VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

VIZUALIZACE PARAMETRŮ MNOHAKANÁLOVÉHO ZVUKOVÉHO SYSTÉMU V INTERNETOVÉM PROHLÍŽEČI

PARAMETER VISUALIZATION OF MULTICHANNEL AUDIO SYSTEM IN WEB BROWSER

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. MARTIN LACH**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. IGOR SZŐKE, Ph.D.**
SUPERVISOR

BRNO 2019

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav počítačové grafiky a multimédií (UPGM) Akademický rok 2019/2020

Zadání diplomové práce

Student: **Lach Martin, Bc.**
Program: Informační technologie Obor: Informační systémy
Název: **Vizualizace parametrů mnohakanálového zvukového systému v internetovém prohlížeči**
Parameter Visualization of Multichannel Audio System in Web Browser
Kategorie: Softwarové inženýrství

Zadání:

1. Seznamte se s dodaným HW a SW pro vícekanálové zpracování audia.
2. Navrhněte strukturu a protokoly pro přenos vizualizačních a řídicích dat v reálném čase.
3. Realizujte programové části embedded systému back endu v Linuxu pro posílání dat do front endu.
4. Realizujte vizualizační a řídicí rozhraní v internetovém prohlížeči. Otestujte stabilitu navrženého řešení.
5. Vyhodnoťte latence, časové a paměťové nároky. Zhodnoťte dosažené výsledky.
6. Vyrobte A2 plakátek a cca 30 vteřinové video prezentující výsledky vaší práce.

Literatura:

- Dle pokynů vedoucího

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Szőke Igor, Ing., Ph.D.**
Vedoucí ústavu: Černocký Jan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2019
Datum odevzdání: 31. července 2020
Datum schválení: 5. srpna 2020

Abstrakt

Tato práce se zabývá systémem na zpracování audia od firmy Audified. Tento vestavěný systém obsahuje ARM procesor, na kterém běží operační systém Linux. V současné době je ovládání parametrů (fantom, gain) komplikované – bez zpětné vazby. V této práci je popsaná tvorba webové aplikace, která umožní snadné nastavení zmíněných parametrů a rovnou zobrazí jejich efekt.

Abstract

This work deals with Audified Audio Processing System. This embedded system includes an ARM processor running the Linux operating system. At present, parameter control (phantom, gain) is complicated, without feedback. In this work, the creation of web application, which will allow easy setting of the mentioned parameters and will show their effect, is described.

Klíčová slova

zpracování zvuku, ARM, embeded system, Linux, klient, server, Javascript, C++, Json, Bootstrap

Keywords

audio processing, ARM, embedded system, Linux, client, server, Javascript, C++, JSON, Bootstrap

Citace

LACH, Martin. *Vizualizace parametrů mnohakanálového zvukového systému v internetovém prohlížeči.* Brno, 2019. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Igor Szőke, Ph.D.

Vizualizace parametrů mnohakanálového zvukového systému v internetovém prohlížeči

Prohlášení

Prohlašuji, že jsem tento semestrální projekt vypracoval samostatně pod vedením Ing. Igora Szőkeho, Ph.D.. Další informace mi poskytli Lubor Přikryl, David Obořil a Jan Havran z firmy Audified. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Martin Lach
5. srpna 2020

Poděkování

Děkuji svému vedoucímu diplomové práce za zadání na míru. Janu Havranovi za seznámení se software a Davidu Obořilovi za přípravu hardware.

Obsah

Kapitola 1

Úvod

Práce se zabývá systémem na zpracování audia od firmy Audified. Jedná se o „černou skříňku“, ke které se připojují mikrofony a počítač (obrázek 2.1). Zvukový systém sbírá data z mikrofonů a ta posílá počítači k uložení. Tento systém se na Fakultě informačních technologií využívá například k měření impulzní odezvy místností.

Ovládání systému probíhá konzolovou C++ aplikací, nastavení je složité a zdlouhavé. Pro nastavení slouží konfigurační soubor a pro aplikování změn je třeba aplikaci vždy restartovat. Cílem práce je vytvořit řešení, které ovládání systému a celkovou práci se systémem zjednoduší.

V kapitole 2 je popsán hardware, jeho jednotlivé součástí a architektura. Dále je v této kapitole nastíněna problematika ovládání a je tu uveden současný stav hardware a software se zaměřením na rozdíly mezi hardware, který je nyní používaný a tím, který je aktuálně vyvíjen firmou Audified.

V kapitole 3 je návrh vylepšení současného stavu. Tato kapitola obsahuje popis dat, návrh architektury a rozhraní aplikace pro ovládání a porovnání existujících řešení pro web server.

V kapitole 4 je popsána implementace serverové části aplikace, která vyřizuje všechny požadavky, nastavuje hardware a získává z něj data, slouží jako web server a odesílá stream nahraných dat.

Kapitola 5 se zaměřuje na klientskou aplikaci. Jsou tu popsány použité technologie, uživatelské rozhraní. Dále jsou zde uvedeny funkce klientské aplikace a jejich použití. Následuje test stability a rychlosti.

Kapitola 6 popisuje výslednou aplikaci. Popsána je komunikace mezi jednotlivými částmi, princip nahrávání, přehrávání a výsledný formát dat.

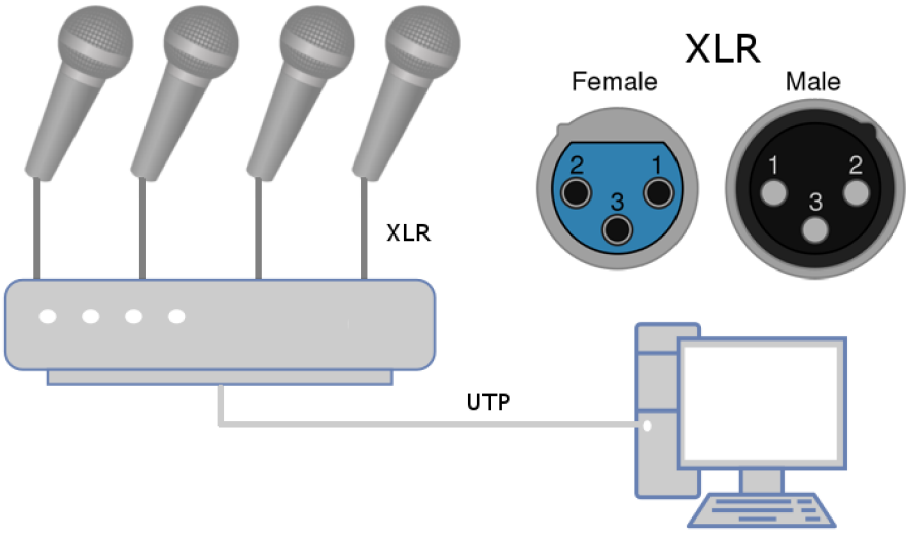
V kapitole 7 jsou představeny nástroje využité při testování, jsou zde popsány jednotlivé fáze testování, průběh testování a dosažené výsledky.

Kapitola 2

Popis dodaného HW a SW

Systém se používá k pořizování záznamu z mikrofonů, nahrávky slouží například k měření impulzní odezvy místností. Měření probíhá tak, že se do dané místnosti umístí zdroj zvuku – reproduktor na kterém se přehrávají připravené nahrávky. Zároveň se po místnosti rozmístí mikrofony připojené do zvukového systému.

K systému se ethernetovým kabelem připojuje počítač, který slouží k ovládání systému a ukládání získaných dat. Schéma zapojení je na obrázku 2.1. Ke komunikaci se systémem slouží konzolová aplikace, která při spuštění dostane jako argumenty cestu pro umístění nahraných souborů a konfigurační soubor, který obsahuje nastavení zařízení. Po spuštění aplikace proběhne inicializace, nastavení a čeká se na uživatelský vstup, kterým je spuštěno nahrávání. Po spuštění uživatel opustí místnost a nahrávání probíhá několik hodin.



Obrázek 2.1: Schéma zapojení systému k počítači a mikrofonům, schematické znázornění XLR konektoru.

2.1 Pohled na HW

Zařízení se skládá z napájecího zdroje, základní desky a modulů zvukových karet. Základní deska obsahuje procesor se dvěma A/D převodníky, operační paměť a porty, jako například RJ-45, USB, nebo RCA. K základní desce se pomocí MikroBUS [1] sběrnic připojují další moduly. Přidáváním a odebíráním modulů se mění skladba vstupních a výstupních portů.

Obrázek 2.2: Fotografie zvukového systému. V levé spodní části se nachází základní deska. Tato konfigurace systému je osazena čtyřmi zvukovými kartami, každá se čtyřmi vstupy XLR. Šedými kabely jsou do některých portů zapojeny mikrofony. Bílé krabičky v popředí jsou mikrofony.

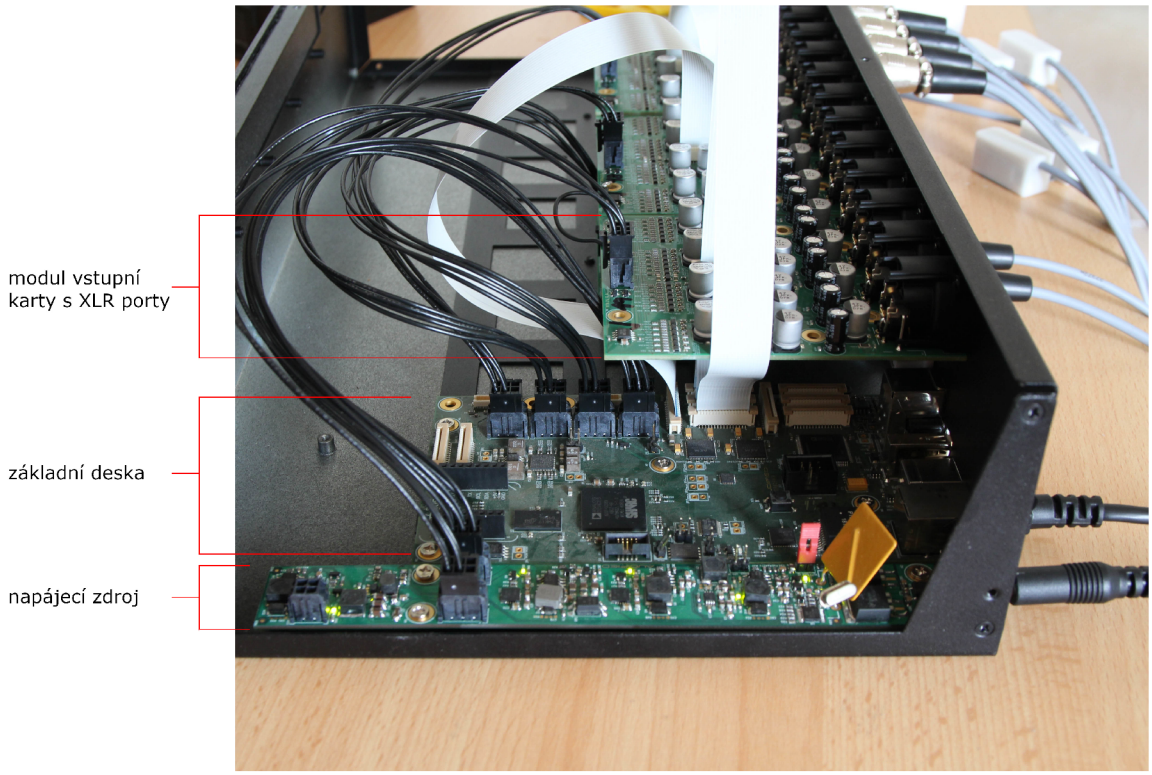
Moduly vstupů/výstupů

Kromě základní desky jsou pro funkci zařízení nezbytné i vstupy pro připojení mikrofonů, případně výstupy k připojení reproduktorů. Tyto porty se nacházejí na menších kartách a protože jde o modulární zařízení, může být jejich rozložení různé. Aktuálně jsou použité vstupní karty s XLR konektory, na každé kartě jsou čtyři vstupy, viz obrázek 2.2. Dalšími typy modulů mohou být například výstupní karty, ke kterým se připojují reproduktory, nebo karty s konektory JACK 6.3.

Boot

Boot probíhá po zapnutí napájení. Používaná je metoda, kdy je systém načten z SD karty do paměti. Karta může být naformátovaná jako *ext*, nebo *FAT* souborový systém. Je ale třeba, aby byl formát kompatibilní s bootloaderem. Průběh bootu je možné sledovat a ovládat v terminálu přes UART (Universal asynchronous receiver-transmitter) rozhraní.

[1] https://www.mikroe.com/mikrobus



Obrázek 2.3: Fotografie otevřeného zařízení. Ve spodní části je základní deska, se kterou jsou ostatní karty propojené černými kabely napájení a bílými datovými kabely.

2.2 Použitý procesor

Na základní desce, se kterou jsem pracoval, je použitý čip od firmy Analog Devices. Konkrétně model ADSP-sc589 [2].

Tento procesor je určen přímo pro zpracování audia, ale využívá se i v jiných oblastech. Hlavními součástmi (obrázek 2.4) jsou dvě SHARC jádra a ARM procesor Cortex A5. K tomuto čipu je připojená DDR paměť. Dále jsou k němu přivedeny slot micro SD karty (z ní se bootuje Linux) a flash paměť, ve které je nyní uložen bootloader a program pro SHARC jádra.

DSP SHARC

SHARC je digitální signálový procesor (DSP), který slouží pro operace s plovoucí řádovou čárkou. Je připojen k A/D převodníku a proudí přes něj veškerá audio data. Kód pro DSP dodala firma Audified. Pro komunikaci s ARM procesorem se nyní využívá MCAPI a v nové verzi se se k datům z DSP přistupuje přes ALSA ovladač.

