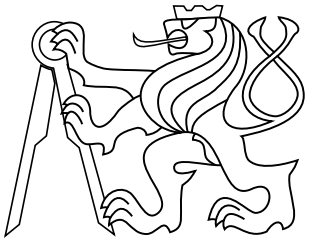
Bakalářská Práce

Prototyp Kytarového Multiefektu

Peter Schmiedt



2015

Vedúci práce: Ing. Adam Sporka, Ph.D.

České vysoké učení technické v Praze
Fakulta elektrotechnická, Katedra počítačové grafiky a interakce

Poďakovanie

Chcem poďakovať všetkým, ktorí mi pomáhali s touto bakalárskou prácou, či už motivačne, vedomosťami alebo obojím.

Špeciálne ďakujem môjmu vedúcemu práce za jeho tipy, rady a know-how, mojím rodičom za trpezlivosť a podporu, Jurajovi za jeho inšpiratívnu hudbu a rady, sestre za motiváciu a všetkým participantom zúčastnených na testovaní.

Bez ich pomoci a podpory by táto bakalárska práca neexistovala.

Prehlásenie

Prehlasujem, že som predloženú prácu vypracoval samostatne a že som uviedol všetky použité informačné zdroje v súlade s Metodickým pokynom o dodržiavané etických princípov pri príprave vysokoškolských záverečných prác.

V Prahe dňa 22. 5. 2015

.....
Peter Schmiedt

Abstrakt

Gitarové efekty boli stále lákadlom pre gitaristov a stále hrajú veľkú rolu v dnešnom hudobnom premysle. Rozpoznávať a vedieť ako funguje široké spektrum gitarových efektov, pre niekoho kto s tým neprichádza často do styku, je obtiažne. Cieľom tejto bakalárskej práce je zoznámiť sa s najvýznamnejšími efektami a implementovať reťazec vybraných efektov do vhodného vstavaného zariadenia. Toto zariadenie následne otestovať s užívateľmi. Táto práca poskytuje informácie pre implementáciu vlastného gitarového multiefektu.

Kľúčové slová

gitara; efekt; reťazec; digitálne spracovanie signálu; C; multiefekt

Abstrakt

Guitar effects were always a precious thing for guitar players and they still play a major role in today's music industry. For somebody who isn't close to guitar effects, to identify and to know how a large spectrum of guitar effects work, is a hard thing to do. The aim of this bachelor's thesis is to get to know some of the most famous guitar effects and to implement an effect chain on a suitable embedded device. After that, to test the device with users. This thesis provides information on implementing your own guitar multi-effect.

Keywords

guitar; effect; chain; digital; DSP; embedded; C; stomp box; multi-effect

Obsah

Skratky a Pojmy

Tu budú definované všetky skratky a pojmy.

DSP	(ang. Digital Signal(Sound) Processing) Digitálne spracovanie signálu (zvuku)
MIDI	(ang. Musical Instrument Digital Interface) technický štandard pre protokol, ktorý slúži na komunikáciu medzi rôznými zvukovými zariadeniami
FIR	(ang. Finite Impulse Response) Filter s konečnou impulzívnou odozvou
IIR	(ang. Infinite Impulse Response) Filter s nekonečnou impulzívnou odozvou
Q	(ang. Quality Factor) Stupeň kvality, parameter Band-Pass filtra
obálkový filter	Filter ktorý sleduje signálovú obálku
LFO	(ang. Low Frequency Oscilator) Oscilátor s nízkou frekvenciou
Doplerov effekt	zmena vĺnovej dĺžky dosiahnutá relatívnym pohybom zdroja a pozorovateľa
gain	Zisk, parameter zosílenia
MIPS	(ang. Million Instructions Per Second) milión inštrukcií za sekundu
I^2C	komunikačný protokol vyvinutý spoločnosťou Philips Semiconductor
kodek	Audio Kodek, zariadenie ktoré premieňa analógový signál na digitálny
debouncing	potlačenie odskoku (kontaktov)

1. Úvod

1.1. Cieľ práce

Cieľom práce je zoznámiť sa s problematikov ohľadom Digitálneho Spracovania Zvuku (DSP) v reálnom čase a s najčastejšie používanými efektmi. Preskúmať aktuálne existujúce riešenia, zvoliť vhodnú hardvérovu platformu a implementovať prototyp na danej platforme tak, aby sa s určitými obmedzeniami dala používať v praxi.

1.2. Existujúce riešenia

Ohľadom DSP pre hudobníkov je mnoho existujúcich riešení od profesionálnych firiem, ktoré svoje plošné spoje a zdrojové kódy nezverejňujú. Preto sa budem zameriavať hlavne na open-source projekty a start-upy.

1.2.1. OpenStomp™Coyote-1

Ako prvý som našiel OpenStomp™Coyote-1. Je to open-source digitálny multiefekt pre gitary a basgitary. Je postavený na na osem-jadrovom procesore „Propeller“ s kmitočtom 80MHz od spoločnosti Parallax. Informácie o kodeku nie sú poskytnuté, ale niektoré zdroje uvádzajú kodek of spoločnosti Wolfson s 44kHz sampling rate a 20-bit hĺbkou.

Je programovaný v Spin-e, čo je programovací jazyk pre Parallax Propeller mikroprocesory. Jednotlivé efekty sú spracované vo forme zásuvných modulov (balíčkov) a dajú sa vymienať za nové, alebo vylepšené efekty. Po zmene efektov je potrebná následná kompilácia. Tvorcovia tohto multiefektu vytvorili k tomuto aplikáciu pod MS Windows „OpenSomp Workbench“, ktorá tieto úkony robí za užívateľa[1].

Multiefekt má microUSB port, s ktorým je možné sa pripojiť k počítaču a pomocou aplikácie nahrať nové efekty.

Efekt si pamätá zoradenie efektov a pre každý efekt aktuálne nastavenia. Nastavenia pre aktuálny efekt sa zobrazujú na malom LCD displeji. Aktuálne nastavenia sa menia pomocou štyroch potenciometrov. Zariadenie si aktuálne pamätá len jedno nastavnenie pre každý efekt. Jednotlivé nastavenia sa menia pomocou štyroch rotačných enkóderov.

Aj keď sa momentálne nevyrába, niekedy sa dal kúpiť za $350. Podľa môjho názoru, je toto zariadenie zbytočne predimenzované (dokonca vie spracovať aj video, načo to slúži, ale autor neuvádza)[1].

Projekt je momentálne neaktívny. Hudobníci dôverujú veľkým a starým značkám ako Roland, Line 6, atď. a dôvod na kúpu iného, než značkového zariadenia je hlavne jeho cena. Autor projektu ale sľubuje, že sa k výrobe chce vrátiť.

Plusy

- efekty ako zásuvné moduly
- grafické rozhranie pre MS Windows

Mínusy

- pamätá si len 1 nastavenie pre každý efekt
- predimenzovaný procesor
- cena $350
- projekt je neaktívny

Obr. 1. OpenStomp ™Coyote-1 [1]

1.2.2. MOD Duo™

Ďalším zariadením je MOD Duo™od spoločnosti Musical Operationg Devices. Jedná sa o veľmi úspešný open-source kickstarter projekt. Toto zariadenie má dva LCD displeje, rotačné enkódery, dva vstupy a výstupy a periférie pre pripojenie externých zariadení (snímačov).

Podobne ako Coyote-1, aj toto zariadenie spolupracuje s PC cez USB a za pomoci pluginu pre Google Chrome sa dá poskladať akýkoľvek reťazec efektov. Existuje aj desktopová verzia aplikácie napísaná v Pythone. Má bohatú komunitu a nové efekty pribúdajú na pravidelnej báze. Komunita zoskupená okolo tohto zariadenia vytvára simulácie starých „old-school“ analógových efektov, ako napr.: Boss DS-1, Ibanez TubeScreamer™, Jimi Hendrix™Fuzz Face®, EHX Big Muff PI, atď. [2]

Grafické rozhranie je v štýle „drag'n'drop“ a nové efekty sa dajú stiahnuť z online repozitára. Existuje aj užívateľmi vytvorené veľké množstvo presetov, ktoré sa taktiež dajú stiahnuť.

Toto zariadenie má možnosť rozšírenia o rôzne periférie. Výrobca ponúka aj Arduino Shield ako jednoduchú knižnicu pre Arduino k zostrojeniu vlastných periférií. Na oficiálnej stránke majú aj video ukážky týchto alternatívnych periférií. Zaujímavé sú napr.: svetelný senzor, akcelerometer, atď.

Zariadenie beží na dvojjadrovom procesore ARM A-7 s kmitočtom 1GHz. Okrem štandartných vstupov a výstupov ponúka toto zariadenie aj MIDI vstup a výstup. Preto je vhodný nielen pre gitary a basgitary, ale taktiež pre iné nástroje a zariadenia (klávesy, externé zvukové karty, atď.)

MOD Duo používa Hi-Fi kodek od spoločnosti Cirrus Logic so samplovaciou frekvenciou na 48kHz a s hĺbkou 24-bit. [2]

Cena tohto zariadenia je približne $350. Čo je primeraná cena vzhľadom k hardvéru, ale aj softvéru v ňom obsiahnutom. Zariadenie je podľa môjho názoru predimenzované, ale za určitým účelom. Dá sa v ňom spraviť komplexný reťazec zapojenia rôznych efektov a funkcií s rôznými nastaveniami a tam už je tá výpočtova sila potrebná.

