Zvukový syntezátor s mikropočítačem

Martin Třináctý

Bakalářská práce
2020

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně

Fakulta aplikované informatiky

Ústav informatiky a umělé inteligence

Akademický rok: 2019/2020

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Martin Třináctý**
Osobní číslo:	**A17635**
Studijní program:	**B3902 Inženýrská informatika**
Studijní obor:	**Softwarové inženýrství**
Forma studia:	**Prezenční**
Téma práce:	**Zvukový syntezátor s mikropočítačem**
Téma práce anglicky:	**A Sound Synthesiser with an Embedded Microprocessor**

Zásady pro vypracování

1. Analyzujte aktuální stav open source implementací pro zvukové syntezátory na embedded systémech.
2. Seznamte se s elektronickou tvorbou zvuku a protokolem MIDI.
3. Vypracujte literární reserši na dané téma.
4. Vyberte vhodný vývojový kit pro zpracování zvukového syntezátoru.
5. Navrhněte a implementujte syntezátor na zvoleném vývojovém kitu s využitím MIDI protokolu.

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

Seznam doporučené literatury:

1. HEROUT, Pavel. Učebnice jazyka C. 4., přeprac. vyd. České Budějovice: Kopp, 2004. ISBN 80-723-2220-6.
2. GUÉRIN, Robert. Velká kniha MIDI: standardy, hardware, software. Brno: Computer Press, 2004. ISBN 80-722-6985-2.
3. VÁŇA, Vladimír. ARM pro začátečníky. Praha: BEN – technická literatura, 2009. ISBN 978-80-7300-246-6.
4. UHLÍŘ, Jan a Pavel SOVKA. Číslicové zpracování signálů. Vyd. 2. přeprac. Praha: Vydavatelství ČVUT, 2002. ISBN 80-010-2613-2.
5. VÍCH, Robert a Zdeněk SMÉKAL. Číslicové filtry. Vyd. 2. přeprac. Praha: Academia, 2000. Česká matice technická (Academia). ISBN 80-200-0761-X.

Vedoucí bakalářské práce: **Ing. Tomáš Dulík, Ph.D.**
Ústav informatiky a umělé inteligence

Datum zadání bakalářské práce: **28. listopadu 2019**
Termín odevzdání bakalářské práce: **15. května 2020**

L.S.

doc. Mgr. Milan Adámek, Ph.D.
děkan

prof. Mgr. Roman Jašek, Ph.D.
ředitel ústavu

Ve Zlíně dne 9. prosince 2019

Prohlašuji, že

- beru na vědomí, že odevzdáním diplomové 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 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 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 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 vyrovnání 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 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 práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem diplomové 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 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 …….

Martin Třináctý v. r.

ABSTRAKT

Tato práce se zabývá digitálními syntezátory a možnostmi jejich open source implementací na mikropočítači. Součástí je také návrh a implementace subtraktivního syntezátoru na mikropočítači s využitím MIDI. Jsou zde popsány způsoby syntézy zvuku, komunikační MIDI protokol a prostředky a techniky pro syntézu zvuku v digitální podobě. Také je zde průzkum open source knihoven pro realizaci syntezátorů s mikropočítači.

Klíčová slova: syntezátor, zvuk, syntéza, MIDI, mikropočítač, STM32F4, signál, filtr, open-source,

ABSTRACT

This thesis deals with digital synthesizers and possibilities of open source implementation of synthesizers with microcomputers. Part of this thesis is also design and implementation of subtractive synthesizer on microcomputer with usage of MIDI. There is a description of sound synthesis methods, MIDI protocol and techniques and resources for sound synthesis in digital way. Also there is a research of open source libraries for realizations of synthesizers with microcomputers.

Keywords: synthesizer, sound, synthesis, MIDI, microcomputer, STM32F4, signal, filter, open-source

Chtěl bych poděkovat panu Ing. Tomášovi Dulíkovi, Ph.D. za umožnění vypracovat bakalářskou práci na toto téma, její vedení a rady při jejím vypracování. Dále bych chtěl poděkovat mé rodině za inspiraci a podporu ve studiu.

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

OBSAH

ÚVOD

Zvukové syntezátory jsou dnes součástí prakticky každé zvukové produkce a denně je slýcháme ať už jako vůdčí nástroj v písni nebo jako zdroj zvukových ruchů při napínavé scéně ve filmu. Zvuk je zde generován elektronickou cestou, syntezátory lze dnes rozdělit do dvou hlavních kategorií, a to na analogové a digitální.

Původně analogové nástroje vznikaly především za účelem simulovat jinak těžce dostupné nebo nepřenosné hudební nástroje, jako například varhany, nebo celé hudební formace jako orchestr.

Na počátku 70. let 20. století vytvořil Robert Moog analogový syntezátor, který umožňoval úpravy parametrů generovaného zvuku, jako například změnit druh signálu generovaného zvuku a dále jej upravit pomocí filtru nebo obálkového generátoru.

Syntezátor lze také rozdělit na jednotlivé části, jako generátor zvuku či zmiňovaný filtr a další, takzvané moduly, z nichž lze následně tvořit nepřeberné množství nových hudebních nástrojů. V 80. letech s příchodem digitálních technologií se objevily nové způsoby syntézy zvuku, jako frekvenční modulace, ale také vznikl komunikační protokol MIDI, který nabízel nové možnosti v komunikaci mezi zvukovými zařízeními a jejich ovládání.

I přesto, že je dnes možné sehnat téměř jakýkoliv syntezátor v softwarové podobě, roste komunita lidí, kteří dávají přednost mnohdy i nedokonalejší verzi, za to ale s možností mít syntezátor v ruce a ovládat jej přímo, namísto pouhého pohybu myší. Proto je právě tato práce věnována tvorbě digitálního syntezátoru s mikropočítačem, který dá uživateli takové možnosti.

I. TEORETICKÁ ČÁST

1 ELEKTRONICKÁ TVORBA ZVUKU

Tvorba elektronického zvuku a nástroje, zvané syntezátory, ve kterých vzniká, se dá rozdělit na dva druhy, podle vzniku a zpracování signálu, na analogovou a digitální. Oba druhy nabízí velké množství výhod i nevýhod, princip vzniku a práce se zvukovým signálem jsou ale velmi podobné a liší se především v prostředcích, pomocí kterých jsou realizovány. [1]

Zásadní význam pro elektronickou tvorbu zvuku spočívá ve znalosti obecných zákonitostí hudebních signálů. Zvukový signál lze charakterizovat vnímáním frekvence, amplitudy a témbru (kvalita zvuku, která charakterizuje tón podle původu jeho vzniku) v daném časovém okamžiku. Tyto vjemové faktory se odvíjí od tvaru zvukového signálu. [1]

1.1 Základní zvukové signály

Zvukové signály můžeme rozdělit do dvou základních kategorií na hudební a nehudební. Hudebním signálem je ten, který vzniká s periodickým kmitáním. Při poslechu lze určovat výšku, frekvenci signálu. Jeho zdrojem je zpravidla hudební nástroj, nebo lidské hlasivky.