Cortex A5 [3]

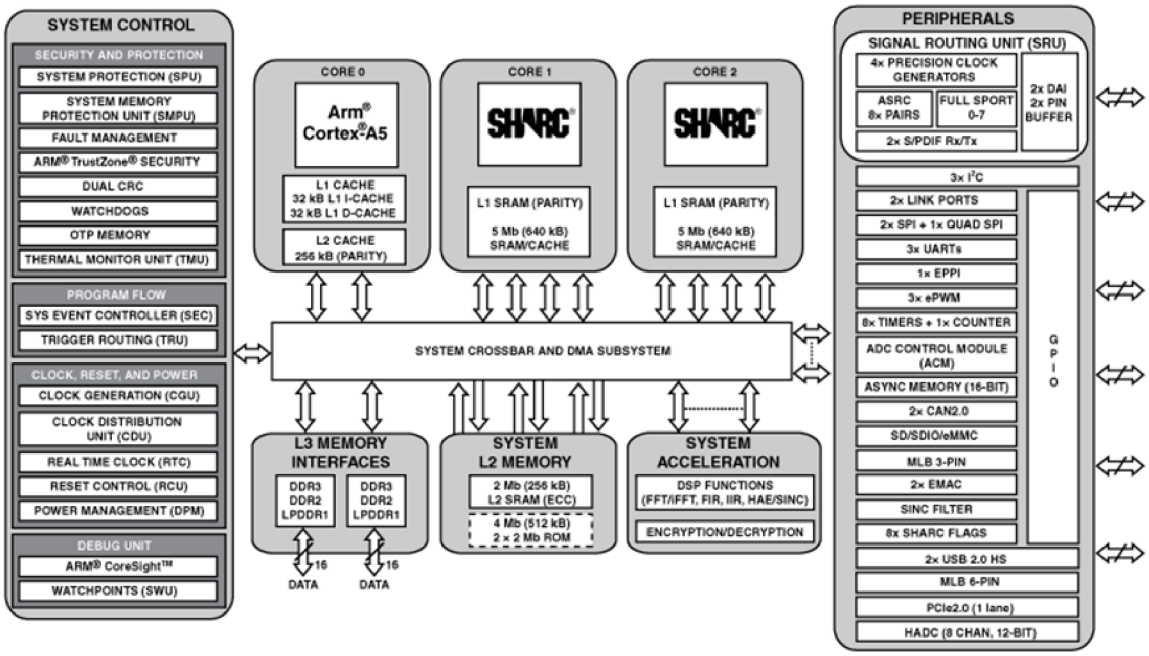
Cortex A5 je 32 bitový ARM procesor z řady ARMv7-A, rozšířený o operace v plovoucí řádové čárce a Neon, aby zvládal zpracovávat data a zároveň přes rozhraní mohl komunikovat periferními zařízeními. Využívá L1 instrukční a datovou cache (každá 32 kB) a 256 kB L2 cache.

Z rozhraní jsou pro projekt nejdůležitější ethernet a slot pro SD kartu. Jsou zde i další, ale u těch chybí programová podpora.

Na tomto jádře běží vestavěný Linux, vytvořený a zkompilovaný přímo pro dané použití. Veškeré úpravy budou implementovány právě pro tento operační systém.

2.2.1 A/D převodníky

AK5558[2] - 32 bitový, 8 kanálový A/D převodník. Umožňuje vzorkovat až frekvencí 768 kHz. Využívá se ale snížená frekvence 48 kHz, která se standardně používá pro nahrávání slyšitelných zvuků. Je vhodná pro následné zpracování i kompatibilitu s dalšími systémy. Další výhodou je nižší datový tok, který šetří přenosové pásmo a snižuje velikost nahraných dat.



Obrázek 2.4: Funkční schéma použitého procesoru Analog Devices adsp-sc589. Převzato z [2].

2.3 Současné SW řešení

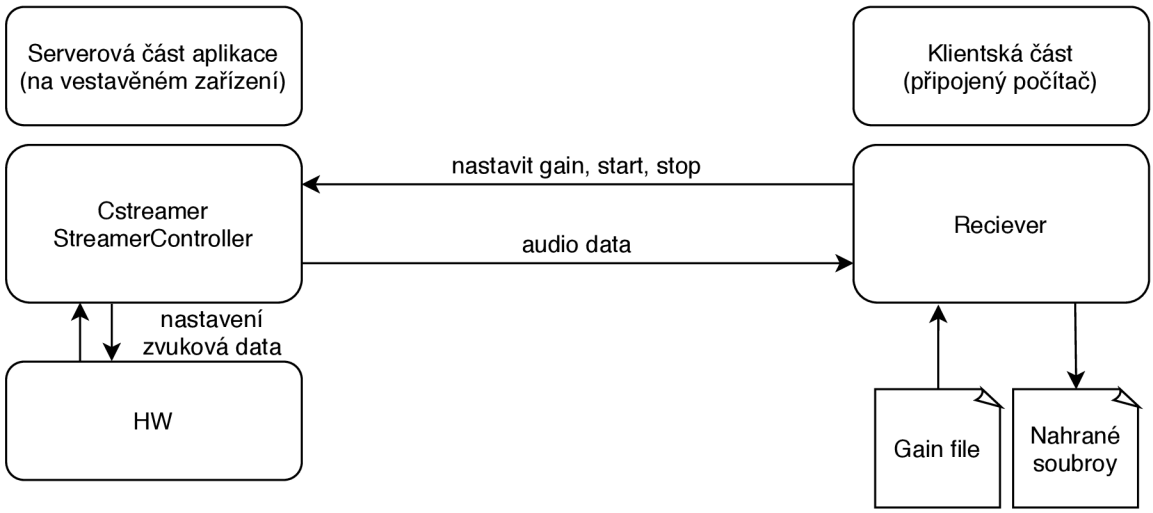
K software není existující dokumentace, veškeré informace jsem byl nucen čerpat ze zdrojových kódů a jejich komentářů. V řešení, které má fakulta k dispozici se k předávání dat používá konzolová aplikace. K nastavení parametrů slouží konfigurační soubor. Průběh práce se zařízením není pohodlný ani intuitivní. Je třeba udělat následující:

- spustit aplikaci, nahrát data, zastavit aplikaci
- otevřít nahraná data a zobrazit úrovně vybuzení

[2]https://www.akm.com/content/dam/documents/products/audio/audio-adc/ak5558vn/ak5558vn-en-datasheet.pdf

- přepsat konfigurační soubor – odhadnout vhodné hodnoty gain
- opakovat

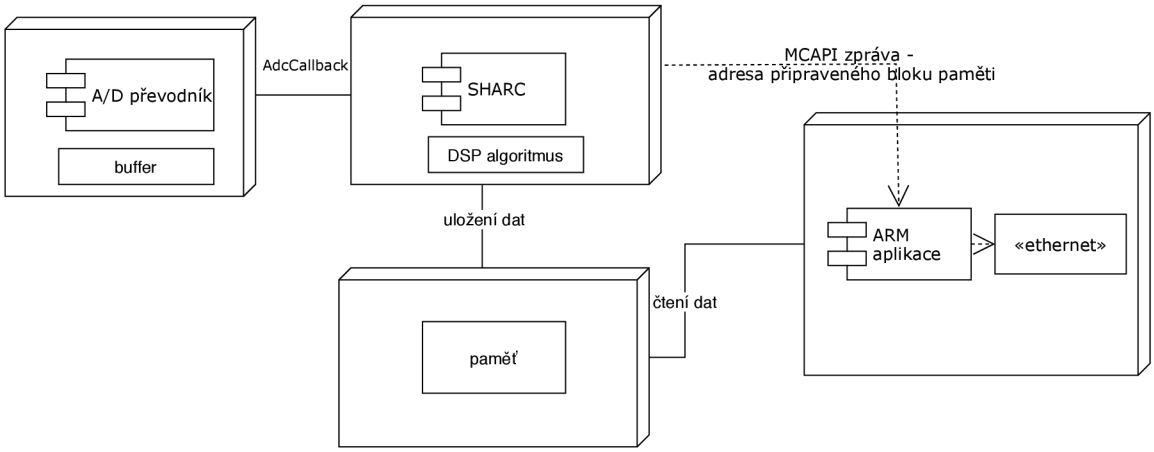
Komunikaci tak, jak v tuto chvíli funguje jsem znázornil na obrázku 2.5.



Obrázek 2.5: Schéma komunikace jednotlivých softwarových částí současného systému.

2.3.1 Audio streaming

Aplikace běžící na zařízení čeká na připojení programu Reciever, komunikuje s programem Recieverer, čte audio data z SHARC jádra a odesílá audio data. Funkční schéma SW je na obrázku 2.6.



Obrázek 2.6: Schéma komunikace jednotlivých částí současného systému.

Streamer

Streamer je třída sloužící pro čtení dat z hardware a jejich odesílání Recieveru. Je připravena na různé implementace pro získání dat, k tomu je v současnosti použita třída `Cstreamer`,

která využívá pro komunikaci se SHARC jádrem MCAPI (Multicore Communications API). Data jsou čtena ze sdílené paměti, implementované jako kruhový buffer. Buffer je pomocí MCAPI synchronizován s činností SHARC jádra.

Současná verze software byla vytvořena pro předchozí generaci hardware. Ten používal 24 bitové A/D převodníky Analog Devices ADAU1979[3]. Přímo z aplikace je nejdříve inicializováno SHARC jádro, vytvoří se MCAPI slave spojení a čeká se na master.

SHARC jádro inicializuje A/D převodník a spustí ho. Následně se vytvoří MCAPI master. Další činnost probíhá v cyklu. Získaná data a se ukládají do sdílené paměti, po uložení dat je odeslána MCAPI zpráva, s adresou připraveného bloku dat.

Aplikace čeká na MCAPI zprávu s adresou dat. Po obdržení zprávy jsou příslušná data zpracována a buffer je opět uvolněn pro další zápis.

Používá se 3 bytový formát vzorků, pro synchronizaci se používá prokládání vzorků viz obrázek 4.3.

StreamerController

StreamerController slouží k inicializaci Streameru a síťového připojení, vyřizuje řídící zprávy. Implementuje rozhraní pro přístup ke Streameru.

2.3.2 Síťová komunikace

Další samostatnou třídou je protokol pro síťovou komunikaci. Tuto třídu využívá StreamerController i Reciever.

Formát paketů

Každý paket je uveden označením typu (výčtem). Nejpoužívanější typy jsou datový paket (kNetMessageData), konfigurační (kNetMessageCfg) a informační (kNetMessageInform) paket. Dále je a připraven paket s údaji o RMS (kNetMessageInformRMS), ale aktuálně není využitý. Využití paměti se liší podle typu paketu, viz. tabulka 2.7.

Datový paket

Používá se ke streamování dat. Kromě vyjádření typu, který obsahují všechny pakety, obsahuje časovou známku (timestamp), která určuje pořadí paketu ve streamu. Po časové známce následuje počet vzorků, který vyjadřuje velikost posílaných dat. Zbytek zabírají samotná data, viz. tabulka 2.7.

Konfigurační paket

Tímto paketem se například zahajuje nahrávání. Historicky sloužilo k nastavování parametrů.

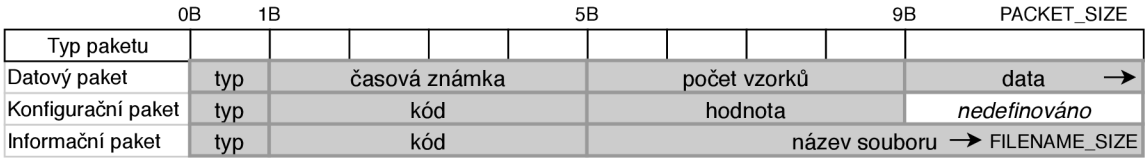
Obsahuje dvě číselné informace. Kód zprávy a hodnotu. Kód je opět vyjádřený výčtem. Hodnota dostává význam podle typu zprávy, viz. tabulka 2.7.

[3] https://www.analog.com/media/en/technical-documentation/data-sheets/ADAU1979.pdf

Informační paket

Informační paket obsahuje řídící, ale i další informace, viz. tabulka 2.7. Kód slouží k odlišení jednotlivých druhů informací (například „spuštění přehrávání, zastavení přehrávání, název přehrávaného souboru“), nebo k bližší identifikaci zprávy.

V části uvnázev souboru se primárně přenáší jména souboru a to až do velikosti `FILENAME_SIZE`. Omezení délky souboru slouží k tomu, aby nebylo třeba posílat informace o délce názvu.



Obrázek 2.7: Typy paketů s využitím paměti.

Odesílání dat (streaming)

Odesílání dat musí být rychlejší, než jejich nahrávání. Výsledná velikost nahraných dat není omezena a proto není přípustné žádné hromadění dat v bufferech. Za minutu vznikne na jednom kanálu cca 10 MB dat. Délka nahrávání se počítá na hodiny, případně dny.

Je třeba zachovat časově synchronizovaná data. Odesílání musí být spolehlivé. Proto je každý paket označen časovou známkou a kontroluje se jeho doručení.

Čím menší velikost paketů, tím větší nároky na režii (přenášení hlaviček a dalších metadat), větší pakety zase zdržují a je problém při jejich znovu zasílání. Je proto nutné najít vhodný kompromis.

2.3.3 Reciever

Program Reciever se spouští na počítači s IP adresou a konfiguračním souborem, případně také složkou pro ukládání dat a názvem souborů jako argumenty. Po připojení k systému odešle přednastavené parametry. Činností Recieveru je ovládat stream (spustit, zastavit), přijímat data a ukládat je do souborů.

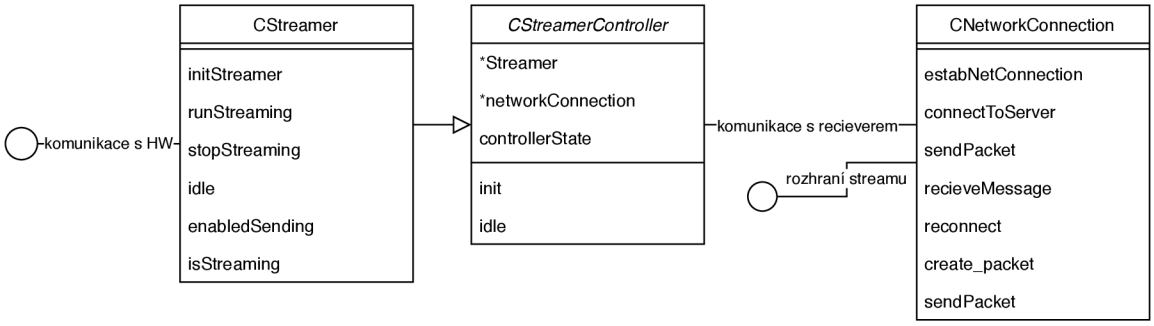
```
./Reciever -IP 192.168.1.9 -gain-file gains [-out-dir slozka_s_nahravkami
-out-file predpona_nazvu_souboru]
```

Zde jsou opět využité kruhové buffery. Kvůli režii je výhodnější ukládat do souboru až větší množství dat a zároveň se zde rozděluje prokládaný formát dat na jednotlivé kanály – pro každý kanál je jeden buffer. Současně se také mění formát vzorků ze 3 bytových na výsledné 4 bytové.