Momentálne sa toto zariadenie ešte nevyrába. Výrobca neuvádza dátum spustenia výroby. Dá sa predobjednať a tým aj podporiť jeho dokončenie. Existuje funkčný prototyp a momentálne sa pracuje na vývoji komerčného zariadenia. Plánované spustenie výroby je stanovené na 31. Júna 2015.

Zariadenie má potenciál uspieť (nakoľko kick-starter je veľmi úspešný). Cena $350 za samotné zariadenie je stále vysoká, ale ak vezmeme do úvahy rôzne rozšírenia a bohatú komunitu, ktorá aktívne vytvára koncept, tak je to primeraná cena. Navyše investujú veľa peňazí do reklamy, čo by mohlo byť základom komerčného úspechu.

Plusy

- efekty vo forme voľne dostupných modulov
- grafické rozhranie ako plugin pre Google Chrome
- veľká komunita, ktorá už dnes vytvára zaujímavé efekty/presety
- možnosť rozšíriť zariadenie o vlastné periférie pomocou Arduina

Mínusy

- zariadenie ešte nie je vo výrobe

Obr. 2. MOD Duo™[2]

1.2.3. pedalSHIELD for Arduino

Zaujímavý projekt od organizácie/komunity ElectroSmash, ktorá sa zaoberá zverejňovaním elektronických obvodov starých gitarových efektov.

Ich projekt spočíva v tom, že pedalSHIELD sa pripojí k Arduinu, ktoré je vopred naprogramované. Programy sa dajú voľne stiahnúť z ich stránky. Tento „shield“ pre Arduino nemá žiadny displej a dokáže spracovať len jeden efekt naraz. [3]

Ovládanie nie je veľmi intuitívne, pretože kód si musí každý kompilovať sám a programovať do Arduina. Je to ale dobrá alternatíva pre ľudí, čo chcú zároveň experimentovať so zvukom, lebo spolu s týmto shieldom majú vlastnú knižnicu, ktorá uľahčuje proces programovania takýchto efektov. S cenou 50$(bez Arduina) je to cenovo prijateľné riešenie.

Plusy
- cenovo výhodné (50$)
- možnosť vytvoriť si vlastný efekt

Mínusy
- zvláda len jeden efekt súčasne
- neintuitívne ovládanie
- nutnosť zakúpiť osobitne Arduino

Obr. 3. PedalShield s krytom (hore) a bez krytu (dole) [3]

2. Analýza

2.1. Gitarové efekty a ich radenie

Poznáme veľké množstvo gitarových efektov. Sú rôzne spôsoby ako deliť gitarové efekty. Môžeme ich deliť podľa zvukovo-časového hľadiska, podľa hudobných žánrov, podľa metódy spracovania zvuku a podľa funkcie.

Pre moje potreby budem používať delenie gitarových efektov podľa toho ako spracuvávajú zvuk.

Delenie gitarových efektov:

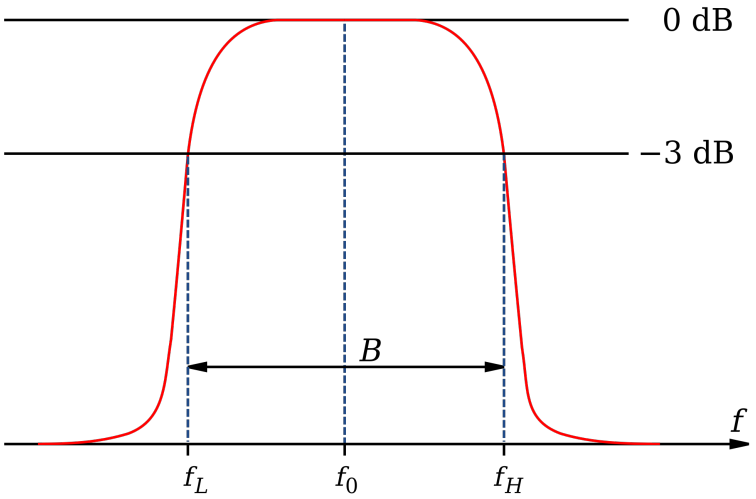
- Základné filtre
- Časovo premenlivé Filtre
- Opozďovacie efekty
- Modulátory
- Nelineárne spracovanie zvuku
- Ostatné efekty

2.1.1. Základné filtre

Low-Pass, High-Pass, Band-Pass a Band-Stop filtre

Tieto filtre odstraňujú alebo redukujú spektrum frekvencií od danej frekvencie. Z implemetačného hľadiska ich môžme deliť na FIR a IIR.

Low-Pass prepúšťa frekvencie nižšie od danej frekvencie. High-Pass naopak prepúšťa frekvencie vyššie. Band-Pass prepúšťa spektrum frekvencií vymedzených intervalom dvoch frekvencií. Band-Stop je presným opakom Band-Pass filtra.



Obr. 4. Ukážka Band Pass filtra z frekvenčného hľadiska, f_L–spodná hranica frekvencie, f_H–horná hranica frekvencie, B–priepustnosť, f_0–stredová frekvencia [4]

Ekvalizér

Ekvalizér pracuje podobne ako Low-Pass, High-Pass, Band-Pass, Band-Stop filtre. Narozdiel od širokopásmových filtrov, zosilňuje len určité (obmedzené) spektrum frekvencie. Na implementáciu Ekvalizéru sa používajú Shelving a Peak filtre. [5]

Shelving Filter

Aktívne zvýrazňuje alebo utĺmuje signál v celom spektre frekvencií daný hraničnou frekvenciou smerom nahor alebo nadol. Veľkosť útlmu alebo zosilnenia určuje GAIN parameter. Narozdiel od klasických útlmových filtrov, dokáže pásmo aj zosilniť. [5]

Peak Filter

Podobne ako Shelving Filter, Peak Filter zvýrazňuje alebo utĺmuje len určité spektrum frekvencií. Taktiež ako pri Shelving Filter, či sa jedná o zvýraznenie/stĺmenie rozhoduje GAIN faktor.

Vďaka Gain faktoru sú tieto filtre vhodné na implementáciu Ekvalizéra.

2.1.2. Časovo Premenlivé Filtre

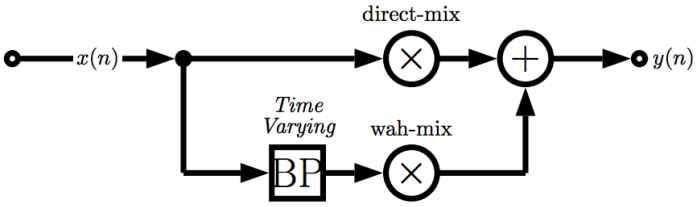
Sú to základné filtre (spomínané v sekcii vyššie), ktoré menia svoje parametre v závislosti na čase.

Wah-Wah Filter

Klasický Wah-Wah Filter, je časovo premenlivý band pass filter s veľmi úzkou priepustnosťou. Horná a dolná hranica frekvencie sa mení v čase smerom nahor a nadol. Filtrovaný signál sa primiešava do pôvodného signálu. To produkuje „wah-wah“ effekt vo zvuku. Existuje niekoľko druhov tohto efektu. [6]

Wah-Wah ovládaný nožným ovládačom s potenciometrom, kde potenciometer mení interval frekvencií Band Pass filtra smerom nahor a nadol. Známy efekt na trhu je „Dunlop CryBaby“

Ďalšou variantou je takzvaný Auto-Wah. Funguje podobne ako Wah-Wah, s tým rozdielom, Band-Pass filter sa neovláda potenciometrom v nožnom ovládači. Namiesto nožného ovládača je interval frekvencií Band-Pass filtra riadený signálovou obálkou (automaticky). Veľmi známy gitarový efekt produkujúci Auto-Wah je „Mu-Tron“. Namiesto signálovej obálky je možné použiť aj LFO. Tento typ je ale málo používaný a vyskytuje sa len ojedinele.



Obr. 5. Diagram Wah-Wah Filtra s časovo premenivým Band Pass Filtrom [5]

Phaser

Pracuje na veľmi podobnom princípe ako Wah-Wah. Namiesto Band-Pass filtra sa používa Band-Stop filter. Najčastejšie na riadenie intervalu frekvencií sa používa LFO. Najznámejším predstaviteľom je „Phaser 90“ od spoločnosti MXR.

2.1.3. Opozďovacie efekty

Opozďovacie efekty sa dajú najlepšie vysvetliť, ak si predstavíme zdroj zvuku vo veľkej miestnosti. Počujeme zvuk priamo zo zdroja zvuku, ale aj zvuk odrazený od stien tejto miestnosti. Tým, že cesta zvuku, ktorý ide priamo a cesta zvuku, ktorý je odrazený od steny je rozdielna, vnímame jeden z týchto zvukov ako „opozdený“. To produkuje efekt zvaný Echo. Naopak zdroj zvuku si môžme predstaviť aj v malej miestnosti. Opakované odrazy sa zmiešavajú a tak menia „farbu“ zvuku.

Na implementáciu týchto efektov sa používa IIR/FIR Comb Filter. Jedná sa o filtre ktoré opozďujú signál a potom ho primiešavajú k pôvodnému signálu (podobne ako signál, ktorý je odrazený od steny). Comb Filter má zvyčajne 2 parametre. O koľko má signál opozdiť a s akou amplitúdou ho má primiešať k pôvodnému signálu.

Vibrato

Je opozďovací efekt s premenlivým časom opozdenia, ktorý simuluje „Dopplerov effekt“. Dopplerov efekt nastáva, ak sa zdroj zvuku pohybuje v smere k poslucháčovi, alebo v smere od poslucháča. Vlnová dĺžka zvuku sa skracuje alebo predlžuje a tak nastáva zmena frekvencie.

