

Digitální audio procesor pro potřeby strunných hudebních nástrojů

Lukáš Kozubík

Bakalářská práce
2016



Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
akademický rok: 2015/2016

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení: **Lukáš Kozubík**
Osobní číslo: **A13171**
Studijní program: **B3902 Inženýrská informatika**
Studijní obor: **Informační a řídicí technologie**
Forma studia: **prezenční**

Téma práce: **Digitální audio procesor pro potřeby strunných hudebních nástrojů**
Téma anglicky: **A Digital Signal Processor for Stringed Instruments**

Zásady pro vypracování:

1. Popište základní principy algoritmů pro zpracování audio signálu.
2. Vypracujte rešerši potřebného hardwaru, firmwaru a vývojových prostředků pro vývoj digitálního audioprocessoru.
3. Proveďte návrh elektronického zapojení audioprocessoru.
4. Realizujte navrhované zapojení.
5. Navrhněte a naprogramujte potřebný firmware realizovaného audioprocessoru.
6. Otestujte navržený hardware a prezentujte výsledky.

Rozsah bakalářské práce: -
Rozsah příloh: -
Forma zpracování bakalářské práce: tištěná/elektronická

Seznam odborné literatury:

1. VÁŇA, Vladimír. ARM pro začátečníky. 1. vyd. Praha: BEN – technická literatura, 2009, 195 s. ISBN 978-80-7300-246-6.
2. PUNČOCHÁŘ, Josef. Operační zesilovače v elektronice. 5. vyd. Praha: BEN – technická literatura, 2002, 484 s. ISBN 80-7300-059-8.
3. KERNIGHAN, Brian W a Dennis M RITCHIE. Programovací jazyk C. Vyd. 1. Brno: Computer Press, 2006, 286 s. ISBN 80-251-0897-x.
4. VÍCH, Robert a Zdeněk SMÉKAL. Číslicové filtry. Vyd. 1. Praha: Academia, 2000, 218 s. Česká matice technická (Academia). ISBN 80-200-0761-x.
5. UHLÍŘ, Jan a Pavel SOVKA. Číslicové zpracování signálů. Vyd. 2. přeprac. Praha: Vydavatelství ČVUT, 2002, vii, 327 s. ISBN 80-01-02613-2.

Vedoucí bakalářské práce: Ing. Dalibor Slovák
Ústav počítačových a komunikačních systémů
Datum zadání bakalářské práce: 19. února 2016
Termín odevzdání bakalářské práce: 27. května 2016

Ve Zlíně dne 19. února 2016



doc. Mgr. Milan Adámek, Ph.D.
děkan



prof. Ing. Vladimír Vašek, CSc.
ředitel ústavu

Prohlašuji, že


- beru na vědomí, že odevzdáním diplomové/bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že diplomová/bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk diplomové/bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji diplomovou/bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – diplomovou/bakalářskou práci nebo poskytnout licenci k jejímu využití jen připouští-li tak licenční smlouva uzavřená mezi mnou a Univerzitou Tomáše Bati ve Zlíně s tím, že vyrovnaní případného přiměřeného příspěvku na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše) bude rovněž předmětem této licenční smlouvy;
- beru na vědomí, že pokud bylo k vypracování diplomové/bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky diplomové/bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem diplomové/bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na diplomové/bakalářské práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně, dne

18.5.2016


.....
podpis diplomanta

ABSTRAKT

Práce se zabývá problematikou návrhu vlastní digitální audio efektové jednotky, tzv. multiektu, která zpracovává zvuk z připojeného strunného hudebního nástroje (elektrické kytary) a následně ho vysílá do zesilovače připojeného na výstup této efektové jednotky. Je zde vysvětlení základních principů digitálního zpracování audio signálu spolu s principy a konkrétními algoritmy k získání různých typů audio efektů. Práce také obsahuje elektrický návrh zapojení vstupního kytarového předzesilovače a zobrazovací jednotky složené z osmi 7segmentových displejů s podrobným popisem funkce jednotlivých součástek.

Klíčová slova: audio, efekt, multiekt, DSP, kytarový zesilovač, A/D převodník, D/A převodník, STM32, Cortex-M4

ABSTRACT

This thesis deals with problematics of designing a custom digital audio effect unit, a.k.a multieffect unit, that processes sound from connected stringed musical instrument (electric guitar) and sends it to amplifier that's connected to the output of this unit. There is an explanation of basic principles of digital sound processing along with principles and specific algorithms to achieve various types of sound effects. This thesis also contains electric design of guitar pre-amplifier input and display unit consisting of 8 seven-segment indicators with detailed role description of individual electric parts.

Keywords: audio, effect, multieffect unit, DSP, guitar amplifier, ADC, DAC, STM32, Cortex-M4

Tímto bych rád poděkoval panu Ing. Daliboru Slovákovi za odborné vedení a rady, které mi při vypracování pomohly. Dále děkuji panu Vítězslavu Královi za rady při elektronickém návrhu zesilovače a také děkuji panu Patriku Sáblíkovi za vyvolání a vyleptání desky plošných spojů.

Prohlašuji, že odevzdaná verze bakalářské/diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

OBSAH

ÚVOD.....	9
I TEORETICKÁ ČÁST	10
1 ZPRACOVÁNÍ AUDIO SIGNÁLU	11
1.1 VSTUPNÍ A VÝSTUPNÍ PŘEVOD AUDIO SIGNÁLU	11
1.2 ALGORITMY ZPRACOVÁNÍ AUDIO SIGNÁLU	13
1.2.1 Efekty s využitím zpožďovací linky	14
Delay.....	16
Echo	17
Vibrato	17
Chorus.....	18
Flanger	19
Reverb.....	19
1.2.2 Efekty s využitím úprav amplitudy signálu	20
Tremolo.....	20
Kompresor dynamiky	21
Limiter	22
Noise Gate.....	22
1.2.3 Efekty s využitím úprav frekvence nebo frekvenční odezvy signálu.....	22
Ekvalizér	23
Octaver.....	23
2 VÝVOJOVÉ PROSTŘEDKY	25
2.1 VÝVOJOVÉ PROSTŘEDÍ	27
2.2 TVORBA ELEKTRONICKÝCH SCHÉMAT	28
II PRAKTICKÁ ČÁST	31
3 ELEKTRONICKÉ ZAPOJENÍ.....	32
3.1 BLOKOVÉ SCHÉMA	33
3.2 PŘIPOJENÍ ENKODÉRŮ A TLAČÍTEK	33
3.3 PŘIPOJENÍ DISPLEJE	36
3.4 PŘIPOJENÍ ZVUKOVÉHO VSTUPU	38
3.5 PŘIPOJENÍ ZVUKOVÉHO VÝSTUPU	41
3.6 PŘIPOJENÍ VAROVNÝCH LED	42
4 PROGRAMOVÉ ŘEŠENÍ	44
4.1 OVLADAČ PRO VSTUPNÍ OVLÁDÁNÍ	45
4.2 OVLADAČ DISPLEJE	46
4.3 OVLADAČ VSTUPNÍHO A/D PŘEVODNÍKU	48
4.4 OVLADAČ VÝSTUPNÍHO D/A PŘEVODNÍKU.....	50
4.5 KNIHOVNA MATEMATICKÉHO ZPRACOVÁNÍ ZVUKU	51
4.5.1 Algoritmus funkce Delay	52
4.5.2 Algoritmus funkce Tremolo	53
4.5.3 Algoritmus funkce Flanger	54
4.5.4 Algoritmus funkce Chorus	56
4.5.5 Algoritmus funkce Octaver	57

4.6	HLAVNÍ PROGRAM.....	57
5	DOKONČENÍ A PREZENTACE ZAŘÍZENÍ.....	60
	ZÁVĚR	62
	SEZNAM POUŽITÉ LITERATURY.....	64
	SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK.....	66
	SEZNAM OBRÁZKŮ	69
	SEZNAM PŘÍLOH.....	71
	SEZNAM ELEKTRONICKÝCH PŘÍLOH	72

ÚVOD

V hudebním odvětví se již více než půl století využívá audio efektů, ať už pro přibarvení zvuku či prosté experimentování se zvukem. Audio efekty jsou, vyjma zvukových techniků v nahrávacích studiích, velmi oblíbené u kytaristů, pro které se vyrábí miniaturní analogové či digitální podlahové verze těchto zařízení tzv. kytarové krabičky (stompboxy), které jsou vyrobeny z kovu, popř. z plastu u levnějších produktů. Tyto zařízení jsou většinou jednoúčelové a zprostředkovávají jeden typ audio efektu, lze ovšem narazit i na tzv. multiefekty či multiefektové jednotky, což jsou digitální zařízení, které obsahují více efektů a umožňují je vzájemně kombinovat. V této práci bude vyjmenováno několik základních druhů audio efektů a popsán základní obecný princip jak dosáhnout většiny z nich.

Multiefekty jsou obecně vzhledem ke složitosti zapojení digitálních signálových procesorů poměrně drahé (ve srovnání s jednoúčelovými efekty) a tudíž vyvstává otázka, zda jde takové zařízení vyrobit levněji a pokud možno s více možnostmi. A právě touto záležitostí se budu zabývat v této práci, kde se pokusím navrhnout základní multiefekt, který bude složen z poměrně levných a jednoduchých obvodů a jeho firmware bude navržen způsobem, který umožňuje jednoduché úpravy v chování efektů.

I. TEORETICKÁ ČÁST

1 ZPRACOVÁNÍ AUDIO SIGNÁLU

Pro digitální zpracování analogového zvukového signálu je nejprve nutné takový signál převést na odpovídající digitální zvukový signál a po digitálním zpracování ho převést zpět na analogový signál. K tomu se využívá podpůrných elektronických obvodů.

1.1 Vstupní a výstupní převod audio signálu

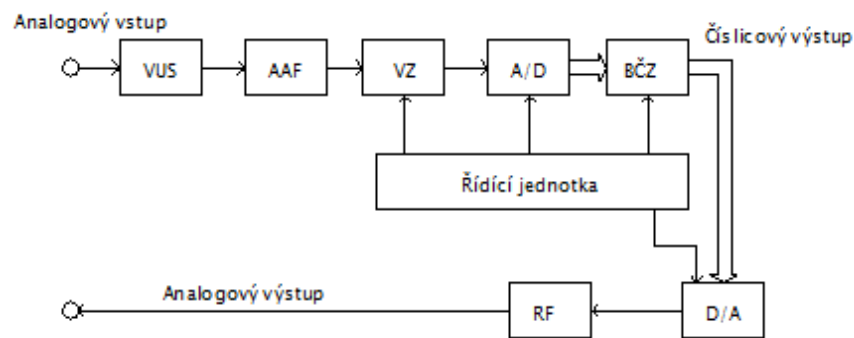
Převod analogového audio signálu na digitální lze rozdělit do několika základních fází, které jsou společné pro jakýkoliv druh A/D převodníku [6]:

1. Vzorkování signálu (Sample & Hold obvod) – odečet vzorků v okamžicích daných vzorkovacími impulsy (obvod drží na svém výstupu poslední zjištěnou napěťovou úroveň ze vstupu až do příchodu dalšího vzorkovacího impulsu)
2. Kvantování vzorků – zaokrouhlení vzorků na nejbližší kvantovací úroveň (výsledek odpovídá jedné z 2^N hodnot rozděleného referenčního napětí, kde N je počet bitů A/D převodníku)
3. Kódování vzorků – převod zjištěné nejbližší hodnoty na číselnou reprezentaci vzorku (např. BCD kód)

Kvantování a kódování většinou probíhá v jediném bloku A/D převodníku.

Po převodu na digitální číslicovou reprezentaci jsou vzorky zpracovány a následně odeslány do D/A převodníku na výstup zařízení. Procesor, který vzorky zpracovává, musí být dostatečně rychlý, aby všechny příslušné operace zpracování byly provedeny v reálném čase mezi vzorkovacími impulsy A/D převodníku.

D/A převodník funguje jednoduše tak, že číselnou reprezentaci vzorku převede zpět na jednu z 2^N hodnot rozděleného referenčního napětí a získanou napěťovou úroveň pošle na analogový výstup zařízení.



Obrázek 1 – Obecné blokové schéma digitálního zpracování [6][7]

Ve schématu (Obrázek 1) lze vidět řadu bloků, které jsou vzájemně propojeny určitými elektronickými technologiemi – jednoduché šipky označují elektrický přenos analogových a hodinových řídicích signálů a složené dvojité šipky označují přenos číslíkových datových signálů.

Vysvětlení zkratk ze schématu:

- VUS – vstupní úprava signálu, provádí se zde úprava vstupní úrovně signálu zesilovačem či děličem napětí, tak aby A/D převodník pracoval s co nejlepším možným napětíovým rozsahem
- AAF – antialiasingový filtr, analogová dolní propust propouštějící složky signálu s frekvencí nižší, než je polovina vzorkovacího kmitočtu A/D převodníku (dle Shannon-Kotělnikova teorému), např. vzorkovací kmitočet 44,1 kHz zajistí s rezervou korektní snímání veškerého pro člověka slyšitelného zvuku do frekvence 20 kHz [6]
- VZ – vzorkovač typu Sample & Hold
- A/D – analogově-číslíkový převodník, zajistí kvantování a kódování signálu
- BČZ – blok číslíkového zpracování, zde se využívá algoritmů pro zpracování diskrétního signálu
- D/A – číslíkově-analogový převodník
- RF – rekonstrukční filtr, odstraňuje z výstupního signálu spektrální složky s frekvencí vyšší než polovina vzorkovací frekvence (harmonické frekvence signálu)

1.2 Algoritmy zpracování audio signálu

Principy, které byly využity při digitálním zpracování, vycházejí převážně z principů minulého století, kde se využívalo především analogových součástek k dosažení žádaných zvukových efektů. Tyto principy se dodnes vyvíjejí, ovšem jejich základ zůstává vesměs stejný.

V této kapitole budou uvedeny základní kategorie efektů často užívaných kytaristy, které budou dále rozčleněny na konkrétní efekty a jejich algoritmy. Zvukových efektů je však mnoho a je také možno je vzájemně kombinovat do složitějších bloků, tudíž zde budou podrobně popsány pouze ty, které se používají v digitálním zpracování nejčastěji a jsou při zpracování relativně nenáročné na systémové prostředky. Některé efekty, i přes to že budou zde zmíněny, je stále vhodnější realizovat v analogovém provedení pro větší dynamičnost a celkově lepší zvukovou kvalitu výsledku. U některých principů bude uveden i reálný příklad konkrétního zařízení z minulosti pro snadnější pochopení zpracování.

Téměř všechny kategorie efektů používají při zpracování mimo jiné tzv. modulační LFO signál (Low-Frequency Oscillation neboli nízkofrekvenční oscilátor), což je element v obvodu, který generuje určitý periodický signál s nastavitelnou amplitudou a frekvencí – tyto parametry jsou obvykle externě nastavitelné uživatelem. Frekvenční rozsah tohoto signálu je obvykle 0 až 20 Hz. Nejčastěji používané průběhy LFO jsou [12]:

- Sinusový – plynulý průběh, který je ideální pro přímou modulaci zvuku
- Trojúhelníkový – průběh s konstantním růstem a poklesem, změna chodu ve špičkách je náhlá
- Logaritmický – plynulý průběh, který ovšem obsahuje strmý skok na konci každé periody

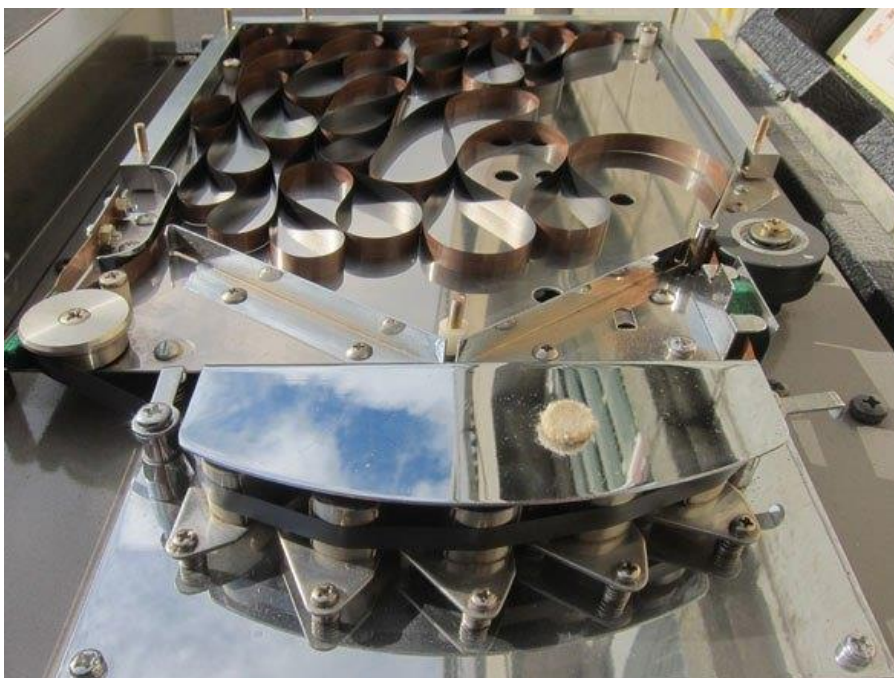


Obrázek 2 – Různé časové průběhy na výstupu LFO [12]

1.2.1 Efekty s využitím zpožďovací linky

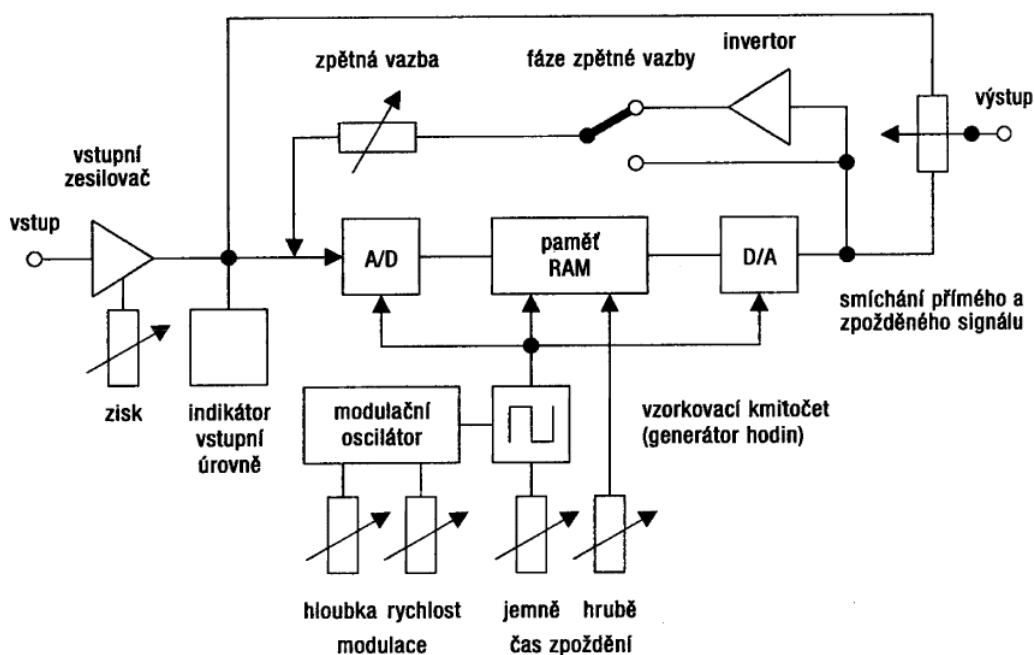
Jedná se o technologii, kdy se využívá paměťového média k nahrávání zvuku ze vstupu. Získanou nahrávku lze poté jednoduše znovupoužít přehráním po určité časové prodlevě pro získání žádaného efektu. Tento jednoduchý princip umožnil v minulosti vznik řadě zvukových efektů, které jsou dodnes velmi oblíbené mezi hudebníky po celém světě.