2.4 Vestavěný operační systém

Po připojení napájení se operační systém zkopíruje z SD karty do operační paměti. Systém nemá přístup k perzistentní paměti. Každé vypnutí je tvrdý restart a po opětovném zapnutí je systém obnoven z SD karty. Veškerá data jsou ztracena.

To přináší následující omezení:



Obrázek 2.8: Znázornění tříd aplikace běžící na zvukovém systému.

- velikost paměti – celkem 105 MB, systém zabírá 22 MB
- embeded systém má méně funkcí a knihoven než plnohodnotná distribuce
- pro přidání programu je třeba systém zkompilovat a přenést na sd kartu
- systém běží v paměti RAM, z čehož vyplývá, že nelze provádět perzistentní změny, data je třeba ukládat na externí úložiště

Velikost paměti v tomto případě, kdy se neukládají žádná data, není příliš omezující. Pro testování se kopírují potřebné soubory pomocí vzdáleného přístupu. K systému je možné se vzdáleně připojit pomocí LAN portu, přes SSH protokol.

Vestavěný systém byl vytvořen pomocí nástroje Buildroot[4], který dokáže ze vstupní konfigurace generovat Linuxové jádro, kořenový systém souborů a toolchain křížové kompilace.

2.5 Nastavované parametry

Pro každý kanál (vstup) je možné nastavit dva parametry – phantom (phantomové napájení) a gain (zisk).

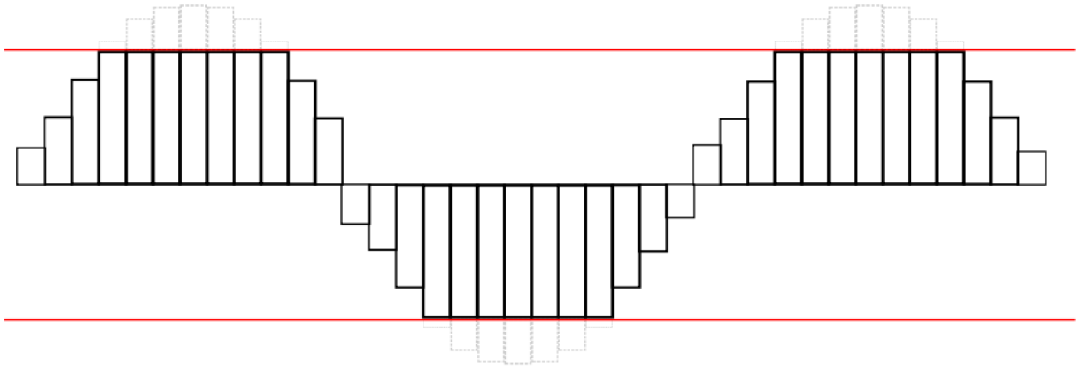
Phantom

Phantomové napájení má polohy zapnuto a vypnuto. Jde o přídavné napájení XLR konektorem. Velikost napětí záleží na výrobci a daném produktu, obvykle bývá +48V.

Využívá se především pro kondenzátorové mikrofony, ale i jiná zařízení, která potřebují napájení. Výhodou tohoto řešení je, že se pro data i napájení používá jeden kabel. Nevýhodou je potom riziko poničení přístrojů, které napájení nepotřebují. Toto riziko je však při správném zapojení velmi malé a pravděpodobnost poničení přístroje tudíž minimální.

Potenciální problém plyne z nesymetrického napájení, způsobeného například odpojováním a připojováním konektoru, poničeným kabelem, nebo opotřebovanými konektory. Nemá cenu spekulovat o tom, jestli by nestačilo mít přivedené napájení po celou dobu. Dobrou praxí je zkrátka zapínání phantomového napájení pouze na těch vstupech, kde je potřeba.

[4]https://buildroot.org/



Obrázek 2.9: Příklad „clippingu“ a PCM.

Gain

Gain je upravení zisku předzesilovače tak, aby úroveň vstupního signálu byla optimální pro AD převodník. Při nedostatečném vybuzení je signál nerozlišitelný od šumu. Při přebuzení dochází k takzvanému „clippingu“ (znázorněno na obrázku 2.9) – zkreslení, kdy největší amplitudy jsou ořezané, protože pro každou hodnotu větší než maximální je zpracována právě maximální hodnota. V systému se rozlišuje sedm úrovní gainu.

Kapitola 3

Přenos vizualizačních a řídicích dat – návrh

Zvukový systém není možné ovládat přímo, nemá na sobě žádná tlačítka, ani display. Cílem je vytvořit aplikaci, která usnadní ovládání systému a dá uživateli zpětnou vazbu o tom, co se zrovna děje. Usnadnění spočívá ve zlepšení přehlednosti – grafické zobrazení úrovní s nepostřehnutelným zpožděním, možností okamžité úpravy parametrů, přehrávání audio souborů a správa playlistů.

Aplikace by měla být integrovaná v zařízení. Načte se jako webová stránka po zadání IP adresy do prohlížeče. Pro plánované využití je třeba zobrazit v jakém stavu se systém nachází, zobrazit úrovně vybuzení jednotlivých kanálů, umožnit změnu gainu, phantomu a ovládat nahrávání zvuku. To vyžaduje vytvoření http serveru a webové aplikace.

Serverová část bude mít následující funkce:

- naslouchání na přidělené IP adrese a portu
- přenos klientské aplikace na připojený počítač
- vyřizování požadavků klientské aplikace
- řízení hardware a získání vizualizačních dat

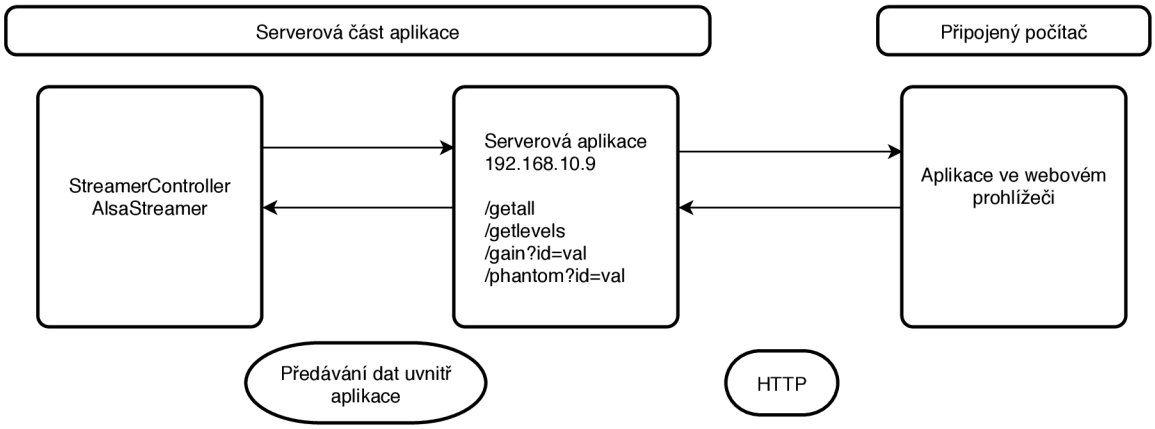
Klientská webová aplikace bude:

- použitelná s většinou webových prohlížečů
- mít přehledný vzhled a intuitivní ovládání
- posílat požadavky serveru a přijímat odpovědi

Vzniká potřeba několika komunikačních rozhraní. Webové rozhraní je mezi serverem a klientem na připojeném počítači. Přes toto rozhraní se získávají informace o stavu systému a nastavují jeho parametry. Další rozhraní potom slouží pro samotné získání zvukových dat.

Data jsou „živá“. Neustále přibývají nové vzorky, které je třeba zpracovat. Rychlost přibývání dat závisí na vzorkovací frekvenci, počtu kanálů a velikosti vzorku. Počet kanálů je proměnný, je použita vzorkovací frekvence 48 kHz a velikost vzorku je 4 byty. K získání stavu hardware a jeho ovládání bylo využito rozhraní třídy StreamerController. To bude rozšířeno tak, aby bylo možné získat všechna potřebná data.

Standard pro posílání dat webové aplikaci je formát JSON. Jedná se o textový formát, který je snadno použitelný s C++ i Javascriptem.



Obrázek 3.1: Schéma rozdělení aplikace s popisem rozhraní http.

3.1 Řídící data

Pro inicializaci a úvodní nastavení je třeba získat počet kanálů s jejich aktuálním nastavením. Dalšími potřebnými informacemi jsou data o stavu přístroje (zda je aktivní nahrávání, je inicializované síťové připojení a jestli je připojený reciever), po rozšíření také playlisty, dostupné zvukové soubory a právě přehrávaný soubor. V neustálých aktualizacích je třeba získávat úrovně vybuzení jednotlivých kanálů i stav přístroje. Zpracovávat a odesílat je třeba změny v parametrech, ovládací příkazy a ukládání playlistů.

Počet kanálů je definován konstantou. Pro nastavení phantomu je využitý jeden bit, vyjadřuje hodnoty zapnuto, vypnuto. Pro zakódování gainu se využívá tří bitů, použito je sedm různých úrovní. Hodnoty parametrů jsou uloženy v Linuxu v souborech. Z těchto souborů se čte aktuální nastavení, zápisem do nich se nastavení změní.

Úroveň vybuzení je vyjádřena procenty, přesnost je omezená na hodnotu. Hodnota úrovně se musí připravit ze vzorků, které byly nasbírány od minulé aktualizace, aby při požadavku mohla být odeslána aktuální odpověď a informace byly konkrétní. Stav přístroje se skládá ze stavu síťového připojení a stavu Streameru.

3.1.1 Úroveň vybuzení

Zvukové vlny se v A/D převodnících kvantifikují na vzorky. Pro optimální funkci A/D převodníku musí být přivedený signál ve správném rozsahu. Síla signálu se nastavuje ziskem – parametrem gain. Pro informaci uživatele slouží zobrazení úrovně vybuzení signálu. Ke získání hodnoty je možné přistupovat několika způsoby. Při zpracování zvuku se využívají převážně dva přístupy[1] a to efektivní hodnota (RMS) a maximální hodnota (peak).

RMS

RMS ukazuje hodnotu výkonu signálu, jakou by měl odpovídající stejnosměrný signál. Počítá se z několika vzorků, smysl dává výpočet ze vzorků sesbíraných přibližně za více než půl sekundy. Efektivní hodnota ukazuje sílu signálu, tato vlastnost je výhodná, protože velikost signálu přibližně odpovídá vnímání lidského ucha, nebo potřebnému výkonu zesilovačů ale nezobrazí například, zda došlo k ořezání (obrázek 2.9).

[1] https://support.biamp.com/General/Audio/Peak_vs_RMS_Meters

Peak

Peak slouží pro zobrazení amplitudy. Výhodou měření maxima (peaku) je to, že na zobrazení těchto hodnot je vidět ořezání. Názory na to, které měření použít se různí, v této aplikaci bude právě díky zmíněné vlastnosti aplikován typ měření peak.

3.1.2 Zvukové soubory a jejich seznamy - návrh

Zvukové soubory mohou být ukládány přímo na SD kartu, konkrétně na oddíl, kde není systém. Soubory se dají na kartu kopírovat přímo fyzicky. Je ale nutné se ke kartě dostat. K tomu je potřeba rozebrat zařízení, to není praktické řešení.

Naštěstí je tu i druhá možnost a to využít SSH, případně SFTP. Linuxové systémy podporují připojení serveru (v tomto případě zvukového zařízení) přímo do souborového systému, potom se s ním dá pracovat podobně jako s jakýmkoliv jiným adresářem, lze také využít protokol SCP. U Windows je třeba použít program podporující přípojení pomocí zmíněných protokolů.

Adresářová struktura

Na příslušném oddíle SD karty jsou dva adresáře. `Audio` slouží právě pro uložení souborů k přehrání. `Playlist` potom pro uložení playlistů.

Formát souboru

Podporovány budou pouze wav soubory se vzorkovací frekvencí 48 kHZ. Počet kanálů může být různý - využije se jich tolik, kolik je maximálně možné. Formát vzorků může být téměř libovolný. Nejvhodnější je `float`, ostatní formáty se musejí převádět, převádění je ale automatické a nemělo by mít vliv na funkci.

Formát playlistu

Playlist je textový soubor formátu csv, který obsahuje názvy zvukových souborů oddělené čárkami. Zpracování souboru bude "rozsekání"podle čárek, což přináší omezení, kdy název souboru nemůže obsahovat čárku.

3.2 Rozhraní pro komunikaci s front end aplikací

Tvorba tohoto rozhraní byla přímočařejší, ikdyž z funkčního hlediska implementuje téměř totéž. Server tvoří http rozhraní, které volá aplikace podle potřeby, nebo interakce s uživatelem. Data směrem od uživatele k serveru jsou předána v GET http požadavku. Odpovědi jsou ve formátu JSON, který je snadno zpracovatelný JavaScriptem. Rozhraní se skládá z následujících zdrojů, které mají své URI:

/getall

Bez parametrů. Vrací nastavení gain a phantom pro všechny kanály. JSON obsahuje dvě pole – gains a phantoms. Číslo kanálu se shoduje s indexem pole.

/getlevels

Bez parametrů. Vrací úroveň signálu pro všechny kanály. JSON obsahuje pole `levels` s hodnotami úrovní. Číslo kanálu se shoduje s indexem pole. Za polem jsou parametry `streaming`, `state`, `playing` a `connected`, určující stav systému.

/gain?id=value

Parametr `id` je číslo kanálu, `value` je nově nastavená hodnota gain. Odpověď je shodná s *getall*.

/phant?id=value

Parametr `id` je číslo kanálu, `value` je nově nastavená hodnota phantom pro daný kanál. Odpověď je shodná s *getall*.

/getaudiofiles

Vrací názvy souborů wav v příslušném adresáři.

/getplaylistfiles

Vrací uložené playlisty.

/getplaylist?name

Vrátí detail vybraného playlistu.

/save?name=values

Přiřadí audio soubory `values` do playlistu `name`.

/record, /playrecord, /stop

Bez parametrů. Spustí, nebo zastaví nahrávání dat, případně současně přehrávání vybraných souborů. Nečeká se na odpověď, změna stavu se projeví v aktualizaci `getlevels`.

3.3 Porovnání existujících řešení pro embeded web server

Samotné tvorbě serveru předcházel průzkum možností. Současné řešení je implementované jazykem C++. Pro zachování kompatibility a jednotného přístupu bude řešení také implementováno v C++.