Pri Vibrate je implementovaný oneskorovací efekt s premenlivou dĺžkou oneskorenia. Zvyčajne je táto dĺžka ovládana LFO. Oneskorený signál sa nezmiešava do pôvodného signálu, ale púšťa sa priamo na výstup. Typický čas opozdenia je 5-10ms. Vibrato sa dá predstaviť ako zdroj zvuku, ktorý sa pohybuje smerom k poslucháčovi a potom smerom od poslucháča s pravidelným intervalom.

Flanger, Chorus, Slapback, Echo

Jedná sa o efekty, ktoré sa implementujú pomocou Comb Filtru. Každý z nich produkuje špecifický zvuk a preto majú rôzne názvy. V parametroch sa rozlišujú v tom o koľko spozďujú zvuk a aká je modulácia spozdenia, ako je to zobrazené v Tabuľke 1. Špeciálnym prípadom je Flanger a Chorus, kde dochádza k časovej modulácii dĺžky opozdenia. Pri Flangeri je to LFO, ktorý osciluje po sinusoide. Pri Choruse sa skopíruje signál niekoľkokrát a na každý signál sa použije náhodné oneskorenie. [5]

Efekt	Dĺžka opozdenia (ms)	Modulácia
Flanger	0-20	Sinusoida (LFO)
Chorus	10-25	Náhodná
Slapback	25-50	-
Echo	>50	-

Tabuľka 1. Porovnanie opozďovacích efektov [5]

Známe efekty z tejto rady sú napríklad: Boss BF-3 Flanger, EHX Small Clone Chorus, MXR Echo, Boss DD-7 Digital Delay.

2.1.4. Modulátory

Modulátory sú efekty, ktoré menia parametre sinusoidového signálu (amplitúda, frekvencia) na základe zvukového signálu. Poznáme viacero druhov modulátorov. Niektoré efekty, ktoré sme už spomínali taktiež patria do tejto kategórie. Sú to napr.: Flanger, Chorus, Vibrato, ktoré patria medzi tzv. Fázové modulátory. Wah-Wah, Phaser patria medzi typické Amplitúdové Modulátory. Táto téma je veľmi rozsiahla a vedie k pokročilým zvukovým efektom. Vzhľadom na rozsah tejto témy si spomenieme len najdôležitejšie efekty.

Ring Modulation

V Ring Modulation (RM) pôvodný audio signál vynásobí sinusoidovým signálom na určitej frekvencii. Matematicky to vyjadruje rovnica 1 [5].

$$y(n) = x(n) \cdot m(n) \tag{1}$$

Kde y(n) je výstupný signál, x(n) je vstupný signál a m(n) je modulátor. V tomto prípade je modulátorom sinusoidový signál na určitej frekvencii.

Týmto efektom sa dá vytvoriť „robotický“ efekt. V okrajových prípadoch sa dajú vytvoriť až „neľudské“ zvuky. [5] Najznámejší predstaviteľ tohto efektu je „MF-102 Ring Modulator“ od spoločnosti Moog.

Amplitúdový modulátor (Tremolo)

Už ako z názvu vyplýva, tento efekt modifikuje výstupnú amplitúdu signálu na základe modulátoru. Ak je modulátorom sinusoida jedná sa o efekt Tremolo. Dá sa zapísať rovnicou 2 [5].

$$y(n) = [1 + \alpha m(n)] \cdot x(n) \tag{2}$$

Kde y(n) je výstupný signál, x(n) je vstupný signál, α je hĺbka modulácie (1- maximálna modulácia, 0- žiadna modulácia), m(n) je LFO s osciláciou po sínusoide na určitej frekvencii.

Aby táto rovnica dávala zmysel, musí byť výstup z LFO normalizovaný na 1.

2.1.5. Nelineárne spracovanie zvuku

Nelineárne spracovanie zvuku vytvára harmonické alebo neharmonické frekvencie vzhľadom na frekvenčné spektrum, ktoré sa nenachádzajú v pôvodnom signále. Môžme ich radiť do 3 hlavných kategórií:

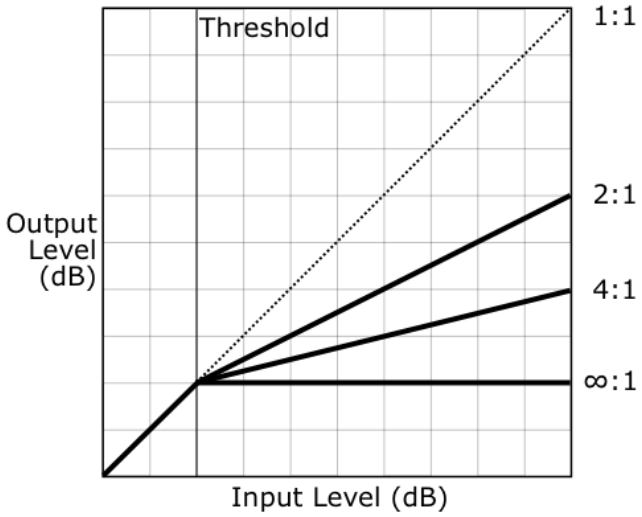
- Dynamické spracovanie zvuku
- Nelineárne spracovanie zvuku
- Signálové budiče

Dynamické spracovanie zvuku

Je postavené na základe signálovej obálky, ktorú využíva algoritmus na modifikáciu signálu. Algoritmus si prepočítava signálovú obálku na premenlivý Gain faktor, ktorý aplikuje na pôvodný signál.

Typickým efektom je Limiter. Ten sa snaží obmedziť veľké amplitúdy signálu a zároveň nezmeniť dynamiku pôvodného signálu. Limiter sa používa zväčša pri štúdiových nahrávkach. Po aplikovaní Limiteru na vstupný signál sa ľahšie zväčšuje amplitúda výstupného signálu(signál už neobsahuje extrémne prvky). Limiter obmedzuje signál nad určitou hranicou, ktorá mu je vopred daná.

Ďalším efektom je Compressor. Je to efekt veľmi podobný Limiteru ale s tým rozdielom, že jeho účel je zmeniť celkovú dynamiku signálu. Compresor ponecháva tiché časti signálu nezmenené, zatiaľ čo hlasité časti sú redukované za pomoci statickej krivky. Podobne ako Limiter aj Compressor má parameter, ktorý určuje hranicu filtra (treshold). Compressor má navyše aj parameter kompresného pomeru.



Obr. 6. Statická krivka Compressor efektu s rôzným kompresným pomerom. [5]

Compressor s vysokým kompresným pomerom (cca. 10:1) plní funkciu Limiteru.

Nelineárne spracovanie zvuku

Má za cieľ pridávať silné harmonické skreslenie (distortion) do pôvodného signálu. Taktiež sa riadi nelineárnou statickou krivkou. Sem patria efekty ako Overdrive, Distortion alebo Fuzz. Sú to typické efekty pre rockovú hudbu.

Overdrive je efektom, ktorý je typický pre klasický rock. Zvuk na nízkej amplitúde je riadený zvukom na vstupe s vysokou amplitúdou. Vzniká „teplý“ skreslený zvuk. Rozdelenie na zvuk s vysokou amplitúdou a nízkou amplitúdou je definované hranicou treshold. Overdrive sa dá definovať trojcestnou funkciou ako je uvedené v rovnici 3 [5].

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

Táto funkcia definuje „x“ ako vstupný signál v rozmedzí od 0 do 1 a podľa toho ho prepočítava. Spodná tretina je zdvojnásobená, stredná tretina je spočítaná pomocou kvadratickej funkcie, aby sa zabezpečila kontinuita výstupného signálu a vrchná tretina je zarovnaná na hranicu 1, čo je maximum.

Distortion/Fuzz je extrémny prípad Overdrive efektu. Je to úplné „odrezanie“ vysokých amplitúd, ktoré potom vytvárajú široké spektrum harmonických tónov. To vytvára

základ Distortion/Fuzz efektu. Tento efekt sa dosahuje znásobením vstupného signálu a potom orezaním špecifickou nelineárnou exponenciálnou funkciou, podľa rovnice 4 [5].

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

V tejto funkcii je α definované ako Gain. To znamená, že α nám zároveň definuje aj množstvo skreslenia. Pri Distortion/Fuzz efekte je časté, že sa pôvodný signál primiešava k skreslenému signálu a to z dôvodu, že pri veľkom skreslení prestáva byť pôvodná frekvencia signálu rozoznateľná. Distortion/Fuzz je jeden z najpoužívanejších efektov na hudobnej scéne. Legendami medzi týmito efektmi sú napríklad Boss DS-2 Turbo Distortion, Dunlop FuzzFace alebo Ibanez TubeScreamer.

2.1.6. Ostatné efekty

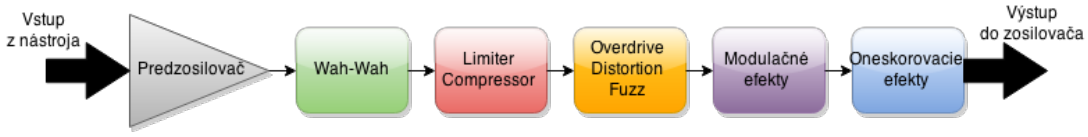
Sem patria efekty ktoré majú príliš špecifické vlastnosti. Napr.: BitCrush efekt.

BitCrush Efekt

Je to digitálny efekt. Princíp efektu je zmenšiť bitovú hĺbku pôvodného vzorku.

