VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ
DEPARTMENT OF TELECOMMUNICATIONS

EXPERIMENTÁLNÍ SOFTWAROVÝ HUDEBNÍ NÁSTROJ NA PLATFORMĚ ANDROID KOMBINUJÍCÍ SAMPLER, SYNTETIZÉR A SEKVENCER

EXPERIMENTAL SOFTWARE MUSICAL INSTRUMENT ON ANDROID PLATFORM COMBINING SAMPLER, SYNTHESIZER AND SEQUENCER

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **Karel Kučera**
AUTHOR

VEDOUCÍ PRÁCE — **doc. Ing. MgA. Mgr. Dan Dlouhý, Ph.D.**
SUPERVISOR

BRNO 2018

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní obor **Audio inženýrství**
Ústav telekomunikací

Student: Karel Kučera ***ID:*** 134345
Ročník: 3 ***Akademický rok:*** 2017/18

NÁZEV TÉMATU:

Experimentální softwarový hudební nástroj na platformě Android kombinující sampler, syntetizér a sekvencer

POKYNY PRO VYPRACOVÁNÍ:

Cílem práce je kompletně naprogramovat mobilní aplikaci, kterou je experimentální hudební nástroj pro platformu Android. Experimentálnost spočívá v kombinaci jednoduchého sampleru, syntetizéru a sekvenceru v rámci mobilní platformy.

DOPORUČENÁ LITERATURA:

[1] Electronic abd experimental music: technology, music and culture. Editace Thom Holmes. 3. vydání, Routledge, New York, 2008, 462 s. ISBN 978-0-415-95782-3.

[2] PUCKETTE, M. Theory and Techniques of Electronic Music, 2006. 337 s. online: http://msp.ucsd.edu/techniques.htm

Termín zadání: 5.2.2018 ***Termín odevzdání:*** 29.5.2018

Vedoucí práce: doc. Ing. MgA. Mgr. Dan Dlouhý, Ph.D.
Konzultant:

prof. Ing. Jiří Mišurec, CSc.
předseda oborové rady

Abstrakt

Cílem bakalářské práce je realizace aplikace jakožto experimentálního softwarového hudebního nástroje na platformě Android. Realizace částečně vychází z návrhu zpracovaného v semestrální práci. Aplikace je určena pro tablety a chytré telefony. Kombinuje funkce sampleru, sekvencéru a syntetizéru a disponuje vestavěnými efekty. Díky těmto vlastnostem umožňuje vytváření zvuků, sekvencí a skladeb. Aplikace je naprogramována pomocí vizuálního programovacího jazyku Pure Data. Grafické uživatelské prostředí je vytvořeno pomoci platformy MobMuPlat. Samotná práce se skládá ze tří částí. První část je teoretickým úvodem, kde jsou probrány teorie digitálního zpracování zvukových signálů, principy a druhy syntézy zvukových signálů, efekty a elektronické nástroje. V neposlední řadě i charakteristika jazyka Pure Data a platformy MobMuPlat. Druhá část je návrh grafického prostředí, kde jsou probrány funkce jednotlivých grafických prvků. V třetí části je pak popsána realizace aplikace pomocí jazyka Pure Data.

Klíčová slova

Sampler, sekvencer, syntetizér, samplovaní, looping, Pure Data, Android

Abstract

The aim of this bachelor thesis is the realization of the application as an experimental software musical instrument on the Android platform. The implementation is partly based on the draft elaborated in semester work. The application is designed for tablets and smartphones. It combines functions of sampler, sequencer and synthesizer and contains built-in effects. With this features, it is possible to create new sounds, sequantions and songs. The applications is programmed by visual programing language Pure Data. The graphic interface is created by the MobMuPlat platform. The thesis is consists of three parts. The first part is a theoretical introduction, where are discussed, the theory of digital audio signal processing, principles and types of sound signal synthesis, effects and electronic instruments. Furthermore, the Pure Data and MobMuPlat platform features. The second part is an updated design of the graphical interface and there are discussed the functions of individual graphic elements. The third part describes implementation of the application by Pure Data

Keywords

Sampler, sequencer, synthesizer, sampling, looping, Pure Data, Android

KUČERA, K. *Experimentální softwarový hudební nástroj na platformě Android kombinující sampler, syntetizér a sekvencer.* Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2018. 75 s. Vedoucí bakalářské práce doc. Ing. MgA. Mgr. Dan Dlouhý, Ph.D..

Prohlášení

Prohlašuji, že svou bakalářskou práci na téma „Experimentální softwarový hudební nástroj na platformě Android kombinují sampler sekvencér a syntetizér“ jsem vypracoval(-a) samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.
Jako autor uvedené bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této bakalářské práce jsem neporušil(-a) autorská práva třetích osob, zejména jsem nezasáhl(-a) nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom(-a) následku porušení ustanovení § 11 a následujících autorského zákona c. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonu (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

V Brně dne ....

podpis autora

Poděkování

Tímto bych chtěl poděkovat doc. Ing. MgA. Mgr Danu Dlouhému Ph.D za odborné rady a cenné připomínky při vedení mé Bakalářské práce. Dále bych chtěl poděkovat doc. Ing. Jiřímu Schimmelovi Ph.D. za poskytnutí odborné literatury a referencím na další zdroje. Nakonec bych chtěl ještě poděkovat MgA. Janu Kavanovi Ph.D. a MgA. Tomáši Hrůzovi za odborné rady ohledně programovacího jazyka Pure Data.

V Brně dne ....

podpis autora

Obsah

Seznam obrázků

Úvod

S příchodem chytrých telefonů a tabletů nastal zlom v tom, jak jsme se do té doby dívali na mobilní telefony. Google přišel s open source operačním systémem Android, jeho rival Apple vyvinul stabilní userfriendly[1] iOS pro Iphony Ipady. Tyto firmy stály u zrodu nového odvětví informačních technologií, které je celosvětově nejdynamičtější vůbec. Mobilní telefony a tablety již přestaly primárně sloužit jen k pouhému telefonování, ale staly se našimi osobními počítači, mnohem osobnějšími než Laptop PC/MAC.

Vlastnosti chytrých telefonů a tabletů jsou jedinečné. Připojení k internetu, dotykový displej, rozšířená konektivita, gyroskop, GPS. Navíc jejich operační systémy umožňují snadné a rychlé vyvíjení nových aplikací. Tohle všechno a mnoho dalších vlastností zvyšují potenciál chytrého telefonu a tabletu coby osobního interaktivního pomocníka a rádce, který neustále nabízí nové možnosti jeho využití.

Jednou z takových možností je využití chytrého telefonu jako hudebního nástroje. Aplikace, které se snaží udělat ze smartphonu nebo tabletu elektrický hudební nástroj, který má člověk neustále po ruce, na sebe nenechaly dlouho čekat. Dnes můžeme na Google-play nebo Apple I-store nalézt mnoho aplikací, jež fungují jako syntezátory, samplery, sekvencery, loopry atd. Většinou jde ale o freeware aplikace, které mají omezené funkce a využívají zažité postupy. Navíc často těmto aplikacím chybí například možnost záznamu a vyexportování případné skladby. Pokud umělec potřebuje něco specifického a originálního, často je nucen si takovou aplikaci vytvořit sám a to bude i mým cílem.

V této bakalářské práci se budu zabývat návrhem a následnou realizací aplikace jakožto experimentálního softwarového nástroje na platformě Android. K tomu budu využívat vizuální programovací jazyk Pure Data. Jedinečnost aplikace bude tkvět v tom, že spojí funkce sampleru, sekvencéru a syntezátoru s vestavěnými efekty. Díky těmto vlastnostem půjde vytvářet jedinečné zvukové vzorky, sekvence a celé skladby. Výsledné sklady půjde z aplikace vyexportovat pro pozdější reprodukci. Případně do ní půjde i naimportovat nové zvukové vzorky. Aplikace by měla být využitelná nejen jako elektronický hudební nástroj pro živé vystupování, ale také jako nástroj pro vytváření a zaznamenávání skladeb elektronické hudby.

Bakalářská práce je rozčleněna do tří částí. První část je teoretická a zabývá se stručně teorií vzniku zvuku a lidského hlasu, digitálním zpracováním zvukového signálu, syntézou zvukových signálů, elektronickými nástroji a některými efektovými procesory a efekty. Dále bude stručně popsán operační systém Android. Konec teoretické části je vyhrazen popisu programovacího jazyka Pure Data a GUI platformě MobMuPlat. V druhé části bude popsán aktualizovaný návrh funkčnosti a grafický návrh aplikace, z kterého jsem vycházel při samotné realizaci. Pomocí blokových schémat a obrázků popíšu jednotlivé funkční bloky aplikace. Dále budou rozebrány funkce jednotlivých ovládacích prvků grafického rozhraní. Třetí část se zabývá samotnou realizací aplikace. Zde bude přiblíženo, jak jsem

[1]Userfriendly - uživatelsky přívětivý.

[2] VST (Virtual Studio Technology) Softwarové rozhraní pro komunikaci mezi hostitelským programem a

postupoval při realizaci aplikace a podrobně popíšu funkci jednotlivých částí. Na závěr shrnu, jestli zvolené metody realizace byly vhodné a jak aplikace funguje.

1 Teoretická část

1.1 Zvuk

Zvuk je mechanické vlnění ve hmotném prostředí, které vyvolává sluchový vjem. Částice takového prostředí kmitají kolem své rovnovážné polohy a působí na sebe pružnými silami. Jedna kmitající částice ovlivní sousední částici a ta zas další sousední částice. Tak vzniká zvukové vlnění. Charakter šíření je závislý na typu prostředí, ve kterém se zvuk šíří. Rozlišujeme pak šíření: příčné (stojaté vlnění na struně), podélné (vlnění ve vzduchu). U postupného podélného vlnění mluvíme o tzv. zhušťování (místa kde jsou částice nejvíce u sebe) a zřeďování částic (místa kde jsou od sebe částice nejdále.) [1][2]

1.1.1 Akustický tlak

Je rozdíl mezi okamžitou velikostí celkového tlaku v daném bodě zvukového pole a statickou hodnotou atmosférického tlaku. Množina akustických tlaků v určitém místě definuje **zvukové pole**. V každém bodě zvukového pole se bude hodnota celkového tlaku měnit v čase.

Hlasitost je subjektivní veličina je závislá právě na velikosti akustického tlaku.

Platí že: $p = \frac{F}{S}\ [Pa; N; m^2]$

Odpovídající měřitelnou veličinou je hladina akustického tlaku.

$$L_p = 20\log\frac{p}{p_0}\ [\mathrm{dB};\ \mathrm{Pa};\ \mathrm{Pa}]$$

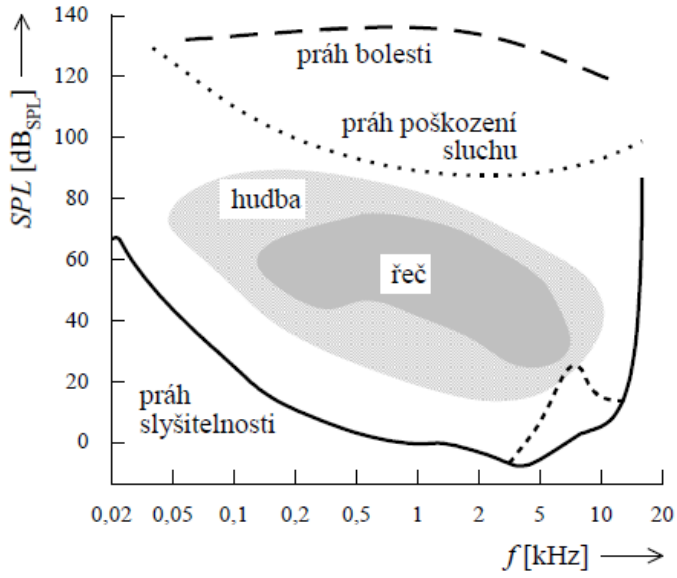
Užívá se logaritmického vyjádření jednotek Decibel, protože slyšitelný rozsah přesahuje sedm dekadických řádů L_p. [1][6]

1.1.2 Lidské vnímání zvuku

Obecně lze říci, že většina lidí vnímá zvuk ve frekvenčním rozsahu 20Hz – 20kHz. Nejvýznamnější část pásma je 2-4kHz kde je lidské ucho nejcitlivější kvůli srozumitelnosti řeči. Práh slyšitelnosti udává minimální hodnotu akustického tlaku, při kterém je lidský sluch schopen ještě rozeznat zvuk s daným kmitočtem. Práh bolesti a práh poškození sluchu udává hodnotu, při kterém sluchový orgán není schopen rozeznat další zvýšení hladiny akustického tlaku a kdy dochází k poškození sluchu. [1][2][4]

1.1.3 Tón a hluk

Hluk je z hlediska hudby nepravidelný zvuk nejednoznačné frekvence a neurčité výšky. Zvuky z neurčité výšky se, ale v hudbě používají. Jsou to například bicí nástroje, šum, syčení. Naproti tomu tón má stálou frekvenci a v čase se pravidelně periodicky opakuje. Při jeho poslechu vzniká v uchu vjem z určité výšky tónu.[2][1]



obr. 1-1: Oblast slyšení [6]

1.1.4 Výška zvuku

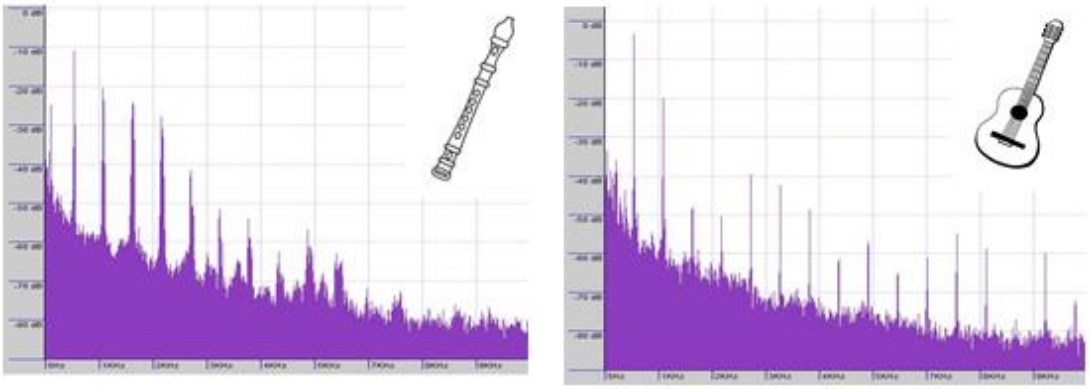
Podle ČSN 01 1600 „ vlastnost zvukového vnímání umožňující uspořádat zvuky na stupnici v rozsahu od hlubokých po vysoké“. Jednotkou výšky zvuku je mel. 1000 melů = 1 kHz při hladině akustického tlaku 40 dBspl. Přesné určení výšky určitého zvuku je u každého člověka individuální. Průměrný posluchač s relativním sluchem může stanovit rozdíly výšky dvou tónu případně i odchylky mezi nimi. Zatímco posluchač s absolutním sluchem má schopnost stanovit výšku tónu i jeho odchylky. Takových lidí je však velmi málo.[1][6]

1.1.5 Maskování

Pokud zvuk o určité výšce a dynamice vyvolá sluchový vjem, který odpovídá určité prahové hodnotě a ve velmi blízkém okamžiku (řádově milisekundy), přijde další zvuk z nižšími hodnotami, může být tento zvuk zeslaben nebo zcela potlačen. Tzn. při souznění několika podobných zvuků, se může stát, že půjde slyšet nejvýraznější z nich, ostatní se skryjí jakoby za něj. Tento jev se nazývá Maskování a způsobují ho naše nervové buňky, které nejsou schopny v jeden okamžik rozeznat různé prahové hodnoty. [2][1]

1.1.6 Barva zvuku

Barva zvuku nebo také Témbr (timbre) je vlastnost zvuku, která určuje odlišnost např. dvou stejných tónů zahraných různými hudebními nástroji. „*Barva zvuku resp. tónu je dána počtem a intenzitou jednotlivých harmonických složek“.(citováno z [1]).* Hudební nástroje se vyznačují určitou nezávislostí na poměru zmíněných intenzit spektrálních složek a jejich absolutních výškách. Kdežto u vokálu je barva určena zvýrazněním jedné nebo dvou charakteristických spektrálních složek, tzv. formantů. [1][2][6]



obr. 1-2: Barva tónů flétny a kytary [19]

1.1.7 Lidský hlas

Lidský hlas vzniká v hlasovém ústrojí, které se skládá z hrtanu a hlasivek. Hlasivky jsou zúžená oblast hrtanu umožňující vznik hlasu rozkmitáváním hlasivkových řas pomocí výdechového vzduchu proudícího z plic. Rozkmitávání vzniká jako výsledek tlakových poměrů, který v hlasivkách panuje. Ojedinělost každého lidského hlasu je dána, různými rozměry součástí hlasového aparátu. Jeho jedinečnost se projevuje odlišnou základní frekvencí vznikající v hrtanu a hlavně vyššími harmonickými frekvencemi, které vznikají nadhrtanových dutinách a tvoří jedinečnou barvu hlasu. Od základní frekvence jsou tedy odvozené všechny další tónové složky řeči. Nadhrtanové dutiny slouží mimo jiné jako rezonátor, který propouští jen některé svrchní harmonické tóny, například souhlásky, ale i některé samohlásky. Hlasivkami je možné vytvořit i šum. V tom případě hlasivky nekmitají, vytvoří pouze štěrbinu, kterou proniká vzduch.

Frekvenční rozsah lidského hlasu určuje v jakém rozsahu, (od jakého nejnižšího tónu po jaký nejvyšší tón), dokáže člověk vydávat tóny pomocí svého hlasového ústrojí. Rozsah hlasu necvičeného dospělého člověka je asi 1,5 oktávy. Šířka frekvenčního spektra hlasu je cca od 100 Hz do 1000 kHz. Lidský hlas se mění věkem. Jak člověk stárne, mění se jeho hlasové ústrojí a barva zvuku které vydává.

Lidský hlas je v hudbě využívám především ke zpěvu, což je nepřirozenější forma hudebního umění. Zpěv se od řeči a recitace odlišuje cíleným střídáním tónů a rytmiky poskládaných do skladby. Z hlediska tónové výšky, které jsou schopni zpěváci a zpěvačky zazpívat, rozlišujeme ve zpěvu hlasy

ženské:

- Soprán(246-1046 Hz)
- Mezzosoprán (196-880 Hz)
- Alt (164-659Hz)

hlasy mužské:

- Tenor (123-523Hz)
- Baryton(98-392)

- Basbaryton
- Bas (82 – 329 Hz).

Existují i alternativní přístupy k pojetí zpěvu např.

- Alikvótní zpěv: technika při, která se vytváří alikvótní tóny a vzniká tak iluze dvojhlasu
- Beatbox: Napodobování zejména bicích, perkusních ale i jiných nástrojů pomocí hlasu.
- Screaming: Hrdelní řev, při kterém hlasivky zkreslují a obohacují spektrum o harmonické složky.
- Growling: je to technika hlubokého mručení a řevu vycházejícího ze screamingu

V této kapitole bylo čerpáno z těchto zdrojů:[1], [2], [3], [4], [6]

1.2 Digitální zpracování zvukového signálu

1.2.1 Zvukový signál

Jako signál označujeme fyzikálního nositele informace což, je v případě zvukového signálu zvuk. Pokud je ale mechanické kmitání částic v prostředí označováno správně jako zvuk (sound), je zde v podstatě mediem vzduch. Všechny ostatní fyzikální nositele zvukové informace označujeme jako zvukové signály (audio/audion signals).

1.2.2 Spektrum zvukového signálu

Téměř každý periodický signál si můžeme pomocí Fourierovy transformace rozložit na základní harmonické tóny, respektive vyjádřit pomocí harmonických funkcí sinuse nebo kosinus. Signály převedeme z časové oblasti do frekvenční se spojitým nebo diskrétním (určitý okamžik) časem. Takže pokud si převedeme jakýkoli zvukový signál do frekvenční oblasti, jsme schopni určit jeho základní harmonickou frekvenci a na základě vyšších spektrálních složek rozeznáváme jeho již zmíněnou barvu. V grafickém vyjádření tvoří spektrum soustava svislých čar náležících jednotlivým frekvencím.

1.2.3 Převod AD

Převod zvukových signálů z analogových na digitální, jejich následné zpracování a převod zpět do digitální podoby, podléhá určitým pravidlům, na zpracování číslicových signálů. Při tomto procesu dochází ke zkreslení signálu, který, pokud není zanedbatelný, se snažíme snížit tak aby nebylo lidským sluchem rozpoznatelné. Převod Analogových signálu na digitální probíhá ve třech etapách: vzorkování, kvantování a kódování.

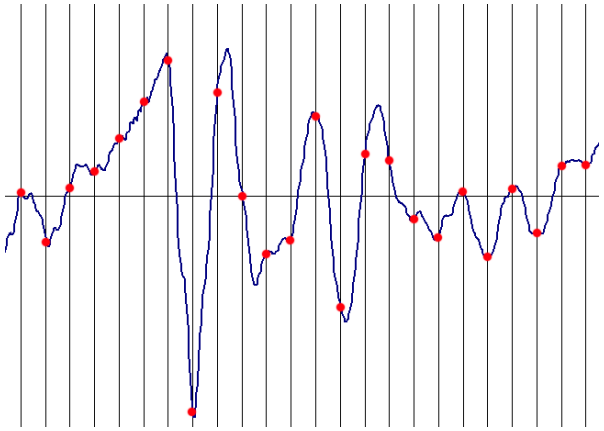
1.2.1 Vzorkování

Je proces, při kterém je spojitý signál rozdělen v čase na sled diskrétních vzorků. To znamená, že časovou osu rozdělím na určité úseky, jejichž úrovně odpovídají původnímu signálu v určitém bodě. Počet těchto vzorků za jednotku času udává **vzorkovací frekvence** f_{vz}. Při vzorkování, ale dochází k určité degradaci spojitého signálu, protože signál je vlastně jen množina bodů s intervalem odpovídajícím f_{vz} . Právě i proto musíme dodržovat vzorkovací teorém (Nyquistův-Shanonův) $f_{vz} > 2 f_{max}$. Ten udává, že vzorkovací frekvence musí být 2 krát vyšší než maximální frekvence vzorkovaného signálu. Jinak může docházet k takzvanému aliasingu a to znamená, že v rekonstruovaném signálu se můžou objevit i složky, které v původním signálu nebyly. Před A/D převodník se proto dává aliasingová dolní propust, která vyfiltruje ze signálu spektrální složky o kmitočtu $f > f_{vz}/2$.

Kmitočtový rozsah slyšení lidského ucha je, 20 Hz – 20 kHz, proto by měla být f_{vz} audio signálu vyšší jak 40 kHz. Nicméně rozsah řeči se pohybuje cca v pásmu 150 Hz – 200 Hz a rozsah hudby se

pohybuje v oblasti 50 Hz – 10 kHz. Takže pro určité zpracování zvuku stačí i nižší vzorkovací frekvence.

Pro vzorkování se používá tří základních vzorkovacích kmitočtů: **32 kHz** (pro rozhlasové digitální vysílání vyhází z šířky pásma 15 kHz pro stereobázi FM vysílání) **44,1 kHz** (používaný v záznamu videa a byl převzat do záznamu na CD) **48 kHz** (byl zaveden pro záznam na pásek a dnes je standardem pro profesionální digitální videozařízení). Pro vyšší kvalitu zaznamenaného zvuku se používají i vzorkovací frekvence dvou až čtyř násobky 44,1 kHz a 48 kHz: 88,2 kHz, 96 kHz, 176,4 kHz a 192 kHz, tyto frekvence jsou ale náročnější, kvůli množství a velikosti zpracovávaných dat. Pro běžný poslech ale dostačují základní vzorkovací frekvence.



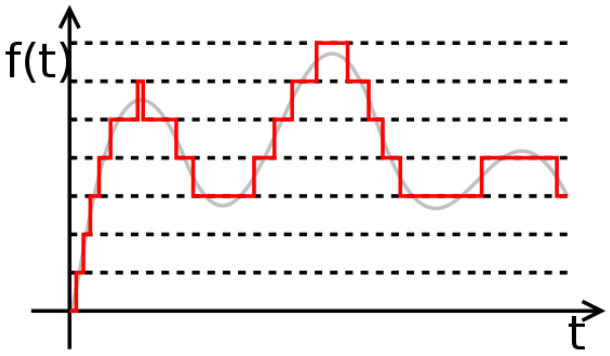
obr. 1-3: Navzorkovaný signál

1.2.2 Sigma – Delta převodník

Dnes se pro A/D převod užívá převážně sigma/delta převodníky, což je synchronní převodník v podstatě lineární filtr, jehož jádrem je integrátor a komparátor. Převodník využívá principu oversamplingu, kdy je navzorkovaný signál znovu Nkrát převzorkován se stupněm převzorkování 64 nebo 128. Díky tomu získáme mnohem vyšší odstup signálu od šumu a signál je tak čistější.

1.2.3 Kvantování

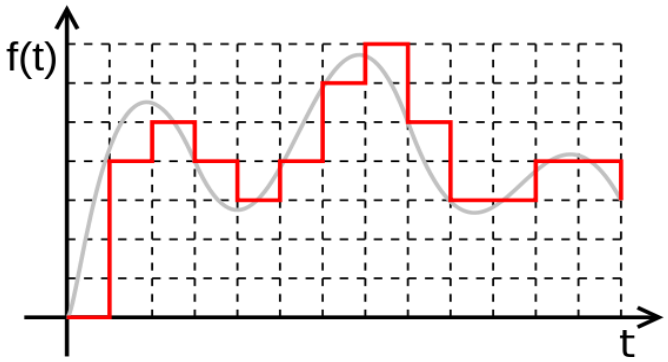
Kvantování je proces, při němž se navzorkovaný signál rozdělí na kvantizační hladiny, které mohou nabývat omezeného počtu úrovní. Počet těchto hladin je dán tzv. bitovou hloubkou (počtem bitů následného kódování). Při kvantování dochází vždy k určité ztrátě původní informace, záleží právě na bitové hloubce, čím je vyšší, tím nižší je ztráta a tím vyšší je kvalita kvantovaného signálu. Pokud je počet úrovní kvantizačních hladin příliš nízký, dochází ke kvantizačnímu šumu, což je odchylka kvantovaného signálu od původního. Kvantovací šum je, ale maskován užitečným signálem, (lidským sluchem nerozpoznatelný), již od rozlišení 16 bitů. Zvýšení bitové hloubky o 1 bit odpovídá odstupu signálu od šumu o 6 dB. Proto i kvůli dynamické rezervě krátkodobých špiček signálu je vhodné požívat vyšší bitové hloubky. Používají se např. 24,32,64,128.



obr. 1-4: Kvantovaný signál

1.2.4 Kódování

Kódování je proces, který následuje hned po vzorkování a kvantování při něm se vzorky i úrovně v diskrétním čase převádí na Binární číslo. To tak, že každé navzorkované a kvantované hodnotě je přidělené určité celé číslo, reprezentované binárním číslem. Při kódování může docházet stejně jako u kvantování a vzorkování, k určité ztrátě informace záleží, jaký typ modulace zvolíme. Především v profesionální audiotechnice se používá tzv. PCM (*Pulse Code Modulation*), při ni nedochází ke ztrátě informace a omezení šířky pásma jako například u delta modulace (DPCM). Princip PCM spočívá v pravidelném odečítání hodnoty signálu a jejím záznamu v Binárním čísle.



obr. 1-5: Digitální signál

1.2.5 Číslicové zpracování

Interní číslicové zpracování diskrétního zvukového signálu v digitální a výpočetní technice musí mít vyšší rozlišení než vstupní číslicový signál a to především z důvodu zaokrouhlovacích chyb při odečítání a dělaní a pak z důvodu zabránění nelineárního zkreslení při překročení hladiny 0dBFS. Používá se rozlišení 24 nebo 32 bitů pro zpracovávaná čísla s pevnou řádovou čárkou a rozlišení 32 nebo 64 pro čísla s plovoucí řádovou čárkou. Zesílení zpracovávaného číslicového signálu se provádí určitou konstantou a je třeba si uvědomit, že krom užitečného signálu se touto konstantou násobí i kvantizační šum.