C++ REST SDK[2]

Projekt k tvorbě server-klient aplikací s nesynchronními operacemi od Microsoftu. Obsahuje nástroje pro připojení a komunikaci s REST rozhraními.
Klady:

- je to komplexní řešení

[2]https://github.com/Microsoft/cpprestsdk

- Vývoj Microsoftem s podporou Linuxu

Zápory:

- Stručná dokumentace (přechod z verze pojmenované Casablanca) s minimem příkladů
- Není přímá podpora funkce pro předávání souborů
- Příliš robustní
- Je určen na připojování se ke službám, ne jejich vytváření

Minihttp server [3]

Minimalistický http server v C++, popsaný v článku na root.cz [4].
Klady:

- Používá čisté C++, bez dalších knihoven
- Je opravdu miniaturní (pouze jeden soubor)
- Je určený právě pro daný účel, podporuje zpracování jednotlivých žádostí pomocí skriptu.

Zápory:

- Používá C++17, křížový překladač ho nepodporuje
- Není přehledný – pouze jeden soubor

Knihovna Boost [1], http server [4]

Boost je velmi rozsáhlá knihovna, která usnadňuje práci v mnoha oblastech. Řadí se mezi standardní knihovny a její součásti postupně přecházejí do norem C++.
Klady:

- Knihovny jsou součástí operačního systému, používají se v Audified
- Velmi dobrá dokumentace a podpora na fórech
- Hotový http server mezi příklady
- Přehledně psaný, snadno rozšířitelný kód

Zápory:

- Jednotlivé verze knihovny nejsou plně kompatibilní – optimální by bylo pracovat s verzí, kterou využívají ostatní aplikace a je integrovaná v operačním systému.

Zhodnocení

Jako nejvýhodnější se ukázalo zvolit kompromis mezi psaním v čistém C (C++) a obsáhlým projektem. Tím je knihovna Boost. Výsledná aplikace využívá jeden z poskytnutých příkladů pro http server.

[3] https://github.com/ondra-novak/minihttp
[4] https://blog.root.cz/novacisko/minimalisticky-http-server-v-cpp/

Kapitola 4

Programová část pro posílání dat klientské aplikaci

Tato kapitola popisuje serverovou část. Jedná se o aplikaci psanou jazykem C++, určenou k běhu na vestavěném systému. Její tvorba přinesla několik výzev. Zejména synchronizaci mnoha vláken, kompilování pro cílový systém, bezchybné vyčítání a posílání dat z ALSA zařízení, funkční a smysluplné propojení celého systému při zachování kompatibility. To vše za používání pouze knihoven, které jsou dostupné v systému – toto se nakonec ukázalo jako slabé omezení. Vlastnosti C++11 je možné využívat plnohodnotně.

4.1 Křížová kompilace (cross compilation)

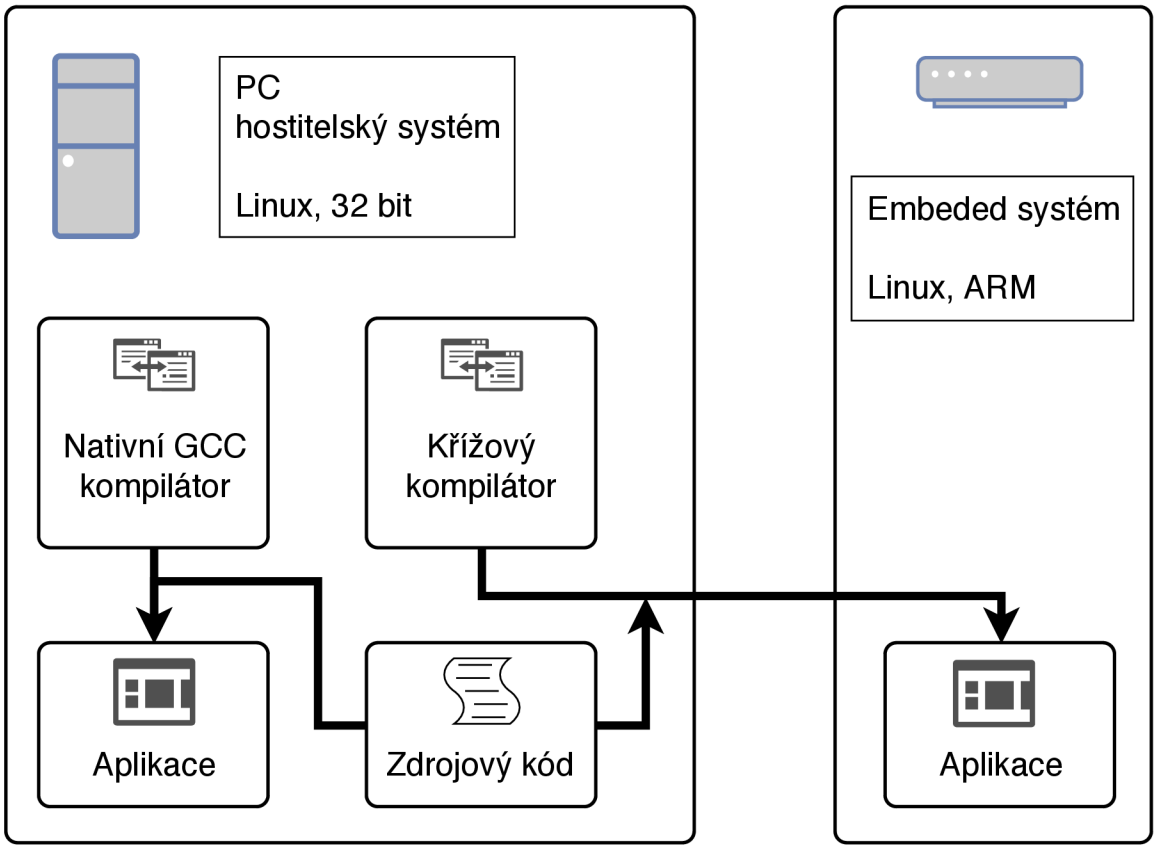
Pro správnou funkci aplikace je ideální ji zkompilovat právě na systému, kde má běžet. Vestavěné systémy mají velmi omezené prostředky. Ať už jde o výkon, přítomnost kompilátoru, nebo dokonce operační systém. Proto se přistupuje ke křížové kompilaci, kde *hostitelský systém* dokáže zkompilovat kód pro cílovou platformu, což je znázorněno na obrázku 4.1.

Křížová kompilace je velmi citlivá na nastavení. Je možné stáhnout různé balíčky rozšiřující gcc o překlad pro jiné platformy. Při testování tohoto přístupu se úspěšně dařilo překládat programy pouze se statickým linkováním knihoven. To samozřejmě není ideální přístup. Jako vhodné řešení se ukázalo použít křížový překladač, který poskytuje přímo Buildroot - nástroj pro generování vestavěného linuxu. Byl nakonfigurovaný Audified pro daný operační systém. Balíček obsahuje gcc (g++), překladač assembleru i hlavičkové soubory knihoven.

4.2 Ukládání dat

Linux se při spuštění bootuje z SD karty, nebo flash paměti. Následně běží v nevolatilní paměti. Omezení, které to způsobuje je, že veškerá uložená data jsou po restartu ztracena. To je zároveň velká výhoda, protože tímto způsobem je systém ochráněn před hrozbou poškození uživatelským zásahem.

Pro zlepšení uživatelské zkušenosti by pomohlo, kdyby systém uměl s nahráváním zároveň i přehrávat soubory. Nedává smysl mít je v připojeném počítači. Nahrávání může probíhat pokaždé s jiným počítačem a přehrávaná data by bylo třeba streamovat na zvukový systém, nebo pokaždé kopírovat. Tím by se ztratila výhoda integrace této funkce,



Obrázek 4.1: Schéma klasické a křížové kompilace.

protože by bylo možné použít jakoukoliv jinou zvukovou kartu a vše vyřešit jednodušeji, například přehráváním z počítače. Bylo třeba vyřešit, jak ukládat audio soubory.

Navržené řešení využívá SD kartu s operačním systémem. Ta byla naformátována na dva oddíly. Jeden na operační systém, druhý právě na trvalé uložení dat. Tato paměť bude připojena příkazem *mount* do souborového systému již při bootování. Rozdíl oproti zbytku souborového systému bude pouze v tom, že data zapsaná na tomto svazku zůstanou dostupná i po restartu. Paměť je omezena velikostí SD karty. K audio souborům přibyly i playlisty, aby bylo možné nahrávat automaticky několik audio souborů po sobě. Playlisty je také třeba ukládat a je nutné pro ně vytvořit další programovou podporu.

4.3 Web server

Zvukový systém obsluhuje jeden počítač připojený přes ethernet. Tím odpadá potřeba paralelního vyřizování požadavků. Při plánovaném využití se předpokládá, že uživatel má fyzický přístup k zařízení, proto nemá význam server zabezpečovat. Aplikace potřebuje přístup k http portu (80), spouští se tedy s právy root.

Server má tři funkční bloky:

1. File server – pokud je specifikován soubor, odešle jeho obsah, nebo zápornou odpověď
2. Práce s daty – pokud není v URI dotaz na soubor, ale na zdroj, vykoná danou činnost a odpoví JSON daty

3. Předávání dat z, nebo do zbytku systému

Aplikace vychází z příkladu [4]. Ve kterém byly upraveny třídy `request_handler` a `mime_types`, které řeší všechny http požadavky a byla přidána nová třída `api_handler`, pro vyřizování požadavků aplikačního rozhraní. Funkce `main` byla odstraněna a server zakomponován do výsledné aplikace.

Request handler

Do této třídy přichází adresa požadavku, ta je zpracována. Požadavky se dělí na žádosti o soubory a data.

Nejdříve se zkontroluje, zda je cesta validní – aby nebyl ke stažení přes toto rozhraní poskytován celý souborový systém. Adresa nesmí obsahovat například „..“ pro přechod do nadřazeného adresáře.

Pokud adresa obsahuje „*?*“, přeposílá se požadavek na `api_handler`. Pokud se v adrese otazník nenachází, zpracuje požadavek přímo `request_handler`. Podle přípony se určí typ souboru a příslušný typ internetového média (MIME - Multipurpose Internet Mail Extensions). Pokud není daná přípona nalezena, použije se *text/plain*. Výjimkou je zakončení adresy lomítkem. V tom případě se požadavek před zpracováním doplní o `index.html`.

Pokud je soubor nalezen, jsou naplněny hlavičky, stav zprávy a společně s daty je vše odesláno. Pokud soubor nebyl nalezen, nebo se vyskytla jiná chyba, Je použita knihovní zpráva `boost::asio::buffer(bad_request)`.

Api handler

Pro posílání dat do frontendu je využito bezstavové aplikační rozhraní. Server vyřizuje jednotlivé požadavky klienta a neukládá stav komunikace ani jiné informace. Server šetří prostředky tím, že pasivně čeká a odesílá data jen pokud přijde požadavek. Klientská aplikace může být škálovatelná. Například když se uživatel rozhodne, že chce dostávat aktualizace o stavu zařízení častěji. Stačí změnit příslušný kód JavaScriptu. Protokol je univerzální a může ho využít i jakákoliv jiná aplikace. A to bez jakýchkoliv úprav na serveru.

Jsou tu i jisté nevýhody. Tou největší je, že pro každou část dat se vytváří nové spojení. Na server tak přichází relativně velké množství požadavků i ve chvíli, kdy pouze zobrazuje úrovně.

Mime types

Víceúčelová rozšíření internetové pošty (Multipurpose Internet Mail Extensions) je internetový protokol vytvořený pro email, ale používaný i dalšími protokoly. Mezi nimi i HTTP. Sloužil k rozšíření elektronické pošty o specifikaci kódování textu (diakritika), přenos binárních dat, příloh a podobně.

Hlavička obsahuje verzi protokolu, typ obsahu, kódování a několik dalších parametrů. Nás nejvíce zajímá použití s HTTP, konkrétně typ obsahu. Bez správně nastaveného typu obsahu webový prohlížeč interpretuje vše jako text. Například CSS soubory bez správného příznaku odmítá použít.

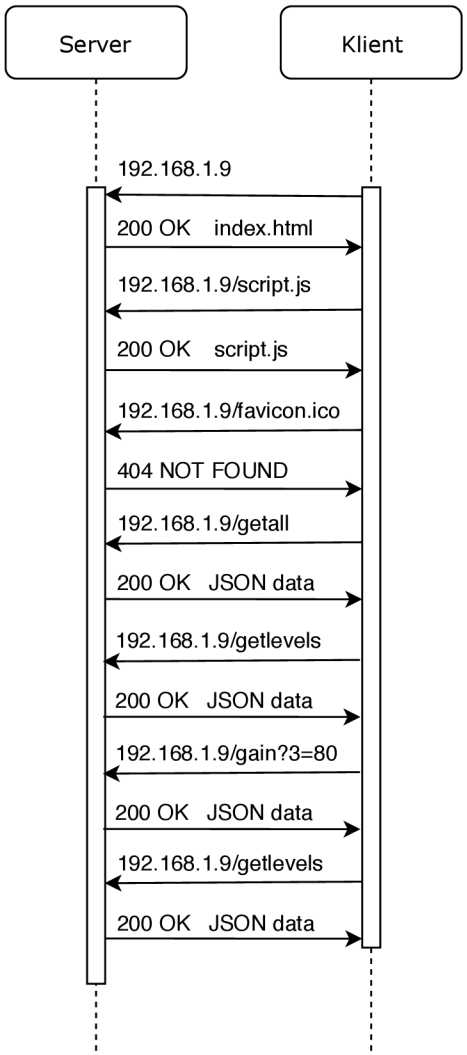
Server rozlišuje typy podle přípony. V základní implementace zvládala pouze text a JPEG obrázky. Bylo třeba doplnit typy pro posílání kaskádových stylů, javascriptu, dalších typů obrázků.

4.3.1 Playlisty a soubory - implementace

Seznam playlistů a souborů se odesílá při prvním požadavku klienta. Názvy souborů se získávají přímo ze složky. Jsou omezeny délkou `FILENAME_SIZE`.

Při ukládání pošle klient název playlistu a názvy souborů, které obsahuje. Vše se uloží do souboru. Pokud soubor s příslušným názvem existuje, tak se přepíše, pokud neexistuje, je nově vytvořen.

