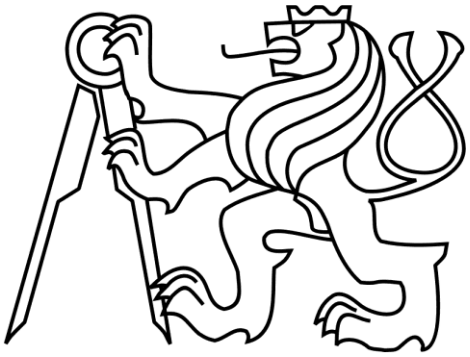


České vysoké učení technické v Praze

Fakulta elektrotechnická
Katedra počítačové grafiky a interakce

SADA ZVUKOVÝCH NÁSTROJŮ

BAKALÁŘSKÁ PRÁCE

Jakub Sebastián Andráš

Program: Softwarové inženýrství a technologie

Vedoucí práce: Ing. Roman Berka, Ph.D.

2022

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Andráš** Jméno: **Jakub Sebastián** Osobní číslo: **475754**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačů**

Studijní program: **Softwarové inženýrství a technologie**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Sada zvukových nástrojů

Název bakalářské práce anglicky:

Sound processing toolbox

Pokyny pro vypracování:

Prozkoumejte možnosti programování platformy Raspberry Pi v oblasti zpracování a interpretace zvukového signálu a na základě výsledků z předchozích semestrálních projektů navrhněte sadu generátorů a filtrů přijímajících povely (select, echo, cross-fade, apod.) a parametry, pro generování a modifikaci zvukových signálů na této platformě. Dále navrhněte a implementujte aplikaci pro OS Android, která reprezentuje funkční uživatelské rozhraní ovládající navržený řetězec generátorů na Raspberry Pi. Vytvořte alespoň 5 filtrů, kde každý filtr nebo generátor bude reprezentovat jeden efekt. Postup návrhu a implementace v pravidelných intervalech konzultujte s vedoucím práce a v textu práce zdokumentujte, případně zdůvodněte, postup návrhu a všechny kroky jeho realizace. Výstup práce otestujte na vhodném výběru uživatelů.

Seznam doporučené literatury:

[1] Poepel, Cornelius, and Roger B. Dannenberg. "Audio signal driven sound synthesis." In ICMC. 2005.
[2] De Poli, Giovanni. "A tutorial on digital sound synthesis techniques." Computer Music Journal 7, no. 4 (1983): 8-26.

Jméno a pracoviště vedoucí(ho) bakalářské práce:

Ing. Roman Berka, Ph.D. Katedra počítačové grafiky a interakce

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) bakalářské práce:

Datum zadání bakalářské práce: **07.02.2022** Termín odevzdání bakalářské práce: **20.05.2022**

Platnost zadání bakalářské práce: **30.09.2023**

Ing. Roman Berka, Ph.D. podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Mgr. Petr Páta, Ph.D. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Student bere na vědomí, že je povinen vypracovat bakalářskou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v bakalářské práci.

. Datum převzetí zadání

Podpis studenta

PODĚKOVÁNÍ

Tímto chci poděkovat mému vedoucímu práce za všechny rady, zpětnou vazbu a za flexibilní přístup ke konzultacím.

Dále nesmím zapomenout na řadu mých přátel, kteří mě vždy dokázali povzbudit a vyslechnout si můj nářek, a na ochotné testery, kteří si dokázali najít čas, energii a vstřícnost. Děkuji.

PROHLÁŠENÍ

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

V Praze dne 20. 5. 2022

.....

Jakub Sebastian Andráš

ABSTRAKT

Sada zvukových nástrojů

Tato práce pojednává o možnostech digitální zvukové syntézy se zaměřením na hudební efekty, a to způsobem návrhu a implementace systému na zpracování a interpretaci zvukového signálu v reálném čase prostřednictvím Raspberry Pi v kombinaci s Android mobilní aplikací, která slouží jako grafické uživatelské rozhraní tohoto systému. Komunikace mezi Raspberry Pi a mobilní aplikací je realizována pomocí Bluetooth technologie. Pro použitelnost řešení je zcela kritická kvalita implementovaných efektů a nerozpoznatelně krátká latence celkového zpracování zvuku. Tyto podmínky, a funkčnost systému obecně, jsou ověřeny testováním s hudebníky, kteří představují cílovou skupinu uživatelů této práce.

Klíčová slova: zvuk, syntéza zvuku, efekt, kytara, C++, mobilní aplikace, Android, Flutter, Bluetooth

ABSTRACT

Sound processing toolbox

This work deals with the options of digital sound synthesis, focusing on music effects. It contains a design and implementation of system to processing interpretations of sound signal in real time via Raspberry Pi in combination with Android mobile app, which serves as GUI of this system. Communication between Raspberry Pi and mobile app is realized by using the Bluetooth technology. For a useability of the system is strongly critical quality of implemented effects and by the human ear unrecognizable latency overall processing. This conditions and system function in general are verified and tested by musicians, who represent the target group of users for this work.

Keywords: sound, sound synthesis, effect, guitar, C++, mobile app, Android, Flutter, Bluetooth

Obsah

ZKRATKY

Definice použitých zkratek a pojmů.

DSP	Digitální zpracování signálu	(Digital Signal Processing)
FIR	Konečná impulsní odezva	(Finite Impulse Response)
IRR	Nekonečná impulsní odezva	(Infinite Impulse Response)
MIDI	Hudební digitální rozhraní	(Musical Instrument Digital Interface)
GUI	Grafické uživatelské rozhraní	(Graphic User Interface)
DAC	Digitálně analogový převodník	(Digital-to-Analog Controller)
ADC	Analogově digitální převodník	(Analog-to-Digital Controller)
I/O	Vstup/Výstup	(Input/Output)
OS	Operační systém	(Operating system)
API	Aplikační programovací rozhraní	(Application Programming Interface)
JSON	JavaScript notační objekt	(JavaScript Object Notation)

1 ÚVOD

Možností digitálního zpracování zvukového signálu (DSP) existuje celá řada. Od čistě softwarových variant, přes kombinaci plošných spojů a jednoduchých algoritmů, až po MIDI kontrolery.

Náplní této práce je v první řadě prozkoumat již existující řešení DSP se zaměřením se na jednodeskové počítače a mobilní aplikace, dozvědět se víc o technikách zvukové syntézy a fyzikálně-matematickém pohledu na kytarové efekty, a poté tyto znalosti využít k implementaci některých známých efektů na vhodně zvoleném hardwaru. V tomto konkrétním případě je volba hardwaru značně zjednodušena, neboť již zadání této práce jasně specifikuje, že se má jednat o Raspberry Pi. Ve druhé fázi této práce potom navrhnout a realizovat mobilní aplikaci, která zprostředkuje rozhraní po obsluhu efektů a komunikaci mezi mobilním zařízením a Raspberry Pi.

1.1 Existující řešení

Dva níže popsané produkty vytvořila internetová komunita ElectroSmash. Zaměřují se na Open Source & Open Hardware návrh a realizaci hudební elektrotechniky, zejména na kytarové efekty, multiefekty a zesilovače. Jako základ svých řešení používají Arduino, Raspberry Pi nebo jiný, podobně koncipovaný hardware.

Poslední zmíněný produkt je velice známá mobilní aplikace Tonebridge, která dokáže v reálném čase s minimální latencí zpracovávat a editovat audio signál, a to vše pouze prostřednictvím mobilního telefonu či tabletu.

1.1.1 PedalSHIELD

Jedná se o sérii již tří programovatelných pedálů pojmenovaných pedalSHIELD UNO, DUE a do třetice MEGA. Všechny tyto projekty jsou postavené na jednodeskových počítačích Arduino, rozšířené o I/O audio konektory a tlačítka pro rychlou a snadnou interakci s uživatelem. Primární programovací jazyk je C/C++. Je to skvělá možnost pro začínající programátory či hudebníky se zájmem o digitální zpracování zvuku. Cena za první dva modely je přibližně 30$, model MEGA vyjde na 45$. Každý model se ale dodává bez základní Arduino desky, tu si musí zákazník koupit zvlášť [11]. Celý systém můžeme rozdělit na tři části:

1. Vstupní – Zesiluje a filtruje zvukový signál.
2. Arduino – Přijímá navzorkovanou podobu signálu od ADC a provádí veškeré digitální zpracování signálu.
3. Výstupní – Nově vytvořený signál je převzat z výstupu a odeslán do dalšího zapojeného zařízení, ať to už je pedál nebo kytarové kombo.

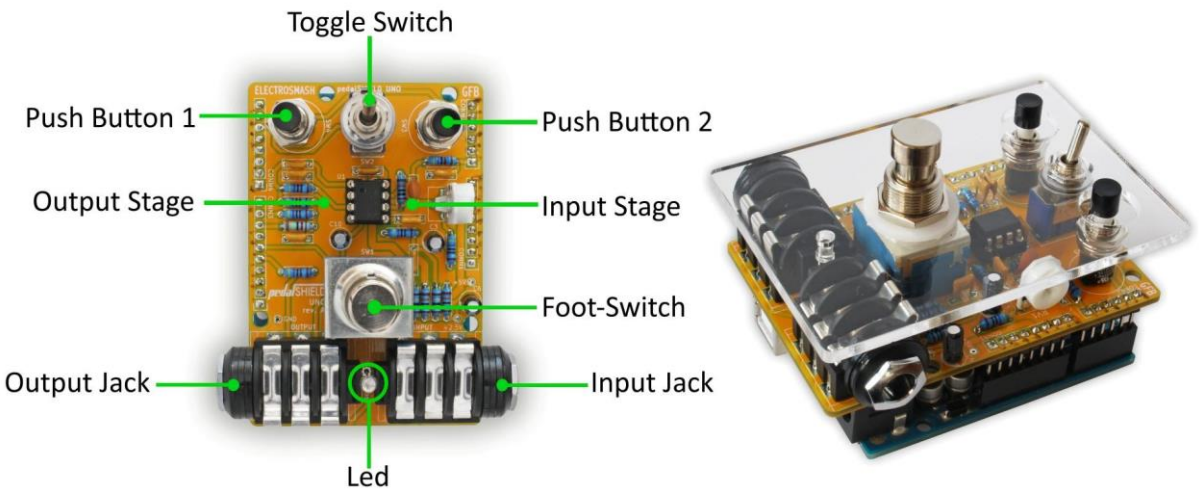
Výhody:

- cenově dostupné
- tvorba vlastních efektů
- možnost použití již předpřipravených efektů
- aktivní komunita

Nevýhody:

- oficiální prodej ukončen
  - lze sehnat jako použité či z jiných neoficiálních zdrojů
- nutnost sestavení z dílčích komponent
- nutnost dokoupit Arduino zvlášť
- v jednom okamžiku **pouze jeden efekt**

- lze částečně vyřešit naprogramováním mnohonásobných efektů
- omezené možnosti zejména pro pokročilejší uživatele

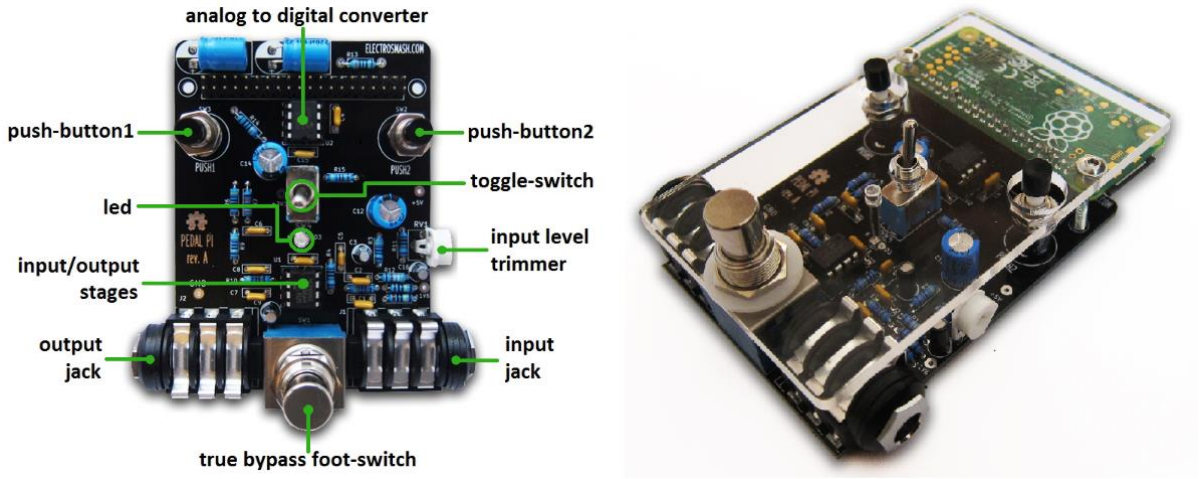


Obrázek 1 – PedalSHIELD UNO model [11]

1.1.2 Pedal PI

U tohoto produktu od ElectroSmash je situace velice podobná jako u již zmíněné série pedalSHIELD. Jediný zásadní rozdíl je, že Pedal PI je navržený pro použití v kombinaci s Raspberry Pi ZERO, tedy pro nejmenší, nejslabší výkonem, ale i nejlevnější jednodeskový počítač od Raspberry Pi. Ideální pro vzdělávací účely v rámci školství nebo pro samouky a nadšence.

Rozdíly oproti pedalSHIELD pro Arduino tu samozřejmě jsou, hlavně co se týče specifikace produktu a elektrotechnického pohledu, nicméně pro potřeby této práce nejsou nijak podstatné. Systém přenosu a zpracování signálu je u obou produktů téměř stejný.



Obrázek 2 – Pedal PI [11]

1.1.3 Tonebridge

Jak už bylo výše řečeno, Tonebridge je volně dostupná mobilní aplikaci, která dokáže v reálném čase zpracovávat audio signál a na základě předpřipravených presetů ho upravit a přeposlat na výstupní zařízení. Narozdíl od většiny jiných produktů tohoto typu, v této aplikaci každý preset představuje

originální zvuk určité písně, tudíž uživatel nepotřebuje žádné znalosti kytarových efektů. Stačí si vybrat svoji oblíbenou píseň a hrát [13].

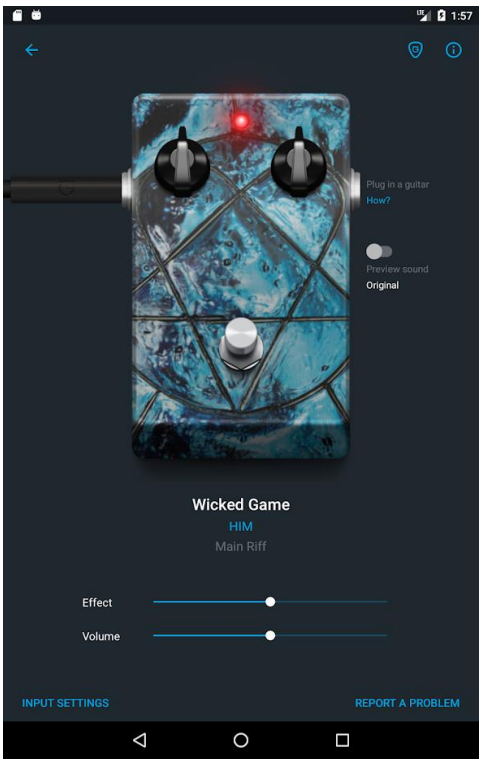
Výhody:

- víc jak 9000 presetů známých písní
- přes 7500 demo nahrávek u vybraných presetů
- podpora Android i iOS zařízení
- doporučené nastavení pro daný preset
- možnost tvorby pedalboardy z až 4 libovolných presetů
- dostatečně nízká latence u iOS zařízeních

Nevýhody:

- absence základních kytarových efektů
- Android zařízení mají **problémy s latencí**, často až znemožňují hraní
- nepodporuje USB, uživatelé jsou odkázáni na sluchátkový port Jack

Na koncerty a pódia se nehodí, avšak pro domácí či amatérské hraní dokáže být velice přínosná.

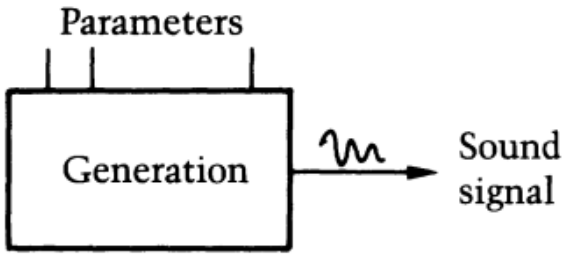


Obrázek 3 – Tonebridge preset [13]

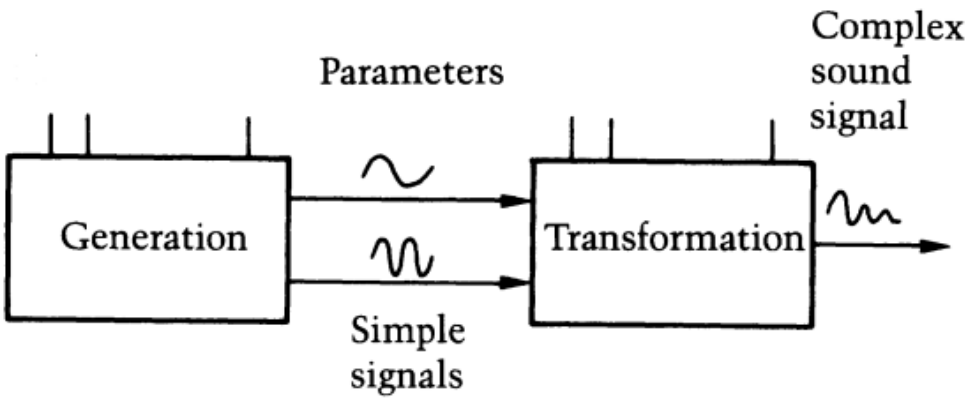
2 ANALÝZA

Syntéza zvuku je proces vytváření zvuku bez pomoci akustických nástrojů. V digitální syntéze je zvuk reprezentován posloupností čísel neboli vzorků (samplů). Tato technika se tedy skládá z výpočetního postupu nebo matematického vzorce, pomocí kterého se počítá hodnota každého vzorku. Typicky vzorec syntézy závisí na několika hodnotách, přesněji parametrech. Mezi nejznámější patří například frekvence a amplituda.