2.2. Sériové zapojenie gitarových efektov

Väčšina hudobníkov, ktorá používa rôzne gitarové efekty si ich zapája v podobnom poradí. Dôvodom zaužívaného usporiadania filtrov je estetický výstupný zvuk. Napríklad zaradenie oneskorovacieho efektu na začiatok reťazca vytvára po spracovaní následnými efektmi na výstupe veľmi neforemný zvuk. Preto sa obyčajne umiestňuje na konci reťazca.



Obr. 7. Diagram klasického zapojenia efektového reťazca.

Dá sa s efektovými reťazcami aj experimentovať, ale to sa netýka štúdiových nahrávok.

2.3. Výber vhodnej platformy

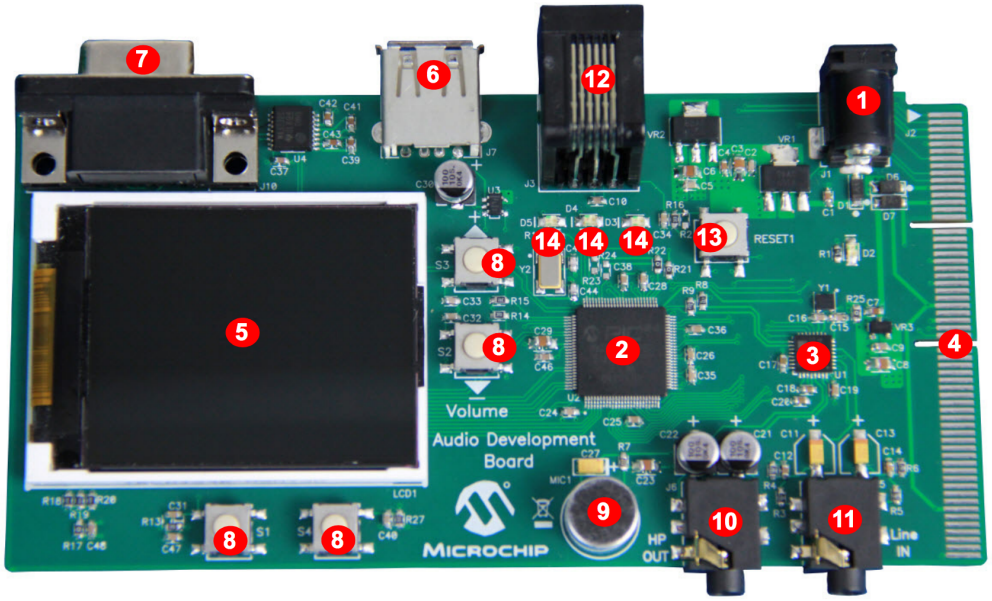
Ako platformu som si zvolil už hotovú vývojovú dosku s osadenými komponentami, nakoľko som sa chcel vyhnúť výrobe vlastnej dosky. Je to časovo a finančne náročná záležitosť.

Požiadavky na dosku som stanovil tak, aby zvládli spracovať zvukový signál v reálnom čase. Vývojová doska by mala obsahovať kodek so samplovaciou frekvenciou aspoň 44,1Mhz a s bitovou hĺbkou aspoň 16 bitov. Doska by mala taktiež mať aj jednoduchý LCD displej a aspoň 2 tlačítka. USB port bol taktiež podmienkou, aby bolo možné nahrať do pamäte rôzne prednastavenia pre dané efekty. Zvukový vstup bol nutnosťou.

Do úvahy som viacero vývojových dosiek od rôznych firiem. Niektoré so spomínaných firiem sú: Atmel, Micorchip, Texas Instruments, Prallax, atď. Zvolil som „Audio Developement Board for PIC32 MCUs“ s typovým označením DM320011 od spoločnosti Microchip.

Vývojovú dosku od Microchipu som si vybral aj preto, že som sa už s ich procesormi stretol na ČVUT vrámci predmetu Struktury a Architektury počítačů.

Táto doska (Obr. 8) obsahuje výkonný procesor(2) „PIC32MX795F512“ s výkonom 80 MIPS s 512KB Flash pamäte a 128KB RAM. Doska ďalej obsahuje 24-bit Hi-Fi kodek(3) s maximálnou vzorkovaciou frekvenciou 48KHz s typovým označením WM8960 od spoločnosti Wolfson. Farebný 2"TFT color LCD displej(5) o rozlíšení 220x176 pixelov. Na doske sa taktiež nachádza aj USB port(6) typu A a RS-232 sériový port(7) [7].



Obr. 8. Vývojová doska „Audio Developement Board“ od spoločnosti Microchip.

2.4. Výber zapojenia konkrétnych efektov

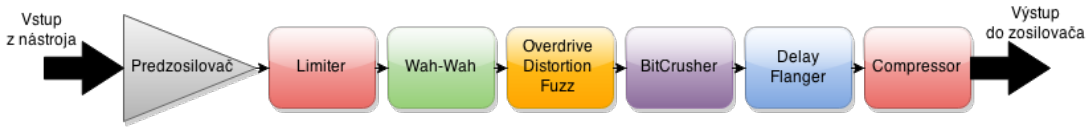
Pri výbere konkrétnych efektov na implementáciu som postupoval podľa najpredávanejších a najčastejšie používaných efektov. Dáta som zbieral z internetových e-shopov ako kytary.cz, muziker.sk, lidlmusic.sk.

Najpredávanejšie efekty boli:

- Distortion
- Overdrive
- Compressor
- Flanger
- Auto-Wah

Na základe tohto zistenia som zostavil reťazec „klasického“ zapojenia efektov. Pre zaujímavosť som do reťazca pridal aj BitCrush efekt. Pri zostavovaní reťazca zapoje-

nia, som priradil Compressor efekt na koniec. Toto zapojenie je typické pre štúdiové nahrávanie. A namiesto Compressor na začiatku reťazca som nahradil Limiterom.



Obr. 9. Schéma konkrétneho navrhovaného zapojenia pre Prototyp gitarového efektu.

2.5. Nahrávanie presetov do zariadenia

Nahrávanie presetov do zariadenia bude prebiehať pomocou USB Flash pamäte, na ktorej sa bude nachádzať nastavovací súbor. Zariadenie sa prepne do režimu príjmu dát a a bude navigovať užívateľa pri postupe nahrávania dát do zariadenia. Ak nahrávanie skončí úspešne, zobrazí sa táto informácia na displeji. Užívateľ sa potom bude môcť prepnúť naspäť na hlavnú obrazovku. Ak sa nahrávanie nepodarí, užívateľ bude o tom informovaný. Možnosť vrátiť sa naspäť na hlavnú obrazovku bude prístupná v akomkoľvek štádiu nahrávania.

3. Implementácia

Implementácia bola realizovaná v jazyku C za pomoci verejne dostupných knižníc a zdrojových kódov poskytnutých firmou Microchip pod licenciou „Microchip Proprietary Licence“. Použitý kompilátor je Microchip MPLAB C32 Compilator v aktuálnej verzii 2.0.2. Ako programovacia a zároveň aj debugovacia jednotka bol použitý „MPLAB ICD 3“.

3.1. Základné nastavenia Procesoru

Kód 3.1 Základné nastavenia procesoru

```
#pragma config UPLLEN = ON          // USB PLL Enabled
#pragma config FPLLMUL = MUL_20     // PLL Multiplier
#pragma config UPLLIDIV = DIV_2     // USB PLL Input Divider
#pragma config FPLLIDIV = DIV_2     // PLL Input Divider
#pragma config FPLLODIV = DIV_1     // PLL Output Divider
#pragma config FPBDIV = DIV_1       // Peripheral Clock divisor
#pragma config FWDTEN = OFF         // Watchdog Timer
#pragma config WDTPS  = PS1         // Watchdog Timer Postscale
#pragma config OSCIOFNC = OFF       // CLKO Enable
#pragma config POSCMOD = HS         // Primary Oscillator
#pragma config IESO   = OFF         // Internal/External Switch-over
#pragma config FSOSCEN = OFF        // Secondary Oscillator Enable (KLO was
    off)
#pragma config FNOSC  = PRIPLL      // Oscillator Selection
#pragma config CP     = OFF         // Code Protect
#pragma config BWP    = OFF         // Boot Flash Write Protect
#pragma config PWP    = OFF         // Program Flash Write Protect
#pragma config ICESEL = ICS_PGx2    // ICE/ICD Comm Channel Select
#pragma config DEBUG  = ON          // Background Debugger Enable
```

Vo vyššie uvedenom zdrojovom kóde nájdeme základné nastavenia procesoru „PIC32 MX795F512L“. Spomeniem len dôležité nastavenia, ostatné sú východzie podľa oficiálneho datasheetu. „UPLLEN“ a „FPLLMUL“ sú hlavné nastavenia pre USB. „UPLLEN“ zapína napätie 5V na USB a „FPLLMUL“ delí aktuálne hodiny na procesory, aby boli synchronizované s hodinami USB, ako to definuje protokol USB. „POSCMOD“ nastavuje odkial bude brať procesor hodiny. Aktuálne procesor berie hodiny z externého 8MHz kryštálu. Procesor má aj vnútorný oscilátor, ktorý je ale nespoľahlivý a pracuje správne len pri presnom napätí, kdežto externému kryštálu malé výkyvy napätia nevadia. Správne hodiny sú pri protokole USB, ale aj pri spracovaní zvuku extrémne dôležité. Výkyvom napätia sa nedá zabrániť. Dôvod pre dôležitosť správne pracujúceho oscilátoru je synchronizácia s pripájanými zariadeniami, ktoré majú väčšinou vlastný oscilátor. [8]

3.2. Použité knižnice

V tejto časti popíšem všetky externé knižnice a zdrojové kódy, ktoré som použil pri implementácii.