K přehrání více souborů nemusí uživatel využít pojmenovaný playlist. Interně je ale chování stejné. Aktuálně vybrané soubory se také ukládají do playlistu. Slouží k tomu `.actualplaylist`, který je před uživatelem skrytý.



Obrázek 4.2: Sekvenční diagram průběhu HTTP komunikace.

4.4 ALSA Streamer

Tato část programu propojuje HW se zbytkem systému, vytváří pakety a odesílá je. *ALSA Streamer*, stejně jako aktuálně používaný *CStreamer*, rozšiřuje třídu *Streamer*. Součástí

aktuálního software byl i soubor *CAlsaStreamer*. Který podle dostupných informací, sloužil jako ověření konceptu. Po jeho prostudování jsem usoudil, že to byl testovací prototyp, který nebyl funkční a převzal jsem jeho strukturu.

4.4.1 Inicializace ovladače a technické parametry

Ke zpřístupnění HW je třeba dodržet protokol, včetně netriviálního nastavení struktur a parametrů hardware. Na začátku je potřeba deklarovat *handle* pro PCM zařízení. Poté zvolit směr streamu, a to buď přehrávání (playback), nebo nahrávání (capture). Zároveň se nakonfigurují další parametry jako velikost bufferu, vzorkovací frekvence, formát vzorků.

```
// handle pro PCM
snd_pcm_t *pcm_handle;
// stream - nahravani
snd_pcm_stream_t stream = SND_PCM_STREAM_CAPTURE;
// Struktura s informacemi o hardware
//vyuziva se ke konfiguraci PCM streamu.
snd_pcm_hw_params_t *hwparams;
```

Nejdůležitějšími rozhraními jsou PCM zařízení *plughw* a *hw*. Při používání rozhraní *plughw* nezáleží tolik na fyzickém stavu hardware. Pokud hardware nepodporuje zadanou konfiguraci, například vzorkovací frekvenci nebo formát vzorku, tak je provedena konverze. Vše funguje a uživatel nic nepozná. Při použití *hw* se komunikuje přímo s jádrem ovladače, je třeba použít podporované parametry. Je to surová komunikace, bez konverzí. Uživatel je buď musí znát, nebo předem zjistit pomocí daného příkazu.

V aplikaci se pro nahrávání využívá *hw*. Díky tomuto přístupu by ovladač při nesprávné konfiguraci zahlásil chybu a tak je možné odhalit problém v hardware, nebo neshody v konfiguraci hardware a software. Je to také doporučený postup firmou Audified. Naopak pro přehrávání je využit *plughw*, aby bylo možné přehrát co největší spektrum formátů, vzorkovací frekvence ale musí být 48 kHz.

Následuje otevření PCM zařízení. Je dostupný blokující a neblokující mód. V neblokujícím módu je volání zápisu (čtení) okamžitě vráceno. Zpracována jsou data dostupná ve chvíli volání. Blokující volání blokuje, dokud nezpracuje daný počet vzorků. Před inicializací a použitím zařízení je třeba inicializovat a naplnit strukturu s parametry.

```
// Nazev zarizeni: hw:0,0
// Prvni cislo oznacuje zvukovou kartu, druhe je cislo zarizeni
char *pcm_name;
// Alokovani struktury snd_pcm_hw_params_t na zasobniku
snd_pcm_hw_params_alloca(&hwparams);

//Otevreni zaarizeni v blokujicim modu
if (snd_pcm_open(&pcm_handle, pcm_name, stream, 0) < 0) {
  fprintf(stderr, "Error opening PCM device %s\n", pcm_name);
  return(-1);

// Init hwparams
if (snd_pcm_hw_params_any(pcm_handle, hwparams) < 0) {
  fprintf(stderr, "Can not configure this PCM device.\n");
  return(-1);
```

```
}
```

Parametry se nastavují sadou funkcí `snd_pcm_hw_params_set_<parameter>`. Kromě výše zmíněných je důležitý ještě typ přístupu (access type). Určuje, jakým způsobem jsou data z více kanálů ukládána do bufferu. Bez prokládání `(NONINTERLEAVED)` se postupně ukládají jednotlivé kanály. V případě sterea by to znamenalo, že první půlku bufferu zaplní levý kanál a druhou pravý. V našem případě, u N-kanálového streamu dává větší smysl druhá možnost – prokládání `(INTERLEAVED)`. Nejdříve je uložen první vzorek z každého kanálu, následuje další vzorek, viz obrázek 4.3.

```
if (snd_pcm_hw_params_set_access(pcm_handle, hwparams,
                                 SND_PCM_ACCESS_RW_INTERLEAVED) < 0) {
  fprintf(stderr, "Error setting access.\n");
  return(-1);
}

// Aplikace nastaveni HW parametru
// PCM zarizeni a priprava zarizeni
if (snd_pcm_hw_params(pcm_handle, hwparams) < 0) {
  fprintf(stderr, "Error setting HW params.\n");
  return(-1);
}
```

Nyní je konečně možné vyčítat data ze zařízení. Využívá se ukazatel `pcm_handle`.

```
snd_pcm_readi(pcm_handle, data, num_frames);
```

Používané parametry jsou:

- vzorkovací frekvence: 48000 Hz
- formát vzorků: float little endian – float_LE
- počet vstupních kanálů: 0 – 15
- počet výstupních kanálů: 0 – 2

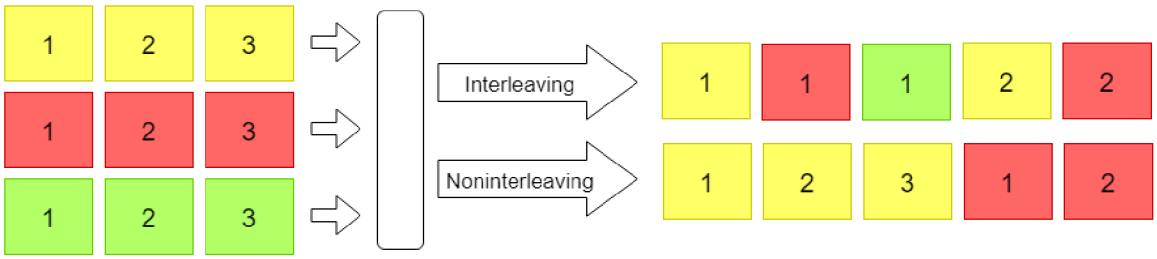
4.4.2 Čtení, způsob uložení dat

Ihned po inicializaci je zahájen sběr vzorků a jejich analýza. Výsledky analýzy (peak) jsou předávány pro zobrazení. Když neběží nahrávání, nemusí se sítí posílat všechna data. Stačí tyto agregované informace. Proto se ani nevyužívá dalších bufferů. Data se v cyklu pouze čtou a analyzují. Pro tuto činnost dostačují vestavěné buffery ovladače.

Při streamování se všechna data posílají do sítě. Proto je mezi čtením a odesláním kruhový buffer, který může uložit několik vzorků před odesláním, tím vyrovnat postupné odesílání paketů.

Data získáváme v prokládaném formátu (obrázek 4.3). To je velmi praktické z hlediska škálovatelnosti, protože se nemusí čekat na větší bloky. Je víceméně jedno, kolik kanálů se použije. Není vhodné rozdělit data kdekoliv, ideální je zarovnání na násobek počtu kanálů, ať je mezi odeslanými daty shodný počet vzorků pro všechny kanály.

Při tomto přístupu je díky prokládání zajištěná i časová synchronizace vzorků. Nestane se, že by byl některý kanál zpožděný.



Obrázek 4.3: Prokládání vzorků.

Kruhový buffer je tedy plněn zvukovými daty. Ta čte vlákno pro odesílání. Činnost těchto dvou vláken je třeba synchronizovat, aby bylo zajištěno, že se nepřepíší data čekající na odeslání a neodešlou se stará, dříve využitá data.

Při odvozování velikosti bufferu je třeba zvážit následující myšlenky:

- Data přibývají konstantní rychlostí pro každý kanál – rychlost roste lineárně s počtem kanálů
- Vzorky budou zarovnané, tzn. dělitelné počtem kanálů
- Buffer by měl být vytvořen pro obecný typ, ale počítat s velikostí vzorku – kolik bytů paměti zabere

Velikost bufferu se skládá ze zvolené konstanty, vynásobené velikostí vzorků a počtem kanálů. Vhodná velikost konstanty byla zkoumána během testování. Aby nebylo třeba synchronizovat každý vzorek, je buffer rozdělen do bloků. Je implementován jako 2D pole, kde první dimenze určuje blok, druhá vzorek v bloku. Velikost bloku je shodná s velikostí čtených dat. Počet bloků je zmíněná konstanta. Každý blok má dva mutexy. Jeden určuje zapsaná data, druhý přečtená. Vlákna si takto vzájemně vylučují přístup k daným blokům.

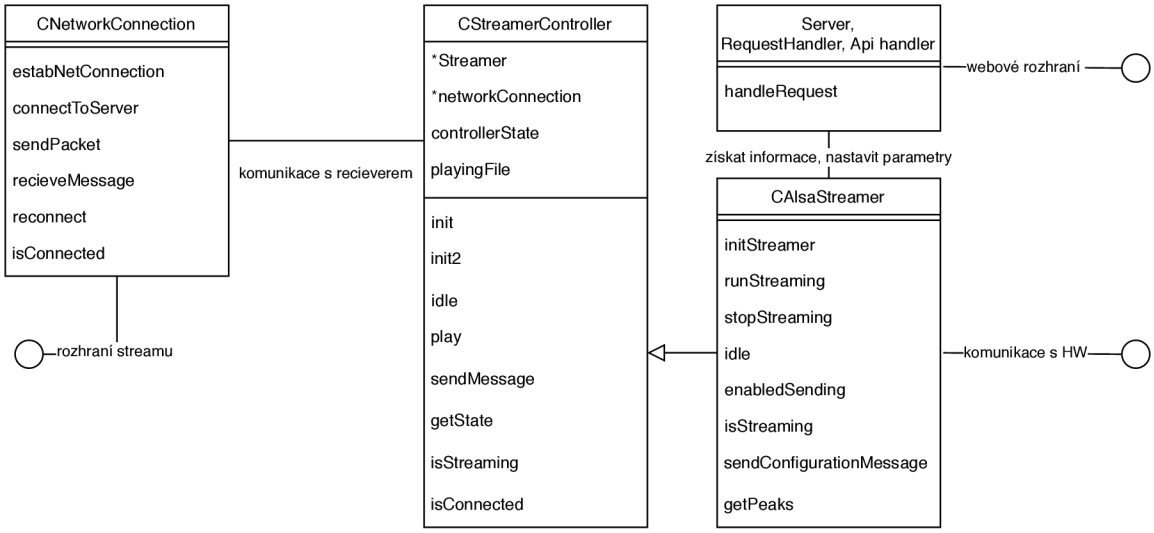
Mutex bloku je uvolněn ihned po odeslání, zároveň je nastaven příznak „ready“ na 0. Pokud ale odeslání selhalo (nepodařilo se do socketu zapsat všechna data), ukončí se odesílací vlákno Streameru.

4.4.3 Odesílání dat

Odesílání dat je přímočaré. Při streamování se spustí odesílací vlákno. To kontroluje mutex na kruhovém bufferu. Pokud se podaří uzamknout, znamená to, že jsou data připravena k odeslání. Zamkne se i druhý mutex na kruhovém bufferu, inkrementuje `timestamp` a data jsou odeslána po síti. Příslušná část bufferu se označí se jako odeslaná. Mutexy jsou uvolněny.

Při dodávání analyzovaných dat nedochází k odeslání v pravém smyslu. Výpočet probíhá bez ustání na všech příchozích datech. Výsledek se ukládá do sdílené paměti. Data se odesílají ve chvíli, kdy o ně někdo požádá. Odesílá se analýza veškerých dat zpracovaných od minulého přístupu.

Nestane se tedy, že by například některé peaky byly ignorovány. Na druhou stranu dochází k mírnému omezení uživatele. Data se změní každým přístupem. Neměl by přistupovat k datům ze dvou různých míst. Například mít otevřená dvě okna prohlížeče. Data v tom případě budou zkreslená a do každého půjde půl analýzy.



Obrázek 4.4: Znázornění tříd serverové části aplikace.

4.4.4 Streamer Controller

Touto třídou se ovládá Streamer. Je to střed celého systému, kde se potkávají data všech součástí.

Je zde spravováno síťové připojení, spouštěn hlavní cyklus. Zpracovávají se požadavky Recieveru. Tato třída zároveň poskytuje informace Webserveru. A to jak o stavu Streameru, tak o připojení.

Výzvou bylo najít způsob jak, umožnit ovládání nahrávání z Recieveru a Web serveru zároveň se synchronizací. Implementované řešení spočívá ve vytvoření handshake. Veškeré příkazy pro nahrávání půjdou přes Reciever, do kterého byla přidána možnost zpracování zpráv.

Průběh je následující. Uživatel zvolí ve webovém prohlížeči možnost nahrávání. Na port 80 se odešle požadavek pro zahájení nahrávání. Webserver pošle zprávu Recieveru „začni nahrávat“. Reciever zprávu zpracuje tak, že pošle požadavek o začátek streamování dat na Streamer Controller, zobrazeno v grafu 6.2. Analogicky funguje i zastavení, s tím rozdílem, že ukončení Recieveru a odpojení je detekováno a stream zastaven. Není třeba ukončovat zprávou. Při využití možnosti přehrávání je Controlleru předán playlist. V tomto případě nahrávání spouští i zastavuje Controller (zprávami přes Reciever).

Přehrávání zvuku

Před zahájením přehrávání je aktualizován soubor `.actualplaylist`. V něm je uložen seznam souborů. Ukládání v souboru jsem zvolil, aby si systém pamatoval poslední přehraný playlist. Ten se použije při spuštění přehrávání z Recieveru.

Jsou načteny názvy souborů. Soubory jsou postupně přehrávány příkazem `aplay`. Je vynucena vzorkovací frekvence 48000 Hz. Pokud náhodou soubor neexistuje, je přeskočen.

Při zahájení a ukončení přehrávání je odeslána zpráva s názvem přehrávaného souboru.

4.4.5 Reciever

Hlavní komplikaci byly s překladem vložené knihovny, která sice vyžadovala standardní závislosti, ale byla využita nestandardní cesta k souborům. Byla to absolutní cesta z jiného systému. Instalací příslušné verze překladače, přepsáním makefile a přeskočením testů bylo vše zprovozněno.