1.2.6 Převzorkování číslicově zpracovaných signálů

Převzorkování se používá ke konverzi vzorkovacích kmitočtů při přenosu signálu mezi systémy s jiným vzorkovacími kmitočty. Signály můžeme podvzorkovat, aby nedošlo k aliasingu, musíme v tom případě omezit kmitočtové spektrum dolní propustí s mezním kmitočtem $f_{vz} / 2M$ (kde M je celé číslo poměru vzorkování). Při nadvzorkování obsahuje kmitočtové spektrum zperiodizované složky původního signálu, které se odstraní aliasingovým filtrem s mezním kmitočtem $f_{vz}/2L$. (kde L je celé číslo poměru vzorkování). Při převzorkování se signál nejprve nadvzorkuje a poté podvzorkuje. Nevýhodu je vyšší náročnost a dvojí filtrace aliasingovým filtrem.

1.2.7 DA převod

DA převod provedeme tak, že zredukujeme rozlišení bitové hloubky signálu podle rozlišení D/A převodníku. Po té se dekódují Binární čísla na signál ve spojitém čase, ale diskrétní úrovní. Nakonec se signál převede rekonstrukční dolní propustí na spojitý signál

1.2.8 Dithering a Jitter

Při převodu D/A kdy se některé úrovně signálu useknou nebo zaokrouhlí, vznikne kvantovací chyba. Ta se odstraní Ditheringem, což je proces kdy se k bitům, které budou useknuty, přičte náhodné číslo. Jitter nastává zase při fázové nestabilitě převodníků A/D a D/A, když jsou jejich vzorkovací frekvence rozdílné, vznikne tzv. Fázový šum. Tyto odchylky odstraníme speciálním obvodem zvaným fázový závěs.

V této kapitole bylo čerpáno z těchto zdrojů: [6], [8]

1.3 Syntéza hudebních signálů

Můj softwarový hudební nástroj bude využívat určité typy syntézy, proto se zde o nich stručně zmíním. Syntéza obecně znamená spojování určitých složek nebo částí do jedno celku. Její princip spočívá v obohacování nebo naopak ochuzování spektra o určité vyšší harmonické kvůli získání požadované barvy zvuku. Syntéz zvuku existuje několik druhů. Z hlediska zpracování signálů, je můžeme rozdělit na Analogové a Digitální. Analogové lze pak rozdělit na lineární a nelineární. Lineární nepodléhají nelineárnímu zkreslení a lze je rozdělit na Aditivní (součtové) a Subtraktivní (Rozdílové). Nelineární metody lze rozdělit na Modulační a Tvarovací. Všechny zmíněné syntézy lze vytvářet digitálně, existují ale speciální digitální syntézy signálů např. tabulková spektrální a fyzikální modelování.

V syntéze hudebních signálu rozlišuje tyto základní obvody.

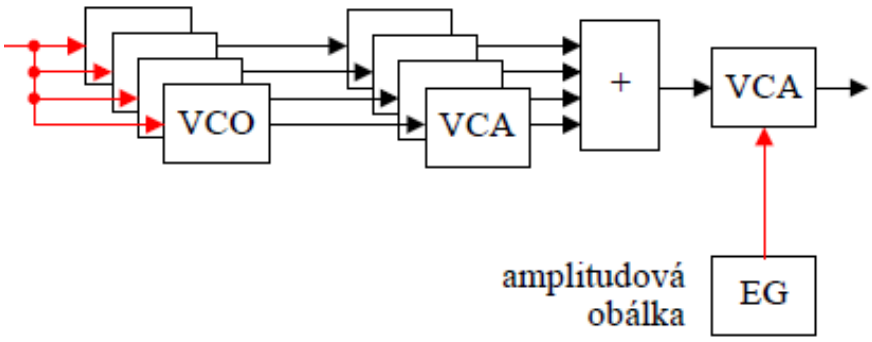
- **VCO -** Voltage Controlled Oscillator, napětím řízený oscilátor: Generátor periodického signálu s okamžitým kmitočtem řízeným velikostí řídicího napětí.

- **VCA** - Voltage Controlled Amplifier, napětím řízený zesilovač: zesilovač, jehož okamžitá hodnota zesílení je dána velikostí řídícího napětí
- **VCF –** Voltage Controlled Filter, napětím řízený filtr: kmitočtový filtr, jehož okamžitá hodnota mezního/středního kmitočtu (podle typu filtru) a případně další parametrů je dána velikostí řídícího napětí
- **NG –** Noise Generator, generátor šumu
- **LFO –** Low Frequency Oscilator, nízkofrekvenční oscilátor: (sub-akustický,pomaluběžný) generátor periodického nebo náhodného signálu s kmitočtem řádově 0,1 – 10 Hz
- **EG –** Envelope Generator, generátor obálky: generátor kterým se mění tvar průběhu v čase. Nejrozšířenějším modelem je čtyř-segmentová obálka ADSR.

[7], [5]

1.3.1 Aditivní Syntéza

Aditivní syntéza vytváří nový zvuk skládáním více signálů. Principiálně vychází z Fourierovy řady (pozn. Pod čarou.) Využívá napětím řízené oscilátory a generátory jejichž výstupní signály sčítá, čímž dochází k obohacování spektra o nové spektrální složky. Výsledná barva se skládá z více barev. Ke vzniku aditivní syntézy dochází tehdy, když modifikujeme amplitudu výstupního signálu oscilátoru (VCO) napětím řízeným zesilovačem (VCA) a následně signály sečteme a na výstupu zase modifikujeme dalším společným zesilovačem řízeným generátorem amplitudové obálky EG (Envelope Generator). Pro signály oscilátorů jsou nejčastěji využívány průběhy: harmonické, parabolické, trojúhelníkové, pilové, obdélníkové, ale i náhodné. Jsou využívány i rozdíly fáze.



obr. 1-6: Princip aditivní syntézy[7]

.

1.3.2 Harmonická syntéza

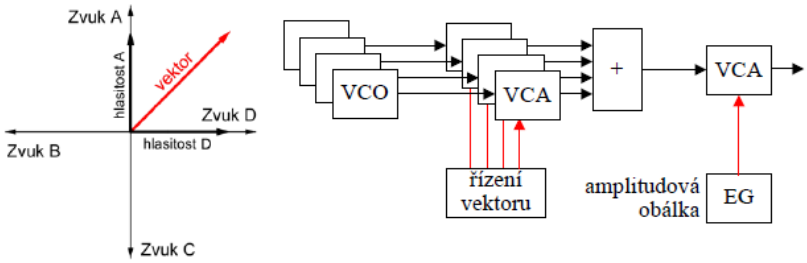
Je speciálním případem aditivní syntézy. Využívá sčítání výstupního signálu harmonických oscilátorů. Je v praxi ale málo využívána, protože využívá pouze vyšší harmonické frekvence. Mnoho akustických nástrojů ale nemá své spektrální složky přesně na vyšších harmonických.

1.3.3 Složková syntéza

Více využívaná je tzv. složková syntéza, která pracuje podobně jako Harmonická syntéza, ale využívá součtu spektrálních složek, které rozlišuje na základě hudebních intervalů. Příkladem jsou například Hammondovy varhany (pozn. pod čarou).

1.3.4 Vektorová syntéza

Využívá změny vektoru hlasitosti, čímž dochází ke změně vzájemného poměru současně znějících zvukových zdrojů[5][7]

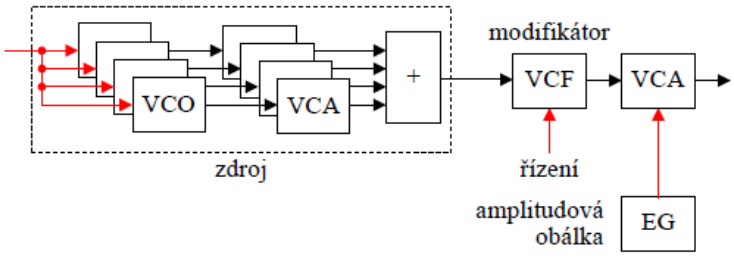


obr. 1-7: Vektorová syntéza [7]

1.3.5 Subtraktivní syntéza

Subtraktivní syntéza nazývána též někdy rozdílová nebo extraktivní vytváří zvuk omezením spektra o určité harmonické složky. Toho docílí pomocí filtrů. Základem je tedy zdrojem (VCO) vygenerovaný signál bohatý na vyšší harmonické spektrální složky a to například: pilu, čtverec, trojúhelník, puls. Takový signál dále prochází přes tak zvaný *modifikátor* což je napětím řízený filtr (VCF), skládající se nejčastěji z filtrů typu: dolní a horní propust, pásmová propust a zádrž, dále pak vrubový (*notch*) filtr, hřebenový filtr a dalších speciálních filtrů. Parametry modifikátoru jsou ovlivňovány hráčem, jenž pomocí tzv. kontrolérů (pozn. P. č) tyto parametry mění, čímž dosahuje změny barvy zvuku v reálném čase. Parametry filtrů jsou typ filtru a řád filtru ty se nejčastěji označují *Type* mezní střední/kmitočet označován jako *Cutoff* a mezní střední kmitočet označovaný jako *Resonance*. Pokud se zvyšuje činitel jakosti filtru, může dojít porušením podmínek stability a rozkmitaní filtru. Filtr se poté stane v podstatě oscilátorem, čehož se často využívá. Nejčastěji se využívá ovlivňování filtru amplitudovou obálkou. Příkladem mohou být například analogové syntezátory Minimoog, MicroKorg nebo Korg MS-20. Existuje i mnoho VST[2] pluginů coby digitálních syntezátorů. Jedním z nejpoužívanějších je asi Massive od Native Instruments.

[2] VST (Virtual Studio Technology) Softwarové rozhraní pro komunikaci mezi hostitelským programem a zásuvnými moduly, kde tyto moduly slouží k úpravě a generování digitálního audio signálu.



obr. 1-8: Blokové schéma subtraktivní syntézy [7]

1.3.6 Modulační syntézy (princip)

Modulační syntézy jsou postaveny na principu modulace signálů, kdy je nosný signál (angl. carrier) ovlivňován signálem modulačním (angl. Modulator). V závislosti na typu syntézy se liší princip ovlivňování signálu. Obecně rozlišujeme tyto typy modulační syntézy:

- frekvenční (FM)
- amplitudová (AM)
- kruhová (RM)
- parametrická
- PWM

Během modulačních typů zvukové syntézy vznikají nové harmonické složky s klesající amplitudou, tzv. postranní pásma. Ta jsou typická úměrnosti součtů a rozdílů přítomných frekvencí. Počet těchto postranních pásem ovlivňuje poměr těchto frekvencí a míru modulace tzv. modulační index

1.3.7 Modulační AM syntéza

AM syntéza pracuje na principu amplitudové modulace. To znamená, že okamžitá hodnota (amplituda) nosného signálu je ovlivňována modulačním signálem. Při nízkých hodnotách modulace se projevuje změna signálu houpavou změnou amplitudy, ale při vyšších hodnotách začíná docházet k obohacování spektra o nové spektrální složky.

1.3.8 Modulační RM syntéza

RM (Ring Modulation) syntéza je speciálním případem AM modulace. Kruhová se jmenuje podle zapojení kruhového diodového modulátoru. V digitální technice je ale tento modulátor realizován násobičkou. Výstupem RM modulace je signál s potlačenou nosnou (základní) frekvencí, ale se zachovanými postranními pásmy.

1.3.9 Modulační FM syntéza

FM syntéza pracuje na principu frekvenční modulace, kdy je frekvence nosného signálu ovlivňována frekvencí signálu modulačního. FM syntéza je v principu realizována pomocí dvou oscilátorů, kdy jeden generuje nosný signál a druhý generuje modulační signál. Tato dvojce oscilátorů bývá nazývána

operátor. Pokud máme více operátorů, můžeme je spojovat mezi sebou do tzv. *algoritmu,* toto zapojení může být paralelní nebo sériové. Při paralelním zapojení řídí jeden operátor amplitudu modulátoru druhého a při sériovém zapojení řídí jeden operátor kmitočet operátoru druhého. Mezi prvními syntezátory nabízejícími algoritmizaci FM syntézy byl například *Yamaha DX7.* Pomocí FM syntézy jde vytvořit nepřeberné množství zvuků, protože „*teoretická šířka spektra je nekonečná, počet vnímaných postranních pásem je ale omezen a narůstá s indexem modulace"* (*citováno z* [7]). Výhodou je, že díky algoritmizaci je tedy možné vytvářet složitá a bohatá spektra signálu, přitom je celý proces v celku nenáročný na výpočet. Nevýhodou je, že se nedá s určitostí předvídat konkrétní zvukový výsledek. Typické pro FM syntézu jsou perkusivní, kovové, zvonivé či jiné neharmonické zvuky.

1.3.10 Modulační PWM syntéza

Při ní se moduluje šířka impulsu obdélníkového průběhu. Protože je spektrum obdélníkového průběhu závislé na jeho střídě, je výsledkem zmodulovaný signál s časově kolísajícím poměrem vyšších harmonických frekvencí a časově proměnným spektrem. V závislosti na vlastnostech oscilátoru je možné využít šířkovou modulaci i pro jiné průběhy.[7]

1.3.11 Číslicová syntéza

Pracuje s tzv. DCO (Digitally Controled Oscilator) což je obdoba VCO. Na výstupu je průběh tohoto oscilátoru nejčastěji modelován pomocí dvou speciálních technik. První je matematický model, kdy se používá přímý výpočet požadovaného průběhu, nebo rekurzivní model. Druhou technikou je tzv. tabulková technika, kdy se průběh ukládá do paměti *Look-up, Table* a pomocí rychlosti čtení vzorků z pamětí se nastavuje frekvence signálu a tím pádem i výška tónu. Jelikož při generování periodických signálů s nekonečným spektrem (např. pila, obdélník) dochází k aliasingovému zkreslení, byly vyvinuty i speciální techniky pro generování těchto signálů.
Jsou to:

- **technika disktrétního součtového vzorce**, kdy se využívá goniometrických funkcí.
- **technika derivace parabolického průběhu**, jejich výhodou je nižší aliasingové zkreslení.
- **technika postupné integrace pásmově omezeného sledu impulsů** jejich generování je výpočetně náročnější.

Při práci s číslicovou syntézou můžeme docílit tzv. **tvarovací syntézy,** kdy se je využíváno různého pořadí čtená vzorků z paměti. Rozlišujeme **nelineární tvarování** a **lineární tvarování.** Při nelineárním tvarování se za DCO zařadí nelineární prvek. Nelineární převodní charakteristika způsobí nelineární zkreslení a vznik nových vyšších harmonických složek. Při lineárním tvarování se pomocí DCO určuje index odečítání z paměti, v níž je uložen průběh nelineární přenosové funkce.

1.3.12 Matematické a fyzikální modelování

Zde se vytváří věrné simulace hudebních nástrojů. Vychází se z fyzikálního modelu určitého hudebního nástroje, přičemž se zaměřuje na interakci mezi zdrojem zvuku a rezonátorem určitého nástroje. Rozlišují se nástroje vydávající tlumené kmity a nucené kmity dále se pak bere zřetel na materiál, z kterého je nástroj vyroben atd. Díky fyzikálnímu a matematickému modelování jsme schopni simulovat i tzv. transienty (přechodové jevy), které určují věrnou barvu zvuku simulovaného nástroje. Toho nejsme schopni pomocí jiných syntéz dosáhnout. Výhodou tohoto modelování jsou i nepřeberné možnosti nastavování simulovaných nástrojů. Jsme například schopni vytvořit kytaru s velmi dlouhým hmatníkem a struny o tloušťce několika desítek cm.

1.3.13 Speciální typy syntéz

Existuje velké množství syntéz zvukových signálů, specializujících se na konkrétní typ zvukového signálu. Například **fázový vokodér,** který využívá dělení signálu na jednotlivá pásma, která pak analyzuje a umožňuje tak jejich zpracovávání. Jsme schopni pak měnit výšky tónu beze změny délky signálu nebo naopak časovou expanzi a kompresi beze změny výšky. Dalším typem je například **granulární syntéza**. Ta pracuje s tzv. grány, což jsou zvukové vzorky dlouhé 20 až 30 ms (nerozeznatelné lidským sluchem), jež umísťuje za sebou nebo individuálně ve stereo bázi a vytváří tak výsledný zvuk. **LPC syntéza** využívá zase křížové syntézy a určitých filtrů k napodobování lidského hlasu.

V této kapitole bylo čerpáno z těchto zdrojů: [7], [5], [9], [10], [11]

1.4 Efekty

Součástí mé aplikace budou efekty, které rozšíří možnosti úpravy výsledných zvuků. Využívány budou především efekty provádějící filtraci zvukových signálů a kmitočtové korektory (equalizery a filtry), dále pak efekty s proměnným zpožděním (Delay/Echo, Reverb), potom i efekty využívající modulaci i demodulaci zvukového signálu a efekty měnící výšku tónu (Vocoder) a nakonec zkreslovací efekty (Distortion).

1.4.1 Kmitočtové korektory a filtry

Jde o efekty, které ovlivňují frekvenční charakteristiky zvukových signálů. Používají se i ke změně barvy zvuku případně potlačení či zesílení některých frekvencí, které nejsou potřeba. Pokud jde o filtry, rozlišujeme tyto základní typy: *low pass* nebo také *high cut* je typ filtru potlačující vysoké frekvence *high pass* nebo také *low cut* je typ filtru potlačující nízké frekvence, *band pass* neboli pásmová propust ořezává vysoké i nízké kmitočty a propouští vysoké. Další typy filtru jsou *shleving, bell* a *notch*. Důležitými parametry filtrů shelving jsou mezní kmitočet a velikost zesílení, u filtrů bell

střední kmitočet, zesílení a jakost filtru *Q*. Kombinacemi filtrů typu shelving a bell vzniknou různé typy kmitočtových korektorů k úpravě zvukového signálu.

Jsou to především:

- *Grafický ekvalizér* – kombinace filtrů typu bell s nastavitelným ziskem nebo útlumem
- *Parametrický ekvalizér* – kombinace většinou tří až čtyř filtrů typu shelving a bell, pro nízké kmitočty low shelving, pro vysoké high shelving a pro střední filtry typu bell. U tohoto typu ekvalizéru je možné nastavovat zisk i útlum, mezní nebo střední kmitočet a šířku pásma.

1.4.2 Efekty s proměnným zpožděním

Jsou to efekty využívající zpožďovací linku k následnému zopakování průchozího signálu. Principiálně lze tyto efekty rozdělit na efekty s konstantním zpožděním a proměnným zpožděním. První z nich lze dělit ještě v závislosti na délce zpoždění. Zpoždění pod 25 ms je využíváno u efektů *phase shifter* nebo *resonator*. Zpoždění mezi 25 a 50 ms je využíváno u *slapback echo* nebo *doubling* . Zpoždění nad 50 ms, které je lidským sluchem vnímáno jako ozvěna, využívají efekty *echo, tap-delay* atd. Technicky jsou dnes řešeny efekty tohoto typu především v digitální formě. Obecná funkce efektů tohoto typu je následující. Vstupní signál je v přímé větvi násoben určitou konstantou a poté sečten se signálem zpožděným o určitý počet vzorků. V podstatě jde o univerzální hřebenový filtr. Dále existují efekty simulující dozvuk, k čemuž využívají zpožďovací linky. Respektive požadovaného efektu se dosahuje kombinacemi několika spojených zpožďovacích linek a filtrů. Nejpoužívanějším takovým efektem je *reverb* v různých jeho obměnách, jako např. *hall* reverb, *church* reverb, *room* reverb atd.

Ve své aplikaci využívám především efekty echo/delay a reverb. Parametry efektů delay a echo jsou *delay time* určující délku mezi opakováními a *feedback*, určující počet nebo úroveň opakování. Parametry efektu reverb jsou *reverb time*, což je čas dozvuku, *wet/dry*, tedy poměr mezi procesovaným a neprocesovaným signálem. Dále pak to jsou parametry *cut off frequency* určující spektrum dozvuku a *room size* udávající velikost prostoru modelu simulace.

1.4.3 Modulační efekty

Princip modulačních efektů vychází z principů modulační syntézy zvukových signálů a ty jsou popsány v předcházející kapitole. Příchozí signál do efektu je většinou nosný, ale může být i modulační. Jsou zde využívány všechny typy modulační syntézy. Nejznámějšími a nejpoužívanějšími efekty jsou *flanger, chorus, phaser*. Poslední zmiňovaný využívám ve své aplikaci, a proto obecně popíšu jeho funkci. Phaser je realizován pomocí několika v sérii zapojených širokopásmových filtrů řízených nízkofrekvenčním oscilátorem. Je zde využit principů hřebenového filtru, kdy se mění ořezávací frekvence a rychlost její změny. Parametry efektu jsou *depth* (míra

změny), *wet/dry* což je poměr mezi čistým a modifikovaným signálem a parametr *feedback* udává množství signálu zpětnovazební větve.

1.4.4 Efekty měnící výšku tónu

Efekty v této skupině jsou řešeny rozdílnými technickými postupy. V mé aplikace je využit především efekt *vocoder,* což je efekt určený k úpravě lidského hlasu. Nicméně technických řešení vokodéru je také několik. Já využívám tzv. *kanálový vokodér* na principu adaptivní filtrace. Při této realizaci vokodéru je signál rozdělen do určitých kmitočtových pásem pomocí filtrů a dále se využívá odhad amplitudy pomocí sledovače obálky.

1.4.5 Zkreslovací efekty

Tato skupina efektů je založena na nelineárním zpracování zvukového signálu, díky čemuž dochází k obohacení spektra o nové spektrální složky a jejich kombinace. Signál, který prochází systémem s nelineární charakteristikou je ořezáván, prahován či usměrňován tak, až je dosaženo požadovaného spektrálního obohacení. Charakter tohoto obohacení je dán tvarem převodní charakteristiky v oblasti vymezené maximální a minimální hodnotou vstupního signálu. Obecně lze říci, že čím ostřejší změny jsou právě ve tvaru převodní charakteristiky, tím větší počet harmonických frekvencí se zvyšující se amplitudou bude ve spektru. Při pozvolnějších změnách charakteristiky bude docházet ke snižování amplitudy spektrálních složek a vzdálenost mezi nimi se bude blížit hudebním intervalům. Do této skupiny patří efekty jako *overdrive, distortion, fuzz, booster, metalizer* atd.

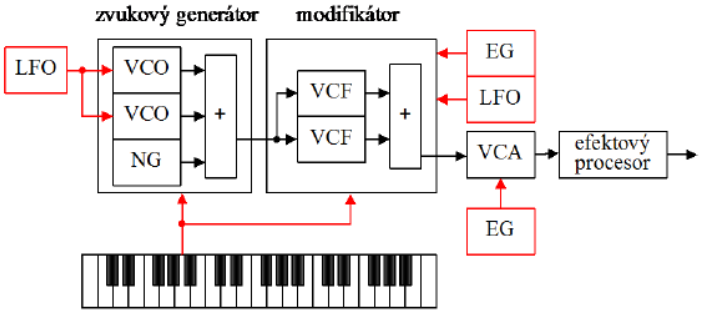
1.5 Elektronické hudební nástroje

Aplikace, coby experimentální hudební nástroj, bude kombinovat několik funkcí různých elektronických a hudebních nástrojů. V této kapitole bych se rád zmínil o těch nejdůležitějších, které budu využívat. Za elektronický nebo elektroakustický hudební nástroj (elektrofon) se považuje takový hudební nástroj, jež pomocí elektrické energie generuje tón nebo jej převádí z akustické oblasti, zesiluje a pak dále zpracovává.[2]

obr. 1-9: Příklad moderního virtuálně analogového syntezátoru

1.5.1 Syntezátory

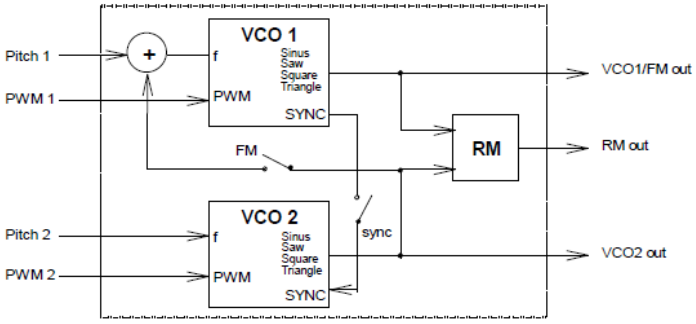
Syntezátor je elektrický hudební nástroj, který využívá syntézy zvukových signálů k vytváření tónů. Základním syntezátorem je analogový syntezátor a ten se většinou skládá z těchto základních bloků: zvukového generátoru, modifikátoru, efektového procesoru a klaviatury. Místo klaviatury se využívají i tzv. kontrolery (spínací prvky). Kontrolerům se mohou přiřadit různé parametry nebo skupiny parametrů, které mohou řídit. Syntezátory bývají rozšířeny ještě o rozhraní MIDI (Music Instruments Data Interface)[3].



obr. 1-10: Blokové schéma analogového syntezátoru [7]

1.5.2 Zvukový generátor

Využívá většinou aditivní syntézu a kombinuje ji s PWM, RM a FM syntézami. Skládá se s několika VCO oscilátorů, z nichž jeden většinou vytváří šum a ostatní generují periodické signály, nejčastěji pilu a obdélník. Oscilátory mohou pracovat synchronně. Stiskem klávesy se generuje požadovaný kmitočet. Signál z klávesy určuje výšku tónu a šířku impulsu při PWM modulaci. Pomocí LFO lze vytvořit efekt vibrato. Výstupní signály jsou přivedeny na sumační zesilovač *Pre-Filter mix*, kde pokračují dále do modifikátoru.



obr. 1-11 Struktura bloku zvukového generátoru [7]

[3] MIDI (Music Instruments Data Interface) je mezinárodní standart používaný jako elektronický komunikační protokol, který dovoluje elektronickým hudebním nástrojů, počítačům a dalším zařízením, komunikovat v reálném čase prostřednictvím sériového rozhraní.

1.5.3 Modifikátor

Modifikátor je banka paralelně řazených VCF filtrů, která provádí filtraci spektra. Jednotlivé filtry lze volit přepínačem nebo je případně úplně vyřadit. Signál poté pokračuje na sumační zesilovač *post-filter mix,* dále pak na VCA a na výstup nebo do efektové jednotky.
Jednotlivé typy syntéz lze řídit řídicími obvody syntezátoru.