3.2.1. Codec Driver Library pre Wolfson WM8960

Táto knižnice nie je priamo dodávaná s kompilátorom a to z dôvodu, že si vyžaduje modifikáciu jej zdrojového kódu pre rôzne typy mikroprocesorov. Bolo potrebné zmeniť na ktorých portoch komunikuje mikroprocesor s kodekom a kde má kodek vstupy a výstupy. Ďalej bolo potrebné správne nastaviť časovače, aby komunikácia cez I^2C protokol s kodekom fungovala správne. Čo sa ale týka potrebnej inicializácie I^2C protokolu a posielania a príjmu dát, bolo už implementované správne a nevyžadovalo si to ďalsie úpravy.

Následne v hlavnom programe je potrebné túto knižnicu(ďalej len kodek) zinicializovať metódou `WM8960CodecOpen()`, následne nakonfigurovať kodek(vzorkovacia frekvencia, ktorý vstup a výstup použiť) a nakoniec ho zapnúť metódou `WM8960CodecStartAudio()`. Kodek potom vzorkuje signál do bufferu o veľkosti 1024 vzoriek. Kodek posiela dáta do mikroprocesoru len na vyžiadanie metódou `WM8960CodecRead()`. To znamená, že pri vzorkovacej frekvencii 48KHz, je potrebné túto metódu volať 46.875 za sekundu, aby nedošlo k strate/prekrývaniu dát. Zapisovanie dát do kodeku funguje na podobnom princípe, ale s metódou `WM8960CodecWrite()` [9].

3.2.2. Knižnice pre ovládanie LCD displeja

OTM2201A LCD controllers driver

Mikroprocesor komunikuje s LCD displejom za pomoci OTM2201A. Je to driver, ktorý zabezpečuje vykresľovanie pixelov na displej. Táto verzia LCD displeja má tento radič vstavaný priamo v displeji. Vie rozpoznávať 262144 farieb.

Táto knižnica je taktiež voľne dostupná na oficiálnej stránke Microchip. Nie je dodávaná spolu s kompilátorom a preto je potrebne manuálne prekladať zdrojové kódy. Zdrojový kód je taktiež potrebné modifikovať. Stačí len definovať na ktorých portoch je zapojený OTM2201A radič [9].

Microchip Graphics Library

Táto knižnica je oficiálne dodávana s kompilátorom C32 a aktívne komunikuje s „OTM2201A LCD controllers driver“ a slúži ako nadstavba pre ňu.

Knižnicu je potrebné inicializovať príkazom `InitGraph()`. V hlavnej slučke je sa potom volá funkcia `GOLDraw()`, ktorá vykreslí na displej požadované pixely(text, widgety alebo obrázky) a zavolá „callback“ metódu `GOLDrawCallback()`, ktorú je potrebné implementovať v hlavnom programe manuálne.

Knižnica je stavaná tak, aby uľahčila prácu s vykresľovaním útvarov na displej. Obsahuje metódy pre vykreslenie rôznych geometrických útvarov ako štvorec, kruh, ovál atď. Obsahuje aj vykresľovanie textu v konkrétnym fontom. Na to slúži metóda `SetFont()` a následne pre vykreslenie konrétneho textu, metóda `OutTextXY()`. Môžme si všimnúť, že metóda `OutTextXY()` si vyžaduje aj parametre pre konkrétnu pozíciu na displeji. Knižnica si pamätá pointer kde sa skončilo posledné vykresľovanie a tak vie nadviazať na tento pointer. Ak by sme za volaním metódy `OutTextXY()` zavolali metódu `OutText()` text by pokračoval ďalej.

Knižnica vie pracovať aj s „widgetmi“, ale to pre naše potreby nebudeme potrebovať. Widgety slúžia ako wrapper na vykresľovanie objektov/útvarov. Funguje to na princípe, že programátor si urobí premennú typu customWidget (napríklad tlačítka), kde zadá parametre ako farba, text, veľkosť, atď. a pre opätovné vykreslenie tohto tlačítka alebo stavu tlačítka (widget môže mať aj rôzne stavy, napr. ON/OFF), len zavolá funkciu na vykresľovanie widgetov.

Taktiež v tejto knižnici funguje „Message Callback“. Týmto spôsobom si widgety predávajú aktuálne stavy. Každý stav sa dá definovať pomocou `enum`. Za každým ako sa začne spracuvávať správa z tejto knižnice sa zavolá callback funkcia `GOLMsgCallback()`, ktorú treba implementovať v hlavnom programe manuálne. [10]

3.2.3. Microchip USB Library

Táto knižnica je štandartne dodávaná s kompilátorom C32. Túto knižnicu budeme používať pre komunikáciu s USB zariadením. Knižnica má dve časti. Prvá časť je ak chceme aby naše zariadenie sa správalo ako „host“ a druhá ako „slave“. Pre naše potreby nám bude stačiť časť, ktorá inicializuje naše zariadenie ako „host“, keďže plánujeme nahrávať presety za pomoci USB Flash pamäti.

Knižnica pre USB host sa incializuje metódou `USBInitialize()`. Pred tým sa však musí nastaviť napätie 5V na VBUS príkazom `AD1PCFGbits.PCFG5 = 1;`. Port „RB5“ spína stabilizátor napätia pre USB. [7]

Ďalej musíme pravidelne kontrolovať, či sa nepripojilo zariadenie príkazom USBHostMSDSCSIMediaDetect(). Tento príkaz vracia `TRUE`, ak zdetekoval „SLAVE“ zariadenie pripojené do USB portu na doske. Ďalej môžme spracovávať „File System“ na pripojenej USB Flash pamäti.

3.2.4. Memory Disk Drive File System library for PIC32 (MDD FS)

Táto knižnica je taktiež zahrnutá v kompilátore C32. Umožňuje nám manipulovať so súbormi. Po úspešnom pripojení USB Flash pamäte je potrebné zinicializovať túto knižnicu príkazom `FSInit()`. Táto knižnica má podobné názvy a pracuje na podobnom princípe ako `<cstdio>` knižnica pre `C`. Príkazy sa odlišujú tým, že pred každým príkazom je predpona `FS`. Po skončení práce s MDD nesmieme zabudnúť ukončiť spojenie príkazom `FSfclose()`.

3.3. Hlavná slučka programu

Hlavná slučka programu nastavuje všetky hárdverové parametre (vstupy, výstupy, časovače, komunikácia s kodekom a s USB, atď.) Všetky tieto procedúry si volám vo funkcii `InitializeHardware()`, ktorú popíšem neskôr. Po inicializačných nastaveniach program vstúpi do nekonečnej slučky „while(1)“. V každom cykle sa najprv obslúži USB port volaním funkcie `USBTasks()`. Týmto sa vykonajú všetky úkony čakajúce na spracovanie. Potom sa zavolá metóda `GOLDraw()`, ktorá obsluhuje vykresľovanie displeja. Táto metóda vracia „FALSE“, ak vykresľovanie ešte neskončilo. Následne sa zavolá callback funkcia `GOLDrawCallback()`, v ktorej sa obslúžia príslušne stavy aplikácie a nastavia zmeny na displeji.

Metóda `CheckButtons()` kontroluje stav tlačítok a zabezpečuje „debouncing“. Táto hlavná slučka je zostavená podľa oficiálnej dokumentácie a odporúčaní pre PIC32 procesory.

Kód 3.2 Hlavná slučka programu

```
int main(void) {

    // Initialize hardware.
    InitializeHardware();

    // Main loop.
    while (1) {
        USBTasks();

        if (GOLDraw()) {
            CheckButtons(&graphicsMessage);
            if (graphicsMessage.uiEvent != EVENT_INVALID) {
                GOLMsg(&graphicsMessage);
            }
        }
    }
}
```

3.4. Incializácia hardvéru

Inicializácia hardvéru prebieha vo funkcii `InitializeHardware()`. Funkcia je príliš dlhá na výpis, preto uvediem len dôležité činnosti, ktoré vykonáva.

- nastaví časovače
- nastaví vstupy a výstupy (tlačítka, led, kodek, displej)
- inicializuje všetky potrebné premenné
- inicializuje kodek, USB a grafickú knižnicu
- zmaže displej
- vykreslí úvodnú obrazovku

3.5. Metódy GOLDraw() a GOLDrawCallback()

Čo robí metóda `GOLDraw()` sme si popísali v stati 3.2.2. Metóda `GOLDrawCallback()` sa volá stále na začiatku metódy `GOLDraw()`. V tejto metóde sa nachádza jednoduchý switch, ktorý rozhoduje čo sa bude diať podľa aktuálneho stavu zariadenia. Vykresľovacia funkcia sa vykonáva 46.875-krát za sekundu, je vhodné do tejto metódy zabudovať aj obsluhu kodeku. Táto metóda obluhuje, okrem kodeku, aj USB.

Kód 3.3 Telo metódy GOLDrawCallback()

```
switch (stateScreen) {
   case STAV_ZARIADENIA:
      //urob potrebne akcie
      break;

   case INY_STAV_ZARIADENIA:
      //urob potrebne akcie
      break;
}
return 1;
```

3.6. Obsluha USB

Na načítanie nových presetov budeme pripájať k násmu zariadeniu USB flash pamäť, na ktorom bude textový súbor s konkrétnymi nastaveniami.