Co se úprav týče, tak zásadní bylo zbavit se části kódu, která se starala o bitové posuny a zpracovávala 24 bitové vzorky. Z ALSA ovladače jsou čteny vzorky ve formátu `float`, který má 32 bitů.

Další odstraňování se týkalo zastaralého nastavení gainů. Program vyžadoval cestu k souboru gainfile jako povinný parametr a jeho obsah odesílal ihned po inicializaci. Rozhraní pro příjem nastavení parametrů tímto způsobem už není aktuální, preferuje se nastavení pomocí webové aplikace.

Rozhraní bylo nahrazeno vzdáleným ovládáním, tak aby příjem dat byl řízen webovou aplikací. Ta může odesílat pokyny ke spuštění a zastavení nahrávání. Také se přenáší název audio souboru při spuštění a ukončení jeho přehrávání.

Při zapnutí zařízení se nastaví parametry na výchozí hodnoty. Nastavení se neukládá.

Kapitola 5

Vizualizační a řídicí aplikace v internetovém prohlížeči.

Jde o webovou aplikaci, která je zpřístupněna pouze lokálně. Aplikace je načtena po zadání IP adresy do internetového prohlížeče. Po připojení jsou přeneseny všechny soubory webové aplikace, o které webový prohlížeč požádá. ve vestavěném serveru jsou uloženy i veškeré styly a JavaScript knihovny, pro správnou funkci nemusí být počítač připojený k internetu.

5.1 Bootstrap, JavaScript, jQuery, JSON

Vyjmenované technologie jsou ve webových aplikacích standardem a pravděpodobně jsou všem známé. Přesto bych rád v krátkosti vysvětlil jejich použití v projektu.

Bootstrap

Je zde kvůli vzhledu. Pro rozložení využívám mřížkový systém (grid layout). Pro zobrazení úrovní jsou použité *progress bary*.

JavaScript

Je využitý pro generování stránky i veškerou funkčnost. Část stránky se skládá z několika řádků, které se od sebe liší pouze identifikátory prvků. Proto jsou vytvořeny cyklem v JavaScriptu. Je tak možné vždy vygenerovat přesně počet audio vstupů, se kterými se pracuje.

jQuery

Tato knihovna je připojena kvůli Bootstrapu. Zajišťuje ale i zpracování veškerých uživatelských vstupů a pomocí *setTimeout* pravidelně získává nové hodnoty úrovní.

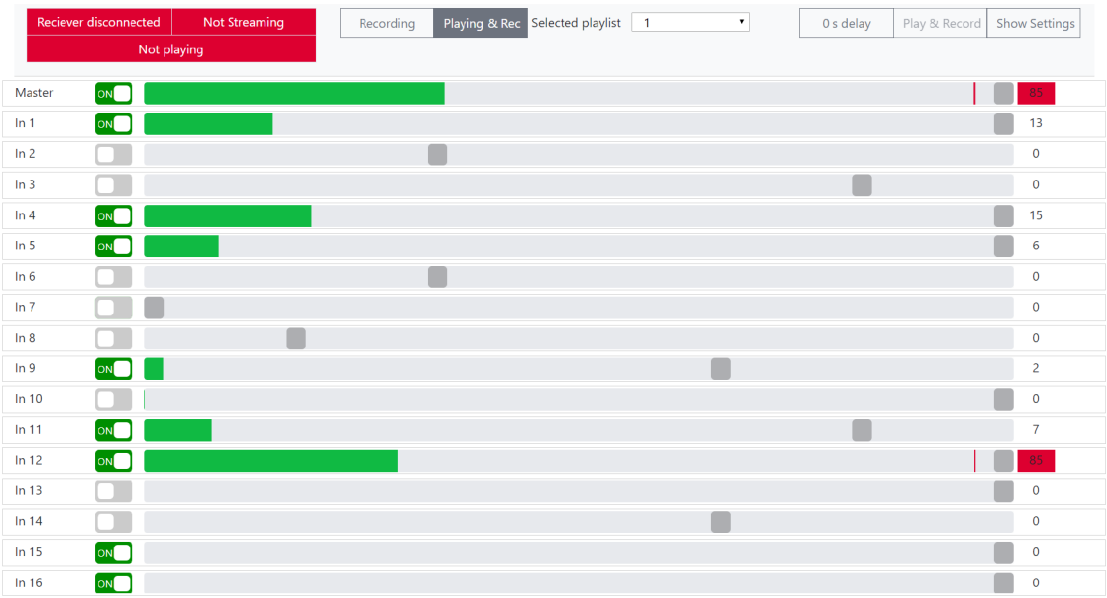
JSON

V tomto formátu posílá server data. Je použit, protože je standardní způsob předávání dat ve webových aplikacích a zpracování dat do proměnných v JavaScriptu je velmi snadné.

5.2 Vzhled a ovládání

Vzhled (5.2) je inspirovaný zvukařským mixážním pultem. Funkčnost totiž je velmi podobná. Převzal jsem myšlenku rozdělení jednotlivých vstupů. Každý vstup má své ovládací prvky. To je dobré pro rychlost ovládání a přehlednost. Vizualizace úrovní je také známá z různých aplikací pro práci se zvukem.

Zaměřil jsem se i na praktická hlediska s ohledem na tvorbu webových stránek. Celé zobrazení je například oproti zmíněnému zvukařskému pultu otočené o devadesát stupňů, aby bylo možné využít html *input range* bez transformací.



Obrázek 5.1: Ukázka webové aplikace.

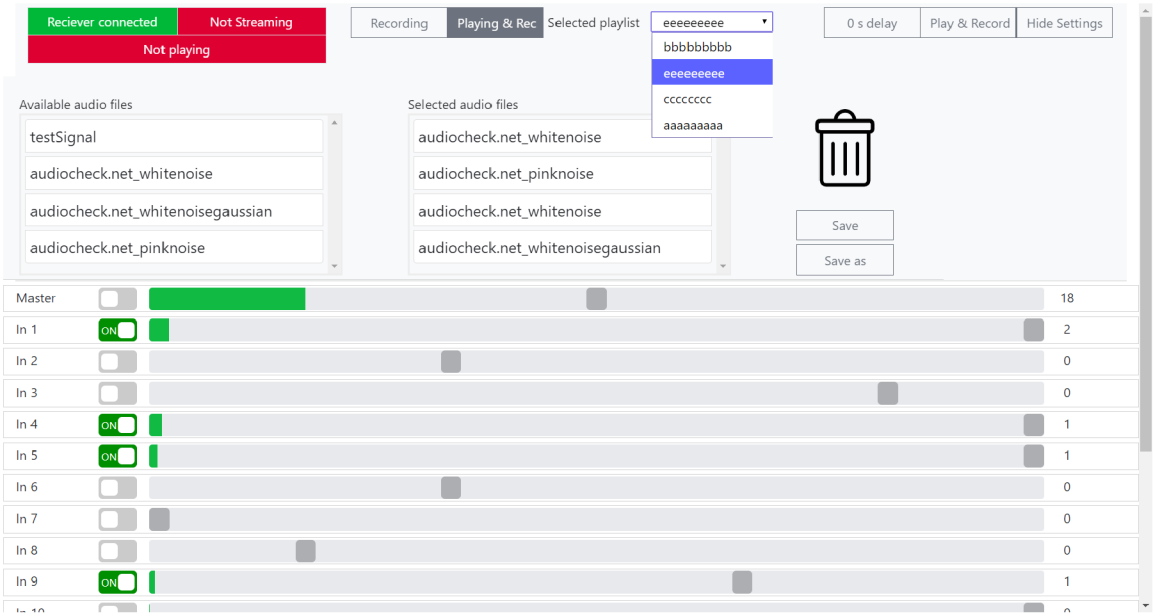
Nahoře se nachází stavové informace, ovládání a nastavení. V informacích se zobrazuje aktuální stav zařízení. Zda je připojený přijímač, jestli je aktivní streamování, případně jaký soubor se přehrává. Vedle je přepínač pro výběr mezi samotným nahráváním a nahráváním včetně přehrávání. Je možné omezit dobu nahrávání. Je zde i tlačítko pro nastavení zpoždění. Po kliknutí iteruje přes hodnoty 0, 5, 10, 15 a 20 sekund. Po kliknutí na tlačítko „show settings“ se zobrazí správa playlistů.

Hlavní část aplikace tvoří jednotlivé zvukové vstupy. Každý vstup je na jednom řádku. Je vidět název, přepínač phantomového napájení, zobrazení úrovně s „faderem“ pro nastavení gainu a vybuzení vstupu. Gain se udává v decibelech, hodnoty jsou 0-6 (0 = 0dB, 1 = 10dB, ... 6 = 60dB).

Ovládání je uzpůsobené pro myš a dotyk. Omezeně je možné využít i klávesnici (tabulátor pro pohyb, mezerník pro phantom a kurzorovými šipkami nastavení gainu).

5.3 Funkce

Po načtení stránky je odeslán požadavek *getall*. Tím se načte počet kanálů, seznam playlistů a souborů. Inicializují hodnoty phantom a gain. Aktualizace úrovní je prováděna periodicky v přednastaveném intervalu. Funkce pro nastavení hodnot je spuštěna vždy s uživatelským vstupem – kliknutí na phantom, nastavení gain, nebo uložení playlistu.



Obrázek 5.2: Ukázka webové aplikace – nastavení.

Interakce uživatele je zachycena událostmi *onchange* a *onclick*, které volají příslušnou funkci s parametry *id* a *hodnota*. Volaná funkce odesílá hodnoty na server.

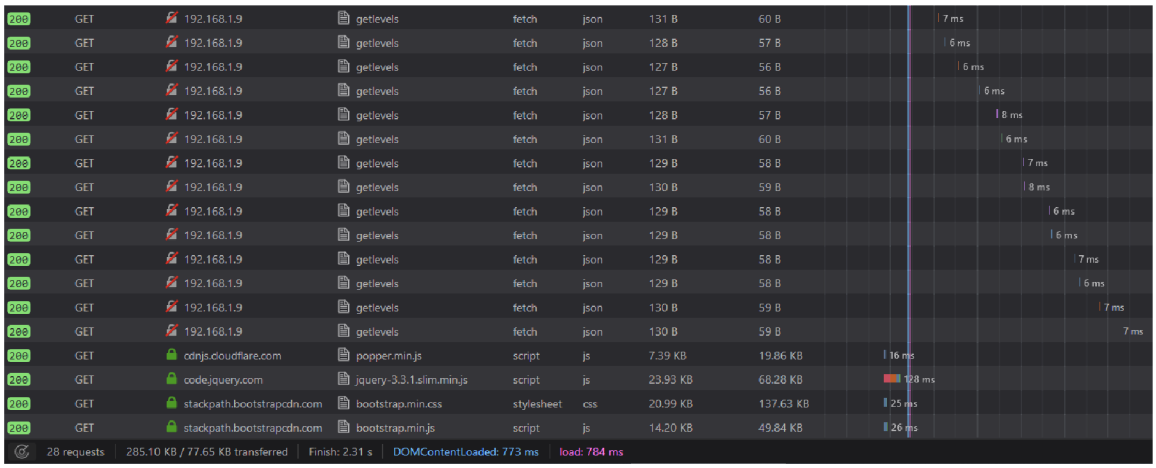
5.3.1 Správa playlistů

Je tu i možnost prohlížet a upravovat playlisty. Po rozkliknutí nastavení se zobrazí možnost výběru playlistu. Zvolený playlist je zobrazen a je možné jej upravovat.

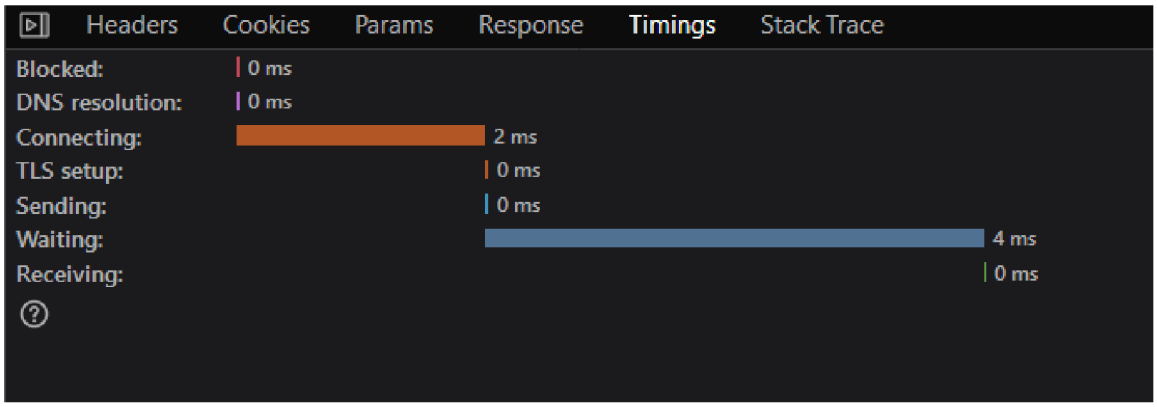
Možné úpravy jsou přidat a odebrat soubor, měnit pořadí. Je možné použít jeden zvukový soubor vícekrát.

Úpravy probíhají přetahováním položek. Přetáhnutím ze seznamu souborů na playlist je soubor přidán do seznamu. Přetáhnutím na ikonu koše je odebrán.

Playlist je možné uložit, případně uložit jako. Při zahájení nahrávání neuloženého seznamu se přehrají soubory aktuálně přiřazené ve webovém rozhraní. Ne ty, které jsou uložené v aktuálně otevřeném playlistu. Jinými slovy zvolený playlist je možné aktuálně pozměnit a změněnou verzi použít k přehrávání, bez změn v použitém playlistu.



Obrázek 5.3: Doba zpracování jednoho požadavku.



Obrázek 5.4: Požadavky webové aplikace.

5.4 Stabilita a rychlost

Očekává se zobrazení v reálném čase. Definovat reálný čas není snadné a elektronické zpracování vždy přinese nějaké zpoždění. Aplikace by neměla být nejpomalejším článkem řetězce. Rozhodl jsem se porovnávat její rychlost s obnovovací frekvencí monitoru. Zvolil jsem standardní 60 Hz monitor. Zobrazuje 60 snímků za sekundu. Snímek potřebuje zhruba každých 16 ms. Zpoždění aplikace měří snadno vývojářský režim webového prohlížeče. Na výpisu sítě 5.3 je vidět, že vyřízení požadavku od odeslání do přijetí trvá mezi šesti a osmi milisekundami. Časování průměrného požadavku je vidět na obrázku 5.4. Dvě milisekundy trvá připojování. Tento čas je možné zkrátit navrženou optimalizací. Prozatím je výkon dostatečný a není třeba aplikaci optimalizovat. Pokud se ukáže být zpracování dat serverem náročnější a tato doba se neúměrně prodlouží, bude muset předávání informací o úrovních probíhat jiným způsobem.