Nehudebním signálem je hluk. Hluk vzniká neperiodickým kmitáním a jeho zdrojem může být prakticky cokoliv, náraz dvou těles, výstřel apod.

1.2 Základní tvary signálu

Tvar signálu vyjadřuje tvar jednoho kmitu, tvořený změnou výchylky signálu v čase. Zdrojem těchto signálů jsou zvukové generátory zvané oscilátory. Nejběžnějšími tvary jsou sinusový, pilový a obdélníkový signál.

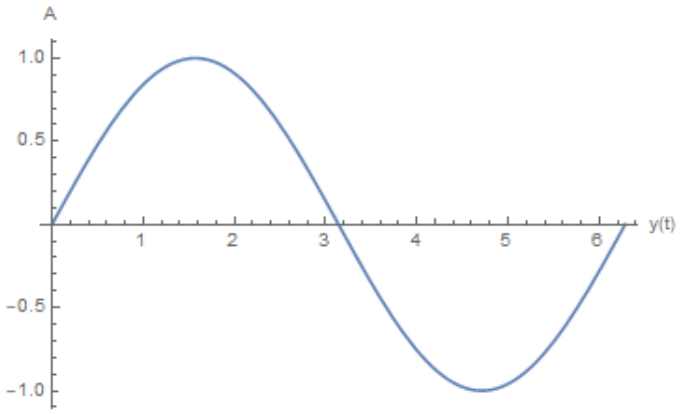
1.2.1 Sinusový signál

Sinusový signál je tvořen pouze základní frekvencí, neobsahuje žádné vyšší harmonické frekvence. Díky tomu zní velmi jemně a nerušivě. Jedná se o nejzákladnější tvar signálu. Pomocí harmonické analýzy lze ze sinusového kmitu skládáním vytvořit všechny další tvary.

Sinusový kmit je vyjádřen následující rovnicí.

$$y(t) = A\sin(\omega t + \varphi) \tag{1.2.1.1}$$

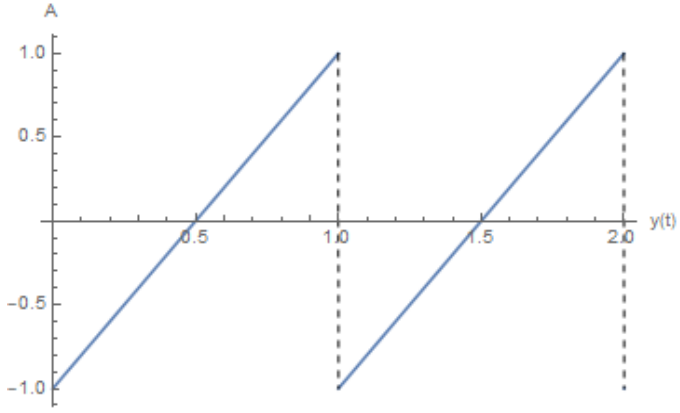
Kde $y(t)$ je okamžitá výchylka v čase, A je amplituda signálu, ω je úhlová frekvence a φ je počáteční fáze. [2]



Obrázek 1 Tvar sinusového signálu

1.2.2 Pilový signál

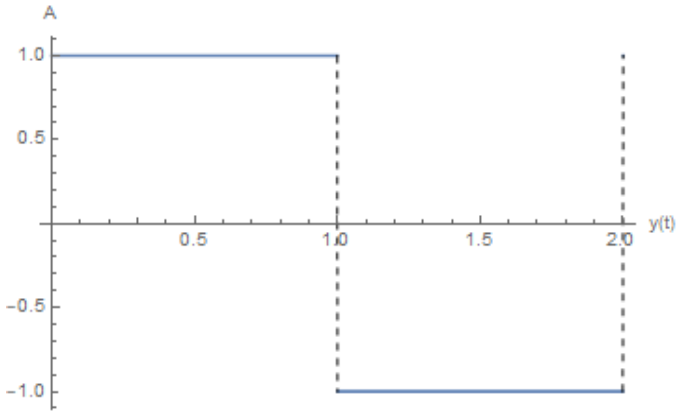
Tvar tohoto signálu nejprve lineárně stoupá, po dosažení maximální amplitudy téměř okamžitě klesne na její minimální hodnotu. Pilový signál obsahuje sudé i liché harmonické frekvence. Zní poměrně ostře a syrově oproti sinusovému kmitu. Díky tomu je velmi často použit při subtraktivní syntéze. [2]



Obrázek 2 Tvar pilového signálu

1.2.3 Obdélníkový signál

Obdélníkový signál kmitá mezi dvěma úrovněmi amplitudy, které se pravidelně střídají. Obsahuje pouze sudé harmonické frekvence. Zní dutě, až kovově a je také velmi hojně využíván při subtraktivní syntéze. V některých případech lze měnit poměr mezi první a druhou úrovní v čase, v tom případě se jedná o pulsně šířkovou modulaci. [2][3]



Obrázek 3 Tvar obdelníkového signálu

1.3 Metody zvukové syntézy

Zvuková syntéza je libovolný proces, který vede ke vzniku předvídatelného i nepředvídatelného zvuku. Existuje velké množství zvukových syntéz, navíc je lze kombinovat.

1.3.1 Aditivní syntéza

Aditivní, česky též součtová syntéza, je jeden z nejstarších způsobů syntézy zvuku, kromě toho je také považována za nejdokonalejší metodu zvukové syntézy, protože její princip odpovídá zvukům slyšitelných v běžném světě, ať už se jedná o hudební zvuky (zvuk piana, zpěv ptáků), nebo nehudební (šplouchnutí vody, šustění listí).

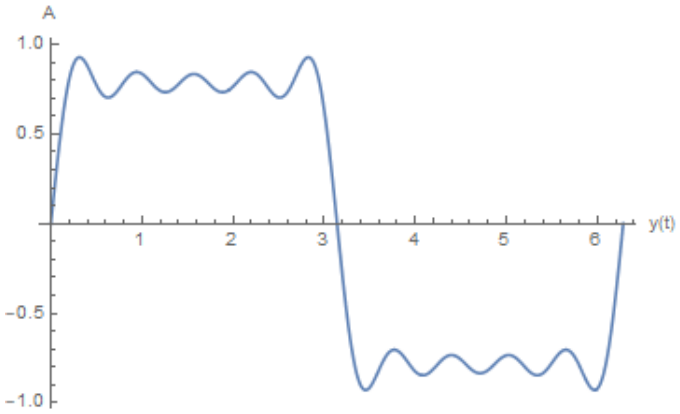
Takové zvuky jsou tvořeny sumací sinusových vln. Výšku tónu určuje fundamentální frekvence, nejnižší v sumaci, ostatní jsou vyšší harmonické, které tvoří témbr.

Aditivní syntéza je vyjádřena v následující rovnici. [4][5]

$$f(t) = \sum_{k=1}^{N} A_k \sin(2\pi f_k t + \varphi_k) \tag{1.3.1.1}$$

Kde N je počet složek harmonických frekvencí, k je n-tá složka frekvence, A_k je amplituda dané složky, f_k je její frekvence a φ_k fázový posun. První frekvence je fundamentální.

