TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

APLIKACE PRO ZPRACOVÁNÍ AUDIO SIGNÁLŮ V OFF-LINE A ON-LINE REŽIMU

Bakalářská práce

Studijní program: B2612 – Elektrotechnika a informatika
Studijní obor: 2612R011 – Elektronické informační a řídicí systémy

Autor práce: **Václav Strašil**
Vedoucí práce: doc. Ing. Zbyněk Koldovský, Ph.D.

Liberec 2014

TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky a mezioborových studií
Akademický rok: **2013/2014**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Václav Strašil**
Osobní číslo:	**M11000174**
Studijní program:	**B2612 Elektrotechnika a informatika**
Studijní obor:	**Elektronické informační a řídicí systémy**
Název tématu:	**Aplikace pro zpracování audio signálů v off-line a on-line režimu**
Zadávající katedra:	**Ústav informačních technologií a elektroniky**

Zásady pro vypracování:

1. Nastudujte způsoby realizace standardních digitálních efektů (např. delay, chorus, flanger, vibrato, pitch shifter, time stretching, atd.)
2. Vytvořte aplikaci v .NET v jazyce C# pro editaci zvukových záznamů v režimu off-line.
3. Doplňte aplikaci o způsob zpracování signálů v režimu on-line.
4. Vytvořte dokumentaci k software a zvukové ukázky.

Rozsah grafických prací: **Dle potřeby dokumentace**

Rozsah pracovní zprávy: **cca 30 stran**

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam odborné literatury:

[1] **B. Porat, "A Course in Digital Signal Processing", John Wiley & Sons, 1997.**

[2] **U. Zölzer et al., "DAFX Digital Audio Effects", John Wiley & Sons, 2002.**

Vedoucí bakalářské práce: **doc. Ing. Zbyněk Koldovský, Ph.D.**
Ústav informačních technologií a elektroniky

Konzultant bakalářské práce: **Ing. Jiří Málek, Ph.D.**
Ústav informačních technologií a elektroniky

Datum zadání bakalářské práce: **12. září 2013**

Termín odevzdání bakalářské práce: **16. května 2014**

prof. Ing. Václav Kopecký, CSc.
děkan

Technická univerzita v Liberci
L.S.
- 14 -

prof. Ing. Zdeněk Plíva, Ph.D.
vedoucí ústavu

V Liberci dne 12. září 2013

Prohlášení

Byl jsem seznámen s tím, že na mou bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci (TUL) nezasahuje do mých autorských práv užitím mé bakalářské práce pro vnitřní potřebu TUL.

Užiji-li bakalářskou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti TUL; v tomto případě má TUL právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Bakalářskou práci jsem vypracoval samostatně s použitím uvedené literatury a na základě konzultací s vedoucím mé bakalářské práce a konzultantem.

Současně čestně prohlašuji, že tištěná verze práce se shoduje s elektronickou verzí, vloženou do IS STAG.

Datum: 13.5.2014

Podpis: Strašil

Poděkování

Tímto bych rád poděkoval doc. Ing. Zbyňkovi Koldovskému Ph.D. za vedení práce a konzultace. Také děkuji rodině za podporu při studiu.

Abstrakt

Tato práce se zabývá principem a využitím několika vybraných audioefektů a jejich realizaci jak v režimu off-line, tak v režimu on-line. Výstupem práce je aplikace naprogramovaná jazykem C#, spustitelná na počítači s operačním systémem Microsoft Windows. Implementovány byly efekty echo, vibrato, flanger, chorus, doubling, wah-wah, phaser, rotary, ring-modulator, tremolo, ekvalizér a filtrování signálu pomocí FIR filtrů. Tyto efekty lze aplikovat jak na audiosoubor s příponou .mp3 nebo .wav, tak i v reálném čase na signál z mikrofonového vstupu zvukové karty. Dále obsahuje algoritmus SOLA umožňující time-stretching a pitch-shifting v offline režimu. Aplikace dovoluje soubor otevírat, přehrávat, ukládat, zkracovat, zobrazit jeho hlasitost v závislosti na čase či spektrogram.

Klíčová slova

DSP, Audioefekty, Time-stretching, Pitch-shifting, Editace zvuku

Abstract

This work deals with the principle and usage of a few selected audio effects and their implementation in both the off-line and in on-line mode. The outcome of this work is an apllication written in C# language, which is executable on a computers running Microsoft Windows. The implemented effects are: echo, vibrato, flanger, chorus, doubling, wah-wah, phaser, rotary, ring-modulator, tremolo, EQ and signal filtration with FIR filters. These effects are applicable to both audio files with .mp3 or .wav extension or to the real-time signal acquired from the input of a sound card. It also contains the SOLA algorithm providing time-stretching and pitch-shifting in the offline mode. The application allows users to open, play, save and shorten files and it also shows volume and spectrogram of a loaded signal.

Keywords

DSP, Audioeffects, Time-stretching, Pitch-shifting, Sound editation

Obsah

Seznam obrázků

Seznam tabulek

Seznam zkratek

FIR - Finite Impulse Response, filtr s konečnou impulzní odezvou

IIR- Infinite Impulse Response, filtr s nekonečnou impulzní odezvou

SOLA - Synchronous OverLap and Add, algoritmus používaný pro time-stretching

A/D - Analog/Digital, převodník z analogového signálu na digitální

FPGA - Field Programmable Gate Array

MCU - MicroController Unit

1 Úvod

Používání audio efektů není běžné pouze pro hudebníky, zvukaře či producenty. Nejobvyklejší použití pro běžného uživatele je například ekvalizér, který dovoluje upravit frekvenční charakteristiku zvuku a přizpůsobit ji tak reproduktoru. Mnoho efektů ocení pak spíše hudebníci. Například efekt Phaser a Wah-Wah je mezi kytaristy poměrně oblíbený. Echo a Chorus najde své uplatnění zejména v akustické hudbě.

Není nutné se však omezovat pouze na hudbu. Pro filmový průmysl je výhodnější simulovat okolní prostředí pomocí těchto efektů uměle, než nahrávat zvuk scény, například přímo v jeskyni. Efekty lze aplikovat také na lidský hlas, díky tomu je možné propůjčit hlas jakékoliv science fiction postavě.

Je tedy patrné, že audio efekty mají široké uplatnění. S rozvojem techniky, zejména díky digitalizaci, se navíc zvyšuje dostupnost takových zařízení. To, na co bylo v minulosti potřeba několik zařízení, stačí dnes jedno. Příkladem by mohli být například kytaristé, kteří před sebou dost často měli i několik desítek analogových „krabiček“ a pedálů. Dnes si však vystačí s jedním digitálním multiefektem s výkonným zvukovým procesorem.

Snahou této práce je pochopit principy několika těchto základních efektů a vytvořit jednoduchý a uživatelsky příjemný program, který umožní použití efektů na audio soubor, popřípadě upravit vzorky přicházející na mikrofonový vstup zvukové karty, upravit je a odeslat na audio výstup zvukové karty.

2 Teoretická část

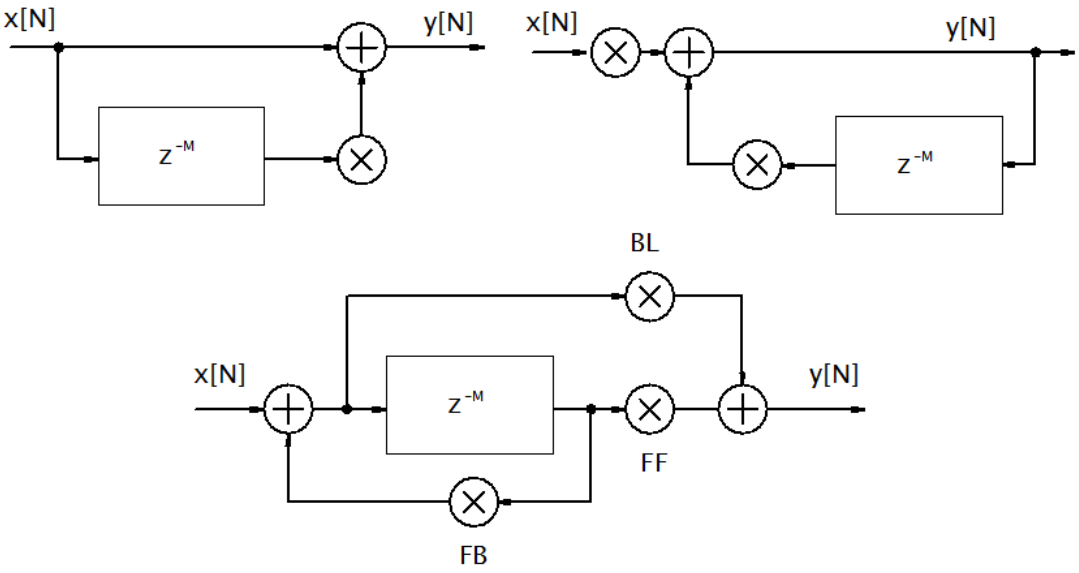
V této části se práce zaměří na vysvětlení principu vzniku a realizaci audioefektů v digitální podobě. Čerpáno bylo především z [1], odkud byly převzaty i některé obrázky, dále pak z [2] a [3].

S vyjímkou efektu rotary jsou vzorce uvedené pro mono signál. Ve všech vzorcích je za *x* považován vstupní buffer a za *y* výstupní buffer. Hodnota F_s je vzorkovací frekvencí signálu.

2.1 Audioefekty s konstantním zpožděním

Tyto efekty jsou založeny na principu ozvěny. Tento jev vzniká, pokud se zvuk odráží od překážky a vrací se zpožděním k posluchači. Zpoždění je závislé na vzdálenosti překážky a na prostředí, ve kterém se pohybuje. Při odrazu od překážky a během průchodu prostředím navíc dochází k utlumení, zpožděný zvuk je tedy oproti původnímu méně hlasitý.

K realizaci se používají 3 základní struktury [1]. První je FIR comb filter (hřebenový filtr s konečnou impulzní odezvou), IIR comb filter (hřebenový filtr s nekonečnou impulzní odezvou) a universal comb filter (univerzální hřebenový filtr).



Obrázek 2.1. Struktury pro realizaci efektů s konstatním zpožděním (vlevo nahoře FIR comb filter, vpravo nahoře IIR comb filter, dole universal comb filter). Převzato z [1].

Zpoždění *M* se spočítá podle vzorce:

$$M = \frac{F_s \times t}{1000} NoCh \qquad \text{(Rovnice 1)}$$

kde: t = zpoždění [ms]

NoCh = počet kanálů signálu

Hřebenový filtr s konečnou impulzní odezvou přičítá ke vstupnímu signálu zpožděný signál o M vzorků vynásobený koeficientem tlumení C_t. Tato struktura je tedy vhodná k simulaci jednoduché ozvěny. Výstup tedy můžeme získat jako:

$$y[n] = x[n] + C_t \times x[n - M] \quad \text{(Rovnice 2)}$$

Hřebenový filtr s nekonečnou impulzní odezvou zpožďuje výstupní signál a násobený koeficientem tlumení C_t ho opět přičítá ke vstupu. Tímto způsobem je tedy ozvěna teoreticky nekonečná - nezmizí, dokud ji koeficient tlumení nepotlačí pod hranici vnímatelnosti. Vypočítáme tedy jako:

$$y[n] = x[n] + C_{tl} \times y[n - M] \quad \text{(Rovnice 3)}$$

Univerzální hřebenový filtr je kombinací obou předchozích. Pokud je nastaveno BL na nulu, výsledkem je hřebenový filtr s nekonečnou impulzní odezvou, pokud je nastaveno FB na nulu, vznikne hřebenový filtr s konečnou impulzní odezvou.

$$z[n] = x[n] + FB \times z[n - M] \quad \text{(Rovnice 4)}$$
$$y[n] = BL \times z[n] + FF \times z[n - M]$$

2.1.1 Echo

Efekt ozvěny vzniká, pokud je překážka vzdálená a ozvěna se k posluchači vrací se zpožděním delším než 100 ms. Za předpokladu, že zdroj zvuku je blízko posluchači a při rychlosti zvuku ve vzduchu (340 m/s) tedy musí být překážka vzdálená přes 17 m, aby tento jev nastal.

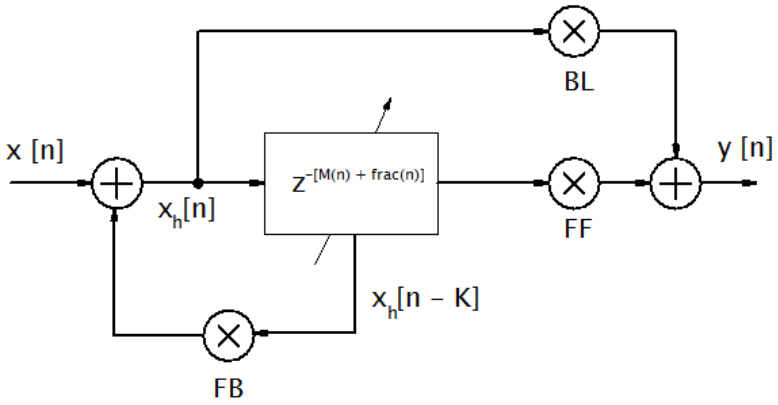
2.1.2 Reverb

Nastává v uzavřených prostorách, kde dochází k mnohonásobnému odrazu a k posluchači se vrací v kratším čase, než by byl vnímán jako ozvěna. Velkou roli zde hraje geometrie prostoru. Profesionální přístroje umožňují simulaci místností například koncertní haly či kostela.