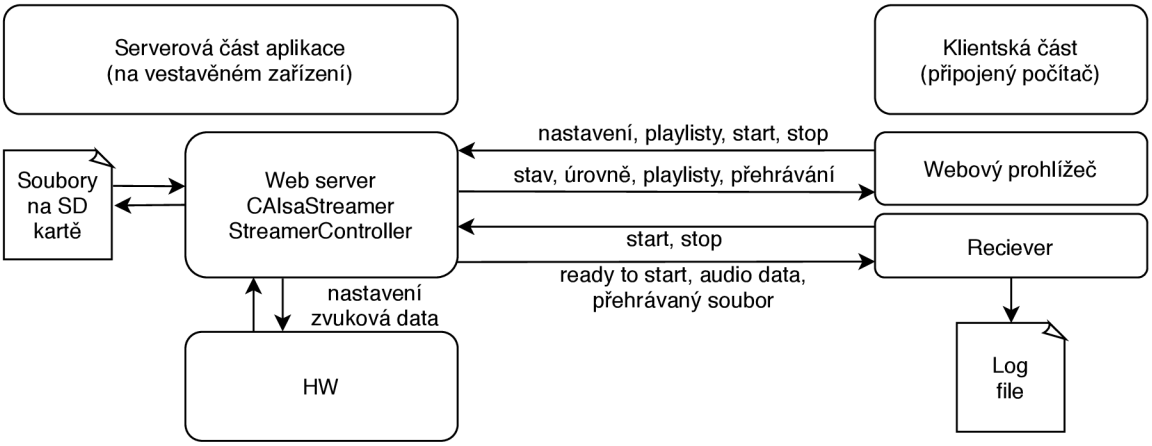
Se stabilitou jsem žádné problémy nezaznamenal. Pokud je server odpojen, přestanou se posílat data. Při opětovném připojení je třeba stránku aktualizovat a pokračuje jako by k výpadku nedošlo.

Po aktualizaci aplikace získá aktuální stav HW a tak nedojde k žádné inkonzistenci mezi reálným stavem a stavem aplikace.

Kapitola 6

Popis výsledné aplikace a práce se systémem

V kapitole 2.3 je popsán současný stav systému ve chvíli, kdy jsem začínal pracovat. Tato kapitola se zabývá výsledným stavem, použitím, ovládáním a některými specifickými vlastnostmi řešení. Na obrázku 6.1 je znázorněná komunikace jednotlivých částí systému po úpravách.



Obrázek 6.1: Scháma komunikace po úpravách.

6.1 Inicializace systému

Při připojení napájení se nabootuje systém, a ihned se spustí server na IP adrese 192.168.10.9, která je napevno určena v konfiguraci systému. Počítač musí pro připojení mít na daném síťovém rozhraní IP adresu stejné sítě.

Systém je v tuto chvíli ve stavu před inicializací – ta se dokončí až po připojení recieveru. V tuto chvíli je možné ve webovém rozhraní spravovat playlisty a nastavovat parametry. Zbytek funkcí je zpřístupněn právě až po připojení recieveru. Celý systém slouží k nahrávání, proto používání bez přijímače nedává smysl. Hodnoty úrovní se aktualizují každých 100 milisekund, tedy desetkrát za sekundu.

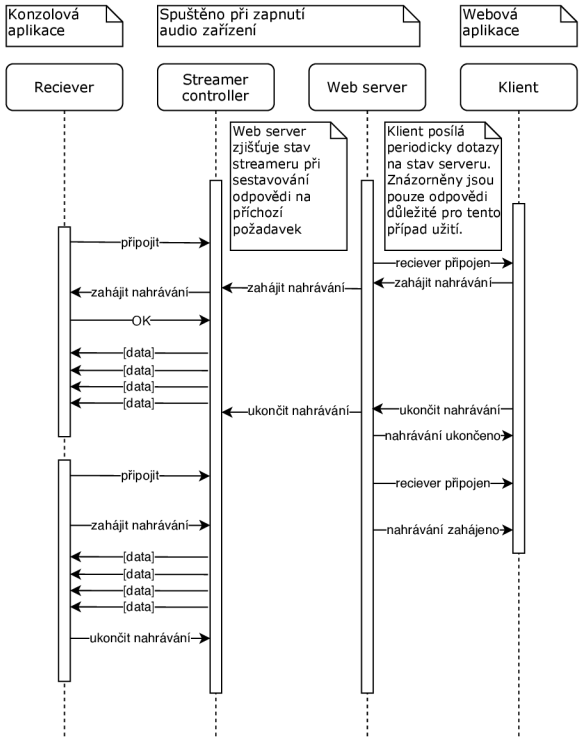
6.2 Nastavení parametrů

Po inicializaci jsou parametry nastavené tak, aby s nimi bylo co nejméně práce. Používají se mikrofony, které využívají phantomové napájení. Phantom je tak aktivován už při zapnutí. Gain je nastaven na střední hodnotu, 30 decibelů.

Tato nastavení je možné měnit v reálném čase. Změna nastavení se okamžitě projeví na zobrazené úrovni. K tomu je zde ještě master - ten upravuje všechny kanály. Jeho změna gainu se relativně projeví na všech vstupech (například pokud se master gain zvedl o dva, ke gainu všech kanálů se přičte 2 – pokud nepřekročí maximum. Analogicky funguje i snižování.)

6.3 Nahrávání a přehrávání

Princip nahrávání funguje tak, jak je vysvětleno výše. Spustit i zastavit nahrávání je možné jak z Recieveru, tak z webového rozhraní. Ve webovém rozhraní je i možnost samospouště (spuštění s časovým odkladem) a zvolit délku nahrávání (čas, po kterém bude nahrávání zastaveno). Při zvolení funkce nahrávání s přehráváním je činnost automaticky zastavena po přehrání posledního souboru v playlistu. Zároveň jsou do Recieveru posílány informace o aktuálně přehrávaném souboru. Ty jsou ukládány do souboru zvoleného parametrem *logfile*. Když je aktivní nahrávání, obnovovací frekvence aplikace je snížena, aby nezabírala přenosové pásmo a výkon serveru. Ty jsou vytíženy přenosem dat. I po snížení obnovovací frekvence je stále zachována správnost zobrazování úrovní vybuzení. Peak je vždy vypočítán ze všech vzorků od minulé aktualizace.



Obrázek 6.2: Diagram komunikace pro spuštění a zastavení nahrávání.

6.4 Nahraná data – raw formát

V počítači je tok dat opět rozdělen na jednotlivé kanály. Každý kanál se ukládá do jednoho souboru jako surová data (raw formát), vzorky jsou formátu float. Kanály jsou synchronizované, všechny soubory obsahují shodný počet vzorků.

Takto uložené soubory neobsahují metadata. Pro správnou interpretaci chybí informace o způsobu uložení dat a vzorkovací frekvenci. Pro rekonstrukci nahrávaného signálu je třeba tyto informace znát. Pro převod do audio formátu je možné využít například software SoX (Sound eXchange).

Kapitola 7

Testování a dosažené výsledky

Aby se vzorky nemusely testovat manuálně, je připraveno několik způsobů, jak zjistit že jsou data nahrána korektně. Data jsou kontrolována ve třech fázích zpracování.

1. Získaná data – fáze vyčítání dat z ALSA zařízení, základní ověření funkčnosti hardware
2. Odesílaná data – ověřuje se správnost uložení do kruhového bufferu, slouží k testování chyb ovlivněných posíláním dat
3. Přijatá data – finální ověření funkčnosti.

7.1 Nástroje

Při testování se porovnává velké množství vzorků a je zásadní, aby při nahrávání žádné vzorky nebyly vynechány. Testovalo se několika způsoby a nástroji.

7.1.1 Testovací skripty v C

Pro testování byly vytvořeny dva skripty, je možné nastavit počet testovaných kanálů, veškerá nastavení se provádí přímo ve zdrojovém kódu.

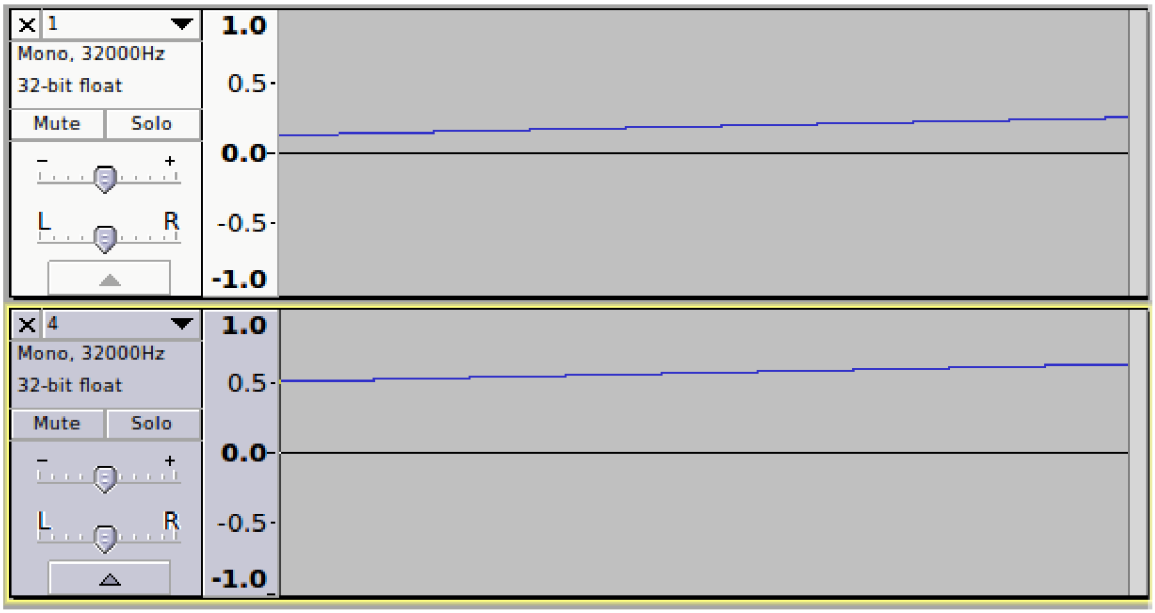
Writer

První ze skriptů vytváří vzorky. S těmito daty lze kontrolovat správnost přiřazení kanálů, pořadí vzorků, případné vynechání nějakého vzorku i synchronizace kanálů.

Vzorky jsou generovány tak, že první 4 bity (podle MSB) označují číslo kanálu. To dává možnost rozlišit od sebe 16 různých kanálů. Zbytek čísla je využitý jako čítač. Generovaná řada pro každý kanál je od vzorku `X000`, do `Xfff`, kde X je číslo kanálu. Celkem se vytvoří 4096 unikátních vzorků pro kanál. Skript vygenerované vzorky ukládá do souboru, který je při testování přehráván (obrázek 7.1).

Reader

Reader slouží ke čtení nahraných souborů a kontroluje, jestli obsahují správné (očekávané) hodnoty. Reader nejdříve nastaví zarovnání, protože kontrolovaný soubor nemusí začínat



Obrázek 7.1: Zobrazení vybraných generovaných testovacích dat.

prvním vzorkem. Potom je kontrolováno, že každý následující vzorek je o jedna větší (případně nula po dosažení hodnoty max), než vzorek předchozí. Výstup Writeru připojený na vstup Readeru musí být označen jako validní.

7.1.2 Audacity

Audacity je editor a nahrávací program pro práci se zvukem. V této práci byl program využitý k vizuální reprezentaci dat, přehrávání, vytváření wav a raw souborů a kontrolu stop.

Vizuální reprezentace a přehrávání

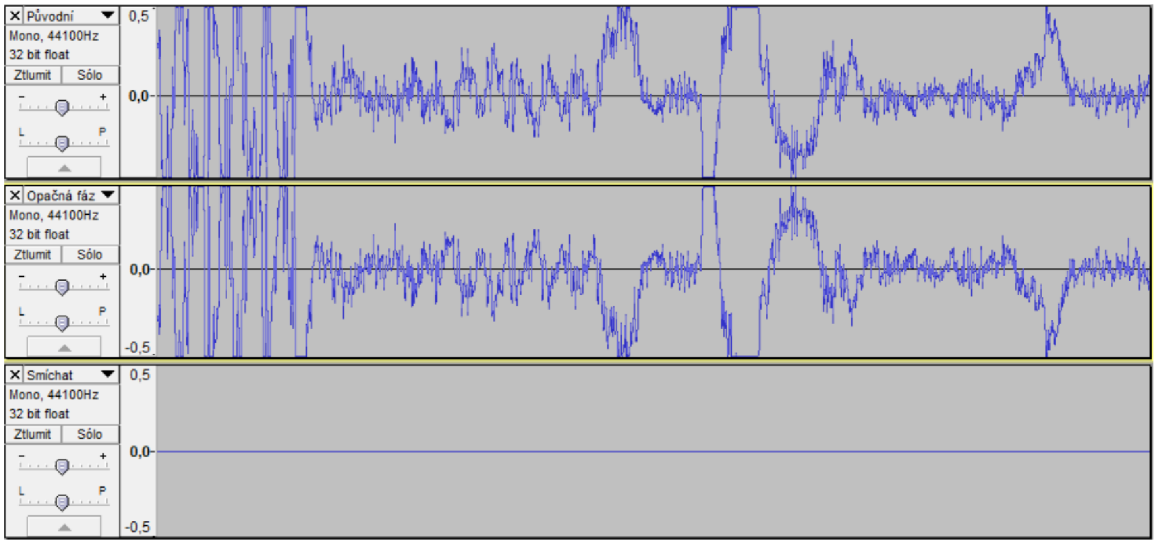
Po otevření dat je zobrazen jejich časový průběh. Je to rychlá metoda jak poznat, jestli je nahrávka podobná zdroji, nebo vůbec neodpovídá. Za čísly se schovává opravdový zvuk. Porovnání poslechem sice nedává žádné objektivní výsledky, velmi dobře jsou ale poznat problémy, kdy je při nahrání vynecháno několik vzorků. Nahrávka „přeskočí“, nebo hlasitě lupne. Rozeznatelný byl i problém s buffery, kdy se část kopírovala stále dokola a z řeči se stalo koktání.