Pomocí aditivní syntézy lze vytvořit klasické signály jako obdélníkový, sčítáním lichých násobků fundamentální frekvence. Pro vytvoření pilového signálu se sečtou sudé i liché násobky.



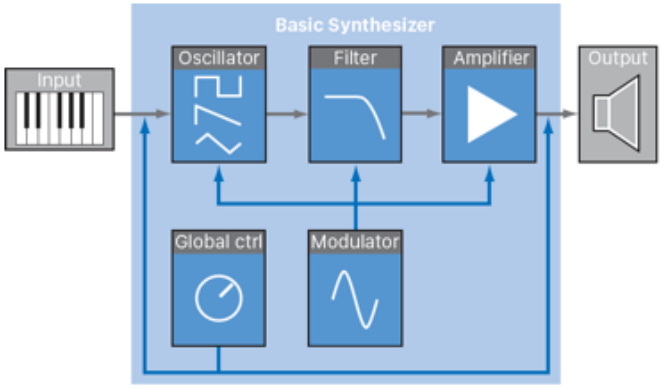
Obrázek 4 Čtvercový kmit vytvořený pomocí aditivní syntézy

1.3.2 Subtraktivní syntéza

Subtraktivní syntéza je pravděpodobně nejrozšířenější. Je složena z generátoru zvuku a modulačních prvků, které ze signálu odstraňují určité prvky.

Generátor zvuku produkuje základní signály jako sinus, pila a obdélník.

Mezi modulační prvky patří především frekvenční filtr, zpravidla dolní propust, která propouští jen frekvence pod stanovenou frekvenci a dále obálkový generátor, který upravuje náběh, útlum, podržení a doznívání signálu. Parametry modulačních prvků mohou být ovládány nízkofrekvenčním oscilátorem s různými tvary signálu. [3][4]

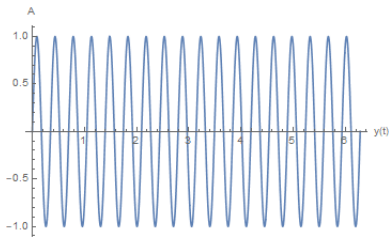


Obrázek 5 Schéma subtraktivní syntézy [6]

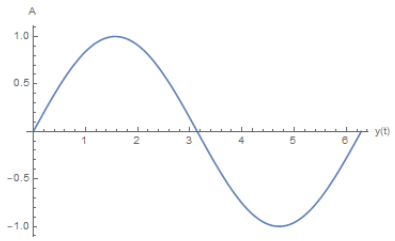
1.3.3 Frekvenčně modulační syntéza

Jak již napovídá název, základem frekvenčně modulační syntézy je modulace kmitočtu nosné frekvence, kmity o jiné frekvenci, obvykle nižší než je nosná frekvence. Oscilátory generující signály se nazývají operátory, jsou potřeba minimálně dva, běžný počet je ale šest. Operátory lze mezi sebou kombinovat, kombinace se nazývá algoritmus. Tvar kmitání operátoru je běžně sinusový. [1]

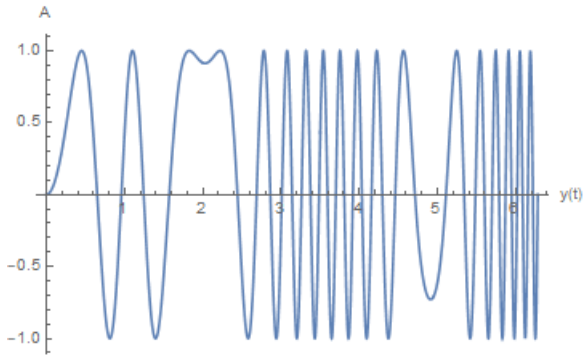
Syntéza umožňuje plynulý přechod mezi harmonickými a neharmonickými strukturami rozladěním nosné frekvence. [5]



Obrázek 6 Nosná frekvence



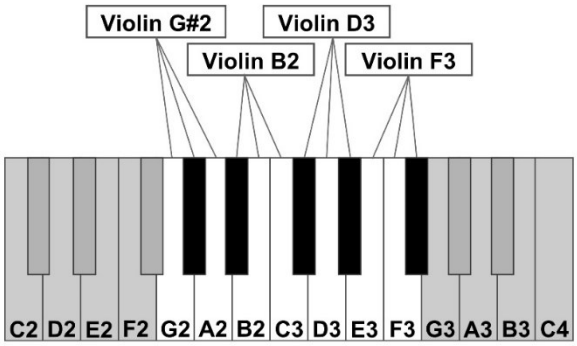
Obrázek 7 Modulační frekvence



Obrázek 8 Výsledný tvar FM syntézy

1.3.4 Samplovací metoda

V tomto případě se nejedná přímo o vytváření nového signálu, ale o přehrávání předem vytvořených a uložených zvuků. V paměťovém uložišti jsou tak uloženy samply, vzorky, například klavíru, kdy pro nejlepší kvalitu výsledného zvuku je zaznamenán každý tón, který je následně přiřazen v syntezátoru odpovídající klávese. Lze ale také použít jeden vzorek, který pokryje více tónů, díky úpravě rychlosti přehrávání vzorků. [5][7]



Obrázek 9 Rozložení čtyř vzorků přes dvanáct kláves [8]

1.4 Historie digitálních syntezátorů

Digitální syntezátory se začaly objevovat ve světě na konci 70. let minulého století. Především díky nárůstu ve výkonu tehdejších technologií. Hnacím motorem pro digitální syntezátory byla také velká náročnost, či dokonce nemožnost vytvořit jiné druhy syntézy pomocí analogových syntezátorů, hlavně aditivní a frekvenčně modulační syntézy.

1.4.1 Synclavier

Tento digitální syntezátor vyšel poprvé v roce 1977, o tři roky později byla vydána nejrozšířenější verze. Pracoval na principu aditivní a frekvenčně modulační, později jej bylo možné rozšířit o samplování, pomocí speciálních desek. Generování signálu bylo pomocí osmibitových sinusových signálů. Samplovaný signál byl zaznamenáván v šestnácti bitovém formátu. Ačkoliv reálně byl syntezátor monofonní, šlo hlas uložit do paměti a nechat jej přehrávat, bylo tak možné uložit až 64 hlasů. Kromě toho také syntezátor disponoval efektovým procesorem. Úprava zvuku se prováděla pomocí panelu s tlačítky pro zvolený parametr a jednoho rotačního enkodéru. Pozdější verze byly také doplněny o displej. Cena tohoto syntezátoru v osmdesátých letech se pohybovala okolo dvou stovek tisíc dolarů. [9][10]

Obrázek 10 Synclavier [11]

