Následně je zavolána funkce „ETH_init“ ve které se pomocí funkce ETH_Init volané s parametrem struktury ETH_InitStruct a PHY adresou transcieveru LAN8742A nastaví Ethernet rozhraní. Funkci ETH_Init lze najít v souboru Src/Library/stm32f4xx_eth.c na řádce 1252, který slouží jako nízkoúrovňový ovladač pro Ethernetové rozhraní.

Po inicializování Ethernetového rozhraní je nutné inicializovat DMA deskriptory a to jak pro příjem funkcí „ETH_DMARxDescChainInit“ tak i pro vysílání pomocí funkce „ETH_DMATxDescChainInit“. Na konec se zavolá funkce „ETH_Start“ která spustí MAC vysílání na rozhraní MII, vyprázdní vysílací FIFO, spustí příjem a DMA přenos. Obě tyto funkce lze najít v souboru Src/Library/stm32f4xx_eth.c. Nyní je již Ethernetové rozhraní plně funkční.

8.2 Příjem paketů

Pro příjem rámců je ve funkci „main“ periodicky volána knihovní funkce „uint32_t ETH_HandleRxPkt(uint8_t *ppkt)“. Funkce je deklarována ve zdrojovém kódu DZM\Src\Library\stm32f4xx_eth.c. Účel funkce je překopírování přijatého rámce do aplikačního bufferu a vrácení délky rámce, která je následně uložena do proměnné FrameLength.

Po získání délky rámce je zavolána funkce „void ETH_frame_parse(uint8_t* pkt, uint32_t FrameLength)“ se vstupními argumenty a to adresou aplikačního bufferu obsahující rámec a délkou daného rámce uložené v proměnné FrameLength. Funkce je deklarována ve zdrojovém souboru DZM\Src\communication.c

V této funkci je uložena zdrojová MAC adresa pro pozdější odpověď na zprávu a následně podle pole Ethertype je rozhodnuto, zda se jedná o protokol ARP nebo IPv4.

8.3 Zpracování ARP požadavku

Pokud se pole Ethertype rovná číslu 806_{16}, indikující že se jedná o ARP protokol, je zavolána funkce „ARP_respond“ pro následné zpracování. V této funkci je následně poskládán paket, který je odpověď na ARP požadavek. Kompletní paket je následně poslán pomocí funkce „ETH_HandleTxPkt“.

8.4 Zpracování přijatého IPv4 paketu

Jestliže má pole Ethertype hodnotu 800_{16}, jedná se o paket protokolu IPv4. Typ protokolu je nutné zjistit z pole „číslo protokolu“. Protokol ICMP má číslo 1, TCP má číslo 6 a UDP má číslo 17. Pomocí této hodnoty jsou pak vybrány jednotlivé funkce pro zpracování zbytku paketu.

8.4.1 ICMP paket

Pokud je protokol ICMP je zavolána funkce „ICMP_handler“. V této funkci se zjistí kód kontrolní zprávy a podle ní se dále tento paket zpracuje. Pokud je zjištěn kód 8, neboli Echo request, je zavolána funkce Ping_reply s úkolem sestavit a poslat správnou odpověď s kódem 0 a to Echo.

8.4.2 Zpracování UDP

Protože na protokol UDP je potřeba nejen poslat odpověď, ale i zareagovat, je jeho zpracování složitější. Nejprve je zavolána funkce „UDP_handler“, ve kterém je zpracován UDP header. Pokud paket není směřován na port 50000 (příkazy) nebo 50001 (data), je zahozen.

Podle toho zda je paket směřován na port pro příjem příkazů nebo dat je zavolána odpovídající funkce.

Funkce „UDP_Command_handler“ se stará o zpracování příkazu přicházející po protokolu UDP. Nejprve je zjištěno o jaký příkaz se jedná a následně se vybere odpovídající akce k danému příkazu. Protože se UDP protokol spoléhá na zaručení doručení aplikační vrstvou, je nutné vždy vyslat odpověď, aby se doručení potvrdilo.

Po vybrání a provedení funkce je následně seskládán paket pomocí funkce ETH_frame a IPv4_packet. Tyto funkce se starají o vytvoření jednotlivých částí paketu. ETH_frame vytvoří základ Ethernet rámce a následně předá jeho konec. Na tento konec naváže funkce IPv4_packet, která vytvoří základ IPv4 paketu.

Následně je doplněn zbytek UDP datagramem přímo v originální funkci.

8.4.3 TCP spojení

Stav TCP spojení je postaven na vnitřní stavovém automatu podle specifikace RFC 793. Tato specifikace popisuje funkci TCP protokolu. Stavy jsou uloženy v proměnné TCP_status v souboru Src/communication.c. Jednotlivá jména stavů jsou uloženy ve výčtovém typu enum pojmenovaném Tcp_status, který lze najít v Src/communication.h

V této práci zařízení funguje výhradně jako TCP server a vždy čeká než jiné zařízení zinicializuje komunikaci. Z této informace je vycházeno ve všech funkcích obsluhující TCP protokol.

Po rozkódování paketu a identifikování, že se jedná o paket s TCP protokolem, je zavolána funkce TCP_handler. Uvnitř funkce jsou uloženy potřebné parametry, například číslo segmentu a příznaky, ze segmentu. Pokud je v TCP segmentu nahozen příznak SYN, jsou uloženy možnosti pro případné použití. V této fázi je kontrolován příznak RST pro případné zresetování spojení a vrácení stavového automatu do původního nastavení.

O následné komunikaci rozhodne vnitřní stav TCP spojení. Pokud je stav ve fázi „Listen“ neboli v nule, je kontrolován příznak SYN v příchozím segmentu. Při detekování příznaku je zavolána funkce „TCP_Listen_stage“. V této funkci se připraví TCP segment dle specifikace třífázového handshaku, který lze vidět na obrázku č.18. S odpovědí na přijatý příznak SYN, je navíc odesláno nastavení maximální velikosti segmentu(MSS) protějšku. Tento parametr značí maximální počet bytů v jednom TCP segmentu.

Pro vyslání segmentu je nutné nejprve poskládat Ethernetový rámec pomocí funkce „ETH_frame“, IPv4 paket funkcí „IPv4_packet“ a následně zavolat funkci „TCP_segment“, která má za úkol poskládat TCP segment.

Po odeslání odpovědi na TCP segment s příznakem SYN, je posunut stavový automat do stavu „SYN_RECIEVED“ a je očekáván v dalším segmentu příznak ACK. Přijetím zprávy s příznakem ACK je spojení považováno za „ESTABLISHED“, neboli navázané.

8.4.4 Příjem souborů TCP protokolem

Nyní jsou již očekávána data obsahující posílaný soubor. Segmenty nesměřované na funkční TCP porty(50000,50001) jsou zahozeny. Pokud je segment směřován na port 50001 je předán funkci „TCP_Recieve_data“.

V této funkci je nejprve zkontrolováno, zda segment obsahuje platná data pomocí příznaku PSH. Pokud ano, jsou tyto data přesunuty do TCP bufferu o velikosti dvaceti řádků a sloupců dle poslané hodnoty MSS. Následně je poskládána odpověď s patřičným potvrzením přijatých bytů a příznakem ACK.

Stejným způsobem jsou zpracovány další přijaté TCP segmenty až do té doby, dokud není buffer zaplněn nebo není přijat příznak FIN, značící ukončení spojení. Následně je otevřen vybraný soubor na mikroSD kartě. Pokud jde o první segment přijatý v sekvenci, je soubor otevřen v módu „FA_CREATE_ALWAYS“. Více o módech a práci s microSD kartou v kapitole **9.4.2**. Tento mód zabezpečí vytvoření nového souboru. V případě existujícího souboru je originální soubor přepsán novými daty a zmenšen na velikost nových dat. Pokud nejde o první přijatý segment, je otevřen soubor v módu „FA_OPEN_APPEND“. Tímto módem je zaručeno zapisovaní dat až za již zapsaná data. Následně jsou data v bufferu zapsána do otevřeného souboru, který je následně zavřen.