V minulosti byla jako paměťové médium používána magnetická páska, u které byl pomocí magnetofonových hlavy nahráván a přehráván zvuk. První magnetofonová hlava na pásku nahrávala vstupní zvuk a další hlavy v pořadí jej s určitým zpožděním a hlasitostí přehrály. Zařízení obvykle obsahovalo více přehrávacích hlav pro širší možnosti nastavení efektu a šlo bez problémů dosáhnout i věrohodné simulace přirozené ozvěny ve velké místnosti. Hlavy měly pro zjednodušení mechanické konstrukce zařízení fixní pozici (délka prodlevy signálu tak byla regulovatelná pouze regulací rychlosti průchodu pásky) a jejich vzdálenost vůči sobě byla nepravidelná pro větší variaci zvuku. Hlasitost čtecích hlav byla závislá na použitém módu a spolu s možným zavedením zpětné vazby výstupního zvuku na vstupní nahrávací hlavu tak zařízení nabízelo širokou škálu nastavení. Řada těchto zařízení v sobě také integrovala pružinový reverb často používaný v někdejších kytarových hlavách pro oblíbenost charakteru jeho zvuku. [10][11]



Obrázek 3 – Páskové echo Roland RE-201 Space Echo – vnitřní pohled na magnetofonové hlavy a na odkrytou pásku [10]

S příchodem digitálních technologií se začalo využívat DDL (Digital Delay Line – Digitální Zpožďovací Linka), kde je jako paměťové médium využívána elektronická paměť RAM. Ve srovnání s analogovými páskovými jednotkami zde nejsou žádné pohyblivé mechanické součástky, odpadá omezení fixního umístění hlav (v tomto případě ukazatelů v paměti) a omezení rozsahem rychlosti pásky a také odpadá starost o mechanické opotřebení média. Zmíněné výhody DDL vedou k mobilnějšímu a kompaktnějšímu zařízení a umožňují rovněž spoustu předtím nevídaných technologických možností jako např. nezávislá rychlost a pozice každého z přehrávacích ukazatelů. Maximální časová prodleva signálu je závislá na velikosti paměti, počtu bitů A/D převodníku a jeho vzorkovací frekvenci. Se současnými technologiemi není problém zkonstruovat zpožďovací linku pro téměř jakoukoliv délku (s běžně dostupnými levnými čipy jednotky až desítky vteřin), ale přesto se používají zpožďovací linky o délce max. 1-2 vteřiny, vzhledem k nepoužitelnosti tak dlouhých prodlev v signálu.



Obrázek 4 – Blokové schéma DDL [9]

Do paměti RAM přichází data konstantní rychlostí (vzhledem ke konstantní vzorkovací frekvenci vstupního A/D převodníku) a jsou zapisovány hlavním ukazatelem do paměti s postupně stoupající adresou až ke konci paměti. Na konci paměti RAM se hlavní ukazatel vynuluje a bude zapisovat data od začátku paměti. Hlavní ukazatel tak bude neustále dokoła procházet paměť a pozice čtecího ukazatele se bude odvíjet od momentální pozice hlavního ukazatele - bude o několik stovek až desítek tisíc vzorků zpožděná od hlavního ukaza-

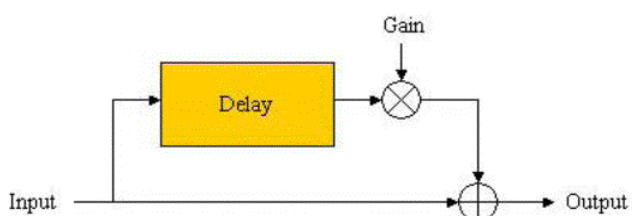
tele v závislosti na žádané prodlevě efektu. Čtecích ukazatelů zde může být libovolný počet a můžou různě měnit rychlost a své parametry nezávisle na ostatních ukazatelích.

Zapojení také obsahuje zpětnou vazbu, která umožňuje přidání určité části zpracovaného signálu k nezpracovanému signálu, který se ukládá do paměti RAM. Tímto lze dosáhnout opakovaného zpoždění, které postupně doznívá v závislosti na hlasitosti zpětné vazby – hlasitost zpětné vazby musí být vždy nižší než 1, jinak by signál ze zpětné vazby narůstal, až by přebudil obvody (clipping). Také je nutné v případě více čtecích ukazatelů zpětnou vazbu patřičně rozdělit či vypustit. [9]

Efekty, které využívají zpožďovací linky:

- Delay
- Echo
- Vibrato
- Chorus
- Flanger
- Reverb

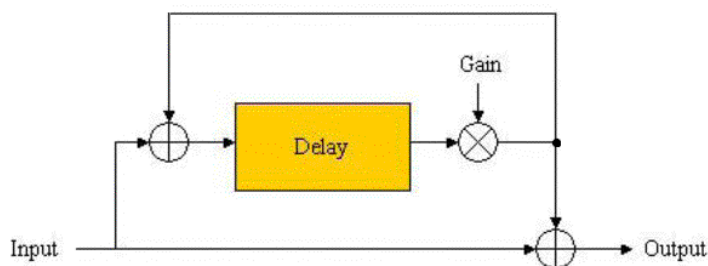
Delay



Obrázek 5 – Blokové schéma efektu Delay [8]

Jde o jednoduchou implementaci zpožďovací linky s jedním čtecím ukazatelem. Z paměti RAM je přehráván zvuk o několik desítek či stovek ms zpožděný od nezpracovaného signálu a je následně s určitou hlasitostí (ve schématu Gain) přičten k nezpracovanému signálu na vstupu. [8]

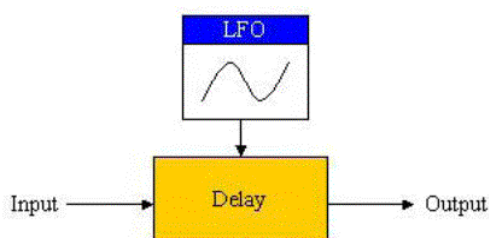
Toto zapojení umožňuje jedno opakování signálu ze vstupu.

Echo

Obrázek 6 – Blokové schéma efektu Echo

Jde o podobný princip fungování jako u efektu Delay, jediný rozdíl je v zavedení zpětné vazby z výstupu efektu zpátky na vstup, který je zapisován do paměti RAM.

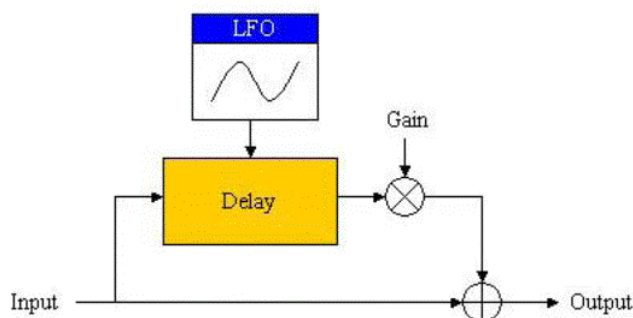
Toto zapojení umožňuje vícero opakování signálu ze vstupu a počet slyšitelných opakování je určen nastavenou hlasitostí zpracovaného zvuku (ve schématu Gain).

Vibrato

Obrázek 7 – Blokové schéma efektu Vibrato [8]

Zvuk se vstupu je v tomto efektu ukládán do paměti RAM a následně je na výstup posílán pouze zpožděný zvuk s proměnlivým zpožděním (obvykle 0 až 10 ms). Okamžitou délku zpoždění určuje proměnlivá amplituda LFO signálu. Výsledným zvukovým efektem je mírné rozlaďování tónů přibližováním a oddalováním signálu od skutečného signálu. Tento efekt je znám jako Dopplerův jev. [8]

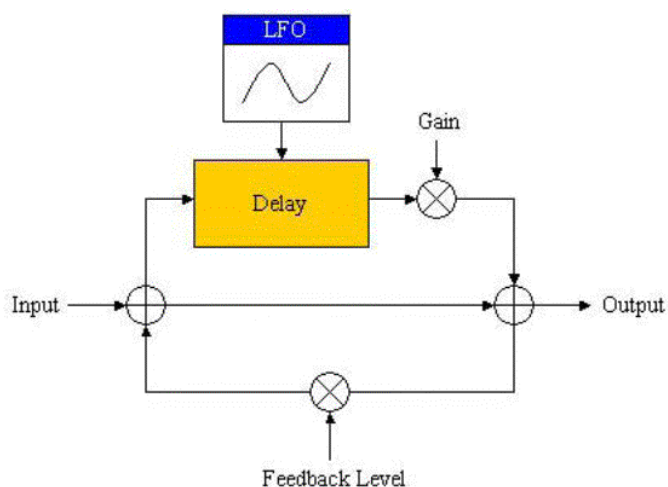
Při prodlužování prodlevy signálu se výška tónů snižuje. Při zkracování prodlevy signálu se naopak výška tónu zvyšuje. Je vhodné nevolit příliš vysokou frekvenci LFO a příliš dlouhé zpoždění signálu, aby nedocházelo k nepřirozeným změnám ladění. [8][9]

Chorus

Obrázek 8 – Blokové schéma efektu Chorus [8]

V zapojení je využíváno efektu Vibrato, který je přičítán k nezpracovanému signálu. Efekt Vibrato v tomto případě využívá delšího zpoždění zvuku, obvykle v rozsahu 20 až 30 ms. Na výstupu efektu je ovšem vhodnější místo přičítání zpracovaného signálu k nezpracovanému použít rozdílové mixování signálů, jelikož k dosažení intenzivnějšího zvukového dojmu je třeba, aby oba signály měly podobnou hlasitost a to je při přičítání signálů téměř nemožné (velké riziko clippingu signálu). Opět je vhodné jako u Vibrata volit nízkou frekvenci LFO, aby se signál příliš nerozladoval a působil tak přirozeněji. [8]

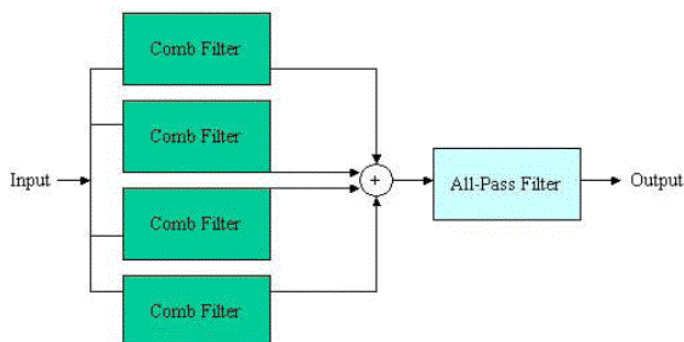
Výsledným zvukovým efektem je iluze, že hraje více hudebních nástrojů stejnou pasáž (noty) - v reálu mají hudební nástroje mírné odchylky v ladění a hráči na tyto nástroje nejsou spolu dokonale sesynchronizováni (nejdou se trefit do stejné doby trvání noty se stejným okamžikem rozeznění tónu), což vede i k vzájemné rozdílné fázi tónů nástrojů. [9][12]

Flanger

Obrázek 9 – Blokové schéma efektu Flanger [8]

V zapojení lze opět nalézt efekt Vibrato, který je zapojen podobně jako u efektu Chorus, ovšem využívá se původního rozsahu časové prodlevy 0 až 10 ms. Na rozdíl od efektu Chorus je v tomto zapojení zavedena kladná zpětná vazba zmixovaného signálu na vstup, což má za následek výrazné barevné změny zvuku. Je možné také použít zápornou zpětnou vazbu pro docílení odlišného charakteru efektu. Opět platí, že pro ideální zvukový dojem je třeba, aby měl zpracovaný signál podobnou hlasitost jako nezpracovaný signál. [9]

Výsledným zvukovým efektem je produkce silné rezonance tónů, která zvýrazní několik frekvenčních pásem ve frekvenční charakteristice zvuku (má hřebenový průběh) – ta se mění interferencí mezi vstupním a zpětnovazebním signálem, a tudíž polohy a velikosti frekvenčních pásem záleží na momentálním zpoždění zpracovaného signálu, resp. na rozdílu fází zpracovaného a nezpracovaného signálu.

Reverb

Obrázek 10 – Blokové schéma efektu Reverb [8]

Toto zapojení je snahou o simulaci reálné ozvěny v nepravidelně tvarované místnosti, která má různě odrazivé povrchy (část zvuku je materiálem na povrchu pohlcena a část je odražena). Jde o paralelní zapojení několika Delay efektů se zpětnou vazbou či bez a s různými frekvenčními odezvami – v simulaci je žádoucí, aby nízké tóny měli jinou rychlost a hlasitost „odrazu“ (opakování) než vysoké tóny. [8]

Tento efekt vyžaduje mnoho výpočtů, a proto je velmi náročný na procesorový čas a velikost pomocné paměti při zpracování v reálném čase (vzhledem k počtu filtrů).

Efekt je využíván především pro dokreslení zvuku dozvukem, pokud zní příliš nepřírodně – pokud je zvuk bez ozvěn, posluchač ho může vnímat jako nepřírodní.

1.2.2 Efekty s využitím úprav amplitudy signálu

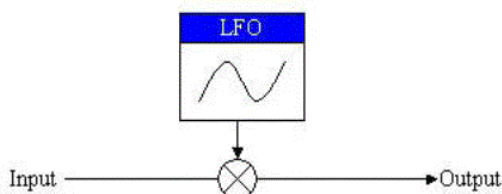
Jedná se o technologii, kdy je upravována amplituda vstupního signálu určitým algoritmem. Toho lze dosáhnout např. změnou zisku zesilovače, kterým prochází vstupní signál, omezením signálu či násobením vstupního signálu určitým modulačním signálem.

Efekty, které využívají amplitudové modulace:

- Tremolo
- Kompresor dynamiky
- Limiter
- Noise Gate

Do této kategorie spadají také kytaristy velmi oblíbené efekty Overdrive a Distortion, které zkreslují signál tím, že ho zesílí a následně omezí jeho maximální úroveň (přebuzení signálu neboli clipping), ovšem na rozdíl od ostatních zmíněných efektů nejsou příliš používané, jelikož při digitálním zpracování nedosahují příliš dobrých zvukových vlastností.

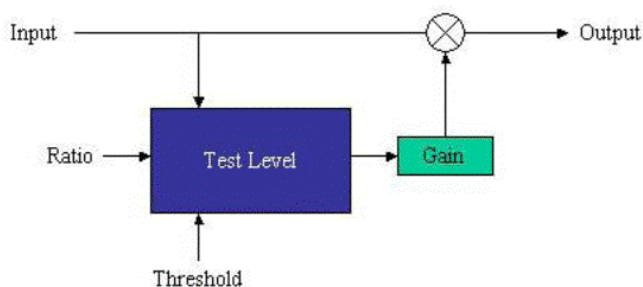
Tremolo



Obrázek 11 – Blokové schéma efektu Tremolo [8]

V tomto zapojení dochází k přímé ovlivňování hlasitosti vstupního signálu signálem LFO. Vstupní signál je násoben momentální hodnotou amplitudy signálu LFO a na výstup je posílán pouze výsledek tohoto násobení. Nastavitelnými parametry jsou frekvence, amplituda a druh průběhu u LFO. V případě že rozsah amplitudy u LFO je 0 až 1, bude výstupní signál kolísat mezi maximální hlasitostí vstupního signálu a tichem. Používané frekvence LFO jsou v řádu jednotek Hz. [8]

Kompresor dynamiky



Obrázek 12 – Blokové schéma Kompresoru dynamiky [8]

Tento efekt je využíván pro snížení dynamiky vstupního signálu – cílem je potlačit hlasitostní rozdíl mezi tichými a hlasitými pasážemi signálu. Úrovně vstupního signálu, které jsou hlasitější než uživatelem nastavený práh hlasitosti (ve schématu Threshold), jsou zeslabeny (ve schématu Gain) podle nastaveného kompresního poměru (ve schématu Ratio) – např. poměr 2:1 značí, že vstupnímu signálu s amplitudou překračující nastavený práh hlasitosti bude zeslabena část, která překračuje tento práh (tj. v případě, že signál překračuje na vstupu práh hlasitosti o 2dB, bude upraven tak, aby na výstupu překračoval práh hlasitosti o 1 dB). Pokud je tedy okamžitá úroveň vstupního signálu menší než práh hlasitosti, je zesílení vstupu nastaveno na 1. Pokud je okamžitá úroveň vstupního signálu vyšší než práh hlasitosti, je zesílení vstupu nastaveno na převrácenou hodnotu nastaveného kompresního poměru. [8]

Vlivem zpracování signálu dochází k opožděné reakci kompresoru v případě překročení úrovně hlasitosti - změna hlasitosti nebude okamžitá, ale zpožděná o dobu reakce kompresoru.

Limitér

Tento efekt má stejnou funkci jako kompresor dynamiky, ovšem má obecně agresivnější průběh komprese signálu. Kompresní poměr je volen od 10:1 výš. [8]

Noise Gate

Tento efekt vychází z kompresoru dynamiky, má ovšem opačnou funkci. Úrovně vstupního signálu, které jsou tišší než uživatelem nastavený práh hlasitosti, jsou zeslabeny podle nastaveného kompresního poměru. Poměr je obvykle volen od 10:1 výš. Pokud je tedy okamžitá úroveň vstupního signálu větší než práh hlasitosti, je zesílení vstupu nastaveno na 1. Pokud je okamžitá úroveň vstupního signálu menší než práh hlasitosti, je zesílení vstupu nastaveno na převrácenou hodnotu nastaveného kompresního poměru. [8]

Výsledným zvukovým efektem je potlačení nechtěných ruchů v pozadí užitečného zvukového signálu (šumu, brumu apod.).

1.2.3 Efekty s využitím úprav frekvence nebo frekvenční odezvy signálu

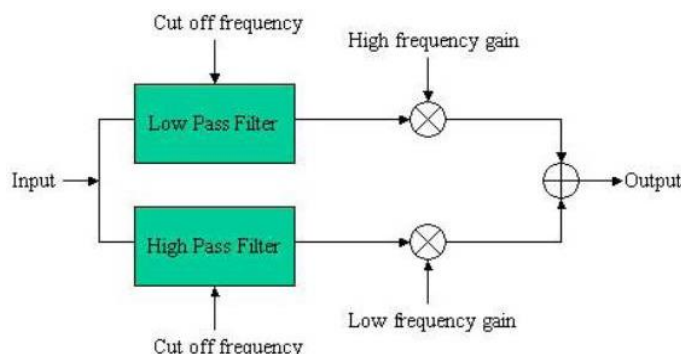
Tyto efekty pracují s frekvencí vstupního signálu a dělí se na ty, které upravují (filtrují) signál podle frekvence a na ty, které přímo mění frekvenci signálu podle určitých parametrů.