2.2 Audioefekty s proměnným zpožděním

Efekty s proměnným zpožděním zpožďují vstup o nekonstatní hodnotu a přičítají tuto zpožděnou smyčku k násobku původního signálu. Pro svoji proměnnou délku zpoždění se také občas nazývají efekty s proměnnou fází. Pro číslicovou realizaci se

používá struktura univerzálního hřebenového filtru, podobně jako u efektů s konstatním zpožděním, s tím rozdílem, že délka zpoždění se v čase mění. Větev vytvářející IIR hřebenový filtr má však zpoždění konstantní.



Obrázek 2.2. Struktura efektu s proměnným zpožděním. Převzato z [1].

Hodnoty zpoždění mohou nabývat buďto charakteru sinusového nebo náhodného. Sinusový průběh je dán podle vzorce:

$$M[n] = A - A \times \sin\left(2\pi \times frq\ \frac{i}{F_s}\right) \qquad \text{(Rovnice 5)}$$

kde: A = amplituda kmitů, tedy maximální hodnota zpoždění

frq = frekvence kmitání

F_s = vzorkovací frekvence signálu

i = časový údaj v diskrétní oblasti

Pro efekty, využívající náhodný průběh zpoždění, se vygenerují náhodná čísla v intervalu {0; A}, kde A udává maximální zpoždění. Tento signál by však zavedl fázové nelinearity, způsobující nepříjemné praskání, je proto tedy nutné z tohoto náhodného signálu vyfiltrovat vysoké frekvence. Na šum se aplikuje filtr typu dolní propust s mezním kmitočtem v řádu jednotek či několika desítek Hz.

Při implementaci těchto efektů je také nutné myslet na to, že zpoždění nemusí být celočíselné. Protože pracujeme s diskrétním signálem, zavádí se interpolace. Příkladem může být lineární interpolace, která vypočítává zpožděný vzorek s neceločíselným zpožděním ze dvou sousedních – celočíselných. Lineární interpolace je tedy dána vztahem:

$$y[n] = x(n - M) \times frac + x\ (n - [M - 1]) \times (1 - frac) \qquad \text{(Rovnice 6)}$$

Při výpočtu zpožděného vzorku tedy obecně vychází reálné číslo, kde *M* ve vztahu značí jeho celočíselnou část a *frac* desetinnou část.

2.2.1 Vibrato

Efekt vibrato se často využívá na strunných nástrojích, kdy hudebník místo setrvání na jednom tónu drobným napínáním a opětovným povolováním struny (například posouváním struny po hmatníku nástroje nahoru a dolů) lehce zvýší a opět sníží frekvenci kmitání struny, čímž se stává výsledný zvuk pro poslech zajímavější. Podobný princip lze použít u dechových nástrojů či u hlasu zpěváka.

Realizace je z efektů s proměnným zpožděním nejjednodušší, neboť není potřeba zpětných vazeb, pouze je vstupní signál fázově modulován zpožďovačem.

2.2.2 Flanger

Zpoždění flangeru je sinusového charakteru, je však v tak krátkém časovém intervalu, že není pro lidské ucho vnímatelné. Smísením tohoto zpožděného signálu s původním vznikne zajímavý efekt, nazývaný lidově „stíhačka“. Ten můžeme slyšet například v písni Are you gonna go my Way od Lennyho Kravitze, či Ain’t Talkin’ bout Love od skupiny Van Halen.

2.2.3 Chorus

První z audioefektů, využívající náhodného nízkofrekvenčního šumu pro zpoždění. Zpoždění má charakter proměnného dozvuku, což ve výsledku působí „sborově“.

2.2.4 Doubling

Jedná se o efekt, který je podobný chorusu, zpoždění je však vyšší, blížící se spíše ozvěně s lehce proměnným časem, vzniká tedy spíš vjem zdvojení původního signálu.

2.2.5 Shrnutí

Použitím výše uvedené struktury (viz Obrázek 2.2) dosáhneme těchto efektů následujícími hodnotami (dle [1]):

Tabulka 1. Parametry efektů s proměnným zpožděním [1]

Efekt	BL	FF	FB	Zpoždění K [ms]	Hloubka [ms]	Řídící signál pro modulaci
Vibrato	0	1	0	0	0–3	0,1–5 Hz sinus
Flanger	0,7	0,7	0,7	0	0–2	0,1–1 Hz sinus
Chorus	0,7	1	−0,7	1–30	1–30	Nf šum
Doubling	0,7	0,7	0	10–100	1–100	Nf šum

2.3 Efekty amplitudově modulující signál

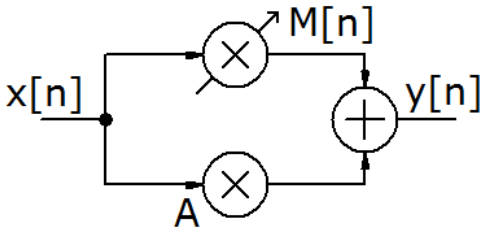
2.3.1 Tremolo

Jde o řízený zesilovač, kdy zesílení je dáno hodnotou sinusového signálu. Tento sinusový signál pak tvoří „obálku“, tedy ovlivňuje amplitudu vstupního signálu. Vznikne tím zvuk, který se harmonicky zesiluje a zeslabuje. Výstupní vzorek získáme podle vzorce:

$$y[n] = A \times x[n] + M[n] \times x[n] \qquad \text{(Rovnice 7)}$$

kde: A = střední hodnota kmitání

M[n] = hodnota modulace (sinus)



Obrázek 2.3 Realizace efektu Tremolo. Do sumátoru vstupuje vstupní signál násobený konstantou A a vstupní signál násobený aktuální hodnotou modulace.

2.3.2 Ring Modulator

Efekt známý občas jako „robot voice“, používáný často ve zpravodajských pořadech k zachování anonymity hlasu mluvícího. Narozdíl od efektu tremolo, zde má sinusový signál úlohu nosné frekvence. Podobného principu se využívá u krátkovlnného přenosu rozhlasu. U toho efektu je však nosná vlna v slyšitelném pásmu. Sloučením nosné frekvence a původního signálu, vzniknou frekvence nové. Efekt proto není vhodné používat na hudební nástroje, protože je rozladí. Výstup je dán vzorcem:

$$y[n] = M[n] \times x[n] \quad \text{(Rovnice 8)}$$

kde: M[n] = hodnota modulace (sinus)

2.4 Stereo efekty

2.4.1 Rotary

Po použití tohoto efektu vzniká dojem rotace signálu kolem posluchače. Efektu rotary se potom docílí díky vzorci:

$$y_L = \cos(\alpha)\, x_L + \sin(\alpha)\, x_R$$
$$y_R = -\sin(\alpha)\, x_L + \cos(\alpha)\, x_R \quad \text{(Rovnice 9)}$$

Použije-li se maticový zápis, lze rovnici napsat jako:

$$\begin{bmatrix} y_L \\ y_R \end{bmatrix} = \begin{bmatrix} \cos(\alpha) & \sin(\alpha) \\ -\sin(\alpha) & \cos(\alpha) \end{bmatrix} \times \begin{bmatrix} x_L \\ x_R \end{bmatrix} \quad \text{(Rovnice 10)}$$

kde: x_L = levý vstupní vzorek

x_R = pravý vstupní vzorek

y_L = levý výstupní vzorek

y_R = pravý výstupní vzorek

Úhel α je počítán podle vzorce:

$$\alpha = \sin\left(2\pi f \frac{i}{F_S}\right) \quad \text{(Rovnice 11)}$$

kde: f = frekvence otáčení kolem posluchače

F_s= vzorkovací perioda

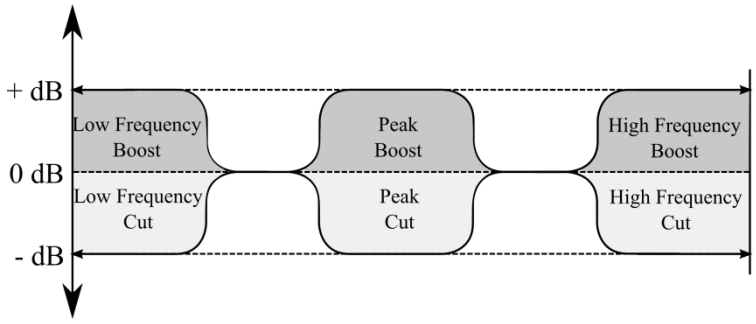
i = časový údaj v diskrétní oblasti

2.5 Ekvalizér

S ekvalizérem je možné se setkat ve většině multimediálních přehrávačích, MP3 přehravačích či u mobilních telefonů. Často výrobce poskytne i řadu přednastavení, vhodných pro určité hudební žánry. Samotný ekvalizér dokáže zvýraznit, či částečně potlačit jistou frekvenční oblast.

2.5.1 Návrh, výpočet koeficientů

Prvním krokem k návrhu ekvalizéru je volba počtu frekvenčních pásem, ve kterých bude probíhat ekvalizace.



Obrázek 2.4. Typy pásem ekvalizéru

Pásmo s nejnižším kmitočtem by mělo být typu Low frequency cut/boost, pásmo s nejvyšším kmitočtem typu High frequency cut/boost, pásma mezi těmito hraničními frekvencemi typu Peak cut/boost. Pokud je zesílení kladné, volí se z tabulky boost, pokud záporné (tlumení) použije se cut. Podle těchto kritérií se vypočítají koeficienty dle následující tabulky.

Tabulka 2. Výpočet koeficientů ekvalizéru. Převzato z [1].

	b0	b1	b2	a1	a2
Low frequency shelving boost	$\frac{1+\sqrt{2V_0}K+V_0K^2}{1+\sqrt{2}K+K^2}$	$\frac{2(V_0K^2-1)}{1+\sqrt{2}K+K^2}$	$\frac{1-\sqrt{2V_0}K+V_0K^2}{1+\sqrt{2}K+K^2}$	$\frac{2(K^2-1)}{1+\sqrt{2}K+K^2}$	$\frac{1-\sqrt{2}K+K^2}{1+\sqrt{2}K+K^2}$
Low frequency shelving cut	$\frac{1+\sqrt{2}K+K^2}{1+\sqrt{2V_0}K+V_0K^2}$	$\frac{2(K^2-1)}{1+\sqrt{2V_0}K+V_0K^2}$	$\frac{1-\sqrt{2}K+K^2}{1+\sqrt{2V_0}K+V_0K^2}$	$\frac{2(V_0K^2-1)}{1+\sqrt{2V_0}K+V_0K^2}$	$\frac{1-\sqrt{2V_0}K+V_0K^2}{1+\sqrt{2V_0}K+V_0K^2}$
High frequency shelving boost	$\frac{V_0+\sqrt{2V_0}K+V_0K^2}{1+\sqrt{2}K+K^2}$	$\frac{2(V_0K^2-V_0)}{1+\sqrt{2}K+K^2}$	$\frac{V_0-\sqrt{2V_0}K+V_0K^2}{1+\sqrt{2}K+K^2}$	$\frac{2(K^2-1)}{1+\sqrt{2}K+K^2}$	$\frac{1-\sqrt{2}K+K^2}{1+\sqrt{2}K+K^2}$
High frequency shelving cut	$\frac{1+\sqrt{2}K+K^2}{V_0+\sqrt{2V_0}K+K^2}$	$\frac{2(K^2-1)}{V_0+\sqrt{2V_0}K+K^2}$	$\frac{1-\sqrt{2}K+K^2}{V_0+\sqrt{2V_0}K+K^2}$	$\frac{2(V_0K^2-1)}{V_0+\sqrt{2V_0}K+K^2}$	$\frac{1-\sqrt{2V_0}K+V_0K^2}{V_0+\sqrt{2V_0}K+K^2}$
Peak boost	$\frac{1+\frac{V_0}{Q}K+K^2}{1+\frac{1}{Q}K+K^2}$	$\frac{2(K^2-1)}{1+\frac{1}{Q}K+K^2}$	$\frac{1-\frac{V_0}{Q}K+K^2}{1+\frac{1}{Q}K+K^2}$	$\frac{2(K^2-1)}{1+\frac{1}{Q}K+K^2}$	$\frac{1-\frac{V_0}{Q}K+K^2}{1+\frac{1}{Q}K+K^2}$
Peak cut	$\frac{1+\frac{1}{Q}K+K^2}{1+\frac{V_0}{Q}K+K^2}$	$\frac{2(K^2-1)}{1+\frac{V_0}{Q}K+K^2}$	$\frac{1-\frac{1}{Q}K+K^2}{1+\frac{V_0}{Q}K+K^2}$	$\frac{2(K^2-1)}{1+\frac{V_0}{Q}K+K^2}$	$\frac{1-\frac{V_0}{Q}K+K^2}{1+\frac{V_0}{Q}K+K^2}$

kde: V_0 = při zesílení = $10^{Gain/20}$, při zeslabení = $10^{-Gain/20}$

Gain = hodnota zesílení/zeslabení [dB]

K = $\tan(\pi F_C / F_S)$

F_C = centrální frekvence příslušného frekvenčního pásma [Hz]

F_S = vzorkovací frekvence vsupního signálu [Hz]

Q = tzv. Q factor - definuje šířku pásma (=F_B/F_C)