Pokud byl v segmentu přijat příznak FIN, je poslána odpověď odpovídající čtyř fázovému handshaku který lze vidět na obrázku č.19 . Stav TCP je následně změněn na požadovaný stav. Zařízení je následně připraveno na další TCP spojení.

9 Implementace externího úložiště

9.1 Úvod[27,15]

Jak je již zmíněno v první kapitole byla vybrána mikroSD karta s použitím rozhraní SDIO.

Ačkoliv je zapisování dat na SD kartu možné s vlastním vytvořeným file managment systémem, znamenalo by to obtížné až nemožné interakce s SD kartou mimo zařízení.

Standardizovaný souborový systém, se kterým umí pracovat každé moderní zařízení, umožňuje rychlé zapsání velkého množství souborů jak z počítače tak i například z chytrého telefonu bez nutnosti dalších aplikací. Tím je i umožněno externí nastavení zařízení pouhým změněním několika souborů na mikroSD kartě.

9.2 SDIO

SDIO interface byl poprvé představen v roce 2001 organizací SD association jako rozšíření stávajícího SD card standardu.

Cílem SDIO rozhraní bylo poskytnutí vysokorychlostní I/O s malou spotřebou energie pro mobilní zařízení.

9.2.1 Typy SDIO karet:

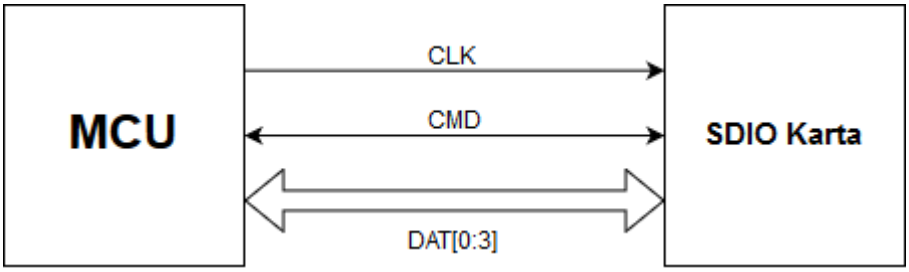
Full-Speed karta:
Podpora SPI,1-bit a 4-bit přenosového módu na rozsahu hodin 0-25MHz.Schopna přenosové rychlosti až 100Mb/s

Low-Speed karta:
Zaručená podpora pouze pro SPI a 1-bit. Maximální frekvence hodin 0-400KHz.

9.2.2 Módy mikroSD karet

1. SPI: Rychlost 20Mbps
2. 1-bit: Rychlost 25Mbps
3. 4-bit: Rychlost 100Mbps

9.2.3 Zapojení SDIO karty



Obr. 21: Blokové schéma zapojení SDIO karty[27]

V případě této diplomové práce jsem zvolil 4-bitový mód, jehož zapojení můžeme vidět na obrázku č.21.

9.3 Inicializace SDIO rozhraní

K inicializaci rozhraní SDIO byla použita knihovna HAL od firmy STMicroeletronics a software STM32Cube. Inicializace rozhraní SDIO je realizována ve knihovní funkci „BSP_SD_Init“ deklarované ve zdrojovém souboru Src/bsp_driver_sd.c na řádce 72. Nejprve se otestuje vložení mikroSD karty a následně je zavolána knihovní funkce „HAL_SD_Init“ s adresou inicializační struktury „hsd“. Struktura „hsd“ se inicializuje a nastavuje ve funkci „MX_SDIO_SD_Init“ v souboru Src/main.c na řádce 399. Funkce „MX_SDIO_SD_Init“ byla vygenerováná v softwaru STM32Cube podle nastavení v grafickém prostředí. Po korektním nastavení hodin a pinů ve funkci „HAL_SD_MspInit“ je inicializace dokončena zavoláním knihovní funkce „HAL_SD_InitCard“.

Ve funkci „HAL_SD_InitCard“ je vidět použití výchozího nastavení při inicializaci mikroSD karty. Komunikace je pouze se šířkou jednoho bitu a dělička hodin je nastavena tak, aby frekvence nepřesáhla maximální úroveň 400KHz. Pokud se při inicializaci neprojeví žádné chyby, je karta již zinicializována s uživatelskými parametry. Komunikace stále probíhá po jednom vodiči.

Pro přepnutí komunikace na 4 bitovou šířku je zavolána funkce „HAL_SD_ConfigWideBusOperation“. Jako vstupní parametrem funkce je adresa inicializační struktury z funkce „MX_SDIO_SD_Init“ a požadována šířka sběrnice.

9.4 File Allocation Table file system[17,9,16,25]

FAT file system vznik okolo roku 1980. Z počátku byl vyvinut jako jednoduchý filesystem pro diskety menší než 500KB. Časem se ukázalo že je potřeba pracovat s daty větší velikosti. Byly proto vyvinuty tři různé typy FAT file systému a to FAT12, FAT16 a FAT32. Tyto typy jsou zpětně kompatibilní.

Jméno FAT souborového systému pochází z metody organizace souborů. File allocation table, ležící na začátku paměťového prostoru a obsahuje index všech souborů v systému. Pro větší spolehlivost je tato tabulka zdvojena.

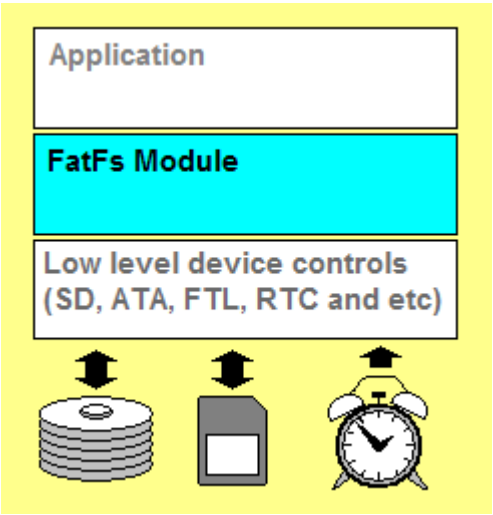
Tab. 8: Ilustrace organizace pomocí FAT souborového systému [16]

Partition boot sektor	FAT1	FAT2 (kopie FAT 1)	Kořenový adresář	Ostatní soubory a složky

9.4.1 Implementace knihovny FatFs[8]

Modul FatFs začal jako osobní projekt jeho autora ChaNa. Software FatFs je podle licenční smlouvy považován za open source s možností kompletní modifikace, pokud je zachováno originální oznámení o autorských právech.

Knihovna FatFs je označován jako „middleware“, neboli prostředník. Jak je patrné z obrázku č.22 FatFs je prostředník mezi aplikací a nízko úrovňovým řízením zařízení, v našem případě SD karta. Spojení mezi SD kartou a FatFs modulem je zprostředkováno knihovnou HAL od firmy STMicroelectronics.



Obr. 22: Znázornění funkce FatFs modulu[8]

Proto je nutné v kódu propojit nízkoúrovňové ovladače s modulem FatFs. Propojení je zprostředkováno funkcí vygenerovanou softwarem STM32Cube uint8_t MX_FATFS_Init(void) ve které je zavolána knihovní(FatFs) funkce uint8_t FATFS_LinkDriver(const Diskio_drvTypeDef *drv, char *path). Vstupními parametry této funkce je adresa SD ovladače a logická cesta k paměťovému médiu, deklaraci této funkce lze nalézt ve zdrojovém souboru DZM\Middlewares\Third_Party\FatFs\src\ff_gen_drv.c. Funkce MX_FATFS_Init je deklarována ve zdrojovém souboru DZM\Src\fatfs.c. Ve výsledku se modul FatFs nestará o provedení nízkoúrovňových ovladačů.