Techniky syntézy lze klasifikovat jako generační techniky (Obrázek 4 – Generační technika), které přímo produkují signál z vstupních dat a transformační techniky (Obrázek 5 – Transformační technika), které se dělí na dvě fáze, a to generování jednoho nebo více jednoduchých signálů a jejich následnou modifikaci. Nejčastěji se však používají více či méně propracované kombinace těchto technik [1].



Obrázek 4 – Generační technika [1]

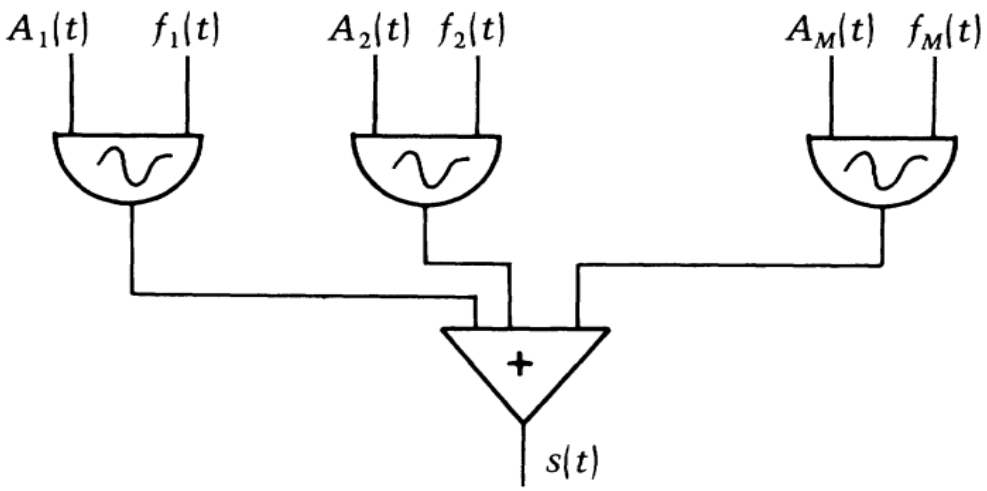


Obrázek 5 – Transformační technika [1]

2.1 Aditivní syntéza

Aditivní neboli součtová syntéza se řadí mezi první z technik generování zvukového signálu. Výstupní komplexní zvuk vzniká prostřednictvím skládání elementárních zvuků do jednoho celku (Obrázek 6 – Aditivní syntéza). Jakýkoliv téměř periodický zvuk lze aproximovat jako součet sinusoid. Frekvence každé sinusoidy je téměř násobek dané základní frekvence a každá sinusoida se vyvíjí v čase. Pro vyšší přesnost lze frekvenci každé složky považovat za pomalu měnící se. Aditivní syntéza se tedy skládá ze sinusových oscilátorů, jejichž amplituda a frekvence se v čase mění [1].

Tato technika dokáže poskytnout dobrou reprodukci i neperiodických zvuků. Její problém ale vzniká kvůli nutnosti specifikovat velké množství dat pro každý tón. Pro každou komponentu musí být specifikovány dvě ovládací funkce, zcela běžně jiné pro každý zvuk v závislosti na jeho trvání, intenzitě a frekvenci. Nicméně, jedná se o nejpřesnější techniku zvukové syntézy, u níž má hudebník neustálou kontrolu nad celým procesem [1][5].



Obrázek 6 – Aditivní syntéza [1]

2.2 Subtraktivní syntéza

Subtraktivní neboli rozdílová syntéza je protikladem k syntéze aditivní. Zatímco již zmíněná aditivní technika postupně skládá elementární zvuky do komplexního celku, přístup subtrakce spočívá v postupném odečítání jednotlivých frekvenčních složek signálu, respektive dochází k jejich potlačení nebo naopak zesílení.

Jedná se tedy o řízenou filtraci komplexního signálu. Zdrojem signálu mohou být signály s komplexní spektrem, například signály s obdélníkovým nebo trojúhelníkovým průběhem (liché spektrum), pilovým či pulzním průběhem (úplné spektrum), případně i šumové signály [3].

Jedním z nejatraktivnějších aspektů digitálního filtrování je fakt, že je analogický s fungováním mnoha akustických hudebních nástrojů (např. dechové nástroje) a také s lidským hlasem. Hrdlo, ústa a nos jsou filtračními dutinami a jejich rozměry se v čase mění. Jejich velká variabilita dělá z lidského hlasu nejbohatší a nejzajímavější hudební nástroj [1].

Existuje i modifikovaná verze subtraktivní syntézy, která spočívá v nahrazení obvyklého oscilátoru před filtry za vstupní audio signál. Použity jsou Bandpass filtry, kterým lze přiřadit libovolné části tónu. Pitch tracker ovládá středové frekvence těchto filtrů. Každé filtrované harmonické lze nastavit šířku pásma a hlasitost. Výhody této modifikace jsou vyšší hratelnost specifickou pro daný nástroj než běžné metody syntézy a snížení latence, protože zvuk je vnímám již při použití vstupního signálu. Velkou nevýhodou je ale nekompatibilita s běžnými rozhraními a stávajícími metodami syntézy [2].

2.3 Modulační syntéza

Nelinearita modulačního procesu, během které vznikají zcela nové harmonické složky úměrné součtům a rozdílům již přítomných frekvencí. Na tom je postavena modulační syntéza. Vyskytují se zde tři hlavní parametry, a to nosná frekvence, modulační frekvence a index modulace. Všechny tyto parametry mohou být časově proměnné. Několikanásobná modulace je taktéž možná, dokonce i modulace sebou samým, tj. zpětnovazební modulace [3].

2.3.1 Frekvenční modulace

Pomocí rozladění modulační frekvenci vůči nosné frekvenci v iracionálním poměru umožnuje plynulý přechod mezi harmonickými a neharmonickými strukturami. FM patří k nejpoužívanějším technikám, hodí se pro různé efekty a ruchy. V následujícím odstavci si blíže popíšeme její modifikovanou verzi [3].

Rozdíl oproti klasické FM spočívá v nahrazení modulačního oscilátoru za vstupní audio signál. Pro možnost nastavení intenzity nosného oscilátoru je signál sledovače obálky namapován na řízení nosné amplitudy. Silné vstupy bohužel mohou způsobovat nepřirozený jas výsledného zvuku, tomu se však dá zamezit pomocí mapování signálu sledovače obálky na ovládání indexu. Chybí ještě frekvenční poměr, který je udržován pomocí pitch trackeru na vstupním signálu [2].

2.3.2 Amplitudová modulace

Amplitudová modulace (AM) byla jednodušeji realizovatelná pomocí analogových elektronických prostředků, díky čemu se používala po mnohem delší dobu. Lze ji popsat následujícím vztahem:

$$y(n) = x(n)\left(1 + \alpha m(n)\right) \quad (1)$$

a to za předpokladu, že vrcholová amplituda m(n) je rovna 1. Koeficient alfa určuje hloubku modulace. Jeho hodnoty se pohybují mezi 0 pro vypnutí efektu a 1 pro maximální účinek modulace. Obvykle se používá se zvukovým signálem jakožto nosnou x(n) a nízkofrekvenčním oscilátorem (LFO) jako modulátorem m(n), tedy amplituda signálu se mění podle amplitudy LFO [4].

2.3.3 Multiplikativní modulace

V analogové rovině taktéž zvaná jako Ring modulation (RM). Jedná se o základní nelineární transformaci, ve které je vstupní audio signál násoben sinusoidou s danou nosní frekvencí. Korektní realizace na analogové úrovni je poměrně obtížná, ale digitální přístup tento problém řeší, neboť násobení signálů se tu stává triviálním procesem. První operand neboli vstupní signál, je označen jako x(n) a druhý, sinusoida s nosnou frekvencí, jako m(n):

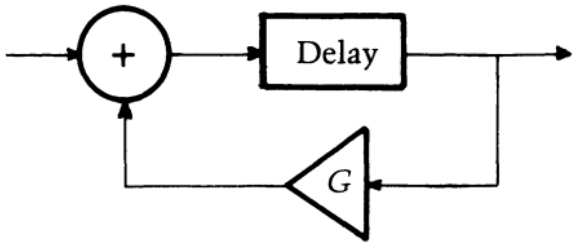
$$y(n) = x(n)m(n) \quad (2)$$

Podobně jako u AM vznikne při násobení dvou harmonických signálů součtová a rozdílová složka, ale v případě RM je původní složka potlačena.

2.4 Dozvuková syntéza

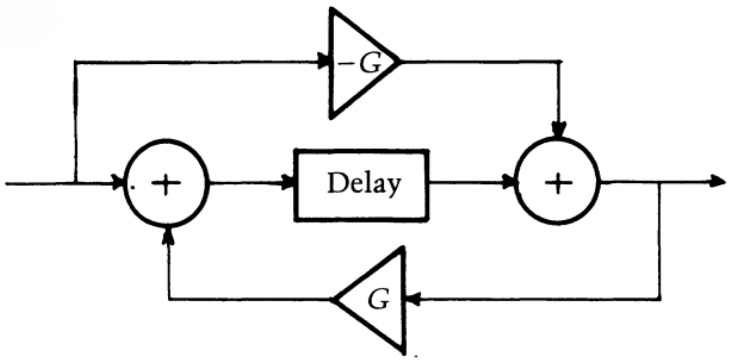
Hudebníky velice oblíbená syntéza, taktéž zvaná jako Reverberation. Digitálně simuluje akustické prostředí, respektive mnohonásobné odrazy zvuku a jejich postupné zpracování posluchačem. Jeden z možných přístupů je distribuce zvuku mezi větší množství reproduktorů a úpravou poměru mezi přímým zvukem a dozvukem. Většinou se ale využívá digitální přístup zpracování zvuku k dosažení požadovaného výsledku. Používají se dva základní filtry.

První z nich je Comb filtr. Signál je v něm zpožděn o určitý počet vzorků, zeslaben a přidán na vstup, čímž je dosaženo exponenciálně se snižující, opakované echo [1].



Obrázek 7 – Comb filtr [1]

Druhý se nazývá Allpass filtr neboli celopropusný filtr, a to z důvodu, že frekvenční odezva je zde plochá a dochází pouze k fázovému posunu. Vstupní signál je zeslaben a odečten od zpožděného signálu. Tento přístup pomáhá s ovládáním zpětnovazebního efektu a zachováním ozvěn [1].



Obrázek 8 – Allpass filtr [1]

Dozvukové efekty bývají konstruovány kombinací těchto filtrů. Standardně by nemělo docházet k opakování výrazně rozpoznatelných zvuků, neboť dozvukový výsledek by měl být složen z dostatečně rozptýleného zvuku [1].

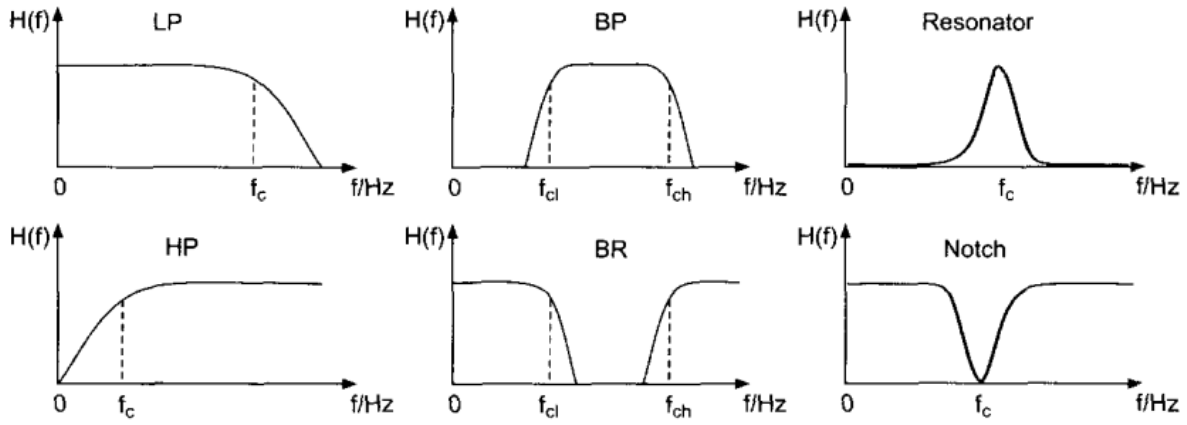
Doba zpoždění každého elementárního filtru musí být zvolena velmi opatrně a s patřičnou znalostí jejího dopadu na výsledný zvuk.

2.5 Filtry a hudební efekty

Tato kapitola se zaměřuje na základní filtry zpracování audio signálu, a poté blíže popisuje na fyzikálně-matematické úrovni audio efekty, jejichž implementace je hlavní náplní této práce. Tudíž cílem není popsat všechny možné existující efekty, ale pouze ty, které budou následně implementovány.

Již zmíněné základní filtry můžeme řadit podle této klasifikace. Její grafické znázornění je vyobrazeno na následující straně [4]:

- **Lowpass** (LP) filtry propouští pouze **nižší** frekvence až po hraniční hodnotu fc.
- **Highpass** (HP) filtry naopak propouští pouze frekvence **vyšší,** než je daná hraniční hodnota fc.
- **Bandpass** (BP) filtry jsou kombinací výše zmíněných filtrů, tudíž propouští pouze určitý interval frekvencí omezený z obou stran.
- **Bandreject** (BR) nebo také Band-stop filtry jsou přesnou negací BP filtrů, tedy propouští všechny frekvence kromě specifikovaného intervalu.
- **Resonator** filtr propouští frekvence v úzké šířce pásma kolem mezní frekvence fc.
- **Notch** je opakem filtru typu Resonator. Frekvence v úzké šířce okolo mezní frekvence fc blokuje a propouští všechny ostatní.
- **Allpass** filtry, jak název napovídá, propouští veškeré frekvence, ale mění fázi vstupního signálu.



Obrázek 9 – Základní filtry [4]

2.5.1 Nelineární efekty

Dávají rockové a metalové hudbě jejich charakteristické rysy. Mají zásadní dopad na vývoj a formulaci těchto hudebních žánrů, zejména z historického pohledu. Obecně v nich dochází k silnému harmonickému zkreslení a vzniká ten rockově známý, elektrický až špinavý zvuk.

2.5.1.1 Overdrive

Na rozdíl od svých bratrů Distortion a Fuzz poskytuje Overdrive jemnější a hladší zkreslení. Nejširší využití nachází v klasickém rock 'n' rollu. Jeden z možných popisů tohoto efektu je pomocí troj-funkce, která plynule přechází z lineární části do části nelineární, a poté skokově na maximální hodnotu [4].

$$f(x) = \begin{cases} 2x & x < \frac{1}{3} \\ \frac{3-(2-3x)^2}{3} & \frac{1}{3} \leq x < \frac{2}{3} \\ 1 & \frac{2}{3} \leq x < 1 \end{cases} \tag{3}$$

Definiční obor funkce jsou hodnoty vstupního signálu transformované na interval od 0 do 1. Proslulých realizací tohoto efektu existuje nespočet, například Fulltone OCD V2, Analog Man King of Tone, Boss JB-2 Angry Driver a mnoho dalších.

2.5.1.2 Distortion

Jedná se o extrémní případ Overdrive efektu, ve kterém dochází k úplného odstranění vysokých amplitud signálu. Základní algoritmus Distortionu se tedy skládá z dvou kroků. Nejdříve se znásobí vstupní signál a poté dojde k ořezání. pomocí dané funkce, například takto, kde x představuje jednotlivé hodnoty vzorkovaného signálu [4]:

$$f(x) = \frac{x}{|x|}\left(1 - e^{x^2/|x|}\right) \tag{4}$$

Někdy se do zkresleného signálu přidá i signál původní, neboť při velkých zkresleních může docházet ke špatné rozpoznatelnosti původních frekvencí. Mezi velice populární Distortion efekty patří například Fulltone Distortion Pro, Boss DS-1 či Dunlop MXR M116 Fullbore Metal.

2.5.1.3 Fuzz

Myšlenka Fuzz efektu je velmi podobná výše zmíněnému Distortion efektu. Když hodnota signálu překročí danou mezní frekvenci, tento signál se ořízne a je nahrazen zcela novou, často maximální nebo i částečně náhodnou hodnotou. Tím je docílen hudebně špinavý, nepřesný a částečně nepředvídatelný zvuk. Nejčastěji se používá na single-note kytarová sóla a hlavní, přímočařejší melodie. Zejména Jimi Hendrix si tento typ efektu zamiloval. Pro ukázku zmiňme Z Vex Fuzz Factory a Dunlop Hendrix Bog Fuzz Mini [6].