Efekty s úpravou frekvenční odezvy signálu:

- Ekvalizér
- Wah-Wah
- Envelope Filter

Efekty se změnou frekvence signálu:

- Harmonizer / Pitch Shifter
- Octaver

Ekvalizér

Obrázek 13 – Blokové schéma Ekvalizéru [8]

Ekvalizér je zvukový efekt, který využívá úpravy hlasitostí jednotlivých frekvenčních pásem vstupního zvukového signálu pro úpravu jeho celkové frekvenční charakteristiky. K získání odděleného frekvenčního pásma zvuku se využívá dolních, horních či pásmových propustí.

V blokovém schématu je zapojení jednoduchého dvouparametrového ekvalizéru pro regulaci výšek (hornopropustní filtr) a basů (dolnopropustní filtr) vstupního signálu. Mezní frekvenci obou filtrů lze měnit a stejně tak i zisk/útlum jednotlivých pásem. [8]

V případě víceparametrového ekvalizéru jsou zpravidla mezní frekvence fixně určeny tak, aby ekvalizér pokrýval celé frekvenční spektrum, které je relevantní pro vstupní signál.

Výsledným zvukovým efektem může být kompletní změna barvy zvuku hudebního nástroje při zvýraznění či potlačení určitých frekvenčních pásem a také lze ekvalizérem potlačit určité rezonanční frekvence prostoru, pokud při reprodukci působí ve zvuku rušivě.

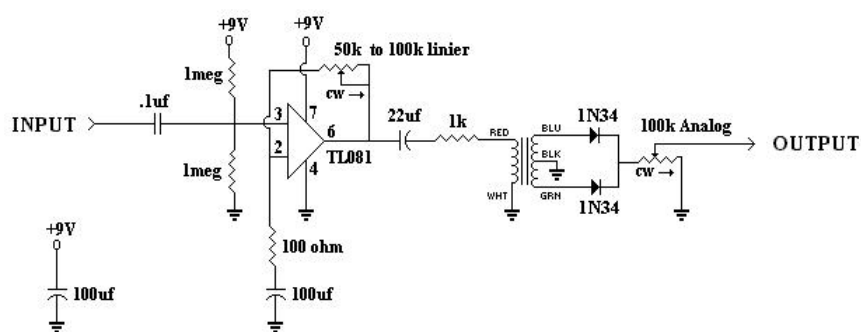
Octaver

Jedná se o efekt, kdy se se vstupním signálem smíchá signál, který má dvojnásobnou či poloviční frekvenci (na rozdíl od efektu Harmonizer, kdy se přimíchává signál zvýšený či snížený o několik tónových intervalů). Tyto signály lze získat několika způsoby.

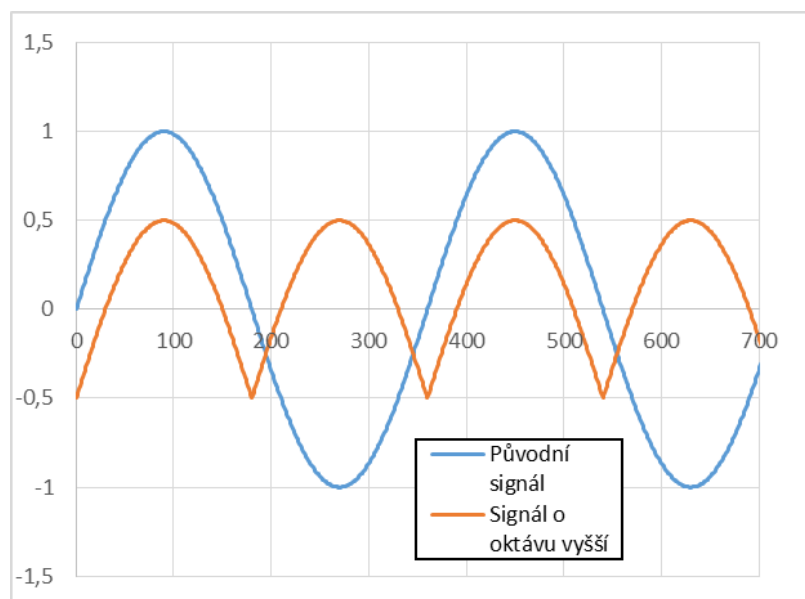
Signál s frekvencí o oktávu nižší lze získat převedením signálu na frekvenční obdélníkový signál a následně využít frekvenční děličky sestavené z klopných obvodů pro vydělení frekvence dvěma pro získání signálu s frekvencí o oktávu nižší.

Signál s frekvencí o jednu oktávu vyšší lze získat např. použitím techniky usměrnění signálu. Vychází se z předpokladu, že zvukový signál kmitá kolem určitého nulového bodu s

jistou periodou. Pokud tento signál usměrníme (získáme absolutní hodnotu signálu vzhledem k nulovému bodu), dojde ke zkrácení této periody na polovinu, čímž se teoreticky zvedne frekvence signálu 2x. Takto získaný signál je poté smíchán s původním signálem pro dosažení cíleného zvukového efektu – pro zlepšení výsledného zvuku efektu se může zpracovaný signál vůči vstupnímu posunout do záporných či kladných hodnot. [13]



Obrázek 14 – Schéma elektronického efektu Octaver Up. Diody 1N34 na výstupu usměrňují zvukový signál pro zvýšení frekvence signálu. [13]



Obrázek 15 – Ukázka zdvojnásobení frekvence vstupního signálu usměrněním

2 VÝVOJOVÉ PROSTŘEDKY

Pro vývoj DSP je nutné, aby měl mikropočítač velký výpočetní výkon s hardwarovou podporou 32bitových instrukcí a koprocесorem FPU pro hardwarové výpočty s čísly s plovoucí desetinnou čárkou s jednoduchou přesností (32bit float). Těmto požadavkům dokonale vyhovují architektury mikrořadičů s jádrem ARM Cortex-M4 a Cortex-M7, ovšem Cortex-M4 výkonem dostačuje a vzhledem k nízké ceně je tak lepší volbou pro tyto účely. [3]

Mikrokontroléry s jádrem Cortex-M4 jsou postaveny na Harvardské architektuře, což znamená, že datová paměť je fyzicky oddělena od paměti pro program. Toto zapojení je výhodné, jelikož paměti nemusí být identické v žádném směru – můžou být postaveny jinou technologií, mít jinou rychlost i jiný počet bitů na slovo. Oddělení pamětí také umožňuje ke každé z nich přistupovat zvlášť (paralelně), což značně urychlí zpracování. [3]

Pro tento projekt byla vybrána vývojová deska STM32F4DISCOVERY od společnosti STMicroelectronics, která se skládá z mikrokontroléru STM32F407VGT6, miniUSB propojení s procesorem přes rozhraní ST-LINK/V2 (ladění, virtuální COM port a USB Mass Storage), tříosý akcelerometr, digitální mikrofon, externí zvukový D/A převodník CS43L22 se zesilovačem třídy D a mikroUSB konektor pro připojení USB OTG zařízení (flash disky, klávesnice, myši apod.).[4]



Obrázek 16 – Vývojový kit STM32F4DISCOVERY [4]

Mikrokontrolér STM32F407VGT6 je postaven na 32bitovém RISC jádru ARM Cortex-M4 a dokáže pracovat při taktovací frekvenci až 180 MHz. V mikrokontroléru lze také nalézt koprocessor FPU pro hardwarové počty s čísly s jednoduchou přesností dle standardu ARM a také je implementována sada DSP instrukcí spolu s jednotkou pro ochranu paměti (MPU). Procesory z této řady obsahují 1 MB flash paměti pro uložení programu a 192 kB statické RAM pro ukládání pracovních dat. Pro audio účely lze využít kterýkoliv ze tří dostupných 12bitových A/D převodníků a dvou 12bitových D/A převodníků či propojit externí převodník využitím až 3 integrovaných sběrnic I²S (Inter-IC Sound), což jsou upravené sběrnice SPI pro přenos zvukových dat.[18]

2.1 Vývojové prostředí

Pro programování procesoru STM32F407VGT6 lze využít několik technologií, ovšem nejjednodušší z nich je propojit vývojový kit pomocí miniUSB konektoru s PC a využít integrovaného rozhraní ST-LINK/V2.

Pro jednoduché programování mikrokontroléru je možno využít utilitu přímo od výrobce kitu STM32 ST-LINK Utility, která umožňuje mazání, čtení a zápis do flash paměti mikrokontroléru a také mimo jiné umožňuje updatovat firmware samotného rozhraní ST-LINK/V2 pokud je to třeba. Do flash paměti lze zapisovat pouze binární soubory (elf, hex, apod.), které jsou výstupem ARM GCC kompilátoru.[4]

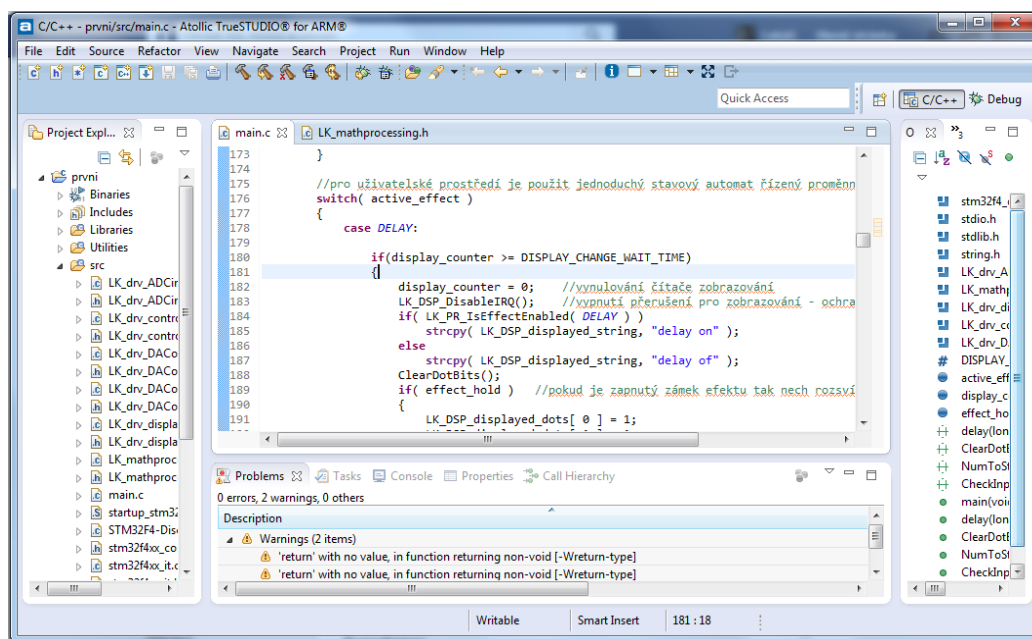
Pro rozsáhlý projekt je vhodné sáhnout po některém vývojovém prostředí, ve kterém již jsou veškeré nutné věci k programování tohoto mikrokontroléru (editor, kompilátor ARM GCC s podporou STM32F407xx, debugger s podporou rozhraní ST-LINK/V2) integrovány. Pro tyto mikrokontroléry je vytvořena spousta komerčních i nezávislých IDE (Coocox, KEIL, mbed, Mikroelectronika...), ovšem mnoho z nich obsahuje i pro studentské účely omezení délky zdrojového kódu pouze na několik desítek kB či mají nestabilní ladící ovladač, který je činí nepoužitelnými. Nakonec bylo využito produktu švédské firmy Atollic – bezplatného IDE TrueSTUDIO, které vychází z oblíbeného vývojového prostředí Eclipse.

Vývojové prostředí TrueSTUDIO nabízí pro bezplatné užití několik služeb [5]:

- IDE zaměřené čistě na vývoj zařízení s ARM
- Neomezená velikost zdrojového kódu
- Využití open-source řešení GCC, GDB a Eclipse IDE, možnost implementace vlastních pluginů
- Široká škála nastavení kompilátoru kódu
- Jedno či více jádrový debugger
- Pohodlná manipulace s projekty
- Intuitivní editor umožňující pružnou navigaci v kódu, vizualizaci apod.
- Udržování aktualizacemi IDE

Vývojové prostředí TrueSTUDIO lze využívat na všech operačních systémech Microsoft Windows od verze Vista (podporována x86 i x64 verze Windows ačkoliv program samot-

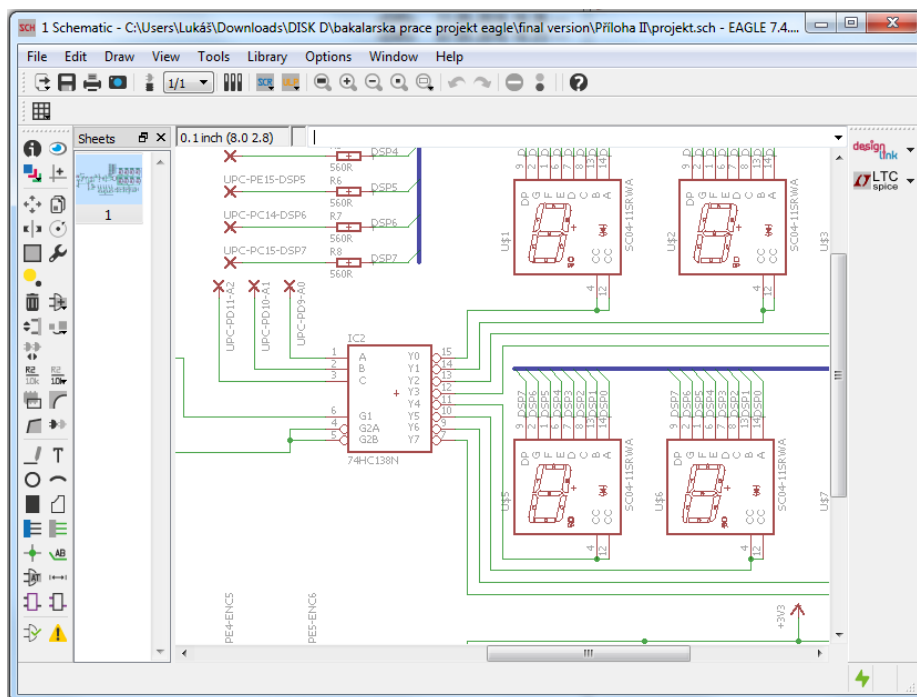
ný je dostupný pouze jako x86). Podporu operačních systému na bázi Linux a Mac OS X plánují vývojáři z Atollic přidat v druhé polovině tohoto roku.[5]



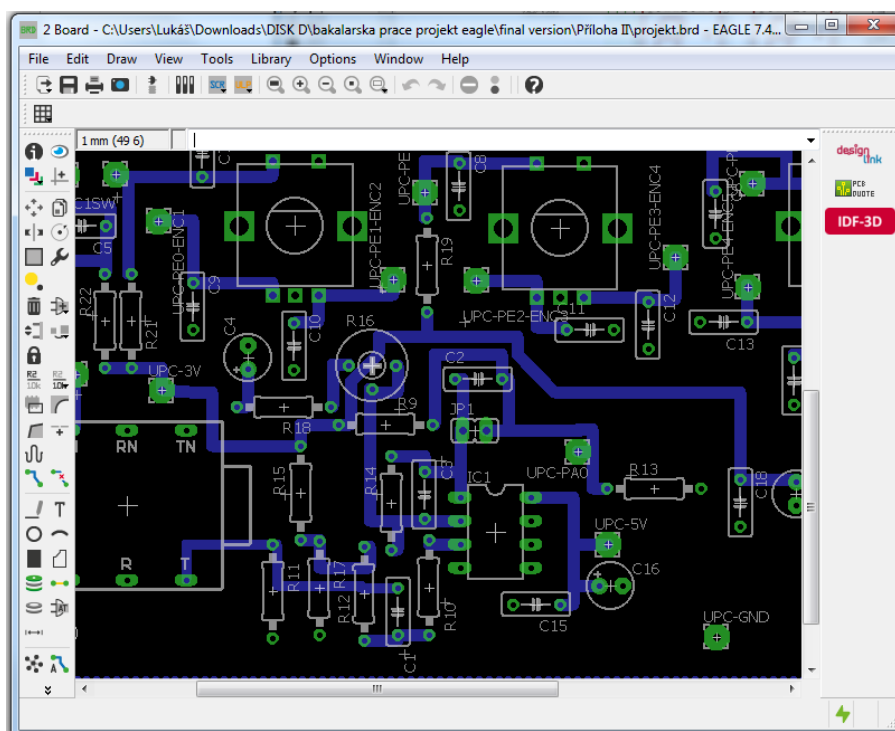
Obrázek 17 – Ukázka vývojového prostředí TrueSTUDIO

2.2 Tvorba elektronických schémat

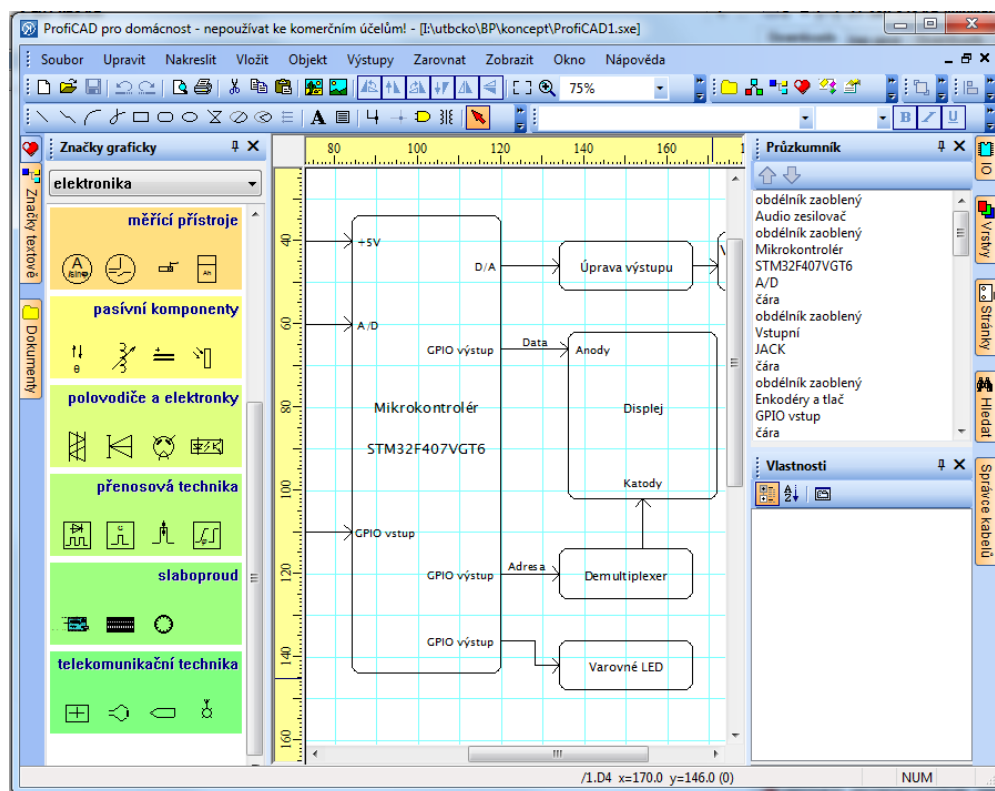
Pro návrh elektronického schématu a desky plošných spojů byl použit software EAGLE 7.4.0 Layout Editor a pro návrh hlavního blokového schématu software ProfiCAD.