1.5.4 Řídicí obvody syntezátoru

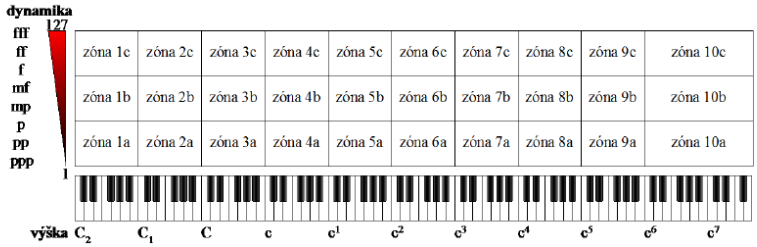
Řídící obvody syntezátoru jsou v podstatě generátory řídících signálů, z nichž se využívají LFO nízkofrekvenční oscilátory a EG generátory obálek. Pomocí LFO lze generovat základní průběhy, měnit šířky impulsů, ale také využívat tzv. *Sample & Hold* sekvence, jež se mění v závislosti na rytmu a vytváří tak novu barvu. Pomocí generátorů obálek se řídí: *Pitch Envelope Generator* (PEG) - kmitočet oscilátoru, *Filter Envelope Generator* (FEG) - filtr mezního kmitočtu a *Amplitude Envelope Generator* (AEG) což je průběh amplitudy výstupního signálu. Samotná obálka určuje charakter průběhu. Tzv. ADSR je nejpoužívanějším typem obálky a je určena těmito parametry: *Attack time* to je doba náběhu signálu (např. hlasitosti) na maximum. *Decay time* je doba sestupu na ustálenou hodnotu. *Sustain Level* je doba (např. držení klávesy), po kterou se signál drží na ustálené hodnotě. *Release Time* je doba (po puštění klávesy), za kterou signál klesne na nulu. Existují ADSR obálky se zdvojenými parametry *Decay* a *Release* nebo dvouparametrové obálky typu *Depth / Decay*. Moderní syntezátory umožňují také volit, jestli budou náběhy a poklesy lineární, logaritmické nebo exponenciální.
Efektový procesor se řadí za syntezátory (řeč je v tomto případě zejména o virtuálních syntezátorech) a skládá se ze čtyř bloků jdoucích za sebou:

- Variačních efektů, což jsou tvarovače signálu (*Distortion, Overdrive, Fuzz*).
- Bloku equalizačního, který se skládá z filtrů typu *HP (horní propust)* a *DP (dolní propust),* nebo grafického či paragrafického equalizéru.
- Bloku modulačních efektů např. *Flanger, Chours, Phaser.*
- Na konci bývá zařazen blok efektů se zpožďovací linkou a směřováním ve stereobázi např. *Deley, Echo, Auto-Panner atd.*

1.5.5 Samplery

Sampler je elektronický hudební nástroj, který přímo nevytváří zvuky, ale v závislosti na stisku klávesy přehraje navzorkovaný zvukový signál, v podstatě tón uložený v paměti. Základem generovaného zvuku je tedy vzorek signálu, nebo zvuku nazývaný též sampl. Sampler disponuje paměťovým médiem, kde jsou jednotlivé vzorky (samply) uloženy a při stisku klávesy jsou z této paměti přehrávány. Každý sampl má u sebe informaci o délce vzorku o odpovídající notě. Po stisku klávesy se nota klávesy porovná s notou samplu, a pokud je vyšší nebo nižší, přetransponuje se na

požadovaný tón. Transpozice se může provádět několika způsoby. Můžeme změnit vzorkovací rychlost přehrávání a docílíme tak požadovaného tónu ale změníme délku signálu. Tato metoda je ale nevhodná pro polyfonní přehrávání, protože pokud zahrajeme několik tónu současně, tak každá nota vyžaduje jinou vzorkovací rychlost. Vhodnější pro polyfonní systémy je metoda **převzorkování signálu**, kde se využije decimace a interpolace signálu a dosáhne se tak požadované transpozice, opět ale dojde ke změně délky signálu. Lepší variantou než převzorkování signálu je tzv. *Pitch shifting,* kdy se signál pomocí časové komprese a expanze roztáhne nebo stlačí v čase a následně převzorkuje. Tato metoda je ale výpočetně velmi náročná. Nejefektivnější je metoda **změny výšky tónu pomocí modulace zpožďovací linky.** Signál se rozdělí na malé části a posílá se do dvou, nebo více zpožďovacích linek. Linky jsou řízeny pilovým signálem. Fáze signálu jsou fázově posunuty a sečteny na výstupu linky pomocí kombinační funkce. Pokud je rychlost čtení ze zpožďovací linky rychlejší než zápis, tón se zvýší a pokud je to naopak rychlost se sníží. U všech zmíněných metod však dochází během zpracování signálu ke změně spektra a následně i ke změně barvy tónu. Změna tónu je však v určitých případech nežádoucí. Aby se tomuto jevu zabránilo, umožňují samplery rozdělit klaviaturu do určitých zón. Rozdělení zón podle výšky tónu se nazývá horizontální zónování a rozdělení podle dynamiky se nazývá vertikální zónování. V rámci horizontálního zónování se využívá výše zmíněných metod ke změně tónu a v rámci vertikálního zónování se využívá zesílení během přehrávání. Každé zóně odpovídá určitý vzorek. Organizace zón bývá různá, ale většinou má při hře prioritu horizontální zóna. Zóny se mohou i překrývat.[7]

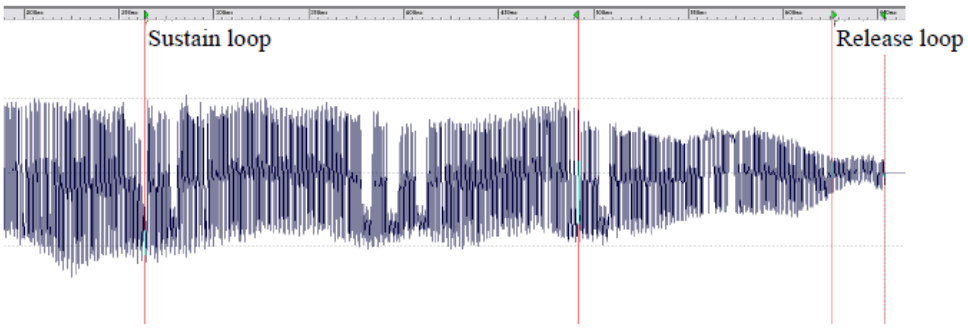


obr. 1-12: Kombinované zónování (Horizontální a - b, vertikální 1 - 10)[7]

1.5.6 Looping

Důležitou vlastností samplerů je tvorba a přehrávání smyček tzv. Looping. U tzv. *one-shot* zvuků dojde při jednom zmáčknutí klávesy k jednomu přehrání samplu. Často je však potřeba aby sampl zněl po dobu stlačení klávesy nebo aby měl určitý dozvuk, např. sampl klavíru. K těmto účelům slouží smyčka (loop), což je určitý usek požadovaného zvukového signálu, který se přehrává stále dokola. Smyčce, která je stále přehrávána po dobu držení klávesy se říká *Sustainig Loop.* Když je klávesa spuštěna přehrávání se hned nepřeruší. Po posledním přehrání smyčky se začne se přehrávat smyčka *Realease Loop,* což je doznívání. Dobu doznívání této smyčky určuje parametr *Release time.* Po tuhle dobu se smyčka dozvuku přehrává a hlasitost klesá na nulu. Při tvorbě smyček je důležité dbát na to,

aby na sebe začátek a konec smyčky plynule navazovali. Tzn. pokud nám smyčka například končí v nule zápornou půlvlnou, musí začínat v nule kladnou půl vlnou.



obr. 1-13: Zvukový soubor s definovanými smyčkami Sustain a Release[7]

Všechny samplery umožňují jejich výstupní signál při přehrávání dále zpracovávat pomocí efektových procesorových jednotek nebo pomocí napětím řízených zesilovačů či Filtrů. Všechny tyto vlastnosti i nastavení zón a použité vzorky lze nastavit do takzvaných *programů* je sdružovat do skupin tzv. *patchů* v nadřazených multiprogramech. Většina samplerů disponuje MIDI rozhráním, pomocí něhož se k nim připojují tzv. *midikontrolery*, což jsou kombinace klaviatury, multipadů ([4]) a dalších ovládacích prvků, kterými lze ovládat jak různé Programy a multiprogramy tak samotné samply. Mezi nejznámější samplery patří např. Akai S 1000 nebo Roland S-550. Dnes však tyto samplery používají jen nadšenci. Jako samplery se využívají komplexní počítačové systémy a sample banky např. Native isntruments Kontakt.[7]

a) b) c)

obr.1-14: Příklady moderních samplerů a) Korg elecrtibe sampler, b) Korg kaoss pad KP3+ c) Native instruments Kontakt

V této kapitole bylo čerpáno z těchto zdrojů:[5], [7], [9], [10], [11]

[4] Mulipady nebo Pady jsou ovládací prvky nemálo podobné klávesám. Bývají pogumované, protože se využívají i jako elektronické bicí nástroje. Multipadům lze přiřadit určité hodnoty, které mají spouštět, většinou se jedná o přehrání samplu.

1.6 Operační systém Android

Andorid je operační systém určený převážně pro mobilní zařízení jako jsou smartphony tablety apod. V současnosti se začínají využívat i pro jiná zařízení například multimediální centra (Google TV) auta a jiné. Historie Androidu sahá někdy před rok 2005. Tehdy odkoupila společnost Google malou firmu Android Inc, (založenou Andy Rubinem, Nickem Searsem a Chrisem Whitem), která vyvíjela operační systém pro mobilní telefony, který měl být více uživatelský, přívětivý a zároveň soběstačnější. V roce 2007 vzniklo konsorcium Open Handset Aliance, které kromě Googlu zahrnovalo i mnoho dalších společností, zabývajících se výrobou mobilních telefonů, (např. Samsung,LG,HTC,Qualcom, Nvidia). Cílem tohoto společenství bylo vyvinout multiplatformní operační systém, který by běžel na různých zařízeních. Google splnil svůj cíl a v roce 2008 přišel s Androidem 1 a v roce 2009 s androidem 1,5(cupcake). Dnes je Google s Androidem hlavním konkurentem Apple IOs a je používám na tisících různých zařízeních a platformách.

1.6.1 Struktura operačního systému Android

Operační systém android je sice primárně určen pro koncové uživatele, ale některé jeho části jsou do určité míry open source [5]. Vývojářům tak nabízí efektivní nástroj pro vývoj aplikací *SDK (Software Development Kit)*. Struktura operačního systému android se skládá z několika vrstev. Nejnižší vrstvou je samotné jádro systému, jež je založeno na jádře Linux[6].Vlastnosti Linuxu jsou využity zejména pro správu paměti a sítí, řízení procesů a v neposlední řadě, kvůli přenositelnosti na různá zařízení. Tato vrstva zprostředkovává komunikaci mezi hardwarem a další nadřazenou vrstvou. Další vrstvou jsou systémové knihovny, které obsahují základní funkce a vlastnosti systému. Vývojářům jsou poskytnuty prostřednictvím Android Application Framework. Nacházejí se zde např. knihovny pro podporu multimédií, webového prohlížeče, vykreslování grafiky apod. Třetí vrstvou je tzv. Android Runtime. Jeho součástí je virtuální stroj, ve starších verzích Dalvik v novější pak *ART (Android RunTime)* úkolem vrstvy je překládat aplikace do Byte kódu, která jsou pak dále zpracovávána virtuálním strojem. Čtvrtou vrstvou je Application Framework. Pro vývojáře je nejdůležitější, poněvadž poskytuje přístup ke službám, které se pak dále využívají v aplikacích. Služby umožňují např. přístup k uživatelskému rozhraní, upozorňovací stavový řádek, možnost práce se soubory apod. Pátá a poslední vrstva je Aplikační vrstva. Jejím úkolem je zprostředkovávat chod všech aplikací a to předinstalovaných i těch stažených s Google marketu[7].

V této kapitole bylo čerpáno z těchto zdrojů:[12], [14], [15], [16]

[5] Open source nebo také Open source software je počítačový software s otevřeným zdrojovým kódem. Otevřenost zde znamená jak technickou dostupnost kódu, tak legální dostupnost licencí.
[6] Linux je open source operační systém.
[7] Google Market nebo také Google play je internetový obchod s aplikacemi pro operační systém Android

1.7 Pure Data

Svou aplikaci, jakožto experimentální hudební nástroj, jsem se rozhodl naprogramovat pomocí programovacího jazyka PureData. V této kapitole stručně uvedu, o jaký programovací jazyk se jedná, a v kostce vysvětlím platformu, která realizuje grafické prostředí pro Pure Data na Androidu.

Pure Data (dále jen PD) je open source vizuální programovací jazyk, který vyvinul Miller Puckette. Historie tohoto jazyka sahá až do roku 1985. Tehdy Puckett pod hlavičkou institutu ICRAM zveřejnil vizuální programovací nástroj MAX, který se přerodil do dnešního známého komerčního vizuálního programovacího nástroje Max/MSP. Pure Data byla vyvinuta v devadesátých letech minulého století, původně jako zjednodušená veřejně dostupná obdoba jazyka MAX, licencovaná zcela zdarma. Dnes už má ale tento jazyk obrovskou komunitu po celém světe a je neustála vylepšován. PD jsou sice převážně využívána hudebníky a výtvarníky, ale jsou vhodná i pro vývojáře a výzkumníky. Nabízí jim možnost realizovat svůj software bez psaní souvislého kódu. PD jsou totiž, stejně jako Max/MSP, programovací jazyky založeny na tzv. DFP (Data Flow Programming), což je metoda programování, která vizuálně popisuje tok dat.

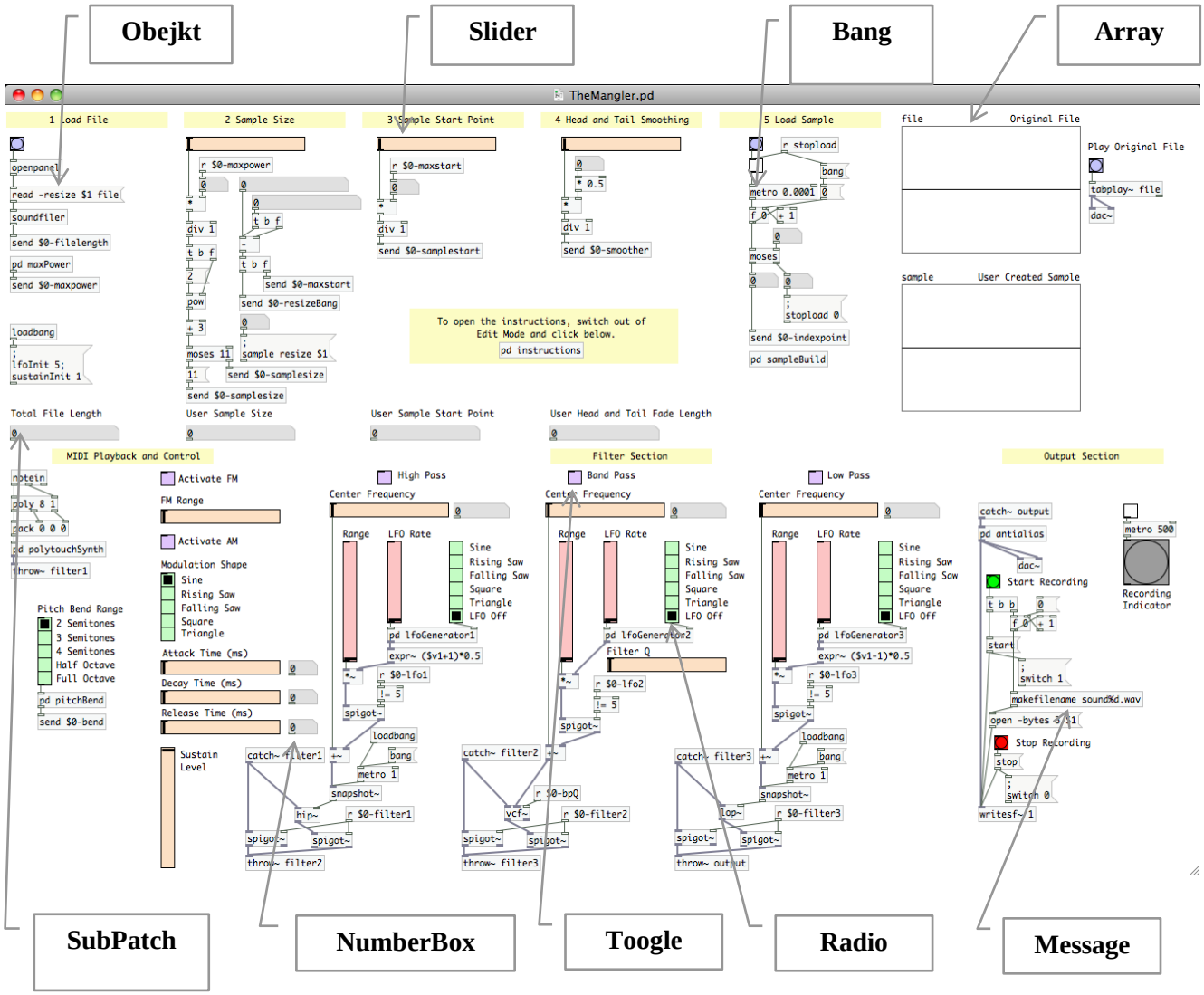
PureData se využívají pro zpracování a vytváření zvuku, videa, 2D/3D grafiky dále např. pro nastavení senzorů, čidel a rozhraní např. MIDI. Mohou snadno pracovat po síti a řídit tak např. osvětlení, pohony atd. PD je nástroj vhodný jak pro zpracování multimédií a jednoduché řízení malých systémů, tak i pro realizaci komplexních systémů pro velké projekty. Výhodou PD je například i to, že jsou v podstatě multiplatformní, mohou totiž pomocí různých GUI platforem a frameworků běžet na mnoha operačních systémech, např. IOs, Android, Windows a zařízeních jako např. PC, Raspberry Pi, MAC, ale i na smartphonech a tabletech.

Když Puckett uvolnil PD k volnému šíření, uvolnil s nimi i kompletní zdrojové kódy. Mnoho nadšenců mělo tak možnost přetvářet si PD k vlastním účelům. V důsledku toho se PD rozdělila do dvou základních větví. Verze PD vannila je udržována samotným Puckettem a obsahuje jádro systému. Větev PD extended, která je udržována komunitou, v sobě obsahuje už i verzi vannila a mnoho dalších vylepšení. PD jsou postaveny na C a C++, ale extended verze může obsahovat i tzv. abstraktní prvky postavené na jiných jazycích.

1.7.1 Základní struktura PD

Základními stavebními kameny programů v PD jsou určité prvky. Hlavními prvky pro vykonávání algoritmických operací a funkcí jsou tzv. objekty. Pomocí nich se programují elementární povely, příkazy a řídící algoritmy. Konkrétní funkci objektu definujeme tak, že do něj zapíšeme povel, který je přiřazen určité funkci. Například povel *adc~* přivede signál ze vstupu zvukové karty. Obecně lze definování objektů přirovnat k implementaci funkcí v jiných jazycích. V následujícím odstavci budou popsány ostatní prvky. Zpráva (*Message*), která slouží k posílání a ukládání informací. Číslo

(*NumberBox*) zase reprezentuje číselné nastavení určitých hodnot, umí odeslat zprávu a při každém výstupu pošle událost bang. Kromě jednoduchého numberboxu existuje i rozšířený numberbox, u kterého lze definovat i lineární nebo logaritmická řada. Symbol (Symbol) slouží jako textový vstup pro zápis textu. Komentář (Comment) využíváme pro jednoduché popisování našich patchů. *Bang* je určitá simulace tlačítka, při jeho stisknutí nebo přijmutí jiného bangu odešle událost bang, což je vlastně jakákoli hodnota. Bang slouží hlavně ke spouštění určitých procesů.



obr. 1-15: Příklad hotového PD patche s popisky

Přepínač *Toogle* je zase simulace přepínače dvou hodnot. V základním nastavení jsou to 0 a 1, ale lze nastavit i jiné hodnoty. Posuvníky Slidery jsou simulace Faderů neboli potenciometrů. Odporovou dráhu simuluje interval čísel, který je předem nastavitelný. Přepínače Radio umožňují přepínat mezi více hodnotami. Rozsah hodnot lze nastavit. Pole Array nebo Table slouží k ukládání dat. K oběma variantám je přiřazen graf, který hodnoty vykresluje. V table je viditelný až po otevření objektu. Všechny tyto prvky jsou umísťovány na tzv. plátna (Canvas) v patchovacím okně (základní okno

vývojářského prostředí). Jednotlivé prvky jsou propojeny tzv. šňůrami (PatchCordy), což jsou čáry reprezentující datové toky. Spojováním objektů různých funkcí s dalšími prvky vytváříme tzv. patche, které vykonávají určité úkoly a mají určité funkce. Funkce a úkoly patchů se mohou lišit v míře složitosti od jednoduchých výpočtů po složité operace jako je např. Fourierova transformace, zpracovávání videa, vytváření algoritmu dozvuku atd. V rámci jednotlivých patchů lze vytvořit i tzv. subpatche a abstrakce, což jsou podprogramy, různé povely a řídící algoritmy. Obdoba tříd a potomků v objektově orientovaném programování. Rozdíl mezi subpatchem a abstrakcí je ten, že abstrakce se musí před použitím uložit v adresáři projektů, ale lze ji pak různě využívat napříč projektem. Subpatch lze opakovaně využívat kopírováním. Nicméně v obou případech se musí dodržovat jedinečnost jednotlivých prvků např. pole atd.

V této kapitole bylo čerpáno z těchto zdrojů:[17], [13]

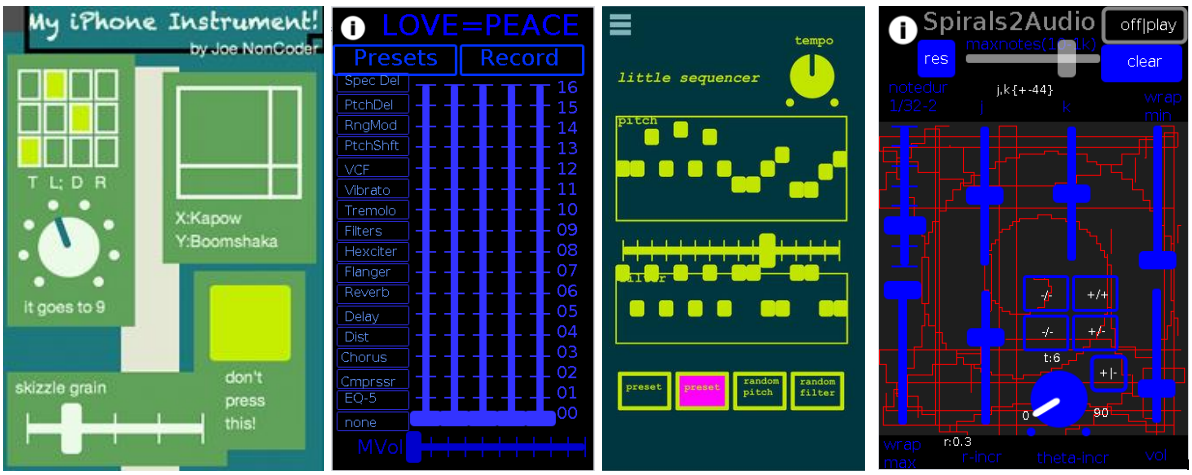
1.7.2 GUI platforma pro PD MobMuPlat

Pro operační systém Android existují platformy, které vytvářejí grafická uživatelská rozhraní a propojují je s PD patchem. Jednou takovou platformou je MobMuPlat (Mobile Music Platform). Ta umožňuje vytvoření grafického rozhraní pro PD patche a jejich spouštění na mobilních zařízeních. Respektive umožňuje funkci PD patchů na operačních systémech Android a iOS. Přičemž dokáže využívat všech vlastností mobilních zařízení, jako jsou gyroskop, shaking, gps, wifi hotspot, komunikace po sítích. Z těchto zdrojů umí přijímat data a ta dále posílat do PD patchů.

MobMuPlat je postavena na libpd což je platforma umožňující vytváření grafických prostředí pro PD na různých operačních systémech. MobMuPlat se skládá z několika částí softwaru: mobilní aplikace (dostupná v Google Play nebo iOS store), editoru pro Windows a iOS (postaven na Jawa Swing) a několika komunikačních protokolů.

Pomocí editoru lze tedy ve Windows vytvořit grafické uživatelské rozhraní, které se pomocí speciálního patche propojí s vytvořeným PD patchem. Ve smartphonu se po té spustí nainstalovaná aplikace MobMuPlat, kde se vytvořený patch i s grafickým rozhraním načte a poté spustí. Grafické prvky jsou podobné jako některé prvky v PD. To platí zejména u posuvníků **Slider** (Slider v PD), tlačítek **Button** (Bang v PD) a přepínačů **Toogle** (Toogle v PD), s tím rozdílem, že hodnoty jsou v základu nastaveny na 0 až 1 nebo 0 a 1 u toogle a buttonu, nicméně hodnoty lze přenastavit. Dalšími prvky jsou otočný knoflík **Knob,** což je obdoba slideru v podobě otočného potenciometru. **XY slider** je plocha pro přijímání dat v podobě souřadnic x a y z určité plochy displeje. **Multitouch** je obdoba XY slideru s podporou vícenásobného dotyku. **Multislider** je kombinace několika sliderů v jednom, respektive vedle sebe. Jejich počet lze nastavit. **Grid** je mřížka M x N, jež může být postavena z Buttonů nebo tlačítek Toogle. **Table** slouží k vykreslování dat z pole array nebo table. Je zde možno nastavit režim označování určité části průběhu nebo režim překreslování. **Label** slouží k popiskům,

ukládání a přijímání textu. **Menu** umožňuje výběr z několika možností. **LCD** slouží k vykreslování jednoduchých obrazců a **Panel** umožňuje zobrazování obrázků.



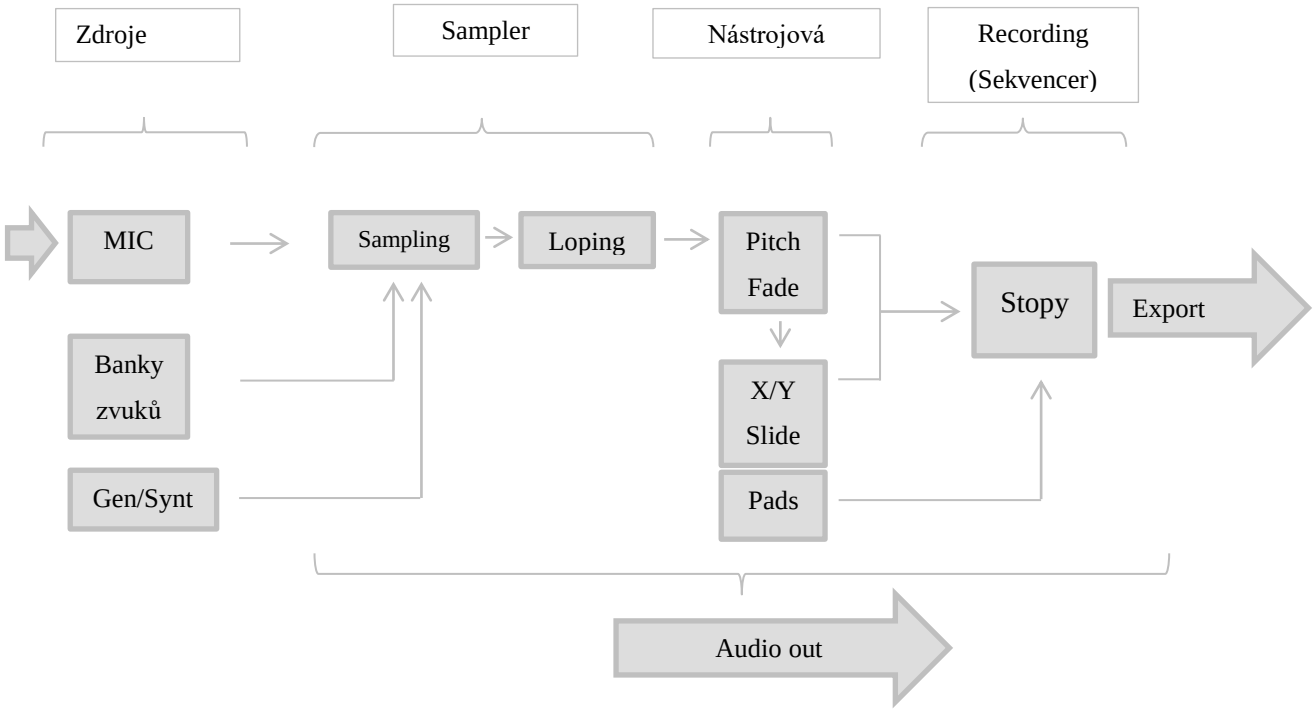
obr. 1-16: Příklady grafického prostředí MobMuPlat.

V této kapitole bylo čerpáno z těchto zdrojů: [18],[20]

2 Návrh

Aplikaci budu programovat pomocí jazyku Pure Data, v nich vytvořím tzv. audio engine. Následně vytvořím GUI rozhraní v MobMuPlat Editoru (PD Grafická platforma pro android). Tím vznikne patch (což bude vlastně GUI aplikace postavená na PD), který se pak spustí ve smartphonu v aplikaci MobMuPlat. Tento patch bude možné nastavit i jako standalone aplikaci, která pak půjde spouštět i mimo aplikaci MobMuPlat. V této kapitole uvedu, jakým způsobem by aplikace měla fungovat.