2.5.2 Dozvukové efekty

Zvuk se šíří po celém prostoru do všech směrů. Na jeho intenzitě v jednotlivých směrech šíření závisí zejména na typu zdroje daného zvuku. Nicméně v každém případě vznikají odrazy zvuku od různých "odrazových" stěn. Ve venkovním prostředí to mohou být hory či budovy. Z technického hlediska je situace mnohem zajímavější v vnitřních prostorách, od obyčejných místností až po koncertní či divadelní sály.

Zvuková vlna odražena od dostatečně vzdálené stěny k nám jakožto k posluchači dorazí později než přímá vlna ze zdroje zvuku, tedy odraženou vlnu vnímáme jako opožděnou. Pokud se ale stěna nenachází v dostatečné vzdálenosti, vnímáme její odrazy jako změnu barvy zvuku.

K realizaci těchto jevů používáme již zmíněné FIR/IIR Comb filtr a Allpass filtr. Primární parametry, se kterými navržené algoritmy pracují, jsou doba zpoždění a jeho amplituda čili intenzita odraženého signálu [4].

2.5.2.1 Delay a Echo

Delay je **jednorázová** replikace původního zvuku, která následuje po určité době za tímto původním zvukem, obvykle v desítkách až stovkách milisekund.

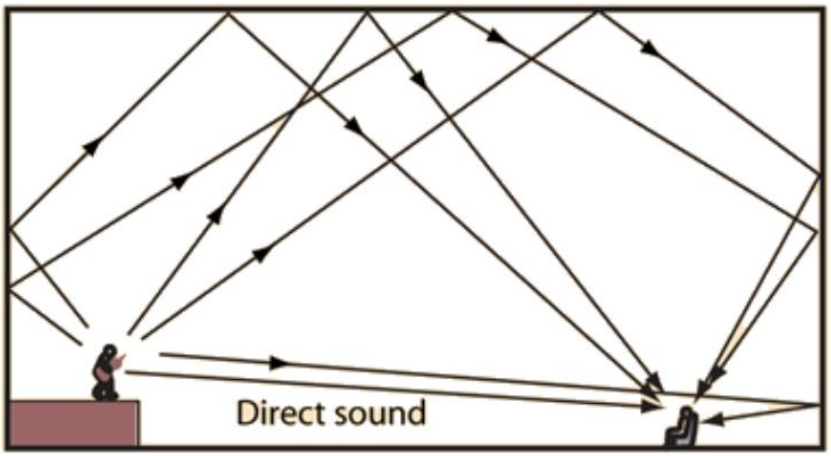
Echo efekt simuluje chování ozvěny. Obsahuje **několik instancí** původního zvuku zpožděného v násobcích časového rámce prvního zpoždění, prvního Delay efektu, přičemž každá další instance má nižší intenzitu. Jejich intenzita neboli hlasitost, takto postupně klesá až na nulu [8].

2.5.2.2 Reverb

Funguje podobně jako Echo, ale s dvěma zásadními rozdíly. Vzdálenost mezi jednotlivými odrazy je exponenciálně menší a existuje exponenciálně víc instancí těchto odrazů. Bývají to stovky až tisíce (ve skutečnosti až nekonečně mnoho) krátkých opakování, díky kterým vnímáme výsledný zvuk jako mnohem obsáhlejší a mohutnější.

Reverb jednoznačně patří k složitějším efektům. Implementovat ho v dostatečně uspokojivé kvalitě není zcela triviální, ale pokud se to podaří, tak při správném použití dokáže výsledný zvuk působit přímo kouzelně.

Snaží se simulovat následující situaci, kde k posluchači kromě přímého zvuku dorazí i nespočet odrazů, a to s různou intenzitou a po různě dlouhé době.



Obrázek 10 – Reverb princip [6]

Zjednodušeně ho můžeme popsat pomocí čtyř aspektů [9].

- Počáteční odrazy na intervalu od 1 do 30 milisekund
- Tělo odrazy od 30 milisekund až po nastavený parametr
- Úpadek když amplituda jednotlivých ozvěn klesne pod přibližně 60 dB

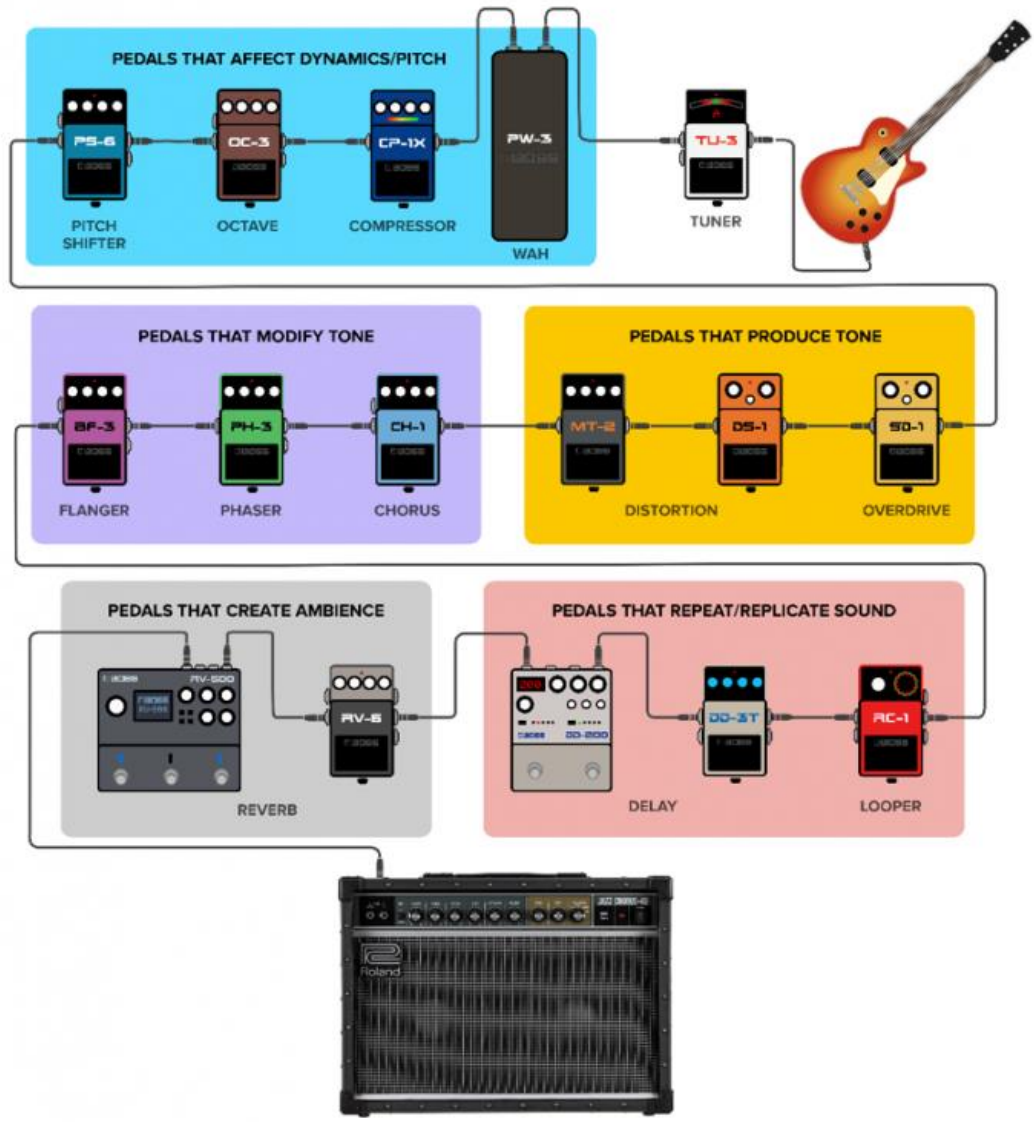
Čtvrtý kousek skládačky je velikost, tvar a materiál virtuální místnosti, ve které tato audio produkce probíhá. Materiál ovlivňuje frekvenční odezvu, velikost prostor zase délku ozvěny a tvar celkově průběh jednotlivých ozvěn v daných směrech.

2.6 Sériové zapojení efektů

Efektové pedály lze dělit do několika skupin podle jejich funkce, či přesněji způsobem, jakým upravují vstupní signál. Mějme tedy následujících pět rodin efektů:

1. Ovlivňující dynamiku a výšku tónu:
   Wah, Compressor, Octave, Pitch Shifter
2. Vytvářející tón:
   Overdrive, Distortion, Fuzz
3. Upravující tón:
   Chorus, Phaser, Flanger
4. Opožďující či opakující tón:
   Looper, Delay
5. Vytvářející vjem prostředí, místnosti či budovy:
   Reverb, Echo

Dodržení pořadí těchto rodin pedálových efektů je silně doporučeno, a to obzvlášť v profesionální sféře. V té je to prakticky pravidlo. Ohledně pořadí konkrétních efektů v dané rodině je situace o něco méně přísnější, avšak i tady se najdou kombinace, které spolu zkrátka nefungují. Například zapojit Delay před Looper. Nicméně netřeba se bát experimentovat a zkoušet různá zapojení. Výše popsané a níže graficky znázorněné doporučené pořadí není zdaleka jediné. Přece jenom se jedná o subjektivně vnímaný sluchový vjem. Preference posluchačů i hudebníků se člověk od člověka liší.

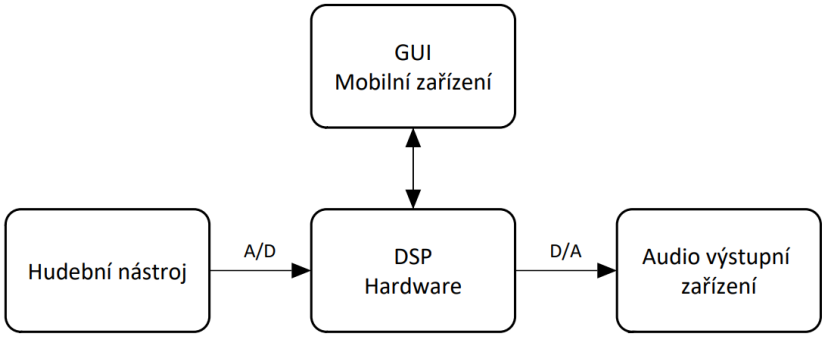


Obrázek 11 – Doporučené pořadí efektů [12]

2.7 Návrh systému

Průzkum trhu ukázal, že podobné projekty už existují. Často jsou realizovány pomocí Arduina či Raspberry Pi, některé i s využitím méně známého hardwaru. Tyto systémy však většinou poskytují pouze fyzické uživatelského rozhraní s omezenými možnostmi a prostorem pro budoucí vývoj.

Dostupná jsou i jiná řešení, pro uživatele zcela jistě výrazně jednodušší na použití. Mobilní aplikace. Stáhnout a vyzkoušet se dají téměř okamžitě. Skvělé řešení této problematiky, no bohužel i to má své nedokonalosti. K největším z nich patří komplikace s nedostatečným výkonem pro potřeby některých efektů. Typicky to bývá neduh implementací Reverbu. Ještě



Obrázek 12 – Blokové schéma systému

kritičtější však dokáže být situace ohledně latence, kdy při použití v některých kombinacích Zařízení-OS-Aplikace je časová prodleva příliš vysoká a kytaristovi to znemožňuje hraní v reálném čase.

Z těchto důvodů jsem se rozhodl pro kombinované řešení, které kloubí externí, dostatečně výkonný hardware a mobilní aplikaci, která slouží pouze jako uživatelské rozhraní. Blokové schéma tohoto systému je znázorněno na obrázku č. 12. Pro upřesnění přikládám popis jednotlivých komponent.

Hudební nástroj	Typicky elektrická kytara. S využitím patřičných snímačů či mikrofonů je možné použít snad jakýkoliv hudební nástroj.
GUI – Mobilní zařízení	Aplikace jakožto uživatelské rozhraní. Použitelná z mobilního telefonu či tabletu.
DSP – Hardware	Jakýkoliv hardware, který poskytuje dostatečné prostředky a výkon pro potřeby DSP. Dále musí být ovladatelný i bezdrátovým přístupem, tudíž se nabízí Wi-Fi, Bluetooth či jiná technologie.
Audio výstupní zařízení	Typicky kytarové kombo, ale mohou být použity libovolné reproduktory či sluchátka.

3 IMPLEMENTACE

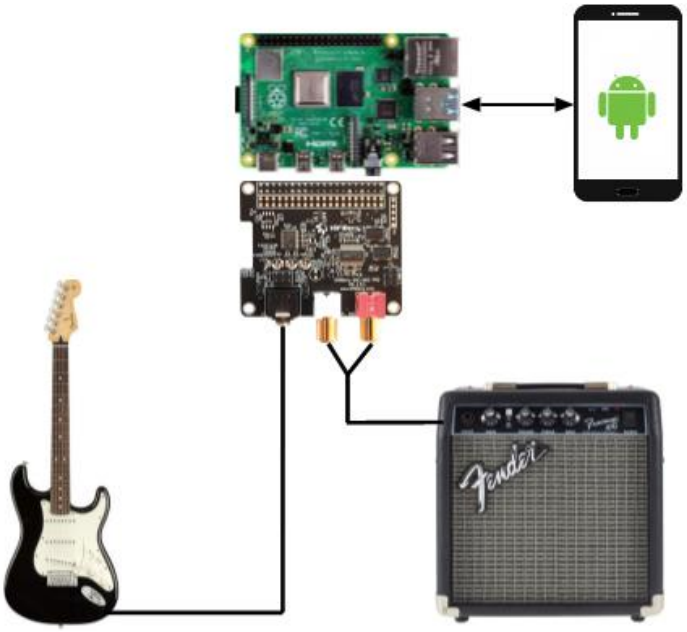
3.1 Představení sytému

Navržený systém lze rozdělit do dvou částí. Na Raspberry Pi běží jádro celého projektu, které je ovládáno pomocí GUI realizovaného Android mobilní aplikací.

Backend se stará o zpracování a přeposílání vstupního signálu na výstupní zařízení. Vstupní signál představují tóny z připojené elektrické kytary, ale může být použit i jiný hudební nástroj. Samotné zpracování signálu probíhá za pomocí zvukové karty HiFiBerry DAC+ ADC Pro a následným úpravám v C++ programu, který simuluje vybrané efekty. Výstupním zařízením se rozumí sluchátka, reproduktory nebo kytarové kombo.

Frontend je realizován pomocí mobilní aplikace pro Android. Obstarává výběr a připojení k danému Raspberry, rozhraní pro nastavení efektů na virtuální pedalboardě a poskytuje nápovědu, která je podstatná zejména pro první použití.

Schéma zapojení může vypadat různě, a to zejména v závislosti na použitých audio vstupních a výstupních zařízení. I zvuková karta je zaměnitelná, tudíž nemusí být použita právě tato. Mnou navržené a používané schéma zapojení přikládám zde (Obrázek 12 – Schéma zapojení).

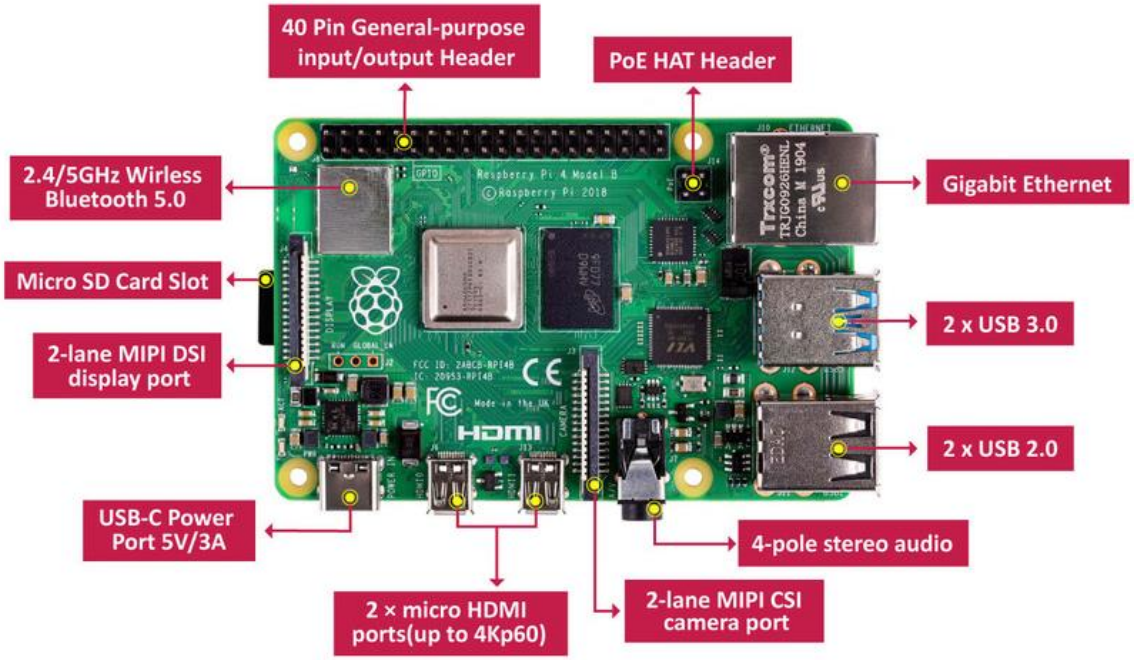


Obrázek 13 – Schéma zapojení

3.2 Platforma Raspberry PI

V mém provedení jsem použil Raspberry Pi 4 Model B s 4 GB operační paměti a již zmíněnou HiFiBerry DAC+ ADC Pro zvukovou kartu, která je přímo kompatibilní s Raspberry Pi. Pro potřeby projektu je to více než dostačující.