Obrázek 18 – Ukázka návrhu elektronického schématu v softwaru EAGLE



Obrázek 19 – Ukázka návrhu DPS v softwaru EAGLE



Obrázek 20 – Ukázka kreslení blokového schématu v softwaru ProfiCAD

II. PRAKTICKÁ ČÁST

3 ELEKTRONICKÉ ZAPOJENÍ

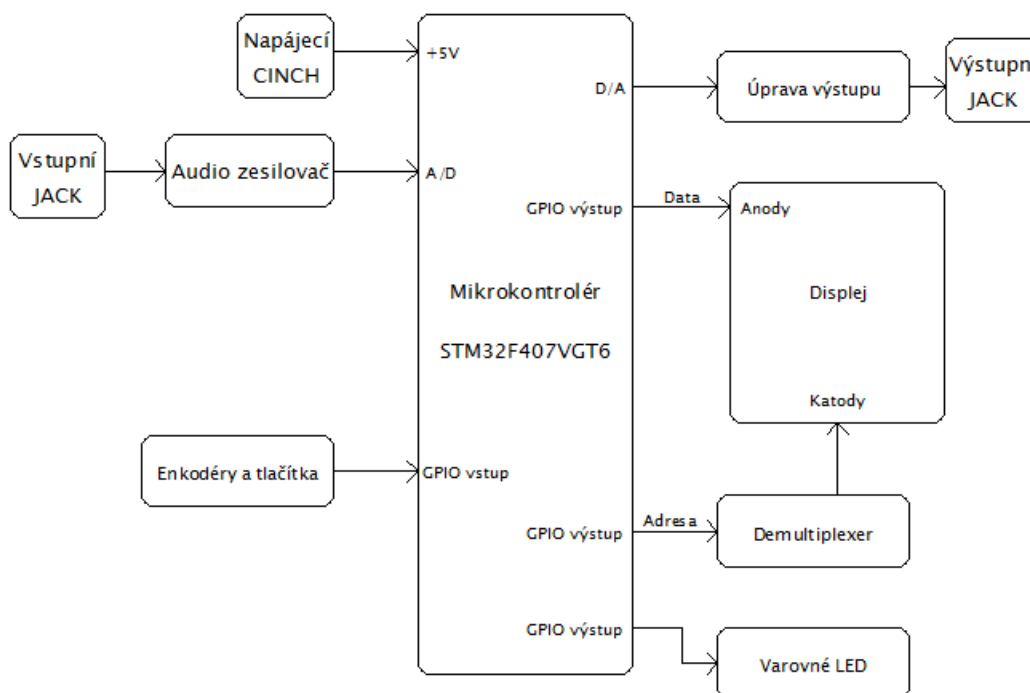
Pro funkci audio procesoru byly navrženy jednoduché podpůrné elektronické obvody pro zajištění propojení vnějších zařízení s vývojovým kitem. Vzhledem k elektronickým parametrům vstupního A/D převodníku bylo nutné navrhnout audio zesilovač pro dosažení ideálního snímání zvuku strunného nástroje. Taktéž výstup D/A převodníku byl upraven, aby vyhovoval obecným elektronickým parametrům vstupů zesilovačů strunných nástrojů, ke kterým bude výsledný audio procesor připojen.

Pro uživatelské rozhraní bylo navrženo ovládání rotačními inkrementálními enkodéry a masivním nožním tlačítkem pro ovládání audio procesoru ve stoje. Odezva uživateli je zprostředkována osmi 7segmentovými displeji, které jsou navrženy, aby zobrazovaly všechna čísla a většinu standardních znaků abecedy.

Vzhledem k technologickým možnostem výroby bude výsledná deska plošných spojů jednostranná, což především znamená, že propojení této desky s vývojovým kitem bude pouze pomocí drátových propojek přímo na jednotlivé vývodky místo využití dvou 50pinových konektorů, jelikož takové zapojení by bylo při použití jednostranné desky velmi náročné na výrobu a nepříliš praktické.

Hlavní elektronické schéma a návrh desky plošných spojů lze nalézt v elektronické příloze, viz Příloha II.

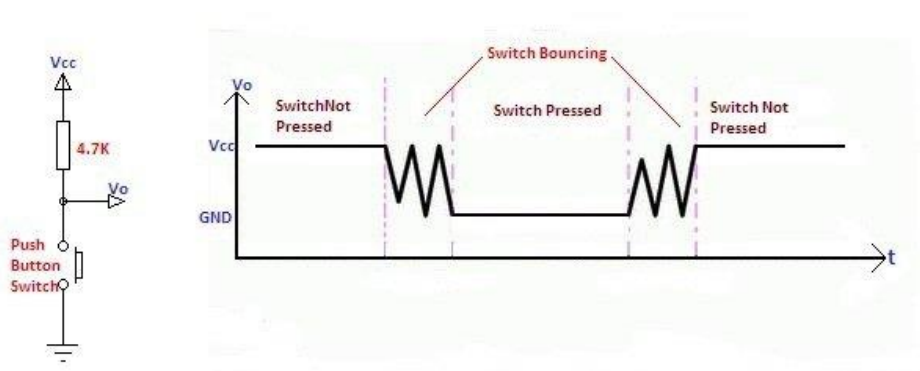
3.1 Blokové schéma



Obrázek 21 – Blokové schéma navrhovaného hardwaru

3.2 Připojení enkodérů a tlačítek

Při použití jakýchkoliv mechanických spínačů a přepínačů v embedded aplikacích je třeba myslet na nedokonalost tohoto typu spínání, konkrétně na zákmity těchto kontaktů. Tento efekt způsobí, že než bude kontakt tlačítka trvale spojen (při stisku tlačítka), dojde k několika nežádoucím spojením a odpojení tohoto kontaktu. Stejný efekt nastane také při rozpojování kontaktu (při uvolnění tlačítka). Tento jev trvá běžně několik jednotek či desítek ms.



Obrázek 22 – Zákmity obecného mechanického spínače při pull-up zapojení [19]

Vzhledem k velmi vysokým rychlostem mikrokontroléru je tento jev nezanedbatelný, jelikož vstupní obvody portů procesoru tyto zákmity zachytí a bez patřičného ošetření je vyhodnotí jako mnohonásobné stisknutí tlačítka, což by se negativně projevilo na nepřívětivém ovládání v uživatelském prostředí. [1][19]

Je mnoho způsobů jak tento jev potlačit [19]:

- využití bistabilního klopného obvodu v případě přepínače
- využití R-C článku a schmittova invertoru
- využití softwarové logiky
- využití specializovaného integrovaného obvodu (zřídka používané)

V navrhovaném hardwaru je dbáno především na jednoduchost zapojení, proto bude použito zapojení s R-C článkem bez schmittova invertoru. Toto zapojení funguje následovně (za ideálních podmínek):

- 1) V základním stavu je kontakt rozpojený a na kondenzátoru je napětí přibližně 3,3 V (velikost napájecího napětí), což mikrokontrolér vyhodnotí jako log. H
- 2) Při stisku tlačítka dojde na spínači k zákmitu, ale v prvním momentu, kdy bude kontakt spojen, se přes něj vybije kondenzátor a bude na něm napětí menší než $0,3V_{DD}$ (0,99 V) [18], což mikrokontrolér vyhodnotí jako log. L
- 3) Při chvilkovém rozpojení kontaktu se začne kondenzátor nabíjet přes odpor určitou rychlostí, ovšem než stihne napětí na kondenzátoru přesáhnout napětí $0,3V_{DD}$ (0,99 V) dojde v důsledku zákmitu tlačítka k dalšímu chvilkovému spojení kontaktu, čímž se opět kondenzátor vybije a mikrokontrolér tak i přes zákmit tlačítka stále vyhodnocuje vstup jako log. L
- 4) Po úspěšném spojení kontaktu tlačítka je na vstupu mikrokontroléru nulové napětí, a tudíž je vstup vyhodnocen jako log. L
- 5) Při uvolnění tlačítka dojde opět k zákmitu a opakuje se chování popsané v bodech 2 a 3
- 6) Po úspěšném rozpojení tlačítka se kondenzátor nabije přes odpor na velikost napájecího napětí a toto napětí je vyhodnoceno mikrokontrolérem jako log. H

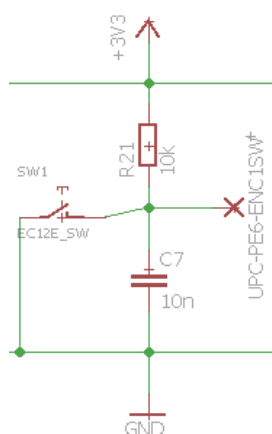
Dané chování závisí především na časové konstantě R-C článku (pro jednoduchost bude hodnota odporu konstantní, pak záleží pouze na velikosti kondenzátoru). Pokud se zvolí

příliš malá kapacita kondenzátoru, nebudou zákmity v případě jejich delších period (vyšší frekvence kmitání) odfiltrovány. Pokud se zvolí příliš velká kapacita kondenzátoru, bude po puštění tlačítka potřeba delší doba k nabití kondenzátoru - tlačítko bude delší dobu v log. L i přesto, že bude kontakt dávno rozpojen, což by vadilo při rychlém mačkání tlačítka, jelikož by mikrokontrolér vůbec nezaznamenal více stisknutí. Také hrozí opotřebení kontaktů (opalování jiskřením) tlačítka při vybíjení takového kondenzátoru, jelikož by se zkratoval daleko větší proud kvůli většímu množství uskladněné energie v kondenzátoru.

V použitém způsobu ovládání jsou použity celkem 4 tlačítka: 3 tlačítka enkodérů a 1 nožní tlačítko pro zapínání/vypínání efektů. V případě tlačítek enkodérů se jedná o zákmity o délce přibližně 2-3 ms a v případě masivního nožního tlačítka jde až o 10x delší dobu zá- kmitávání. [17]

Všechna tato tlačítka jsou ve schématu zapojeny stejným způsobem – využívá se R-C článků k částečnému potlačení zákmitů tlačítek a k bitům portu mikrokontroléru jsou připojeny středy článku (mezi napájecí sběrnicí a zemí).

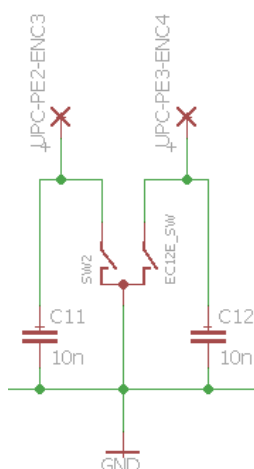
Hodnoty odporu i kondenzátoru v R-C článku byly zvoleny empiricky. Odpor je zvolen o velikosti 10 k Ω , aby nebyl příliš namáhán stabilizovaný zdroj 3,3 V na vývojové desce. Kapacita kondenzátorů je zvolena dostatečně nízká (10 nF), aby se neopotřebovávaly kontakty spínačů jiskřením při jejich zkratování. Tímto zapojením je potlačena většina vysoko- frekvenčních zákmitů tlačítek. Nízkofrekvenční zákmitů tlačítek co zapojením potlačeny nejsou, jsou potlačeny softwarově.



Obrázek 23 – Ukázka zapojení spínače u prvního enkodéru

U otočných mechanických spínačů enkodéru dochází ke stejnému jevu – zákmitům jako u běžných mechanických spínačů. Jediný rozdíl v zapojení těchto kontaktů je v pull-up rezis-

toru, který je v tomto případě nahrazen vnitřním pull-up rezistorem mikrokontroléru a má hodnotu odporu přibližně 40 k Ω . [17][18]



Obrázek 24 – Ukázka zapojení kontaktů u druhého enkodéru

3.3 Připojení displeje

Displej je tvořen z osmi sedmisegmentových displejů s červenými LED segmenty, které jsou časově multiplexovány pomocí mikrokontroléru. Toto řízení je vybráno za účelem zjednodušení připojení displejů k mikrokontroléru, především k ušetření počtu použitých pinů mikrokontroléru.

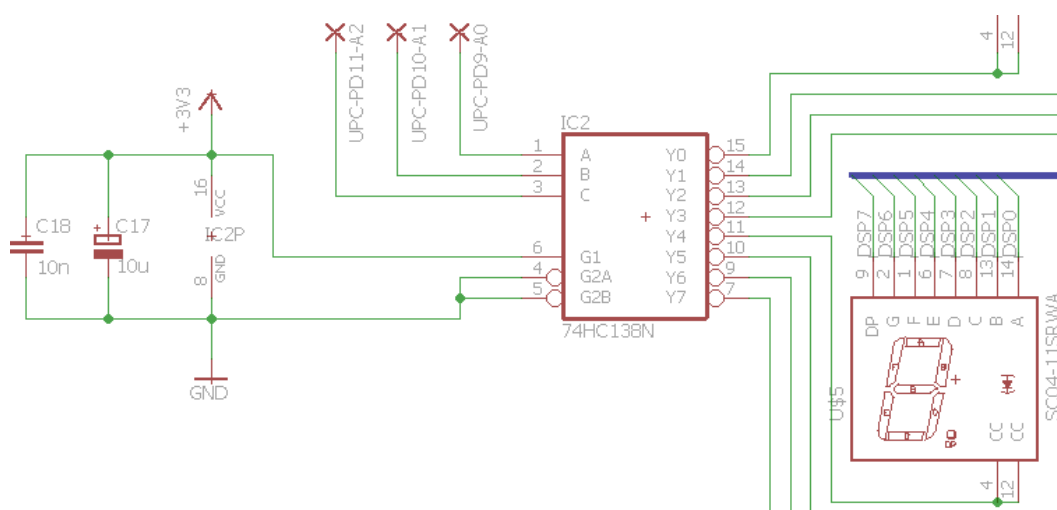
Jedná se o řízení, kdy je nejprve aktivován jeden z osmi dostupných displejů a poté je na datovou sběrnici, kterou všechny displeje sdílejí, odeslána informace, jaké segmenty se mají na displeji rozsvítit a které mají zůstat zhasnuté. Toto je provedeno 8x po sobě, aby došla řada na každý displej. Dostatečně rychlým periodickým přepínáním displejů je poté docíleno efektu, kdy lidské oko díky své setrvačnosti nezaznamenává blikání, ale naopak se mu jeví, že všechny displeje svítí najednou.

Pro ušetření počtu použitých pinů na mikrokontroléru je zakomponován do zapojení pomocný demultiplexer 74HC138N. Tento demultiplexer má negované výstupy (pokud je vybraný výstup aktivní, je v log. L) a proto je nutné použít sedmisegmentové displeje se společnou katodou (pokud by bylo nutné použít displeje se společnou anodou, bylo by nutné ještě výstupy zinvertovat pomocným obvodem, popř. použít jiný demultiplexer). [16]

Použitý demultiplexer je zapojen dle pravdivostní tabulky a dle technických vlastností zjištěných z datasheetu IO [16]:

- vstup G1 je připojen na napájecí napětí (log. H), aby byl obvod aktivní

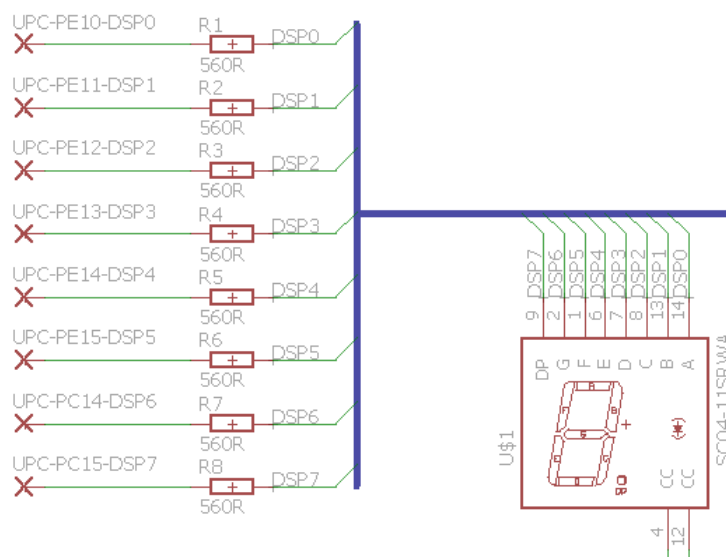
- vstupy G2A a G2B jsou připojeny na zem (log. L), aby byl obvod aktivní
- napájení obvodu je 3,3 V (místo 5 V), aby byly vstupy korektně nastaveny na napěťové úrovni CMOS logiky mikrokontroléru, a také je preventivně jištěno dvěma kondenzátory fyzicky umístěnými co nejblíže u IO, aby nedocházelo k ovlivňování ostatních komponent (především zesilovače audio signálu) při přepínání výstupů
- adresní vstupy A, B a C jsou připojeny přímo na piny mikrokontroléru (určené 3 adresní bity)
- výstupy Y0 až Y7 jsou připojeny přímo na katody displejů U\$1 až U\$8



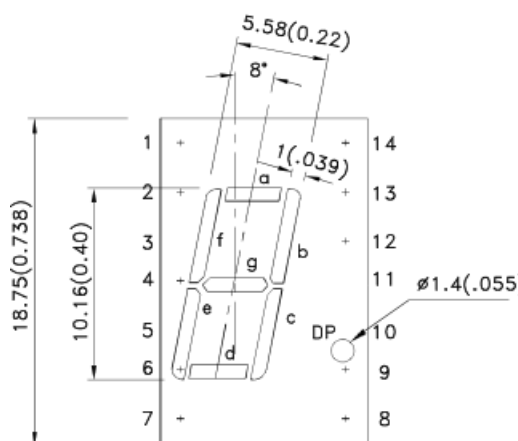
Obrázek 25 – Ukázka zapojení demultiplexeru – všechny vstupy a výstup pro pátý displej

Jednotlivé anody (segmenty) všech displejů jsou spojeny paralelně a přes rezistory o velikosti $560\ \Omega$ (což poskytuje proud přibližně 2,7 mA na jeden segment) jsou připojeny přímo na piny mikrokontroléru, tak aby bylo možné ovládat všechny displeje společnými 8 datovými bity. Intenzitu osvětlení segmentů lze zvyšovat proudem (zmenšením odporu budících rezistorů) až do 20 mA, ale je třeba při takové úpravě vzít v potaz maximální povolený proud piny mikrokontroléru (8 až 20 mA podle možností chlazení pouzdra) a maximální celkový proud napájecích sběrnic mikrokontroléru (240 mA). Intenzitu lze také v případě nutnosti snižovat programově PWM buzením anod. [15][18]

Ve schématu je toto připojení realizováno přes sběrnici pro lepší čitelnost a bity (segmenty) jsou pojmenovány stylem DSP0 až DSP7, aby korespondovaly se značením v kódu programu.