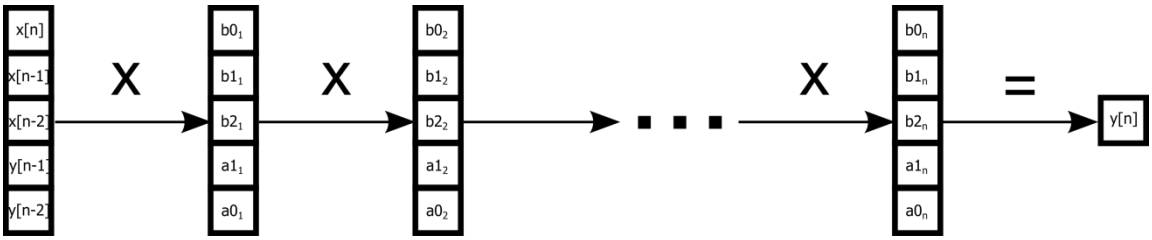
F_B = šířka pásma [Hz]

2.5.2 Aplikace ekvalizéru

Aplikace ekvalizéru probíhá podobně, jako kdyby se aplikovala série IIR filtrů. Vzorec (Rovnice 12) demonstruje ekvalizaci v jednom pásmu. Tyto pásma jsou za sebou v sérii, tedy výstupem jednoho pásma je vstup do dalšího pásma.

$$y[n] = x[n]b0_m + x[n-1]b1_m + x[n-2]b2_m + \\ -y[n-1]a1_m - y[n-2]a2_m \quad \text{(Rovnice 12)}$$

kde: $b0_m$, $b1_m$, $b2_m$, $a1_m$, $a2_m$ = koeficienty příslušného pásma

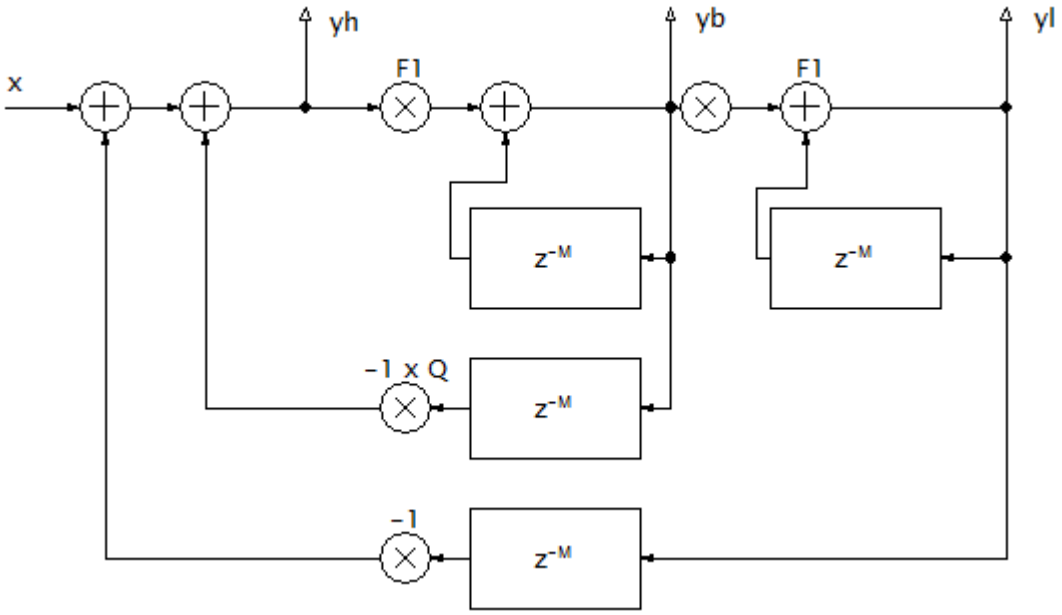


Obrázek 2.5. Demonstrace průchodu vstupního signálu jednotlivými pásmy ekvalizace

2.6 Efekty s časově proměnnou filtrací

Hlavní myšlenkou těchto efektů je filtrovat vstupní signál filtrem, jehož mezní kmitočet se bude v čase měnit. Změna kmitočtu má většinou průběh sinusového či trojúhelníkového charakteru. Existují struktury jak s jedním filtrem, tak i se sérií filtrů se společnou rychlostí oscilací.

Jedním z možných řešení je použít strukturu tzv. SV filtru (state variable filter), uvedenou v [3]. Pomocí dvou vzorců je možné spočítat koeficienty pro centrální kmitočet a šířku pásma (vznikne pásmová propust) a získat v každém okamžiku odděleně tři signály. První signál obsahuje frekvence pod nižší hranicí této pásmové propusti, druhý je právě pásmová propust a poslední signál se skládá pouze z frekvencí nad vyšším kmitočtem pásmové propusti.



Obrázek 2.6. Struktura State Variable Filter. Převzato z [3].

Koeficienty se spočítají ze vzorce:

$$F1 = 2\sin(\pi f_c / f_s)$$
$$Q = 2d \qquad \text{(Rovnice 13)}$$

Samotné signály potom získáme:

$$y_H[n] = x_{[n]} - y_L[n-1] - Q \times y_B[n-1]$$
$$y_B[n] = F1 \times y_H[n] + y_B[n-1] \qquad \text{(Rovnice 14)}$$
$$y_L[n] = F1 \times y_B[n] + y_{L[}[n-1]$$

2.6.1 Wah-Wah

Díky svému charakteristickému zvuku se nazývá také „kvákadlo“. Využíván je zejména kytaristy, kdy pomocí nožního pedálu pohybují pásmovou propustí a tím vznikne „kvákavý“ zvuk. Pokud není řízen pedálem, ale vnitřně vlastním řídícím signálem, nazýváme ho auto-wah. Použije-li se struktura SV filtru, pracuje se pouze se signálem y_B tedy s pásmovou propustí.

2.6.2 Phaser

Pracuje na podobném principu jako Wah-Wah s tím rozdílem, že není použita pásmová propust, ale pásmová zádrž. Při použití výše zmíněné struktury SV filtru toho lze docílit sečtením signálů y_L a y_H.

2.7 FIR filtry

Přestože se nejedná o audioefekt, filtrace patří mezi základní úlohy úpravy zvukových stop. FIR (finite impulse response) filtr, tedy filtr s konečnou impulzní odezvou, umožňuje odfiltrovat nežádané frekvence, například vysokofrekvenční šum, či frekvenci 50 Hz, která se mohla v zesilovači indukovat z transformátoru.

Existuje několik typů. Dolní propust (Low-pass) odfiltruje frekvence vyšší, než je mezní kmitočet. Filtr typu horní propust (High-pass) odfiltruje frekvence nižší než mezní kmitočet. Pásmová propust (Band-pass) zanechá frekvence mezi mezními kmitočty a pásmová zádrž (Band-stop) naopak odfiltruje frekvence mezi mezními kmitočty.

Při výpočtu koeficientů FIR filtru se vycházelo z [2]. Výhodou FIR filtrů oproti IIR (infinite impulse response) je, že jsou vždy stabilní a můžeme dosáhnout lineární fázové charakteristiky. Nevýhodou je větší HW náročnost k dosažení stejného útlumu (pro stejný útlum je potřeba většího řádu filtru, tedy většího počtu operací sčítání a násobení).

2.7.1 Výpočet koeficientů FIR filtru

Tato kapitola popisuje, jak postupovat při návrhu FIR filtru. Základní myšlenka návrhové metody spočívá v tom, že impulsní odezva je inverzním obrazem obdélníkové funkce (ideální filtr) ve frekvenční oblasti. Postupuje se následujícím způsobem:

1) Vymezení frekvenčního pásma (zvolí se mezní kmitočet/kmitočty, tím vznikne ideální filtr)
2) Frekvence se přepočítá na normalizovanou => $F_{Norm.} = F_{mezní} / F_{vzorkovací}$
3) Provede se inverzní diskrétní Fourierova transformace (přesun z frekvenční oblasti do časové oblasti) ideálního filtru a vybrání ze vzniklé funkce N + 1 bodů (N je řád filtru)

Tabulka 3. Inverzní Fourierova transformace obdélníkové funkce pro návrh FIR filtru [2]

Typ	koeficienty
Low-pass (dolní propust)	$h_d[n] = \frac{\sin[\omega_c(n-M)]}{\pi(n-M)}; n \neq M$ $h_d[n] = \frac{\omega_c}{\pi}; n = M$

High-pass (horní propust)	$h_d[n] = -\frac{\sin[\omega_c(n-M)]}{\pi(n-M)}; n \neq M$ $h_d[n] = 1 - \frac{\omega_c}{\pi}; n = M$
Band-pass (pásmová propust)	$h_d[n] = \frac{\sin[\omega_{c2}(n-M)]}{\pi(n-M)} - \frac{\sin[\omega_{c1}(n-M)]}{\pi(n-M)}; n \neq M$ $h_d[n] = \frac{\omega_{c2} - \omega_{c1}}{\pi}; n = M$
Band-stop (pásmová zádrž)	$h_d[n] = \frac{\sin[\omega_{c1}(n-M)]}{\pi(n-M)} - \frac{\sin[\omega_{c2}(n-M)]}{\pi(n-M)}; n \neq M$ $h_d[n] = 1 - \frac{\omega_{c2} - \omega_{c1}}{\pi}; n = M$

4) Tím vznikne impulzní odezva v časové oblasti, která je však nekonečně dlouhá. Je tedy nutné ji omezit přenásobením koeficientů filtru okénkovou funkcí

Tabulka 4. Výpočet okénkové funkce. Převzato z [2]

Okénko	Vzorec pro výpočet
Rectangular	$w[n] = 1; 0 \leq n \leq N-1$
Bartlett (Trojúhelníkové)	$\begin{cases} w[n] = \frac{2n}{N-1}\ ;\ 0 \leq n \leq \frac{N-1}{2} \\ w[n] = 2 - \frac{2n}{N-1}\ ;\ \frac{N+1}{2} \leq n \leq N-1 \end{cases}$
Hann	$w[n] = \frac{1}{2}\left(1 - \cos\left(\frac{2\pi n}{N-1}\right)\right);\ 0 \leq n \leq N-1$
Bartlett - Hann	$w[n] = 0{,}62 - 0{,}48\left\lvert\frac{n}{N-1} - 0{,}5\right\rvert +$ $0{,}38\cos\left(2\pi\left(\frac{n}{N-1} - 0{,}5\right)\right);\ 0 \leq n \leq N-1$
Hamming	$w[n] = 0{,}54 - 0{,}46\left(1 - \cos\left(\frac{2\pi n}{N-1}\right)\right); 0 \leq n \leq N-1$
Blackman	$w[n] = 0{,}42 - 0{,}5cos\left(\frac{2\pi n}{N-1}\right) + 8\cos\left(\frac{4\pi n}{N-1}\right);$ $0 \leq n \leq N-1$

$$h[n] = h_d[n] \times w[n] \qquad \text{(Rovnice 15)}$$

2.7.2 Filtrace

Po výpočtu koeficientů FIR filtru je již možné filtrovat signál. Je-li signál stereofonní, je nutné filtrovat každý kanál odděleně. Filtrace se provede aplikací vzorce:

$$y[n] = \sum_{m=0}^{N} h[m] \times x[n-m] \qquad \text{(Rovnice 16)}$$

kde: h = vektor koeficientů filtru

N = počet koeficientů filtru (řád filtru + 1)

2.8 Časová segmentace

2.8.1 Time-stretching

Pojmem time-stretching se nazývá nástroj, který dovoluje audio signál prodloužit či zkrátit v čase beze změn ve frekvenčním pásmu. Přehraje-li se audio signál rychleji, za běžných okolností bude mít zvuk vyšší kmitočet, naopak zpomalí-li se přehrávání, zvuk nahrávky bude hluboký. Toto se dá snadno pozorovat u audiokazety. Po aplikaci time-stretchingu by tento jev neměl nastat. Toho lze využít hned několika způsoby. Pokud se bude hudebník snažit odposlouchat rychlou část skladby (např. kytarové sólo), může si tuto část zpomalit, aniž by se změnily tóny. Dalším příkladem může být nahrávání zvuku k reklamnímu spotu, kdy řečník neodhadne předem danou délku své řeči a krátce svůj proslov protáhne. Díky nástroji time-stretching není nutné nahrávat znovu, stačí řeč patřičně softwarově zkrátit.