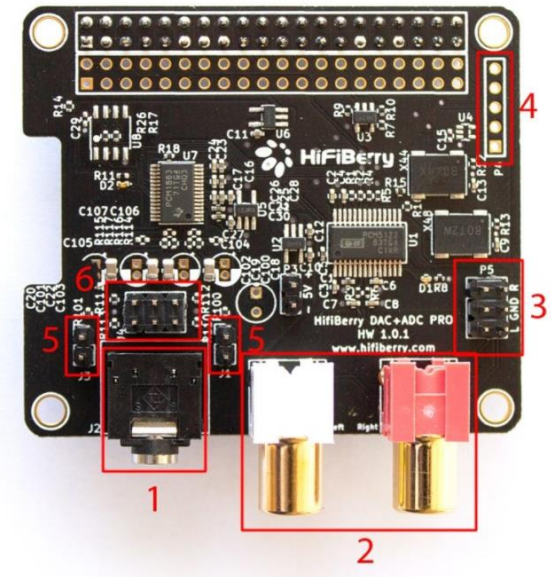
Raspberry Pi patří k nejrozšířenějším jednodeskovým počítačům na světě, jedná se o téměř plnohodnotnou náhradu počítače. Oproti počítačům je však výrazně menší, levnější a flexibilnější, díky čemuž se dá použít v mnoha projektem podobného, ale i zcela jiného charakteru. Samotné Raspberry Pi by však pro mnou navržený systém nestačilo. Další klíčová komponenta je zvuková karta, neboť průměrný počítač či čisté Raspberry Pi nemají adekvátní I/O konektory, DAC ani ADC. Tedy nejsou uzpůsobeny k zpracování audio signálu na dostatečné úrovni pro potřeby této práce.



Obrázek 14 – Raspberry Pi 4 Model B [10]

Výhoda této zvukové karty je zejména v přítomnosti analogového vstupu i výstupu a vysoké vzorkovací frekvenci dosahující až 192 kHz. Popis jednotlivých konektorů vůči obrázku [10]:

1. Analogový vstup, Jack
2. Analogový výstup, RCA
3. Analogový výstup, piny
4. I2S sběrnice
5. Propojka pro předpětí mikrofonu
6. Analogový vstup, piny



Obrázek 15 – HiFiBerry DAC+ ADC Pro

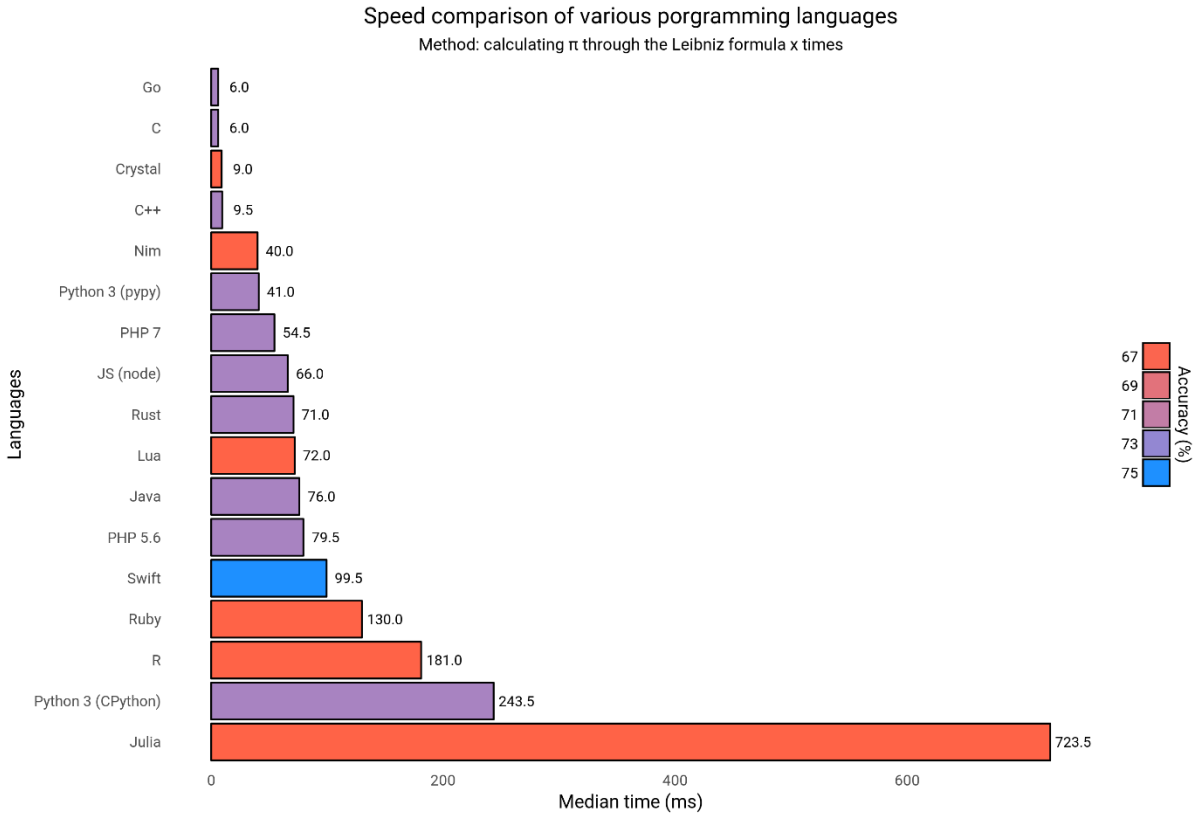
3.3 Volba jazyka pro DSP

Jednodeskový počítač Raspberry Pi sice defaultně podporuje pouze Python 2/3, C/C++ a Scratch, ale s trochou snahy na něm lze zprovoznit téměř jakýkoliv programovací jazyk. Dříve se DSP realizovalo hlavně pomocí Assembly, VHDL, BASIC či jazyku C. Nyní je však v kurzu primárně C++ a Matlab, popřípadě Python.

Za zmínku určitě stojí i funkcionální jazyk Faust (Functional Audio Stream), který nabízí jednodušší a programátorsky přívětivější variantu za klasické C++. Už jeho název správně naznačuje, že se jedná o jazyk přímo vyvinutý pro syntézu a zpracování audio signálu.

Mou volbou se nakonec stalo již několikrát zmíněné C++, a to primárně ze dvou důvodů. Zaprvé je to v této problematice velice často používaný jazyk, jelikož dokáže být rychlý a efektivní, což jsou pro zpracování audio signálu v reálném čase klíčové vlastnosti. Druhý pro mě důležitý faktor byl ten, že v

rámci předmětu Multimédia 1 a Semestrální projekt jsem si zkusil práci s knihovnou IIM-AVLib, která poskytuje API pro práci s audio a video obsahem. Knihovnu jsem si oblíbil, a navíc jsem se přesvědčil, že její funkcionalita je pro potřeby této práce dostačující. Víc si o ni povíme v další kapitole.



Obrázek 16 – Přehled programovacích jazyků podle výpočetní rychlosti [14]

3.4 Knihovna IIM-AVLib

Tato knihovna pochází z dílny Institutu Intermedií. Jak už bylo výše stručně řečeno, poskytuje API na zpracování audio a video obsahu. Vzhledem k tématu této práce se zaměříme pouze na audio část.

Velkým přínosem je zejména díky poskytnutí většiny základních potřeb při práci s audio signálem. Nabízí předpřipravené řešení problémů, kterým se programátor v těchto vodách nevyhne. Uveďme několik základních.

- výběr I/O zařízení
- načtení/uložení dat z/do wav souboru
- zpracování a přesměrování audio signálu v reálném čase
    - defaultní velikost bloku dat je 512 vzorků
    - při bitové hloubce 16 bitů
- nastavitelná vzorkovací frekvence
    - defaultně 44,1 kHz, Playthrough režim používá 192 kHz
- podpora stereo zvuku

Nejdříve jsem zamýšlel knihovnu pouze použít, no díky jejímu open-source přístupu jsem měl možnost svoji práci postavit přímo na ni a tím ji rozšířit o několik efektů, filtrů a pár menších vylepšení. V následujících řádcích si přiblížíme již zmíněná předpřipravená řešení. Hlavní koncept tvoří objekty typu **filtr** a **sink** [15].

Filtr může být buď to zdrojový, který generuje audio vzorky, nebo editační, který na vstupu přijímá navzorkovaný signál a nějakým předepsaným způsobem ho upravuje.

Sink představuje speciální druh filtru, jehož účelem je dané vzorky přehrát na výstupním zařízení, uložit do souboru či provést jinou operaci podobného charakteru.

Ukázkový příklad si můžete prohlédnout na následujícím obrázku, kde zpracování probíhá formou filtračního řetězce, který se obvykle skládá ze zdrojového filtru, libovolného počtu editačních filtrů a sinku na konci.

```
using namespace iimavlib;
auto sink = filter_chain<WaveSource>("file.wav")
        .add<SimpleEchoFilter>(0.2)
        .add<PlatformSink>(device_id)
        .sink();
```

Obrázek 17 – Příklad použití filtrů v řetězci [15]

Zmíněný příklad vytváří řetězec filtrů, který načte vstupní soubor *file.wav*, přidá *echo* se zpožděním 200 milisekund a poté přidá *sink* pro přehrání takto upraveného vstupu prostřednictvím audio výstupního zařízení, které je specifikováno svým *device_id* [15].

3.5 Rozšíření knihovny IIM-AVLib

V této kapitole je blíže popsán backend navrženého systému, který lze považovat i za rozšíření knihovny IIM-AVLib. Jedná se zejména o modifikaci spustitelných souborů *playthrough* a *playback* a tvorbu nových efektů a pomocných filtrů. Pojďme tedy popořadě.

3.5.1 Playthrough & Playback

Oba tyto režimy již v základní verzi knihovny poskytovaly většinu potřebné funkcionality. Playthrough je uzpůsobený pro použití v již představeném systému, tudíž při spuštění bere jako své parametry názvy jednotlivých efektů a hodnoty jejich parametrů. Následně v cyklu postupně iteruje efekty a přidává je v uživateli zvoleném pořadí do řetězce zpracování. Podstatnou částí této operace je i škálování, neboť v GUI se pro jednoduchost pohybují hodnoty všech parametrů pouze na intervalu 0-100.

Playback obdržel výrazně méně úprav, jelikož v primární verzi aplikace není použit (viz kapitola 4.8.). Slouží pouze pro účely sekundární, respektive konzolové verze aplikace. Byl tedy obohacen pouze o nové efekty.

3.5.2 Volume

Nejjednodušší efekt, co v pravém slova smyslu ani kytarový efekt není. Triviálně násobí každou vzorkovanou hodnotu vstupního signálu, čímž ji procentuálně zeslabuje nebo zesiluje. Při použití je třeba dávat pozor na možné přebuzení signálu. Nachází využití i při škálování efektů Distortion a Fuzz, kdy při síle zkreslení má stoupat i celková hlasitost.

```
template<typename T>
void change_volume(T dest, size_t samples, double volume_value)
{
    for (size_t sample = 0; sample < samples; ++sample) {
        (*dest).setLeft((*dest).getLeft() * (volume_value / 100));
        (*dest).setRight((*dest).getRight() * (volume_value / 100));
        dest++;
    }
}
```

Obrázek 18 – Volume implementace

3.5.3 Overdrive

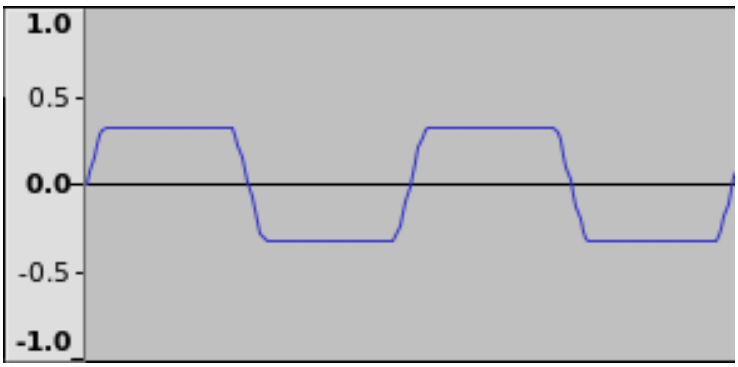
Implementace proběhla pomocí již zmíněné troj-funkce, kterou lze nalézt v podkapitole Nelineární efekty. Bohužel i přes zdánlivou jednoduchost efektu se jedná o ten nejproblematičtější, neboť ve výsledném zvuku se místy vyskytují zcela jasně slyšitelné artefakty, které výrazně kazí hudební dojem.

Chování algoritmu si lze prohlédnout na následujících obrázcích, které znázorňují časový průběh a spektrum sinusovky po aplikaci efektu.

Všechny nelineární efekty byly testovány pomocí sinusovku s těmito parametry:

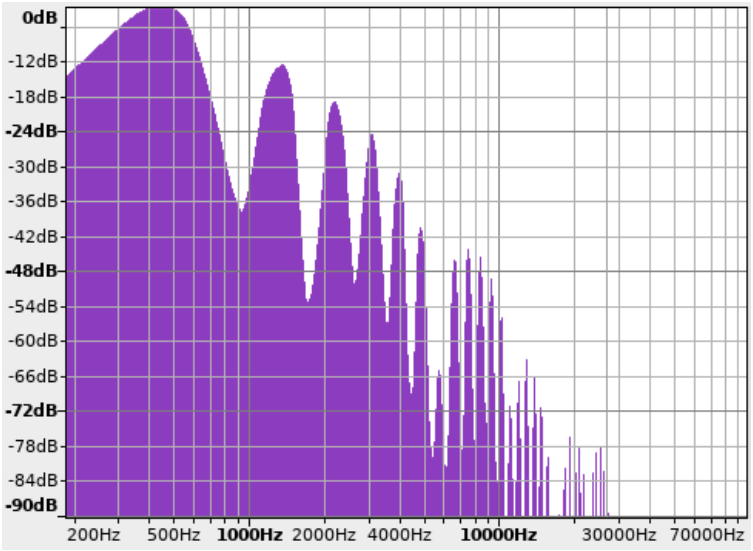
- frekvence = 440 Hz
- amplituda = 0.6

Způsob implementace umožňuje uživateli zadat maximální hodnotu neboli threshold algoritmu jako vstupní parametr. Tudíž toto provedení i zesiluje či zeslabuje celkovou hlasitost. Ukázkový časový průběh má takto nastavenou threshold na přibližně 0.3 z číselného rozsahu.



Obrázek 19 – Overdrive časový průběh

Neduhy u ostatních efektů jsou poměrně malé nebo zcela neslyšitelné, no Overdrive naráží nejvíce. Jeho vylepšení je jedním z hlavnějších bodů pro možný, budoucí vývoj. Následuje spektrum časového průběhu upravené sinusovky a hlavní metoda implementace efektu.



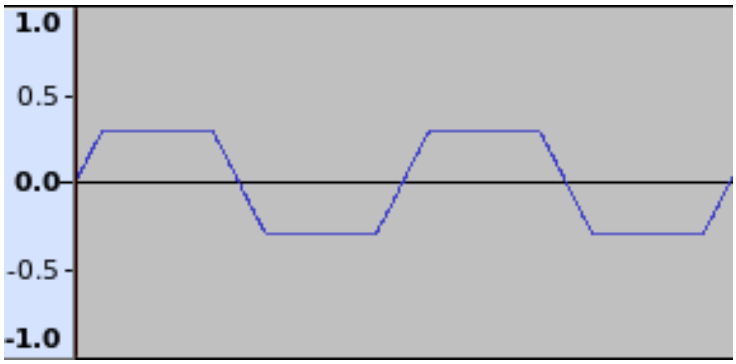
Obrázek 20 – Overdrive spektrum časového průběhu

```
void Overdrive::add_overdrive(std::vector<audio_sample_t>::iterator dest, size_t samples)
{
    for (size_t sample = 0; sample < samples; ++sample)
    {
        // first part of function
        if (*dest > -part1_ && *dest < part1_)
        {
            (*dest).setLeft((*dest).getLeft() * 2);
            (*dest).setRight((*dest).getRight() * 2);
        }
        // second part of function
        else if (*dest > -part2_ && *dest < part2_)
        {
            if (*dest > 0) {
                (*dest).setLeft(countPart2Value((*dest).getLeft(), max_value_));
                (*dest).setRight(countPart2Value((*dest).getRight(), max_value_));
            }
            else {
                (*dest).setLeft(countPart2NegativeValue((*dest).getLeft(), max_value_));
                (*dest).setRight(countPart2NegativeValue((*dest).getRight(), max_value_));
            }
        }
        // third part of function
        else if (*dest > part2_)
        {
            (*dest).setLeft(max_value_);
            (*dest).setRight(max_value_);
        }
        // third part of function for negative values
        else
        {
            (*dest).setLeft(-max_value_);
            (*dest).setRight(-max_value_);
        }

        dest++;
    }
}
```

Obrázek 21 – Overdrive implementace

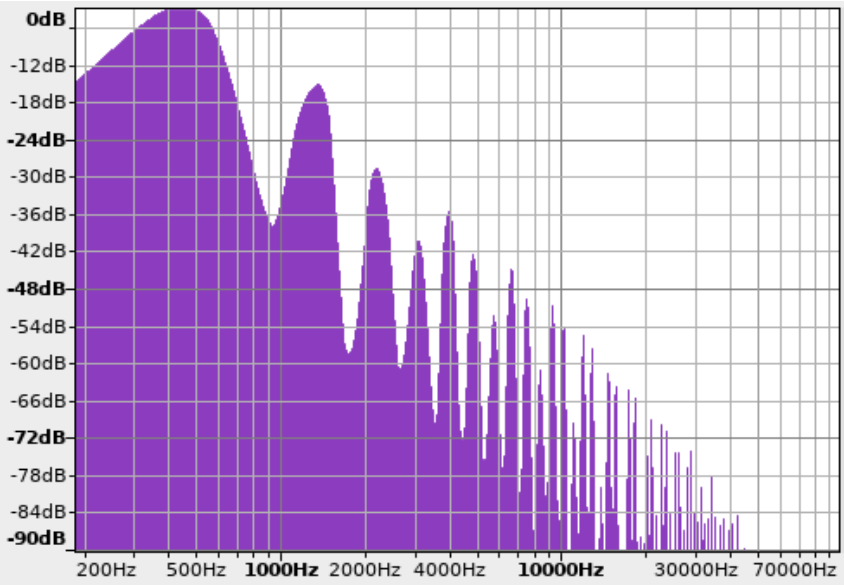
3.5.4 Distortion

U tohoto efektu jsem se rozhodl nenásledovat pro něj určený vzorec uvedený v podkapitole Nelineární efekty. Distortion lze zjednodušit na situaci, ve které pokud signál překročí určitou hranici neboli threshold, tak je oříznut a nahrazen danou mezní hodnotou.