Obrázek 26 – Ukázka zapojení anod prvního displeje k pinům mikrokontroléru



Obrázek 27 – Fyzické rozměry a rozmístění jednotlivých segmentů u použitého typu 7segmentového displeje [15]

3.4 Připojení zvukového vstupu

Jako zesilovač je použit jeden ze dvou OZ z čipu LM358N. Napájení tohoto zesilovače je nesymetrické a je vyvedeno přímo z vývojové desky (+5 V a 0 V). Toto napájení je také jištěno dvěma kondenzátory C15 a C16, které jsou na DPS umístěny co nejbliž u IO pro odfiltrování nežádoucích výkyvů v napájení.[2]

Vstupní zesilovač pro strunný nástroj je řešen jako standardní neinvertující zapojení.

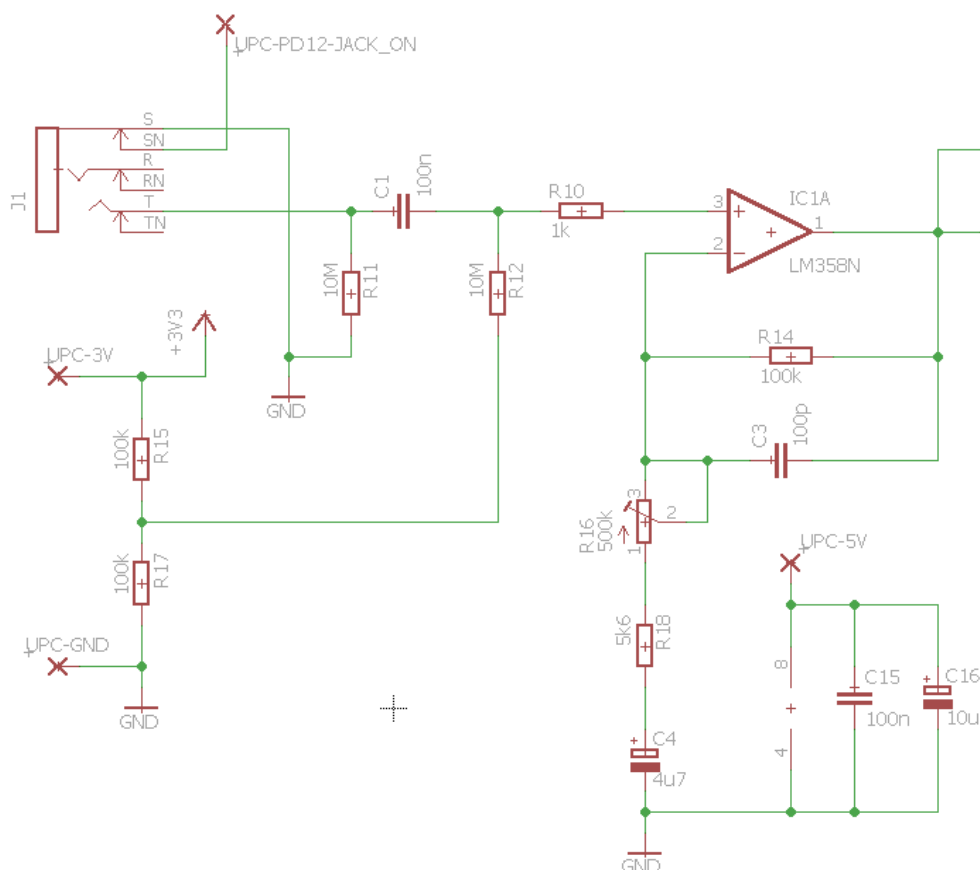
Na vstupu je plastový 6,3mm stereo JACK, který má 3 signálové a 3 pomocné kontakty, které jsou spojené se signálovými a v případě vsunutí konektoru se rozpojí. Jeden z těchto kontaktů je přímo připojen k portu mikrokontroléru a je označen jako JACK_ON, aby korespondoval s kódem programu. Tento kontakt bude buď rozpojený, nebo spojený přímo se zemí, proto je nutné pro správnou funkci zapnout na pinu v mikrokontroléru vnitřní pull-up rezistor.

Jelikož je zesilovač napájen nesymetrickým zdrojem tak je nutné vytvořit umělou zem +1,65 V (polovina maximálního vstupního napětí A/D převodníku), kolem které bude kmitat užitečný signál. Tohoto je dosaženo odporovým napěťovým děličem 1:1 tvořeného rezistory R15 a R17. Tyto rezistory mají dostatečně malou hodnotu (100 k Ω), aby umělá zem nebyla ovlivňována šumem.[2]

Signál ze špičky konektoru vstupuje do vstupní části zesilovače. Nejprve je rezistorem R11, který je vstupní impedancí, upraven vstup – odstranění nežádoucího šumu. Vstupní impedance je dostatečně vysoká, jelikož snímače strunných nástrojů mají v pasivním provedení velmi slabý výstup (výstupní impedance běžně okolo 1 M Ω). Signál dále prochází kondenzátorem C1, pomocí kterého se užitečný signál ze strunného nástroje superponuje na předem zmíněnou umělou zem a přes odpor R10 vstupuje do neinvertujícího vstupu OZ. Jelikož signál ze snímače má příliš malou sílu, je nutné mezi umělou zem a užitečný signál ještě zařadit odpor R12 s dostatečně velkým odporem (10 M Ω), aby nebyl signál amplitudově deformován při snaze se superponovat na tvrdý odporový dělič.

Ve zpětné vazbě nalezneme několik zapojení:

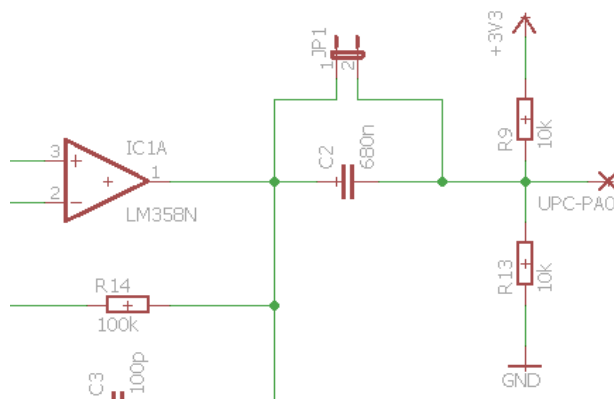
- hodnoty R14 a R16 + R18 určují velikost zesílení. R16 je nastavitelný odpor, kterým se určí žádané zesílení. Zapojení umožňuje nastavení zesílení v rozsahu přibližně 1,2 až 18,9 (1,5 až 25,5 dB).
- odpor R14 s kapacitou C3 tvoří dolnoproustní filtr 1. řádu s mezní frekvencí přibližně 15,9 kHz
- kondenzátor C4 slouží k potlačení napěťové nesymetrie OZ pro procházející střídavé signály a zároveň se jedná o stejnosměrné oddělení invertujícího vstupu, které je nutné pro korektní funkci OZ s nesymetrickým napájením



Obrázek 28 – Zapojení vstupní, napájecí a zpětnovazební části vstupního zesilovače

Vzhledem k nedokonalosti použitého zapojení je přímo na výstupu OZ lehce posunutá umělá zem (naměřeno +1,8 V), což by mohlo mít nežádoucí vliv na některé algoritmy zpracování, pracující s amplitudou vztaženou právě k umělé zemi +1,65 V. Z tohoto důvodu je výstup ještě doplněn o jeden odporový napěťový dělič 1:1 tvořený rezistory R9 a R13 a na vytvořený napěťový potenciál se superponuje signál z výstupu zesilovače pomocí kondenzátoru C2 a ten je přímo přiveden na pin mikrokontroléru, který je propojený s žádaným A/D převodníkem (v tomto případě PA0).

Tento způsob snímání výstupu lze obejít zkratováním kondenzátoru C2 mechanickou spojkou vodičů JP1 – v takovém případě by se snímal přímo výstup zesilovače. Tato spojka nemá pro funkci nadále význam, byla využita především v rané fázi vývoje k usnadnění ladění zesilovače a A/D převodníku.

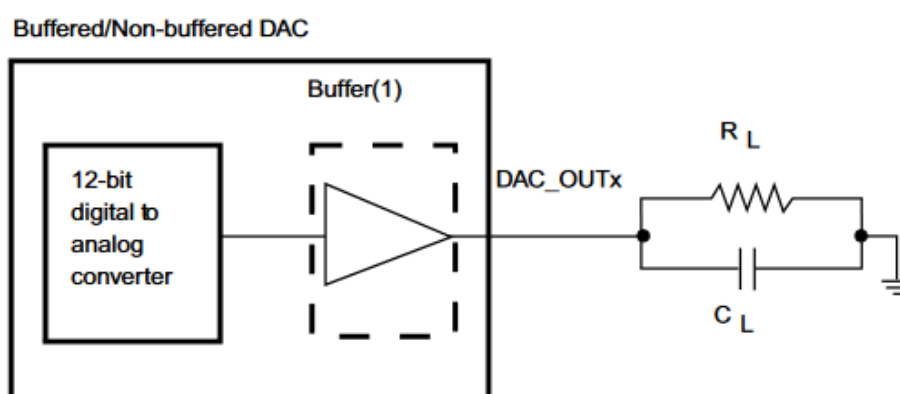


Obrázek 29 – Zapojení výstupu zesilovače ke vstupnímu pinu mikrokontroléru

3.5 Připojení zvukového výstupu

Výstupní JACK (konkrétně špička konektoru) je připojen přes kondenzátor C1 přímo k výstupu druhého D/A převodníku (druhý kanál D/A je připojen k pinu PA5 na mikrokontroléru).

Pro snížení výstupní impedance je třeba zapnout zabudovaný buffer stejnosměrného signálu na výstupu D/A převodníku v procesoru, čímž dosáhneme také nezávislosti přesnosti převodníku na výstupní impedanci. Negativními stránkami zapnutého bufferu jsou značné zvýšení šumu na výstupu a omezení napětového rozsahu výstupu na 0,2 až 3,4 V (odpovídá 12bitovým číslům E0H až F1CH). [18]

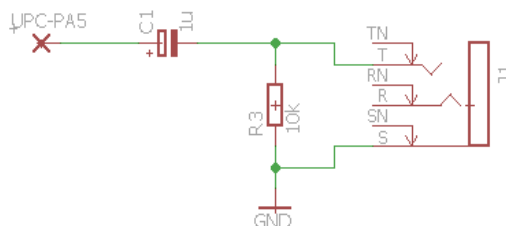


Obrázek 30 – Zapojení D/A převodníku v mikrokontroléru spolu s náhradním schématem výstupní zátěže [18]

Kondenzátor C1 v tomto zapojení blokuje stejnosměrný signál a projde jím pouze v čase proměnný proud (signál) – není zvykem posílat do vstupů zesilovačů střídavý signál se stejnosměrnou složkou. Rezistor R3 funguje v zapojení jako výstupní impedance z důvodu

snížení šumu. Toto zapojení také funguje jako horní propust 1. řádu s mezní frekvencí přibližně 15,9 Hz, což je v tomto případě nežádoucí efekt – tato frekvence by šla snížit zvýšením odporu R3 nebo kapacity C1. Tímto zapojením je také určena minimální připojitelná impedance 10 k Ω (paralelní kombinace připojené impedance a R3), jelikož minimální možná výstupní impedance na výstupu bufferu převodníku je 5 k Ω . [2][18]

Toto zapojení není součástí hlavní DPS a proto není součástí ani hlavního schématu.

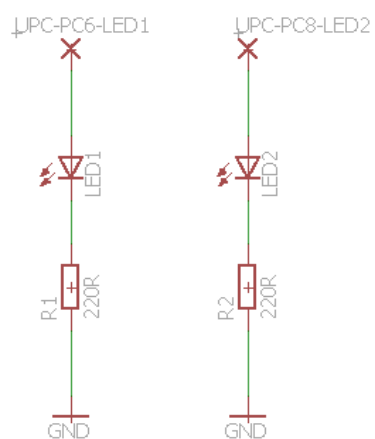


Obrázek 31 – Zapojení zvukového výstupu k výstupnímu pinu mikrokontroléru

3.6 Připojení varovných LED

Zapojení obsahuje 2 varovné červené LED diody, umístěné u vstupního (LED1) a u výstupního (LED2) JACKu. Jejich buzení je řešeno přímo piny mikrokontroléru a rezistory s odporem 220 Ω (proud přibližně 8 mA na každou). Jejich zapojení není součástí hlavní DPS a proto nejsou součástí ani hlavního schématu.

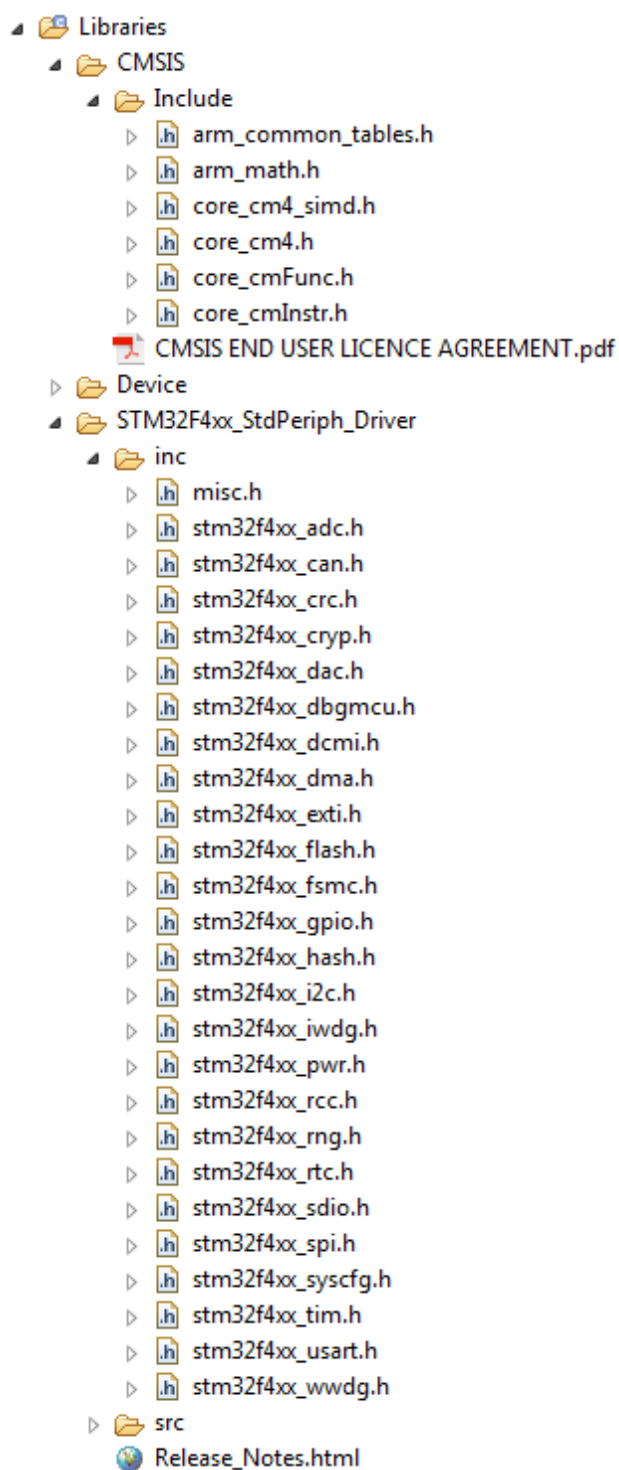
Jejich účelem je signalizace clippingu signálu na vstupu a na výstupu a poskytují tak jednoduchou zpětnou vazbu pro správné nastavení zisku zesilovače při oživování zařízení. Při nastavování zesílení je třeba na strunný nástroj hrát silněji (hruběji) než obvykle, aby se nasimulovala extrémní situace, která může při hraní nastat, a přitom zvyšovat zesílení změnou posuvného odporu R16 až do bodu, kdy již začne blikat jedna z varovných LED – v ten moment se ještě jemně sníží zesílení a nastavování je dokončeno. Tímto nastavením maximalizujeme použitý napěťový rozsah převodníků (jak vstupních tak výstupních) a dosáhneme vyšší kvality zvuku.



Obrázek 32 – Zapojení va-
rovných LED

4 PROGRAMOVÉ ŘEŠENÍ

Při programovém řešení byla využita základní kostra programu pro použitý mikrokontrolér obsahující mimo inicializační procedury také základní systémové ovladače CMSIS a periferní ovladače, které jsou kompatibilní pro všechny mikrokontroléry řady STM32Fxx.



Obrázek 33 – Použité CMSIS knihovny a univerzální periferní ovladače

Veškeré zdrojové kódy lze nalézt v elektronické příloze, viz Příloha III, popř. kompletní projekt, viz Příloha IV.

4.1 Ovladač pro vstupní ovládání

Ovladač sestává ze dvou souborů – z hlavičkového souboru „LK_drv_controls.h“ a ze zdrojového souboru „LK_drv_controls.c“. Tento ovladač poskytuje rozhraní pro uživatelské prostředí (ovládací prvky) - mechanické spínače a enkodéry.

Ovládání hlavního programu změnou ovládacích prvků je realizováno skrz globální proměnné, jejichž názvy odpovídají daným ovládacím prvkům, např. „LK_CTRL_enc1change“ nebo „LK_CTRL_footswpressed“. Jednotlivé proměnné jsou pole booleovských typů s velikostí 2. Také je definována globální proměnná pole booleovských typů „LK_CTRL_busy“ s velikostí 2, která slouží k vyřešení souběhu hlavního programu a ovladače – pokud bude hlavní program právě číst a zpracovávat údaje v nultých indexech polí tak zapíše do nultého indexu „LK_CTRL_busy“ log. H, aby ovladač při zápisu použil první indexy polí (opačné chování, pokud bude hlavní program zpracovávat první indexy polí). Hlavní program si cyklicky střídá tyto indexy. Na rozdíl od pouhého zakázání přerušení při čtení hlavním programem takto nedojde ke ztrátě změny uživatelského ovládacího prvku, což se projeví na přívětivější a responsivnější uživatelské odezvě.

Pro zpřehlednění kódu je definována řada maker v hlavičkovém souboru. Tyto umožňují snadnou změnu elektronických a číslcových parametrů pro snadnější ladění či změnu chování programu.

V ovladači je nejprve povoleno časování použitých sběrnic (AHB1 pro GPIO, APB2 pro SYSCFG - EXTI). Poté jsou inicializovány všechny použité GPIO porty dle základního nastavení v makrech. Následně se postupně přidává každý ovládací prvek, pro který chceme povolit přerušení, do nastavení EXTI (nastavení externího přerušení) a povoluje se přidáním dané EXTI linky do vektoru přerušení NVIC.

Každá z obsluh přerušení má podobné chování – nejprve se zkontroluje přerušovací příznak, a pokud je platný tak se zavolá funkce „LK_CTRL_ProcessInput()“ a předá se jeden z členů typu „LK_CTRL_input“. Teprve až po provedení funkce je vynulován přerušovací příznak.