3.6.1. Štruktúra nastavovacieho súboru

Nastavovací súbor je jednoduchý textový súbor. Program si pamätá 16 presetov. Štruktúra tohto súboru zahŕňa nastavenie každého s presetov. Jednotlivé presety sú od seba oddelené prázdnym riadkom. Zakončenie riadkov je vo formáte CLRF(Windows Style). Kódovanie je ASCII. Každý riadok obsahuje číslo, ktoré nastavuje určitý parameter každého efektu. Súbor musí mať špeciálny názov „FX.TXT“ a musí byť umiestnený v „root“ adresári. Štruktúra nastavovacieho textového súboru nepozná komentáre, preto je dôležité postupovať podľa nastavovacej šablony v ktorej je každý riadok popísaný. Súbor nemá žiadne obmedzenia pre nastavenia efektov, aby mal užívateľ možnosť využiť každý efekt na maximum. Význam každého z riadkov a odporúčané hodnoty nastavení je uvedený v šablóne nastavovacieho súboru, Kód 3.4.

Kód 3.4 Šablona nastavovacieho súboru

```
1.  // Fuzz/Distortion (0-OFF, 1-ON)
2.  // Fuzz - Gain Faktor (20-100)
3.  // Fuzz - mix original (1-len skresleny zvuk - 20)
4.  // Overdrive (0-OFF, 1-ON)
5.  // Overdrive - Treshold (1500-5000)
6.  // Auto-Wah (0-OFF, 1-ON)
7.  // Auto-Wah - rychlost efektu (1000-3000)
8.  // Auto-Wah - dolna hranica frekvencie udavana v Hz. (300-1000)
9.  // Auto-Wah - horna hranica frekvencie udavana v Hz. (1000-2000)
10. // Auto-Wah - quality factor (4-10)
11. // Compressor - (0-OFF, 1-ON)
12. // Compressor - Kompresny pomer (2-20)
13. // Compressor - hranica (500-2000)
14. // BitCrush (0-OFF, 1-ON)
15. // BitCrush - kolko bitov ponechat
16. // Limiter (0-OFF, 1-ON)
17. // Limiter - hranica (500-2000)
18. // Flanger (0-OFF, 1-ON)
19. // Flanger - frekvencia LFO v Hz (1-5)
```

Detailnejší popis efektov a čo ktorý parameter robí si povieme v následujúcich častiach.

3.6.2. Spracovanie nastavovacieho súboru

Samotná detekcia USB zariadenia prebieha v callback funkcii `GOLDrawCallback()`. Ak sa súbor úspešne otvorí, zmení sa stav zariadenia z `STATE_SHOW_INSERT_USB` na `STATE_SHOW_USB` a zavolá sa funkcia s názvom `ReadUSBData()`. Táto funkcia spracuje dáta z požadovaného súboru a nahrá dáta do štruktúry `FX_DATA_STRUCT`. Následne ukončí prácu so súborom a zároveň sa na displeji zobrazí hlásenie o ukončení činnosti. Ak však USB Flash pamäť neobsahuje tento súbor, alebo je pripojená pamäť, ktorej „File System“ knižnica MDD nepodporuje, neudeje sa žiadna akcia. [10]

3.7. Obsluha LCD Displeja

Obsluhe LCD displeja už bola čiastočne spomenutá v stati 3.4. Teraz si to priblížime viac. Funkcia `GOLDrawCallback()` má v tele `switch`, ktorý vykonáva akcie, podľa toho aký je stav zariadenia. Všetky stavy aplikácie si ukážeme detailnejšie v kóde 3.5.

Kód 3.5 Enum s všetkými stavmi aplikácie

```
typedef enum {
    STATE_DISPLAY_SHOW_LOOPBACK,  //pripravi sa na zobrazenie loopback efektu
    STATE_SHOW_CLEAR_LOOPBACK,   //vycisti displej pre loopback
    STATE_SHOW_LOOPBACK,         //nakresli na displej nastavenia efektov a
        spusti efektovu slucku
    STATE_DISPLAY_SHOW_USB,      //pripravi sa na zobrazenie USB
    STATE_SHOW_CLEAR_USB,      //vycisti displej pre USB
    STATE_SHOW_INSERT_USB,     //vykresli na obrazovke instrukcie o vlozeni usb
    STATE_SHOW_USB,            //spracuvava data s usb
    STATE_SHOW_USB_DONE        //skonci spracovanie dat z usb
} STATES_GRAPHICS;
```

Pre rôzne vykreslenie displeja sa volajú rôzne funkcie ako napr.: `CreateScreenLoopback()`, `RedrawScreenLoopback()` alebo `CreateScreenUSB()`.

3.8. Obsluha tlačítok

Obsluha tlačítok prebieha v hlavnej slučke, kde sa volá funkcia `CheckButtons()`. V tele tejto funkcie je obslúžené každé tlačítko zvlášť a zároveň je aplikovaný „debouncing“ tlačítok. Debouncing robíme podľa odporúčaného postupu z oficiálnej dokumentácie Microchipu.

Kód 3.6 Obluha a debouncing tlačítka

```
#define DEBOUNCE_TIME (100 * (TICKS_PER_SECOND / 1000ull))

static BOOL switchBouncing = TRUE;
static UINT32 switchTime = 0;

if (switchBouncing) {
   if (BUTTON == 0) {
        switchTime = TickGet();
   } else if (TickGet() - switchTime >= DEBOUNCE_TIME) {
      switchBouncing = FALSE;
   }
} else if (BUTTON == 0) {

   //tlacitko bolo stlacene

   switchBouncing = TRUE;
}
```

Je definovaná konštanta `DEBOUNCE_TIME`, s ktorou porovnávame čas ako dlho bolo tlačítko stlačené. Ak je čas stlačenia tlačítka väčší ako debounce time, program vyhodnotí, že sa jedná o reálne stlačenie tlačítka a nie o zákmit. Premenná `switchBouncing` registruje stlačenie tlačítka ak je nastavená `FALSE`. Ak je tlačítko stlačené `BUTTON = 0`, tak sa do premennej `switchTime` zapisuje aktuálny „tick“ na časovači. Tick je aktuálny

stav časovača. Akonáhle tlačítko pustíme, porovná rozdiel časov aktuálneho „ticku“ a posledného a ak je väčší ako `DEBOUNCE_TIME`, tak tlačítko bolo stlačené. Premenná `switchBouncing` sa nastaví na `FALSE`, zbehne obsluha stlačeného tlačítka a opetovne sa nastaví `switchBouncing` na `TRUE` [8].

3.9. Efektová slučka

Efektová slučka, `AudioLoopback()` sa volá v `GOLDrawCallback()`. Pred prvým zavolaním sa reštartuje kodek. Na začiatku slučky sa nachádza funkcia `WM8960CodecRead()`, ktorá načíta dáta z kodeku. Po spracovaní zvukového signálu sa všetko odošle naspäť do kodeku zavolaním funkcie `WM8960CodecWrite()`. Detajlnejší opis efektov bude vysvetlený v ďalších statiach. Poradie efektov je fixné, avšak dajú sa vypínať a zapínať podľa potreby.

3.10. Compressor/Limiter

Compressor utlmuje signál nad určitou hranicou podľa aktuálnej amplitúdy(obálky). Funkcia `Compressor(int i)` stále spracováva postupne len jednu vzorku. Ak je obálka väčšia ako treshold, signál je násobený kompresným koeficientom, ktorý si vypočítame vzorcom 5 [5].

$$g(n) = \frac{e(n)}{T}^{\frac{1}{R}-1} \tag{5}$$

Parametre vzorca sú: G(n) – kompresný koeficient, T – hranica (treshold), e(n) – obálka, R – kompresný pomer.

Ak je signálová obálka menšia ako treshold, ponechá aktuálny signál na pôvodnej hodnote.

Limiter je vlastne Compressor s vysokým kompresným pomerom, ktorý je v programe danný fixne (10).

3.11. Distortion/Fuzz

Distortion/Fuzz najprv znásobí(zosilní) signál o Gain faktor, potom na neho aplikuje matematickú funkciu 6 [5].

$$z(n) = sgn(x(n)) \cdot (1 - e)^{-\left|\frac{x(n)\cdot G}{32767}\right|} \cdot 32767 \tag{6}$$

Kde x(n) – vstupný signál, z(n) – dočasný kontainer pre polospracovaný signál, G – gain faktor. Dannú rovnicu delíme a opetovne násobíme číslom 32767, čo je maximálne číslo pre `INT16`.

Dočasný signál zmiešame s pôvodným pomocou zmiešavacej rovnice 7 [5] .

$$y(n) = MIX \cdot z(n) + (1 - MIX) \cdot x(n) \tag{7}$$

kde y(n) – výstupný signál, MIX – mixovací faktor[1 - iba sfiltrovaný signál], x(n) – pôvodný signál, z(n) – signál z predošlej rovnice 6.

3.12. Overdrive

Funkcia efektu overdrive je popísaná v sekcii 2.1.4. Následne je uvedená jeho implementácia.