1.4.2 Yamaha DX7

Tento digitální syntezátor se objevil v roce 1983. Velkou oblibu si získal především díky schopnosti simulovat různé klavíry, varhany, perkusní zvuky a další. Velmi oblíbené a využívané jsou například také zvuky připomínající zvony. Jeho základem jsou zvukové čipy obsahující FM operátory, generující sinusový signál. V DX7 je jich celkem šest. Ty se mezi sebou seřadí podle zvoleného algoritmu, kterých je zde dostupných 32. K tomu jsou sinusové signály ovlivněny zpětnou vazbou, sofistikovaným obálkovým generátorem, který oproti subtraktivní syntéze neřídí amplitudu, nízkofrekvenčním oscilátorem a dalšími parametry. Tyto čipy byly také využívány v 90. letech ve zvukových kartách pro počítače. Celkově má syntezátor 145 parametrů pro editaci zvuku. Ve dvou režimech lze používat syntezátor s 16 hlasy, nebo jako monofonní. U tohoto syntezátoru se také poprvé objevil protokol MIDI, ačkoliv používal oproti standartnímu rozsahu 0-127 pouze rozsah 0-99. Nevýhodou také byl v prvních verzích použitý 10 - ti bitový D/A převodník. Přesto se svou cenou okolo dvou tisíc dolarů získal velkou oblibu a je dodnes používán a také se dočkal spousty reedicí, ať už od firmy Yamaha nebo dalších. [12]

Obrázek 11 Yamaha DX7 [13]

1.4.3 Waldorf Blofeld

Tento syntezátor je jedním z novějších zástupců, na trh byl uveden v roce 2008. Je založen na digitální syntéze zvuku pomocí vyhledávací tabulky. Zvuk je zde tvořen třemi oscilátory, každý z nich nabízí klasické signály jako pila, obdélník, trojúhelník a další, kromě toho také lze u prvních dvou oscilátoru vybírat z dalších 68 různorodých signálů. Dále je zde také možnost modulací pomocí nízkofrekvenčních oscilátorů nebo obálkových generátorů, které se můžou modulovat navzájem pomocí smyček. Kromě toho Blofeld nabízí také několik druhů filtru, simulaci analogového filtru použitého u prvního syntezátoru využívajícího vyhledávací tabulku, PPG Wave. Součástí je také efektový procesor a sekvencér. [14]

Obrázek 12 Waldorf Blofeld [15]

2 MIDI PROTKOL

MIDI, Music Instrument Digital Interface, je digitální komunikační protokol vytvořený pro komunikaci mezi softwarovými, hardwarovými hudebními nástroji a jejich kontroléry. Základní myšlenkou tohoto protokolu je dálkové ovládání hudebních nástrojů, kdy hudebník může například ovládat více nástrojů z jednoho kontroléru, či použití sekvencéru, který obsahuje. Ačkoliv je tento standard téměř 40 let starý, je stále využíván a objevuje se v nových produktech ve stále stejném provedení.

2.1 Struktura MIDI

MIDI zprávy jsou složeny ze skupin 8 bitových slov - bytů, které jsou sériově přenášeny. Existují dva druhy MIDI bytů, stavový a datový. MIDI zprávy jsou přenášeny sériově rychlostí 31250 baudů. Přenáší se ve skupině jeden stavový byte a dva datové byty. [16]

Tabulka 1 Příklad MIDI zprávy Note on, pro notu 64 s rychlostí stisku 89

	Stavový byte	**Datový byte 1**	**Datový byte 2**
Popis	Status/kanál	Nota	Rychlost stisku
Binární data	10010000	01000000	01011001
Hodnota	Note on/CH 1	64	89

2.1.1 Stavový byte

Stavový byte se používá pro identifikaci funkce MIDI zprávy a identifikaci kanálů, pro který je zpráva určena. První čtyři bity zprava jsou rezervovány pro určení kanálu, což znamená, že je možné posílat MIDI zprávy až na šestnáct kanálů. Další tři bity jsou stavové a určují funkci, kterou má MIDI zpráva provádět. Nejčastěji se jedná o Note on (stisknutí klávesy), Note off (puštění klávesy), nebo CC zprávu. Poslední bit odlišuje stavový byte od datového tím, že je nastaven na hodnotu 1. Jedná se o takzvaný nejvýznamnější bit (bit nejvíce vlevo). [16]

2.1.2 Datový byte

Datový byte přenáší hodnotu dané zprávy. Nejvýznamnější bit je nastaven na hodnotu 0, což dává prostor pro hodnotu 127, jako maximální hodnotu dat. Při funkci Note on stavového bytu lze tak zahrát 128 různých not, spolu se 128 různými rychlostmi stisku. Při funkci PitchBend, změny výšky tónu lze ale hodnoty datových bytů spojit pomocí bitových operací a získat tak rozsah až 16384 hodnot při 14 bitovém rozlišení.

2.2 Druhy MIDI zpráv

MIDI zprávy lze rozdělit do tří kategorií, podle zaměření funkcí na hlasové, systémové a realtimové.

2.2.1 Hlasové MIDI zprávy

Hlasové MIDI zprávy jsou ty, které přímo ovládají zvuk produkovaný zařízením, které je ovládáno MIDI.

Tabulka 2 Hlasové MIDI zprávy[17]

Zpráva	**Aktivita MIDI jednotky**
Note off	Nota byla puštěna a měla by přestat znít
Note on	Nota byla stisknuta a měla by začít znít
Aftertouch / key pressure	Tlak aplikovaný na stisknutou notu
Controller	Mění hodnotu kontroléru, např. potenciometru, slideru
Program change	Přiřazení programu MIDI kanálu, programem je myšlen nástroj, patch, preset
Channelpressure	Tlak aplikovaný na noty celého kanálu
Pitch wheel / modulation wheel	Změna výšky tónu

2.2.2 Systémové MIDI zprávy

Systémové MIDI zprávy se nevztahují ke kanálu, ale k MIDI jednotce, se kterou MIDI jednotka komunikuje. Neočekává se ale okamžitá odezva.

Tabulka 3 Systémové MIDI zprávy[18]

Zpráva	**Aktivita MIDI jednotky**
Systemexclusive (sysex)	Provádí specifickou aktivitu (nahrání zdrojového kódu přes MIDI)
Quarterframe	Synchronizace s jinou MIDI jednotkou
Song position pointer	Ukazatel na připravenou sekvenci k přehrání
Song request (song select)	Nastavení sekvence k přehrání
Tune	Naladění

2.2.3 MIDI zprávy reálného času

MIDI zprávy reálného času se taktéž nevztahují ke kanálu, ale k MIDI jednotce, očekává odpověď od jednotky v reálném, určitém čase. Můžou být umístěny mezi jiné MIDI zprávy, bez toho aniž by je ovlivnily. Jedná se o časové zprávy, které určují čas spuštění, zastavení MIDI sekvence, nebo synchronizace s dalším zařízením, které udává tempo. [8-19]

Tabulka 4 MIDI zprávy reálného času[19]

Zpráva	**Aktivita MIDI jednotky**
MIDI clock	Vyrozumí MIDI jednotku o pozici MIDI hodin, pokud je synchronizována s jinou
Start	Spustí přehrávání MIDI sekvence
Continue	Pokračování přehrávání MIDI sekvence
Stop	Zastaví přehrávání MIDI sekvence
Activesense	Vyrozumí MIDI jednotku o existující MIDI připojení (pokud nejsou žádné jiné MIDI zprávy)
Reset	Resetuje do původního stavu

3 SYNTEZÁTORY S MIKROPOČÍTAČEM

Syntezátory s mikropočítači využívají pro syntézu zvuku techniky číslicového zpracování signálu. Pomocí těchto technik se vypočítá vzorek signálu, který je následně pomocí D/A převodníku převeden na analogový signál, který putuje dále buďto do reproduktoru, nebo k dalšímu zpracování.