9.4.2 Použití modulu FatFs[8]

Deklarace knihovních funkcí popsaných níže lze nalézt ve zdrojovém souboru DZM\Middlewares\Third_Party\FatFs\src\ff.c.

Nejprve je nutné mikroSD kartu namountovat, to je provedeno pomocí knihovní funkce „FRESULT f_mount (FATFS* fs,const TCHAR* path,BYTE opt)“ zavolané hned po inicializaci SDIO rozhraní a při znovu vložení karty. Vstupní parametry této funkce je objektová struktura souborového systému která bude použita. Je nutné alokovat dostatek místa v paměti pro objektovou strukturu. Dalším vstupním parametrem funkce je logické číslo ovladače, tento parametr je potřeba při použití více ovladačů. Posledním parametrem se určuje zda dojde k připojení média ihned, nebo až při příští operaci s médiem.

V knihovně se často pracuje se strukturou FIL. Definici lze najít v hlavičkovém souboru DZM\Middlewares\Third_Party\FatFs\src\ff.h na řádce 155. Struktura obsahuje identifikátor objektu, stavové příznaky souboru, chybový kód, zapisovací/čtecí ukazatel a další.

Po připojení média je již možné použít knihovní funkci FRESULT f_open (FIL* fp,const TCHAR* path,BYTE mode) k otevření souboru na médiu. Funkce ke své práci potřebuje tři vstupní parametry a to adresu prázdného objektu typu FIL, jméno souboru a mód přístupu k souboru. Tyto módy lze kombinovat logickou funkci | a jsou následující:

- FA_READ – Data lze pouze číst ze souboru
- FA_WRITE – Data lze pouze zapisovat
- FA_OPEN_EXISTING – Otevře pouze existující soubor, nahlásí chybu pokud soubor neexistuje (Výchozí mod)
- FA_CREATE_NEW – Vytvoří nový soubor, funkce nahlásí chybu FR_EXIST pokud soubor již existuje
- FA_CREATE_ALWAYS – Vždy vytvoří nový soubor, starý soubor se stejným jménem je přepsán a zmenšen na velikost zapsaných dat
- FA_OPEN_ALWAYS – Otevře existující soubor a nebo vytvoří nový pokud neexistuje soubor se zadaným jménem
- FA_OPEN_APPEND – Stejný efekt jako předchozí mód, s výjimkou že ukazatel pro čtení a zápis je posunut na konec souboru

Pro čtení z média se použije knihovní funkce FRESULT f_read (FIL* fp, void* buff, UINT btr, UINT* br). Funkci je nutné zavolat se čtyřmi parametry. Prvním je adresa objektu typu FIL, použitým ve funkci f_open. Následně je nutný buffer do kterého se budou data číst, počet bytů k přečtení a adresa proměnné do které se uloží počet přečtených bytů.

Lze také na médium zapisovat pomocí knihovní funkce FRESULT f_write (FIL* fp, const void* buff, UINT btw, UINT* bw). Stejně jako u čtecí funkce, první parametr je adresa objektu typu FIL, použitým ve funkci f_open. Dalšími parametry je odkaz na buffer s daty k zapsání, počet bytů k zapsání a adresu proměnné do které se uloží počet zapsaných bytů.

Pokud je počet zapsaných bytů menší než počet bytů v datovém bufferu, znamená to že je paměťové médium plné.

V případě že se nechce číst nebo zapisovat od začátku je možné posunout ukazatel pro čtení a zápis pomocí knihovní funkce FRESULT f_lseek (FIL* fp, FSIZE_t ofs). Funkce je zavolána s adresou objektu typu FIL, ve kterém je nutno ukazatel posunout. Druhým parametrem je počet bytů o kolik se má daný ukazatel posunout.

Po provedení operace je nutné ho následně zavřít knihovní funkcí FRESULT f_close (FIL* fp). Pokud soubor není uzavřen je při pokusu o otevření nahlášena chyba FR_LOCKED.

10 Ovládání zařízení

Zařízení příjímá příkazy po protokolu UDP na portu 50000. Po přijmutí každého příkazu je nutné vyčkat na odpověď opět na portu UDP 50000, pro ujištění že se zpráva doručila.

10.1 Možnosti ovládání

Ačkoliv je zařízení koncipováno pro ovládání ze vzdáleného místa pomocí lokální sítě, je možné i ovládání zařízení přímou manipulací.

Při zapojení zařízení do síťového přepínače (switche) je možné ovládání z celé lokální sítě.

10.2 Lokální ovládání

Pro lokální ovládání zvukového modulu slouží čtyři tlačítka, umístěná na rozšiřujícím modulu. Tato tlačítka slouží pro základní ovládání přehrávače – spuštění a zastavení přehrávání, výběr předchozí skladby, výběr následující skladby a pozastavení přehrávání. Jednotlivé funkce tlačítek lze vidět v tabulce číslo 9. Jako indikace stavu zvukového modulu slouží čtyři LED diody, které indikují zapnutí zařízení, přehrávání a ukládání souboru ze sítě. Jejich podrobný popis je umístěn v tabulce číslo 10. Umístění tlačítek a LED diod je zřejmé z přílohy B.

Tab. 9: **Funkce tlačítek**

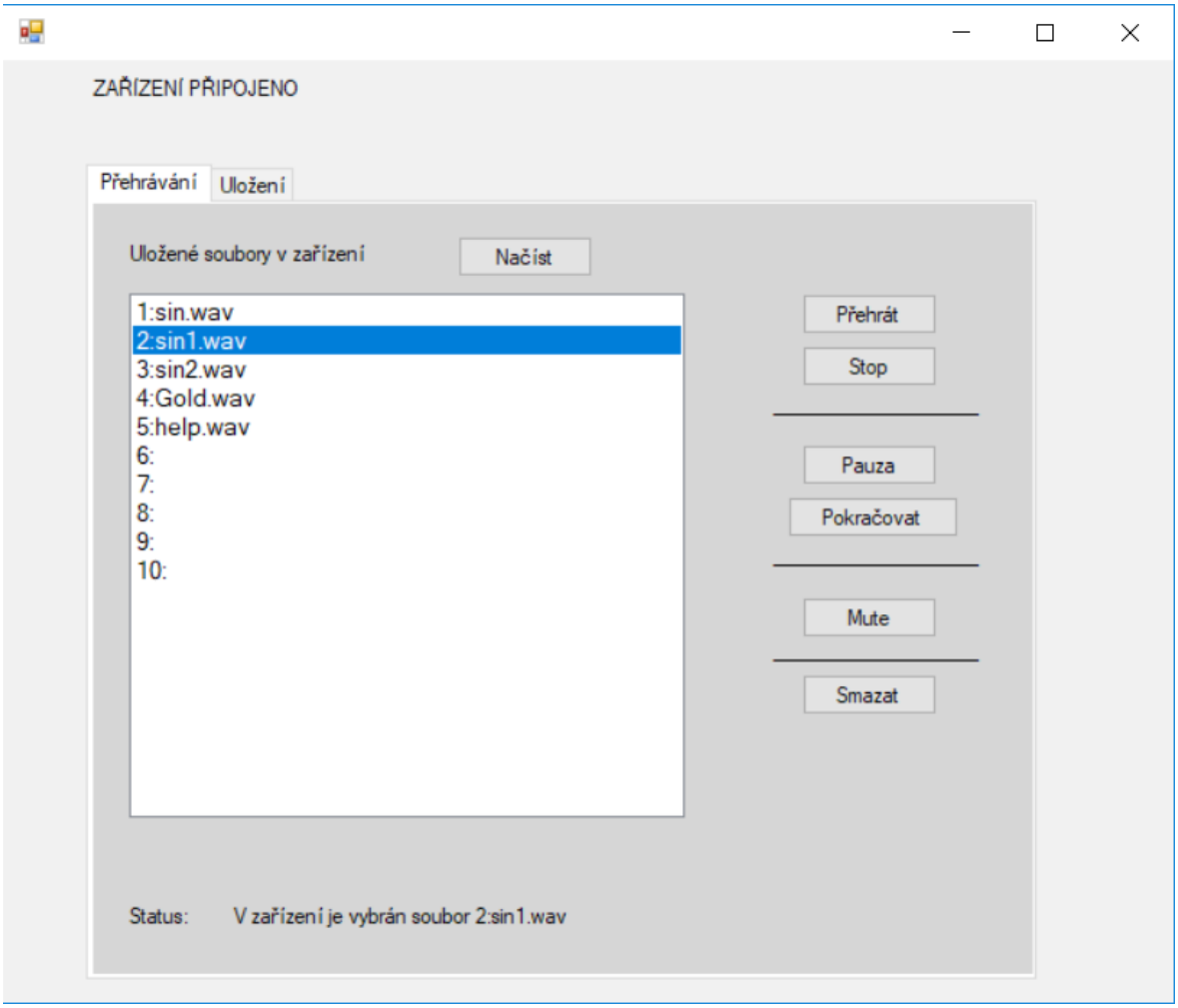
Tlačítko	Funkce
S1	Výběr předchozí skladby
S2	Zastavení nebo spuštění skladby
S3	Výběr následující skladby
S4	Pokračování přehrávání od posledního místa