Základní blokové schéma ukazuje strukturu a funkce aplikace. Pro názornost jsem schéma rozdělil do čtyř sekcí, které obsahují bloky s různými funkcemi. Funkce sekcí jsou následující. Sekce zdroje zvuku je určena k výběru jednoho ze tří zdrojů zvuku a jeho následného poslání do sekce samplovací. Ta je určena k vytvoření zvukového vzorku a smyčky (Loop), ve které je pak zvuk přehráván. Přehrávaná smyčka následně pokračuje do nástrojové sekce, kde je možné měnit výšky zvuku/tónů, filtrovat signál v reálném čase a přidávat beaty pomocí bicího automatu. Nakonec je zvuk směrován do nahrávací/sekvenční větve, kde půjde jednotlivé smyčky ukládat do stop (jejich počet bude omezený), vrstvit je na sebe, vytvářet sekvence a následně je vyexportovat jako jednu skladbu. Jednotlivé stopy bude možné i editovat a přehrávat. Ve všech sekcích kromě sekce zdroje zvuku bude možné zapnout nebo vypnout odposlech pro monitoring právě zpracovávaného zvuku. Jednotlivé sekce následně podrobněji popíšu



obr. 2-1: Funkční blokové schéma aplikace

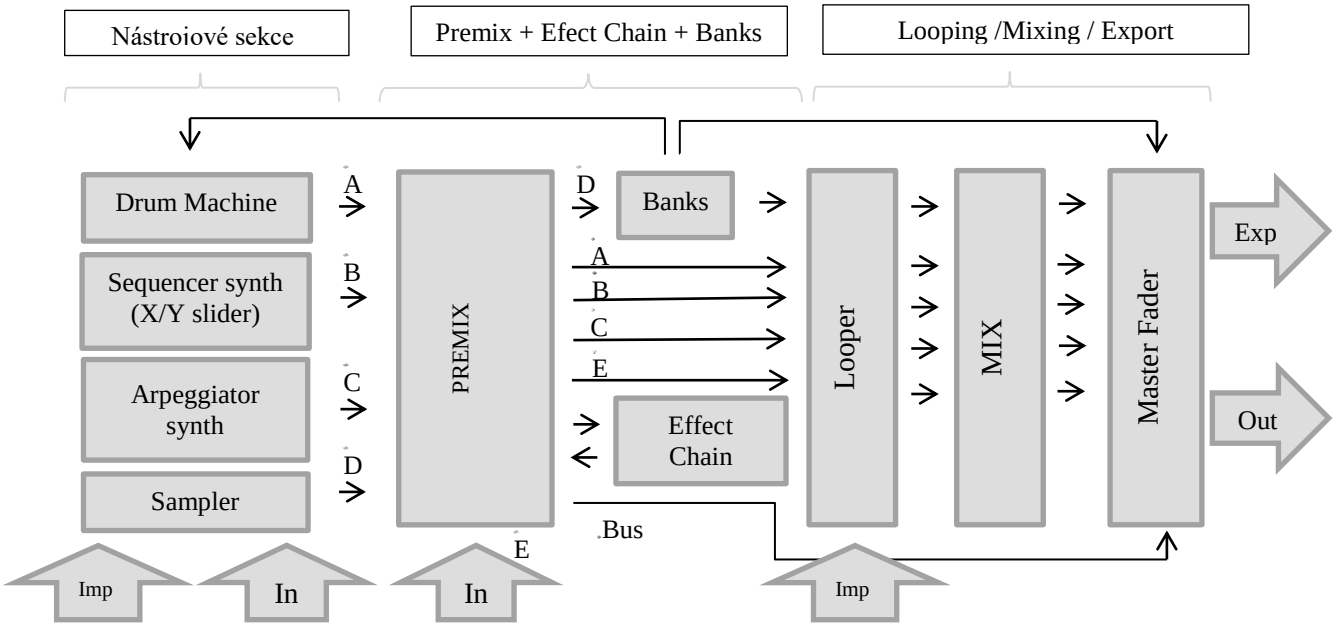
2.1 Aktualizovaný návrh a návrh grafického rozhraní.

Při realizaci jsem zjistil, že z původního návrhu v některých případech vycházet nelze, a proto jsem se rozhodl návrh poupravit a zvolil jsem jiný přístup k řešení některých částí aplikace. Původní návrh dělil aplikaci do několika sekcí, přičemž se v podstatě celou dobu pracovalo s jedním samplem, jehož zdrojem mohl být zvuk nahraný pomocí mikrofonu zařízení, zvuk načtený z paměti zařízení, nebo zvuk vygenerovaný zvukovým syntezátorem. Samply mělo jít na sebe následně skládat, lepit a vrstvit a vytvářet tak sekvence. Prvním problémem, na který jsem narazil, bylo, jak samply na sebe skládat a lepit. Pure Data sice nabízejí mnoho možností pro práci se samply, ale spojování více samplů dohromady je poněkud komplikované. Navíc veškeré operace se samply jsou náročné na paměť zařízení a to jak operační tak pevnou. Dalším problémem bylo, že původní návrh počítal s možností přelaďování samplů pomocí metody zpožďovací linky, která je sama o sobě komplikovanější na programování. Nicméně sama metoda přelaďování by nebyla takovým problémem jako charakter jednotlivých samplů. Klasický mikrofon smartphonu nemá zrovna parametry pro kvalitní nahrávání a při pořízení takového samplu je přítomen i nemalý šum. Pokud bychom se snažili nahrát např. souhlásku a, a tu pak přelaďovat tak, že bychom chtěli pomocí ní vytvořit melodii, tak výsledná efektivita by nebyla valná, nemluvě o náročnosti na paměť a CPU při každém přehrávání vzorku. Vzhledem k těmto skutečnostem jsem se rozhodl poněkud přehodnotit přístup pojetí aplikace. Cílem samozřejmě zůstalo, že aplikace jako softwarový hudební nástroj musí být veskrze experimentální a její pojetí vycházelo z kombinace Sampleru sekvencéru a syntetizéru. Změnil jsem jen přístup k řešení jednotlivých částí aplikace.

2.1.1 Rozvržení

Aplikace je rozdělena na několika stran, mezi kterými lze plynule přecházet slidem. Přičemž celá aplikace by se dala rozdělit do několika sekcí, které se liší určením: nástrojové sekce, premix/efektové sekce, samplovací sekce a výstupní mix export sekce. Nástrojová sekce se skládá z bicího automat. Na úvodní straně je potom možné vytvořenou skladbu znovu naimportovat do Sampleru nebo Looperu a znovu je upravovat. Importovat lze i samozřejmě i jiné soubory. (strana 1), jednoduchého sekvenceru (strana 2), Arpeggiator syntetizéru (strana 3) a bank uložených zvuků (strana 4). Sekce slouží k vytvoření rytmů a melodií a pomocí bank lze vyvolávat nahrané a uložené zvuky. Následuje premix a efektová větev. Premix sekce (strana 5) je mixovací konzole, do které jsou směřovány všechny nástroje a audio vstup. Slouží k nastavování úrovní signálů a umožňuje nasměrování signálu do efektové větve. Efektová větev nebo smyčka (strana 6 a 7) se skládá z několika efektů pro úpravu signálu. Následující sekcí je Sampler/Looper sekce, která se skládá ze dvou rekordérů. První z nich je Sampler (strana 8), který umožňuje rozšířené možnosti práce s jedním samplem, který pak může být uložen do bank. Další je Looper-recording (strana 9) ten umožňuje nahrát několik stop, jejichž zdrojem mohou být všechny nástroje a audio vstup. Úkolem Looperu je mimo jiné skládání různých

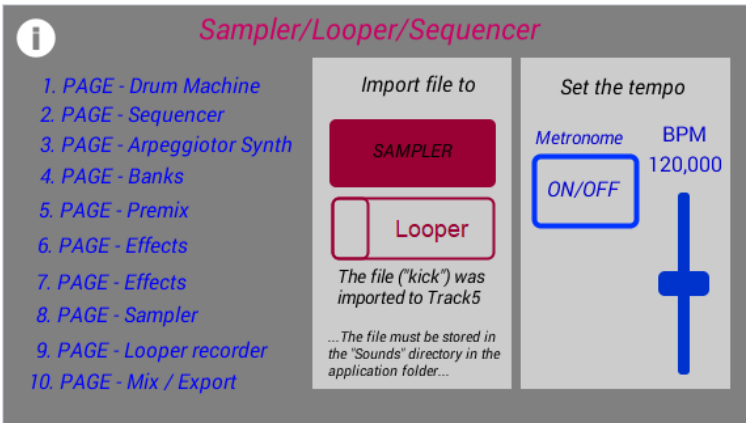
částí skladby, pomocí skládání smyček na sebe. Poslední sekcí je Mix/Master/Export sekce (strana 9), která umožňuje pomocí jednoduchého mixu úpravu hlasitosti a panoramy jednotlivých stop z Looper-recording sekce a jejich následný export do souboru Wav na paměť zařízení



obr. 2-2: Schéma rozvržení

2.1.2 Úvodní sekce

V úvodní sekci aplikace se nachází seznam jednotlivých stránek aplikace. Dále je zde možné importovat soubory. Tlačítko *Sampler* je pro importování do Sampleru a tlačítko *Looper* je pro import do Looper recorderu. Po stisknutí tlačítka Sampler se objeví dialogová okno s výzvou zadání názvu souboru, který se má naimportovat. Po zadání názvu a potvrzení se soubor naimportuje. Po stisknutí menu Looper se objeví seznam tracků, pro výběr do kterého z nich se má soubor naimportovat.



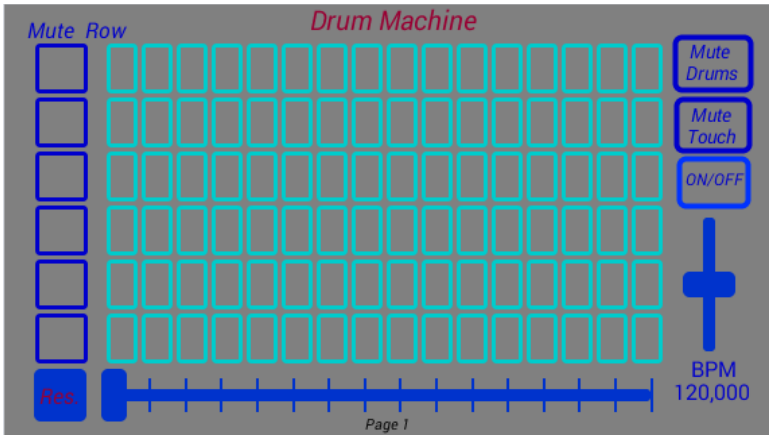
obr. 2-3: Úvodní strana

Po zvolení jednoho tracku se objeví stejné dialogové okno jako u Sampleru. Po naimportování souboru se objeví v labelu pod tlačítky hlášení o tom, kam byl soubor naimportován. Dole v labelu je

poznámka o tom že, „*soubor pro import musí být umístěn v adresáři Sounds*“. V úvodní sekci je také umístěn hlavní toogle pro vypnutí a zapnutí metronomu. Slider napravo slouží k nastavení tempa v Bpm. Základní nastavení je 120 Bpm. Metronom je společný pro následující tři sekce.

2.1.3 Drum Machine

Sekce Drum Machine umožňuje pomocí hlavní tabulky grid (uprostřed) zaznamenat a přehrát téměř libovolnou smyčku bicích nástrojů. Tabulka se skládá z 96 PADů/přepínačů, jsou to vlastně tlačítka toogle. Každý řádek odpovídá jiné části bicí soupravy. Konkrétně odshora dolů řádek 1. kick drum, řádek 2. snare drum, řádek 3. tom, řádek 4. closed hi-hat, řádek 5. open hi-hat, řádek 6. cymbal. Sloupec Mute Row (vlevo) umožňuje úplně vypnout libovolný řádek. Tato funkce je výhodná například pro přechod mezi různými groovy, kdy je možné nechat hrát např. jen činely a postupně si vytvořit nový groove a ten pak postupně odkrýt. Tlačítko Res. slouží pro reset tabulky. Vymažou se tak veškeré zatrhnuté groovy a tabulka se obnoví.



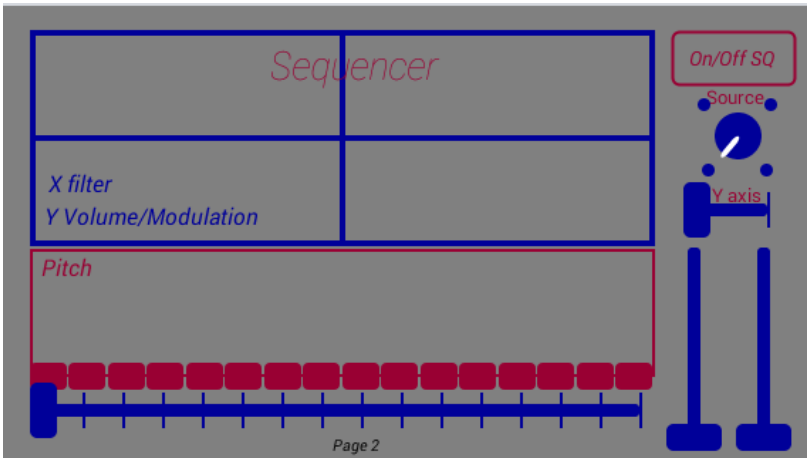
obr. 2-4: Drum Machine

Pod tabulkou se nachází slider s šestnácti pozicemi, které ukazují aktuálně přehrávaný sloupec. I zde se nachází toogle pro zapnutí a vypnutí metronomu. A slider pod ním slouží pro nastavení tempa. Toogle *Mute Drums* slouží ke ztlumení celého bubeníka a toogle *Mute touch* slouží ke ztlumení zvuku při dotyku, což je výhodné zejména tehdy pokud měním smyčku za chodu.

2.1.4 Sequencer

Pomocí sequenceru je možno vytvořit sekvenci o 16 notách, jejichž tón lze měnit pomocí multislideru Pitch. Výchozí tón je vlastně souzvuk dvou tónů, jejichž výšku lze nastavit pomocí dvou sliderů vpravo. Rychlost přehrávání se odvíjí od nastaveného tempa z předcházející stránky Drum Machine. Jejich přehrávací fronty jsou spřažené. Pozice na slideru odpovídá aktuálně hrané notě. Nad blokem Pitch se nachází tzv. XY slider. Pomocí něhož lze kontinuálně měnit hodnotu pásmové propusti (osa x) v závislosti na hlasitosti (osa y). Po překlopení přepínače *Y axis* do druhé polohy se přepne osa y do

módu modulace a místo hlasitosti lze na ose y ovlivňovat míru FM modulace. Tlačítko v pravém horním rohu umožňuje vypnutí a zapnutí sequenceru, respektive ztlumení. Knob pod tímto tlačítkem umožňuje změnu zdroje zvuku pro výchozí tón. Zdroje zvuku jsou oscilátory s různými průběhy. V základním nastavení je zvolen pilový průběh, dále lze zvolit sinusový, trojúhelníkový a nakonec obdélníkový průběh.

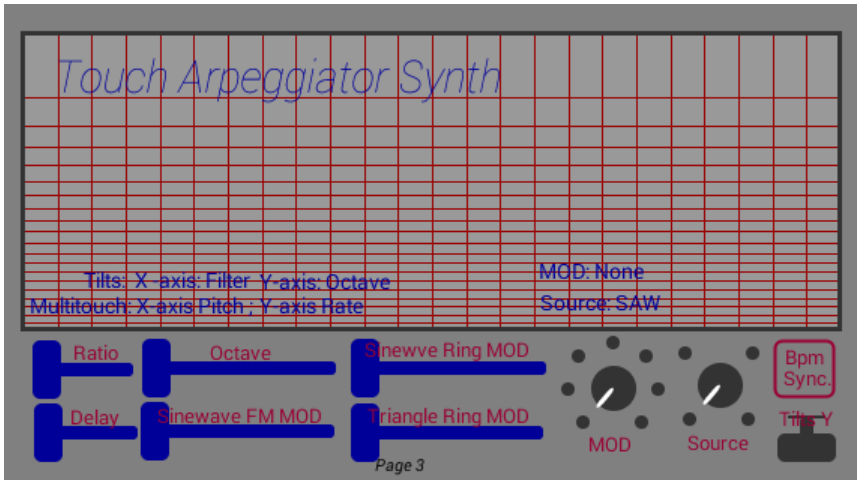


obr. 2-5: Sequencer

2.1.5 Touch Arpeggiator Synth

Arpegiattor synth je kombinace arpegigatoru a syntetizéru s multi-dotykovým MIDI ovladačem a možností ovlivňování výsledného zvuku pomocí gyroskopu zařízení. Středobodem nástroje je zmiňovaný multi-dotykový MIDI ovladač. Ten umožňuje současný dotyk, až na tolika místech, kolik dovoluje zařízení. V současnosti (2018) jsou běžné 10 bodové dotykové obrazovky. Dotykový panel je rozdělen do určitých zón. V ose x představují zóny klávesy MIDI ovladače s rozsahem tří oktáv. Rozsah oktáv lze ale ještě měnit dvěma způsoby, buď sliderem *Octave* nebo pomocí gyroskopu tzn. náklonem zařízení v ose y. Teoretický rozsah nástroje je potom 8 oktáv, od subkontra až po čtyřčárkovanou. Rozdělení zón v ose y představuje tempo opakování tónu. Délku náběhu a doběhu mezi tóny lze upravit sliderem *Ratio*. Přepínačem *Bpm Sync*. Je možné synchronizovat tempo opakování tónu s hlavním metronomem. To znamená, že všechny tři nástroje můžou být rytmicky synchronizovány. Při dotyku se tedy rozezní tón v závislosti na výšce tónu a tempu opakování. Pokud je dotyků více, jsou tóny opakovány v takovém pořadí, v jakém proběhly dotyky na panel. Barvu tónu lze měnit přepínači *MOD* a *Source*. Přepínač source umožňuje volbu mezi čtyřmi průběhy signálu a to pilovým, sinusovým, trojúhelníkovým a obdélníkovým průběhem. Volba průběhu zobrazuje v labelu *Source* v pravé části panelu. V základu je nastaven pilový průběh. Přepínač MOD umožňuje přepínat mezi modulacemi rozehraných tónů. V základu (pozice 1) není zvolena žádná modulace. V pozici 2 je zvolena sinusová kruhová modulace, v pozici 3 je zvolena trojúhelníková kruhová modulace, v pozici 4 je to FM modulace, v pozici 5 je to kombinace sinové a trojúhelníkové kruhové modulace, v pozici šest je to kombinace FM modulace a kruhové sinusové modulace a pozici 7 je to kombinace FM

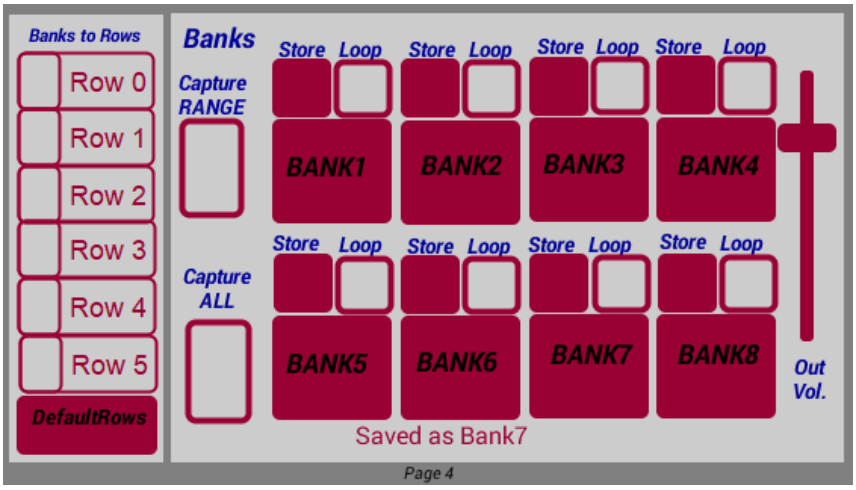
modulace a trojúhelníkové kruhové modulace. Míru jednotlivých modulací lze ovládat slidery *Sinewave FM MOD, Sinewave Ring MOD, Triangle Ring MOD*. Přepínač *Tilts y* umožňuje přepínat mezi zmiňovaným ovládáním oktáv a ovládáním míry zvolené modulace pomocí náklonu zařízení. Slider *Delay* umožňuje nastavit zpoždění vestavěného efektu delay.



obr. 2-6: Touch Arpegiator Synth

2.1.6 Banks

Banky jsou místa pro uložení Samplů, jsou reprezentovány PADy (tlačítky) při jejichž stisku je z paměti vyvolán uložený sampl. Toho lze využívat například, při nahrávání v Sekci Recording, kde můžu pomocí bank vyvolávat samply a nahrávat je přímo do stopy. Případně lze využít i zmiňované možnosti načíst celý obsah banky do jedné stopy. Přepínačem *Store Range* se pro uložení do bank zachytí označený výběr ze sekce Sampler/Looper. Přepínačem *Store Loop* se pak vybere celý sampl.



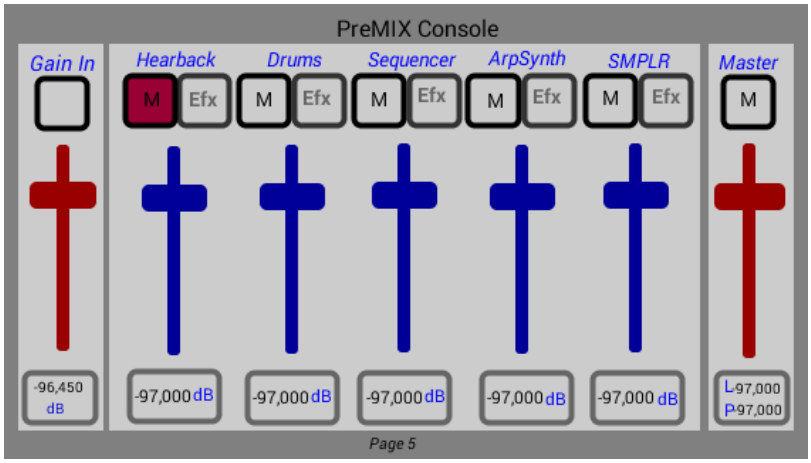
obr. 2-7: Banks

Uložení do příslušné banky se provede tlačítkem *Store*. Po uložení se v labelu ve spodní části objeví zpráva, do které banky byl Sampl uložen. Velké tlačítko např. *Bank1* slouží pro přehrání uloženého Samplu. Tlačítko *Loop* umožňuje přehrávání samplu ve smyčce. Slider *Out Vol*. nastavuje úroveň

výstupního signálu z bank. Sloupec *Banks to Rows* v levé části umožňuje načítání zvuků uložených v bankách do jednotlivých řádku bicího automatu. Po stisknutí některého z *Row 0 - 5* se objeví menu s nabídkou všech bank, pro uložení konkrétní banky do konkrétního řádku. Tlačítko *DefaultRows* obnovuje původní zvuk v řádcích bicího automatu.

2.1.7 Premix

Sekce *Premix* je určena k úpravě hlasitosti před případným nahráváním v sekci looper. Každý zmiňovaný nástroj má svůj slider pro nastavení výstupní hlasitosti. Svůj slider má zde i Sampler a to z důvodu, že je využíván zejména pro ukládání do bank. Odposlech ze vstupu *Hearback* zde disponuje také vlastním sliderem, ale je od startu aplikace ztlumen, aby se předešlo zpětné vazbě. Úplně vlevo se nachází slider *Gain In*, který slouží pro nastavení úrovně vstupního signálu a úplně vpravo je slider *Master* což je master fader, který slouží pro nastavování úrovně hlasitosti hlavního výstupu aplikace. Oba dva fadery, jak vstupní tak výstupní, jsou opatřeny indikací průchozího signálu, který je zobrazen v číselných hodnotách RMS v dB suplujících VU metr. Překročení prahové hodnoty indikuje červená kontrolka pod nimi. Kontrolka je vlastně toggle přijímající vstupní hodnoty. Všechny zmiňované nástroje i Sampler a odposlech je před faderem možné nasměrovat do efektové smyčky. K tomu slouží přepínač *Efx*. Přepínač *M* slouží ke ztlumení větve. I zde je možné indikovat průchozí signál.



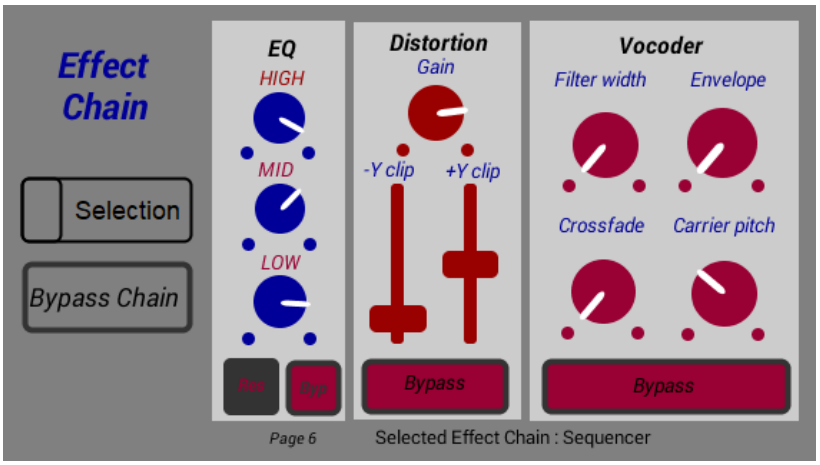
obr. 2-8: Premix Console

2.1.8 Effect Chain

V efektové sekci se nacházejí 6 efetků ekvalizér, distortion, vocoder, FM-synth, phaser a delay/echo. Sedmým efektem je reverb, ten byl ale kvůli nefunkčnosti na androidu odpojen. Efekty jsou zapojeny v takovém pořadí, v jakém jsou zobrazeny. Každý zmiňovaný nástroj i Sampler a Hearback má svou efektovou větev, která se skládá ze stejných efektů. Ovládání konkrétní efektové větve pro konkrétní nástroj se vybírá pomocí menu *Selection*. Při nastavování parametrů se parametry ukládají. To znamená, že při přepínání efektových větví zůstávají zadané parametry v konkrétních větvích uloženy. Přepínač *Bypass chain* umožňuje obejití celé vybrané efektové větve. Bypass tlačítkem disponují

i všechny efekty a i bypass je parametr, který lze uložit. Nyní proberu ovládací parametry jednotlivých efektů.

Jednoduchý třípásmový parametrický ekvalizér je zařazen na začátek smyčky. Skládá se s filtrů dolní propust 0 – 200 Hz, pásmová zádrž 200 – 2000Hz a horní propust 2000 Hz – 20 kHz. Pomocí tlačítka *Res*. lze nastavit výchozí úroveň 0 dB. Tlačítko „*Byp*.“ slouží jako Bypass.

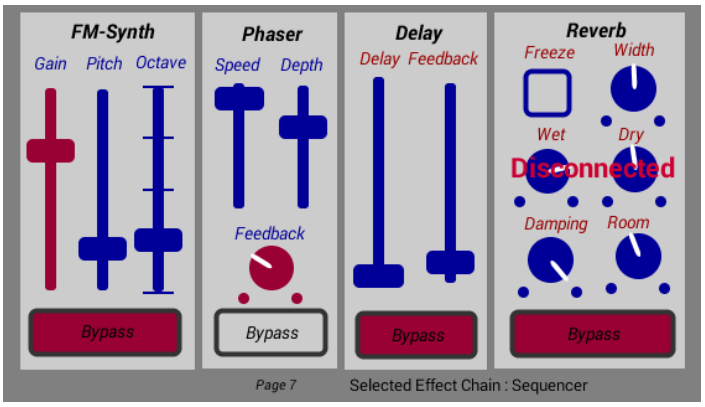


obr. 2-9: Effect Chain (strana6)

Za ekvalizérem je zařazen efekt Distortion. Slider Y- a Y+ umožňuje ostrou limitaci signálu a to jak kladných tak záporných hodnot. Díky této limitaci vznikají vyšší harmonické. Potenciometr *Gain* slouží k zesílení limitovaného signálu. Následujícím efektem je *Vocoder*.

Tento efekt je primárně určen pro úpravu lidského hlasu, ale lze využít i pro úpravu jiných signálu pocházejících z jiných zdroje než lidských hlasivek. Knobem *Filter width* lze nastavit šířku filtrace vstupujícího signálu, *Envelope* slouží pro nastavení citlivost sledovače obálky. *Crossfade* nastavuje poměr mezi zpracovaným a nezpracovaným zkresleným signálem a nakonec *Carrier pitch* nastavuje výšku upraveného zvuku.