Obrázek 22 – Distortion časový průběh

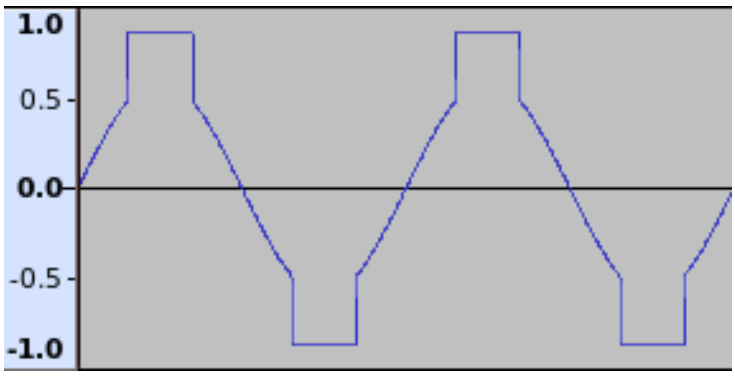
Časový průběh sinusovky po aplikaci efektu na první pohled vypadá velice podobně jako při Overdrivu, no lze si povšimnout, že v tomto případě části signálu mezi kladnou a zápornou hranicí rostou či klesají pomaleji než u právě zmíněného Overdrivu. Následuje spektrum oříznuté sinusovky a hlavní metoda efektu, která je stejně jako u ostatních efektů postupně aplikována na všechny vzorky signálu.



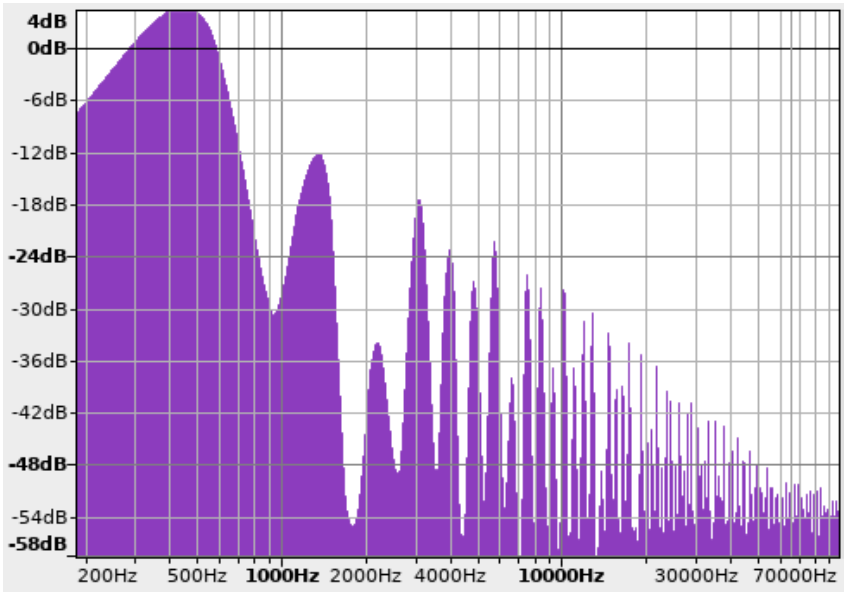
Obrázek 23 – Distortion spektrum časového průběhu

3.5.5 Fuzz

Základním kamenem již existující Distortion efekt. Rozdíl je pouze v nově nastavené amplitudě při překročení threshold. Tato nová amplituda je stejně jako threshold parametrem efektu. Pro zpřístupnění efektu uživateli je důležité rozumné nastavení škálování, respektive číselných intervalů, na které se překládají uživatelem zadané hodnoty. Fuzz nabízí silně nelineární zkreslení, tudíž se na něj musí opatrně. Snadno se dá sklouznout ke kombinacím parametrů, při kterých přestane znít dobře. Ukázkový časový průběh má nastavenou threshold na přibližně 0.5 a novou amplitudu na 0.9 z číselného rozsahu. Na další straně následuje hlavní metoda tohoto efektu, která se od Distortion metody liší pouze v hodnotě, která se v případě překročení threshold použije.



Obrázek 24 – Fuzz časový průběh



Obrázek 25 – Fuzz spektrum časového průběhu

```
void Fuzz::add_fuzz(std::vector<audio_sample_t>::iterator dest, size_t samples)
{
    for (size_t sample = 0; sample < samples; ++sample)
    {
        if (*dest > threshold_)
        {
            (*dest).setLeft(maxValue_);
            (*dest).setRight(maxValue_);
        }
        else if (*dest < -threshold_)
        {
            (*dest).setLeft(-maxValue_);
            (*dest).setRight(-maxValue_);
        }
        dest++;
    }
}
```

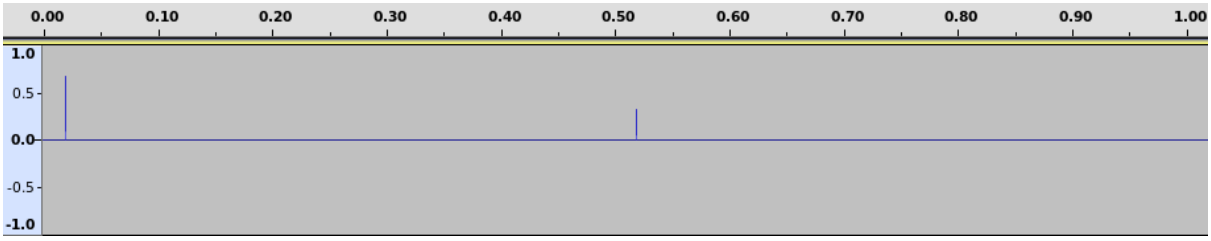
Obrázek 26 – Fuzz implementace

3.5.6 Delay

Vstupní signál je průběžně ukládán do pomocného zásobníku, ze kterého je postupně o daný časový úsek přidáván na výstup. Výstupem algoritmu je tedy vždy součet aktuálního vstupu se vstupem předešlým.

V následující kapitole je popsáno, že knihovna IIM-AVLib již obsahuje implementaci efektu Echo, tudíž místo tvorby zcela nového algoritmu jsem se v tomto případě rozhodně vzít tento již existující a pouze ho upravit pro potřeby Delay efektu. Úpravy to byly poměrně malé, neboť stačilo přestat dávat na výstup zpožděný signál mnohonásobně, ale pouze jednou. Algoritmus má dva parametry, a to poměr hlasitosti mezi vstupním a opožděním signálem a časový údaj zpoždění, které se pohybuje na intervalu 0-1 vteřina.

Parametry Delay odpovídající zobrazenému časovému průběhu jsou zpoždění 500 milisekund a level o hodnotě 33, což představuje necelou jednu třetinu z amplitudy vstupního signálu.



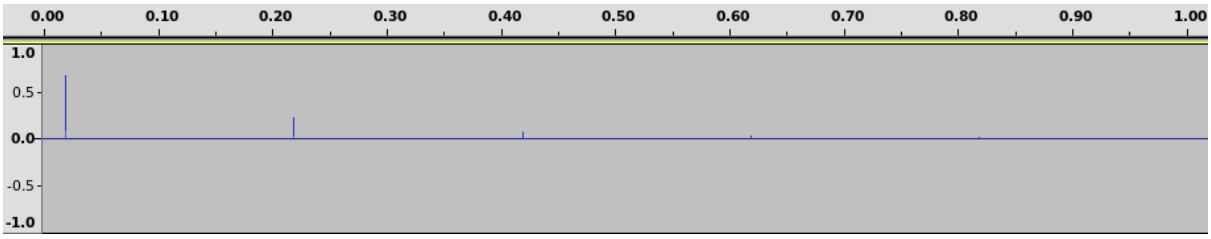
Obrázek 27 – Delay časový průběh Diracova impulsu

3.5.7 Echo

Stejně jako Delay přidává na výstup zpožděnou hodnotu aktuálního signálu, ale navíc k ní přičítá i předchozí signál. Zpožděný signál je tak přidán mnohonásobně s postupně klesající intenzitou, a tím simuluje doznívající ozvěnu.

Nejčastěji se dozvukové efekty implementují použitím Combo nebo All-pass filtrů. Avšak knihovna IIM-AVLib již Echo efekt obsahuje a výsledky této implementace jsem pro mé potřeby dostačující, tudíž jsem se rozhodl ho neměnit. Algoritmus dělá přesně to, co by z definice i měl, jak si můžete prohlédnout na časovém průběhu Diracova impulsu po aplikování efektu. Neoptimální je pouze jeho provedení v kódu, neboť je psaný bez využití modernějších kontejnerů, čímž připomíná spíše staré *Céčko* než

dospělé *C++*. Ohledně parametrů je situace podobná jako u efektu Delay. Parametry jsou dva. Poměr hlasitosti a doba zpoždění mezi jednotlivými impulsy.



Obrázek 28 – Echo časový průběh Diracova impulsu

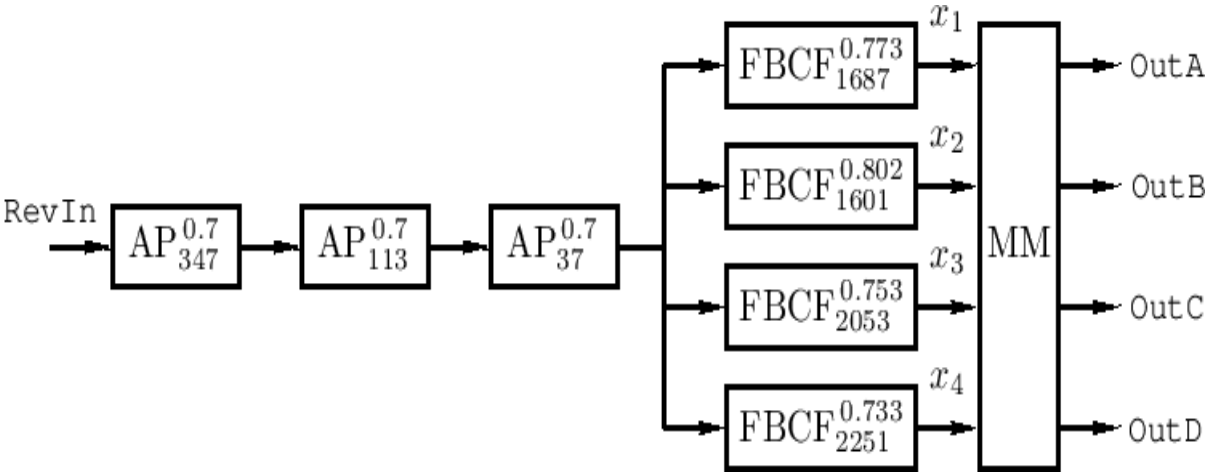
Parametry Echa odpovídající zobrazenému časovému průběhu jsou zpoždění 200 milisekund a level o hodnotě 33.

3.5.8 Reverb

Simuluje zvuk, který dopadá k posluchači jako odrazy od okolních stěn a objektů v místnosti či obecně prostředí, ve kterém se daný zdroj zvuk a posluchač nacházejí.

Při prvním pokusu o implementaci Reverbu jsem použil zjednodušený, a ne zcela korektní přístup. Jednalo se o trojnásobné použitý Echo efektu. Při správných kombinacích řídících parametrů se výsledný zvuk přibližoval v cílenému Reverbu, nicméně v koncové části nebyl dozvuk na mé požadavky dostatečně plynulý a splývající. Tyto nedokonalosti jsou pochopitelné, neboť jednak s postupně klesající intenzitou odrazů byla vzdálenost mezi jednotlivými odrazy stále stejná, přičemž by měla klesat, a za druhé obecně nebyla dodržena struktura Reverbu.

Druhému pokusu předcházel už poněkud obsáhlejší průzkum možných technik a implementací, s cílem dodržet strukturu Reverbu nebo se jí alespoň přiblížit. Nápomocný mi byl zejména internetový zdroj Spin Semiconductor, který detailně rozebírá řadu efektů a jejich možnou implementaci [16]. Klíčový byl však až pohled na Schroederovu implementaci reverbu s využitím sériového zapojení All-pass filtrů s blokem paralelně zapojených Combo filtrů.



Obrázek 29 – Schroederův algoritmus reverbu [17]

Vrchní hodnota každého filtru ve výše znázorněném schématu představuje zesílení na daném filtru, zatímco hodnota spodní udává délku zpoždění v počtu samplů, nikoliv v milisekundách. Vysvětlení zkratek filtrů následuje zde:

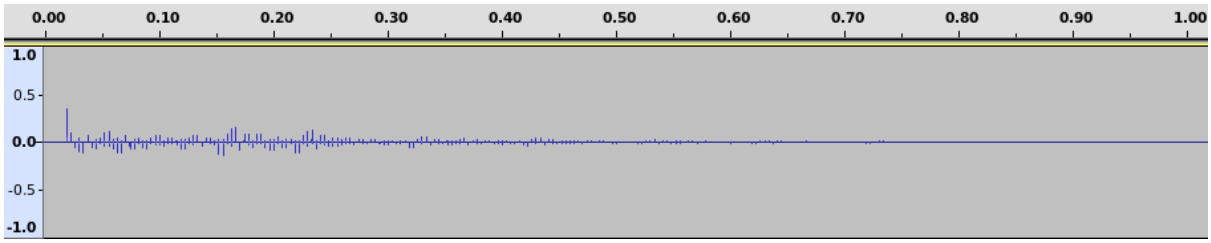
AP	Allpass filtr
FBCF	Feedback Combo filtr
MM	Mixážní matice (z důvodu multikanálového zvuku)

Hodnoty parametrů uvedené ve schématu představují pouze doporučení, a to už jen z důvodu výběru použité vzorkovací frekvence. I samotné schéma lze měnit. Obecně jsem čím dál tím víc přesvědčen, že ohledně digitálního zpracování zvuku, obzvlášť co se efektů týká, jsou k dispozici spíše doporučení než striktně daná pravidla. Vnímání zvuku jakožto hudby je silně subjektivní a do toho se odvíjí i subjektivní preference možných řešení.

Nyní už k mému druhému pokusu. Vycházel jsem ze Schroederova schématu. Už na začátku jsem věděl, že mixážní matici potřebovat nebudu, neboť si vystačím s jedno či dvoukanálovým zvukem, což knihovna IIM-AVLib podporuje. Dokonce jsem ve finálním algoritmu nepoužil ani Combo filtry, neboť už výsledek samotných Allpass filtrů byl dostatečně uspokojivý a plnil svoji roli.

Finální implementace je tedy postavena na čtveřici Allpass filtrů zapojených do série. Uživatel má k dispozici nastavení dvou parametrů, konkrétně trvání dozvuku a poměr hlasitosti mezi přímým zvukem a simulovaným dozvukem. Poměr hlasitosti (Wet parametr) je triviální, no s dobou dozvuku je situace trochu složitější.

Jak si lze všimnout ze Schroederova reverbu, každý další Allpass filtr opožďuje vzorky o dvě třetiny kratší dobu než Allpass filtr před ním. Tento fakt opět není striktní pravidlo, ale spíš jen doporučení. Rozhodl jsem se ho ale v mé implementaci dodržet. Uživatel tedy nastavuje hodnotu prvního filtru, ze které se následně dopočítají hodnoty pro ostatní filtry.



Obrázek 30 – Reverb časový průběh Diracova impulsu

Parametry znázorněného časového průběhu jsou:

Zpoždění prvního Allpass filtru	200 milisekund
Wet (poměr hlasitosti)	0.6

Wet parametr je implementovaný jako poměr, tudíž pokud je jeho hodnota například 0.6, tak intenzita přímého neboli dry zvuku je 0.4 (viz Reverb časový průběh Diracova impulsu). Hlavní metodu implementace si lze prohlédnout na následujícím obrázku, kde *sample_x* představuje vzorek přímého zvuku.