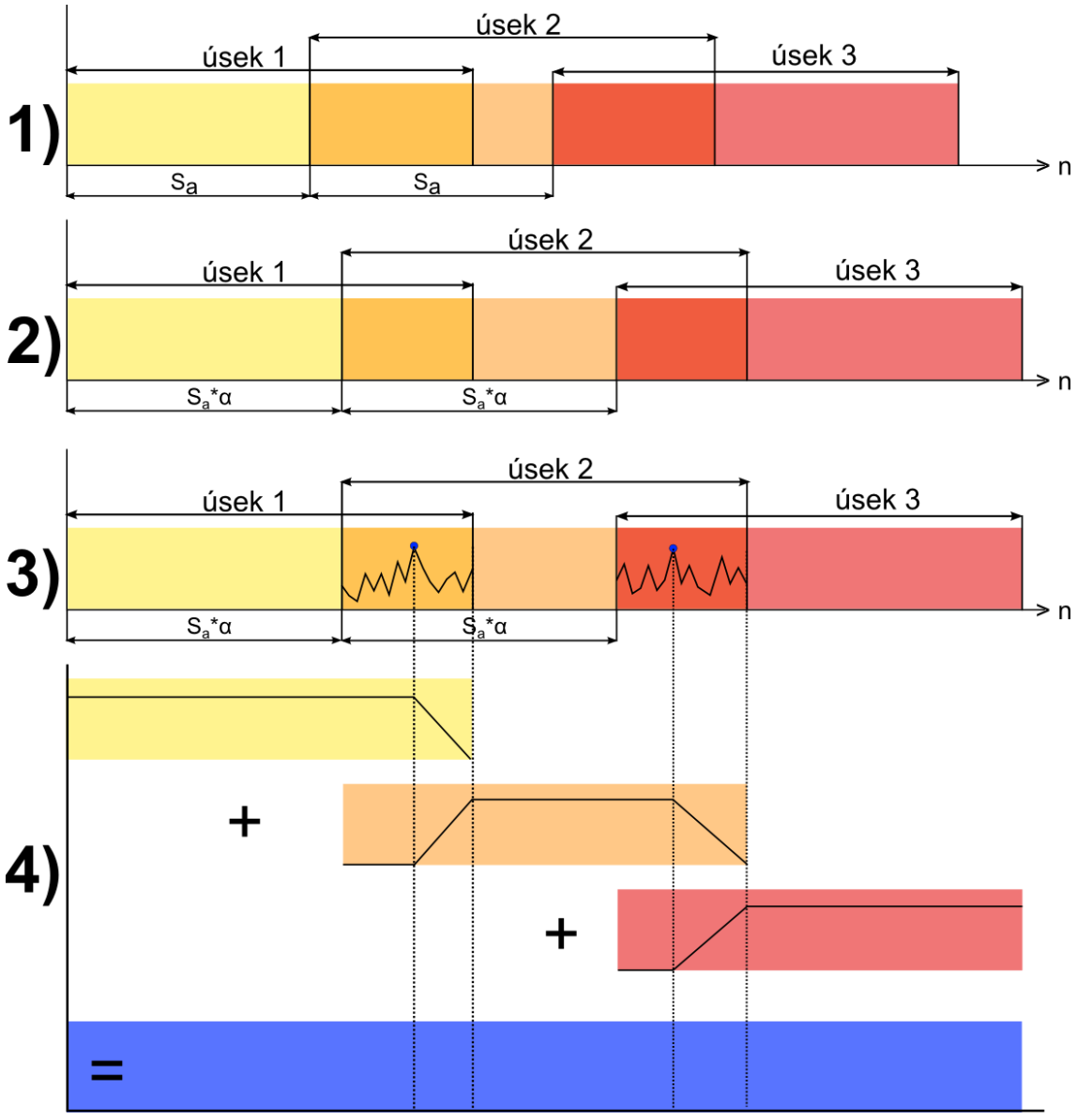
Time-stretching využívá algoritmus SOLA (Synchronous OverLap and Add). Zaprvé (viz Obrázek 2.7) se nahrávka rozloží na krátké úseky s pevnou délkou po přesně dané délce S_a. Je nutné, aby se tyto úseky překrývaly (tj. měly část vzorků společných), délka překryvu pak ovlivní maximální rozsah zvětšení/zmenšení.

Zadruhé se úseky posunou v čase o koeficient $S_a \times \alpha$. Tento koeficient ovlivňuje roztažení, či zkrácení souboru, tj. je-li $\alpha > 1$ soubor se roztahuje, je-li $\alpha < 1$ soubor se zkracuje. Pokud je $\alpha = 1$ soubor zůstává zachován. Posunem o $S_a \times \alpha$ již překrývané vzorky shodné nebudou a překrývaná část se zvětší či změnší.

Třetím krokem je v této nově vzniklé překrývané části najít, kde jsou si úseky nejvíce podobné. K tomu slouží korelace. Ta je definována vzorcem:

$$K_{ab} = \sum_{m}^{M} a[m] \times b[n+m] = \sum_{m}^{M} a[m-n] \times b[m] \quad \text{(Rovnice 17)}$$

Po nalezení požadovaného indexu sečteme oba úseky, a to tak, že z prvního úseku vezmeme část před maximální hodnotou korelace, a od této hodnoty lineárně zeslabujeme signál do konce překryvu. U druhého úseku potlačíme část před indexem s maximální korelací, poté lineárně zesilujeme do konce překryvu (ta se sčítá s částí která je v prvním úseku zeslabována) a ponecháme část za překryvem (viz Obrázek 2.7 – u čtvrtého kroku je značena na úsecích hlasitost). Z tohoto výsledku vytvoříme další úsek, který stejným způsobem přičteme k dalšímu úseku vstupního signálu.



Obrázek 2.7. Průběh algoritmu SOLA: 1) Segmantace 2) Posunutí v čase 3) Hledání maxima korelace 4) Sečtení úseků, vyznačena hlasitost

2.8.2 Pitch-shifting

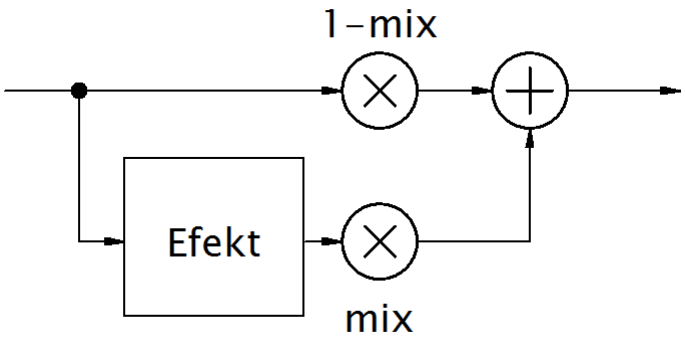
Pomocí pitch-shiftingu je naopak možné posunout frekvence, ale zachovat čas. Toho se dá využít například u tzv. AutoTune, kdy má toto zařízení informaci o frekvencích tónů a na tyto tóny dolaďuje vstupní signál. Pokud tedy zpěvák zpívá falešně, nebo je nástroj špatně naladěný, pomocí AutoTune zůstanou tóny ve stupnici. Další možností využití je posunutí vstupního signálu například o tón níž, čímž hudebník nemusí přelaďovat nástroj.

Jedním z principu fungování pitch-shiftingu je provést time-stretching a převzorkovat signál. Jak bylo řečeno výše, time-stretching provede změnu v čase, ale zachová frekvence. Převzorkováním signálu se obnoví původní délka signálu, ale již se provede změna ve frekvencích. Pokud soubor navzorkujeme poloviční vzorkovací frekvencí, ale budeme ho posléze přehrávat tak, jako by ke změně vzorkovací frekvence nedošlo, zvýšíme tím všechny frekvence obsažené v souboru na dvojnásobek.

Protože při operaci převzorkování budou obecně vycházet reálná čísla a nikoliv celá a pracuje se s diskrétním signálem, který má konečný počet vzorků v přesně definovaném čase, je nutné provést interpolaci, podobně jako je popsáno v kapitole 2.2**.**

2.9 Dry/Wet Struktura

V některých případech není vítané, aby audioefekt zcela ovlivnil vstupní signál, spíše je žádané, aby byl jenom něčím zvláštním, zajímavým na pozadí. K možnosti tohoto nastavení je možné nalézt na mixážních pultech či efektových krabičkách potenciometr dry/wet (dry = suchý – nezměněný, wet = mokrý – upravený) nebo mix. Pomocí parametru mix (který nabývá hodnot v intervalu nula až jedna) je možné nastavit, jakou měrou ovlivní audioefekt vstupní signál.



Obr. 2.1. Dry/Wet struktura

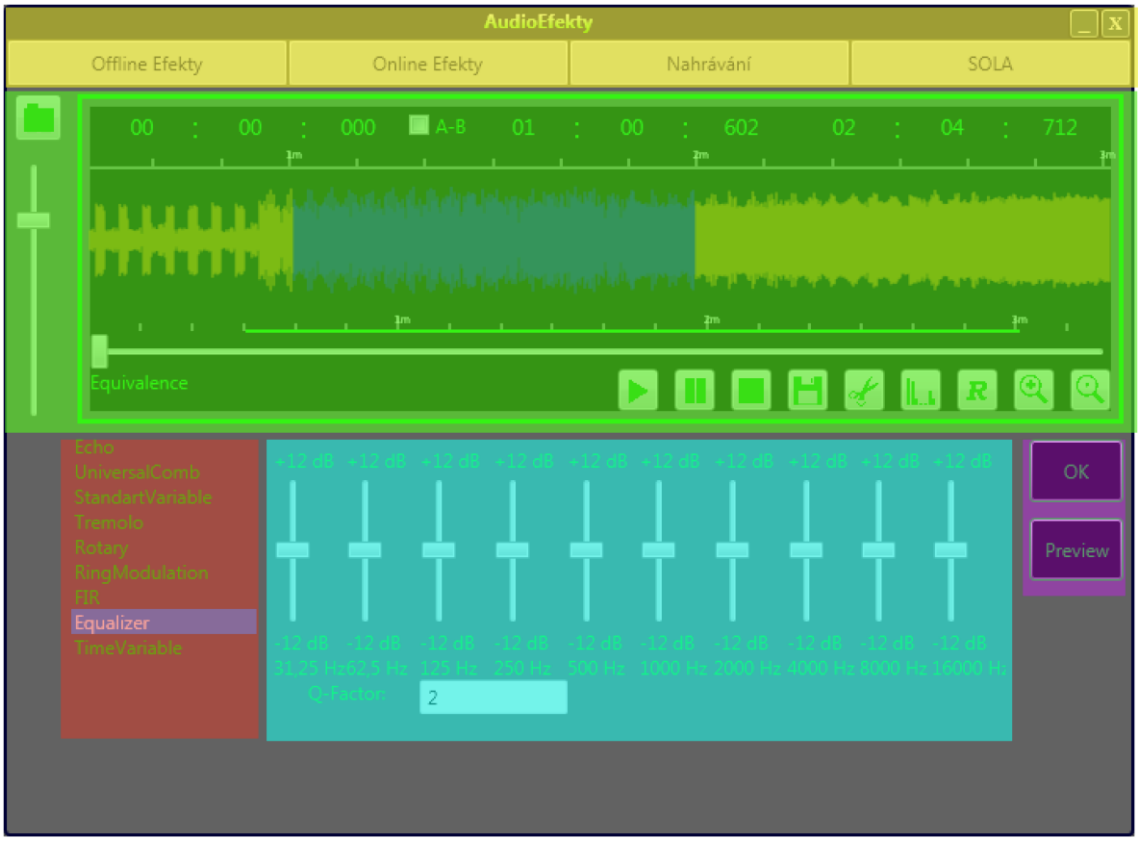
2.10 Ochrana proti přetečení

Ať už pracujeme s číslicovým či analogovým signálem, vždy je nutné hlídat, aby nepřesáhl jisté meze, což by mohlo v případě číslicového signálu způsobit přetečení hodnoty (a tím s největší pravděpodobností vznik hodnoty způsobující nelinearitu – praskání) a v případě analogového signálu poškození zařízení na výstupu.

Pracuje-li se s číslicovým signálem a s floating point logikou, signál by měl nabývat hodnot <−1;1> a tuto hranici by neměl překročit. To se dá zajistit například tímto způsobem:

$$x[n] = \begin{cases} x[n], & -0{,}9 \le x[n] \le 0{,}9 \\ sgn(x[n] \times \{0{,}9 + [0{,}1(1 - e^{-(0{,}1+|x[n]|)})]\}, & |x[n]| > 0{,}9 \end{cases} \quad \text{(Rovnice 18)}$$

3 Popis a obsluha aplikace



Obrázek 3.1. Okno offline režimu se zvýrazněnými hlavními částmi

Žlutě vyznačená část je hlavní panel aplikace, kterou mají všechny režimy společné. Vpravo nahoře je tlačítko pro zavření a minimalizování okna. Kliknutím na nápis AudioEfekty a táhnutím je možné přesouvat okno aplikace. Pod touto lištou jsou čtyři tlačítka pro výběr režimu. Pokud není vybrán žádný režim, je možné okno zvětšit či zmenšit táhnutím pravého spodního rohu.

V zelené části jsou ovládací prvky, horizontální posuvník času, vertikální posuvník hlasitosti a zobrazovač signálu. V levém horním rohu zobrazovače je údaj o aktuálním čase v audiosouboru, vedle povolení A-B smyčky (opakování od času A do času B) v aktuálním výběru. Další dva časové údaje zobrazují počáteční čas a konečný čas aktuálního výběru.

Součástí zobrazovače jsou dvě pravítka. Horní pravítko zobrazuje časové značky v aktuálním výběru, pravítko ve spodní části zeleně zvýrazňuje, která část je momentálně zobrazována (po přiblížení).

Samotný graf ukazuje energii signálu v daném čase. Je-li signál dvoukanálový (stereofonní) objeví se kanály pod sebou odděleně. Držením levého tlačítka, tažením

a puštěním tlačítka je možné vybrat úsek v audiosouboru. V tomto úseku je pak možné zapnout smyčku, nebo na tento úsek zazoomovat. Signál je zobrazen oranžovou barvou, vybraný úsek žlutou barvou. Při vybírání úseku se energie vybírané části zvýrazní modrou barvou. Výběr je možné také měnit vepsáním do jednoho z příslušných časových údajů. Celá tato zobrazovací struktura zobrazovače se objevuje v režimech Offline, Nahrávání a SOLA. Následující tabulka popisuje funkci jednotlivých ovládacích tlačítek.