3.1 Číslicové zpracování signálů

Číslicové zpracování signálu je matematická a algoritmická manipulace se signály za účelem jejich zpracování a získání určitých informací.

Signály jsou funkce o jedné nebo více nezávislých proměnných, které přenáší informace o svém zdroji, nebo informace v něm zakódované. Systémy jsou pak určeny k tomu, aby se signálem pracovaly, například jej transformovaly a vytvořily nový signál. Za systém tak můžeme považovat syntezátor, který generuje různé průběhy signálu a dále na tyto signály aplikuje například filtr dolní propusti, případně některý z audio efektů. [20]

3.2 Výpočet tvaru signálu

Existuje několik variant výpočtu průběhu signálu, z nichž každá poskytuje výhody i nevýhody. Výběr je ovlivněn faktory jako výpočetní rychlost mikropočítače, či velikost paměťových uložišť.

3.2.1 Přímý výpočet signálu

Téměř všechny vyšší programovací jazyky poskytují funkci pro výpočet hodnoty sinus, což se přímo nabízí jako nejzřetelnější metoda pro generování sinusového signálu. Obecně lze výpočet n-tého vzorku sinusového signálu vyjádřit následující rovnicí s použitím běžně vestavěné funkce sin. [21]

$$S(n) = A\sin(2\pi f n/F_s) \tag{3.2.1.1}$$

Kde f je frekvence, A je amplituda, n je n-tý vzorek, F_s je konstanta vyjadřující vzorkovací frekvenci ve formě počet vzorků za sekundu. Pro ušetření výpočetního času lze vytvořit konstantu. [21]

$$K = 2\pi/F_s \tag{3.2.1.2}$$

A následně ji použít ve funkci:

$$S(n) = A\sin(Knf) \tag{3.2.1.3}$$

Vzorky se vypočítají dosazením do proměnných A, f a následným iterováním od 0 až po M, kde M ve vztahu:

$$M/F_s \tag{3.2.1.4}$$

vyjadřuje dobu trvání signálu. [21]

Pro další signály nejsou obvykle poskytovány funkce, jejich realizace je ale poměrně jednoduchá. Pro výpočet pilovitého průběhu lze použít rovnici. [21]

$$S(n) = A\left[2\left(\frac{\mathrm{nF}}{F_s}\%\ 1.0\right) - 1\right] \tag{3.2.1.5}$$

Kde operace % (modulo) vrací zbytek po dělení výrazu nalevo výrazem napravo. Z tohoto průběhu lze vytvořit trojúhelníkový průběh podle následující rovnice. [21]

$$S(n) = 2A\left[\left|S_{n_{pila}}\right| - \frac{S_{n_{pila}}}{2}\right] \tag{3.2.1.6}$$

3.2.2 Vyhledávací tabulka

Přestože je předchozí metoda poměrně jednoduchá a přímočará, je časově náročná na výpočet. Za předpokladu, že průběh se příliš neliší mezi jednotlivými frekvencemi, můžeme jeden kmit průběhu uložit do tabulky a posléze pomocí specifických vyhledávacích technik získávat vzorky. [21]

Získávání vzorků probíhá kruhovým procházením tabulky ve smyčce, při překročení nejvyšší pozice v tabulce je nutné, aby získávání vzorku probíhalo opět od pozice, která je ve vzdálenosti o délku tabulky od pozice, která překročila nejvyšší možnou pozici.

Pro získání určité frekvence je potřeba tabulku procházet určitou rychlostí, která se mění podle požadované frekvence. To by vyžadovalo změnit vzorkovací frekvenci, která je ale konstantní. Namísto toho se tedy změní velikost kroku, s jakým se prochází tabulka. To znamená, že jsou vynechány některé vzorky z tabulky. Kvůli tomu vznikne zkreslení signálu, kdy se harmonické složky vyšší než vzorkovací frekvence vrátí zpět do signálu jako nižší frekvence. Tento jev nastává především u komplexnějších signálů, jako pilového a obdélníkového, které jsou v ideální formě tvořeny nekonečnými násobky harmonických frekvencí a projevuje se u vysokých tónu. [21]

Zkreslení lze odstranit několika způsoby, některé metody jako BLIT jsou ale výpočetně náročnější. Poměrně dostačující jsou metody, kdy je vytvořeno několik tabulek, obvykle minimálně jedna pro oktávu, které obsahují určitý počet harmonických složek signálu. Tón,

který lze přehrávat s krokem, který obsáhne všechny vzorky v tabulce, má přiřazenou tabulku s nejvyšším počtem harmonických složek, v ostatních se tabulkách se potom počet harmonických složek snižuje.

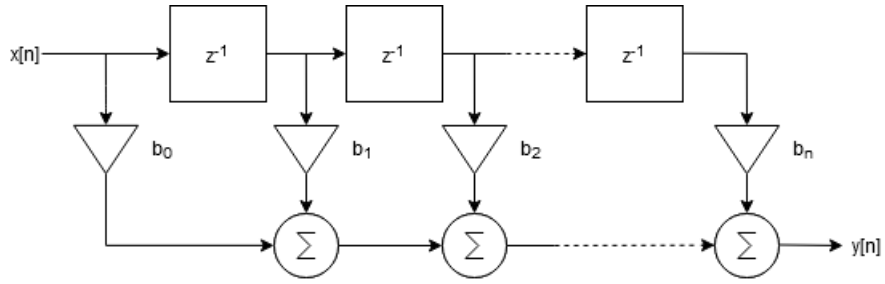
Metoda „přechodové rozdělení“ má tabulku, ve které je vygenerovaný přechod s nízkým počtem harmonických složek. Při výběru vzorku z tabulky s původním signálem se detekuje pozice v tabulce a před přechodem mezi posledním a prvním vzorkem z tabulky se nahradí tyto vzorky odpovídajícím vzorky z přechodové tabulky. [22]

3.3 Číslicové filtry

Číslicové filtry představují jednu z možných realizací systémů číslicového zpracování signálů. Úkolem číslicových filtrů je požadovaným způsobem ovlivnit frekvenční spektrum původního signálu. Například lze vybrat část frekvenčního spektra, které zůstane zachováno a ostatní frekvence odfiltrovat. Kromě klasických aplikací jako filtr dolní propust, horní propust, pásmová propust lze způsobem podobným číslicovým filtrům realizovat také různé efekty využívající zpožďovací linky. [23]