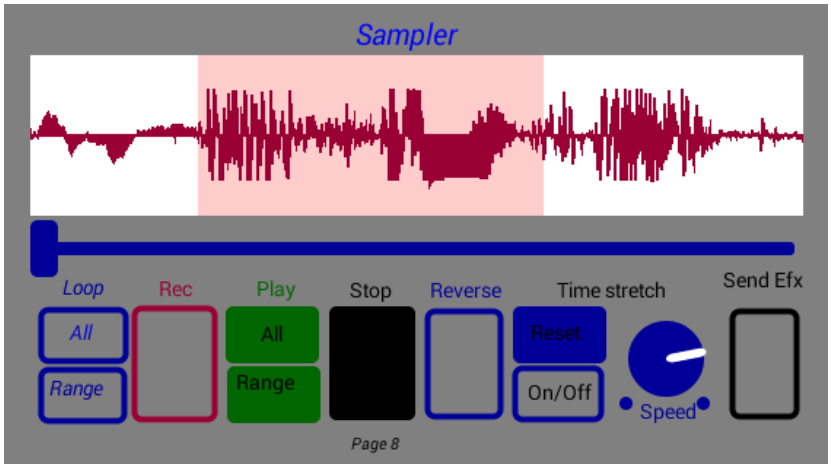
Dalším efektem v řadě je *FM-synth.* Tento efekt umožňuje monofonní FM syntézu procházejícího signálu. Slider *Gain* umožňuje zesílení průchozího signálu. Parametr *Pitch* nastavuje výšku modulovaného signálu a parametr *Octave* umožňuje nastavení výšky v oktávových intervalech.
Následujícím efektem je *Phaser* . Ten upravuje průchozí signál pomocí fázové modulace. Parametrem *Speed* se nastavuje frekvence modulačního oscilátoru parametrem *Depth* nastavuje hloubku modulace a *Feedback* počet opakování zpožďovacích linek.
Posledním použitým efektem v řadě je *Delay*, což je echo/delay efekt sloužící k vytváření simulovaných ozvěn. Parametr Delay slouží pro nastavení doby dalšího opakování a parametrem Feedback se nastavuje zpětná vazba resp. počet opakování



. **obr. 2-10: Effect Chain (strana 7)**

2.1.9 Sampler

V sekci Sampler je možno nahrávat nové samply pomocí mikrofonu zařízení, importovat nové samply z adresáře Sound (import lze provést v úvodní sekci), nebo je možné poslat a načíst stopu z následující strany *Looper/Recorder*. Při nahrávání pomocí mikrofonu se vstupní úroveň signálu nastavuje pomocí slideru Gain in v Premix sekci. Nahrávání se spustí přepínačem *Rec* a zastaví jeho opětovným stiskem. Nahraný nebo načtený sampl lze přehrát jednou a celý pomocí tlačítka *Play*, nebo přehrávat ve smyčce pomocí přepínače *Loop All* a to i pozpátku tlačítkem *Reverse*.

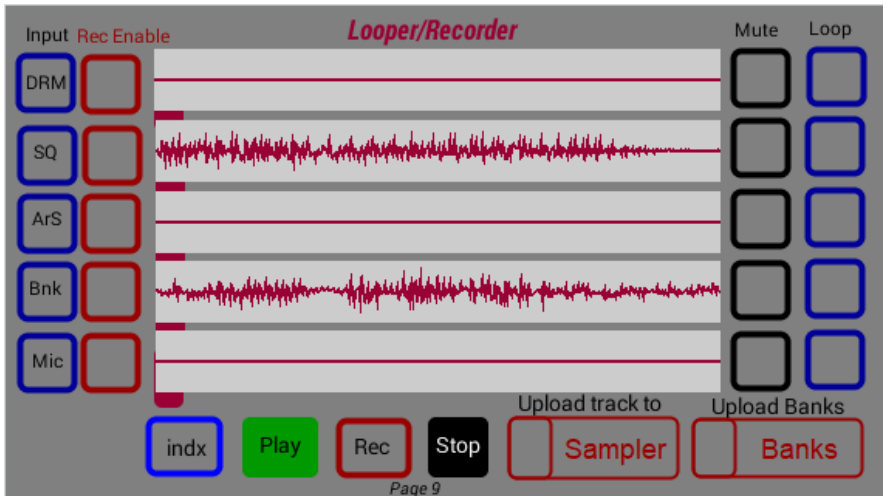


obr. 2-11: Sampler/Looper

Pomocí panelu lze dotykem provést výběr určité části samplu a i ten přehrávat ve smyčce pomocí *Loop Range*, samozřejmě i s možností *reverse* nebo přehrát jednou pomocí *Play range*. Tlačítko Stop zastaví zvolené přehrávání a vrátí vše na začátek. Funkce Time stretch umožňuje nastavovat rychlost přehrávání vzorku a tím v podstatě přelaďovat výšku zvuku. Funkce se zapne pomocí tlačítka *On/Off* . Rychlost přehrávání se nastavuje knobem *Speed*. Tlačítkem *Reset* se rychlost vrátí do normálu. Pomocí přepínače *Send Efx* lze také posílat aktuálně nahraný/načtený sampl do efektové větve, kde jej můžeme dále upravovat.

2.1.10 Looper/Recorder

Looper-Recorder je určen pro nahrávání nástrojů v aplikaci, ale i nahrávání nových samplů. Kromě nahrávání lze i naimportovat do jednotlivých stop nové samply. Výběr pro import se provádí v úvodní sekci. Nahrávat lze tedy z několika zdrojů, to znamená ze všech nástrojů. Routing neboli výběr pro nahrávání se provádí pomocí přepínačů *Input* úplně vlevo. Přepínač *DRM* slouží pro Drum Machine, přepínač *SQ* pro Sequencer, přepínač *Ars* pro Touch Arpeggiator Synth, přepínač *Bnk* pro banky a přepínač *Mic* slouží pro mikrofon zařízení. Výběr stopy pro nahrávání se provádí pomocí přepínačů *Rec Enable* v druhém sloupci zleva. Je možné přiřadit více nástrojů k jedné stopě (resp. všechny), nebo povolit nahrávání jednoho nástroje do více stop (resp. všech). V podstatě je možno nastavit nahrávání všech nástrojů do všech stop. Nahrát je možné maximálně pět stop libovolné délky. Délka je reálně omezena velikostí místa na velkokapacitní paměti zařízení a zbývajícím místem v paměti RAM. Velikost vstupní úrovně hlasitosti se nastavuje v sekci *Premix*. Po stisknutí přepínače *Rec*. se začne vstupní signál nahrávat do zvolené stopy. Pokud jsou vybrány nástroje DRM, SQ nebo ArS, spustí se hlavní metronom a smyčka se začne nahrávat přesně od začátku fronty. Nahrávání se zastaví opětovným stiskem a v případě nahrávání zmiňovaných nástrojů se nahrávání zastaví, až přehrávání dojde na konec fronty. Tím pádem je možné například přehrávat smyčku z nahraného bicího automatu plynule bez nechtěných přechodů.



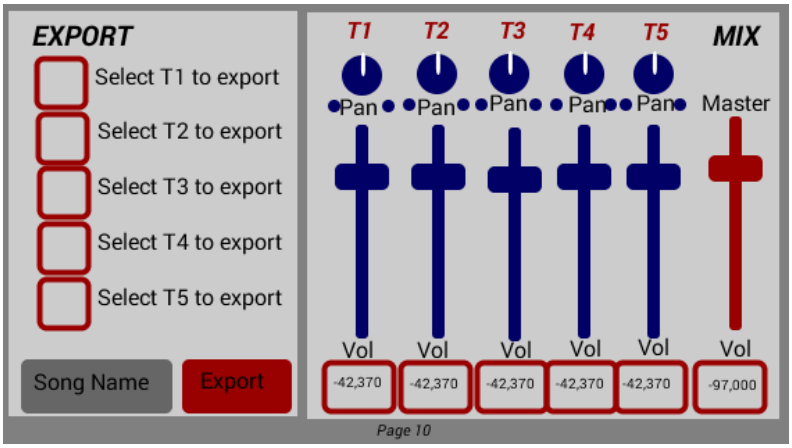
obr. 2-12: Looper/Recorder

Jednotlivým stopám odpovídají šedé panely uprostřed, po nahrání je vykreslen waveform nahraných stop. Každé nahrané stopy se v panelu roztáhnou na jeho maximální délku. Pod každým panelem kurzor, který ukazuje aktuální pozici při přehrávání. Je zřejmé, že pří různé délce stop skončí v jiný čas. Kurzory tedy slouží k tomu, aby šlo rozpoznat, v jaké pozicí se nachází přehrávání konkrétní stopy. Přepínač *indx* slouží právě k vypnutí a zapnutí index sliderů, kvůli případné náročnosti na CPU. Stopy lze přehrávat jednou pomocí tlačítka *Play*. Pro přehrávání ve smyčce je možné vybrat určité stopy nebo všechny naráz, k tomu slouží přepínače *Loop* úplně vpravo u každé stopy. Tlačítko *Stop* zastaví přehrávání. Looper/Recorder umožňuje odeslat libovolnou nahranou nebo naimportovanou

stopu do předcházející sekce Sampler. Pro tuhle volbu menu *Upload track to Sampler* úplně vpravo dole. Dále je možné načítat jednotlivé banky zvuků do konkrétních stop pomocí menu *Upload Banks*. Výběr stopy pro upload se provádí pomocí *Rec Enable*. Jednotlivé stopy lze ztlumit pomocí přepínače *Mute*.

2.1.11 Mix / Export

Poslední sekcí je sekce Mix/Master/Export. Jejím primárním úkolem je export hotové skladby složené v sekci Looper/Recorder. Slouží ale i jako jednoduchý mixpult pro zmiňovanou sekci. Výběr skladby pro export se provádí pomocí tlačítek *Select to export*. Slidery T1 – T5 umožňují nastavení výstupní úrovně hlasitosti. Slider *Master* což je master fader, který slouží pro nastavování úrovně hlasitosti hlavního výstupu aplikace a výstupu pro export. Je spřažen s master faderem v sekci *Premix*. V podstatě se jedná o jeden master fader s dvěma místy regulace. Pomocí *Pan* T1 – T5 lze nastavit orientaci stopy v panoramě. Hodnoty pod slidery jsou výstupní hodnoty RMS v dB a suplují VU metr. Při překročení prahové hodnoty se navíc rozsvítí červeně kontrolka pod nimi. Tlačítkem *Song Name* je vyvoláno dialogové okno a uživatel je vyzván k zadání jména písničky. Ta se uloží do kořenového adresáře MobMuPlat do složky Sounds. Při stisknutí tlačítka *Export* začne export označených stop do souboru Waw.



obr. 2-13: Export

3 Realizace

Celá aplikace (projekt) by se dala rozdělit do dvou vrstev. Do vrstvy grafické a vrstvy aplikační. Grafickou vrstvu jsem již představil v aktualizovaném návrhu. Její propojení s aplikační vrstvou a aplikační vrstvu samotnou představím nyní. Aplikační vrstva je vlastně audio engine, který je kompletně naprogramovaný pomocí jazyku Pure Data. Grafická vrstva je poté vytvořená na GUI platformě pro Pure Data MobMuPlat, která kromě grafického rozhraní zprovozní PD soubory v Androidu. Obě vrstvy jsou úzce propojené a obě udržují aplikaci v chodu.

3.1.1 Struktura Projektu

Celý projekt se skládá z hlavního patche, který sdružuje pod sebou několik Subpatchů a abstrakcí. Struktura celého projektu je následující:

- Hlavní patch: ProjektSamplLoopSeq
  - Drum Machine
    - pd RowPlay
    - pd GridSet
    - pd row 0
    - pd row 1
    - pd row 2
    - pd row 3
    - pd row 4
    - pd row 5
  - pd XYsliderSequencer
  - pd arpeggiatorSynth
  - pd Banks
  - pd Premix
    - pd efects Drums
      - pd Equalizer, pd Distortion, pd Vocoder, pd FmSynth, pd Phaser, pd Delay
    - pd efects Sequencer
      - pd Equalizer, pd Distortion, pd Vocoder, pd FmSynth, pd Phaser, pd Delay
    - pd efects ArpeggiatorSynth
      - pd Equalizer, pd Distortion, pd Vocoder, pd FmSynth, pd Phaser, pd Delay
    - pd efects Sampler
      - pd Equalizer, pd Distortion, pd Vocoder, pd FmSynth, pd Phaser, pd Delay
    - pd efects Hearback
      - pd Equalizer, pd Distortion, pd Vocoder, pd FmSynth, pd Phaser, pd Delay

- pd Sampler
- pd Looper/Recorder
  - pd track 1
  - pd track 2
  - pd track 3
  - pd track 4
  - pd track 5
  - pd uploadBanks
- pd Mix/Import/Exports
  - pd Export
  - pd Import

3.1.2 Propojení s MobMuPlat

Na úvod představím způsob propojování s GUI platformou MobMuPlat. Veškeré grafické prvky jsem vytvářel v editoru MobMuPlat, který je postaven na jazyku Java. Zde se definuje, pro jaké zařízení bude aplikace určena, zda pro tablet nebo smartphone. Dále se nastaví, kolik stránek bude GUI mít, vytvoří se zde a definují ovládací prvky a zobrazovací prvky. Každý prvek musí mít svou jedinečnou adresu, pokud chceme, aby jeden prvek ovládal jednu věc. Je samozřejmě možné přiřadit více prvkům stejnou adresu. Pro propojení s grafickou platformou je nutné nejprve otevřít patch „*PdWrapper.pd“*. Ten zajišťuje propojení vstupů a výstupů GUI platformy s hlavním patchem.

Z grafických prvků přichází většinou hodnoty 0 a 1, 0 až 1, nebo list hodnot. Definice výstupu z GUI se provádí bezdrátově pomocí objektů „*receive“* a „*send“*. Ve všech případech se výstupy definují následovně: „*receive fromGUI“, „route list“* nebo „*list trim“,* a dále se pak k rozlišení více prvků používá podmíněný průchod typu „*route“*.

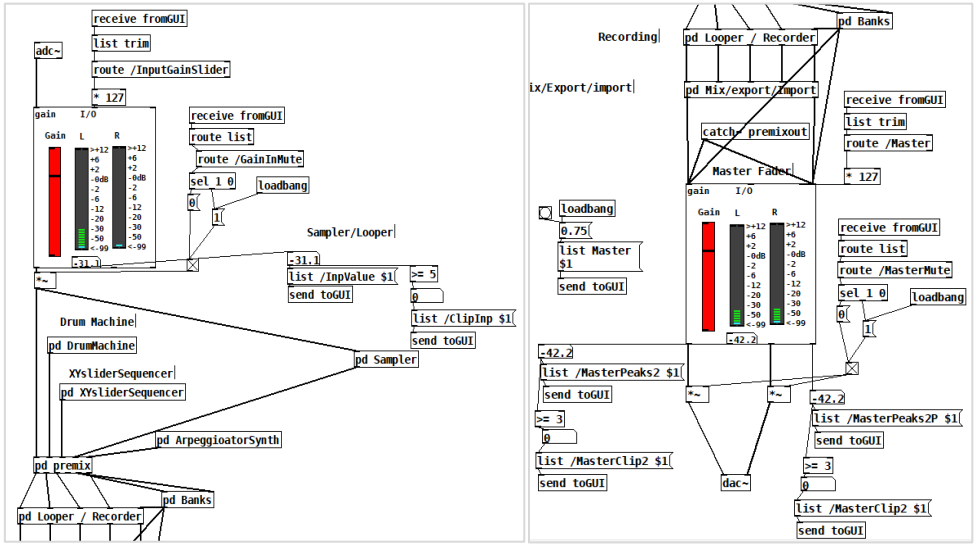
Vstup do GUI se provádí pomocí posíláním zpráv a to vždy ve tvaru „*list /adresa“*. V případě odesílaných hodnot se používá symbol dolaru „*$1“*. Zprávy jsou odeslány pomocí objektu s příkazem *„send toGUI“*. Pro komunikaci s operačním systémem, např. při vyvolání dialogového okna, se používá příkazů „*send toSystem“* a pro příjem dat „*receive fromSystem“*.

3.1.3 Hlavní patch Projekt SamplLoopSeq

Jak již bylo zmíněno, hlavní patch sdružuje a zastřešuje všechny subpatche a abstrakce. V hlavním patchi je definován hlavní audio vstup (*adc~*) a audio výstup (*dac~*). Za hlavním vstupem a před hlavním audio výstupem jsou zařazeny abstrakce „*gain“*. Ty nastavují vstupní a výstupní úroveň audio signálu. Pro vstupní signál funguje abstrakce jako zesilovač vstupních signálů a pro výstupní signál funguje abstrakce jako Master fader. Jelikož téměř všechny slidery v GUI mají základně nastavenou hodnotu 0 – 1 násobí se vstupní hodnota **127*. Slider abstrakce Gain je nastaven na hodnotu 0 – 127

kvůli větší citlivosti při volbě rozsahu a možnosti zesilování slabých signálů. Abstrakce Gain je postavena na základě známého zapojení z publikace Pure data [13] a využívá objektů *dbtorms* k převedení signálu v dB na průměrnou hodnotu signálu RMS. Objekt přijímající data od ovládacích prvků v GUI je pro input gain „*InputGainslider"* a pro master fader *Master* .

Signál z mikrofonu tedy vstupuje do patche přes vstup *adc~* a přes abstrakci gain a objekt (*~) fungující jako mute pokračuje do subpatche „*pd Sampler"* a do subpatche „*pd premix"*. Do subpatche premix je vstupní signál přiveden kvůli možnosti odposlechu vstupního signálu a ovlivňovaní jeho úrovně. Do subpatche Sampler je vstupní signál přiveden kvůli nahrávání. Výstup z tohoto subpatche je přiveden do subpatche premix. Do subpatche premix vstupují i další důležité subpatche a to: „*pd DrumMachine"* což je subpatch zajišťující funkci bicího automatu, dále pak „*pd XYsliderSequencer*" zajišťující funkci sekvencéru a nakonec ještě „*pd ArpegiatorSynth"*, zajišťující funkci Touch Arpeggiator syntetizéru. Subpatche premix je důležitou spojnicí nástrojů hlavního vstupu a Sampleru. Zde je každý signál přiveden do své větve, kde lze ovládat úroveň průchozího signálu a případně signál odbočit do efektové větve „*pd Efects"*.



a) b)

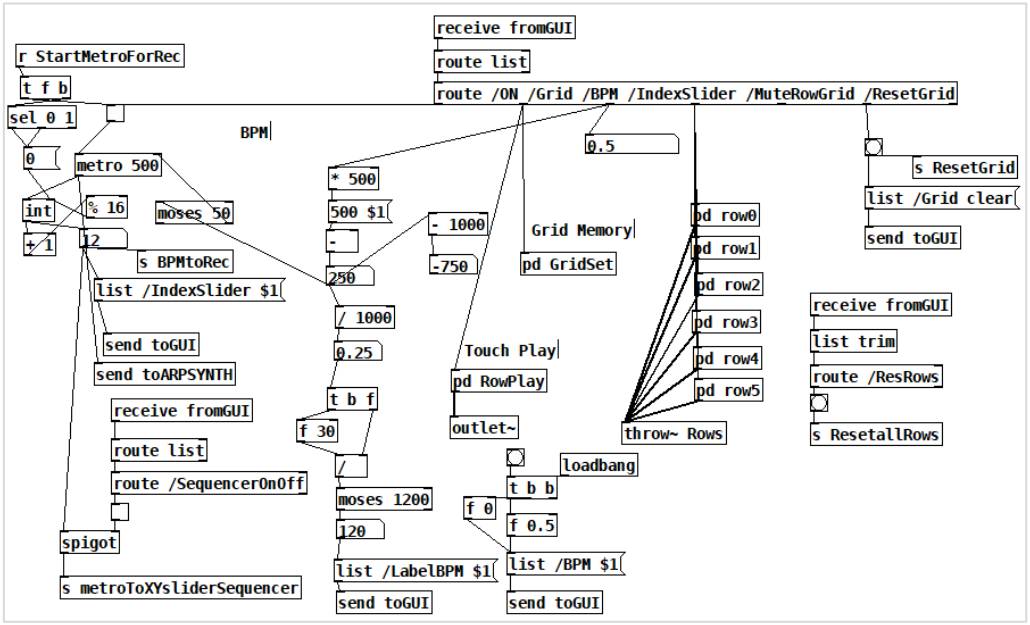
obr. 3-1: Hlavní patch a) vstup, b) výstup

V subpatchi efects jsou efekty zařazeny tak, jak je popsáno výše v kapitole 3.1.1. Signál je v subpatchi premix rozdělen do dvou sběrnic. Jedna je svedena přímo na master fader a slouží pro přímý odposlech a druhá přivádí signál do subpatche „ *pd Looper/Recorder"*. Tento subpatch zajišťuje funkci Looper/Recorderu. Ze subpatche premix je ještě přímo z větve Sampleru posílán signál do „*pd Banks"*. Tento subpatche slouží pro banky zvuků a odsud jsou bezdrátově posílány samply pro změnu řádků v drum machine. Subpatch Banks pokračuje přímo na master fader a do subpatche Looper/Recorder. Není sveden zpět přes premix, protože se nepočítá s opakovaným efektováním bank touto cestou a navíc disponuje vlastní regulací výstupního signálu. Opakovaného efektování bank je možné docílit nahráním do Looper/Recorderu a následným posláním do Sampleru. Ze subpatche

Looper/Recorder je signál přiveden do subpatche *„pd Mix/Export/Import“*, zde je realizován jednoduchý mixpult pro úpravu výstupního signálu a panoramy před exportem. Export umožňuje subpatch *„pd export“*, který je umístěn v ně Mix/Export/Import. Zde se nalézá i subpatch *„pd Import“* zajišťující import zvukových souborů.

3.1.4 Drum Machine

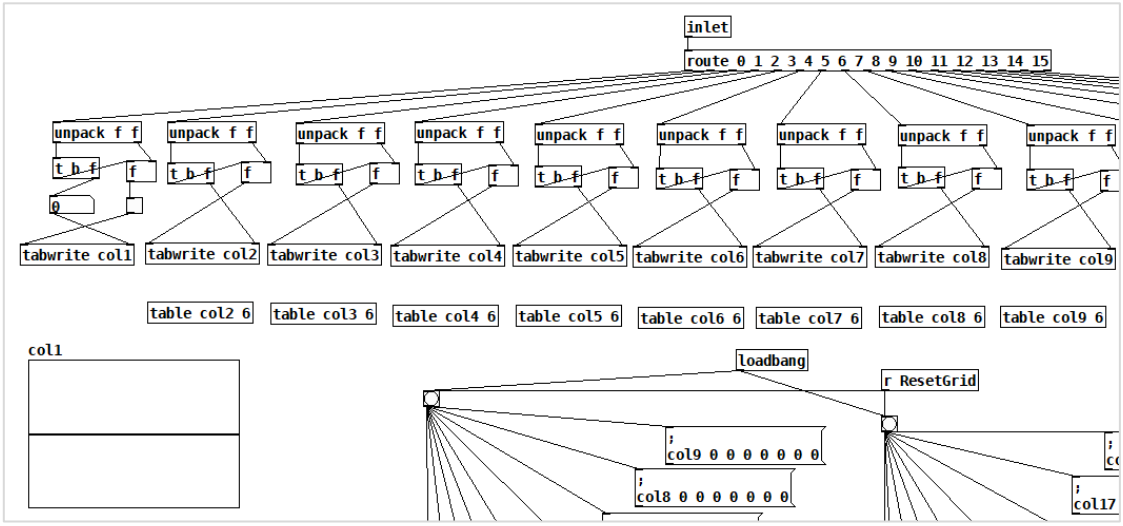
Subpatch *DrumMachine* je enginem bicího automatu. Skládá se z dalších subpatchů - *„pd Gridset“, „pd RowPlay“ a „pd row 0 – 5“*. Ovládací prvky GUI jsou adresovány jako *„route On /Grid /BPM /IndexSlider /MuteRowGrid /ResetGrid“*. Adresa on slouží pro vypnutí a zapnutí hlavního metronomu, který zároveň spouští smyčku automatu. Je označen jako hlavní, protože udává tempo i ostatním nástrojům a subpatchům. Metronom je definován objektem metro. Na horkém vstupu přijímá hodnotu 1 pro zapnutí a hodnotu 0 pro vypnutí a na studeném vstupu se nastavuje hodnota opakování v milisekundách za klik. Defaultně je v metronomu nastaven parametr 500 ms což odpovídá hodnotě 120 BPM. Nicméně hodnota BPM se dá měnit pomocí slideru BPM v GUI. Jeho rozsah hodnot 0 – 1 se násobí * 500 a odečítá od původní. Objekt moses *50* pak zabraňuje překročení hodnotě 50, protože za touto hodnotou se metronom velmi zrychlí a je rytmicky nepoužitelný. Metronom v pravidelných intervalech přičítá hodnotu +1 z objektu int. Vznikne tak cyklus o 16 krocích, které zobrazuje v GUI Index Slider. Stejný cyklus se odesílá dále do XYsliderSeqencer a ArpeggiatorSynth. Metronom přijímá také povel k zapnutí při nahrávání v sekci Looper/Recorder.



obr. 3-2: pd DrumMachine

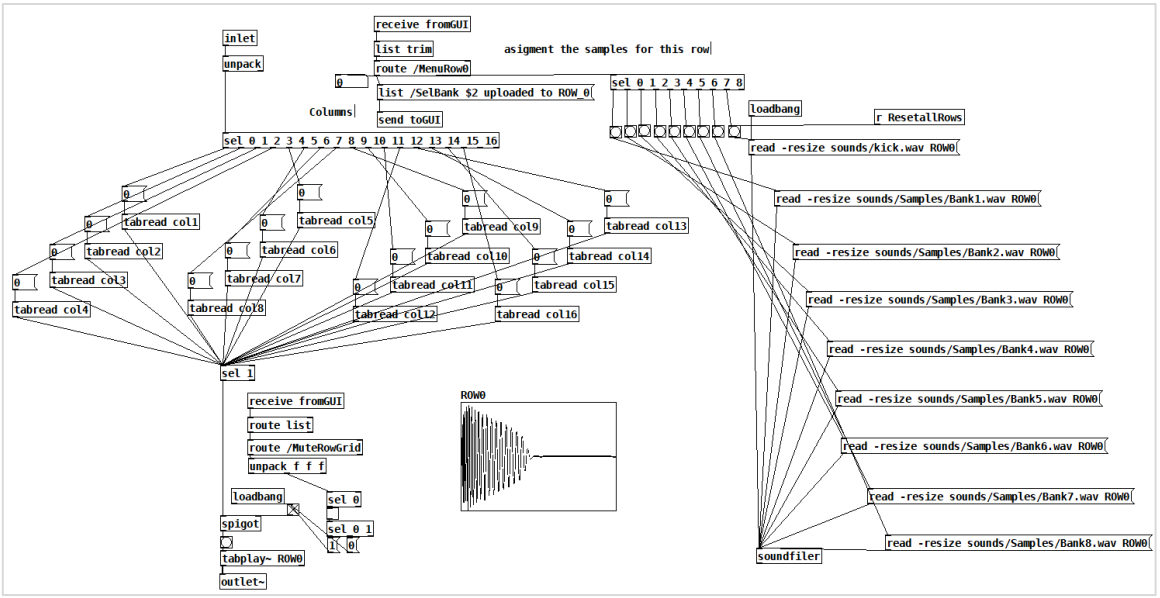
Adresa *Grid* slouží pro příjem hodnoty z tabulky GUI. Subpatch GridSet přijímá tyto hodnoty a pomocí objektu *route* rozhoduje, z kterého sloupce a řádku došla hodnota 0 nebo 1. Každému sloupci je přiděleno jedno pole nebo table, do kterého se vejde šest hodnot, což je počet řádků. Zde se ukládají hodnoty zadané v GUI tabulce automatického bubeníka. Pomocí adresy *ResetGrid* je možné

vymazat uložené hodnoty a obnovit tabulku. To se provádí pomocí zprávy „*list /Grid clear“* a uložením hodnot 0 do polí. Z adresy *Grid* přijímá povely i subpatche *RowPlay,* ten slouží pro přehrávání samplu při dotyku na tlačítko. Hned za inletem subpatche je zařazen objekt *unpack,* který rozdělí hodnoty seznamu a pomocí objektu *sel* je pak vybrán odpovídající řádek a poslán na *tabplay*. Přehrávání uložených hodnot v subptachi GridSet zajišťují Objekty *row0 – row5*. Jeden objekt odpovídá jednomu řádku tabulky a všechny objekty jsou identické, liší se jen v hodnotě, kterou přijímají a kterou posílají.



obr. 3-3: pd GridSet

Všechny objekty přijímají hodnotu z adresy *IndexSlider,* která odpovídá sloupci tabulky. V každém subpatchi *Row* je ihned za inletem zařazen objetkt *unpack,* ten rozděluje seznam podle odpovídajícího řádku a posílá jej do objektu *sel.* Odtud se v závislosti na hodnotě odpovídajícího sloupce odešle bang odpovídajícím výstupem na zprávu s hodnotou odpovídajícího řádku do objektu *tabread* odpovídajícího sloupce. Poté se odešle do objektu *sel 1* hodnota, a pokud ta odpovídá jedničce odešle se bang objekt (*tabplay~),* který přehraje sampl a odešle ho na *outlet~* a odtud dále na audio výstup. V rámci subpatche row je realizováno i načítání samplů z bank do konkrétních řádků. V závislosti na tom, z jakého menu pro odpovídající řádek v okně *Banks* je vybrána konkrétní banka, přijde do subpatche tohoto řádku číselná hodnota vybrané banky. Ta je pomocí *sel* vyselektována a poslána na objekt (*read –resize sounds/...)* čímž se přepíše pole „*ROW“* odpovídajícího řádku. Při každém načtení hlavního patche je pomocí *loadbang* načten výchozí sampl pro odpovídající řádek. Změnou souboru, které DrumMachine načítá z pevné paměti, lze docílit i jejich přemazání přímo v adresáři. Soubory se ale musí jmenovat stejně jako původní. Tímto způsobem lze tedy v Drum Machine také libovolně měnit vyvolávané samply. Podmínkou pouze je, že musí být ve formátu wav. a název souboru musí zůstat nezměněn.