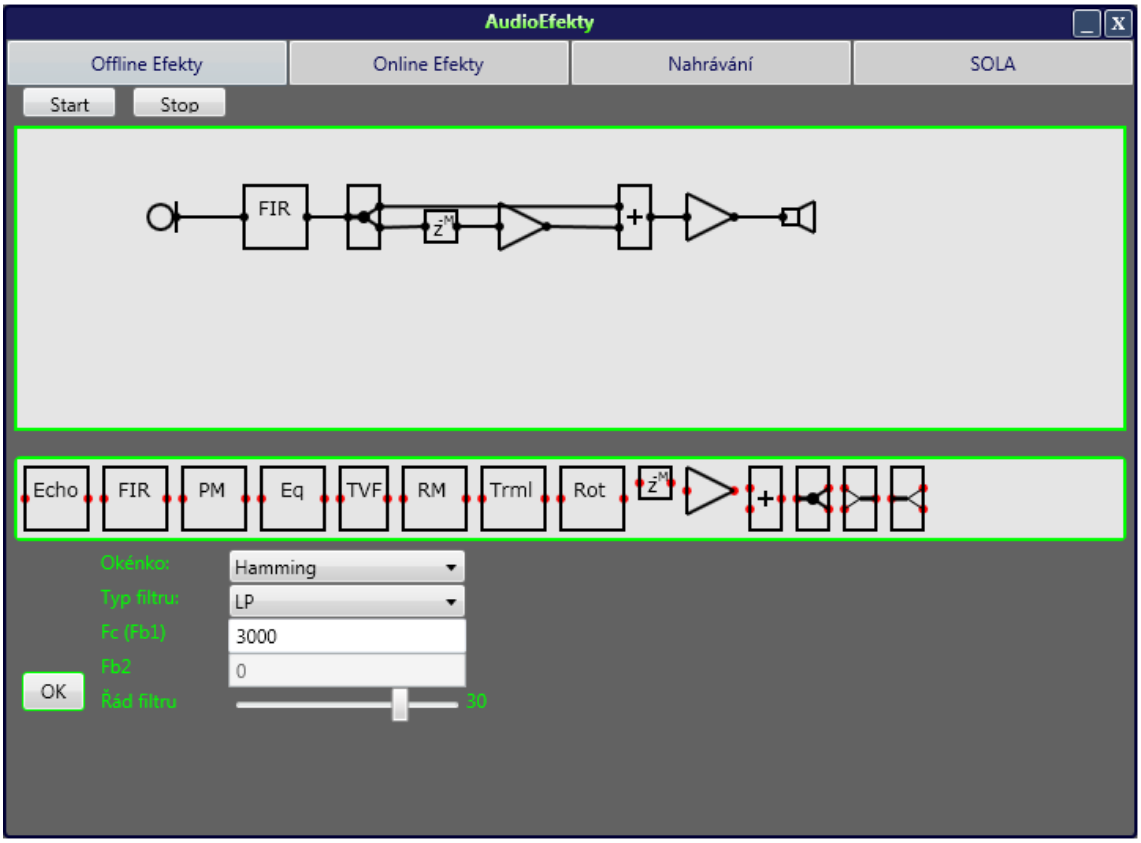
Tabulka 5. Význam ovládacích prvků v aplikaci

Symbol	Funkce
	Otevře audiosoubor s příponou .wav nebo .mp3
	Spustí přehrávání
	Pauza
	Zastaví přehrávání
	Uloží soubor na disk ve formátu *.wav
	Ořízne audiosoubor
	Zobrazí spektrogram signálu
	Reset změn (načte znovu původní soubor z disku)
	Zoom na aktuální výběr
	Reset zoomu

3.1 Offline režim

V tomto režimu se pracuje se audio soubory s příponou .wav nebo .mp3. Obrázek 3.1 zobrazuje okno offline režimu. V červeně vyznačené části je list implemetovaných audioefektů. Po kliknutí na efekt se otevře v modré části panel s parametry efektu. Ve fialové části jsou dvě tlačítka. Po stisknutí tlačítka OK se aplikuje audioefekt na vybranou část v souboru. Průběh lze vidět na horizontálním posuvníku zobrazovače. Stisknutím tlačítka Preview a spuštěním lze přehrát soubor s aktuálně nastaveným audioefektem. Pokud se změní parametry efektu, či celý efekt, je nutné preview režim vypnout a následně znovu zapnout.

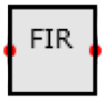
3.2 Online režim

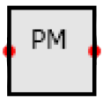
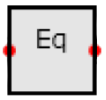
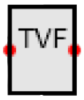
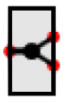
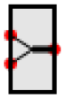


Obrázek 3.2. Okno Online režimu

Pod hlavním panelem je dvojice tlačítek spouštějící a vypínající smyčku. Níže pak panel, na kterém se schématicky zapojují bloky. Bloky se vkládají stylem Drag & Drop přetáhnutím z výběru. Stisknutím levého tlačítka myši ve schématu na konektor, tažením nad druhý konektor a puštěním se bloky propojí. Pravým tlačítkem myši se bloky a cesty odebírají. Levým stisknutím myši na blok se objeví v dolní části panel s parametry příslušného bloku, držením levého tlačítka myši je možné bloky přesouvat. Změny v bloku se aplikují po stisknutí tlačítka OK. Parametry je možné měnit i při spuštěné smyčce.

Tabulka 6. Význam bloků v online režimu

Symbol	Funkce
	Efekt ozvěny
	FIR filtry

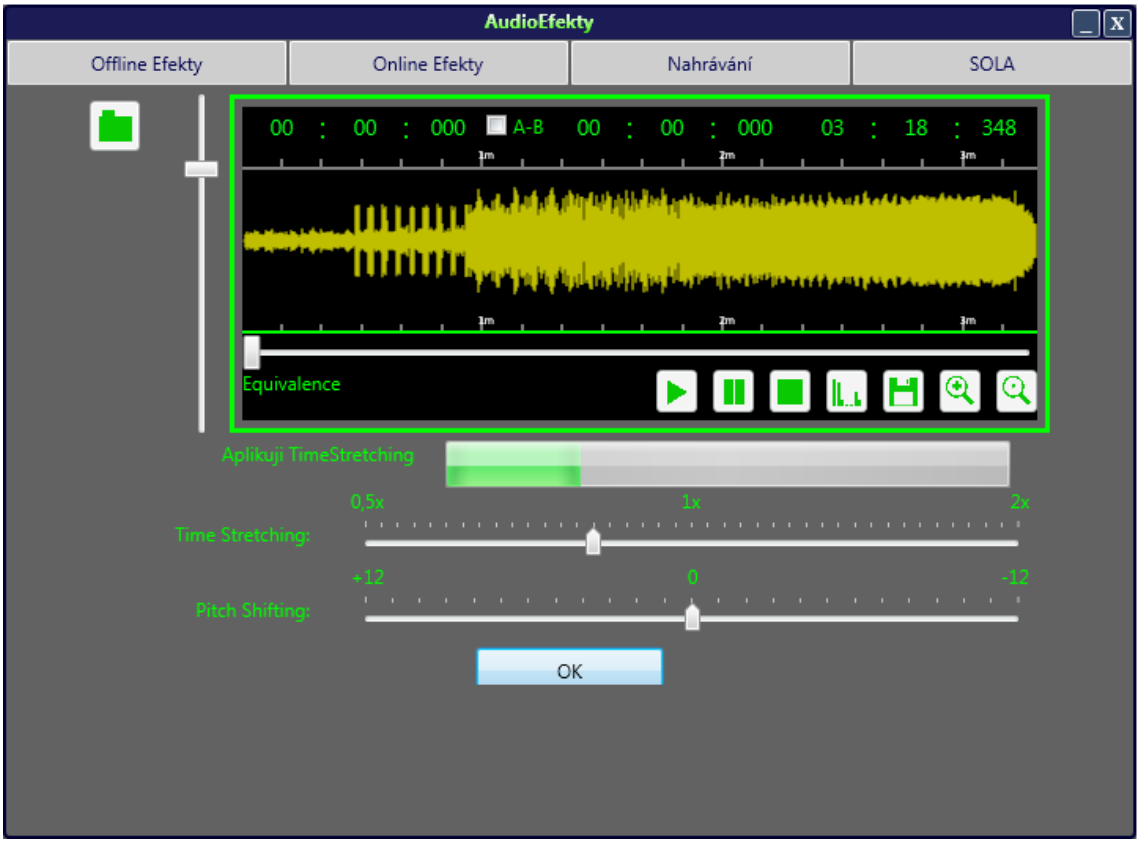
	Efekty s proměnnou fází
	Ekvalizér
	Efekty s časově proměnnou filtrací
	Ring modulator
	Tremolo
	Rotary
	Zpožďovač
	Zesilovač
	Sumátor
	Uzel (rozděli signál na dva)
	Rozdělí stereo signál na levý a pravý
	Spojí levý a pravý kanál do jednoho stereo signálu

3.3 Režim nahrávání

Obrázek 3.3. Okno nahrávacího režimu

V levé horní části je seznam dostupných vstupních zařízení, parametry nahrávání a tlačítko pro spuštění. V pravé části je formulář pro vygenerování sinusového signálu. Dole je zobrazovač signálu s ovládacími prvky.

3.4 Režim SOLA



Obrázek 3.4. Okno režimu SOLA

Režim SOLA je režimem aplikace, ve kterém se provádí time-stretching a pitch-shifting, tedy úpravy v časovém či frekvenčním pásmu, založené na principu časové segmentace, přesněji algoritmu SOLA. Po otevření souboru se podobně jako v Offline režimu zobrazí energie signálu na zobrazovači. V dolní části jsou dva posuvníky. Horní posuvník řídí time-stretching a udává délku souboru po aplikaci. Pod ním je posuvník pitch-shiftingu udávající o kolik tónů se signál frekvenčně posune. Obě úpravy značí změnu oproti originálnímu souboru, tedy opakovaná aplikace se stejnými parametry povede ke stejnému výsledku. Po stisknutí tlačítka OK se objeví signalizace průběhu.

4 Praktická část

Aplikace je naprogramována v jazyce C#, pomocí rozhraní WPF, v programu Microsoft Visual Studio 2010. WPF je „mladší bratr“ technologie WinForms, který přináší mimo jiné spoustu nových možností:

- značkovací jazyk XAML (vycházející z XML) pro návrh layoutu.
- data binding - provázání určitých dat do společné paměti
- styly
- layout - relativní velikosti počítané v run-time
- vektorová grafika
- vykreslování grafiky pomocí Direct 3D

K práci se zvukem byla využívána open source knihovna NAudio verze 1.6. Ta byla použita pro práci se zvukovou kartou (vstup a výstup), čtení a zápis audio souborů z disku a algoritmus FFT pro zobrazení spektrogramů. Tato knihovna také umožňuje práci s ASIO, WASAPI, MIDI, streamované audio, atd.

4.1 Popis vytvořených tříd

V následující kapitole je stručně popsán účel nejdůležitějších tříd. Z důvodu interakce mezi třídami má tento popis logickou návaznost.

4.1.1 Třída AudioFile

Tato třída umožňuje práci se soubory s příponou .wav nebo .mp3. Hlavní metodou je *OpenFile(string s)*, která nejdřívě ověří, zda je přípona platná a poté se soubor uloží s příponou .wav na pevný disk do adresáře, kde se nachází spouštěcí soubor aplikace. Je-li otvírán soubor *.wav, je pouze překopírován, pokud má soubor příponu .mp3, je převeden na soubor *.wav a uložen. Tento přístup byl zvolen pro svou menší náročnost na operační paměť počítače. Třída AudioFile obsahuje několik metod, které umožňují z tohoto souboru číst. Při čtení (vyjma jedné metody) převádí vzorky na datový typ floating point.

4.1.2 Abstraktní třída AudioEffect

Z této třídy jsou odvozeny všechny implementované audioefekty (viz Příloha B). Obsahuje několik abstraktních metod, které zajistí, že tyto metody bude mít i potomek.

- *SetWaveFormat(WaveFormat wave)* - nastaví parametry vstupního signálu (vzorkovací frekvence, počet kanálů)

- *Initialize()* - inicializace stavu efektu, tzn. přepočítání koeficientů apod.
- *SetParameters(object Params)* - nastaví parametry samotného efektu
- *GetForm()* - každý efekt má svůj vlastní formulář, ve kterém se nastavují jeho parametry. Tato metoda vrací tento formulář.
- *Read(ref float[] buffer, int offset, int count)* - použití samotného efektu. Vstupní buffer se posílá jako reference a po výstupu z metody se stává výstupním bufferem. Jelikož velikost bufferu je malá a za krátký čas je těchto bufferů zapotřebí hned několik, tímto způsobem se obchází volání garbage collectoru, který by neustále alokovoval novou paměť a znovu ji uvolňoval. Takto nám stačí neustále používat dokola několik paměťových polí (bufferů) dokola.

Pro použití efektu je nutné nejdříve oznámit parametry signálu, nastavit parametry samotného efektu, poté efekt inicializovat a pak už je možné zasílat referenci na buffer, na který bude efekt použit.

4.1.3 Třída Player

Účelem třídy Player je číst vzorky z třídy AudioFile (tedy ze souboru) a posílat je na výstup zvukové karty. Kromě standardních funkcí přehravače jako přehrávat, pauza, stop, posun v souboru, zajišťuje ovládání hlasitosti, zacyklení ve smyčce A-B a přehrávání s efektem (možnost preview v offline režimu).

4.1.4 Třída PlayerControl

Tato třída slouží jako prostředník mezi uživatelským rozhraním a přehrávačem. Propojuje vizuální prvky přehravače a reaguje na příkazy od uživatele. Mezi propojené prvky patří přehravač, časový posuvník, posuvník hlasitosti a zobrazovač signálu. Současně zajišťuje většinu operací, které se provádí v offline režimu a to počínaje zkracováním na vybraný úsek v audiosouboru, přes aplikaci efektu až po zobrazení spectrogramu.