3.3.1 Filtry s konečnou impulzní odezvou

Číslicové filtry lze dělit na dva základní druhy, prvním z nich jsou filtry s konečnou impulzní odezvou. Běžně jsou označovány jako FIR filtry, tyto filtry neobsahují zpětnovazební smyčky, jsou stabilní, ale také mají velké prostorové nároky pro ukládání koeficientů a velké nároky pro jejich výpočet. Pro dosažení kvalitních výsledku je také nutné použít vyšší počet koeficientů (přenosový řád), kvůli vyššímu řádu dochází také k většímu zpoždění při zpracování vstupního signálu. [23]

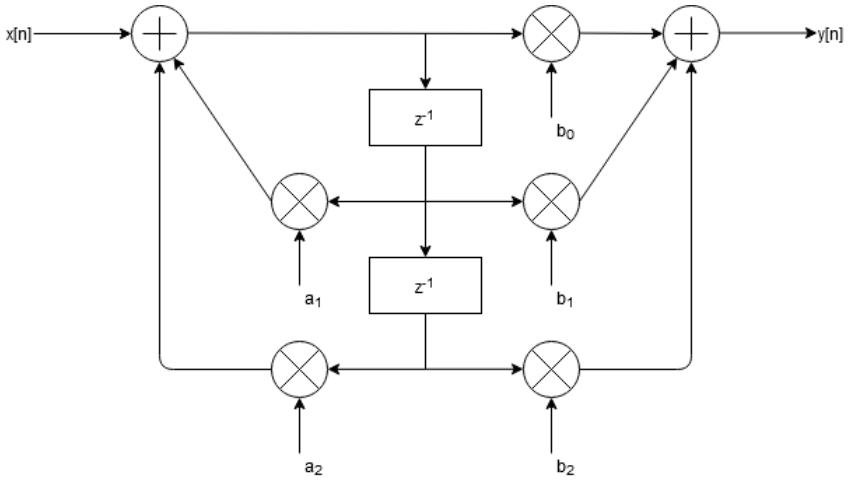


Obrázek 13 Schéma filtru s konečnou impulzní odezvou n-tého řádu

3.3.2 Filtry s nekonečnou impulzní odezvou

Dalším druhem jsou pak filtry s nekonečnou impulzní odezvou, označovány jako IIR filtry. Pro realizaci IIR filtrů je nutné použít zpětnovazebních smyček, oproti FIR filtrům dosahují

IIR filtry lepších výsledků při nižších přenosových řádech, což také zaručuje menší zpoždění. Díky tomu jsou IIR filtry často využívány v syntezátorech a podobných aplikacích, kde je vyžadována, pokud možno okamžitá odezva. Také k nim lze nalézt analogové ekvivalenty. Oproti FIR filtrům mohou mít problém se stabilitou. [23]



Obrázek 14 Schéma filtru s nekonečnou impulzní odezvou druhého řádu

3.4 Vzorkovací frekvence

Vzorkovací frekvence udává počet vzorků za vteřinu. Podle Nyquist-Shannonova teorému by vzorkovací frekvence měla být dvakrát vyšší než nejvyšší frekvence v signálu.

Pro syntezátory se používá vzorkovací frekvence obvykle mezi 44,1 kHz a 48 kHz

Při použití frekvence 48 kHz znamená, že nejvyšší frekvence v audio signálu může být 24 kHz. Tato frekvence se nazývá Nyquistova. To znamená, že je pokryta slyšitelnost zvuku, která se pohybuje od 16Hz do 20 kHz. [24]

Problémem může být ale takzvaný aliasing. Aliasing je jev, při kterém je jistá frekvence reprezentována jinou, například při vzorkovací frekvenci 48 kHz, sinusový tón o frekvenci 15 kHz bude vzorkován pouze třemi vzorky, což může být zaměněno s jinou frekvencí. [24]

Řešením je filtr dolní propusti, který odfiltruje vysoké frekvence způsobující aliasing.

3.5 Bitová hloubka

Bitová hloubka je parametr udávající maximální hodnotu vzorku nebo také rozlišení D/A převodníku. Čím je vyšší hodnota bitové hloubky, tím přesněji lze vyjádřit hodnotu vzorku.

V audio aplikacích se běžně používá bitová hloubka mezi 16 až 24 bity, někdy až 32 bitů, při takovém rozlišení už téměř nelze postřehnout rozdíl z hlediska posluchače, proto se používá především pro další zpracování a ne pro finální výstup. [24]

3.6 Open Source implementace zvukových syntezátorů

V dnešní době se lze setkat se spoustou dostupných open source implementací zvukových syntezátorů. Již delší dobu není nutné využívat jazyka symbolických adres pro programování mikropočítačů, což spolu s jejich rozšířením mezi veřejnost umožnilo vznik knihoven právě pro implementaci zvukových syntezátorů nebo jim přidružených MIDI kontrolérů a dalších zařízení.

3.6.1 MIOS MIDIbox Operating System

Knihovna vznikla v reakci na požadavek flexibilních MIDI aplikací, jako jsou MIDI kontroléry, procesory, syntezátory, sekvencéry. Oproti komerčním řešením poskytuje otevřené řešení jak po stránce hardwarové, tak i softwarové. [25]

MIOS používá vlastní operační systém reálného času, založený na Free RTOS, což umožňuje provádění více operací najednou s různou prioritou. Existuje ve dvou verzích, MIOS8, který pracoval na 8 bitových procesorech PIC. Valná většina projektů je napsána v jazyce symbolických adres a dnes již nejsou další projekty vyvíjeny.

Druhá verze je MIOS32, který pracuje na 32 bitových procesorech STM32. Podpora USB MIDI v této verzi umožňuje rychlé nahrání aplikace do MIOS32 pomocí sysex MIDI zpráv. Je potřeba do mikropočítače nahrát také zavaděč. [25]