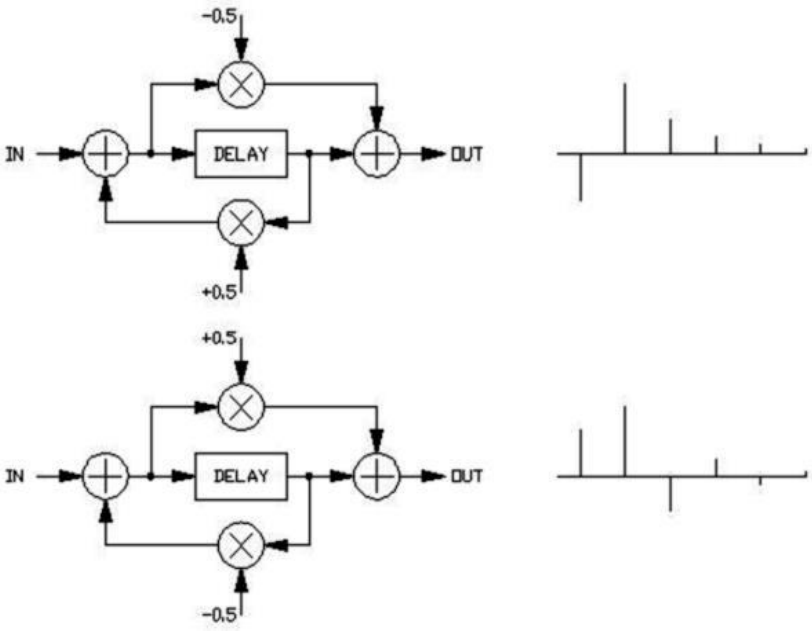
```
audio_sample_t Reverb::add_reverb(audio_sample_t sample_x)
{
    auto y0 = allpass_f0_.do_filtering(sample_x);
    auto y1 = allpass_f1_.do_filtering(y0);
    auto y2 = allpass_f2_.do_filtering(y1);
    auto y3 = allpass_f3_.do_filtering(y2);

    return wet_ * y3 + (1 - wet_) * sample_x;
}
```

Obrázek 31 – Reverb implementace

Místy záporná impulsivní odezva na kladný Diracův impuls je způsobena tím, že Allpass filtry použitého algoritmu jsou vytvořeny s na střídačku kladným a záporným znaménkem u koeficientu násobení. Více na toto téma v následující kapitole.

3.5.9 Základní filtry

Jako základní, pomocné filtry byly implementovány Allpass, Combo a FIR filtr, a to všechny primárně za účelem realizace Reverbu. FIR filtr se však moc nepovedl, nakonec nebyl v efektech ani použit. Combo filtr je implementovaný podle obecně známého schématu [4]. Stejně tak je realizovaný i Allpass filtr, pro který přikládám následující schéma. Vyobrazen je hned dvakrát, a to z důvodu možnosti prohození znamének u koeficientů násobení, což má za následek změnu impulsivní odezvy.



Obrázek 32 – Allpass filtr s různou impulsivní odezvou [16]

3.6 Android aplikace

S vývojem mobilních aplikací jsem doposud neměl žádné zkušenosti, tudíž mým prvním krokem byl průzkum existujících řešení podobné problematika a programovacích jazyků či frameworků zaměřených na tvorbu mobilních aplikací. Z představeného systému jasně vyplívá, že mezi kritické požadavky na aplikaci musí patřit Bluetooth komunikace s dalšími zařízeními a nějaký uživatelsky přívětivý způsob zvolení efektů, nastavení jejich parametrů a tvorba JSON objektu s těmito daty, který je následně přeposlán na Raspberry Pi. Těmito požadavky nejsem při výběru technologie nijak limitován, neboť jejich realizace je možná v kterékoliv z uvážených alternativ. Volil jsem tedy mezi Java, Kotlin a Flutter technologiemi.

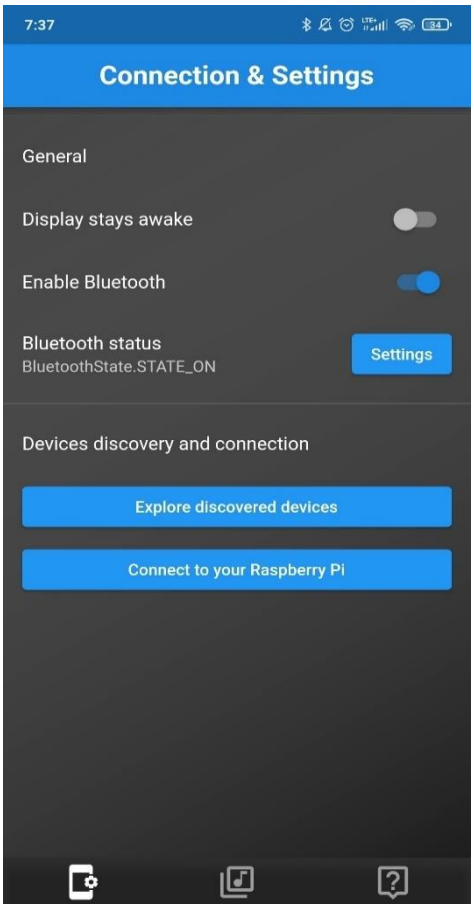
Zde byla moje volba poněkud strastiplnější, neboť v zadání této práce je pouze požadavek na podporu Android zařízení, což dokáže splnit jakákoliv z uvedených technologií. Navíc jsem doposud neměl žádné konkrétní preference. Z uvedených technologií mě nejvíce zaujal Flutter od společnosti Google. Jedná se o open-source multiplatformní technologii postavenou na programovacím jazyku Dart, díky které lze tvořit aplikace použitelné na široké škále různých zařízení. Důvodů proč mě zaujal bylo hned několik. Líbila se mi jeho prvotní rychlost a jednoduchost použití, Hot-reload funkce pro extrémně rychlé testování, multiplatformní přístup a i fakt, že ještě donedávna jsem netušil, že technologie jako Flutter vůbec existuje, a to výrazně zvětšilo moji zvědavost. Nicméně vítězem mého výběru se stal až v momentě, kdy jsem si v něm zkusil napsat malou, testovací aplikaci, během čehož jsem zkusil využít již existující knihovnu zaměřenou na Bluetooth komunikaci [20]. Tento pokus se vydařil, tudíž jsem věděl, že tady nešlápnu vedle.

	FLUTTER	KOTLIN	JAVA	REACT NATIVE
Supported Platforms	Android Jelly Bean, v16, 4.1.x and iOS 8+	Android and iOS 8+ versions	Android apps	Android 4.0.3+ versions and iOS 8+
Language Stack	Dart	JavaScript, and Native	Java works on JVM	JavaScript and React.JS
Performance	Removed JavaScript bridging, enhanced app speed	Interoperable with Java and Java Virtual Machine	Fewer bugs	Higher performance that that of native applications
Market and Community	New platform by Google	Rated among top ten programming languages	Huge community network of experienced developers	Utilizing ReactJS library and JavaScript
User Interface	Easy-to-use interface	Remarkable user experiences	Flexible, extensible, and scalable	Impressive graphical user interface
Pricing	Open-source platform	Free of cost	Paid updates for JDKcost	React native is open source
Specific Advantage	Hot-reload feature.	General-purpose programming language	Strong communities of experienced java developers	The framework offers high code reusability across platforms

Tabulka 1 – Porovnání technologií pro vývoj mobilních aplikací [21]

3.6.1 Komponenta Connection & Settings

Při spuštění aplikace je uživateli zobrazena první ze tří hlavních sekcí, a to *Connection & Settings* obrazovka. Jak název napovídá, tato obrazovka slouží zejména k zapnutí či vypnutí Bluetooth a dalším nastavením, jako například usínání uživatelova telefonu či průzkumu okolních zařízení. Poslední důležitá funkce je navázání spojení s Raspberry Pi, které je nutné pro primární funkci aplikace. Při zahájení komunikace je uživatel přesměrován na druhou ze tří hlavních sekcí, a to na Pedalboardu.



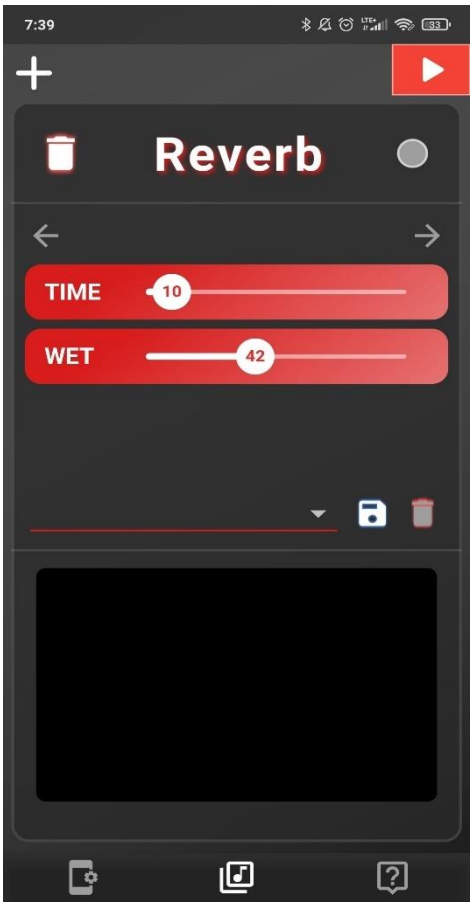
Obrázek 33 – Úvodní obrazovka

Bluetooth část *Connection & Settings* sekce je implementována prostřednictvím API externí knihovny [20]. Zapotřebí bylo pouze několika málo úprav z důvodu propojení se zbytkem aplikace. Bohužel ohledně modifikací ve způsobu komunikace jsem byl použitím této knihovny poměrně omezen. Většinou to nebyl nijak velký problém, kromě jedné situace s JSON objekty. Více o tomto tématu v kapitole 3.7 Bluetooth komunikace.

3.6.2 Komponenta Pedalboard

Představuje hlavní část aplikace, neboť se z ní ovládá veškerá funkcionalita spojená s efekty a živým hraním. Při prvním spuštění či přechodu na obrazovku je pedalboarda prázdná, konkrétně jsou na ni zobrazeny jen dva objekty, a to tlačítko

Plus pro přidání efektu a přepínací tlačítko Play/Stop pro aktivaci či pozastavení přenosu a zpracování audio signálu skrz Raspberry Pi.



Obrázek 34 – Vertikální pedalboarda

Plus tlačítko je triviální. Po kliknutí na něj je uživateli zobrazen list obsahující jako položky všechny dostupné efekty. Kliknutím na libovolnou položku je efekt, který tato položka představuje, přidán na pedalboardu.

Play/Stop přepínací tlačítko je zajímavější. Pokud je ve stavu Play, tak Raspberry Pi obdrží JSON objekt reprezentující aktuální nastavení pedalboardy pokaždé, když dojde ke změně tohoto nastavení. Příkladem budiž přidání nového efektu, smazání již přidaného efektu, zapnutí či vypnutí efektu, změna parametru zapnutého efektu atd. Při jakékoliv operaci, kterou může uživatel na pedalboardě provést a změnit tím její nastavení z pohledu řetězce efektů, je Raspberry Pi okamžitě informováno a patřičně na to zareaguje. Pro doplnění, změna parametrů na vypnutém efektu nemění aktuální nastavení pedalboardy, neboť se jedná o vypnutý efekt a podobně. Dokud je přepínací tlačítko ve stavu Stop, tak přes Raspberry Pi nepotečou žádná data, tudíž si může uživatel připravit pedalboardu podle svých preferencích a až poté ji zapnout, čímž aktivuje Raspberry Pi a může začít hrát. Výhoda tohoto přístupu spočívá i v tom, že pokud uživatel omylem opustí aplikaci nebo dojde k vypnutí či výpadku Bluetooth, tak na Raspberry Pi stále zůstává aktivní jeho poslední nastavení, tudíž ze strany mobilního zařízení nemůže dojít k nechtěnému či náhodnému výpadku.

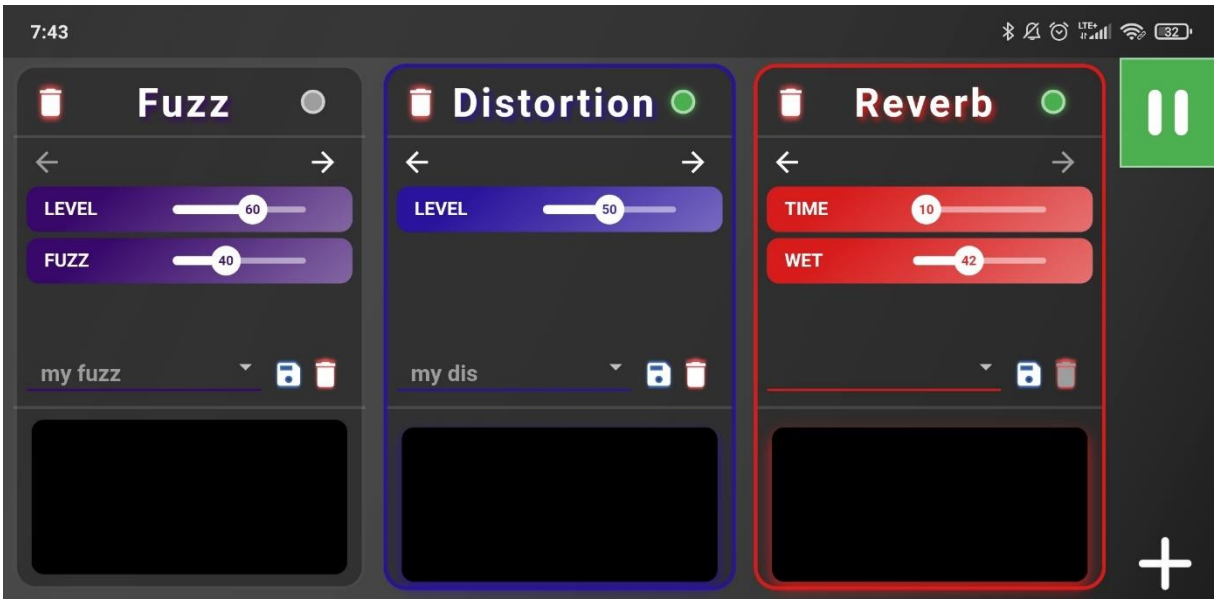
Efekty se skládají z dílčích komponent, kde každá takováto komponenta nabízí jistou část funkcionality efektu. Při pohledu odshora dolů tyto části jsou:

- Tlačítko odstranění efektu z pedalboardy
- Název efektu + indikátor, zda je efekt zapnutí či nikoliv
- Tlačítka šipek pro změnu pořadí efektu na pedalboardě
- Parametry efektu
- Sekce presetů
    - Vybrání presetu
    - Uložení aktuálního nastavení parametrů jako nový preset
    - Smazání vybraného presetu
- Tlačítko pro zapnutí či vypnutí efektu

Presety slouží k uložení aktuálního nastavení parametrů daného efektu. Lze je využít zejména při budoucím použití či rychlému a přesnému nastavení efektu podle již ověřené kombinaci parametrů. Každé efekt má své vlastní presety.

Při použití více jak jednoho efektu přestává být vertikální pohled dostatečně přehledný, neboť listování efektů je realizováno horizontálně. Z toho důvodu byl optimalizován i horizontální režim, ve kterém jsou efekty vizuálně zmenšeny, díky čemuž jich lze současně zobrazit vícero, ale neztratit během toho zásadně důležitou přehlednost. Z důvodu organizace efektů a snaze zmenšit je v poměru k jejich

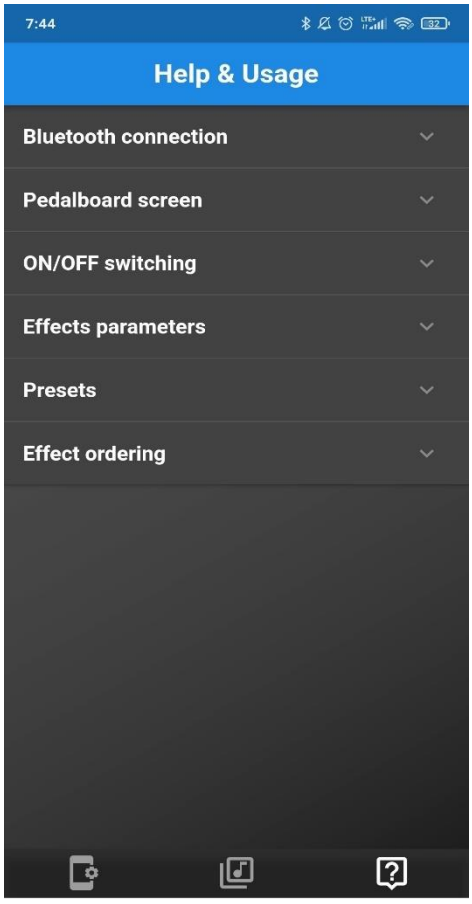
vertikální podobě, bylo pro tento pohled odstraněno navigační menu. Taktéž tlačítko pro přidání efektu bylo přemístěno do pravého spodního rohu. Je to cena za zachování co nejlepší přehlednosti, neboť jinak by musely být efekty vizuálně výrazně menší, což by značně zhoršilo jejich použitelnost.