4.1.5 Abstraktní třídy OnlineObject a OnlineEffect

Třída OnlineObject představuje blok, který se v online režimu vykresluje na pracovní ploše. Uchovává informaci o poloze bloku, jeho velikosti a stará se o vykreslení konektorů. Z této třídy je odvozen reproduktor (je koncovým blokem v řetězci, tedy nečtou se z něj vzorky) a další abstraktní třída OnlineEffect. Ve třídě

OnlineEffect je navíc implementováno rozhraní IAESampleProvider a metoda *SetParameters(object par)*. Z třídy OnlineEffect jsou následně odvozeny všechny online efekty a další online bloky (viz Příloha A)

4.1.6 Třídy FloatToWave a WaveToFloat

Tyto dvě třídy slouží pro konverzi z datového typu floating point do WAV formátu a naopak. Ve WAV formátu jsou vzorky uložené ve formátu byte po 2×8 bitech (rozlišení A/D převodu), střídavě vzorek levého a pravého kanálu. V 16 bitové hloubce můžou nabývat hodnot datového typu signed short (−32 768 až 32 767). Pro převod na floating point hodnotu se nejdříve převedou dvě osmibitové hodnoty na jeden vzorek datového typu short (16-bit), vydělí se maximální hodnotou shortu a přetypuje se na datový typ float. Pro opačný převod se naopak reálné číslo vynásobí maximální hodnotou datového typu short, přetypuje se na short a rozloží na dvě hodnoty datového typu byte. Třída FloatToWave navíc obsahuje ochranu proti přetečení.

Do těchto tříd nevstupují jednotlivé vzorky, ale pole vzorků. Pole typu float má vždy poloviční velikost, než pole byte. Kvůli garbage collectoru je ale výhodné s každým převodem nevytvářet nové pole, ale vytvořit při prvním zavolání omezený počet polí (dvě, případně 3) a pouze hlídat, zda je velikost výstupního pole poloviční (dvojnásobná pro převod float to byte).

4.1.7 Další třídy

Úplný přehled tříd, struktur a rozhraní (vyjma tříd OnlineObject a AudioEffect a jejich potomků, které jsou přiloženy jako Příloha A a Příloha B) je graficky vyjádřen v příloze (viz Příloha C), kde je rozdělen do skupin objektů, které spolu souvisí. Skupin je celkem devět.

1) Struktury obsahující parametry jednotlivých efektů
2) Formy jednotlivých efektů s rozhraním IReturningDataForm, které zajišťuje, že formulář dokáže vrátit nastavené parametry efektu
3) Třídy využívané online režimem
4) Pomocné třídy pro efekty a další DSP operace
5) Třídy pro převody mezi datovým typem byte a datovým typem float
6) Třídy využívané režimem SOLA
7) Skupina tříd, zajišťující zobrazení hlasitosti signálu
8) Formy režimů, hlavní okno a okno spektrogramu

9) Třídy pro offline režim

Nejdůležitější z těchto tříd byly podrobněji popsány v předchozích podkapitolách. Zde je stručný význam dalších tříd.

- OfflineEffect - obsahuje třídu AudioEffect, zajišťuje zapojení do offline řetězce, díky rozhraní ISampleProvider
- EffectFileAplicator - čte po bufferech data ze souboru a ukládá je do nového souboru na disku. V zadaném čase A začne aplikovat libovolný efekt, v čase B s aplikováním přestane. Poté ukládá opět čistá data ze vstupního souboru.
- SignalView - zobrazuje hlasitost signálu v daném časovém úseku pro mono signál
- SignalPanel - „zobrazovač“. Je-li načten nový zvukový soubor, zjistí celkový čas a vykreslí pravítka. Zjistí také, zda je signál mono, či stereo. Je-li stereo, vytvoří dvě instance třídy SignalView, je-li mono, vytvoří pouze jednu.
- EnergySolver - je obsažena ve třídě SignalPanel. Počítá energie pro třídy SignalView.
- SOLA - třída umožňující time-stretching
- PitchShifter - třída umožňující pitch-shifting
- Prevod - soubor statických metod pro převody mezi datovými typy (float[] na byte[] a naopak, pro spectrogramy pole byte[] na pole complex[] a naopak). Dále převádí z informace o indexu vzorku, vzorkovací frekvenci a počtu kanálů na čas a naopak)
- MouseSolver - používá se v online režimu k zjištění, na který objekt uživatel kliknul myší
- QueueField<T> - pole s chováním podobnému frontě, používané jako paměť z předchozích bufferů. Jelikož je nutné, například u efektů s proměnnou fází, získávat vzorky z libovolného místa ve frontě, fronta implementovaná v prostředí .NET nevyhovuje požadavkům aplikace.

4.2 Vstupně výstupní řetězce

Aplikace používá v různých režimech několik možností vstupů a výstupů. V této kapitole je popsáno, jak průchod tímto řetězcem tříd probíhá.

Vstupem do řetězce může být:

- Soubor na disku
- Data ze zvukové karty
- Generování vstupu v běhu programu

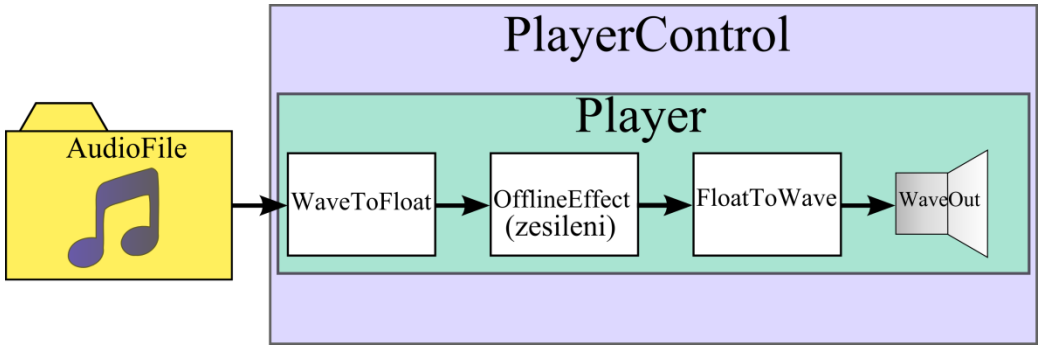
Výstupem pak může být:

- Soubor na disku
- Výstup na zvukovou kartu

4.2.1 Přehrávání (Offline režim, Nahrávání, SOLA režim)

Vstup: soubor na disku

Výstup: výstup na zvukovou kartu

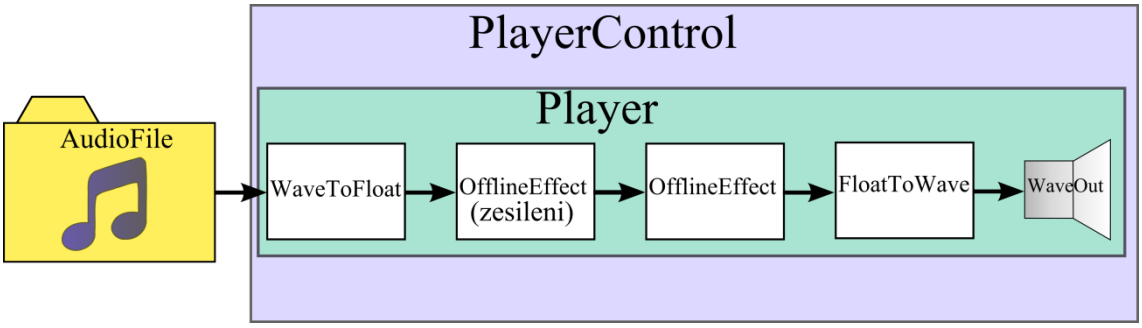


Obrázek 4.1. Řetězec přehrávání

Třída WaveOut vytvoří buffer byte[] a zažádá třídu FloatToWave o vzorky. Ten v prvním zažádání vytvoří buffer float[], v dalších již používá ten samý (má-li odpovídající velikost, viz kapitola 4.1.6), a referenci na tento buffer předá třídě OfflineEffect. Ta zatím žádné vzorky také nemá, předá buffer třídě WaveToFloat. Tato třída, podobně jako třída FloatToWave, buď již má vytvořený buffer byte[], nebo si ho jednou vytvoří a zažádá třídu AudioFile, která vzorky přečte ze souboru. Tím třída WaveToFloat získala buffer byte[], který začne vzorek vzorku překládat na float a ukládat do bufferu float[], na který třídě WaveToFloat přišla reference od třídy OfflineEffect. V tomto řetězci je třída OfflineEffect pouze zesilovač, vzorky tedy vynásobí koeficientem zesílení. Nyní může třída FloatToWave zesílené vzorky přeložit do buffer byte[], který již třída WaveOut posílá na výstup zvukové karty.

4.2.2 Přehrávání s efektem - preview (Offline režim)

Řetězec je identický s předchozí strukturou, pouze je mezi zesílení a převod FloatToWave zařazena aplikace efektu.

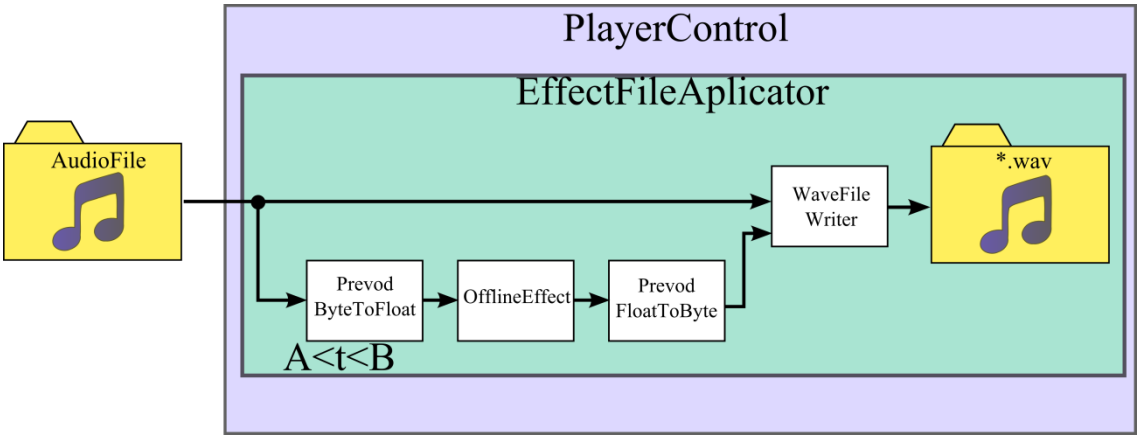


Obrázek 4.2. Řetězec preview

4.2.3 Řetězec aplikace audioefektu na soubor (Offline režim)

Vstup: soubor na disku

Výstup: soubor na disku



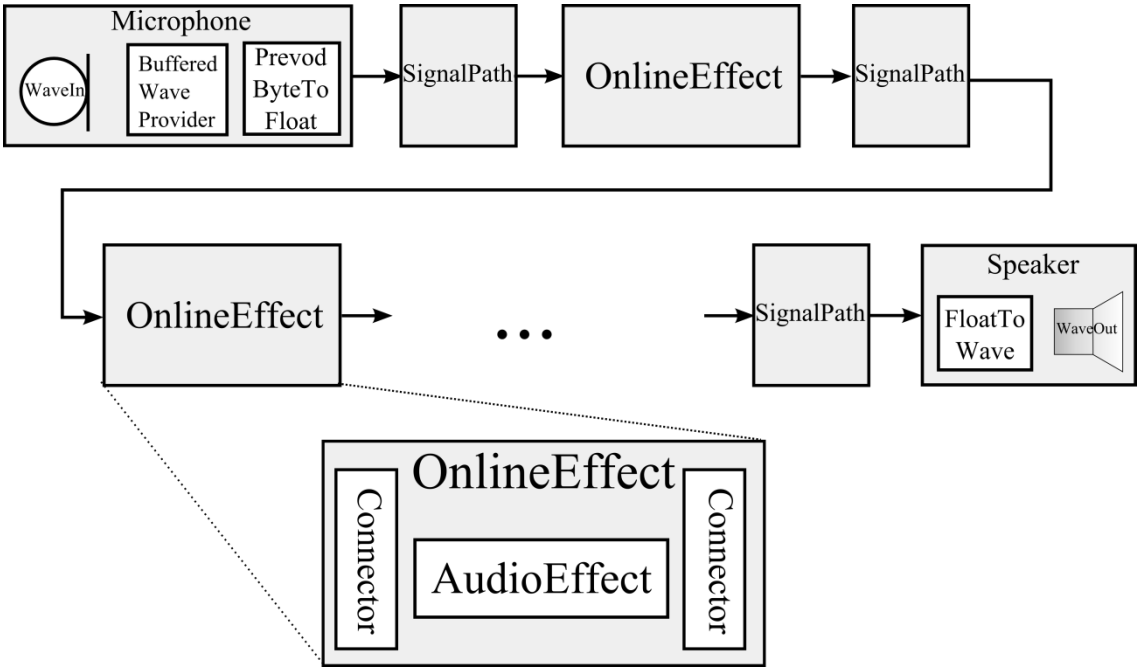
Obrázek 4.3. Použití efektu na soubor na disku

Třída EffectFileAplicator umožňuje použít efekt jen v úseku mezi časy A a B. Je-li načtený buffer mimo tento interval, přečtená data (datového typu byte[]) ze třídy AudioFile jsou přímo zapisovány pomocí třídy WaveFileWriter (z knihovny NAudio) do souboru. V intervalu mezi A a B je však nutné převést byte[] buffer na float[] aplikovat libovolný efekt a tento buffer převést zpět na byte[] a zapsat.