Vytváření wav a raw souborů

Surová data jsou zmíněna v podkapitole 6.4. Formát wav na rozdíl od rawu obsahuje i metadata a tak je snadno přenositelný. Audacity® je při testování využito pro převod mezi zmíněnými formáty a vytváření dalších testovacích souborů.

Odečítání stop

Když mají dva signály opačnou fázi, tak se navzájem vyruší. Tato vlastnost byla využita na ověření, zda jsou stopy shodné. V Audacity® je funkce pro otočení fáze a sečtení stop.



Obrázek 7.2: Součet stop s opačnou fází.

Pokud jsou zdrojová a nahraná stopa časově synchronizované, tak po otočení fáze jedné z nich a jejich součtu vznikne nulový výsledek, viz obrázek 7.2.

7.1.3 Wireshark

Tento program slouží k analýze sítí. Byl používán na kontrolu připojení, posílaných paketů a dat v nich obsažených. Také sloužil pro kontrolu zahlcení sítě a přenosové rychlosti.

7.1.4 Feeder

Feeder je aplikace napsaná v C++. Pro účely ladění byl reálný vstup ze zvukových vstupů nahrazen souborem. Všechna data zapsaná do tohoto souboru fungují v systému stejně, jako by byla přímo z hardware vstupu.

Program Feeder simuluje vstupy tak, že čte připravené audio soubory a ve smyčce je zapisuje do `/dev/sharc_0`.

Pro ladění je velmi vhodné, že můžeme ovlivnit vstup a poslat na něj přesně data, která je potřeba zkoumat. Takovou simulací vytváříme laboratorní podmínky, kde se mohou přesně porovnávat původní a nahraná data.

Potenciální problém může nastat, pokud se vykytuje chyba, kterou by zamaskovalo právě periodické opakování signálu. Tomu lze předejít použitím různě dlouhých souborů, aby se chyba projevila alespoň v některém. Zrádné je v tomto případě používání mocnin dvou, lepší je využít náhodnou délku.

7.1.5 Soubory a formát dat

Zde je pro přehlednost seznam použitých souborů.

1. Feeder pracuje s 16 bit raw soubory. Pro každý kanál jeden soubor.

2. Z Alsy jsou nahrávána data v prokládaném raw formátu. Nejdříve 16 bitový fixed point, potom float 32 bit. V tomto formátu jsou dat odesílána.

3. Reciever data rozdělí na jednotlivé kanály. Výsledkem je jeden raw soubor pro každý kanál.

4. Pro přehrávání v zařízení se využívají wav soubory. Při přehrávání se použije konfigurace uložená v souboru. Je potřeba dodržet vzorkovací frekvenci 48 kHz. Kanálů se využije tolik, kolik je v zařízení aktivních výstupů.

7.2 Způsob testování

Manuální kontrola není reálná, proto jsou navrženy různé způsoby testování, jak s vytvořenými vzorky, tak s reálnými daty. Systém data zpracovává od úrovně hardware přes odeslání, k uložení do počítače. Testování probíhalo nejdříve po jednotlivých částech a následně byla kontrolována funkčnost celého systému jako celku.

7.2.1 Data vyčtená z hardware

První kontrolou dat je výpis do souboru ihned po jejich získání. Data se získávají příkazem `snd_pcm_readi()`. Jde především o kontrolu konfigurace. Používají se umělá data vygenerovaná aplikací Writer, výsledná data jsou porovnána pomocí aplikace Reader.

Test pro syntetická data velikosti 4096 vzorků pro každý kanál procházel spolehlivě, ale jakákoliv jiné testovací nahrávky byly deformované. Problém byl ve velikosti bufferů, která odpovídala velikosti dat, ale buffery se naplnily pouze jednou a potom odesílaly stále stejná data.

Dále se při použití celé serverové aplikace začala v souborech objevovat prázdná místa. V určitých chvílích byly zaznamenávány pouze nulové vzorky. Při změně velikosti paketů se mění délka intervalu, kdy se nenahrává, zároveň tato „hluchá“ místa vznikají ve chvíli, kdy se začne přehrávat audio soubor. Problém se projevoval i bez spuštěného Recieveru.

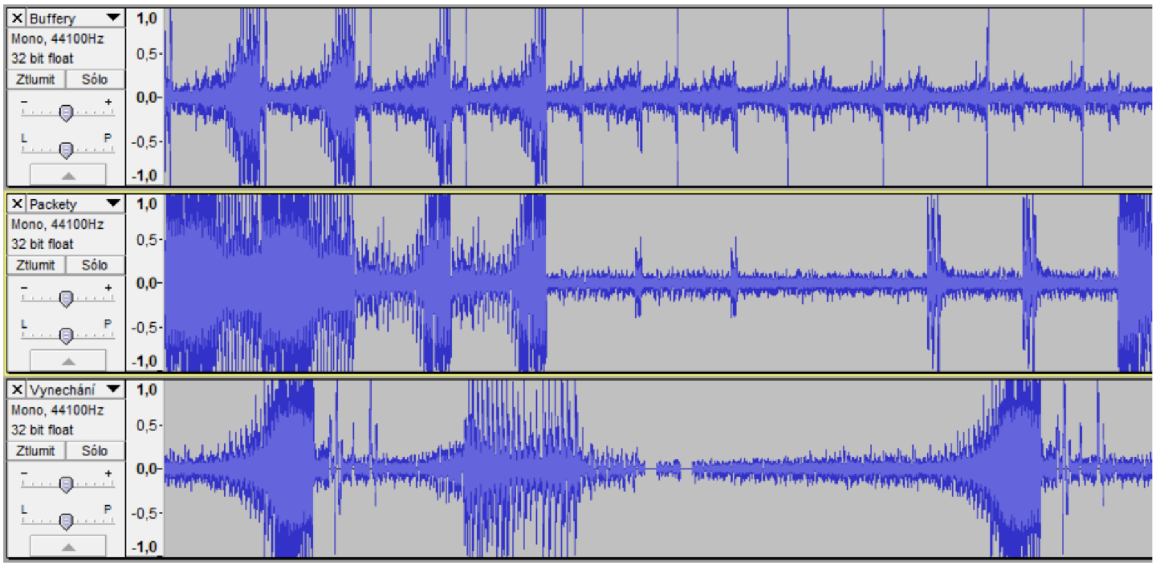
Podstatné je, že byly nahrány nulové vzorky. ALSA nedávala žádné chybové údaje (při jakékoliv chybě se ukončí program, aby se zamezilo chybné načítání dat). A tato chyba se nikdy neobjevila s reálnými vstupy. Z toho usuzuji, že byl omezený výkon programu Feeder, který nestíhal dodávat data do souboru, ze kterého se vyčítají, proto byly nahrazeny nulami. Po mírné optimalizaci, která spočívala ve změně velikosti paketů, se problém přestal projevovat.

7.2.2 Odeslaná data

Další testovací metoda je porovnání velikosti dat a času nahrávání. Do výsledku vstupuje chyba měření, která se dá velmi dobře minimalizovat opakovaným měřením a delšími úseky měření. Samozřejmě nikdy nebude metoda přesná na vzorek. Časová přesnost je na milisekundy. Podle velikosti vzorku a vzorkovací frekvence lze vypočítat přesnou očekávanou velikost souboru za určitý čas. Že se neztrácejí jednotlivé vzorky bylo ověřeno syntetickými testy. Touto metodou se testuje, zda nedochází ke ztrátě paketů, nebo jiným větším chybám při přenosu do počítače.

7.2.3 Přijatá data

U přijatých dat se opět kontroluje, zda se někde neztrácejí pakety, dalšími potenciálními problémy jsou správné rozdělení na jednotlivé kanály a bezchybné ukládání do souboru. Porovnávání dat před a po odeslání je možné například programem diff. Nejdříve je ale



Obrázek 7.3: Ukázka různě poškozených dat.

třeba nahraná data zarovnat se zdrojovými daty. Po zarovnání je možné různými způsoby porovnat shodnost jednotlivých souborů.

Výsledná použitá velikost paketu je 128 vzorků z každého kanálu. Paketu se skládá ze 128 vzorků, vynásobeno šestnácti kanály a čtyřmi byty na vzorek, k tomu deset bytů pro metadata (typ zprávy, timestamp, počet bytů vzorku a kanálu). Celková velikost vychází cca 8 MB. Technicky se nejedná o paket, ale o blok dat, který je zapisován do socketu.

7.2.4 Přehrávání

Tato část je otestovaná pouze v software. Hardware v poskytnuté konfiguraci neobsahuje žádné výstupy. Ani ve spolupráci Audified se nám nepodařilo potřebné moduly včas zprovoznit. I když není vyzkoušeno na hardware, předchozí testování ukázalo, že chování ALSA zařízení je při simulaci shodné s chováním hardware. Vše tedy nasvědčuje tomu, že přehrávání funguje.

1. Audified má otestováno, že hardware při použití Alsy zvládá propustnost 16 + 16 kanálů. Víc nebylo testováno.

2. Software se chová, jako by přehrával – posílá do ovladače data.

3. Při testování s ovladačem přesměrovaným do souboru vše fungovalo.

7.2.5 Výkon aplikace

Výkon z pohledu webové aplikace je popsaný v kapitole 5, testování bylo zaměřeno především na latenci a odezvu. U části aplikace, která běží na ARM procesoru má smysl zjišťovat, kolik využívá systémových prostředků. Souhrn využití paměti a procesoru je v tabulce 7.1.

Vysoké nároky má streaming také na paměť připojeného zařízení. Každá minuta záznamu má pro 16 kanálů velikost přibližně 176 MB. Den záznamu odpovídá zhruba 250 GB
.

Stav zařízení	využití paměti [%]	využití procesoru [%]
Po spuštění	0	5
Reciever připojen	11	15
Stream	14	85 – 95

Tabulka 7.1: Využití prostředků pro 16 kanálů

7.3 Shrnutí výsledků

Na obrázku 7.3 je ukázka několika různě poškozených dat.

Většina testování probíhala simulací na základní desce. Po odladění bylo vše otestováno i s připojenými hardwarovými vstupy. K dispozici bylo zařízení se šestnácti vstupy, což odpovídá maximální testované propustnosti deklarované firmou Audifed.

Nejdelší provedené testování trvalo cca 2 hodiny, to je záznam velikosti 22 GB. To exponenciálně převyšuje velikost vyrovnávacích pamětí, problémy způsobené nedostatkem paměti by se projevily dříve. Použity byly audio soubory náhodné velikosti přehrávané ve smyčce programem Feeder, upraveným na 32 bitové vzorky. Nejdříve bylo testováno, zda odpovídá velikost souborů očekávání. Následně byly jednotlivé kanály zarovnány na začátek přehrávaného souboru a zkontrolovány na shodnost s předlohou v cyklu.

Testování potvrdilo, že systém funguje téměř tak, jak má.

- Z Alsy jsou vzorky vyčítány spolehlivě.
- Vzorky jsou správě odesílané po síti.
- Přenosová rychlost se pro 16 kanálů pohybuje v rozmezí 2,9 – 2,95 MBs^{-1}. To v průměru odpovídá očekávané hodnotě 2,93 MBs^{-1}.
- Nahrané soubory jsou kompletní.
- Frekvenci aktualizace zobrazování úrovní byla omezena na 10 Hz. Větší rychlost je možná, ale subjektivně u rychlejší aktualizace není rozdíl. Není tedy potřeba více zatěžovat zdroje.
- Nastavení parametrů funguje spolehlivě.
- Přehrávání je otestováno na základní desce. Pravděpodobně nic nebrání tomu, aby fungovalo i na hardware.

Kapitola 8

Závěr

V práci jsem se seznámil s dostupným HW a SW, navrhl a implementoval řešení a ukázal, že funguje. Seznámil jsem se SW, který je používán, ale není dokumentovaný, ani podporovaný na novějším typu zařízení. Zároveň nebyl nikdo z autorů dostupný pro úpravy. Tento kód jsem dokázal pochopit a aktualizovat pro současné požadavky.

Výsledkem je aplikace, která byla vložena do vestavěného Linuxu a tak integrovaná do zvukového systému. Při zapnutí zařízení je aplikace spuštěna. Při spuštění je inicializován hardware, zpřístupněn web server a rozhraní pro komunikaci s konzolovou aplikací.

Tato aplikace umožňuje vyčítání dat z hardware, dokáže data analyzovat a výsledek je zpřístupněn na webovém rozhraní. Přes toto rozhraní je dále možné ovládat systém, nastavovat parametry hardware a pracovat s playlisty. Úrovně vybuzení se získávají se zpožděním cca 10 milisekund, dá se říct, že jde o zobrazení v reálném čase.

Webová aplikace je načtena v internetovém prohlížeči po připojení k serveru. Aplikace dokáže zobrazovat úrovně vybuzení jednotlivých kanálů, zobrazovat audio soubory uložené na zařízení, prohlížet, vytvářet a ukládat playlisty, ovládat parametry systému a zobrazovat jeho stav.

Reciever je konzolová aplikace, která slouží k příjmu a ukládání dat. I touto aplikací je možné spustit a zastavit nahrávání dat. Části kódu byly převzaty z původního řešení, všude jsem ale bylo třeba upravovat kritické sekce. U Streameru to bylo vyčítání dat z ALSY a jejich balení do paketů. Reciever bylo třeba upravit pro aktuální formát dat. Celý tento SW jsem potom upravoval pro příjem konfiguračních zpráv a předávání dat web serveru.

Literatura

[1] ABRAHAMS, D.: *Boost.org*. [Online; navštíveno 16.12.2018].
URL `https://www.boost.org/`

[2] ANALOG DEVICES: *ADSP-SC589*. [Online; navštíveno 10.12.2018].
URL `https://www.analog.com/en/products/adsp-sc589.html`

[3] ARM Developer: *Cortex-A5*. [Online; navštíveno 10.12.2018].
URL `https://developer.arm.com/products/processors/cortex-a/cortex-a5`

[4] KOHLHOFF, C. M.: *HTTP Server*. [Online; navštíveno 04.11.2018].
URL `https://www.boost.org/doc/libs/1_69_0/doc/html/boost_asio/examples/cpp11_examples.html#boost_asio.examples.cpp11_examples.http_server`

Příloha A

Obsah přiloženého paměťového média

A.1 Software

A.1.1 Zdrojové soubory k implementovanému systému

A.1.2 Programy a knihovny pro křížovou kompilaci

A.1.3 Readme

A.2 Videoprezentace práce

A.3 Plakát