Pomocná funkce „LK_CTRL_ProcessInput()“ slouží ke zpracování vstupních ovládacích prvků – vybírá se jeden z nich, který vyvolal přerušení, při volání funkce prvkem z typu

„LK_CTRL_input“. Ve funkci se nejprve kontroluje stav pole „LK_CTRL_busy“, který z indexů se nepoužívá pro čtení – hlavní program probíhá velmi rychle a je velká pravděpodobnost, že téměř vždy při zápisu bude zároveň probíhat i čtení. Ten, který se nepoužívá pro čtení, se využije pro zápis (index je zaznamenán do pomocné proměnné „used_index“). Následuje obecný debounce delay, pro všechny tlačítka a enkodéry - jeho délka trvání byla empiricky na několik desítek až stovek μ s. Po skončení prodlevy se zkontroluje použitý prvek z typu „LK_CTRL_input“ a podle něj dojde ke kontrole logického stavu příznačného vstupního portu (externí přerušení je vyvoláno vždy jen pádem z log. H do log. L, tudíž je chování vždy jasně dané). Pro enkodéry se podle fáze signálu u druhého z jejich pinů inkrementuje či dekrementuje hodnota v příslušném poli na indexu „used_index“. Pro tlačítka se kontroluje, zda jsou i po prodlevě v log. L, pokud ano je zapsána do příslušného pole na index „used_index“ hodnota log. H (stisknuto). Jelikož robustní nožní tlačítko („FOOTSW“) vyvolává větší množství zákmitů, bylo nutné doplnit před kontrolou jeho logického stavu ještě delší debounce delay (až na několik jednotek ms).

4.2 Ovladač displeje

Ovladač sestává ze dvou souborů – z hlavičkového souboru „LK_drv_display.h“ a ze zdrojového souboru „LK_drv_display.c“. V tomto ovladači jsou všechny potřebné funkce pro ovládání osmi 7segmentových displejů pro zobrazování informací.

Ovládání displejů z hlavního programu je realizováno skrz globální proměnnou „LK_DSP_displayed_string“ a globální pole booleovských hodnot „LK_DSP_displayed_dots“. Přepisováním těchto proměnných lze měnit zobrazený sled znaků na displejích a rozsvěcovat/zhasínat tečky na jednotlivých displejích.

Pro usnadnění ovládání je v hlavičkovém souboru definována řada maker, které značně zpřehledňují kód. Také je ve zdrojovém souboru definováno pomocné makro pro testování určitého bitu v 8bitové proměnné.

V ovladači je nejprve povoleno časování použitých sběrnic (AHB1 pro GPIO a APB1 pro časovač TIM). Následuje inicializace všech použitých portů displeji a po inicializaci jejich nastavení na log. L po dobu inicializace. Poté je inicializováno přerušení od časovače TIM – nejprve je přidáno do vektoru přerušení NVIC a následně je inicializován samotný časovač TIM danými číselnými konstantami, zaručujícími obnovování displeje jako celku frek-

venci 100Hz (empiricky zvolená hodnota). Na konci je časovač TIM povolen příkazem ENABLE.

Ovladač obsahuje také funkce k zakázání a povolení přerušení pro displej za účely vyhnutí se souběhu funkcí (nedojde k výpisu neplatné informace na displej, zatímco je string přepisován v hlavním programu). Tyto funkce jsou realizovány přímými příkazy časovači TIM pro vypnutí a zapnutí čítání (DISABLE a ENABLE).

V obsluze přerušení je nejprve zkontrolován příznak přerušení, pokud je časovač TIM opravdu zdrojem přerušení, pokud ano, vymaže se tento příznak a časovač začne hned počítat znovu. Následně je definována statická proměnná „display_cnt“, která slouží k číslování displejů a zároveň k indexaci stringu. Poté je zkontrolováno, zda-li je hodnota této proměnné validní, pokud ne je proměnná nulována. Následně je předán funkci „LK_DSP_Show()“ znak z globálního stringu z indexu „display_cnt“, který se má zobrazit na „display_cnt“-tém displeji a jako poslední parametr funkce je předán booleovský typ, určující zda má být na daném displeji rozsvícena tečka – tento údaj je načten z globálního pole „LK_DSP_displayed_dots“ z indexu „display_cnt“. Po provedení funkce je zvýšena hodnota proměnné „display_cnt“ o 1.

Ve funkci „LK_DSP_Show()“ dochází k samotnému nastavování výstupů podle předaných vstupních parametrů. Nejprve je podle pořadí displeje „display_num“ aktivován příslušný displej nastavením příslušné adresy na adresních výstupních bitech. Následuje převod vstupního znaku „character“ na posloupnost 7 bitů uložené v 8bitové proměnné „segments“, kde každý bit odpovídá příslušnému segmentu na displeji (šestý bit odpovídá segmentu G, nultý bit odpovídá segmentu A). K usnadnění grafického návrhu znaků bylo využito předem vytvořené excelové tabulky volně dostupné ze zdroje [20] od tvůrce Jose Pino, která přímo generuje 8bitové číslo podle segmentů, které jsou v excelu označeny, že mají svítit (vyplněné číslem 1). Při designu nebylo využito tečky u displejů, jelikož rozsvěcování a zhasínání teček je ovládáno nezávisle na zobrazeném stringu a to při zobrazování desetinných čísel a při odezvách uživatelského prostředí prostřednictvím výše zmíněného pole „LK_DSP_displayed_dots“. Po převedení je 8bitové číslo bit po bitu testováno a jsou podle něj rozsvěcovány/zhasínány odpovídající segmenty.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA	AI	AAIAI	AG	AI	AAAIAI	AM	AI	AAAIAI	AS	A'AA		
1																																							
2																																							
3																																							
4																																							
5																																							
6																																							
7																																							
8																																							
9																																							
10																																							
11																																							
12																																							
13																																							
14																																							
15																																							
16																																							
17																																							
18																																							
19																																							
20																																							
21																																							
22																																							
23																																							
24																																							
25																																							
26																																							
27																																							

Obrázek 34 – Ukázka excelové tabulky pro grafický návrh znaků displeje

4.3 Ovladač vstupního A/D převodníku

Ovladač sestává ze dvou souborů – z hlavičkového souboru „LK_drv_ADCinput_gtr.h“ a ze zdrojového souboru „LK_drv_ADCinput_gtr.c“. V tomto ovladači se nacházejí všechny funkce související s prací vstupního A/D převodníku.

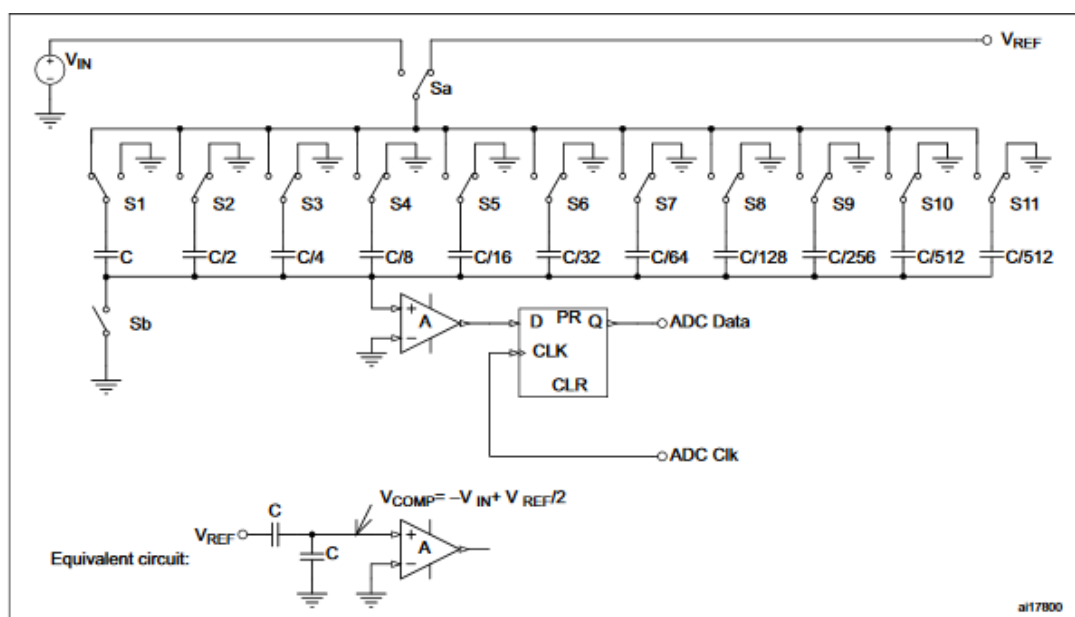
Pro větší pohodlnost a flexibilitu ovladače je definována řada maker v hlavičkovém souboru, které umožňují snadnou záměnu vstupního převodníku za jiný (celkem jsou k dispozici 3 v procesoru), záměnu vstupního A/D kanálu (celkem jich je k dispozici 16) a také umožňují snadnější změnu parametrů převodu pro účely ladění převodu. [18]

Vstupní GPIO port je nastaven na alternativní analogový mód a rychlost časování portu je nastavena na 50MHz (druhá nejrychlejší).

A/D převodníky v procesoru fungují na principu aproximativního SAR (Successive approximation register neboli Registr postupného přibližování), kdy je hodnota postupně přibližována nejprve změnou MSB a poté postupnou změnou bitů až k LSB – hodnota tak postupně konverguje (přibližuje se) ke správné hodnotě. Na rozdíl od běžné implementace s použitím pomocného D/A převodníku k porovnávání nastavené hodnoty se vstupní hodnotou je zde využíváno nabíjení dvanácti kondenzátorů (počet podle počtu bitů převodníku) – jejich hodnoty jsou odstupňovány dělením kapacit funkcí 2^N . Po nabití sady kondenzátorů

na hodnotu vstupního napětí V_{IN} se přepne vstup pro nabíjení na V_{REF} a začne se postupně porovnávat nabitá hodnota s určitým poměrem referenčního napětí (poměr je zvětšován jmenovatel funkcí 2^N) – tento poměr se mění v závislosti na právě připojených kondenzátorech (přepínání kondenzátorů je realizováno pomocí sady elektronicky ovládaných přepínačů). Např. pokud v prvním kroku převodu byl zjištěný $MSB = 1$, tak v druhém kroku převodu se bude porovnávat vstupní napětí V_{IN} s napětím $\frac{3}{4}V_{REF}$ dle pravidla $\left(\frac{1}{2^1} + \frac{1}{2^2}\right)V_{REF}$.

Pokud by byl $MSB = 0$, tak v druhém kroku by se porovnávalo vstupní napětí V_{IN} s napětím $\frac{1}{4}V_{REF}$ podle pravidla $\left(0 + \frac{1}{2^2}\right)V_{REF}$. Na obrázku níže lze vidět druhý krok převodu – kondenzátory jsou již nabity z prvního kroku na vstupní napětí V_{IN} a právě se porovná vstupní napětí s polovinou referenčního napětí V_{REF} . Výsledek porovnání určí, zda bude $MSB = 1$ (napětí na vstupu je vyšší než polovina V_{REF}) nebo $MSB = 0$ (napětí na vstupu je nižší než polovina V_{REF}). [14]



Obrázek 35 – Ukázka zapojení A/D převodníku v druhé fázi převodu [14]

V ovladači je nejprve povoleno časování použitých sběrnic (AHB1 pro GPIO a APB2 pro A/D převodníky) a poté jsou inicializovány GPIO porty, jeden ke kterému je připojen vstup ze zesilovače (port PA0) a druhý ke kterému je připojena varovná LED (port PC6). Následuje inicializace samotného A/D převodníku a v závěru inicializační funkce je povolen A/D převodník příkazem ENABLE.

Samotné čtení vzorku z A/D převodníku je poté odstartováno podpůrnou funkcí „ADC_SoftwareStartConv()“. Po odstartování následuje čekání na EOC flag od použitého A/D převodníku, který značí úspěšný konec převodu vzorku. V čekání je také zabudována jednoduchá ochrana proti poruše převodníku - pokud by převod trval déle než má, vrátí čtecí funkce 0 namísto hodnoty převedeného vzorku. Po úspěšném převodu je vzorek přečten z registru převodníku a před vrácením je ještě zkontrolováno, zda nedošlo ke clippingu signálu (k němu dojde, pokud je hodnota vzorku větší nebo rovna číslu $FFF_H - 1$ nebo pokud je menší nebo rovna číslu $000_H + 1$) – pokud ke clippingu došlo, rozsvítí se varovná LED.

A/D převodník je nastaven pro jednotlivé čtení vzorků „na počkání“, tudíž kromě zmíněného volání převodník sám další vzorky nečte a pouze čeká na další start převodu.

4.4 Ovladač výstupního D/A převodníku

Ovladač sestává ze dvou souborů – z hlavičkového souboru „LK_drv_DACOutputMCU_gtr.h“ a ze zdrojového souboru „LK_drv_DACOutputMCU_gtr.c“. V tomto ovladači se nacházejí veškeré výstupní operace D/A převodníku a také je zde definovaná hlavní přerušovací funkce, zajišťující zjištění vzorku ze vstupu, následné zpracování vzorku a jeho odeslání do D/A převodníku.

V ovladači je nejprve přiřazena callback funkce, která se bude volat při přerušení. Tato funkce obstarává vše od přečtení po zpracování a jejím výstupem je 16bitový zpracovaný vzorek, který se má poslat na výstup. Následuje povolení časování použitých sběrnic (AHB1 pro GPIO a APB1 pro D/A převodníky a pro časovač TIM) a poté dochází k inicializaci použitých GPIO portů tj. jeden ke kterému je připojen výstup D/A převodníku (port PA5) a druhý, ke kterému je připojena varovná LED (port PC8). Poté je inicializován samotný D/A převodník. Po všech inicializacích portů a jejich módů se inicializuje přerušení časovačem TIM, které je nejprve přidáno do vektoru přerušení NVIC a poté je inicializován samotný časovač TIM číselnými konstantami zaručujícími přerušení o frekvenci 48 kHz ke vzorkování zvuku (jejich výpočet je v komentáři kódu v hlavičkovém souboru). Na závěr jsou povoleny časovač TIM a D/A převodník příkazem ENABLE.

Ovladač taktéž umožňuje zakázání a povolení přerušení za účelem utlumení výstupu, pokud víme, že na vstupu je pouze neplatný signál (k tomu dochází např. při nepřipojeném vstupu, kdy zesilovač zesiluje jenom šum). Tyto funkce jsou realizovány povolením či

zakázáním čítání časovače přímým příkazem časovači TIM (DISABLE pro vypnutí časovače, ENABLE pro zapnutí časovače) - pak nemůže přerušení nastat, jelikož bez čítání nedojde k přetečení registru časovače, které toto přerušení vyvolává. Při vypnutém přerušení je na výstupu D/A převodníku konstantní hodnota (konstantní napětí) a ta je rovna poslední nastavené hodnotě z poslední obsluhy přerušení – na výstupním JACKu je pak nulové napětí, jelikož nedochází ke změně napětí a stejnosměrné napětí na výstupu je blokováno.

V obsluze přerušení je nejprve definována statická proměnná pro uložení posledního vzorku z posledního proběhlého přerušení – v prvním přerušení se inicializuje na hodnotu 0. Následně je zkontrolován příznak přerušení časovače TIM, a pokud je validní, je vymazán, aby započalo nové čítání. V dalším kroku je na výstup převodníku poslána hodnota vzorku z předem zmíněné statické proměnné. Tato hodnota je následně otestována, zda nedošlo ke clippingu a pokud je zjištěn clipping signálu tak je rozsvícena varovná LED. Na konci je zavolána callback funkce zpracování, která zajistí nový zvukový vzorek pro další přerušení a uloží ho po úpravě na 12bitové číslo do statické proměnné.

Tímto řešením dochází k prodlevě mezi vstupním a výstupním signálem a ta je v nejhorším případě rovna délce periody vzorkování (20,83 μ s). V reálném případě je to o několik jednotek μ s méně, jelikož odeslání dat na D/A převodník, kontrola clippingu vzorku a přečtení vzorku ze vstupu dohromady trvá určitou dobu, o kterou je tato vstupně-výstupní prodleva kratší.

4.5 Knihovna matematického zpracování zvuku

Knihovna sestává ze dvou souborů – z hlavičkového souboru „LK_mathprocessing.h“ a ze zdrojového souboru „LK_mathprocessing.c“. V této knihovně se nacházejí všechny funkce, potřebné ke zpracování zvukového vzorku.

Funkce v knihovně jsou navrženy především jako pomocné funkce pro volání z hlavní zpracovávací callback funkce v přerušení, nicméně jejich algoritmus je navržen způsobem, který neztrácuje jejich použití i v jiných druzích volání – např. zpracovávací efektové funkce jsou po správné inicializaci pomocných proměnných schopny korektně pracovat samy o sobě (pouze je nutné brát v potaz, že využívají několik globálně přístupných proměnných).

Pro optimalizaci rychlosti zpracování je v hlavičkovém souboru předdefinováno pole konstant s datovým typem float, které obsahuje předpočítané hodnoty matematické goniometrické funkce sinus (pomocí standardní knihovny <math.h> trvá výpočet sinu příliš mnoho cyklů, v řádu stovky až tisíce cyklů, na rozdíl od vracení hodnoty z pole, které je záležitostí několika jednotek až desítek cyklů). Počet hodnot v poli je 181 a indexování je od 0, což je vhodné pro snadnější použití v kódu – funkce je vypočítána pro úhly 0 až 180° a pole se volá přímo s indexem rovnajícím se žádanému úhlu. Přesnost vypočítaných hodnot je na 6 desetinných míst a spolu s počtem úhlů dostačují pro použití ve zpracování.

U všech funkcí lze částečně měnit chování a charakter efektů změnou parametrů. Zapínání efektů je řešeno pomocí podpůrné funkce „LK_PR_EnableEffect()“ a vypínání pomocí funkce „LK_PR_DisableEffect()“ – obě funkce se volají jedním z členů z výčtového typu „effect_type“. Taktéž je možné vypnout všechny efekty funkcí „LK_PR_DisableAllEffects()“. Změna parametru 1 (příznačný parametr efektu) nebo 2 (hlasitost) se provádí funkcí „LK_PR_SetEffectPar()“, která se volá jedním z členů z výčtového typu „effect_type“, hodnotou o kterou se má parametr zvětšit či zmenšit a booleovský typ určující zda se právě nastavuje parametr 2. Taktéž je možné pomocí funkce „LK_PR_SetEffectParDefault()“ nastavit u žádaného efektu základní hodnoty. Všechny tyto externě nastavitelné parametry budou pro zpřehlednění v algoritmech označovány jako globální ovladače.

Následuje popis jednotlivých zpracovávacích algoritmů zvukových efektů, jejich základní princip byl již uveden v teoretické části této práce a algoritmy budou s drobnými úpravami mechanismu založené na nich. Všechny popisované funkce přebírají zvukový vzorek „sample“ jako float a po skončení funkce vrací zpracovaný vzorek ve stejném formátu.