Obrázek 35 – Horizontální pedalboarda

3.6.3 Komponenta Help & Usage

Poslední z hlavních sekcí aplikace se zaměřuje na pomoc uživatelům a dovysvětluje možnosti, které aplikace nabízí. Je zde blíže popsáno doporučené pořadí efektů, horizontální režim pedalboardy, efekty včetně jejich parametrů a další důležité informace, a to zejména pro zcela nové uživatele či neznalce kytarových efektů. Zobrazované objekty efektů obsahují už tak dost funkcionality a dílčích komponent, tudíž byly záměrně navrženy co nejminimalističtějším způsobem, a to i za cenu komplikovanějšího prvního použití, neboť při použití v reálném čase má hudebník jen omezený čas na použití a každé kliknutí je drahé. Navíc bylo třeba pro hladkou použitelnost dodržet i uživatelské potřeby, jako například velké a snadno použitelné zapínání a vypínání efektů, a to i za cenu spotřeby značného místa na obrazovce.



Obrázek 36 – Nápověda

3.7 Bluetooth komunikace

Poslední klíčová komponenta mého návrhu je způsob řešení komunikace mezi výše popsaným backendem na Raspberry Pi a mobilní aplikací. Řešení musí splňovat podmínky na dostatečnou rychlost a stabilitu spojení. Poměrně velká výhoda je skutečnost, že přes tento komunikační kanál nemusí být přenášeny žádná audio data, a to ani v reálném čase ani nijak jinak. Tato komunikace má sloužit pouze k přenosu aktuálně nastavených efektů, jejich parametrů a dalších řídících dat, tudíž komunikační kanál je zatížen pouze při některé uživatelské interakci s aplikací.

Při průzkumu možných řešení mého problému jsem váhal mezi dvěma technologiemi. Mý kandidáti byli Wi-Fi a Bluetooth.

Wi-Fi poskytuje v porovnání s Bluetooth větší dosah spojení, je lépe zabezpečená, dokáže v jeden moment zprostředkovat komunikaci mezi mnohem větší skupinou uživatelů. Naopak ale její provoz je náročnější na spotřebu energie a vyžaduje širší pásmo [18]. Pro a proti by se pro tyto technologie dalo nalézt značné množství. Nicméně jak už název kapitoly napovídá, vybral jsem si Bluetooth. Hlavním důvodem byla jednoduchost a flexibilita, respektive absence nutnosti internetového připojení. Navíc pro potřeby této práce je Bluetooth zcela dostačující.

Nyní se přesuneme ke konkrétnímu provedení. Mobilní aplikace GuitarPi pomocí Bluetooth naváže spojení s Raspberry Pi, na kterém běží skript napsaný v jazyce Python, který podle přijatých dat spustí backend s požadovanými parametry a následně se stará o je obsluhu. Zmíněný skript si lze prohlédnout na následujícím obrázku.

Skript představuje serverovou stranu komunikace, která donekonečna čeká na požadavek od klienta. Požadavky jsou formulovány jako JSON objekty, které obsahují informace o aktuálním stavu GuitarPi pedalboardě. Obsluha požadavku probíhá takto:

1. přečtení dat z JSON objektu
2. kontrola dat, zda má být pedalboarda aktivní
    2.1. pokud ne, aktuálně běžící nastavení pedalboardy je ukončeno
3. postupné zpracování a příprava dat pro následující použití
4. finální příprava příkazu
5. kontrola, zda proces předchozího nastavení pedalboardy je stále aktivní
    5.1. v případě aktivity dochází k jeho ukončení
6. tvorba nového procesu, který spustí pedalboardu s aktuálním nastavením

```
def data_received(data):
    global proc
    global path
    print(data)
    try:
        jsonData = json.loads(data)
    except Exception as e:
        print("Multiple json object!!")
        return
    settings = ""
    if jsonData['isPedalBoardActive'] == True:
        for effect in jsonData['effects']:
            if effect['isActive'] == False:
                continue
            settings += ' ' + effect['name']
            for parametr in effect['parameters']:
                settings += ' ' + str(parametr['value'])
        cmd = path + settings
        poll = proc.poll()
        if poll is None:
            os.killpg(os.getpgid(proc.pid), signal.SIGTERM)
        proc = subprocess.Popen(cmd, shell=True, preexec_fn=os.setsid)
    else:
        poll = proc.poll()
        if poll is None:
            os.killpg(os.getpgid(proc.pid), signal.SIGTERM)

s = BluetoothServer(data_received)
signal.pause()
```

Obrázek 37 – Bluetooth skript

V některých situacích však může hned v prvním bodě nastat problém, pakliže Bluetooth zrovna “zaspí” a uživatel provede dva obzvlášť rychlé požadavky. Příkladem budiž dvojklik na ON/OFF tlačítko efektu na zapnuté pedalboardě. Problémem je, že Raspberry Pi přijme ony dva požadavky jako jeden, což v případě JSON formátu znamená dvojitý JSON objekt, no ten bohužel není podporovaný, a tudíž je nevalidní. Bohužel se mi nepovedlo přijít na korektní způsob řešení tohoto problému, neboť z klientské strany využívám externí Bluetooth knihovnu, a tudíž jsem značně omezen v její modifikaci. Spokojil jsem s workaround řešením, a to že takto nevalidní požadavek jednoduše nezpracuju, což shodou okolností je pro výše popsaný případ správný výsledek.

Závěrem této kapitoly bych pouze retrospektivně zhodnotil, že volba pro Bluetooth a jeho provedení nebylo úplně šťastné. Raspberry Pi část komunikace je postavená na externí BlueDot knihovně, která zprostředkovává Bluetooth API [19]. Mobilní aplikace GuitarPi ke komunikaci skrz Bluetooth taktéž využívá externí knihovnu, tudíž mé možnosti ohledně ladění byli značně omezené. Jsem ale přesvědčen, že i v případě rozhodnutí se pro Wi-Fi by nastaly poměrně pestré a zábavné problémy.

3.8 Verze aplikace podle UI

Projekt má dvě verze, které se liší v způsobu interakce s uživatelem.

1. **Primární** provedení spočívá v již představeném systému, tedy backend běžící na Raspberry Pi a frontend je realizován jakožto Android mobilní aplikace.

2. **Sekundární** neboli konzolová verze projektu představuje pouze backend část. Lze ji použít na libovolném počítači pod operačním systémem Windows nebo Linux. Ovladatelná je z příkazové řádky a slouží zejména k rychlému testování efektů či postprodukci.

Primární verzi systému jsme již probrali výše, nyní si krátce vysvětlíme možnosti a použití konzolového přístupu. Na výběr jsou dva režimy:

1. **Playthrough** – editace vstupního signálu a jeho následné přesměrování na výstupní zařízení v reálném čase. V případě použití na libovolném zařízení (stolní počítač, notebook, Raspberry Pi) je však nutností přítomnost dostatečně flexibilní zvukové karty. Přesněji musí mít DAC, ADC, schopnost vzorkovat signál na dostatečné vysoké frekvenci (ideálně až 192 kHz, ale použitelných je i 44,1 kHz) a splňovat podmínku minimální bitové hloubky, která je 16 bitů. Až zbytečně dobrým příkladem je výše zmíněná zvuková karta HiFiBerry DAC+ ADC Pro.

2. **Playback** – editace již existujícího signálu a jeho následné přesměrování na výstupní zařízení, včetně možnosti uložení nově upraveného záznamu do souboru.

4 TESTOVÁNÍ

4.1 Volba testerů

Pro formální testování byli vybrání celkem čtyři testeři. První dva z nich jsou mladí, aktivně hrající muzikanti. Oba hrají v studentských kapelách. Jeden z nich dokonce ve třech zároveň. Druhou dvojici tvoří amatérští hudebníci, hrající pouze příležitostně. Kombinací těchto lidí vznikla pestrá a vyvážená skupina, a to jak ohledně zkušeností s hudbou a hraním, tak i znalostí kytarových efektů a jejich přímým použitím v praxi.

Toto testování je označeno za formální proto, že neformálně jsem aplikaci a efekty testoval i já a několik mých přátel.

4.2 Testovací scénáře

Následující scénáře byly sepsány tak, aby pokryly všechny aspekty funkcionality aplikace GuitarPi. Ověřuje se pouze funkčnost a uživatelská přívětivost mobilní aplikace a hudební provedení daných efektů, tudíž každý tester je postaven před předem připravené Raspberry Pi s nainstalovaných backendem systému. Tester si pouze nainstaluje aplikaci GuitarPi na svůj mobilní telefon.

Všem testerům byla předem pouze stručně vysvětlena myšlenka této práce. Dále byli obeznámeni, že v případě nejistoty či neschopnosti provést daný úkol mají nejdříve přímo v aplikaci nahlédnout do sekce Help & Usage a pakliže jim ani tyto informace nepomohou, bude daný úkol označen za Neúspěšný. Některé scénáře zahrnují i hru na kytaře. Například scénář pátý a jedenáctý.

1. Spusťte aplikaci GuitarPi a zapněte Bluetooth.
    a. volitelné – aktivujte režim Display stays awake.
2. Připojte se k Raspberry Pi.
3. Přidejte na pedalboardu efekt Distortion.
4. Nastavte jeho Level parametr na hodnotu 50 a přepněte ho do stavu ON.
5. Přepněte pedalboardu do stavu ON a krátce si zahrajte na kytaře.
6. Přidejte na pedalboardu efekt Reverb a přepněte ho do stavu ON.
7. Nastavte jeho parametry. Time na hodnotu 10 a Wet na hodnotu 40.
8. Změňte pořadí Vámi přidaných efektů tak, aby byly zapojeny v doporučeném pořadí.
9. Na Distortion efektu změňte hodnotu Level parametru na libovolnou jinou a vytvořte pro toto nastavení preset.
10. Opět na Distortion efektu změňte hodnotu Level parametru na libovolnou jinou.
11. Na Distortion efektu použijte Vámi vytvořený preset, krátce si zahrajte na kytaře a poté daný preset smažte.
12. Přepněte pedalboardu do stavu OFF a vypněte Bluetooth.

Scénáře je z důvodu ověření uživatelské přívětivosti nutné testovat s každým uživatelem dvakrát, neboť obrazovka Pedalboard nabízí v horizontálním zobrazení odlišné listování jednotlivými efekty. Funkcionalita však zůstává stejná. Druhý průchod tedy ověřuje, zda jsou efekty dostatečně čitelné a použitelné i v zmenšené podobě. Navíc díky tomu měli všichni testeři druhou příležitost si vyzkoušet nejen práci s aplikací, ale také i dostatek prostoru pro poslech a posouzení hudební kvality implementovaných efektů. K tomu se dostaneme v doplňujících otázkách testování.

4.3 Průběh testování

Scénář	Tester č. 1	Tester č. 2	Tester č. 3	Tester č. 4
1.	**Úspěch**	**Úspěch**	**Úspěch**	**Úspěch**
2.	**Neúspěch** nekompatibilita se Samsung zařízeními (chybějící právo pro Display stays awake funkcionalitu).	**Úspěch**	**Neúspěch** Raspberry Pi nenalezeno, komplikace s verzí Android 12.	**Úspěch**
3.	**Úspěch**	**Úspěch**	**Úspěch**	**Úspěch**
4.	**Částečný úspěch** Musel se podívat do Help & Usage.	**Úspěch**	**Částečný úspěch** Na první pokus zapnul místo efektu Pedalboardu.	**Částečný úspěch** Musel se podívat do Help & Usage.
5.	**Úspěch**	**Úspěch**	**Úspěch**	**Úspěch**
6.	**Úspěch**	**Úspěch**	**Úspěch**	**Úspěch**
7.	**Úspěch**	**Úspěch**	**Úspěch**	**Úspěch**
8.	**Částečný úspěch** Musel se podívat do Help & Usage (doporučené pořadí znal, nevěděl jak prohodit efekty).	**Částečný úspěch** Funkcionalita změny pořadí efektů pro něj nebyla intuitivní.	**Částečný úspěch** Tester nezná problematiku pořadí efektů. Trvá mu to, no s pomocí Helpu úlohu dokončil.	**Částečný úspěch** Musel se podívat do Help & Usage.
9.	**Částečný úspěch** Při tvorbě váhal, co je po něm vyžadováno (pojmenování presetu).	**Úspěch**	**Úspěch**	**Úspěch** Nebyl si však jistý, zda preset skutečně vytvořil. Uvítal by notifikaci o tvorbě presetu.
10.	**Úspěch**	**Úspěch**	**Úspěch**	**Úspěch**
11.	**Úspěch**	**Úspěch**	**Úspěch**	**Úspěch**
12.	**Úspěch**	**Úspěch**	**Úspěch**	**Úspěch**

Tabulka 2 – Průběh testování

4.4 Doplňující otázky

Po hlavní fázi testování proběhl s každým testerem krátký dialog ohledně jeho názoru na možnosti a provedení aplikace a na hudební kvalitu a parametry jednotlivých efektů. Během něho byl systém stále k dispozici, tudíž si testeři mohli dodatečně vyzkoušet libovolný efekt či kombinaci efektů. Důraz byl kladen na tyto otázky:

1. Bylo pro Vás použití efektů na pedalboardě dostatečně přehledné a intuitivní?
2. Dívali jste se do sekce Help & Usage a pokud ano, pomohla Vám dostatečně?
3. Měli byste zájem o samostatnou sekci pro správu veškerých presetů?
4. Co jiného byste zlepšili, popřípadě jakou další funkcionalitu byste uvítali?
5. Líbí se Vám po hudební stránce jednotlivé efekty?
6. Měl byste zájem o použití tohoto systému v praxi, například na pódiu při živém koncertě nebo na domácí či jiné hraní?

Odpovědi na první otázku se poměrně shodovali. Nejčastěji bylo zmíněno nepřehledné a neintuitivní prohození efektů při vertikálním zobrazení. Tento problém však řeší horizontální zobrazení, které všichni testeři chválili a od momentu jeho objevení chtěli používat pouze to. Druhý nejčastější problém byla nejasná funkce ON/OFF tlačítka ve spodní části efektu, a to z důvodu jeho grafického vzhledu, který testery nevybízel ke kliknutí na něj. Další komplikace nastaly už jenom jednotlivě. Například tester č. 1 by uvítal jasnější popis a tvorbu presetů.

Do sekce Help & Usage se kromě testera č. 2 podíval každý, a to hned několikrát. Všem ale dokázala dostatečně pomoct, aby daný úkol nakonec zvládli. Navíc se většinou jednalo pouze o prvotní neznalost, která se při opakovaném průchodu z důvodu horizontálního zobrazení vytratila.

Všichni testeři se shodli, že tuto sekci by spíš nevyužili. Tester č. 2 však přišel s myšlenkou existence presetů na celou pedalboardu, konkrétně dát uživateli možnost uložit celý řetězec efektů v daném pořadí i s jejich právě nastavenými parametry. S nápadem silně souhlasím, sám jsem nad ním ve fázi návrhu uvažoval. Tato funkcionalita bude zařazena do budoucího vývoje.

Čtvrtou otázku převážně pokryly odpovědi z předchozích otázek. Nově (od testerů č. 1 a 2) tu zazněla pouze úprava v způsobu změny pořadí efektů, respektive v jejich prohazování. V aktuální verzi musí uživatel kliknout na levou či pravou ikonu šipky podle toho, do kterého směru chce daný efekt posunout. Návrh na změnu spočívá v možnosti plynulého přetažení efektu prstem do daného směru místo zmíněného klikání na ikony šipek. Pro horizontální zobrazení by to byla příjemná funkcionalita, no pro vertikální poměrně nepoužitelná. Případná implementace této funkcionality by vyžadovala větší uživatelský průzkum a zamyšlení se, zda stojí za případnou nekonzistenci. Dále také tester č. 3 by uvítal tooltip či jiné potvrzení o úspěšném vytvoření nového presetu.

Pátá otázka zabrala nejvíce času, jelikož poslechnu efektů se testeři věnovali jak během průchodu primárním testem, tak i po jeho skončení, kdy následovaly doplňující otázky a volné hraní. Pro přehlednost jsou jejich odpovědi a pocity sepsány do následující tabulky.

Testování s první dvojicí testerů probíhalo ve školní učebně. S druhou dvojicí v domácím prostředí, tudíž poslechové podmínky nebyly zcela optimální ani v jednom z těchto případů. Dále bych ještě uvedl, že jako audio výstup bylo použito tranzistorové kombo FENDER Frontman 10G Black. Jedná se o jednodušší kombo nabízející pouze ovladač Gain nastavující zkreslení, dvoupásmový ekvalizér, tedy potenciometry Bass a Treble, a nakonec ovladač hlasitosti Volume. Během testování bylo měněno i nastavení tohoto komba, tudíž výstupy testování do jisté míry ovlivnil i tento faktor. Stejně tak výběr a použití dané kytary či v případě testera č. 1 použití jeho baskytary.