Kód 3.7 Implementácia efektu Overdrive

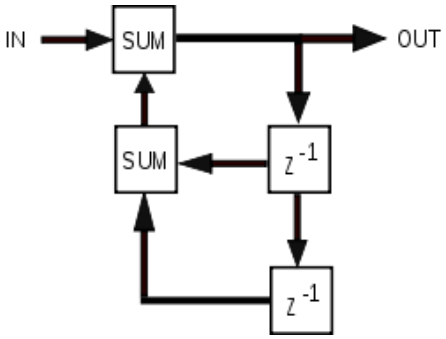
```
INT16 current = abs(Sin[i].leftChannel);

if (current < fxCurrentData->overdriveTreshold) { //spodna tretina
   Sin[i].leftChannel *= 2;
} else if (current < fxCurrentData->overdriveTreshold * 2) { //stred
   if (Sin[i].leftChannel > 0) { //+-?
      Sin[i].leftChannel = (3 - pow(2 - 3 * Sin[i].leftChannel /
          tresholdTimesThree, 2)) / (3 / tresholdTimesThree);
   } else {
      Sin[i].leftChannel = -(3 - pow(2 - 3 * Sin[i].leftChannel /
          tresholdTimesThree, 2)) / (3 / tresholdTimesThree);
   }
} else { // horna tretina
   if (Sin[i].leftChannel > 0) {//+-?
      Sin[i].leftChannel = tresholdTimesThree;
   } else {
      Sin[i].leftChannel = -tresholdTimesThree;
   }
}
```

Táto funkcia rozhoduje v ktorej tretine sa nachádza aktuálny signál. Ak sa nachádza v spodnej tretine, tak signál zdvojnásobí. Ak sa nachádza v strednej tretine tak na signál sa aplikuje exponenciálnu funkciu a ak sa signál nachádza v hornej tretine, tak signál zarovná na hornú hranicu, tak ako je popísané vo vzorci 3.

V strednej a hornej tretine sa program rozhoduje, či má signál záporné alebo kladné hodnoty a podľa toho priraďuje aj znamienko.

3.13. Auto-Wah

Auto-Wah je komplikovanejší efekt na implementáciu. Je potrebné najprv implementovať „Band Pass“ filter, ktorý môže meniť svoje parametre z časového hľadiska a to konkrétne stredovú frekvenciu. Implementoval som Band-Pass filter 2. rádu, typu IIR. Vypočítava sa za pomoci koeficientov, ktoré musíme z časového hľadiska meniť.



Obr. 10. Schéma typického IIR filtra. [11]

Koeficienty násobíme s pôvodným signálom a sčítavame (ak sa jedná o koeficienty „B“) alebo odčítame (ak sa jedná o koeficienty „A“). a tak dostávame vzorku sfiltrovaného signálu.

V metóde `BandPassInitialize()` si inicializujeme všetky potrebné premenné a filter objekt `H`. Metódou `BandPassProcess()` sfiltrujeme aktuálnu vzorku a metóda `BandPassSetup()` slúži na načítanie nových koeficientov do filtra.

Teraz, keď máme implementovaný Band-Pass filter, môžme implementovať obsluhu. V metóde `AutoWah_init()` inicializujeme Band Pass filter a vypočítame všetky potrebné koeficienty. V efektovej slučke potom pravidelne voláme funkciu `AutoWah_process()`, ktorá volá Band Pass filter na aktuálnu vzorku a následne sa zavolá funkcia `AutoWah_sweep()`, ktorá sleduje obálku signálu a podľa toho nastavuje koeficienty Band-Pass filtra. [12]

3.14. BitCrush

Bitcrush je jednoduchý efekt, ktorý zoberie 16-bit vzorku signálu a prevedie ju na vzorku s nižšou bitovou hĺbkou. Tým dostávame efekt prehrávania zvukov na starých zariadeniach.

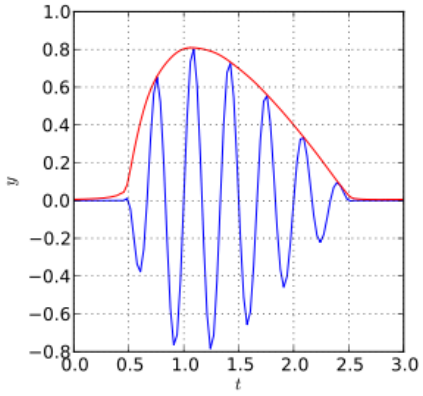
Kód 3.8 Implementácia BitCrush efektu

```
y = x & (-1 << (16 - bitsToKeep));
```

Tento efekt zoberie pôvodný signál a urobí bitový `AND` na -1 (-1 má binárny zápis v signed registroch samé jednotky) a „shiftne“ register smerom doľava o požadovaný počet bitov. Tým získame prevod z 16-bit hĺbky do nižšej bez akejkoľvek kvantizácie.

3.15. Flanger

Flanger je oneskorovacím efekt s premenlivou dobou oneskorenia (1ms-15ms). Dobu oneskorenia riadi LFO na nízkej frekvencii (1-5Hz). Efekt si zinicializujeme metódou `FlangerInit()` a potom na každej vzorke voláme metódu `FlangerProcess()` a hneď za ňou obsluhu LFO `FlangerSweep()`. Implementácia oneskorovacieho buffera obsahuje tzv. „Forward Feed“. Ten zabezpečuje kontinuitu signálu pri zmene času oneskorenia. [5]



Obr. 11. Ukážka signálovej obálky (červená) z pôvodného signálu (modrá) [13]

3.16. Envelope (obálka)

Obálku detekujeme pomocou veľmi jednoduchého Low Pass filtra (IIR), ktorý aplikujeme na absolútnu hodnotu signálu. Vo funkcii `EnvelopeInit()` si vypočítame potrebné koeficienty (B0, B1, B2, A1, A2) [6] a vo funkcii `Envelope()` tieto koeficienty násobíme s pôvodným signálom. Nakoniec v premennej `y[2]` dostaneme obálku pre aktuálnu vzorku. Obálka signálu v závislosti na čase je popísaná v Obr. 11.

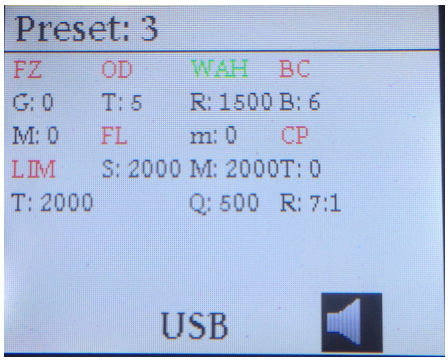
4. Grafické rozhranie a test s užívateľmi

4.1. Grafické rozhranie

Grafické rozhranie je intuitívne a detailne popisuje užívateľovi čo sa práve so zariadením deje (v akom stave sa nachádza). Ak zapneme zariadenie, privíta nás uvítacia obrazovka.

Obr. 12. Uvítacia obrazovka na zariadení.

Uvítacia obrazovka je nastavená tak, aby sa zobrazila po dobu 3 sekúnd. Potom sa zapne hlavná slučka programu. Na displeji sa nám zobrazí hlavná obrazovka, ktorá obsahuje dáta o aktuálnom nastavení efektov, ako to vidíme na 13.



Obr. 13. Hlavná obrazovka

V hornej časti sa nám zobrazuje aktuálny preset. Preset sa mení pomocou tlačítok „plus“ a „mínus“, ktoré sa nachádzajú vpravo od displeja.

Každý efekt je pomenovaný skratkou (kôli nedostatku miesta) a o tom, či je zapnutý alebo nie, nám ukazuje farba písma názvu efektu. Ak je farba písma zelená, efekt je zapnutý a ak je farba červená, efekt je vypnutý.

Skratky: FZ – Fuzz(Gain, Mix), LIM – Limiter(Treshold), OD - Overdrive(Treshold), FL – Flanger(Speed), WAH - Auto-Wah(Rate, Min, Max, Quality factor), BC – BitCrush(Bits to keep), CP – Compressor(Treshold, Compress Ratio).

V spodnej časti obrazovky sa nachádza text „USB“ a ikonka reproduktoru. Priamo pod týmito ikonami sa nachádzajú ďalšie 2 hardvérove tlačítka. Tlačítkom pod textom „USB“ sa prepína zariadenie do režimu príjmu dát z USB Flash pamäťi. Tlačítkom pod ikonkou reproduktoru sa prepína stíšenie zvuku. Pri stíšenom zvuku sa ikonka zmení na reproduktor s krížikom.

počas príjmu dát z USB Flash pamäťi, je užívateľ navádzaný textovými správami na displeji. V tomto móde je funkčné iba jedno tlačítko s popisom „FX“, ktoré vráti užívateľa naspäť na hlavnú obrazovku.

4.2. Test s užívateľmi

Cieľom testovania, je nájsť užívateľské chyby a získať spätnú väzbu o celkovom dojme multiefektu. Pre testovanie som vybral celú obsluhu zariadenia a rozdelil som ju do 3 scénarov.

4.2.1. Scénare

1. Scénar

Užívateľ dostane k dispozícii už správne zapojené zariadenie a USB Flash pamäť s vopred nahraným súborom s presetmi.

Úloha znie: „Nahrajte do zariadenia prednastavenia, ktoré sa nachádzajú na USB Flash pamäťi.“

2. scénar

Ďalej ma užívateľ za úlohu zvoliť preset číslo 4. Číslo presetu mu ale nie je prezradené. Je mu prezradený len názov efektu.

Úloha znie: „Zvoľte na zariadení efekt Auto-Wah.“

3. scénar

Užívateľ po úspešnom zvolení efektu je podporený aby si zahral na hudobnom nástroji a vyskúšal aj iné presety.

Uloha znie: „Skúste si zahrať a poprípade vyskúšať aj iné Presety.“

4.2.2. Post-test dotazník

Z Post-test dotazníku vyplynulo, že užívatelia mali najviac problémov s 1. úlohou (1. scénarom). Báli sa zapojiť USB flash pamäť do zariadenia zo strachu, že dôjde k poškodeniu. S 2. a 3. úlohou už nemali problémy.

Zadanie bolo zrozumitelné každému.