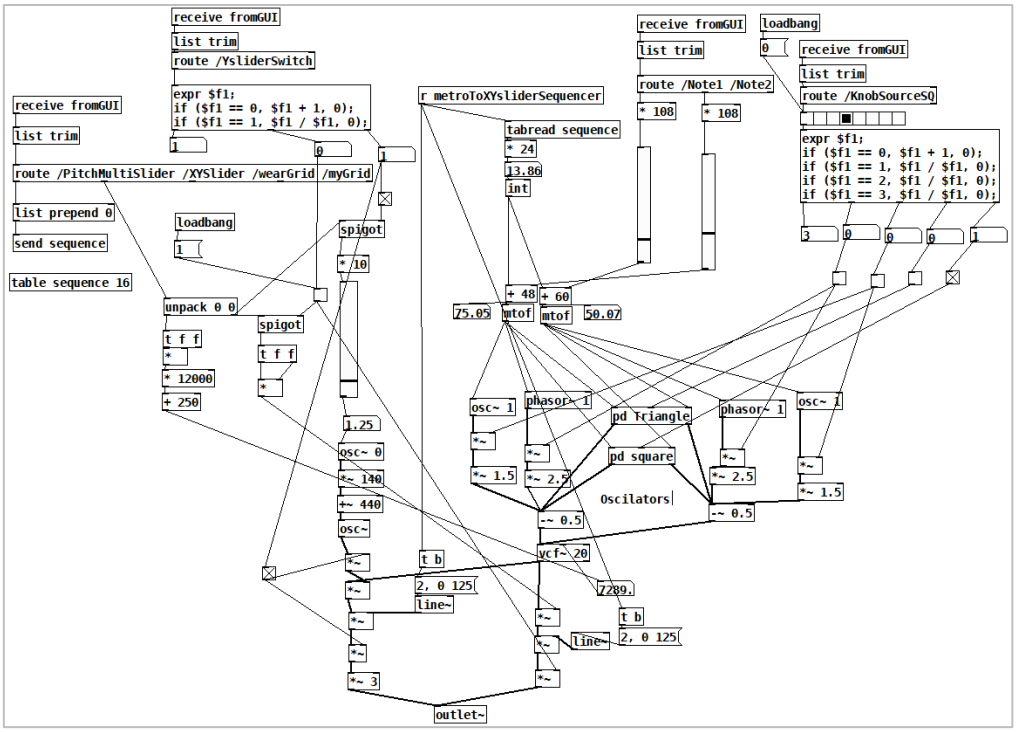


obr. 3-4: pd row0

3.1.5 Sequencer

Tento subpatche umožňuje funkci sekvencéru. Subpatch je postaven na základě známého zapojení v šablonách MobMuPlat[18]. Sequencer přehrává synchronně s bicím automatem ve smyčce sekvenci slouženou z 16 tónu. Změna hodnot výšky jednotlivých tónu v GUI pomocí Multislideru se přijímá adresou „*PitchSlider“* a ukládá do objektu „*table sequence 16“*. Bezdrátovým inletem *„metroToXYsliderSequencer“* pak vstupuje výstup metronomu z „*pd DrumMachine“* a odpovídajícím tempem se čtou uložené hodnoty z „*table sequence"*. Přečtená hodnotu se dále násobí hodnotou 24 a ta pak pokračuje do dvou objektů, kde se v jednom přičte k hodnotě 48 a v druhém k hodnotě 60. Takhle dostaneme konečné hodnoty MIDI a oba dva objekty po té navazují každý na objekt *mtof,* který převádí číslo MIDI noty na tón odpovídající frekvenci. Čtená hodnota je tedy souzvuk dvou tónů, jejichž výška se pomocí změny hodnoty vstupující do objektu mtof může měnit. To se provádí pomocí dvou sliderů „*Note1“ a (/Note2).* Hodnoty lze tedy měnit od 24 až do 108. Hodnota MIDI 24 odpovídá tónu C1 a hodnota 108 tónu C8, což je rozsah sedmi oktáv. Hodnoty v hertzích poté vstupují do generátoru průběhu. Výběr generátorů probíhá pomocí objektu *expr$f1;* ten přijímá hodnoty z „*KnobSourceSQ“* a v závislosti na přijaté hodnotě je buď splněna podmínka a na konkrétní výstup je poslána hodnota 1, nebo není splněna podmínka a na výstup je poslána 0. Binární hodnoty přijímá objekt audio přepínače (*~) a ten vypne nebo zapne konkrétní generátor. Generátory jsou po dvojicích a rozlišujeme čtyři druhy: a to generátor pilových, sinusových, trojúhelníkových a obdélníkových průběhů. Generátor pilových průběhů je realizován pomocí objektu *phasor~,* kde dochází ke generování pilového signálu. Generátor sinusový průběhů je realizován pomocí objektu osc~. Generátor trojúhelníkového průběhu je realizován pomocí dvou pilových generátorů *phasor~,* kdy má jeden posunutou počáteční fázi a začíná v maximu. Průběhy jsou pak sečteny a poslány dál. Generátor Obdélníkových průběhů je realizován pomocí dvou pilových generátorů *phasor~,* kdy jeden generátor

má posunutou počáteční fázi, tzn. začíná v maximu a ještě je fázově otočen o 90°. Nakonec jsou průběhy sečteny a vznikne tak obdélníkový průběh. Signál z vybraného generátoru pokračuje dále do objektu *(vcf~),* což je napěťově řízená pásmová propust (*Voltage Controled bandpass Filter*). Její parametry jsou nastavovány z adresy *XYslider,* kdy hodnota x vstupuje do druhého inletu objektu, kde se nastavuje střední frekvence filtru a hodnotou y se násobí výstupní hodnota audio signálu z *vcf~*, čímž se signál zesiluje nebo zeslabuje. Funkce y osy XYslideru se dá přepnout pomocí „*YsliderSwitch"* místo násobení výstupními hodnotami, se začne výstupní signál modulovat fm modulátorem. Ten je realizován pomocí známého zapojení fm modulátoru [21]. V podstatě jde o kombinaci frekvenční modulace a kruhové modulace. Hodnotami z osy y XYslideru nastavujeme modulační frekvenci oscilátoru *osc~ 0,* který moduluje nosnou frekvenci oscilátoru *osc~*. Takto zmodulovaný signál pak ovlivňuje výstupní frekvenci z *vcf~* v objektu *~. Výsledný signál je pak pomocí ramp (line~) v daných intervalech daných tempem hlavního metronomu odesílán na signálový outlet.

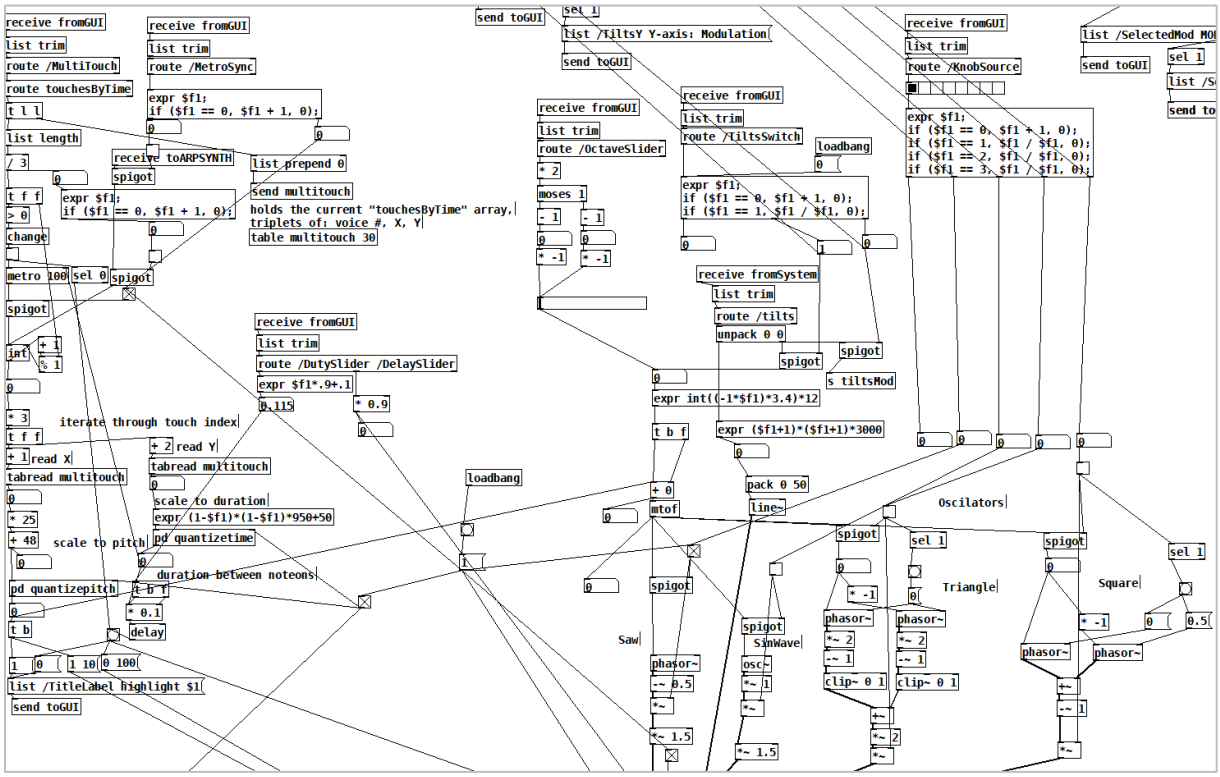


obr. 3-5: Sequencer

3.1.6 Arpeggiator Synth

Tento subpatche je postaven na základě šablony zapojení dostupné v MobMuPlat [18]. Zapojení je ale vylepšeno a přeprogramováno k použití s ostatními nástroji. Pokud dojde v GUI k dotyku na multitouch jsou data souřadnic dotyků pomocí objektu *„route touchesBytime"* seřazováno podle času a pomocí obejktu trig („*t l l"*) posíláno do table "*multitouch"* a do patche, který počítá počet dotyků. Pokud je počet dotyků větší jak 0, spustí se metronom. Pokud je ale zapnuta synchronizace s hlavním metronomem začne se přijímat tempo z hlavního metronomu. Tohle přepínání zajišťuje soustava

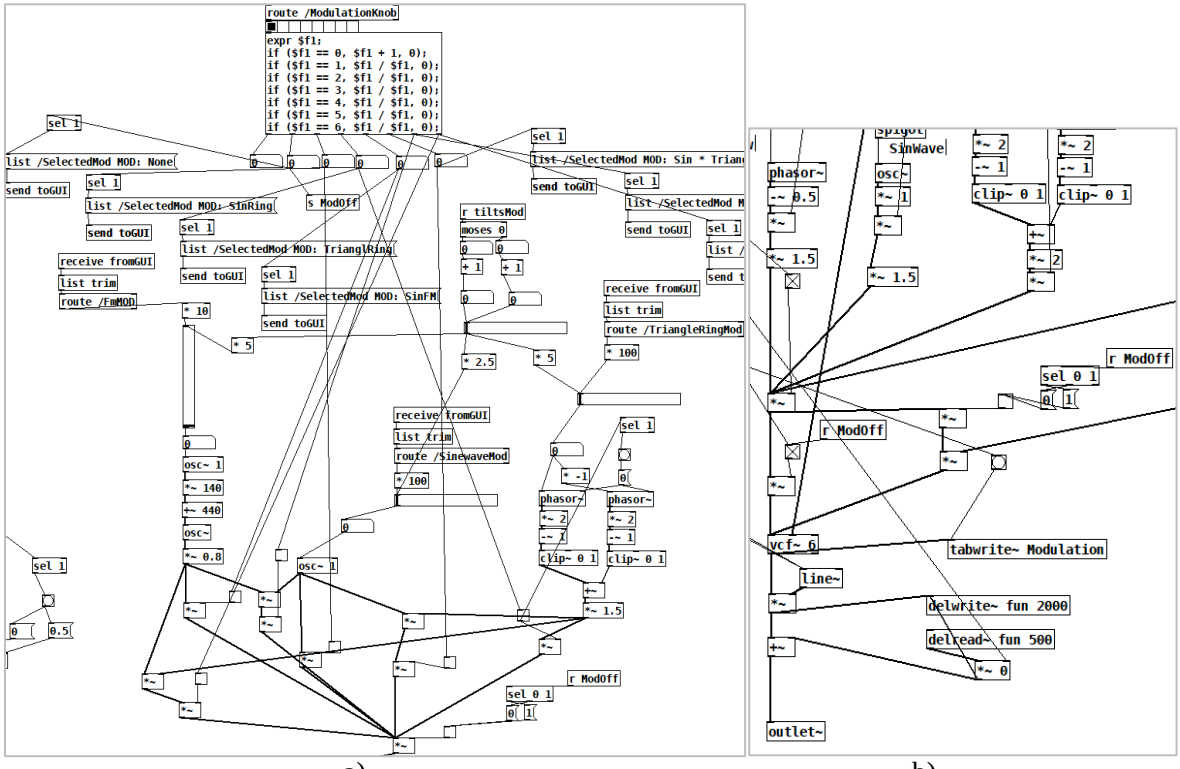
objektů expr a spigot. Pomocí objektu int a přijímaného tempa z metronomu bude probíhat iterace v závislosti na počtu dotyků. Dále jsou přes objekt trig hodnoty rozděleny a probíhá načítání z table „*multitouch*“. Z osy y jsou načítány hodnoty trvání noty a z osy x hodnoty výšky tónu. Hodnoty ještě prochází přes „*měřítka*“ trvání (osa y) a měřítko výšky tónu (osa x) a vstupují do subpatchů *„pd QunatizeTime“ a „pd QunatizePitch“*, které realizují rozdělení na zóny v rámci panelu. Výstup ze subpatche *„pd QunatizeTime“* je směrován na zpožďovací linku, pomocí které je regulován čas mezi doběhem a náběhem další noty (*ratio*). Výstupem hodnot x a y jsou pak události bang probíhající v určitých časových intervalech. Pomocí bang se odesílají hodnoty formou zpráv do rampy *line~*. S jejíž pomocí je signál přicházející z generátorů průběhů odesílán v určitých intervalech na výstup.



obr. 3-6: Arpeggiator Synth

Generátory průběhů jsou čtyři: generátor pilového, sinusového, trojúhelníkového a obdélníkového průběhu. Přepínání mezi nimi a jejich zapojení je identické jak u generátorů v subpatchi XYsliderSequencer v části 3.1.5. Vstupní hodnoty pro generátory jsou získávány ze zmiňovaného „*pd QuantizePitch*“. Ten odesílá MIDI noty, které se pomocí objektu *mtof* převedou na hodnoty v hertzích. Do tohoto objektu vstupují také hodnoty z regulátoru oktáv. Změny oktáv můžou být řízeny buď pomocí slideru *„OctaveSlider“* nebo gyroskopem zařízení (nahýbaní v ose y). Pro použití druhé možnosti změny oktáv je nutné použít přepínač *„TiltsSwitch“*. Přepínač může přepínat mezi dvěma režimy, které rozlišují, kam budou data o nahýbání zařízení v ose y posílána. V režimu jedna (přepínač v GUI ve spodní poloze) ovlivňuje nahýbání zařízení oktávy signálu a v režimu dva (přepínač v GUI je v horní poloze) ovlivňuje nahýbání míru zvolené modulace. Samotné přepínání

zajišťuje objekt expr a dva objekty spigot. Informace o poloze přepínače jsou odesílány do GUI na label *„TiltsY“*. Jak již bylo zmíněno procházející signál je možno v tomto nástroji modulovat. Opět jde o kombinace kruhové modulace s jinými druhy. V rámci tohoto subpatche jsou ale využity dva druhy modulátorů a to: frekvenční modulátor a kruhový modulátor. Přičemž modulátory kruhové modulace by se daly rozdělit z hlediska průběhu modulačního signálu na kruhovou sinusovou modulaci a kruhovou trojúhelníkovou modulaci. Princip zapojení frekvenčního modulátoru je uveden v předcházející kapitole (3.1.5). Princip zapojení modulátoru pro kruhovou sinusovou modulaci spočívá ve využití oscilátoru *osc~*, jehož frekvenci lze ovlivňovat buď pomocí slideru *„SinewaveMod“* nebo pomocí zmiňovaného nahýbaní v ose y. Modulátor pro kruhovou modulaci s trojúhelníkovým průběhem je realizován pomocí generátoru trojúhelníkových průběhů (princip zapojení viz. 3.1.5), jehož modulační frekvence lze ovlivňovat sliderem *„TriangleRingMod“* nebo také zmiňovaným nahýbáním v ose y. O přepínání mezi druhy modulace se stará v GUI otočný přepínač *„ModulationKnob“*, ten posílá do objektu expr hodnoty pro přepínání mezi šesti režimy. Režimy přepínání byly zmíněny v kapitole (2.1.5). Při přepnutí na konkrétní režim modulace je do GUI na label „SelectedMOD“ odesláno o jaký režim jde. Pokud je tedy zvolen nějaký režim modulace, tak modulační signál ovlivňuje nosný signál v násobičce (*~) a takto zmodulovaný signál pokračuje na filtr typu pásmová propust (vcf~), který je regulován rampou, jejíž vstupní hodnoty jsou ovlivňovány také pomocí gyroskopu zařízení (naklánění v ose x). Nakonec signál prochází na signálový outlet. Před outletem je ještě vestavěn delay, realizovaný pomocí zpožďovací linky „*fun*“. Jeho feedback lze ovládat pomocí slideru *„DelaySlider“*.

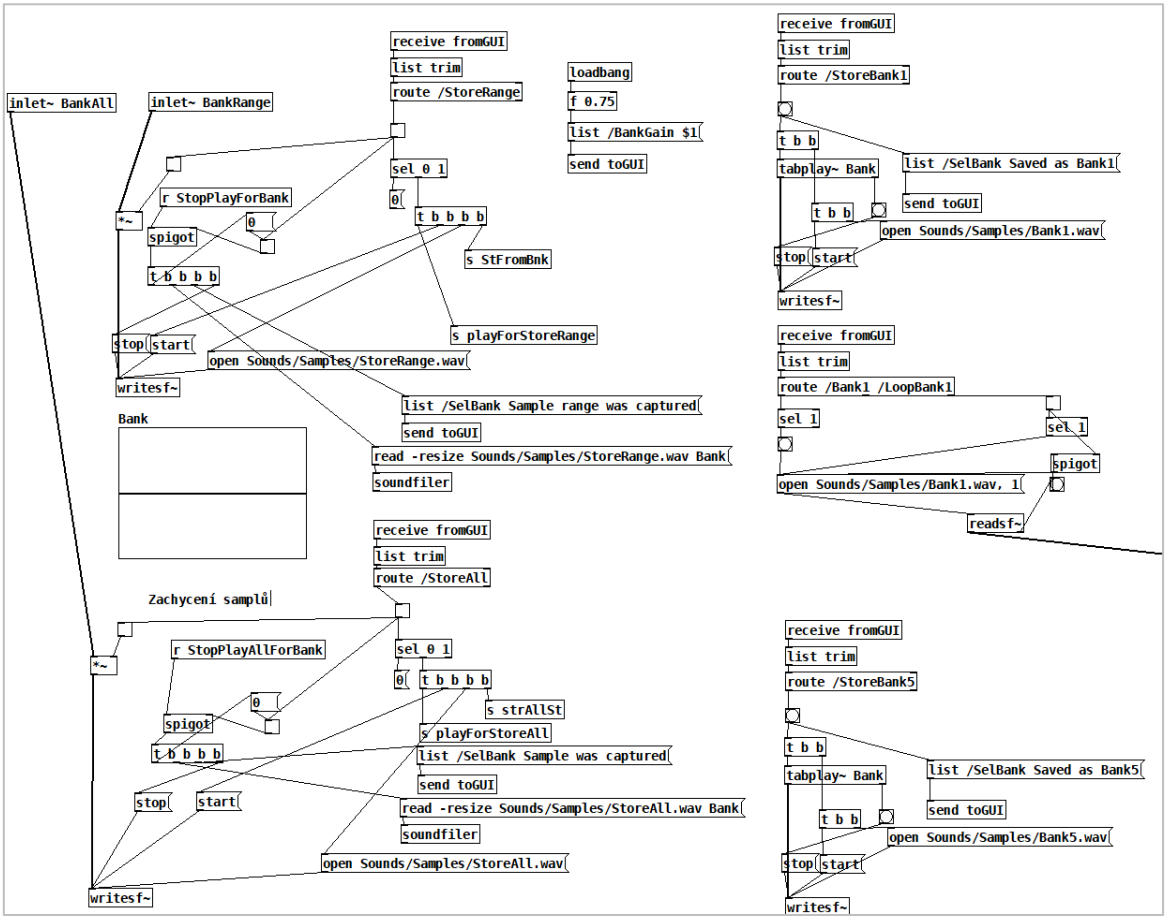


a) b)

obr. 3-7: Arpeggiator Synth a) modulátory b)výstup

3.1.7 Banks

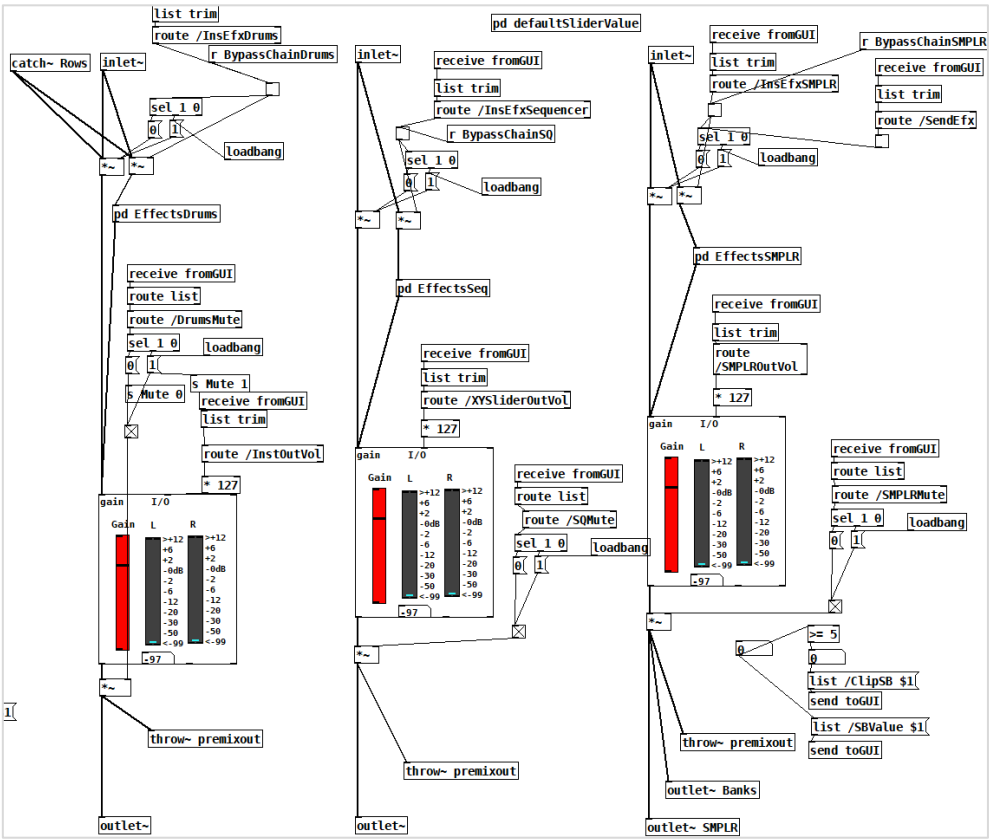
Subpatch Banks slouží pro ukládání vzorků, které jsou načteny nebo nahrány v subpatchi *Sampler* a mohou sem být posláný přes subpatch premix. Subpatch disponuje dvěma signálovými inlety do nichž sice vstupuje identický signál, ale rozdělení má smysl, protože jedna cesta vede do patche, který zachytává výběr samplu a druhá cesta vede do patche, který zachytává celý sampl. Zachytávání se v obou případech provádí tak, že se nejprve pomocí trigeru otevře soubor v „opnefile *Sounds/Samples/StoreRange.wav*“, následně se spustí stream pomocí objektu *writesf~* a nakonec se bezdrátovým outletem pošle povel do subpatche Sampler pro start přehrávání samplu. Při skončení přehrávání je přijata událost bang a pomocí ní je nahrávání zastaveno. Tímto způsobem se zachytí sampl a následné uložení do zvolené banky se provede v GUI pomocí tlačítka *Store*. To spustí přehrávání z pole *Bank* a pomocí objektu *writesf~* se začne zapisovat stejným způsobem do souboru *Sounds/Samples/Bank.wav*. Jakmile přehrávání skončí, odešle se bang na zprávu *stop* a stream se tak ukončí. Přehrávání uloženého samplu se pak provede pomocí objektu *readsf~*. Smyčka je zde realizována tak, že při stisknutí spínače *loop* se pošle do objektu *Spigot* hodnat 1, tím se uvolní cesta pro událost bang *(ta* je vždycky vyslána z objektu *readsf~* po ukončení přehrávání*)*, která opětovně spustí přehrávání. Signálové výstupy i vstupy jdou do subpatche opět přes abstrakci *Gain*, která umožňuje nastavit úrovně signálu.



obr. 3-8: Banks

3.1.8 Premix

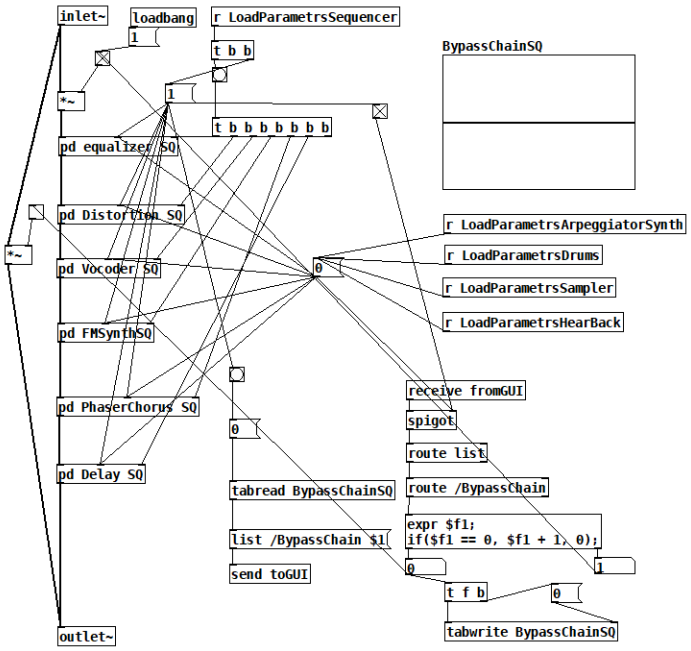
Úkolem tohoto subpatche je rozdělení signálu do jednotlivých větví s možností odbočení do efektové větve. Dalo by se říci, že jde svým způsobem o sběrnici. Subpatch má pět signálových inletů a stejný počet signálových outletů a bezdrátových signálových sběrnic. Zapojení všech větví je identické. Signál z inletu prochází přes signálový přepínač (objekt „*~“), který funguje jako mute celé větve. Poté signál prochází rozvětven do dvou dalších signálových přepínačů, které realizují přepínaní mezi odbočkou do efektové větve, nebo přímým směřováním na abstrakci gain. Při odbočení do efektové větve vstupuje signál do subpatche *„pd Effects…“* kde jsou zapojeny efekty. Každá větev signálu má svou vlastní subpatche *„pd effects“*, nicméně zapojení efektů je pro všechny subpatche stejné. Rozlišeny jsou kvůli ukládání hodnot. Signál po přímém průchodu, nebo průchodu přes efektovou větev vstupuje na abstrakci gain, s jejíž pomocí se reguluje průchozí signál. Poté signál pokračuje na signálový outlet a bezdrátovou sběrnici *„throw~“*. Z bezdrátových sběrnic je signál posílán rovnou na master fader a ze signálových outletů je signál posílán do subpatche *„pd looper“*. Signálová větev pro subpatche sampler disponuje navíc jedním signálovým outletem, pomocí kterého je signál směrován do subpatche „*pd Banks*“.