4.5.1 Algoritmus funkce Delay

Funkce si při prvním zavolání vytvoří pomocné statické proměnné:

- pole „delay_buffer“ s délkou n vzorků (počet určuje konstanta „LK_PR_DELAY_BUFFER_SIZE“) pro ukládání vzorků
- proměnnou „delay_buffer_i“ pro indexaci pole „delay_buffer“

Funkce také vytvoří pomocnou proměnnou „ret_value“, která bude představovat návratovou hodnotu funkce.

Algoritmus efektu:

- 1) ulož hodnotu nezpracovaného vzorku z proměnné „sample“ do proměnné „ret_value“
- 2) zkontroluj globální ovladač „LK_PR_delay_on“, zda-li je efekt zapnut či nikoliv - pokud ano pokračuj dalším bodem, pokud ne pokračuj bodem 7
- 3) zjisti hodnotu indexu, zpožděného o x vzorků (určuje globální ovladač „LK_PR_delay_sample_count“) ku poslednímu použitému indexu (proměnná „delay_buffer_i“)
- 4) ulož hodnotu z pole „delay_buffer“ z indexu zjištěného v předchozím bodě do nově deklarované pomocné proměnné „pom“
- 5) vynásob hodnotu v proměnné „pom“ určitou mírou hlasitosti v rozmezí 0 až 0,8 (určuje podíl globálního ovladače hlasitosti „LK_PR_delay_vol“ a čísla 1000)
- 6) získanou hodnotu přičti k hodnotě v proměnné „ret_value“ a výsledek ulož do „ret_value“
- 7) ulož do pole „delay_buffer“ na index „delay_buffer_i“ hodnotu „ret_value“
- 8) zvyš hodnotu „delay_buffer_i“ o 1
- 9) pokud je nová hodnota „delay_buffer_i“ větší nebo rovna délce pole n (určuje „LK_PR_DELAY_BUFFER_SIZE“) tak vynuluj proměnnou „delay_buffer_i“
- 10) vrať hodnotu „ret_value“ do místa volání funkce

Pokud by se jakýmkoliv způsobem podařilo zapnout efekt (globálním ovladačem „LK_PR_delay_on“) před naplněním pole zvukovými vzorky (cca několik prvních stovek ms po zapnutí zpracovávání) tak by efekt vracel pouze nezpracované vzorky, jelikož pole je před přepsáním prázdné (vyplněno nulami).

4.5.2 Algoritmus funkce Tremolo

Funkce si při prvním zavolání vytvoří pomocné statické proměnné:

- proměnnou „tremolo_cnt“ pro počítání prodlevy mezi změnami amplitudy
- proměnnou „tremolo_index“ pro ukládání momentálního používaného úhlu funkce sinus

Funkce také vytvoří pomocnou proměnnou „ret_value“, která bude představovat návratovou hodnotu funkce.

Algoritmus efektu:

- 1) ulož hodnotu nezpracovaného vzorku z proměnné „sample“ do proměnné „ret_value“
- 2) zkontroluj globální ovladač „LK_PR_tremolo_on“, zda-li je efekt zapnut či nikoliv – pokud ano pokračuj dalším bodem, pokud ne pokračuj bodem 10
- 3) pokud je hodnota proměnné pro počítání prodlevy „tremolo_cnt“ větší nebo rovna hodnotě nastavené v globálním ovladači „LK_PR_tremolo_speed_divider“ tak pokračuj dalším bodem, pokud ne pokračuj bodem 7, ale nejprve zvyš hodnotu proměnné „tremolo_cnt“ o 1 (v obou případech)
- 4) vynuluj proměnnou pro počítání prodlevy „tremolo_cnt“
- 5) zvyš hodnotu v proměnné „tremolo_index“ o 1
- 6) pokud je hodnota proměnné „tremolo_index“ vyšší nebo rovna 181 (v tomto případě velikost pole s pomocnou předpočítanou funkcí sinus) tak vynuluj proměnnou „tremolo_index“
- 7) zjistí hodnotu sinu pro úhel „tremolo_index“
- 8) vynásob nalezenou hodnotu z předchozího bodu s hodnotou v proměnné „ret_value“ a ulož ji do „ret_value“
- 9) vynásob hodnotu v proměnné „ret_value“ určitou mírou hlasitosti v rozmezí 0 až 1 (určuje podíl globálního ovladače hlasitosti „LK_PR_tremolo_vol“ a čísla 1000) a výsledek ulož do „ret_value“
- 10) vrať hodnotu z proměnné „ret_value“ do místa volání funkce

4.5.3 Algoritmus funkce Flanger

Funkce si při prvním zavolání vytvoří pomocné statické proměnné:

- proměnnou „flanger_speed_count“ pro počítání prodlevy mezi změnami délky prodlevy efektu
- proměnnou „flanger_delay_count“ pro ukládání momentální délky prodlevy mezi vzorky, při startu je inicializována hodnotou z makra „LK_PR_FLANGER_MIN_DELAY“

- proměnnou booleovského typu „flanger_count_up“ pro ukládání momentálního druhu operace (zvyšování/snižování prodlevy mezi vzorky), při startu je inicializována na hodnotu 1 (zvyšování prodlevy)
- pole „flanger_buffer“ s délkou n vzorků (určuje hodnota makra „LK_PR_FLANGER_MAX_DELAY“) pro ukládání vzorků
- proměnnou „flanger_buffer_i“ pro indexaci pole „flanger_buffer“

Funkce také vytvoří pomocnou proměnnou „ret_value“, která bude představovat návratovou hodnotu funkce.

Algoritmus efektu:

- 1) ulož hodnotu nezpracovaného vzorku z proměnné „sample“ do proměnné „ret_value“
- 2) zkontroluj globální ovladač „LK_PR_flanger_on“, zda-li je efekt zapnut či nikoliv – pokud ano pokračuj dalším bodem, pokud ne pokračuj bodem 12
- 3) pokud je hodnota proměnné pro počítání prodlevy „flanger_speed_count“ větší nebo rovna hodnotě v globálním ovladači „LK_PR_flanger_speed“ tak pokračuj dalším bodem, pokud ne pokračuj bodem 8, ale nejprve zvýš hodnotu proměnné „flanger_speed_count“ o 1
- 4) vynuluj proměnnou pro počítání prodlevy „flanger_speed_count“
- 5) pokud je momentální délka prodlevy mezi vzorky (proměnná „flanger_delay_count“) delší nebo stejná jako nejvyšší povolená prodleva (makro „LK_PR_FLANGER_MAX_DELAY“) tak nastav směr čítání prodlevy (určuje proměnná „flanger_count_up“) na odečítání (snížování prodlevy mezi vzorky) a pokračuj bodem 7
- 6) pokud byla předchozí podmínka nesplněna zkontroluj, zda je momentální délka prodlevy mezi vzorky (určuje proměnná „flanger_delay_count“) kratší nebo stejná jako nejnižší povolená prodleva (makro „LK_PR_FLANGER_MIN_DELAY“), pokud ano tak nastav směr čítání prodlevy (určuje proměnná „flanger_count_up“) na přičítání (zvyšování prodlevy mezi vzorky)

- 7) pokud je směr čítání prodlevy (určuje proměnná „flanger_count_up“) nastaven na přičítání tak zvýš hodnotu momentální délky prodlevy (proměnná „flanger_delay_count“) o 1, pokud není, sniž tutéž hodnotu o 1
- 8) vypočti hodnotu indexu zpožděného o x vzorků (proměnná „flanger_delay_count“) ku poslednímu použitému indexu (proměnná „flanger_buffer_i“)
- 9) ulož hodnotu z pole „flanger_buffer“ z indexu vypočteného v předchozím bodě do proměnné „ret_value“
- 10) vynásob hodnotu v proměnné „ret_value“ určitou mírou hlasitosti v rozmezí 0 až 0,8 (určuje podíl globálního ovladače hlasitosti „LK_PR_flanger_vol“ a čísla 1000) a výsledek ulož do „ret_value“
- 11) vynásob hodnotu nezpracovaného vzorku z proměnné „sample“ s rozdílem míry hlasitosti $1 - \text{použitá míra hlasitosti}$ v předchozím bodě ($1 - \text{„LK_PR_flanger_vol“}/1000$), výsledek přičti k hodnotě v „ret_value“ a výsledek ulož do „ret_value“ (jednoduché zmixování 2 vzorků)
- 12) ulož hodnotu z „ret_value“ do pole „flanger_buffer“ na index „flanger_buffer_i“ a následně zvýš hodnotu „flanger_buffer_i“ o 1
- 13) pokud je index „flanger_buffer_i“ větší nebo rovno nejvyšší povolené prodlevě (makro „LK_PR_FLANGER_MAX_DELAY“) tak vynuluj proměnnou „flanger_buffer_i“
- 14) vrať hodnotu z „ret_value“ do místa volání funkce

Stejně jako u efektu delay i tady se algoritmus potýká s velmi malou pravděpodobností toho, že se podaří efekt zapnout dříve, než bude pomocné pole „flanger_buffer“ zaplněno platnými daty, ale i tady je problém vyřešen navrácením nezpracovaných vzorků.

4.5.4 Algoritmus funkce Chorus

Algoritmus je téměř totožný s algoritmem funkce Flanger, jediné rozdíly jsou jiné rozsahy hodnot pro proměnné prodlevy (rozsah je posunut výš – nejvyšší i nejnižší možná prodleva jsou delší než u funkce Flanger) a je vypuštěna zpětná vazba při ukládání vzorků do pomocného pole (v bodě 12 se do pole místo hodnoty z proměnné „ret_value“ ukládá hodnota nezpracovaného vzorku z proměnné „sample“)

4.5.5 Algoritmus funkce Octaver

Funkce nepoužívá statické pomocné proměnné, pouze si vytvoří pomocnou proměnnou „ret_value“, která bude představovat návratovou hodnotu funkce.

Algoritmus efektu:

- 1) ulož hodnotu nezpracovaného vzorku z proměnné „sample“ do proměnné „ret_value“
- 2) zkontroluj globální ovladač „LK_PR_octaver_on“, zda-li je efekt zapnut či nikoliv – pokud ano pokračuj dalším bodem, pokud ne pokračuj bodem 7
- 3) pokud je vstupní vzorek (proměnná „sample“) kladný (větší nebo rovno 0) tak ulož jeho hodnotu do proměnné „ret_value“, pokud ne tak ulož jeho opačnou hodnotu do proměnné „ret_value“
- 4) vynásob hodnotu v proměnné „ret_value“ určitou mírou hlasitosti v rozmezí 0 až 1 (určuje podíl globálního ovladače hlasitosti „LK_PR_octaver_vol“ a čísla 1000) a výsledek ulož do „ret_value“
- 5) posuň hodnotu v proměnné „ret_value“ do kladných či záporných hodnot tak, že přičteš podíl globálního ovladače „LK_PR_octaver_displacement“ a čísla 1000 k hodnotě v „ret_value“ a výsledek zapišeš do „ret_value“
- 6) vynásob hodnotu nezpracovaného vzorku z proměnné „sample“ s rozdílem míry hlasitosti 1 – použitá míra hlasitosti v bodě 4 ($1 - \text{LK_PR_octaver_vol}/1000$), výsledek přičti k hodnotě v „ret_value“ a výsledek ulož do „ret_value“ (jednoduché zmixování 2 vzorků)
- 7) vrať hodnotu z proměnné „ret_value“ do místa volání funkce

4.6 Hlavní program

Tento program se nachází ve zdrojovém souboru „main.c“.

Program zahrnuje všechny výše zmíněné ovladače, několik standardních knihoven (<stdio.h>, <stdlib.h>, <string.h> a <stddef.h> pokud je používán NULL) a knihovnu „stm32f4_discovery.h“, která zahrnuje „stm32f4xx.h“, kde jsou všechny základní funkce použitého procesoru a nutné inicializace procesoru. Program také obsahuje makro pro nastavení prodlevy mezi změnami stringů pro displej „DISPLAY_CHANGE_WAIT_TIME“,

kterým se dá snadno doladit uživatelské prostředí, pokud by změny hlášek byly příliš rychlé či pomalé.

V programu je dále definováno několik privátních proměnných:

- proměnná „active_effect“ pro potřeby stavového automatu v hlavní while smyčce, je v ní uložen momentálně zobrazovaný efekt, inicializováno na efekt „DELAY“
- proměnná „display_counter“ pro počítání prodlevy v hlavní while smyčce, při překročení hodnoty z makra „DISPLAY_CHANGE_WAIT_TIME“ zpravidla dojde ke změně stringu na displeji
- proměnná booleovského typu „effect_hold“, která označuje aktivitu zámku právě zobrazovaného efektu

Při spuštění je nejprve spuštěna funkce „SystemInit()“, kterou se inicializuje časování a důležité registry procesoru. Následuje inicializace displeje, ovládání, A/D převodníku, matematické knihovny a D/A převodníku. Při inicializaci jsou předány příslušné callback funkce matematické knihovně („LK_ADC_Read16b()“) a D/A převodníku („LK_PR_ReturnProcessedSample()“) pro správnou funkčnost ovladačů.

Následně je deklarován pomocný string „last_displayed_string“ a booleovská proměnná „while_entered“, které slouží k obsluze situace, kdy není zasunutý vstupní JACK. Tato situace nastane na začátku hlavní while smyčky pokud je mechanický kontrolní kontakt JACKu v log. L – pak je vstoupeno do while smyčky, která bude probíhat tak dlouho, dokud nebude vsunut vstupní JACK. V této smyčce se poprvé provede zakázání výstupu D/A převodníku a na displej je zapsána varovná hláška „NO INPUT“ přičemž se uloží předchozí vypsaný string na displeji, pro návrat k předchozímu stavu před vysunutím JACKu. Také je do proměnné „while_entered“ zapsána log. H, aby se v tomto cyklu neprováděly tyto operace vícekrát. Při opětovném zasunutí vstupního JACKu je opuštěn cyklus while a hned na to je (jelikož je proměnná „while_entered“ v log. H) po určité debounce prodlevě zobrazen zpět původní zobrazený string, povolen výstup D/A převodníku a do proměnné „while_entered“ je zapsána log. L.

V hlavní smyčce while je dále implementován jednoduchý stavový automat, který je řízen privátní proměnnou „active_effect“. Chování pro různé efekty se příliš neliší, liší se pouze zobrazenými stringy na displeji. Při vstupu do automatu je nejprve zkontrolován stav privátní proměnné „display_counter“ zda je vyšší nebo rovna makru

„DISPLAY_CHANGE_WAIT_TIME“, pokud ano tak je proměnná vynulována a na displej je zapsána základní hláška podle aktivního efektu, např. „delay on“. Také se popř. provede rozsvícení prvních třech teček, pokud je aktivní zámek efektu (proměnná „effect_hold“).

Na konci smyčky while je zavolána pomocná funkce „CheckInputs()“ a hodnota proměnné „display_counter“ je zvýšena o 1.

Funkce „CheckInputs()“ slouží ke zpracování dat z ovladače uživatelského ovládání. Nejprve si funkce vytvoří statickou proměnnou „used_index“ inicializovanou na hodnotu 0 a do pole „LK_CTRL_busy“ na index „used_index“ zapíše log. H, což značí, že budou čteny nulté indexy polí a v případě přerušení od některého ovládacího prvku v průběhu čtení se má zapisovat do prvního indexu. Poté je podmínkami kontrolováno, zda je některá z hodnot z pole změněná (má hodnotu jinou než 0) a pokud ano, proběhne příslušná obslužná funkce. Po kontrole a případném zpracování všech polí je uvolněn index zápisem log. L do pole „LK_CTRL_busy“ na index „used_index“ a následně je hodnota v proměnné „used_index“ znegována, díky čemuž se při průchodech touto funkcí zaručí, že bude index střídát pouze hodnotu log. L a log. H.

Funkce jednotlivých enkodérů a tlačítek jsou naprogramovány následovně:

- první enkodér slouží k přepínání efektů, pokud není povolen zámek efektu, v takovém případě by došlo pouze k probliknutí teček na všech displejích
- druhý enkodér slouží k nastavení prvního parametru aktivního efektu – délky prodlevy u efektu Delay, frekvenci LFO u efektu Flanger, Tremolo a Chorus a velikosti odchylky upraveného signálu u efektu Octaver
- třetí enkodér slouží k ovládání hlasitosti efektu nebo popř. poměru mixování efektu
- tlačítko prvního enkodéru slouží k zapnutí či vypnutí zámku právě aktivního efektu
- tlačítko druhého enkodéru slouží k nastavení základní hodnoty prvního parametru u právě aktivního efektu
- tlačítko třetího enkodéru slouží k nastavení základní hodnoty hlasitosti nebo poměru hlasitosti u právě aktivního efektu
- nožní přepínač slouží k zapnutí či vypnutí právě aktivního efektu

5 DOKONČENÍ A PREZENTACE ZAŘÍZENÍ

Výsledný hardware spolu s vývojovým kitem a konektory je vměstnán do obyčejné plechové krabičky. Ke správné funkčnosti zařízení bylo ještě nutné vyrobit napájecí kabel – ten je realizován jako propojovací redukce USB – CINCH s napájecím kabelem dlouhým 3 metry. Výsledné zařízení tak lze napájet jakýmkoliv obyčejným USB adaptérem nebo USB portem počítače.

Vzhled zařízení viz PŘÍLOHA P I.

Za účely prezentace byla také nahrána řada zvukových vzorků zařízení.

Nejprve byl nahrán přímo výstup zařízení propojením výstupního jacku audio procesoru a linkového vstupu zvukové karty PC. Pro propojení byla použita jednoduchá redukce mono JACK 6,3mm – stereo JACK 3,5mm.



Obrázek 36 – Propojovací redukce mono JACK 6,3mm – stereo JACK 3,5mm

Další řada vzorků byla nahrána využitím skutečného kytarového elektronkového komba typu Laney LC50-II. Reproduktor komba byl snímán dynamickým mikrofonom MONARCH DM-4100 a před připojením do linkového vstupu zvukové karty PC byl zvuk z mikrofону předzesílen využitím mixážního pultu Stage Line MMX-1621.



Obrázek 37 – Ukázka snímání zvuku z reproduktoru komba mikrofonom

V obou případech byla použita zvuková karta VIA High Definition VT1705 a zvuk byl nahráván v softwaru Audacity.

Veškeré nahrané audio vzorky jsou v elektronické příloze viz. Příloha V.

ZÁVĚR

V praktické části této práce byl navrhnout podpůrný hardware a software pro vývojový kit STM32F4DISCOVERY, který měl nabídnout jednoduché řešení digitálního zpracování zvuku v hudebním odvětví speciálně zaměřeném na multiefektové kytarové jednotky. Toto řešení se ukázalo jako dostatečně kvalitní s přihlédnutím na relativně nízkou cenu výroby pro spoustu běžných hudebníků. Např. zvolené řešení snímání zvuku integrovanými 12bitovými převodníky zní velmi dobře i přes technologické zvukové nedokonalosti takového počtu bitů.