Tab. 10: **Funkce indikačních LED diod**

Indikační LED	Funkce
LED1	Indikace napájení
LED2	Nevyužita
LED3	Indikace přehrávání
LED4	Indikace přenosu souboru
LED5	Nevyužita

10.3 Aplikace pro ovládání zvukového modulu

Aplikace pro ovládání zvukového modulu ze standardního PC byla vytvořena ve vývojovém prostředí Visual Studio 2019 od fimy Microsoft v programovacím jazyku C#. Zdrojový kód lze najít na přiloženém DVD disku ve složce PC_Aplikace_kod. Cílem bylo vytvořit jednoduchý, uživatelsky příjemný program pro ukládání a přehrávání zvukových souborů. Při vývoji bylo vycházeno z příkladu TCP přenosu dat na adrese [7]. Formát zvukových souborů je omezen pouze na typ souboru *.wav. Program obsahuje 2 záložky pro přehrávání souborů již uložených ve zvukovém zařízeni a pro ukládání zvukových souborů do zařízení. Po startu se aplikace snaží připojit k danému zvukovému modulu. Po úspěšném připojení je uživatel informován zprávou nahoře ve formuláři a automaticky se načte a zobrazí seznam uložených souborů v zařízení na záložce “Přehrávání“ - viz obrázek číslo 23.



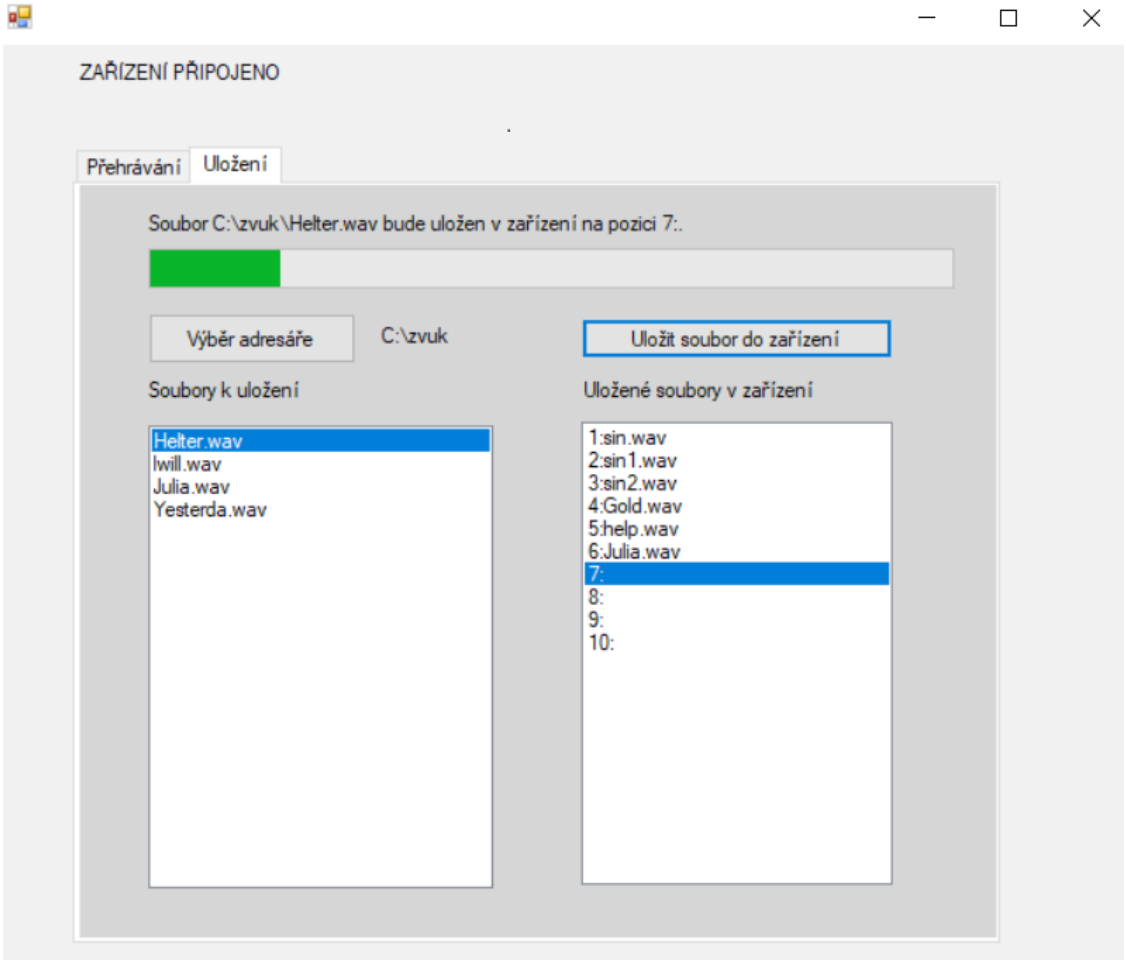
Obr. 23: Snímek obrazovky aplikace na záložce přehrávání

Uložené soubory jsou zobrazeny ve formátu pozice:nazev_souboru. Přiřazení každému souboru pevnou pozici umožňuje ovládání přehrávání přímo ze zvukového modulu tlačítky. Na zobrazeném formuláři je umístěn seznam uložených souborů v zařízení a několik ovládacích tlačítek. V seznamu se lze pohybovat pomocí kurzorových tlačítek nebo myši. Jednotlivá tlačítka mají funkce, popsané v tabulce číslo 11.

Tab. 11: **Funkce tlačítek programu**

Tlačítko	Funkce
Přehrát	Přehrávání označeného souboru v seznamu
Stop	Zastavení přehrávání souboru
Pauza	Přerušení přehrávání souboru s možností pokračovat v reprodukci
Pokračovat	Pokračovat v přehrávání souboru po volbě Pauza
Mute	Vypnutí zvuku při přehrávání
Smazat	Smazání označeného souboru v seznamu

Pro možnost ukládání zvukových souborů do zařízení je nutné zvolit záložku „Uložení“ zobrazenou na obrázku číslo 24.