4.2.4 Online řetězec

Vstup: data ze zvukové karty

Výstup: výstup na zvukovou kartu



Obrázek 4.4. Online řetězec

Celý online řetězec je postaven na rozhraní IAESampleProvider, který je implementován v abstraktní třídě OnlineEffect, třídě Connector a třídě SignalPath. Toto rozhraní zajišťuje tři druhy komunikace mezi bloky. Propojením dvou bloků v návrhovém panelu online režimu se vytvoří nová instance třídy SignalPath, která nese informaci, na které konektory je připojena. Každý konektor má informaci k jakému bloku náleží a případně jaká instance třídy SignalPath je k němu připojena.

První komunikace zajistí, že těsně předtím, než se spustí smyčka, proběhne celým řetězcem signál *Initialize()*. Ten jde směrem od vstupu ke konci řetězce a zajistí, že si všechny bloky načtou správnou vstupní vzorkovací frekvenci a počet kanálů. Díky tomu si mohou přepočítat koeficienty (FIR, Wah-Wah, ..) nebo alokovat dostatek paměti (Echo). Druhá komunikace proběhne hned po první, tentokrát od výstupu směrem ke vstupu. Díky metodě *Control()* se ověří, zda je řetězec uzavřený, tedy zda existuje cesta od výstupu až ke vstupu. Proběhne-li tato kontrola v pořádku, smyčka se spustí.

Samotná smyčka probíhá podobně, jako u 4.2.1. Výstup (tedy blok WaveOut ve třídě Speaker) si zažádá o buffer typu byte[] třídy FloatToWave. Ten vytvoří buffer typu float[], který projde dále řetězcem skrze cesty, konektory a samotné bloky s efekty, až se dostane k třídě Microphone. Tato třída obsahuje třídu WaveIn, která čte vzorky ze vstupu zvukové karty. Vzorky se ukládají do mezipaměti (třída

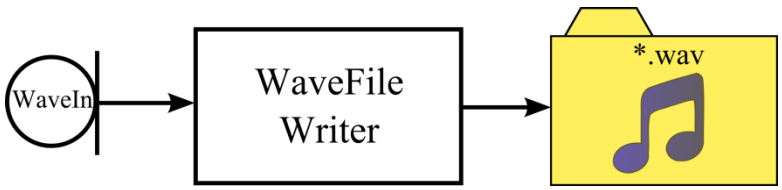
BufferedWaveProvider). Dostane-li třída Microphone žádost o buffer, požadovaný počet vzorků přečte z mezipaměti, vzorky převede na datový typ float a naplněný buffer postupuje zpět stejnou cestou, skrze konektory, cesty a bloky (ve kterých se na buffer aplikuje efekt), až se dostane zpět ke třídě FloatToWave, kde proběhne kontrola proti přetečení, převede se do bufferu typu byte[], na který již čeká třída WaveOut. Je patrné, že tento proces trvá jistou dobu.

Latence signálu (zpoždění výstupu oproti signálu na vstupu) je přibližně 100 ms. Tato hodnota je daná velikostí bufferů a latencí závislou na hardwarových prostředcích. Hodnota latence je vždy konstantní, zatížením procesoru se nezmění, v případě extrémního zatížení by došlo k vytvoření „hluchých“ míst v každém bufferu, což v důsledku znamená praskání a přerušování výstupního signálu. Latence byla v aplikaci empiricky nastavena s malou rezervou na minimální hodnotu pro použité zvukové rozhraní.

4.2.5 Nahrávací řetězec

Vstup: data ze zvukové karty nebo generovaný signál

Výstup: soubor na disku



Obrázek 4.5. Řetězec nahrávání

Po naplnění vstupního bufferu vyvolá třída WaveIn událost. V ní se právě naplněný buffer zapíše díky třídě WaveFileWriter do souboru. Aplikace má také možnost generování sinusového signálu, kdy se naplní buffer vzorky, převede na datový typ byte[] a opět třída WaveFileWriter zajistí zápis do souboru.

4.3 Realizace efektů

V následující kapitole je uvedeno, jak jsou řešeny programově jednotlivé efekty.

4.3.1 Echo

V inicializaci se přepočítá časová hodnota zpoždění na počet vzorků (Rovnice 1). Poté se vytvoří fronta s odpovídající velikostí. Po spuštění efektu vstupní signál prochází na výstup a zároveň je ukládán do fronty. Po naplnění fronty se ke

každému dalšímu vstupnímu vzorku přičte poslední vzorek z fronty, vynásobený koeficientem hlasitosti ozvěny.

4.3.2 Universal Comb struktura

Nejdříve je potřeba vytvořit instanci třídy QueueField<float>, kde se v konstruktoru předá velikost pole, které třída alokuje. Velikost pole (tedy zpoždění) se spočítá ze vzorce (viz Rovnice 1). Po spuštění, dokud není naplněno pole, se pouze řadí vzorky do fronty a na výstup se dostanou vynásobeny koeficientem BL (struktura viz kapitola 2.1). Po naplnění fronty se na konec fronty přidává vstupní vzorek, ke kterému se přičte vzorek z fronty na pozici zadaného zpoždění vynásobený koeficientem FB. Výstupním vzorkem je potom součet tohoto vzorku, vynásobeného koeficientem BL, a vzorku z fronty na pozici zpoždění, vynásobeného koeficientem FF.

4.3.3 Efekty s proměnným zpožděním

Výstupní vzorek se počítá stejným způsobem jako u 4.3.2, pouze s tím rozdílem, že hodnota vzorku z fronty, která je násobena koeficientem FF, má hodnotu zpoždění proměnnou. Vzorek s hodnotou zpoždění je nutné interpolovat podle vzorce (Rovnice 6) ze dvou vedlejších vzorků.

Je-li hodnota zpoždění sinusového charakteru (vibrato, flanger), je počítána ze vzorce (Rovnice 5). Pokud je zpoždění náhodného charakteru (chorus, doubeling), vytvoří se v inicializaci pole o tisíci členech s náhodnými hodnotami v intervalu <0;hloubka>. Při počítání náhodného zpoždění se cyklicky prochází toto pole a filtruje se IIR filtrem typu dolní propust, navrženým v programu MATLAB s mezní frekvencí 10 Hz při vzorkovací frekvenci 44100 Hz.

4.3.4 Ring modulator

Zde není potřeba nic inicializovat. Výstupní vzorek je dán podle vzorce (Rovnice 8). U stereo signálu je nutné počítat každý kanál zvlášť.

4.3.5 Tremolo

U efektu tremolo není potřeba žádná inicializace. Podobně jako u efektu ring modulator se výstup, který je dán vzorcem (Rovnice 7), u stereo signálu musí řešit pro každý kanál zvlášť (jinak by byla frekvence kmitání dvojnásobná).

4.3.6 Rotary

Efekt rotary byl implementován pouze pro stereo signál, v inicializaci je nutné tento požadavek zkontrolovat. Poté se výstupní vzorky počítají ze vzorce (Rovnice 9). Tento efekt byl implementován s dry/wet strukturou popsanou v kapitole 2.9.

4.3.7 Ekvalizér

Nejdříve proběhne kontrola, zda se uživatel nesnaží zesilovat/zeslabovat frekvence, které z Nyquistova teorému v signálu nemůžou existovat. Je-li vzorkovací frekvence vstupního souboru 16 kHz, můžou se v tomto souboru objevit frekvence maximálně do 8 kHz. Pásmo 16 kHz se tedy neprojeví. Následně se vytvoří pole, kam se ukládají vzorky z předchozího bufferu. Nakonec se spočítají koeficienty. Samotná filtrace probíhá ve for cyklu, kdy se postupně aplikuje vzorec (Rovnice 12) pro každé pásmo, které má nenulovou hodnotu zesílení.

4.3.8 Efekty s proměnnou filtrací

V inicializaci proběhne kontrola na porušení Nyquistova teorému. Poté se navrhne SV filtr a jako počáteční mezní frekvence se zvolí střed oscilací. Po spuštění. se přepočítá pro každý vzorek mezní frekvence filtru podle vzorce:

$$f = F_C + A \sin\left(2\pi F_K \frac{i}{F_S}\right) \qquad \text{(Rovnice 19)}$$

kde:
- f = výsledná mezní frekvence
- F_C = centrální frekvence [Hz]
- A = hloubka [Hz]
- F_K = frekvence kmitání [Hz]
- F_S = vzorkovací frekvence
- i = časový údaj v diskrétní oblasti

Provede se filtrace pro spočítaný mezní kmitočet. Poté se postup liší pro jednotlivé efekty. V případě efektu wah-wah je výstupem vzorek z pásmové propusti SV filtru. U efektu phaser je výsledkem součet vzorku dolní propusti a vzorku horní propusti. Tento efekt je implementován pomocí dry/wet struktury, výstup je tedy v poměru mezi vstupním signálem a upravenám signálem v závislosti na parametru mix.

4.3.9 FIR filtry

Během inicializace se zkontroluje, zda nebyl porušen Nyquistův teorém, tedy nesnaží-li se uživatel filtrovat frekvence, které v signálu nemohou existovat. V případě porušení teorému je uživatel upozorněn. Následně se spočítají koeficienty filtru (podle návodu v kapitole 2.7.1). Aplikace filtru je dána vzorcem (Rovnice 16). U filtrování je nutné si ukládat poslední vzorky z předchozího bufferu (počet vzorků je dán řádem a počtem kanálů signálu) a použít je u počátku nového bufferu.

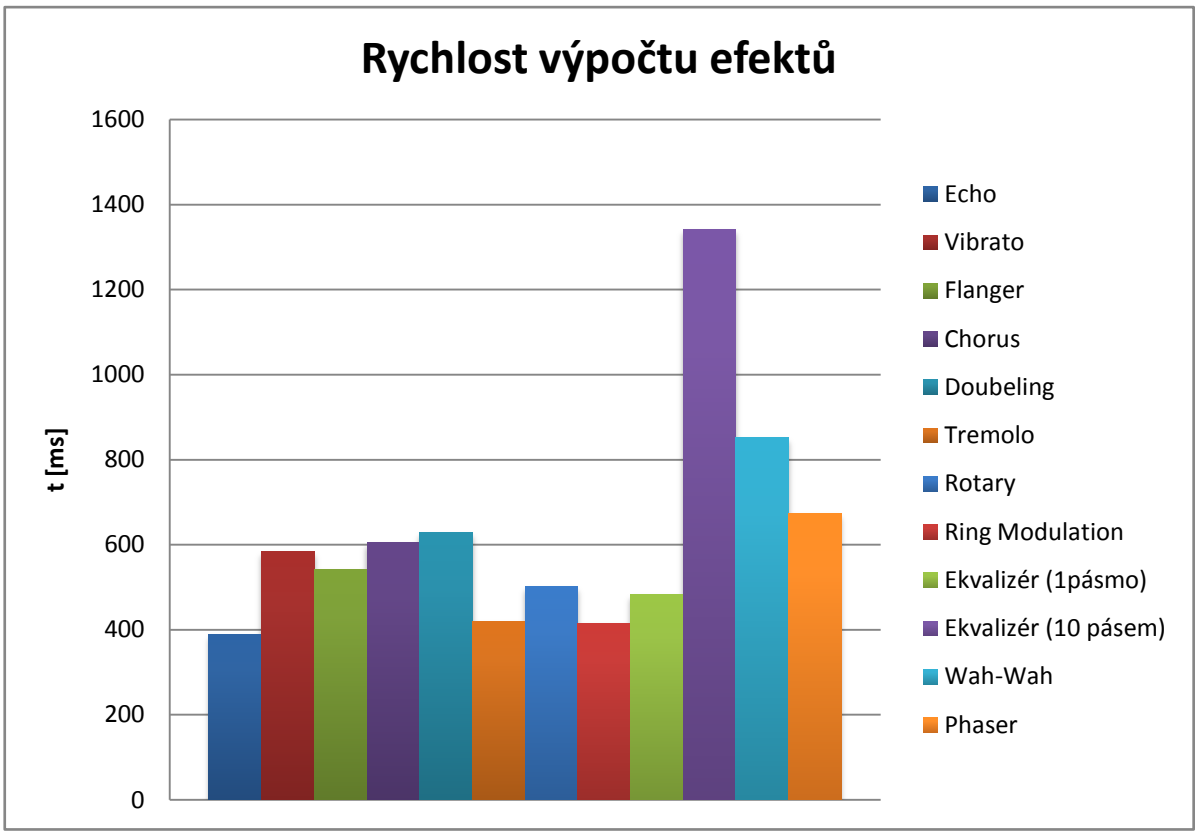
5 Rychlost výpočtu

Následující tabulka zobrazuje rychlost výpočtu jednotlivých efektů na stereofonním audiosouboru o délce 10 sekund a 123 milisekund. Měření bylo provedeno diagnostickým nástrojem StopWatch z prostředí .NET, výpočet proběhl na počítači s dvoujádrovým procesorem Intel® Core™ i3 330M s taktovací frekvencí 2,53 GHz a 6 GB RAM s frekvencí 1066 MHz. Výsledky jsou orientační, měření bylo částečně ovlivněno momentálním zátížením počítače, přestože v aktuální dobu neběžel žádný náročný proces na pozadí.

Sloupek *Poměr* v Tabulce 7 značí poměr mezi délkou výpočtu ku délce souboru. Díky tomu je patrné, že nejnáročnější efekt (ekvalizér při změně ve všech deseti pásmech) zabere přibližně 13% reálného času, tedy pokud by nedocházelo k zatížení testovací sestavy, bylo by teoreticky možné v reálném čase počítat smyčku, která by měla těchto deseti-pásmových ekvalizérů za sebou až sedm.

Tabulka 7. Rychlost výpočtu efektů na stereo soubrou o délce přibližně 10,123 sekundy.