Efekt	Tester č. 1 Basa	Tester č. 2 Kytara	Tester č. 3 Kytara	Tester č. 4 Kytara
Delay	Ok, dělá co by měl.	Spokojenost, ocenil by navíc možnost nastavení počtu opakování.	Není zvyklý na Delay, opožděný zvuk ho mate.	Líbí se mu, nachází pro něj využití.
Echo	Ok, líbí se mu jeho chování, baví ho.	Dost ho baví různá nastavení, prý je ideální na takzvaný “cave rock”.	Lepší než pouhý Delay, dává mu větší smysl.	Líbí se mu, baví ho zkoušet různá nastavení.
Reverb	Ok, taktéž ho chválí. Líbí se mu různá nastavení parametrů.	Dobrý, líbí se mu. Ocenil by ořez výšek.	Prý nejlepší efekt, hodně si ho chválí. Má z něho pocit obří místnosti.	Taktéž se mu líbí, prý nejzajímavější efekt.
Distortion	Přirovnává ho k Fuzz efektu podle jeho zkušeností. Nezní mu symetricky (vysoké a nízké tóny).	Líbí se mu, na studentský punk prý ideální.	Vyhovují mu jen decentnější nastavení, silné zkreslení prý nemá rád. Nejlepší nelineární efekt.	Oceňuje jeho rockový zvuk, baví ho.
Overdrive	Nepoužitelný, zvuk je místy trhavý a praská.	Nelíbí se mu, zvuk občas praská.	Zápasí s dobrým nastavením, příliš se mu nelíbí.	Oceňuje rockový zvuk, vadí mu ale občasné praskání.
Fuzz	Zní “strašně”, ale baví ho nejvíc, asi nejlepší z nelineárních efektů.	Podobný názor jako na Distortion, líbí se mu v kombinaci s Reverbem.	Na jeho preference je to už moc silné zkreslení, Distortion mu vyhovuje víc.	Prý hodně podobný jako Distortion, není špatný.

Tabulka 3 – Hodnocení kvality efektů

Na poslední, šesté otázce se silně projevily rozdíly mezi testery. Konkrétně mezi první dvojicí, kterou tvořili poloprofesionální hudebníci, a druhou dvojicí, která se skládala z amatérských muzikantů.

První dvojice sdílela názor, že v tomto stavu se testovaný systém na pódium nehodí. Jako důvody uvedli nedostatečnou kvalitu efektů v porovnání s profesionálními řešeními a nemožnost zapínat a vypínat efekty s pomocí nohou. Ohledně domácího hraní je však situace jiná. Na takové použití by tento systém byli ochotni použít. Obzvlášť tester č. 2 projevil silný zájem testovat různé kombinace efektů a jejich nastavení i nad rámec testovaných scénářů a doplňujících otázek.

Druhá dvojice si v první řadě neuměla vůbec představit hraní na pódiu jako takové, neboť zatím hráli jen v hudební škole během výuky či s partou přátel v domácím prostředí. Další faktor byl ten, že oba tito testeři neměli žádnou předchozí zkušenost s kytarovými efekty, respektive s jejich použitím při hraní. Otázka živého hraní na pódiu šla tedy z cesty. Oba se však na sobě nezávisle shodli, že na domácí hraní a trénink je to skvělé řešení, které je bavilo používat a hrát si s možnostmi efektů. Tester č. 3 je sám začínající programátor, tudíž toto open-source řešení by dokázal využít i z edukačního měřítka.

4.5 Závěr testování

Musím přiznat, že testování bylo velice přínosné. Každý z testerů mi dal trochu jiný pohled na mé řešení a problematiku kytarových efektů obecně. Formálně jsem testoval systém pouze s čtveřicí hudebníků, s každým jsem však strávil dlouhých 90 až 120 minut. Detailně jsem zaznamenal všechny jejich poznámky a nápady na vylepšení, kterých nebylo vůbec málo, za co jsem jim neskutečně vděčný.

Testování přineslo řadu nápadů na možné budoucí změny a vylepšení, a to jak mobilní aplikace jakožto uživatelského rozhraní, tak i samotných efektů po hudební stránce. Nápomocný byl i fakt, že každý tester testoval aplikaci ze svého mobilního zařízení, čímž se odhalily drobné nedostatky ohledně několika málo nekonzistentností ve velikosti dílčích komponent efektů na pedalboardě.

Výsledek testování jasně dokazuje, že realizovaný systém má své neduhy a odladit ho pro maximální spokojenost cílové skupiny uživatelů by bylo velmi náročné, neboť v máločem se většina uživatelů shodne, a to jak v ohledu na mobilní aplikaci, tak i na hudební stránku efektů. Každopádně realizované řešení je funkční a dokazuje, že má smysl a využitelnost. Část ze zmíněných nedostatků již byla odstraněna, některé však stále zůstávají aktuální. Více o nich v kapitole 5.1 Budoucnost.

5 ZÁVĚR

Práci považuji za úspěšnou, neboť jejím cílem bylo navrhnout a implementovat systém za účelem digitálního zpracování zvuku v reálném čase na platformě Raspberry Pi, který bude možné ovládat prostřednictvím mobilní Android aplikace, a to se podařilo.

V teoretické části této práce jsem se nejprve zaměřil na již existující řešení, které sloužily jako inspirace pro mnou navržený systém, ve kterém jsem si dal za sekundární cíl odstranit některé klíčové nedostatky těchto už hotových řešení, a to z důvodu konkurenceschopnosti a odlišitelnosti. Poté jsem provedl průzkum v možnostech digitální zvukové syntézy s důrazem na typy filtrů a teoretickou přípravu pro následující implementaci hudebních efektů. Část efektů je realizována striktně podle této teorie. Některé efekty, příkladem budiž efekt Distortion, jsou ve finální verzi práce implementovány matematicky odlišným způsobem, a to z důvodu jednoduchosti v kombinaci s dodržením dostatečné hudební kvality efektu.

V praktické části jsem představil mnou navržený systém a popsal jeho realizaci. Práce byla otestována čtveřicí uživatelů z cílové skupiny, což jasně prokázalo její funkčnost a použitelnost. Avšak byly zaznamenány i různé nedostatky a z nich vyplívající návrhy na možné úpravy či vylepšení.

Ze zmíněných nedostatků považuji za ty největší následující dva, a to chybějící podporu mobilní aplikace pro Android verze 12, neboť mnou použita externí Bluetooth knihovna tuto verzi aktuálně nepodporuje, a nedostatečně kvalitní provedení efektu Overdrive, který po hudební stránce je jen těžko použitelný, jelikož nechtěné následky jeho zkreslení nejsou zamaskovány dostatečným způsobem a tím narušují výsledný zvuk.

Nedostatky efektu Overdrive se mi nepovedlo vyřešit z časových důvodů. Aktuální nepodpora Bluetooth knihovny pro Android 12 se doufám brzy vyřeší sama. Oba tyto případy si však každopádně zaznamenávám do budoucích úprav a vylepšení.

Na začátku zimního semestru 2021/2022 jsem o digitální zvukové syntéze a Raspberry Pi nevěděl vůbec nic. Toto téma jsem si zvolil z důvodu, že jsem kytarista samouk a začínající programátor, tudíž jsem zde cítil silné sympatie a viděl příležitost. Myslím, že jsem se vydal správným směrem.

5.1 Budoucnost

Možností pro budoucí vývoj a zlepšení je celá řada. Nejdříve by bylo adekvátní se zaměřit na výstupy z testování, neboť většina z nich jsou opodstatněná a poukazují za slabiny této práce, a to jak z pohledu uživatelského rozhraní, tak i efektů samotných. Realizace některých výstupů bude poměrně časově náročná, a to zejména vyladění některých efektů a vyřešení situace ohledně zájmu o možnost ovládání efektů prostřednictvím nohou. Je otázka, zda by se do druhého zmíněného výstupu mělo vůbec smysl pouštět a pokud ano, jak by měl být řešen a co přesně by mělo být jeho výstupem.

Dále se nabízí rozšířit systém o další efekty či dát uživatelům možnost tvorby presetů celé pedalboardy neboli si uložit nastavení a zapojení všech aktuálně použitých efektů pro opětovné použití.

Mě osobně zaujala i myšlenka na realizaci Looper funkcionality, kterou lze stručně popsat jako tvorbu, uložení a přehrání hudební nahrávky. To vše ale v reálném čase a s kapacitou na několik samostatných nahrávek, které lze v libovolném pořadí zapínat, vypínat, mazat a znovu vytvořit.

Mobilní aplikace je napsána pomocí multiplatformní technologie Flutter, což přináší výhodu snadné podpory iOS zařízeních. Bohužel v rámci realizace této práce jsem neměl přístup k žádnému iOS zařízení, tudíž jsem se na tuto cestu nevydal, avšak dveře jsou otevřeny dokořán.

Příloha A.

ZDROJOVÝ KÓD

Projekt je dostupný prostřednictvím tří GitLab ČVUT FEL repositářů:

1. Raspberry Pi část - https://gitlab.fel.cvut.cz/andrajak/audio-system-pi
    a. Backend systému
        i. Rozšíření knihovny IIM-AVLib (implementace efektů v C++)
        ii. Python skript pro potřeby Bluetooth komunikace

2. Android aplikace – https://gitlab.fel.cvut.cz/andrajak/guitarpi-app
    a. Frontend systému
        i. Android aplikace vytvořená prostřednictvím technologie Flutter

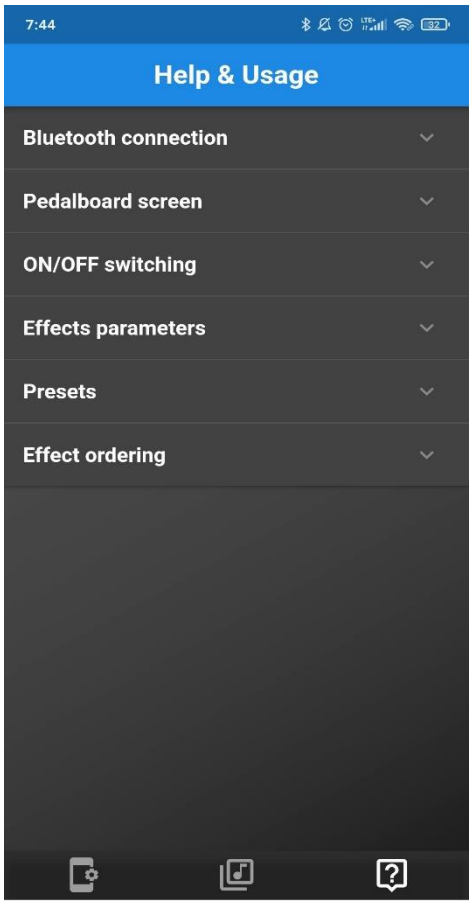
3. Instalační soubory - https://gitlab.fel.cvut.cz/andrajak/sada-zvukovych-nastroju-bp
    a. Raspberry Pi image
        i. Základní verze operačního systému Raspbian (datum verze: 2022-04-04) rozšířená o výše zmíněný backend a potřebné změny v konfiguraci Bluetooth.
    b. APK soubor
        i. Instalační soubor mobilní aplikace
    c. 3D modely držáků pro mobilní telefony
        i. Pro snadnější použití doporučuji využít držáku na mobilní zařízení. Jako možný příklad řešení nabízím dva volně dostupné 3D modely.

Příloha B.

PŘÍPRAVA A PRVNÍ ŠPUŠTĚNÍ

Prvním krokem je instalace přiloženého image souboru na Raspberry Pi a instalace APK souboru na dané mobilní zařízení. Poté následuje ještě krok druhý, a to propojení mobilního telefonu prostřednictvím Bluetooth s Raspberry Pi ručně, neboť Raspberry Pi potřebuje mít toto zařízení uložené jako známé. Pro jakékoliv další použití stačí pouze zapnout Raspberry Pi a připojit se k němu již jednoduše přímo z Vašeho mobilního telefonu.

V případě komplikací s použitím mobilní aplikace prosím nahlédněte do její nápovědy.



Obrázek 36 – Nápověda

TYPICKÝ PRŮCHOD APLIKACÍ

Před samotným použitím aplikace je nejprve nutnost připravit a zapojit potřebný hardware, konkrétně Vaše Raspberry Pi a příslušné vstupní a výstupní audio zařízení. Vstup může být kytara či jiný hudební nástroj podobného charakteru a jako výstup lze použít kytarové kombo, reproduktory či sluchátka. S takto připraveným a zapnutím hardwarem můžeme přejít k další fázi, a tou je spuštění aplikace GuitarPi.

Po kliknutí na ikonu aplikace Vás přivítá obrazovka zaměřená na Bluetooth komunikaci a nastavení obecně. Kromě Bluetooth je zde možné také nastavit, zda obrazovka Vašeho telefonu má usínat neboli po stanoveném čase zhasnout či nikoliv. Primární účel této sekce je v připojení se k Raspberry Pi, po kterém budete okamžitě přesměrováni do následující sekce aplikace, konkrétně na obrazovku Pedalboard. Tato obrazovka představuje nejdůležitější část celé aplikace, neboť právě zde lze spravovat hudební efekty a řídit Vaše Raspberry Pi.

Prázdná pedalboarda nabízí pouze dvě operace, a to ***Play/Stop*** přepínač pro řízení toku audio dat skrz Raspberry Pi a tlačítko ***Plus*** pro přidání efektu.

Pro použití daného efektu je tedy nutné ho přidat na pedalboardu, libovolně nastavit jeho parametry a aktivovat ho kliknutím na velké černé tlačítko ve spodní části každého efektu. Zda je efekt aktivní či nikoliv lze rozpoznat díky rozsvícení jeho diody v pravém horním rohu každého efektu a zároveň barevným zvýrazněním jeho obvodu. K odstranění efektu slouží tlačítko ***Delete***, které je znázorněno jako koš v levém horním rohu každého efektu.

Další důležitý faktor je **pořadí efektů** na pedalboardě, neboť přesně v tomto pořadí jsou jednotlivé efekty aplikovány na data z audio vstupu. Ke změně pořadí efektů lze použít šipky v horní části každého efektu. Pro více informací ohledné této a případně i jiných problematikách prosím nahlédněte do nápovědy, která představuje poslední sekci aplikace (Obrázek 35 - Nápověda).

LITERATURA

Odborná literatura, recenzované články:

[1] De Poli, Giovanni. "A tutorial on digital sound synthesis techniques." Computer Music Journal 7, no. 4 (1983): 8-26.

[2] Poepel, Cornelius, and Roger B. Dannenberg. "Audio signal driven sound synthesis." In ICMC. 2005.

[3] BERKA, Roman, František RUND, Libor HUSNÍK a Adam J. SPORKA. Multimédia I. Praha: České vysoké učení technické v Praze, 2016. ISBN 978-80-01-05859-6.

[4] Udo Zolzer. DAFX – Digital Audio Effects. JOHN WILEY and SONS, 2002.
ISBN: 0-471-49078-4.

Internetové články:

[5] Metody zvukové syntézy [online]. Telotone, 2009 [cit. 2022-01-09]. Dostupné z: http://elektronicka-hudba.telotone.cz/clanky/metody-zvukove-syntezy

[6] How to Start Programming Pedal-Pi [online]. ElectroSmash: Ray, 2017 [cit. 2022-01-09]. Dostupné z: https://www.electrosmash.com/forum/pedal-pi/202-how-to-start-programming-pedal-pi

[7] Audio Filter Types [online]. Producer Hive: Martin Peploe [cit. 2022-01-09]. Dostupné z: https://producerhive.com/music-production-recording-tips/audio-filter-types/

[8] Effects [online]. Teach Me Audio, 2020 [cit. 2022-01-09]. Dostupné z: https://www.teachmeaudio.com/mixing/equipment/effects

[9] Mixing with Reverb: How to Use Reverb for Depth Without Creating a Mess [online]. LedgerNote: JARED H. [cit. 2022-01-09]. Dostupné z: https://ledgernote.com/columns/mixing-mastering/mixing-with-reverb/

[10] Raspberry Pi [online]. České Budějovice: Michal Prenner [cit. 2022-01-09]. Dostupné z: https://rpishop.cz/

[11] PedalSHIELD UNO Arduino Guitar Pedal [online]. ElectroSmash [cit. 2022-01-09]. Dostupné z: https://www.electrosmash.com/pedalshield-uno

[12] How to Guide: Signal Chain for Your Pedalboard [online]. Roland Corp: Ed Lim [cit. 2022-05-02]. Dostupné z: https://rolandcorp.com.au/blog/order-effects-chain-simple-guide

[13] Tonebridge Guitar Effects [online]. Google Play: Ultimate Guitar USA LLC [cit. 2022-05-18]. Dostupné z: https://play.google.com/store/apps/details?id=com.ultimateguitar.tonebridge&hl=cs&gl=US

[14] Speed comparison of programming languages [online]. GitHub: Niklas Heer [cit. 2022-05-07]. Dostupné z: https://github.com/niklas-heer/speed-comparison

[15] IIM AV library [online]. Institut Intermédií [cit. 2022-05-08]. Dostupné z: https://projects.iim.cz/support:iimavlib:audio

[16] Audio Effects [online]. Spin Semiconductor [cit. 2022-05-11]. Dostupné z: http://spinsemi.com/knowledge_base/effects.html#Reverberation

[17] Schroeder Reverberators [online]. CCRMA – Stanford University: Julius Orion Smith III [cit. 2022-05-11]. Dostupné z: https://ccrma.stanford.edu/~jos/pasp/Schroeder_Reverberator_called_JCRev.html

[18] Difference between Bluetooth and Wi-Fi [online]. Geeks for Geeks: Manmeet Juneja [cit. 2022-05-12] Dostupné z: https://www.geeksforgeeks.org/difference-between-bluetooth-and-wi-fi/

[19] Bluetooth Comm API [online]. Blue Dot: Martin O'Hanlon [cit. 2022-05-17]. Dostupné z: https://bluedot.readthedocs.io/en/latest/btcommapi.html

[20] Flutter Bluetooth Serial [online]. GitHub: Edufolly [cit. 2022-05-17]. Dostupné z: https://github.com/edufolly/flutter_bluetooth_serial

[21] Mobile App Development [online]. Heptagon [cit. 2022-05-17]. Dostupné z: https://heptagon.in/2021/01/10/mobile-app-development-kotlin-vs-java-vs-flutter-vs-react-native/