Obr. 24: Grafické prostředí pro nahrávání souborů

Na tomto formuláři se v pravé části automaticky zobrazí soubory uložené v zařízení. V levé části je nutno vybrat adresář na PC, ve kterém se nacházejí požadované zvukové soubory. Pak se zobrazí seznam souborů z tohoto adresáře. Po výběru souboru k uložení a pozice v zařízení, kam se má daný soubor uložit, lze tlačítkem „Uložit soubor do zařízení“ vybraný soubor odeslat do zvukového modulu.

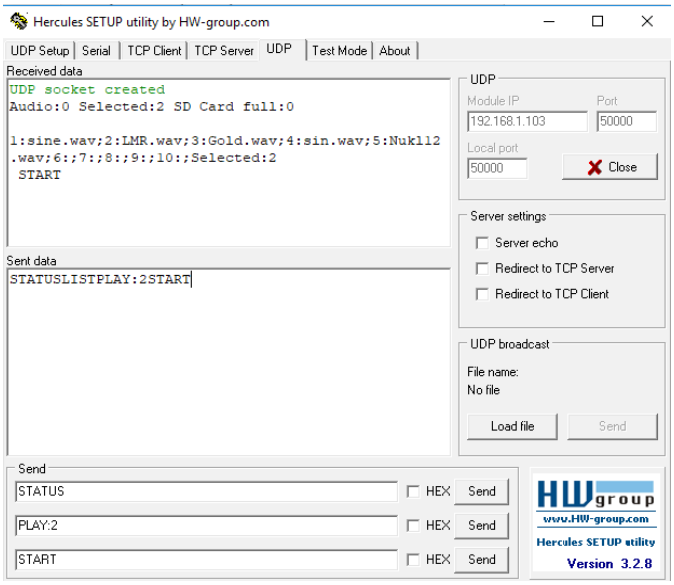
Postup přenosu souboru je indikován ukazatelem průběhu.

10.4 Aplikace třetích stran

10.4.1 Ovládání z operačního systému Windows

Pro ovládání zařízení je nutné používat aplikaci která je schopna posílat a přijímat UDP datagramy. Lze nalézt desítky aplikací určené k posílání síťových paketů. Jak proprietární, tak i otevřený software. Jeden z volně dostupných je aplikace Hercules SETUP. K dispozici je jako freeware na adrese: (https://www.hw-group.com/cs/software/aplikace-hercules-setup).

Na obrázku č. 25 je vidět ukázka aplikace Hercules při ovládání zařízení.



Obr. 25: Příklad ovládání přístroje z aplikace Hercules

10.4.2 Ovládání z operačního systému Linux

Posílání UDP datagramů je v operačním systému Linux s integrovaným příkazovým procesorem Bash velice ulehčeno. Do terminálu lze zadat příkaz: „echo -n "XXXX" >/dev/udp/YYY.YYY.YYY.YYY/50000“. Za písmena X se doplní žádaný příkaz a za písmena Y je nutno dosadit IP adresu zařízení. O následné poslání se postará samotný operační systém.

10.5 Příkazy

Příkazy jsou zadávány ve formě řetězce ASCII znaků ukončených binární nulou.

Zpracování všech příkazů které se posílají protokolem UDP je možné najít ve funkci UDP_Command_handler ve zdrojovém souboru Src/communication.c na řádce 451. Kromě několika výjimek které jsou níže popsány odpovídá zařízení stejným příkazem jako byl přijat. Při nerozpoznání příkazu je poslána zpět zpráva „Wrong command“.

Pokud je potřeba doplnit příkaz, který zařízení neobsahuje, je nutné pouze přidat hodnotu příkazu do define v communication.h a následně přidat položku do přepínače switch v souboru Src/communication.c na řádce 532. Odpovídající rozšíření programu je nutné provést i na straně ovládací aplikace v jazyku C#.

10.5.1 Příkazy pro přehrávání

START

Po poslání příkazu start se nejprve dají všechny proměnné týkající se přehrávání do výchozího nastavení, aby se zajistilo že soubor bude přehrán od začátku a následně se spustí přehrávání vybraného zvukového souboru, který byl vybrán pomocí příkazu PLAY,NEXT a nebo PRE. Pokud není zapnut mód přehrání všech zvukových souborů příkazem PLAY, je po skončení zvukového souboru přehrání ukončeno.

STOP

Příkaz STOP pozastaví právě probíhající přehrávání. Je možné navázat na takto pozastavený hudební soubor pomocí příkazu RESUME.

RESUME

Příkaz RESUME navazuje na příkaz STOP a spouští přehrávání z posledního místa kde bylo pozastaveno.

NEXT

Příkaz NEXT vybere další zvukový soubor ze seznamu jako aktuální. Zařízení odpoví na tento příkaz zprávou „Selected:X“, kde X je číslo vybrané položky.

PRE

Příkaz PRE vybere předchozí zvukový soubor ze seznamu jako aktuální. Zařízení odpoví na tento příkaz zprávou „Selected:X“, kde X je číslo vybrané položky.

LIST

Příkazem LIST se dá zařízení pokyn k vypsání seznamu zvukových souborů, ze kterých je možné vybírat ostatními příkazy. Pro lepší čitelnost jsou zvukové soubory očíslovány a odděleny středníkem. Tato odpověď je více popsána v kapitole 10.6.

MUTE

Příkazem MUTE se umlčí přehrávání zvukového souboru. Ačkoliv je zvuk umlčen a na výstupu kodeku je 0, přehrávání souboru pokračuje dál. Při vypnutí umlčení se pouze změní hodnota na výstupu na data ze souboru.

UNMUTE

Příkaz UNMUTE vypne umlčení.

PLAY

Příkaz PLAY vybere zvukový soubor ze seznamu. Syntaxe příkazu je „PLAY:XX“ bez uvozovek, kde XX je číslo vybraného zvukového souboru. Pokud je číslo menší než 10 je nutné číslo uvádět ve tvaru 0X. Validní znaky jsou buď ASCII čísla a nebo znak A. Při poslání příkazu „PLAY:A“ se vybere první zvukový soubor jako aktivní a zároveň se nahodí příznak pro postupné přehrání všech zvukových souborů v seznamu. Zařízení odpoví na tento příkaz zprávou „Selected:X“, kde X je číslo vybrané položky.

10.5.2 Příkazy pro práci se soubory

DEL

Příkaz DEL kompletně smaže aktuální seznam souborů a všechny zvukové soubory na mikroSD kartě. Servisní soubory jako IP, jsou zachovány.

FILE

Příkaz FILE je použit před přenosem dat pro určení jména souboru. Syntaxe příkazu je následující „FILE:XXXXXXXX.YYY“, kde XX je jméno souboru a YY je přípona. Z důvodu omezení fat souborového systému je omezení na délku jména osm znaků a délky přípony tři znaky.

10.5.3 Servisní příkazy

DZMIP

Příkaz DZMIP je očekáván v broadcast zprávě. Odpověď na tento příkaz je nastavená IP adresa zařízení. Pomocí tohoto příkazu lze rychle zařízení najít. Tato odpověď je více popsána v kapitole 10.6.

STATUS

Příkaz STATUS pošle aktuální stav zařízení. Posílá se zda zařízení přehrává, který soubor je vybrán a zda je mikroSD karta zaplněna. Tato odpověď je více popsána v kapitole 10.6.

10.6 Posloupnost příkazů pro přehrávání

Pokud není známá IP adresa zařízení, začíná se příkazem DZMIP na broadcast adresu sítě a UDP port 50000. Je nutné zároveň poslouchat na UDP portu 50000 pro odpověď ve tvaru „DZMIP:XXX.XXX.XXX.XXX“, kde XXX je IP adresa na kterou zařízení reaguje.