Softwarové vybavení zařízení je minimální a slouží pouze pro demonstraci – programové řešení obsahuje pouze několik audio efektů a jejich uživatelských nastavení. V reálném použití by vývojový kit umožnil použít nespočet dalších efektů a možností jejich ovládání (např. propojení s PC pomocí USB OTG nebo Ethernet rozhraní za účely vytvoření vlastní efektové smyčky, oddělení a větší využití různých průběhů LFO, ukládání nastavení efektů, ...).

Zvolené elektronické řešení vstupního zesilovače je velmi dobré a i přes velké odpory v signálové cestě šumí jen minimálně a umožňuje dostatečně velké nastavení zesílení pro všechny běžné typy pasivních snímačů. Snímání výstupu zesilovače A/D převodníkem probíhá také bez problémů. Zapojení enkodérů a tlačítek je poměrně dobře vyřešeno R-C článkem a kromě hlavního nožního tlačítka nebylo třeba většího softwarového zásahu na omezení jejich zákmitů. Zobrazovací jednotka má ovšem s danými budícími odpory poměrně nízkou intenzitu osvětlení a bylo by vhodné ji zvětšit použitím menších hodnot budících odporů pro lepší čitelnost na slunci. Výstupní D/A převodník se také ukázal jako nevhodný pro audio využití, neboť příliš šumí (s připojeným i odpojeným výstupním bufferem). Vzhledem k velké hlasitosti zvukových aparatur, ke kterým má být zařízení připojeno, tak řešení naráží na první větší nevýhodu nízké ceny elektronického řešení hardwaru.

V softwarovém řešení je dobře vyřešena komunikace mezi jednotlivými ovladači, která umožňuje rychlé volání čtecích a zpracovávacích funkcí. V matematické knihovně se ovšem příliš neosvědčilo zpracování s datovým typem float – i přes to, že zařízení obsahuje koprocessor FPU, mikrokontrolér na zpracování v reálném čase mezi vzorkovacími impulsy nestačí (bylo nutné povolit optimalizaci kódu v kompilátoru GCC, což znemožňuje hloubkové ladění programu).

Pro výrobu v budoucnu by tedy bylo vhodné navrhnout nové elektronické zapojení audio výstupu nejlépe s externím D/A 12bitovým převodníkem, pro potlačení nežádoucího šumu ve výstupním audio signálu, popř. rovnou vyměnit vstupní i výstupní převodníky za kvalitnější externí 16bitové pro rozšíření dynamického rozsahu zvuku (značně se zvýší cena řešení).

V případě dalšího rozšiřování kódu o audio efekty by bylo nutné:

- opustit od používání datových typů float a přepsat kód matematické knihovny např. na 32bitové integery (hrozí snížení dynamického rozsahu zvuku)
- omezit počet současně zapnutých efektů pro šetření systémových prostředků (klesne výsledná flexibilita zařízení)
- optimalizovat kód použitím pouze statických a globálních proměnných (vede k nepřehlednému zdrojovému kódu)
- zvýšit taktovací frekvenci procesoru (ze 144 MHz na maximálních 180 MHz) + upravit časování v ovladačích na nové frekvence sběrnice

SEZNAM POUŽITÉ LITERATURY

- [1] VÁŇA, Vladimír. *ARM pro začátečníky*. 1. vyd. Praha: BEN – technická literatura, 2009, 195 s. ISBN 978-80-7300-246-6
- [2] PUNČOCHÁŘ, Josef. *Operační zesilovače v elektronice*. 5. vyd. Praha: BEN - technická literatura, 2002, 484 s. ISBN 80-7300-059-8.
- [3] TIŠNOVSKÝ, Pavel. Architektura mikrořadičů s jádrem ARM Cortex-M4. *ROOT.CZ* [online]. 2016 [cit. 2016-05-04]. Dostupné z: <http://www.root.cz/clanky/architektura-mikroradicu-s-jadry-arm-cortex-m4/>
- [4] STM32F4DISCOVERY. *STMicroelectronics* [online]. [cit. 2016-05-05]. Dostupné z: http://www.st.com/content/st_com/en/products/evaluation-tools/product-evaluation-tools/mcu-eval-tools/stm32-mcu-eval-tools/stm32-mcu-discovery-kits/stm32f4discovery.html
- [5] TrueSTUDIO IDE for ARM Development | Atollic. *Atollic* [online]. [cit. 2016-05-05]. Dostupné z: <http://timor.atollic.com/truestudio/>
- [6] Digitalizace analogového signálu. *Fyzika :: MEF* [online]. 2011 [cit. 2016-05-06]. Dostupné z: <http://fyzika.jreichl.com/main.article/view/1355-digitalizace-analogoveho-signalu>
- [7] NOVÁK, Petr. *Průmyslové řídicí systémy* [online]. Ostrava, 2013 [cit. 2016-05-16]. Dostupné z: http://projekty.fs.vsb.cz/463/edubase/VY_01_022/. Studijní materiál. Vysoká škola báňská - Technická univerzita Ostrava.
- [8] *Digital Audio Effects Processing with Csound* [online]. [cit. 2016-05-16]. Dostupné z: http://www.donreiman.com/Introductory_Home_Page.htm
- [9] VLACHÝ, Václav. *Praxe zvukové techniky*. Praha: Muzikus, 1995. ISBN 80-901537-6-3.
- [10] PREBBLE, Tim. Beautiful Tech: Roland RE 201 Space Echo. *Music of Sound* [online]. 2012 [cit. 2016-05-16]. Dostupné z: <http://www.musicofsound.co.nz/blog/beautiful-tech-roland-re-201-space-echo>
- [11] BYBEE, Jim. The Roland RE-201 Space Echo Story. *Roland* [online]. [cit. 2016-05-16]. Dostupné z: <http://www.roland.co.uk/blog/the-roland-re-201-space-echo-story/>

- [12] SHAFFER, David. Chorus guitar effects. *Your Hobby Hour* [online]. [cit. 2016-05-16]. Dostupné z: http://www.hobby-hour.com/guitar/chorus_effects.php
- [13] *General Guitar Gadgets* [online]. Missouri (USA) [cit. 2016-05-16]. Dostupné z: <http://www.generalguitargadgets.com/how-to-build-it/technical-help/schematics/>
- [14] *AN2834 Application Note: How to get the best ADC accuracy in STM32Fx Series* [online]. 2013, , 45 [cit. 2016-05-16]. Dostupné z: http://www.st.com/content/st_com/en/products/microcontrollers/stm32-32-bit-arm-cortex-mcus/stm32f4-series/stm32f407-417/stm32f407vg.html
- [15] SC04-11SRWA. *Kingbright* [online]. 2011, , 6 [cit. 2016-05-16]. Dostupné z: <http://www.hestore.hu/files/sc04.pdf>
- [16] 74HC138 3-to-8 line decoder/demultiplexer. *Nxp* [online]. 2015, , 18 [cit. 2016-05-16]. Dostupné z: http://www.nxp.com/documents/data_sheet/74HC_HCT138.pdf
- [17] EC12E. *Alps* [online]. , 6 [cit. 2016-05-16]. Dostupné z: <http://www.alps.com/prod/info/E/PDF/Switch/Encoder/EC12E/EC12E.PDF>
- [18] STM32F407VG datasheet. *STMicroelectronics* [online]. 2016, , 201 [cit. 2016-05-16]. Dostupné z: http://www.st.com/content/st_com/en/products/microcontrollers/stm32-32-bit-arm-cortex-mcus/stm32f4-series/stm32f407-417/stm32f407vg.html
- [19] SHENOY, Manoj. Switch Debouncing. *ElectroSome* [online]. [cit. 2016-05-16]. Dostupné z: <https://electrosome.com/switch-debouncing/>
- [20] PINO, Jose. 7-Segment ASCII character Set. *Jose Pino's Projects & Tidbits* [online]. 2009 [cit. 2016-05-16]. Dostupné z: <http://www.josepino.com/microcontroller/7-segment-ascii>

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

ms	Milisekunda, tisícina sekundy
μ s	Mikrosekunda, miliontina sekundy
embedded	Zabudovaný jednoúčelový systém
pull-up	Element v elektronickém obvodu zajišťující připojení elektrody (pinu) ke kladnému napájecímu napětí v klidovém stavu
pull-down	Element v elektronickém obvodu zajišťující připojení elektrody (pinu) k nulovému napětí (zemi) v klidovém stavu
debouncing	Ošetření tzv. bouncingu, což je zakmitávání mechanických kontaktů
R-C článek	Sériová kombinace rezistoru a kondenzátoru
log. H	Logická hodnota HIGH (true, 1)
log. L	Logická hodnota LOW (false, 0)
multiplexer	Element v elektronickém obvodu, který posílá na výstup informaci z jednoho z n vstupů zvoleného adresou
demultiplexer	Element v elektronickém obvodu, který posílá vstupní informaci na jeden z n výstupů zvoleného adresou
multiplexování	Kombinování více datových toků do jednoho signálu za účelem efektivního využití daného přenosového média
datasheet	Katalogový list, dokument uvádějící technické vlastnosti elektronické součástky
IO	Integrovaný obvod, zapouzdřená elektronická součástka
TTL	Transistor-Transistor-Logic, logika která používá napětí přibližně 5 V pro log. H a přibližně 0 V pro log. L
CMOS	Complementary Metal-Oxide-Semiconductor, logika která používá napětí přibližně 3,3 V pro log. H a přibližně 0,8 V pro log. L (korektní napětí pro použitý mikrokontrolér)

JACK	Konektor používaný pro přenos elektrického zvukového signálu
OZ	Operační zesilovač
DPS	Deska plošných spojů
clipping	Forma průběhu zkreslení, ke které dojde při přebuzení elementů obvodu signálem, který je nad maximálními možnostmi zapojení
callback	Část zdrojového kódu (většinou funkce či procedura), která je předána jako argument do jiné části kódu k provedení v určitém časovém okamžiku
float	Datový typ, číslo s pohyblivou řádovou čárkou s jednoduchou přesností
intX_t	Datový typ, znaménkové celé číslo o velikosti X bitů
uintX_t	Datový typ, bezznaménkové celé číslo o velikosti X bitů
statická proměnná	Druh lokální proměnné ve funkci, která si po ukončení funkce zachová poslední hodnotu pro úpravu chování dané funkce při jejím opětovném volání. Inicializuje se jen jednou a v případě číselného pole se inicializují všechny členy na 0
booleovská proměnná	Druh proměnné, která nabývá pouze dvou stavů – log. H a log. L. Např. v případě použití datového typu integer budou použity pouze čísla 1 a 0.
index	Kladné celé číslo označující pořadí prvku v poli. Indexování polí v programovacím jazyku C začíná číslem 0.
GPIO	General-purpose input/output, obecný pin mikrokontroléru, jehož druh chování (vstup/výstup) je ovlivnitelný uživatelem při chodu programu
MSB	Most Significant Bit, nejvýznamější bit, v binárním vyjádření čísla je to bit, který má nejvyšší hodnotu, zpravidla bit nejvíce vlevo
LSB	Least Significant Bit, nejméně významný bit, v binárním vyjádření čísla je to bit, který má nejnižší hodnotu, zpravidla bit nejvíce vpravo

buffer (zvuk)	Obecné označení pro zvukový zesilovač, který posiluje vstupní signál za účely dalšího snímání – používá se, pokud existuje reálná možnost, že při přímém snímání vstupního signálu dojde k jeho deformování
buffer (paměť)	Vyrovňovací paměť, část paměti RAM určená k ukládání dat, které posílá určité vstupní zařízení (např. A/D převodník), za účely pozdějšího znovupoužití
flag	Informační bit či posloupnost bitů, slouží k indikaci dokončení nebo změny stavu určité komponenty či funkce
string	Textový řetězec, slouží k uložení posloupnosti znaků, v programovacím jazyku C je realizován jako pole datového typu char
souběh	Nežádoucí stav při komunikaci mezi procesy (funkcemi) při kterém hrozí jejich nepředvídatelné chování, dochází k němu při špatném načasování operací
RAM	Random Access Memory, polovodičová paměť s přímým okamžitým přístupem pro čtení a zápis.
A/D	Analogově-Digitální převodník, elektronický obvod, který převádí vstupní analogovou informaci na její digitální číselnou reprezentaci
D/A	Digitálně-Analogový převodník, elektronický obvod, který převádí vstupní digitální číselnou reprezentaci informace na její analogovou formu
GCC	GNU Compiler Collection, sada překladačů různých programovacích jazyků (pro práci relevantní jazyk C a C++) vytvořených v rámci GNU (svobodný software)
FPU	Floating-Point Unit, matematický koprocessor pro vykonávání operací s čísly s pohyblivou řádovou čárkou neboli datovým typem float

SEZNAM OBRÁZKŮ

Obrázek 1 – Obecné blokové schéma digitálního zpracování [6][7].....	12
Obrázek 2 – Různé časové průběhy na výstupu LFO [12]	13
Obrázek 3 – Páskové echo Roland RE-201 Space Echo – vnitřní pohled na magnetofonové hlavy a na odkrytou pásku [10]	14
Obrázek 4 – Blokové schéma DDL [9]	15
Obrázek 5 – Blokové schéma efektu Delay [8]	16
Obrázek 6 – Blokové schéma efektu Echo	17
Obrázek 7 – Blokové schéma efektu Vibrato [8]	17
Obrázek 8 – Blokové schéma efektu Chorus [8]	18
Obrázek 9 – Blokové schéma efektu Flanger [8]	19
Obrázek 10 – Blokové schéma efektu Reverb [8]	19
Obrázek 11 – Blokové schéma efektu Tremolo [8]	20
Obrázek 12 – Blokové schéma Kompresoru dynamiky [8].....	21
Obrázek 13 – Blokové schéma Ekvalizéru [8]	23
Obrázek 14 – Schéma elektronického efektu Octaver Up. Diody 1N34 na výstupu usměrňují zvukový signál pro zvýšení frekvence signálu. [13]	24
Obrázek 15 – Ukázka zdvojnásobení frekvence vstupního signálu usměrněním.....	24
Obrázek 16 – Vývojový kit STM32F4DISCOVERY [4].....	26
Obrázek 17 – Ukázka vývojového prostředí TrueSTUDIO	28
Obrázek 18 – Ukázka návrhu elektronického schématu v softwaru EAGLE.....	29
Obrázek 19 – Ukázka návrhu DPS v softwaru EAGLE	29
Obrázek 20 – Ukázka kreslení blokového schématu v softwaru ProfiCAD.....	30
Obrázek 21 – Blokové schéma navrhovaného hardwaru.....	33
Obrázek 22 – Zákmity obecného mechanického spínače při pull-up zapojení [19].....	33
Obrázek 23 – Ukázka zapojení spínače u prvního enkodéru.....	35
Obrázek 24 – Ukázka zapojení kontaktů u druhého enkodéru	36
Obrázek 25 – Ukázka zapojení demultiplexeru – všechny vstupy a výstup pro pátý displej	37
Obrázek 26 – Ukázka zapojení anod prvního displeje k pinům mikrokontroléru	38
Obrázek 27 – Fyzické rozměry a rozmístění jednotlivých segmentů u použitého typu 7segmentového displeje [15].....	38
Obrázek 28 – Zapojení vstupní, napájecí a zpětnovazební části vstupního zesilovače	40

Obrázek 29 – Zapojení výstupu zesilovače ke vstupnímu pinu mikrokontroléru	41
Obrázek 30 – Zapojení D/A převodníku v mikrokontroléru spolu s náhradním schématem výstupní zátěže [18].....	41
Obrázek 31 – Zapojení zvukového výstupu k výstupnímu pinu mikrokontroléru	42
Obrázek 32 – Zapojení varovných LED	43
Obrázek 33 – Použité CMSIS knihovny a univerzální periferní ovladače	44
Obrázek 34 – Ukázka excelové tabulky pro grafický návrh znaků displeje.....	48
Obrázek 35 – Ukázka zapojení A/D převodníku v druhé fázi převodu [14]	49
Obrázek 36 – Propojovací redukce mono JACK 6,3mm – stereo JACK 3,5mm.....	60
Obrázek 37 – Ukázka snímání zvuku z reproduktoru komba mikrofonom	61

SEZNAM PŘÍLOH

Příloha P I: Fotodokumentace

SEZNAM ELEKTRONICKÝCH PŘÍLOH

Příloha II: Projekt EAGLE 7.4.0 – Schéma a DPS

Příloha III: Zdrojové kódy programového řešení

Příloha IV: Projekt TrueSTUDIO programového řešení

Příloha V: Nahrané audio vzorky zařízení

PŘÍLOHA P I: FOTODOKUMENTACE



Pohled shora – nahoře lze vidět zobrazovací jednotku s osmi 7segmentovými displeji, níže 3 enkodéry pro ovládání a naspodu lze vidět masivní nožní tlačítko pro zapínání/vypínání vybraného efektu



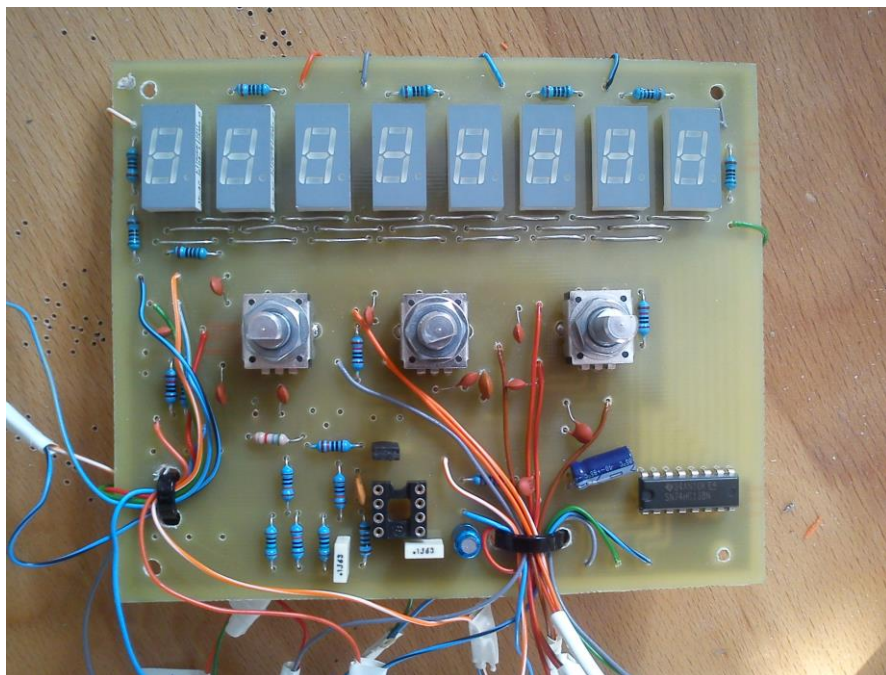
Pohled zprava na vstupní JACK a červenou LED indikující clipping vstupního A/D převodníku



Pohled zleva na výstupní JACK a červenou LED indikující clipping výstupního D/A převodníku



Pohled zezadu na napájecí CINCH



Pohled na osazenou DPS uvnitř zařízení v době ladění bez osazeného OZ, elektronické součástky vyšší než displeje a enkodéry jsou osazeny zespodu DPS



Napájecí redukce pro zařízení, USB lze připojit do portu v PC nebo do USB nabíječky