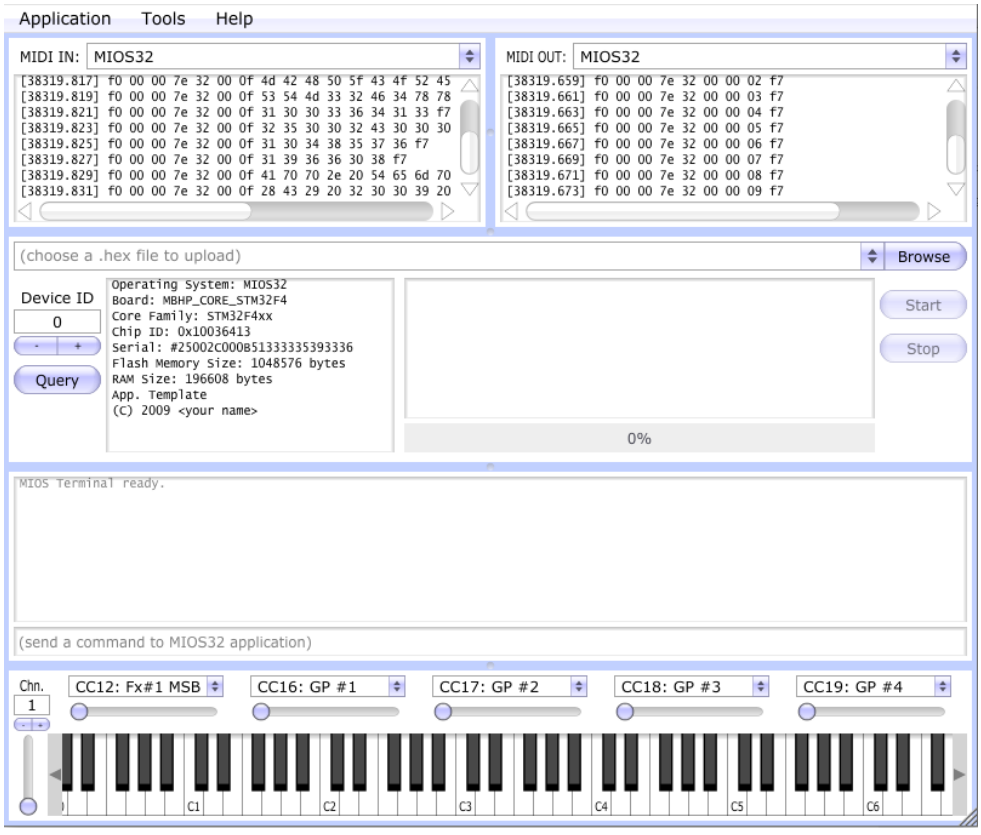
Přes to, že je knihovna zaměřena spíše na MIDI kontroléry, některé podporované mikropočítače poskytují přímo na vývojové desce D/A převodník s výstupním audio konektorem, například STM32F4 Discovery. To poskytuje ideální možnosti pro vývoj syntezátoru, přes USB lze vysílat MIDI zprávy a naopak získávat debugovací zprávy v MIDI monitoru a vyvíjet tak syntezátor bez připojení jakýchkoliv dalších externích periferií.

Z toho vyplývá, že knihovna poskytuje především kvalitní ovladače pro periferie na mikropočítači a dále funkce pro zpracování MIDI, časové generátory pro využití v sekvencérech a dalších. Funkce pro syntézu zvuku je nutné implementovat, zde lze narazit na jistou limitaci, protože i přes možnou přítomnost FPU na vývojové desce, MIOS používá starší verzi Free RTOS, kvůli přenositelnosti na více mikropočítačů, která neumožňuje použití FPU. Kvůli tomu jsou výpočty s datovým typem float výrazně pomalejší a projeví se

především na výsledném zvuku. Řešením je použití pevné desetinné čárky a celočíselných datových typů pro výpočty.

Pro psaní aplikací pro MIOS32 se používá jazyk C nebo C++, které je možné kombinovat s jazykem symbolických adres.

Nahrání aplikací probíhá přes MIOS Studio, což je platformě nezávislé prostředí pro zpracování MIDI, umožňující nahrávání aplikací pomocí sysex MIDI zpráv do mikropočítače. Kromě toho poskytuje také MIDI monitory, virtuální MIDI klávesy s možností posílání MIDI CC zpráv a MIOS terminál, přes který lze komunikovat s aplikací pomocí příkazů implementovaných v aplikaci nebo zobrazovat zde ladící zprávy z aplikace. [26]



Obrázek 15 MIOS studio 2

Aplikace využívající tuto knihovnu jsou především kontroléry. Především ty, které pouze pracují s MIDI zprávami, jedním z nich je MIDIbox SEQ V4, sekvencér určený pro živé hraní, poskytující velké možnosti ovládání syntezátorů pomocí MIDI zpráv. K sekvencéru lze připojit až 12 syntezátorů.

Syntezátorové aplikace v této knihovně jsou především ovladače pro samotné čipy, produkující syntézu zvuku. Ty pak dále implementují ovládací prvky umožňující měnit

parametry čipu a ovládat syntézu zvuku. Jedná se o čipy zvukových karet z 90. let, například Yamaha YMF262, který pracuje na principu FM syntézy, nebo SID čipy používané v počítačích Commodore 64. Pro desku STM32F4 Discovery, která obsahuje výstupní audio konektor, existuje aplikace Goom, která byla přenesena na MIOS ze staršího projektu používající mikropočítač LPC1343. Poskytuje čtyři různé průběhy, kombinaci mezi nimi, filtr dolní propusti a obálkové generátory. [27]

Knihovna také poskytuje hardwarovou podporu pro mikrokontrolery, konkrétně rozšiřující desku pro vývojové desky mikrokontrolerů, na kterou je možné připojit další rozšiřující moduly. Například modul pro LCD displeje, ke kterému lze připojit jednoduché alfanumerické displeje, grafické displeje a dotykové displeje. Další moduly umožňují připojení rozšiřujících vstupů a výstupů pro MIDI, modul pro analogové vstupy dovolí připojit až 64 potenciometrů, modul s D/A převodníkem a audio výstupem.

Všechny zdrojové kódy i schémata a návrhy plošných spojů pro moduly jsou volně dostupné, nejsou ale určené pro komerční účely, přesto autoři umožňují, speciálně u oficiálních projektů jako je MIDIbox SEQ V4 pod licencí TAPR NCL vyrobit a prodat 10 výrobků používajících MIOS aplikace a moduly.

3.6.2 Teensy Audio Library

Tato knihovna je vytvořena přímo tvůrci řady mikropočítačů Teensy. Mikropočítače Teensy řady 2 používají 8 bitové procesory AVR, řada 3 používá výkonnější 32 bitové procesory ARM. [28]

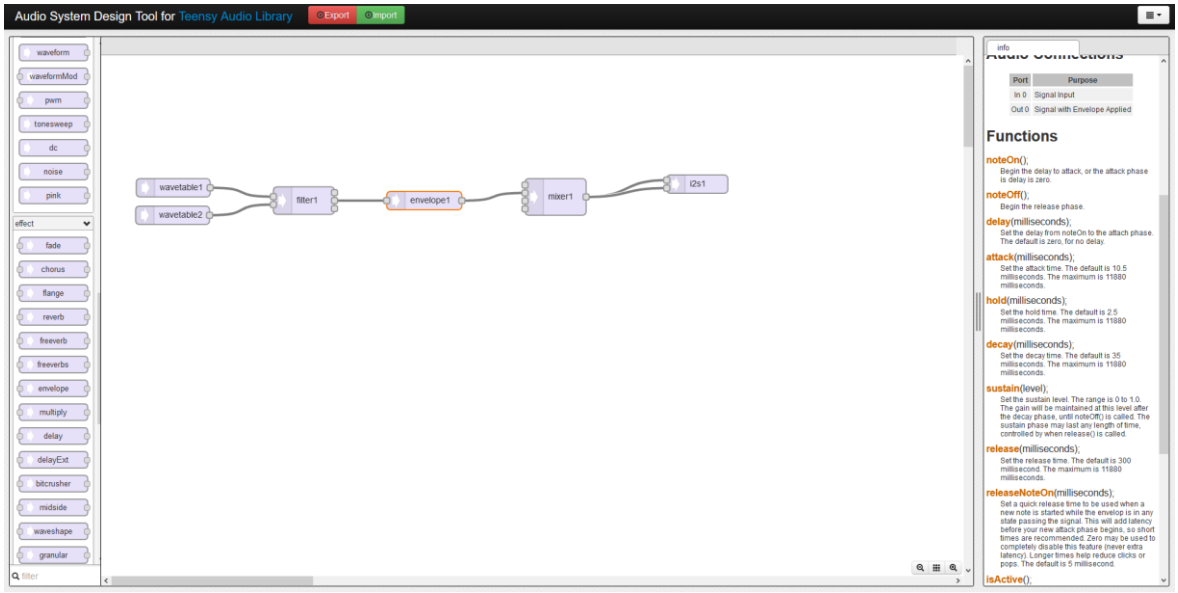
Pro vývoj aplikací lze použít vývojové prostředí ArduinoIDE, do kterého lze nainstalovat knihovny Teensy. Pro nahrání aplikace do mikropočítače slouží USB konektor.