Aby se dal vybrat který zvukový soubor se má přehrát, je nutné poslat příkaz LIST, na který přijde odpověď ve tvaru „1:XXXXXXXX.YYY;2:XXXXXXXX.YYY;…...“. Pokud jsou například v zařízení uloženy dva testovací signály a to sine1.wav a sine3.wav na pozicích 1 a 3, bude odpověď vypadat následovně: „1:sine1.wav;2:;3:sine3.wav;”. Na pozici 2 není nic uloženo, proto je poslána prázdná položka.

Příkazem STATUS se zjistí jaký soubor je nyní vybrán a zda zařízení přehrává tento soubor. Zpět přijde zpráva ve tvaru „Audio:0 Selected:1 SD Card full:0“. Nula na pozici Audio značí, že zařízení nepřehrává. Jednička za Selected indikuje vybraný soubor a SD Card full nulou indikuje že není mikroSD karta plná.

Dalším krokem je vybrat zvukový soubor který chceme přehrát. Například pro přehrání souboru na pozici 3 a to sine3.wav, je možné použít dva příkazy. Vybrat číslo souboru v seznamu lze použitím příkazu „PLAY:3“, zpět přijde odpověď „Selected:3“ jako potvrzení. Alternativa je použití příkazu NEXT, který zvětší vybraný soubor o jedničku. Protože byl vybrán soubor číslo jedna, přijde zpět odpověď „Selected:1“, k dosažení vybrání třetího souboru je nutné poslat tento příkaz znovu.

Po vybrání požadovaného souboru již stačí použít příkaz START.

10.7 Nahrávání souborů

Ve výchozím nastavení je maximální počet položek v seznamu deset. Lze tento počet navýšit přepsáním definice „Playlist_count“ v hlavičkovém souboru DZM\Src\ communication.h na řádce 35, na požadovanou hodnotu.

Postup nahrávání souborů je podobný jako u přehrávání. Je nejprve nutno vybrat pozici na kterou se soubor má zapsat. Je možno vybrat jak prázdné místo v seznamu, tak již obsahující soubor. Při využití předchozího příkladu, byla vybrána 3. pozice v seznamu která již obsahuje soubor „sine3.wav“. Dalším krokem je poslání příkazu „FILE:test.wav“, který vymaže z mikroSD karty soubor který byl na 3. pozici (sine3.wav). Zároveň připraví že příští přijatý soubor se uloží na tuto pozici se jménem „test.wav“. Tato aktualizace seznamu souborů se zapíše i na mikroSD kartu aby tato změna nebyla ztracena při restartu.

Dalším krokem je poslání souboru po TCP protokolu na port 50001. Soubor je poslán bez další redundance a uložen na mikroSD kartu. O veškeré zabezpečení přenosu je zařízeno samotným protokolem TCP.

O příjem dat se stará funkce „TCP_Recieve_data“ v souboru Src/communication.c na řádce 910.

10.8 Nastavení IP adresy

IP adresa je při každém spuštění a vložení mikroSD karty načtena ze souboru „IP“. Obsahem tohoto souboru je IP adresa ve formátu „XXX.XXX.XXX.XXX“. Je nutné tento soubor připravit při první instalaci zařízení. Přečtení této IP adresy je provedeno funkcí Read_IP_SDIO ve zdrojovém souboru file_operations.c na řádce 11.

10.9 Offline nastavení seznamu zvukových souborů

Pro uchování seznamu zvukových souborů při ztrátě napájení je při každé změně tento seznam uložen na mikroSD kartu do souboru „Playlist.txt“. Při manuálním nahrání souborů je nutné tento soubor upravit a doplnit jména souborů na patřičné pozice. Seznam zvukových souborů je načten při každém resetu zařízení a vložení mikroSD karty.

11 Použitý software

11.1 Konfigurační software

11.1.1 STM32CubeMX

V práci byl použit grafický nástroj STM32CubeMX verze 5.0.1 pro jednoduché nakonfigurování periferií mikrokontroléru. Program je zdarma k dispozici na oficiálních stránkách https://www.st.com/en/development-tools/stm32cubemx.html. Pro vývoj aplikace byl využit firmware verze FW_F4 V1.23.0 pro řadu mikrokontrolérů F4.

11.2 Vývojová prostředí

11.2.1 TrueSTUDIO

K vyvíjení softwaru pro mikrokontroler bylo použito vývojové prostředí TrueStudio verze 9.0.1 zakoupené firmou STMicroelectronis. Tím je zaručena kompatibilita s produkty STM32 a se softwarem od stejné firmy, například zmíněný STM32CubeMX. Vývojové prostředí je k dispozici zadarmo na stránce: https://atollic.com/truestudio/.

11.2.2 Microsoft Visual Studio

Pro vývoj aplikace, v jazyce C#, na operační systém Windows bylo použito vývojové prostředí Visual Studio od firmy Microsoft. Community verze Visual Studia je k dispozici zadarmo na stránkách: https://visualstudio.microsoft.com/.

11.3 Software pro návrh zařízení

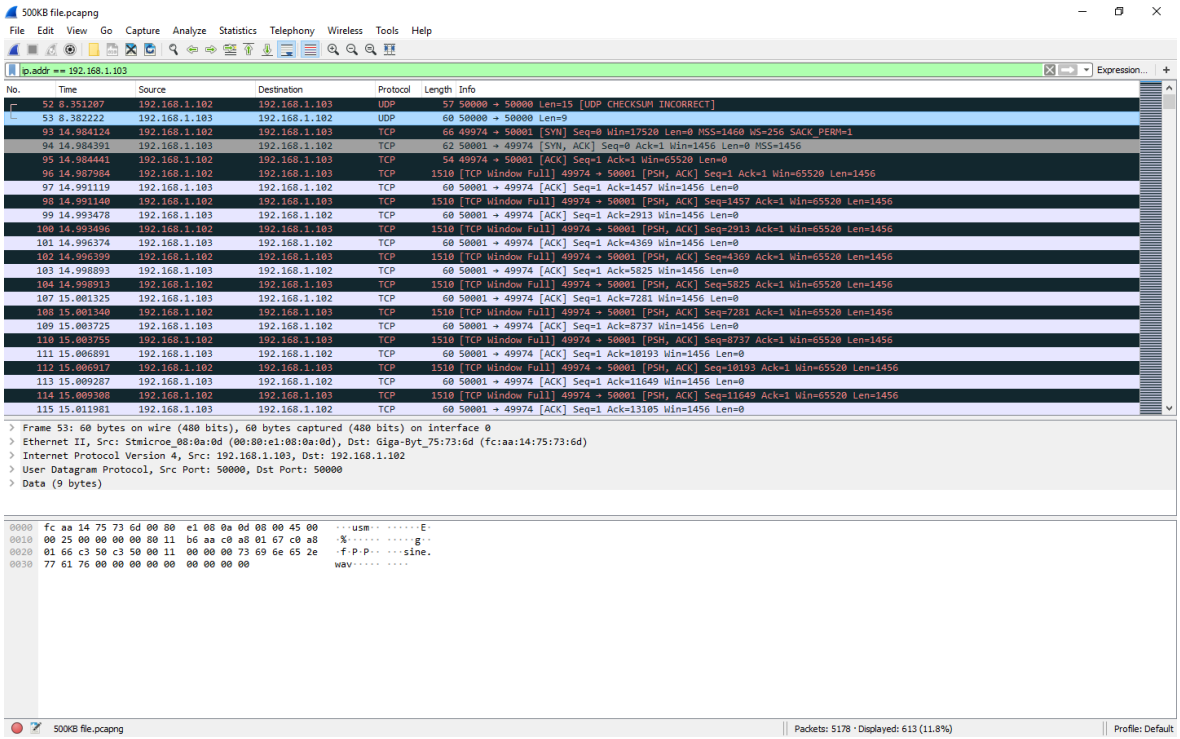
11.3.1 Eagle

Pro návrh rozšiřujícího modulu byl použit návrhový software Eagle verze 7.4.0 Light. Ačkoliv je zkušební verze omezena pouze dvěma vrstvami a maximální velikostí desky plošných spojů, pro účely této práce byla dostačující. Zkušební verzi softwaru lze nalézt na stránce: https://www.autodesk.com/products/eagle/overview.