Participantom sa zariadenie páčilo, ale to či by ho využívali v praxi neboli participanti jednotní. Vznikli dve majoritné skupinky, z ktorých jedná tvrdila, že by zariadenie určite používali a druhá tvrdila

4.2.3. Nájdené problémy a ich prípadné riešenie

Tlačítka

Každý participant si prehodil tlačítka na prepínanie presetov. Umiestnenie tlačítok má v tomto prípade prednosť pred značkami + a -, ktoré su vedľa daných tlačítok.

Participanti si mysleli, že pri stlačení tlačítka ktoré je vizuálne nižšie, sa číslo presetu inkrementuje. Nie je to závažná chyba.

Riešenie je veľmi jednoduché. Stačí prehodiť v deklarácii tlačítok príslušné porty a ich funkcia bude vymenená.

Kryt

Participanti sa boli nesmelí, keď videli „holú“ dosku so súčiastkami a pripojenými káblami. Participanti museli byť posmelení, aby sa nebáli zasunúť USB Flash pamäť do dosky. Tento problém je mierne závažný.

Riešenie je jednoduché. Zariadenie by potrebovalo ochranný kryt.

4.2.4. Zhodnotenie zariadenia

Zariadenie malo pozitívne ohlasy. Intuitívne sa ovládalo a na kvalitu efektov boli pozitívne ohlasy.

5. Záver

Cieľom tejto bakalárskej práce bolo navrhnúť a naprogramovať Prototyp gitarového multiefektu. Túto úlohu sa podarilo úspešne splniť. Navrhnuté efekty sú implementované správne a boli otestované s reálnymi hudobnými nástrojmi a hudobnou aparatúrou.

Pri testovaní som ale narazil na určité hardvérové obmedzenia. Platforma napriek deklarácii parametrov výrobcom nedosahuje požadovaný výkon pre danú aplikáciu. V mojich prvotných výpočtoch náročnosti sa ukazoval výkon procesora 80MIPS ako niekoľkonásobne prevyšujúci potrebu výkonu pre výpočty a spracovanie signálu. Obsluha displeja a tlačítok mala predstavovať iba zlomok výkonu a času procesora.

V praktickom testovaní sa však ukázalo, že obsluha displeja je ďaleko náročnejšia na čas procesora oproti pôvodným predpokladom. V prípade, že boli zapnuté viac ako 3 efekty súčasne, dochádzalo k neúmernému prerušovaniu procesora pri pričom vznikali výpadky vo spracovávanom zvuku. Následným ladením a optimalizácie aplikácie som zistil, že v prípade samotného spracovávania zvuku je výkon procesora dostatočný. Pridaním rutín pre obsluhu displeja a užívateľských tlačítok začalo dochádzať k výpadkom vo zvuku.

K týmto výpadkom dochádzalo napriek deklarácii dostatočného výkonu procesora výrobcom dosky. Ideálnym riešením by bol výber výkonnejšieho procesora, ale na inej platforme, pretože firma Microchip výkonnejší typ v rade PIC32MX už neponúka.

Riešením, pri ktorom by sa naďalej využívala navrhnutá platforma a procesor, by bolo taktiež obmedziť počet súčasne zapnutých efektov na počet, pre ktorý je ešte výkon procesora postačujúci a nedochádza k výpadkom vo zvuku.

Možným riešením by bolo taktiež pridanie ďalšieho mikroprocesora, ktorý by obsluhoval samotný displej a tlačítka a komunikoval by cez rýchle SPI rozhranie s procesorom ktorý by spracovával iba zvuk.

Mnou navrhnuté zariadenie predstavuje ekonomickú alternatívu k zariadeniam ktoré momentálne ponúka trh, pretože cena tejto vývojovej dosky na úrovni 80$, je výrazne nižšia aj v porovnaní komerčne ponúkanými lacnejšími efektami.

5.1. Budúcnosť

Ako preukázala implementačná časť a testy s užívateľmi, zariadenie by sa dalo vylepšiť. Výberom inej platformy a mikroprocesoru, alebo pridaním ďalšieho procesora, a ďalej dopracovaním počítačového grafického rozhrania, cez ktoré bude užívateľ komunikovať so zariadením priamo z počítača.

Po mechanickej stránke je potrebné dopracovať vonkajší kryt a poprípade doplniť možnosť pripojenia nožného ovládačas potenciometrom.

Dodatok A.

Pokyny pre kompiláciu

Pre kompiláciu je potrebný oficiálny kompilátor C32 vo verzii 2.0.2 od Microchipu. tento kompilátor je dostupný iba pre operačný systém systém Windows.

Pri kompilácii využijeme program „make“. Vďaka súboru `Makefile-default.mk` (viď. Dodatok C), je možné zadať príkaz
`make -f Makefile-default.mk SUBPROJECTS= .build-conf`,
ktorý nám skompiluje náš projekt do hexa kódu pripraveného na programovanie priamo na mikroprocesor.

Skompilované súbory sa nachádzajú v priečinku `dist`.

Dodatok B.

Post-Test Dotazník

1. Ako hodnotíte testované zariadenie? (Ako v škole 1-5)

2. Bolo zadanie zrozumitelné? (zakružkujte)

   - áno
   - viac áno
   - viac nie
   - nie

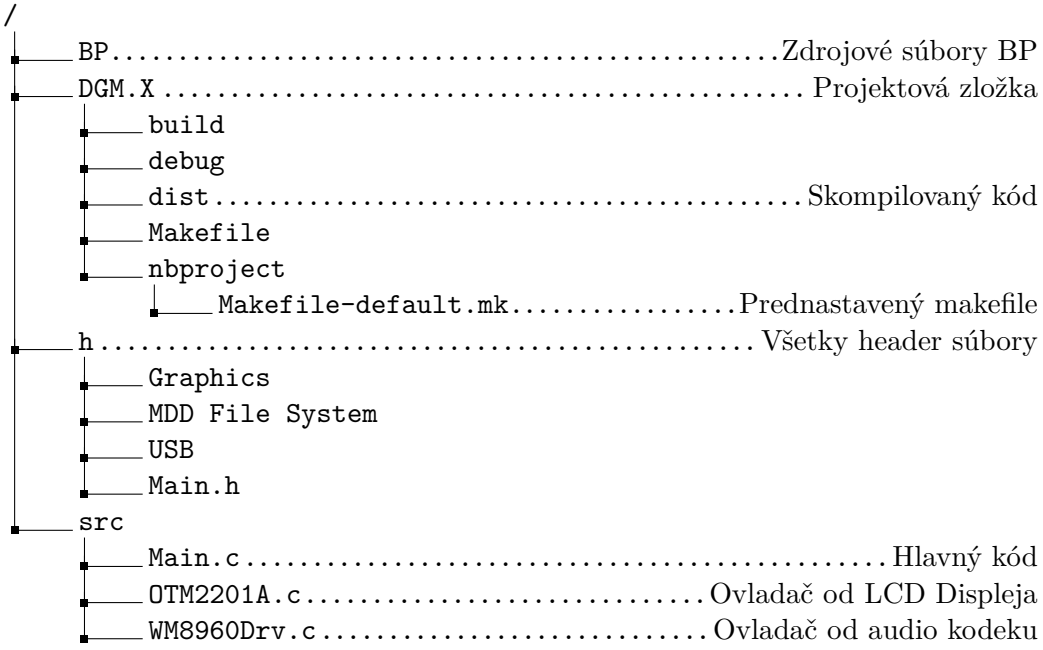
3. S ktorou úlohou ste mali problémy?

4. Používali by ste takéto zariadenie?

Dodatok C.

Obsah priloženého CD disku

Výpis obsahuje len adresáre a súbory, ktoré sú dôležité.



Literatúra

[1] Eric Moyer. *Coyote-1 User's Manual*. 2008. URL: http://www.openstomp.com/download/coyote1_manual_1.1.pdf.

[2] *MOD DUO*. URL: http://portalmod.com/products/duo.

[3] *PedalSHIELD*. URL: http://www.electrosmash.com/pedalshield.

[4] *Bandwidth.svg.png*. URL: http://upload.wikimedia.org/wikipedia/commons/thumb/6/6b/Bandwidth_2.svg/2000px-Bandwidth_2.svg.png.

[5] Udo Zolzer. *DAFX - Digital Audio Effects*. Edition 1. JOHN WILEY and SONS, LTD, 2002. ISBN: 0-471-49078-4.

[6] MikroElektronika. *Digital Filter Design*. URL: http://www.mikroe.com/chapters/view/73/chapter-3-iir-filters/.

[7] Microchip Technology Inc. *Audio Development Board User's Guide*. 2011. URL: http://ww1.microchip.com/downloads/en/DeviceDoc/ADB%20for%20PIC32%20User%20Guide.pdf.

[8] Microchip Technology Inc. *PIC32MX5XX-6XX-7XX*. 2013. URL: http://ww1.microchip.com/downloads/en/DeviceDoc/61156H.pdf.

[9] Microchip Technology Inc. *MPLAB® C Compiler For PIC32 MCUs User's Guide*. 209. URL: http://ww1.microchip.com/downloads/en/DeviceDoc/51686B.pdf.

[10] Microchip Technology Inc. *32-Bit Language Tools Libraries*. 2012. URL: http://ww1.microchip.com/downloads/en/DeviceDoc/51685E.pdf.

[11] *IIR Filter.png*. URL: http://upload.wikimedia.org/wikipedia/commons/d/d2/IIRFilter2.svg.

[12] Gabriel Rivas. *AUTO WAH AUDIO EFFECT (LFO CONTROL)*. 2011. URL: http://www.dsprelated.com/showcode/213.php.

[13] *Analytic.svg.png*. URL: http://upload.wikimedia.org/wikipedia/commons/thumb/d/d7/Analytic.svg/300px-Analytic.svg.png.