Efekt	1. měření [ms]	2. měření [ms]	3. měření [ms]	4. měření [ms]	5. měření [ms]	Průměr [ms]	Poměr
Echo	446	312	384	435	367	388,8	0,0384
Vibrato	697	571	514	602	533	583,4	0,0576
Flanger	515	531	542	581	543	542,4	0,0536
Chorus	608	596	612	604	607	605,4	0,0598
Doubeling	713	609	561	674	584	628,2	0,0621
Tremolo	381	381	558	360	421	420,2	0,0415
Rotary	618	485	526	431	450	502	0,0497
Ring Modulation	344	322	463	360	590	415,8	0,0410
Ekvalizér (1pásmo)	413	634	431	492	443	482,6	0,0477
Ekvalizér (10 pásem)	1246	1317	1252	1224	1666	1341	0,1325
Wah-Wah	747	614	809	1099	988	851,4	0,0841
Phaser	720	548	939	581	582	674	0,0666



Graf 1. Porovnání HW náročnosti implementovaných audioefektů

6 Závěr

Vytvořená .NET aplikace v jazyce C# obsahuje deset audio-efektů, ekvalizér, filtrování FIR filtry, time-stretching a pitch shifting. Kromě posledně dvou zmíněných je možné použít efekty jak v offline, tak i v online režimu. Ovládání aplikace je intuitivní, velice rychle si ho osvojí i uživatel bez odborné znalosti problematiky audioefektů či editace signálu. Aplikaci však ocení především hudebníci, pro které může být zajímavý zejména režim SOLA.

Díky poměrně vysoké latenci je méně praktický režim online. Účelem tohoto režimu bylo však pochopit princip úpravy zvuku v reálném čase. Pokud by byl hlavním cílem aplikace online režim, bylo by vhodnější kvůli menší latenci zvolit jiné komunikační rozhraní se zvukovou kartou (například ASIO nebo WASAPI) nebo se odprostit od PC a samotný online režim implementovat na FPGA či MCU.

Aplikace by se dala kromě uživatelského rozhraní dále rozvíjet několika směry. Prvním směrem by bylo rozšířit portfolio audioefektů, například o nelineární audioefekty (kompresor, noise gate, zkreslení) či v této práci zmíněný reverb. Dále by bylo možné rozpracovat time-stretching jinými algoritmy než SOLA (PSOLA, WSOLA), stejně tak pitch-shifting. Poslední možností je v předchozím odstavci zmíněný online režim, který by bylo možné realizovat pomocí jiného komunikačního rozhraní.

Seznam použité literatury:

[1] ZÖLZER, Udo. *DAFX: Digital Audio Effects*. Vyd. 1. Wiley, 2002. 552 s. ISBN 978-047149078

[2] *Digital Filter Design* [online]. [cit. 24-10-2013] URL: < http://www.mikroe.com/chapters/view/72/chapter-2-fir-filters/>

[3] MARSHALL, David. *Basic Digital Audio Effects* [online]. [cit. 22-03-2010] URL: <http://www.cs.cf.ac.uk/Dave/CM0268/PDF/10_CM0268_Audio_FX.pdf/>

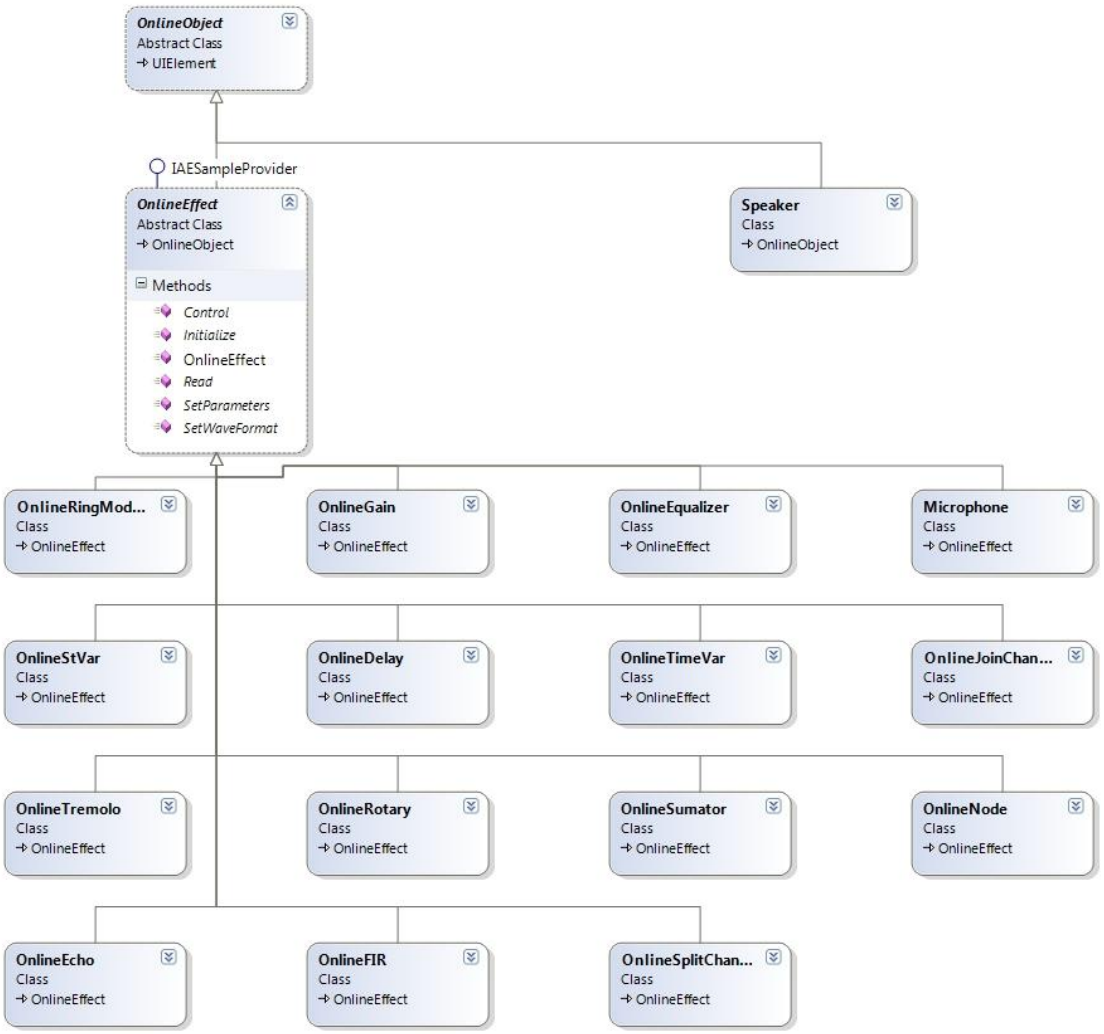
[4] PETZOLD, Charles. *Mistrovství ve Windows Presentation Foundation*. 1. vydání. Computer Press, 2008. ISBN 9788025121412.

[5] *WPF Tutorial.net* [online]. [cit. 12-10-2013] URL: <http://www.wpftutorial.net/>

[6] SMITH, Steven W. *The Scientist and Engineer's Guide to Digital Signal Processing*. 2. vydání. California Technical Publishing, 1999. ISBN 0-9660176-6-8.

7 Přílohy

Příloha A

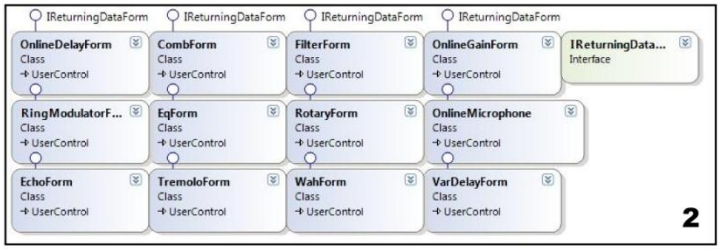


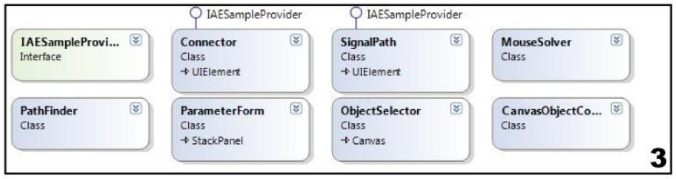
Příloha B

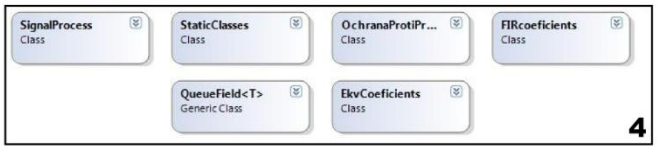

AudioEffect
Abstract Class
Fields
_WaveFormat
Properties
WaveFormat
Methods
GetForm
Initialize
Read
SetParameters
SetWaveFormat
ToString
Events
OnWaveFormat...
Nested Types
Tremolo
Class
AudioEffect
Gain
Class
AudioEffect
StandartVariable
Class
AudioEffect
RingModulation
Class
AudioEffect
Echo
Class
AudioEffect
Equalizer
Class
AudioEffect
UniversalComb
Class
AudioEffect
Rotary
Class
AudioEffect
FIR
Class
AudioEffect
TimeVariable
Class
AudioEffect

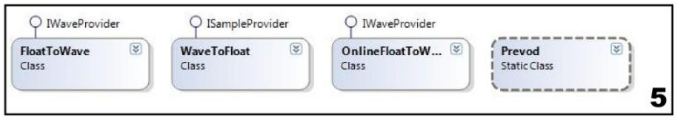
Příloha C

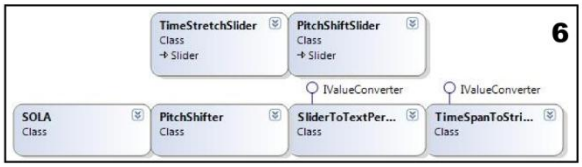

EchoParameter
Struct
EkvParameter
Struct
FIRParameter
Struct
RingModulationPar...
Struct
StandVarParam...
Struct
UniversalComb...
Struct
WahParameter
Struct
GainParameter
Struct
RotaryParameter
Struct
TremoloParame...
Struct
1

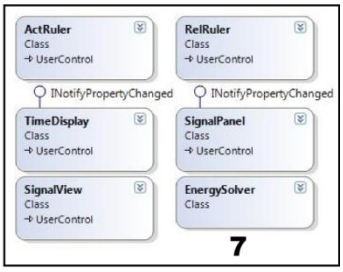

IReturningDataForm
IReturningDataForm
IReturningDataForm
IReturningDataForm
OnlineDelayForm
Class
UserControl
CombForm
Class
UserControl
FilterForm
Class
UserControl
OnlineGainForm
Class
UserControl
IReturningData...
Interface
RingModulatorF...
Class
UserControl
EqForm
Class
UserControl
RotaryForm
Class
UserControl
OnlineMicrophone
Class
UserControl
EchoForm
Class
UserControl
TremoloForm
Class
UserControl
WahForm
Class
UserControl
VarDelayForm
Class
UserControl
2

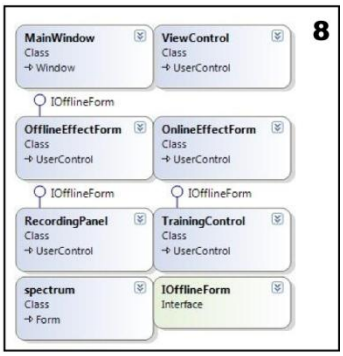

IAESampleProvider
IAESampleProvider
IAESampleProvi...
Interface
Connector
Class
UIElement
SignalPath
Class
UIElement
MouseSolver
Class
PathFinder
Class
ParameterForm
Class
StackPanel
ObjectSelector
Class
Canvas
CanvasObjectCo...
Class
3

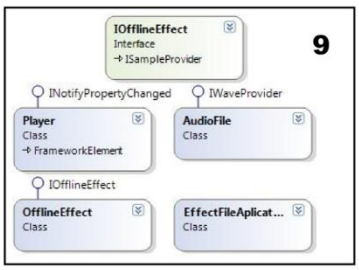

SignalProcess
Class
StaticClasses
Class
OchranaProtiPr...
Class
FIRcoeficients
Class
QueueField<T>
Generic Class
EkvCoeficients
Class
4


IWaveProvider
ISampleProvider
IWaveProvider
FloatToWave
Class
WaveToFloat
Class
OnlineFloatToW...
Class
Prevod
Static Class
5


TimeStretchSlider
Class
Slider
PitchShiftSlider
Class
Slider
6
IValueConverter
IValueConverter
SOLA
Class
PitchShifter
Class
SliderToTextPer...
Class
TimeSpanToStri...
Class


ActRuler
Class
UserControl
RelRuler
Class
UserControl
INotifyPropertyChanged
INotifyPropertyChanged
TimeDisplay
Class
UserControl
SignalPanel
Class
UserControl
SignalView
Class
UserControl
EnergySolver
Class
7


8
MainWindow
Class
Window
ViewControl
Class
UserControl
IOfflineForm
OfflineEffectForm
Class
UserControl
OnlineEffectForm
Class
UserControl
IOfflineForm
IOfflineForm
RecordingPanel
Class
UserControl
TrainingControl
Class
UserControl
spectrum
Class
Form
IOfflineForm
Interface


IOfflineEffect
Interface
ISampleProvider
9
INotifyPropertyChanged
IWaveProvider
Player
Class
FrameworkElement
AudioFile
Class
IOfflineEffect
OfflineEffect
Class
EffectFileAplicat...
Class