11.4 Software pro ladění

11.4.1 Wireshark

Při ladění síťové komunikace zařízení s počítačem byl použit program Wireshark verze 3.0.0 pro zachytávání paketů a analyzování protokolů. Program Wireshark lze stáhnout zadarmo na oficiálních strankách: https://www.wireshark.org/. Pomocí snadné filtrace je možné snadno zachytit pouze potřebné pakety.



Obr. 26: Snímek obrazovky aplikace Wireshark

Závěr

Cílem diplomové práce byla realizace přehrávače zvukových souborů s jednoduchým ovládáním pro domácí použití. Zvukový modul byl realizován ve formě funkčního prototypu. Pro vytvoření prototypu byl zvolen vývojový kit od firmy STMicroelectronics, obsahující mikrokontrolér STM32F429ZI. Ten byl doplněný rozšiřujícím modulem, na kterém jsou umístěny chybějící komponenty, nutné k realizaci přehrávače. Zapojení rozšiřujícího modulu je na obr. XY. Firmware pro daný mikrokontrolér byl vytvořen v jazyce C ve vývojovém prostředí Atollic TrueStudio. Obsahuje modul pro komunikaci s nadřazeným zařízením a ukládání zvukových souborů na externí paměť. Soubory jsou ukládány na externí paměť vyjímatelnou mikroSD kartu. Další modul umožňuje vlastní reprodukci zvukových souborů. Zařízení lze ovládat lokálně, pomocí čtyř tlačítek, nebo z nadřazeného zařízení pomocí definovaných příkazů pro řízení, poslaných po lokální síti. Příkazy pro ovládání modulu z nadřazeného zařízení byly v práci kompletně zdokumentovány spolu s očekávanou posloupností pro různé akce. Je tedy možné realizovat návrh vlastní aplikace. Součástí diplomové práce je ovládací program pro operační systém Windows, napsaný v programovacím jazyce C#.

V práci byly porovnány různé zvukové formáty a protokoly pro přenos zvukových souborů v kapitolách 3 a 4. V kapitolách 5,7 a 8 byly popsány možnosti přenosu souborů a nástroje pro komunikaci mezi zařízeními. Kapitola 9 pojednává o možnosti ukládání souborů na externí úložiště. V poslední části práce je popsáno softwarové vybavení a možnosti ovládání zvukového modulu.

Další možnosti vývoje zvukového modulu jsou vytvoření zařízení bez vývojového Nucleo kitu, přidání obvodu pro digitální ovládání hlasitosti nebo možnost přímého streamování dat v reálném čase. Jisté zlepšení přenosové rychlosti by se dalo dosáhnout optimalizací nízkoúrovňových ovladačů Ethernetových knihoven na použití zřetězených deskriptorů. Dalším možným rozšířením je přidání možnosti přehrát zvukové soubory o jiném formátu než je .wav.

Seznam literatury a informačních zdrojů

[1] *Audio Codec '97* [online]. Revision 2.3. Berkeley (Kalifornie): Intel Corporation, 2002 [cit. 2019-05-23]. Dostupné z: http://www-inst.eecs.berkeley.edu/~cs150/Documents/ac97_r23.pdf

[2] Bezztrátový formát - co to je?: Bezztrátová komprese. *FLIPPWERWORLD* [online]. 2019 [cit. 2019-05-26]. Dostupné z: https://cs.flipperworld.org/pc/bezztratovy-format-co-to-je-bezztratova-komprese

[3] CS4344/5/8: 10-Pin, 24-Bit, 192 kHz Stereo D/A Converter [online]. Cirrus Logic [cit. 2019-05-26]. Dostupné z: https://cz.mouser.com/datasheet/2/76/CS4344-45-48_F2-1141644.pdf

[4] Ethernet frame. *Study CCNA* [online]. [cit. 2019-05-23]. Dostupné z: https://study-ccna.com/ethernet-frame/

[5] FAIRHURST, Godred. Address Resolution Protocol. *Electronics Research Group* [online]. Aberdeen: Electronics Research Group, 2005 [cit. 2019-05-22]. Dostupné z: https://erg.abdn.ac.uk/users/gorry/course/inet-pages/arp.html

[6] GREGAR, Petr. *Zvukový formát MP3* [online]. Pardubice, 2007 [cit. 2019-05-26]. Dostupné z: https://dk.upce.cz/bitstream/handle/10195/24757/GregarP_Zvukovy%20format_JH_2007.pdf;jsessionid=7E814B11C6BA4F2067712072804C6132?sequence=1. Bakalářská práce. Univerzita Pardubice.

[7] HESAMI, Amir. Sending Files using TCP. *Code Project* [online]. Toronto Ontario, 2009 [cit. 2019-05-27]. Dostupné z: https://www.codeproject.com/Articles/32633/Sending-Files-using-TCP

[8] CHAN. FatFs - Generic FAT Filesystem Module. *The Electronic Lives Manufacturing* [online]. [cit. 2019-05-22]. Dostupné z: http://elm-chan.org/fsw/ff/00index_e.html

[9] CHAN. FAT Filesystem. *The Electronic Lives Manufacturing* [online]. 2017 [cit. 2019-05-22]. Dostupné z: http://elm-chan.org/docs/fat_e.html

[10] CHLUBNÝ, Ing. Jaroslav. Základní zvukové formáty. *Internetový portál COPTEL* [online]. COPTEL, 2013 [cit. 2019-05-26]. Dostupné z: https://coptkm.cz/portal/?action=2&doc=40425&docGroup=-1&cmd=0&instance=1

[11] *I2S bus specification* [online]. Revize 18. Philips Semiconductor, 1986 [cit. 2019-05-22]. Dostupné z: https://www.sparkfun.com/datasheets/BreakoutBoards/I2SBUS.pdf

[12] KABELOVÁ, Alena a Libor DOSTÁLEK. *Velký průvodce protokoly TCP/IP a systémem DNS*. 5., aktualiz. vyd. Brno: Computer Press, 2008. ISBN 978-802-5122-365.

[13] KATZ, David a Rick GENTILE. *Embedded Media Processing*. 2nd Edition. Saint Louis (Missouri): Newnes, 2005. ISBN 9780750679121.

[14] *LAN8742A/LAN8742Ai: Small Footprint RMII 10/100 Ethernet Transceiver with HP Auto-MDIX and flexPWR® Technology* [online]. Microchip, 1989 [cit. 2019-05-26]. Dostupné z: http://ww1.microchip.com/downloads/en/DeviceDoc/DS_LAN8742_00001989A.pdf

[15] LI, Yannik. SD and SDIO. *Yannik's Home Page* [online]. 2016 [cit. 2019-05-23]. Dostupné z: https://yannik520.github.io/sdio.html

[16] LSOFT TECHNOLOGIES INC. FAT File Systems. FAT32, FAT16, FAT12. *NTFS.com* [online]. [cit. 2019-05-22]. Dostupné z: http://www.ntfs.com/fat-systems.htm

[17] Microsoft Corporation. *Microsoft Extensible Firmware Initiative FAT32 File System Specification: FAT: General Overview of On-Disk Format* [online]. Verze 1.03. Washington: Microsoft Corporation, 2000 [cit. 2019-05-23]. Dostupné z: https://staff.washington.edu/dittrich/misc/fatgen103.pdf

[18] Nistor, F.; Colceriu, D. Automotive ENET Interface Broadcast. *Today Software Magazine* [Online]. https://www.todaysoftmag.com/article/2427/automotive-enet-interface-broadcast (accessed May 22, 2019).