Obr. 3-9: Premix

3.1.9 Effects

Jak již bylo zmíněno v předešlé kapitole u subpatche premix je v každé signálové větvi možnost odbočit do efektové větve. Ta je realizována pomocí subpatche *„pd effects“*, kterým disponuje každá signálová větev. Volba pro odbočení do efektové větve je možná v sekci premix. Mezi více zvolenými efektovými větvemi lze přepínat. Přepínání jednotlivých efektových větví pomocí GUI bylo ale problematické. Ovládací prvky jednotlivých efektů mají jedinečné adresy a nastal tak problém, jak pomocí GUI ovládat více stejných subpatchů zároveň. Nenašel jsem použitelný způsob, jak v GUI přepínat mezi více okny se stejnými efekty, ale jinými adresami. Proto jsem nakonec využil možnost jednoho univerzálního grafického rozhrání pro efekty a samotné přepínání jsem řešil v PD. Každý subpatche konkrétního efektu v efektové větvi disponuje jedinečným polem pro ukládání parametrů. Parametry jsou ukládány v závislosti na vybrané efektové větvi. Pokud je vybrána například efektová větev pro sequencer, je v subpatchi *„pd EffectsSeq“* přijata bezdrátovým inletem událost bang, která pošle do všech subpatchů jednotlivých efektů hodnotu 1. Tím se otevře v jednotlivých efektech objekt spigot, který začne propouštět přijímaná data o změnách parametrů efektů v GUI. Od této chvíle se v jednotlivých efektech začnou ukládat data o změně parametrů do polí. Událost bang odeslaná při výběru efektové větve pro sequencer, je přijata i ve všech ostatních efektových větvích, tam je ale do jednotlivých efektů poslána hodnota 0, která zavře objekt spigot a tím i průchod dat z GUI. Takto je ukládána i hodnota zapnuto, vypnuto pro bypass efektové větve. Ten je realizován tak, že se bezdrátovým inletem odešle hodnota do subpatche premix, kde se přepne průchod signálu zpět na přímou větev. Ve výsledku tedy lze říci, že zapojení subpatchů jednotlivých efektových větví je identické, rozdíl je ale v pojmenování polí pro ukládání parametrů efektů zvolené větve. Dále tedy rozeberu funkci jednotlivých efektů umístěných v subpatchi effects.



obr. 3-10: Efektová větev pro Sequencer

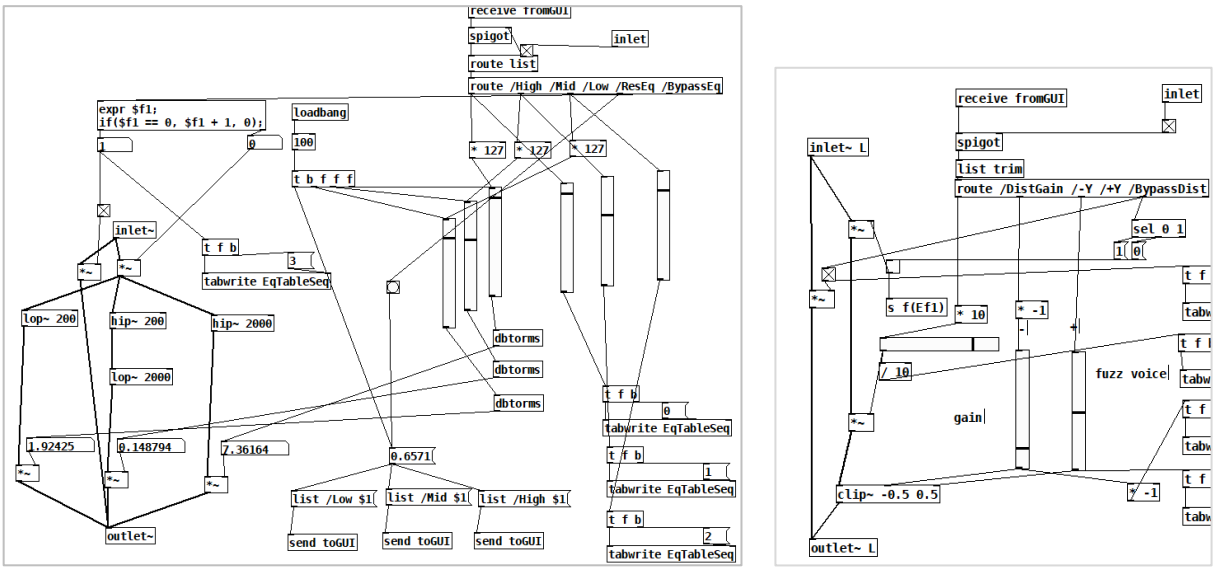
Equalizer

Prvním efektem v efektové větvi je equalizer. Zapojen je na základě zapojení uvedeného v publikaci Pure Data [13]. Využívá se objektů *lop~ (low pass filter) a hip~ (high pass filter)*. Objektům lze zadat parametr mezní frekvence filtruce signálu. Signál vstupuje z inletu a prochází přímo přes zmíněné objekty. Pro basy je nastaven objekt *lop~ 200*, pro středy jsou nastaveny dva objekty *hip~ 200 a lop~ 2000*, pro výšky je pak nastaven objekt *hip~ 2000.* Signál pak pokračuje dále přes signálový spínač a násobič (*~) a pak přes *outlet~* ven. Hodnoty filtrace se nastavují pomocí tří sliderů, které mění hodnoty v rozsahu 0 – 127. Tato hodnota se pak pomocí objektu *dbtorms* přepočítává na hodnotu v Hz a odesílá se bezdrátovým vstupem na chladné vstupy objektů (*~). I zde je realizován bypass pomocí objektů násobiče (*~) a *expr*

Distortion

Druhým efektem v pořadí je efekt distortion. Ten je tvořen pomocí objektu (*~), na jehož chladný vstup je připojen slider, který může hodnotou 0 – 100 násobit vstupní signál a zesilovat ho. Takto zesílený vstupní signál vstupuje do objektu *clip~ -0.5 0.5*, který limituje signál v záporných i kladných hodnotách. Parametr limitace se nastavuje pomocí dvou sliderů, jenž regulují ořez v rozsahu 0 až 1 a 0 až -1.

.



a) b)

obr. 3-11: a)Equalizer, b)Distortion

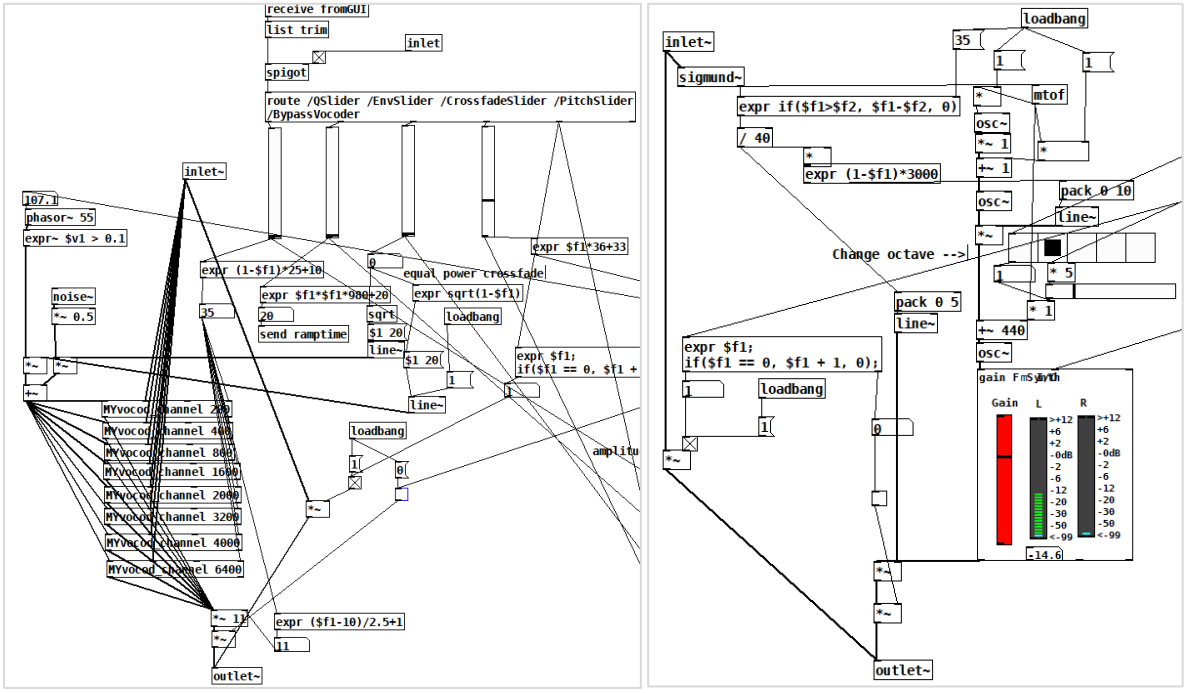
Vocoder

Dalším efektem v pořadí je efekt vocoder. Ten je s určitými úpravami zapojen podle šablony zapojení vocoderu dodávané se softwarem MobMuPlat [18]. Realizace zapojení je postavena na principu adaptivní filtrace (uvedeno v kapitole 1.4.4). Signál přicházejí ze signálového inletu je směrován do devíti abstrakcí *„MYvocod_chanel“*, kde dochází k filtrování signálu pomocí filtrů *bp~*. Signál

přicházející z páru filtrů, je dále směrován do sledovače obálky *env~*, kde dochází k odhadu amplitudy a dále pak převodu na RMS hodnoty pomocí *dbtorms*. Tyto hodnoty poté odchází spolu s hodnotami přijímanými ze slideru *„EnvSlider“* do rampy, která řídí průchod signálu přicházejícího z druhého páru filtrů *bp~*. Do tohoto páru filtrů je posílán signál pilového průběhu z generátoru *„phasor~ 55“*, jehož výstupní frekvence je regulována pomocí knobu *„pitchslider“*, kterým se vlastně nastavuje výška výsledného upraveného zvuku. Do tohoto signálu ještě může být přimíchán šum z objektu *„noise~“*. Míra přimíchání tohoto šumu je řízena jako equal power crossfade(*„CrossfadeSlider“*). Zapojení crossfadu je realizováno pomocí objektů expr a ramp signálu.

FM monophonic synth

Čtvrtím efektem v pořadí je efekt FM synth, který převádí vstupní signál na MIDI noty a ty frekvenčně moduluje. Zapojení je postaveno na základě zapojení dostupného z internetového zdroje [22]. Vstupní signál je pomocí objektu *„sigmund~“* převáděn na MIDI noty a potom dále v objekt *mtof* převáděn na hodnotu v hertzích. Tyto hodnoty pak vstupují do frekvenčního modulátoru, jehož princip zapojení je uveden v kapitole 3.1.5. Z druhého výstupu objektu sigmund přichází hodnoty sledovače obálky na rampy signálu. Ty pouští signál na výstup v závislosti na tom, zda se objeví nějaký signál na vstupu. Výšku zmodulovaného signálu lze měnit pomocí sliderů *„OctaveFmSynth“* a *„PitchFmSynth“*. Pomocí abstrakce gain je zesilován nebo zeslabován výstupní signál.



a) b)

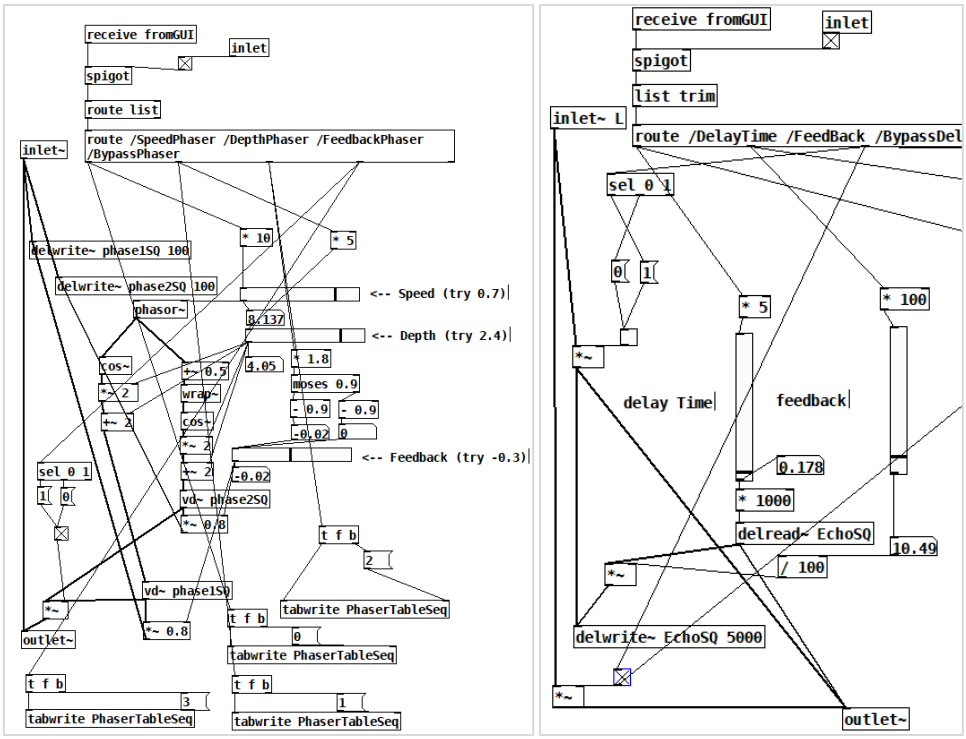
obr. 3-12: a) Vocoder, b) FM synth

Phaser

Pátým efektem v efektové větvi je efekt phaser. Ten je realizován na základě známého zapojení dostupného z internetového zdroje [22]. Efekt phaser je realizován v subpatchi „pd PhaserChorus“. Signál přicházející do subpatche se rozdělí do tří větví. Jedna větev jde přímo na výstup a další dvě větve jdou na zpožďovací linky *„phase1SQ 100“* a *„phase2SQ 100*. Signál je poté opakovaně načítán ze zpožďovacích linek pomocí nízkofrekvenčního oscilátoru, realizovaného pomocí sinových oscilátorů. Délka načítaných zpoždění je řízena pomocí parametru depth, který dvakrát násobí a poté přičítá hodnotu 2 k výstupům z oscilátorů. Rychlost tohoto načítání je řízena pomocí generátoru pilového průběhu *phasor~* a parametrem *„SpeedPhaser“*. Výstupní zpoždění signál ze zpožďovacích linek, je směrován na výstup, kde je smíchán s přímým signálem. Výstup ze zpožďovacích linek je zpětnovazebně zaveden na jejich vstup. Pomocí této zpětné vazby lze řídit počet opakovaných načtení signálu. Bypass je v tomto efektu řešen pouze odpojením signálu přicházejícího ze zpožďovacích linek.

Delay/Echo

Posledním použitým efektem v efektové větvi je efekt Delay/Echo. Efekt je realizován pomocí objektu *delwrite~ Echo 5000,* z něhož se čte pomocí objektu *delread* ~. Parametr zpoždění je opět přiváděn na vstup objektu *delread~* v rozsahu 0 – 1000 ms. Z výstupu objektu *delread~* je pak zapojena zpětná vazba do objektu delwrite a jedna větev signálu přímo na výstup. Parametr zpětné vazby se nastavuje pomocí slideru *feedback* připojeného na jeho chladném vstupu. Rozsah nastavovaných hodnot může být 0 – 100.

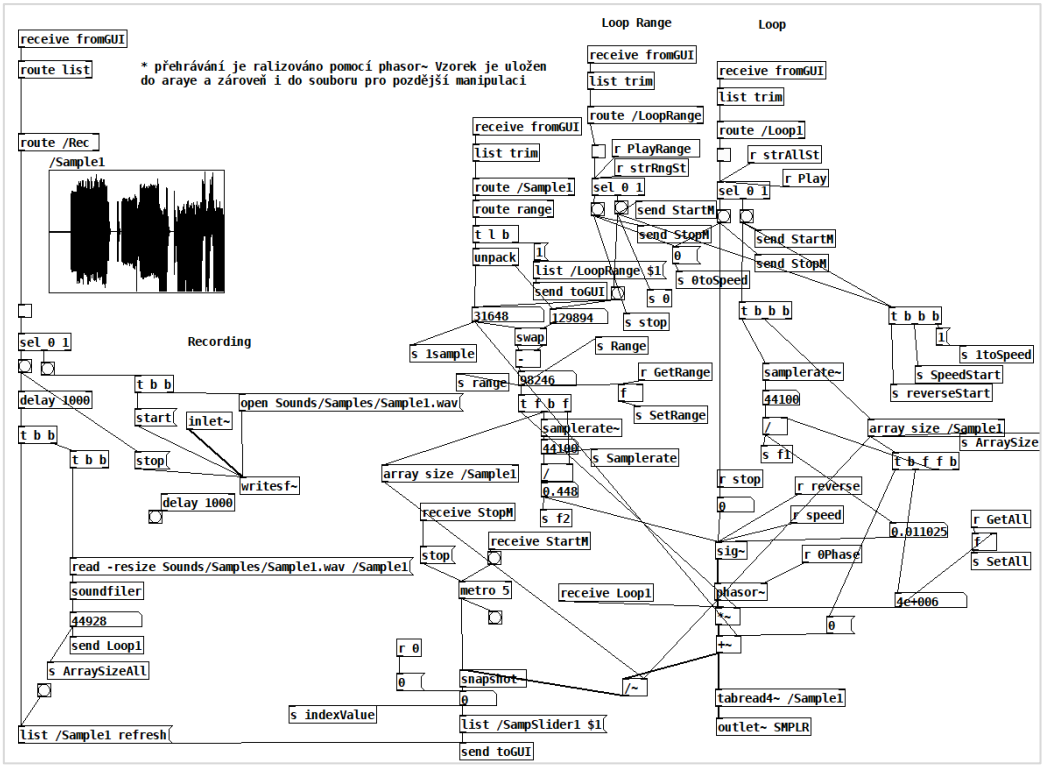


a) b)

obr. 3-13: a) Phaser b)Delay

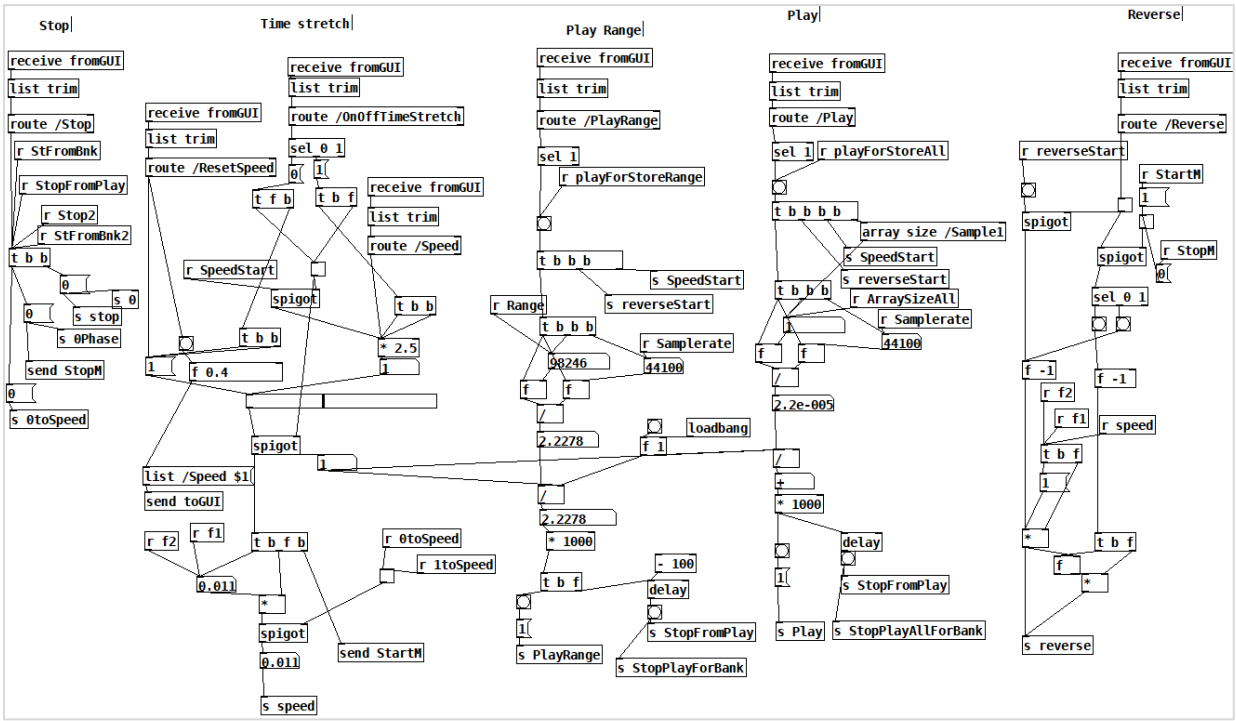
3.1.10 Sampler

Tato sekce je realizována pomocí stejnojmenného subpatche, který slouží pro práci s jedním samplem. Disponuje jedním signálovým inletem pro příjem signálu z *adc~* (mikrofon zařízení) a jedním signálovým outletem. Vstupní úroveň signálu se nastavuje v premix. Nahrávání je založeno na stejném principu jako zachytávání samplů v bankách (kapitola 3.1.7). Při stisknutí spínače record se začne zapisovat signál do souboru Sounds/Samples*/Sample1*.wav. Po ukončení streamu se soubor načte do pole */Sample*1. Nakonec je do GUI odeslána zpráva *list /Sample1 refresh* pro obnovení waveformu v panelu. Tohle dvojí ukládání samplu prvně do pevné paměti zařízení a nakonec prostřednictvím pole do paměti RAM je zvoleno, protože při vypnutí aplikace se obsah pole z paměti RAM vymaže, kdežto poslední zaznamenaný sampl do pevné paměti zůstává uložen a při opětovném startu aplikace se opět načte. Tímto způsobem supluji i chybějící funkci Save. V podstatě se o ukládání nemusím starat. To, co je uloženo do paměti zařízení, lze později i načíst.Sample lze přehrát jednou nebo ve smyčce, a to buď celý sampl nebo jen výběr *range*. Proto je v GUI povolena funkce výběru určité částí samplu *range*. Přehrávání je založeno na objektu *phasor~,* který je ideální pro přehrávání smyčky, neboť jak už bylo zmíněno, generuje pilový signál nabývající hodnot 0 – 1. Tato možnost je zvolena mimo jiné i proto, že pro samotné načítání vzorku z pole je použit objekt *tabread4~*. Na jeho horký vstup musí přicházet signál, který určuje jaký index pole se má číst a jak rychle. Při přehrávání ve smyčce je tedy nejprve zjištěna velikost pole *array size /Sample1*. Dále je zjištěna vzorkovací frekvence a ta je podělena velikostí pole. Tím je získána frekvence pro přehrávání. Aby bylo možné sampl v reálném čase přehrát, násobí se výstupní signál phasoru velikostí pole.



obr. 3-14: Sampler

Signál pak pokračuje dále na výstup. Při přehrávání výběru *range* je princip totožný jen s tím rozdílem, že velikost pole je určená rozdílem hodnot, které udává levý a pravý lokátor výběru. Vzorkovací frekvence je totožná. Klasické přehrávání je realizováno pomocí dělení délky pole a vzorkovací frekvence. Po této operaci dostaneme dobu trvání přehrávání. Následně se odešle bezdrátově povel pro běžné přehrávání ve smyčce a hodnota trvání v milisekundách do objektu *delay*. Ten po uplynutí obdrženého času pro zpoždění odešle událost bang pro povel zastavení přehrávání. Kurzor pozice přehrávání je realizován pomocí objektu *snapshot~,* který převádí signál na číslo. Do něj vstupuje signál odesílaný na výstup spolu se signálem z objektu *metro5*. Výstupem snapshotu je pak hodnota, která nabývá od 0 do 1 rychlostí danou délkou přehrávaného samplu. Tato hodnota se poté odesílá pomocí *list /IndexSliderT1 $1* na indexslider, který ukazuje pozici přehrávání. Subpatch Sampler umožňuje měnit rychlost přehrávání tzv. *time stretch* a tím transponovat sampl. Time stretch se provádí tak, že do objektu *sig~* vstupuje hodnota, která odpovídá násobkům použité frekvence. Tuhle hodnotu nastavuje slider *speed*. Signál se tím pádem nadvzorkuje případně podvzorkuje. Při hodnotě 0 neprobíhá žádné přehrávání ani generování signálu z objektu *phasor~*. Funkce time stretch se dá vypnout a zapnout pomocí *OnOffTimestretch,* případně resetovat pomocí tlačítka *reset speed* tak, že se pošle hodnota 1 na vstup objekt *sig~*. Kromě změny rychlosti umožňuje subpatch i přehrávání pozpátku. Toho docílíme, pokud frekvenci přehrávaného signálu násobíme (*- 1*). Výstupní signál ze subpatche je přes signálový outlet směřován do subpatche premix.



obr. 3-15: Sampler

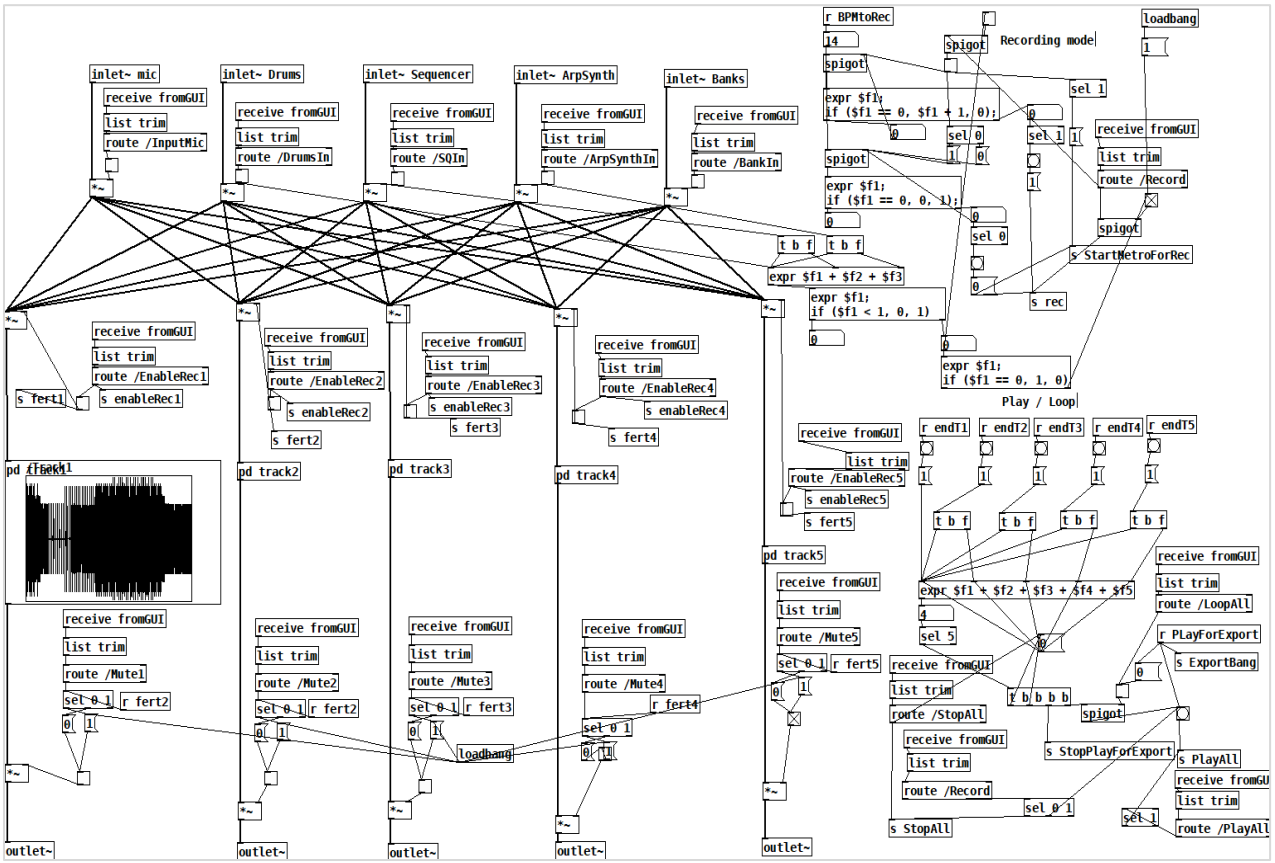
3.1.11 Looper/Recorder

Subpatche Looper/Recorder zajišťuje nahrávání jak z mikrofonu zařízení, tak ze všech nástrojů a bank. Je také možné do něj importovat nové stopy nebo posílat nahrané stopy do subpatche *Sampler/Looper*. Jeho hlavní funkcí je ale nahrávání smyček a následné seřazení tracků pro export do jedné stopy.