[19] PETERKA, Jiří. Část XX.: Příběh Ethernetu. *PC World*. 2006, (12).

[20] PETERKA, Jiří. Filosofie TCP/IP (II.). *CHIPweek*. 1996, (31), 13.

[21] PETERKA, Jiří. Protocol stack. *CHIPWeek*. 1995,(18), 33.

[22] PETERKA, Jiří. Protokoly TCP/IP (I.). *CHIPweek*. 1996, (32), 13

[23] PETERKA, Jiří. Síťový model TCP/IP. *Computerworld*. 1992, (31).

[24] PRAŽÁK, Daniel. Audio - Rozdíl mezi FLAC a MP3. *Hi-Fi poradna* [online]. 2015 [cit. 2019-05-26]. Dostupné z: https://www.hifiporadna.cz/audio-rozdil-mezi-flac-a-mp3-a464

[25] Přehled systémů souborů FAT, HPFS a NTFS. *Podpora Microsoftu* [online]. Redmond (Washington): Microsoft [cit. 2019-05-22]. Dostupné z: https://support.microsoft.com/cs-cz/help/100108/overview-of-fat-hpfs-and-ntfs-file-systems

[26] *RM0090 Reference manual: STM32F405/415, STM32F407/417, STM32F427/437 andSTM32F429/439 advanced Arm®-based 32-bit MCUs* [online]. Revize 18. STMicroelectronics, 2019 [cit. 2019-05-22]. Dostupné z: https://www.st.com/content/ccc/resource/technical/document/reference_manual/3d/6d/5a/66/b4/99/40/d4/DM00031020.pdf/files/DM00031020.pdf/jcr:content/translations/en.DM00031020.pdf

[27] SDIO/iSDIO. *SD Association* [online]. San Ramon (California) [cit. 2019-05-26]. Dostupné z: https://www.sdcard.org/developers/overview/sdio/index.html

[28] SIMMONS, M. a Microchip Technology Inc. *Ethernet Theory of Operation* [online]. Chandler (Arizona): Microchip Technology, 2008 [cit. 2019-05-23]. Dostupné z: http://ww1.microchip.com/downloads/en/appnotes/01120a.pdf

[29] *STM32F427xx STM32F429xx: 32b Arm® Cortex®-M4 MCU+FPU, 225DMIPS, up to 2MB Flash/256+4KB RAM, USB OTG HS/FS, Ethernet, 17 TIMs, 3 ADCs, 20 com. interfaces, camera & LCD-TFT* [online]. STMicroelectronics, 2018 [cit. 2019-05-26]. Dostupné z: https://www.st.com/en/microcontrollers-microprocessors/stm32f429zi.html

[30] *TDA2822D: DUAL LOW-VOLTAGE POWER AMPLIFIER* [online]. Ženeva: STMicroelectronics, 2003 [cit. 2019-05-26]. Dostupné z: https://www.st.com/resource/en/datasheet/tda2822d.pdf

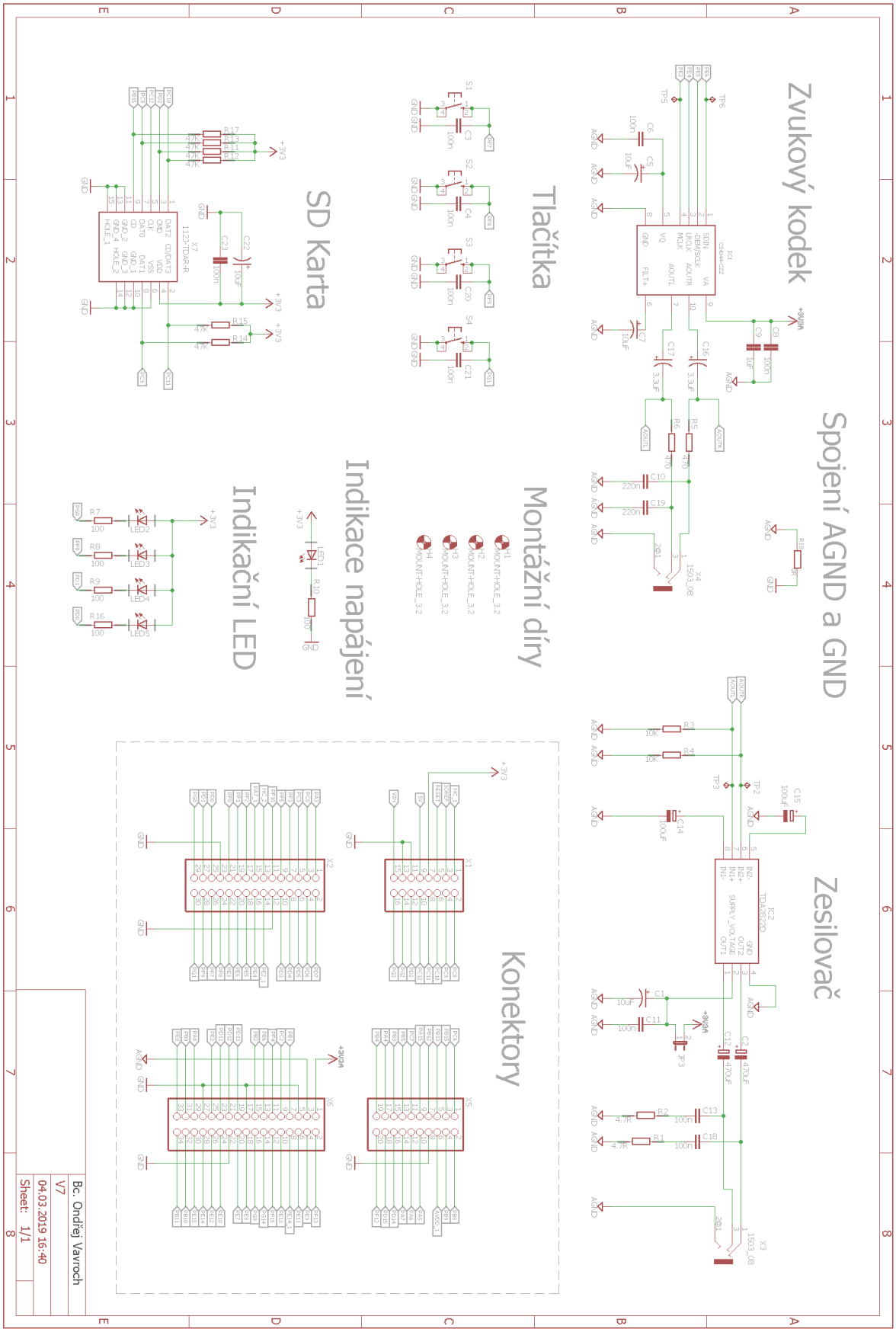
[31] *Time Division Multiplexed Audio Interface: A Tutoria* [online]. Cirrus Logic, 2006 [cit. 2019-05-23]. Dostupné z: https://d3uzseaevmutz1.cloudfront.net/pubs/appNote/AN301REV1.pdf

[32] *UM1974 user manual: STM32 Nucleo-144 boards* [online]. Rev 7. Ženeva: STMicroelectronics, 2017 [cit. 2019-05-26]. Dostupné z: https://www.st.com/content/ccc/resource/technical/document/user_manual/group0/26/49/90/2e/33/0d/4a/da/DM00244518/files/DM00244518.pdf/jcr:content/translations/en.DM00244518.pdf

[33] Zvuk – základní pojmy. *Aldebaran* [online]. Praha: Aldebaran Group for Astrophysics [cit. 2019-05-26]. Dostupné z: https://www.aldebaran.cz/onlineskola/etapy/zvuk/

Přílohy

Příloha A – Schéma rozšiřujícího modulu



Příloha B – Layout desky plošných spojů