Subpatch disponuje pěti signálovými inlety, do kterých jsou přivedeny výstupy ze subpatche premix, což jsou větve signálu. První z leva je přímý signál z mikrofonu (v subpatchi premix veden přes větev *hearback)*, na druhý vstup je přiveden *DrumMachine,* na třetí *XYsliderSequencer,* na čtvrtý *Arpeggiator synth* a na páty *Banks*. Inlety lze pomocí audio přepínačů (*~), přiřadit k jakékoliv nahrávací větvi. Vstupní úroveň signálu je nastavována právě v sekci premix. Směřování signálu do nahrávacího subpatche *„pd track…“* povoluje tlačítko *„EnableRec“* opět pomocí signálového přepínače *~ s připojeným tooglem na studeném vstupu. Signál se při zapnutém spínači dostává dále do subpatche, kde probíhá záznam. Subpatch *track* disponuje jedním signálovým inletem, který vstupuje do objektu *writesf~* . Ten slouží pro zápis do souboru *Sounds/Samples/Track1.wav*. Princip záznamu je stejný jako v subpatchi *Sampler* a *Banks* (kapitola 3.1.10 a 3.1.7). Po příjmu hodnoty jedna z adresy *Record* se hodnota odešle bezdrátově pomocí „*r rec“* do všech subpatchů track a dále na objekt spigot. Pokud je tam přijata hodnota 1 od přepínače *EnableRec,* tak je objekt spigot otevřen a může jím projít hodnota, která je směřována do objektu „*sel 0 1“,* kde pokud je na vstupu 1 spustí se přehrávání. Pokud je naopak přijata hodnota 0, nahrávání se zastaví. Pro subpatche DrumMachine, XYsliderSequencer a Arpeggiator synth je ale naprogramován speciální režim nahrávání. Pokud je alespoň jeden z těchto nástrojů přiřazen do nahrávací větve, přepne se pomocí objektů *expr* a *spigot* režim nahrávání, který při spuštění nahrávání spustí zároveň hlavní metronom, nebo vrátí spuštěný metronom na začátek. Při vypnutí nahrávání je pomocí objektu *expr* hlídáno dokud *indexslider* hlavního metronomu nedojde nakonec fronty, aby se mohlo ukončit nahrávání. Díky tomuto režimu nahrávání na sebe zaznamenané smyčky plynule navazují.

Přehrávání konkrétního tracku pak umožňuje objekt *tabplay~* . Ten se spouští pomocí události bangu za určitých podmínek, které popíšu později. Při spuštění přehrávání pomocí bezdrátového vstupu se nejprve odešle hodnota vzorkovací frekvence na horký vstup objektu a poté na studený vstup hodnota velikosti pole. Tím se spočítá frekvence zaznamenaného signálu. Ta se dále odešle na vstup objektu *phasor~,* který produkuje pilový signál o hodnotách 0 až 1. Při zapnutém nahrávání se zároveň přehrávají i ty stopy, na nichž není zapnut record enable. Při jakémkoliv přehrávání se vždycky přehrávají všechny samply. Přehrávat ve smyčce lze jednotlivé stopy a to tak, že pokud je zvolena volba *loop* určité stopy, tak je otevřen objekt *spigot* v této stopě, který propustí událost bang přijatou z objekt *taplay~* po ukončení přehrávání a tu odešle na vstup tohoto objektu, čímž se spustí opět přehrávání. Ztlumit určitou stopu lze pomocí přepínače *Mute*. Ten je umístěn za výstupem z každého subpatche *track.* Mute umožňuje přerušení cesty, díky čemuž se signál nedostane na výstup. Funkce *Upload track to sampler* je realizována tak, že při výběru stopy v menu se odešle *bang,* který spustí

zápis do souboru *Sounds/SAmples/Sample1.wav* právě zvolené stopy a následně se do pole /*Sample1* načte příslušná stopa. V rámci subpatche Looper/Recorde je definovaný ještě subpatche *Upload Banks.* Ten slouží pro načítání samplů z Bank zvuků do zvolené stopy. Vývěr stopy se provádí pomocí přepínače *EnableRec.* Při jeho sepnutí se zároveň odešle hodnota do subpatche *Upload Banks* k příslušnému tracku. Hodnota může být 0 nebo 1 a otevírá nebo zavírá stavidlo spigot, které následně umožňuje průchod události bang vyslané po výběru banky z menu *LoadBank.* Bang spustí přepisování, do pole tracku je pomocí triggeru nejprve načtena zvolená banka *read –resize Sounds/Samples/Bank.wav /Track* a následně se nový track přepíše v paměti *write Sounds/Samples/track.wav.* V obou případech pomocí objektu *soundfiler*. Signál vystupující z toho subpatche je dále směrován na subpatche *Mix Export.*

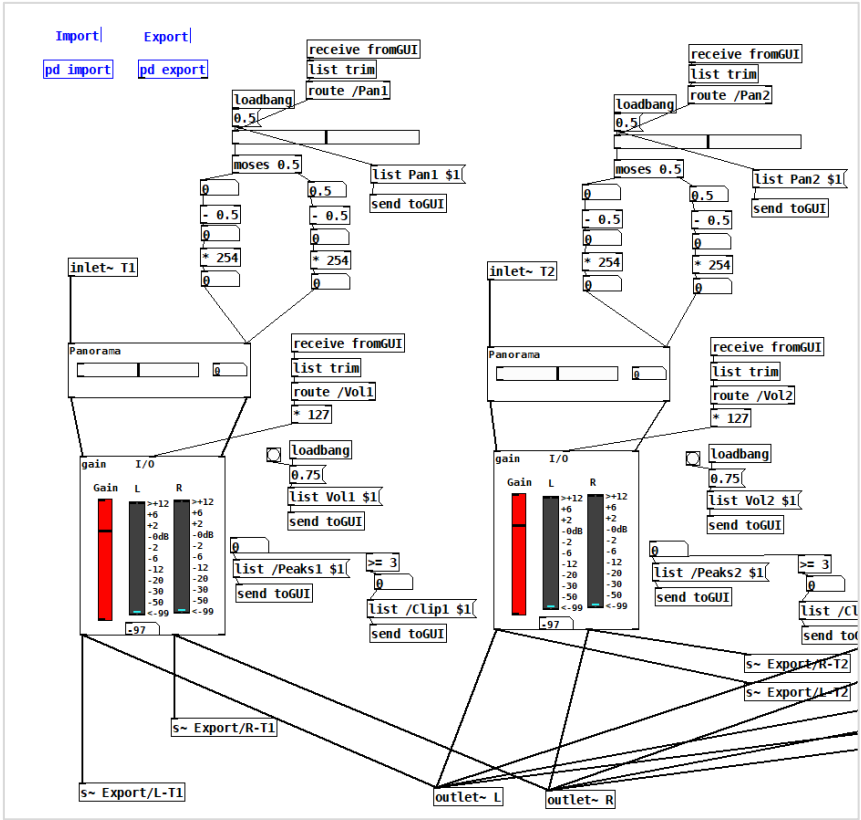


obr. 3-16: Looper/Recorder

3.1.12 Mix Export

Subpatch *„pd Mix/Export/Import“* je spojem přímo ze subpatchem Looper/Recording a slouží pro jednoduchý mix a následný export. Signál z každého tracku subpatche Looper/Recording vstupuje do subpatche *export* svým přiděleným inletem. Prochází přes abstrakci *panorama*, kde je možné signál měnit ve stereo bázi. Jelikož je v GUI ovládacím prvkem panoramy *knob*, který má nastaven rozsah hodnot 0 – 1 prochází vstupní hodnota přes objekt *moses.* Zde je rozdělena napůl, poté je od ní odečtena hodnota 0,5 a nakonec je hodnota vynásobena * 254. Po tomto přepočtu může vstupní

hodnota do abstrakce nabývat požadovaných hodnot –127 až +127. Dále signál pokračuje přes abstrakci *gain,* která nastavuje výstupní úroveň signálu. Výstup z abstrakce gain je dále směřován bezdrátově do subpatche *„pd export“* a zároveň na dva outlety pro pravý a levý kanál, odkud se pokračuje dále na MasterFader. Signál vstupující do subpatche *„pd Exports*“, je směřován do klonu master faderu a z něj vystupující levý a pravý kanál, které jsou připojeny na objekt *writesf~* pro zápis. Jak již bylo zmíněno, sepnutím tlačítka export se vyvolá dialogové okno pro zadání jména souboru. Stisknutím tlačítka export se odešle *bang* na zprávu *start,* což je pokyn pro začátek streamu a současně se odešle událost *Bang* bezdrátově do subpatche Looper/Recorder, kde se spustí přehrávání od začátku. Přehrávaný signál se tímto způsobem začne zapisovat do souboru wav. Po skončení přehrávání se odešle událost *bang* na zprávu *stop* a stream se tím ukončí. V Subpatchi *„pd Mix/Export/Import“* je umístěn i subpatch pro import *„pd import“.* V něm jsou pomocí objektu „*route /ImportToLooper*/*ImportToSampler“* přijímány povely z objektů v GUI. Příchozí povely prochází přes objekt *expr,* kde se rozhoduje, jestli přišel povel pro import souboru do Sampleru nebo do konkrétní stopy Looper/Recorderu. Jakmile je rozhodnuto, vyvolá se dialogové okno s požadavkem zadání názvu souboru pro import. Jakmile je soubor vybrán, je pomocí objektu *read -resize/$1.wav /...* načten do příslušné stopy nebo Sampleru. Na závěr se odešle zpráva o nahrání souboru do GUI.



obr. 3-17: Mix track1 a track 2

Závěr

V rámci bakalářské práce byla navržena a naprogramována aplikace jakožto experimentální softwarový hudební nástroj na platformě Android kombinující sampler, syntetizér a sekvencér. Práce se skládá ze třech částí a to: teoretického úvodu, návrhu grafického rozhraní a realizace.

V teoretické části byla stručně probrána teorie vzniku zvuku a lidského hlasu. Popsal jsem digitální zpracovávání zvukového signálu a základní metody syntéz zvuku. Probral jsem efekty, které využiji v mé práci. Dále jsem se zabýval funkcemi syntetizéru a sampleru, jelikož z jejich funkčností vychází má aplikace. Stručně jsem popsal strukturu operačního sytému Android a uvedl jsem charakteristiku programovacího jazyka Pure Data, který využívám pro svou práci. Ten jsem si vybral, protože je vhodný pro práci se signálovými toky. Je koncipován vizuálně a programátor vidí a slyší změny v reálném čase. Kód, resp. program není potřeba kompilovat. Díky tomu jdou poměrně rychle vytvářet jednotlivé funkční části. Stručně jsem popsal i GUI platformu MobMuPlat. Tu jsem si vybral pro její jednoduchost a možnosti propojení s objekty vytvořenými v PD.

V části Návrh jsem se zabýval původním návrhem funkčnosti aplikace, z kterého se mělo vycházet při realizaci aplikace jakožto softwarového hudebního nástroje. Potom jsem představil nový návrh a grafický návrh grafického rozhraní, kde jsem vysvětlil funkčnost a princip jednotlivých sekcí, ze kterých se aplikace skládá.

V části realizace byla zrealizována aplikace a popsána funkce jednotlivých částí. Při realizaci jsem zprvopočátku vycházel z návrh obsaženého v semestrální práci. Postupem času se ale ukázalo, že některé navrhované postupy byly poněkud komplikované na realizaci. Proto jsem rozhodl návrh mírně poupravit. Byl jsem nucen některé vlastnosti vynechat, jiné jsem ale zase přidal.

Aplikace se skládá z těchto funkčních částí: bicího automatu, sekvencéru, arpeggiator-syntetizéru, bank zvuků, vestavěných efektů, Sampleru, Looper/Recorderu a pomocných mixpultů a výstupních rozhraní. Bicí automat disponuje mřížkou pro vyvolávání samplů. Délka mřížky odpovídá délce možné sekvence, tj. 16 políček. Řádků je celkem šest a jeden řádek odpovídá jednomu zvukovému vzorku. V jednotlivých řádcích lze zvukové vzorky měnit. Přehrávání sekvence je odvozeno od tempa vestavěného metronomu. Sekvencér pak umožňuje synchronně s bicím automatem přehrávat sekvenci šestnácti tónů, jejichž výšku lze měnit pomocí 16 sliderů a díky vestavěnému XY slideru lze výstupní zvuk modulovat a filtrovat. Navíc je možné vybrat si ze čtyř zvukových generátorů. Arpeggiator synth kombinuje funkci arpeggiatoru a syntezátoru s kontrolérem. Obsahuje také čtyři zvukové generátory, jejichž průchozí signál může být modulován pomocí frekvenční a kruhové modulace a jejich kombinací. Pro vygenerování arpeggia slouží kontrolér, což je multidotykový panel, který simuluje klávesy. Jejich teoretický rozsah je 8 oktáv. Kromě generování arpeggia je klávesy možné používat i pro klasické hraní. Nástroj může být rytmicky synchronizován s předcházejícími. Zajímavou vlastností nástroje je, že umí reagovat na nahýbání zařízení, díky čemuž lze upravovat průchozí signál.

Nahýbáním v ose x se signál filtruje, nahýbáním v ose y je možné měnit oktávu nebo míru modulace. Nástroj disponuje také vestavěným efektem Delay. Banky zvuků umožňují zachytávání zvukových vzorků nahraných pomocí sampleru a jejich následné ukládání. Uložené zvuky lze pak vyvolávat pomocí padů nebo je možné je poslat do konkrétních řádků v bicím automatu. Sampler slouží pro nahrávání zvukových vzorků a jejich ukládání do bank. Umožňuje nahrát sampl z mikrofonu zařízení. Z nahraného zařízení lze vyselektovat určitý výběr. Sampler umožňuje přehrávání ve smyčce, reverzní přehrávání a funkci time stretch. Je z něj možné odbočit do efektové větve. Efektová větev se skládá z efektů: ekvalizér, distortion, vocoder, FM synth, phaser a delay. Každý nástroj disponuje vlastní efektovou větví. Takže kromě sampleru do ní mohou odbočit všechny zmiňované nástroje. Zadané parametry efektu zůstávají uloženy i při přepínání mezi větvemi. Odbočování do efektových větví je možné s pomocí vestavěného mixpultu. Ten může upravovat i výstupní signál jednotlivých nástrojů. Looper/Recorder umožňuje nahrávání smyček a sekvencí jednotlivých zmiňovaných nástrojů a vytváření skladeb. Do Looper/Recorderu je možné i importovat nové soubory nebo banky zvuků. Pomocí vestavěného mixpultu lze upravovat výstupní signál a panoramu jednotlivých stop Looper/Recorderu a následně je všechny exportovat do souboru. Vyexportované skladby je možné znovu naimportovat a dále zase upravovat.

Aplikace je schopná provozu na telefonech a tabletech s operačním systémem Android. Díky MobMuPlat by aplikaci mělo jít spustit i na zařízeních s iOS, nicméně jsem to netestoval. Pomocí editoru je samozřejmě možné spustit aplikaci i v operačním systému Windows.

Aplikaci jsem realizoval v programovacím jazyku Pure Data, přičemž jsem se opíral o znalosti nastudované z odborné literatury, nebo jsem využíval volně dostupných tutoriálů na internetu. Při samotné realizaci se mi vyskytlo několik chyb a komplikací, které mě velmi zdržely, nicméně se mi je nakonec podařilo odstranit. Hardwarová náročnost je ale o něco větší než jsem očekával. Je to způsobeno právě rozšířenou prací se zvukovými vzorky. Všechny jsou ve formátu wav, což je formát o větších datových velikostech. Při provozu si aplikace díky tomu zabírá mnoho místa na paměti RAM. Navíc jazyk Pure Data není prvoplánově určen na realizaci takto rozměrných aplikací na přenosných zařízeních, ale je určen spíše pro laptopová a desktopová řešení. Vhodnější by bylo naprogramovat aplikaci v C++ s pomocí JUCE. Ten je přímo určen pro vytváření pluginů a aplikací zpracovávajících zvuk. Možná by taková aplikace šla realizovat i v herním enginu UNITY. Nicméně obě tyto možnosti jsem objevil, až když jsem měl určité části aplikace naprogramované pomocí PD. Takže už jsem neuvažoval o tom, že bych začal od začátku. Zadáním práce ale bylo zhotovit aplikaci jako experimentální hudební nástroj a požadavky na hardwarovou náročnost nebyly zadány. Navíc Pure Data mají i své výhody. Nabízejí mnoho možností pro práci se zvukem a díky tomu, že se zde vyloženě nepíše kód, vidí člověk hned výsledek. V kombinaci s MobMuPlat můžou být Pure Data nakonec silným nástrojem pro dosažení zajímavých výsledků. Mé pojetí aplikace je postaveno právě na kombinaci několika elektronických nástrojů a pomocí PD a MobMuPlat jsem ji mohl dát mnoho

zajímavých vlastností, které z ní dělají obstojný experimentální softwarový hudební nástroj na platformě Android. Nástroj kombinuje sampler, sekvencér a syntetizér přesně tak, jak je napsáno v zadání, nabízí řadu zajímavých funkcí a využívá jedinečné vlastnosti chytrých telefonů a tabletů. Využití nástroje může být mnoho. Například jako multifunkční mobilní stanice pro zachytávání vzorků, vytváření sekvencí a skladeb nebo jako sampler pro vytváření vzorků, které se dají potom dále zpracovávat i pomocí jiných zařízení a efektů. Zajímavou možností je využití nástroje jako syntezátoru a generátoru zvuků, pro externí efekty, nebo pro souhru s jinými nástroji. V podstatě jde o více nástrojů v jednom nebo jeden nástroj s mnoha možnostmi, který může mít člověk neustále u sebe. Na základě toho se domnívám, že jde o poměrně originální řešení softwarového nástroje. Něco podobného jsem já osobně na chytrých telefonech a tabletech dosud nezaznamenal.

Seznam použité literatury

[1] SYROVÝ, Václav. *Hudební akustika*. Praha: Akademie múzických umění, 2003. Akustická knihovna (Akademie múzických umění v Praze. Hudební fakulta. Zvukové studio). ISBN 8073319012

[2] SYROVÝ, Václav a Milan GUŠTAR. *Malý slovník základních pojmů z hudební akustiky a hudební elektroniky*. 2. vyd. V Praze: Akademie múzických umění, 2012. ISBN 9788073312374.

[3] SYROVÝ, Václav. *Hudební zvuk: příspěvek k teorii zvukové tvorby*. V Praze: Akademie múzických umění, 2009. Akustická knihovna Zvukového studia Hudební fakulty AMU. ISBN 9788073311612

[4] SCHIMMEL, Jiří. *Elektroakustika*. Brno: Vysoké učení technické v Brně Fakulta elektrotechniky a komunikačních technologií Ústav telekomunikací, 2016. ISBN 978-80-214-4716-5.

[5] TEOCHARISOVÁ, Vanda. *Sound design: zvuková syntéza a tvůrčí programování zvuků v praxi*. 1. Praha: Muzikus, c2009. ISBN 9788086253534.

[6] SCHIMMEL, Jiří. *Studiová a hudební elektronika*. 2. Brno: Vysoké učení technické v Brně Fakulta elektrotechniky a komunikačních technologií Ústav telekomunikací, 2015. ISBN 978-80-214-4452-2.

[7] SCHIMMEL, Jiří. *Studiová a hudební elektronika*. 1. Brno: Vysoké učení technické v Brně Fakulta elektrotechniky a komunikačních technologií Ústav telekomunikací, 2015. ISBN 978-80-214-4452-2.

[8] *Analýza signálů a soustav - BASS*. Brno: Vysoké učení technické v Brně Fakulta elektrotechniky a komunikačních technologií Ústav komunikací, 2012. ISBN 978-80-214-4453-9.

[9] RUSS, Martin. *Sound synthesis and sampling*. Oxford ; Boston: Focal Press, 1996. ISBN 02-405-1429-7.

[10] BOULANGER, Richard Charles. *The Csound book: perspectives in software synthesis, sound design, signal processing, and programming*. Cambridge, Mass.: MIT Press, c2000. ISBN 0262522616.

[11] SÝKORA, Rudolf, Jaroslav VČELAŘ a František KRUTÍLEK. *Elektronické hudební nástroje a jejich obvody*. Praha: Státní nakladatelství technické literatury, 1981. Řada elektrotechnické literatury.

[12] ALLEN, Grant. *Android 4: průvodce programováním mobilních aplikací*. Brno: Computer Press, 2013. ISBN 9788025137826.

[13] KAVAN, Jan. *Pure Data: platforma pro tvorbu interaktivního díla*. Brno: Janáčkova akademie múzických umění v Brně, 2013. ISBN 9788074600333.

[14] *Jak šel čas s operačním systémem android* [online]. 2015. Dostupné z: http://www.androidtip.cz/jak-sel-cas-s-operacnim-systemem-android/

[15] Operační systém Android. *Metodický Portál* [online]. 2012 [cit. 2016-12-13]. Dostupné z: http://clanky.rvp.cz/clanek/c/G/15449/operacni-system-android.html/

[16] Úvod do programování v androidu. *ITnetwor.cz* [online]. [cit. 2016-11-30]. Dostupné z: http://www.itnetwork.cz/java/android/tutorial-programovani-pro-android-v-jave-uvod

[17] *Pure Data* [online]. 2013 [cit. 2016-12-1]. Dostupné z: https://puredata.info/

[18] *MobMuPlat* [online]. 2014 [cit. 2017-12-13]. Dostupné z: http://danieliglesia.com/mobmuplat/

[19] Audacity aneb jak zviditelnit zvuk. *Souhrnný sborník Veletrhu nápadů učitelů fyziky* [online]. [cit. 2016-12-14]. Dostupné z: http://vnuf.cz/sbornik/prispevky/17-17-Jerje.html

[20] IGLESIA, Daniel. The Mobility is the Message: the Development and Uses of MobMuPlat. *Http://www.danieliglesia.com/mobmuplat* [online]. [cit. 2018-05-23]. Dostupné z: http://www.danieliglesia.com/mobmuplat/IglesiaMobMuPlatPaper.pdf

[21] KREIDLER, Johannes. *Programming Electronic Music in Pd* [online]. [cit. 2018-05-25]. Dostupné z: http://www.pd-tutorial.com/english/index.html

[22] *Guitar Extended* [online]. [cit. 2018-05-27]. Dostupné z: https://guitarextended.wordpress.com/category/software/pure-data/

Použité zkratky

ADR - android runtime

AM - amplitude modulation (amplitudová modulace)

BPM - Beats per Minute (počet úhozu metronomu za minutu)

CPU – Central Processing Unit

DFP - Data Flow Programing

Efx. - Efekty efektová sběrnice

EQ - Ekvalizér

FM - frequency modulation (frekvenční modulace)

GPS -Global Positioning System

GUI - Graphical user interface (grafické uživatelské rozhraní)

MIDI - Musical Instrument Digital Interface (digitální rozhraní pro hudební nástroje)

PD - Pure Data

RAM – Random access memory

SDK - Software development kit

USB - universal serial bus (univerzální sériová sběrnice)

VCA - Voltage Cotrol Amplifer

VCF - voltage controlled filter (napětím řížený filtr)

Vol - Volume

VST - virtual studio technology (technologie virtuálního studia)

Wav. - Waweform audiofile formát

Seznam příloh

A Obsah přiloženého DVD

Na přiloženém DVD se nachází v adresáři *„Projekt BC“* PD soubor *„ProjektSamLopSeq_V2.2.pd“*, což je hlavní patch aplikace. Ve stejném adresáři se nachází soubor mmp *„SLSQProjekt4.mmp“*, což je soubor platformy MobMuPlat, spustitelný v editoru nebo v chytrém telefonu a tabletu s operačním systémem Android v aplikaci MobMuPlat. Ve stejné složce se nachází i abstrakce a složka se zvukovými soubory, potřebnými k chodu aplikace.

V kořenovém adresáři DVD se dále nachází dokumentace k instalaci a spuštění aplikace v operačním systému Android a Windows. Ve složce MobMuPlat se nachází editor MobMuPlat, potřebný ke spuštění aplikace ve Windows. Nakonec se zde nachází soubor bakalářské práce ve formátu PDF.

B Příručka pro uvedení do provozu

Postup pro uvedení do provozu na PC.

1. Nejprve je třeba extrahovat všechny soubory a adresáře
2. Následně spustit Java editor MobMuPlat. (**MobMuPlatEditor.jar**). Ten se nachází v adresáři *\MobMuPlat\MobMuPlatDistribution_1.80iOS_0.30Android\Editor\CrossPlatformJava* Po spuštění editoru je třeba dále v editoru otevřít soubor **SLSQProjekt4.mmp** tím se otevře GUI aplikace.
3. Po té je třeba spustit PD vanila a otevřít PD file **PDWrapper.pd**, který se nachází v adresáři *...\MobMuPlat\MobMuPlatDistribution_1.80iOS_0.30Android ...* Ten zajišťuje propojení aplikace a editoru.
4. Nakonec je třeba otevřít samotný „engine“ aplikace PD patch **ProjektSamLopSeq_V2.2.pd** ten se nachází v adresáři … \Projekt BC

*pozn: v iOS by měl být postup obdobný, jen se tam spouští jiný editor.

Postup pro uvedení do provozu na operačním systému Android

1. Stáhnout z Google Playstore aplikaci MobMuPlat
2. Do adresáře MobMuPlate na hlavním uložišti zařízení je třeba nakopírovat soubor **SLSQProjekt4.mmp,** dále pak adresář **sounds** hlavní PD patch **ProjektSamLopSeq_V2.2.pd** a všechny abstrakce.
3. Spustit aplikaci MobMuPlat a otevřít soubor **SLSQProjekt4.mmp**

*pozn: návod pro iOS je uveden na http://www.danieliglesia.com/mobmuplat/doc/index.htm

MobMuPlat je freeware platforma (od iglesia intermedia) postavena na LibPD. Informace

Zde: http://www.danieliglesia.com/mobmuplat/IglesiaMobMuPlatPaper.pdf

a Zde: http://www.danieliglesia.com/mobmuplat/