Mikrokontrolery obsahují na vývojové desce D/A převodník s rozlišením 12 bitů, není zde však výstupní konektor, je tak nutné pomocí dostupných schémat jej připojit. Existuje také rozšiřující adaptér, který lze dokoupit a použít jako vstupní či výstupní. Tento adaptér obsahuje 16 bitový D/A převodník a poskytuje audio vzorky o frekvenci 44,1 kHz. [28]

Knihovna jako taková poskytuje funkce pro polyfonní hraní, nahrávání audia, syntézu zvuku, analýzu, audio efekty jako obálkové generátory, ozvěny, zpoždění, chorus efekt. [29]

Také poskytuje tři různé druhy filtrů: FIR filtry, IIR a stavový variabilní filtr, který nabízí dolní, horní, pásmovou propust a pásmovou zádrž.

Pro návrh audio aplikace je zde poskytnuta webová aplikace Audio System Design Tool. Z grafických bloků lze zde sestavit architekturu aplikace. Bloky se mezi sebou propojují, což utváří cestu signálu. Ke každému bloku se zobrazí v aplikaci příslušné funkce, takže tato webová aplikace funguje také jako dokumentace knihovny. Po dokončení návrhu se exportuje kód s inicializací tříd náležících použitým blokům. Použitý programovací jazyk je C a C++. [29]



Obrázek 16 Webová aplikace Audio System Design Tool

Přímo součástí Teensy Audio Library není, ale tvůrci také poskytují knihovnu pro MIDI rozhraní. Funkce v ní obsažené umožní vytvořit MIDI kontrolér, který je schopen posílat různé druhy MIDI zpráv. Pro přijímání zpráv je nutné nejprve implementovat funkci, kterou má daná zpráva spustit, takzvaně tyto funkce přetížit. MIDI komunikace je možná i přes USB konektor na vývojové desce.

3.6.3 MozziLibrary

MozziLibrary je knihovna pro zvukovou syntézu používaná především na mikro kontrolérech Arduino. Lze jí ale také využít na mikropočítačích Teensy, ESP8266 a STM32F103. [30]

Pracuje na podstatně nižších vzorkovacích frekvencích, otestovaná a doporučená frekvence je 16384 Hz, vzorkovací frekvence 32768 Hz je pak uvedena jako experimentální. Výstupní bitovou hloubku audio signálu je možné zvolit jako 8 bitovou nebo 14 bitovou, u mikropočítače Teensy je rozlišení 12 bitů. Na mikro kontrolérech Arduino je audio výstup namísto D/A převodníku řešen pomocí pulsně šířkové modulace. [30]

K syntéze zvuku knihovna používá předpřipravené vyhledávací tabulky se signály, pomocí skriptu poskytnutého knihovnou je možné vytvořit vlastní, nebo lze nahrávat vlastní vzorky. Dále jsou k dispozici funkce pro filtrování, dolní propust a stavový variabilní filtr, která poskytuje dolní, horní a pásmovou propust a pásmovou zádrž a funkce pro obálkový generátor.

Nenachází se zde přímo implementované audio efekty, jsou zde ale funkce pro práci se zpožďovací linkou, která se využívá pro vytvoření audio efektů, jako ozvěna, chorus efekt a další.

Z hlediska MIDI jsou zde funkce pouze pro konvertování MIDI noty na frekvenci pro zpracování knihovnou.

Firma Bastl Instruments používá tuto knihovnu ve své řadě digitálních syntezátorů Trinity, které používají procesor Atmega 328, známý z vývojových desek Arduino Uno. Knihovnu lze použít k uzpůsobení syntezátoru Trinity a dle vlastních představ jej přeprogramovat. Použitý programovací jazyk je C a C++. [31]

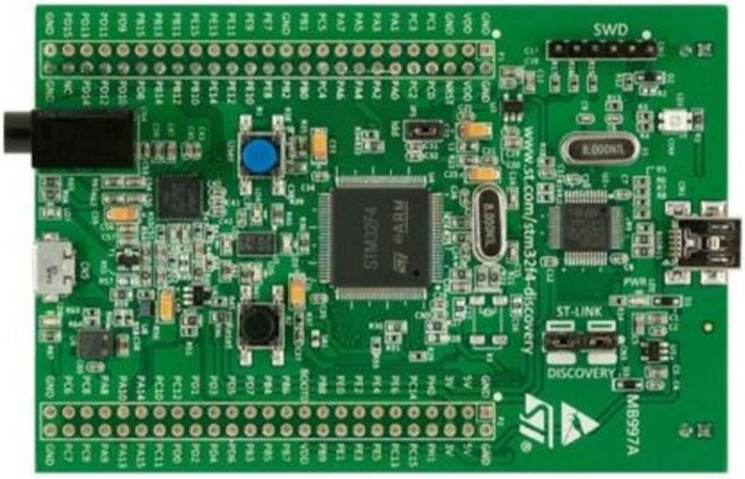
Obrázek 17 Digitální syntezátor Trinity vytvořený pomocí Mozzi Library [32]

4 VÝVOJOVÉ PROSTŘEDKY PRO IMPLEMENTACI

Pro zpracování zvukového signálu v podobě hudebního nástroje je kritické, aby nedocházelo ke zpoždění, v takovém případě je to velmi znatelné a nežádoucí. Proto je důležité zvolit procesory, které disponují vysokým výkonem. Velmi užitečné jsou také další periferie jako A/D a D/A převodníky, dostatek GPIO pinů a další. Tyto podmínky naplňuje vývojová deska STM32F4 Discovery.

4.1 STM32F4 Discovery

Vývojová deska STM32F4 Discovery je vybavena 32 bitovým ARM procesorem. Přímo na desce je také stereo D/A převodník CS43L22. Deska je určena především pro audio aplikace. Dalším vybavením pro audio aplikace na desce je digitální mikrofon. Velikost flash paměti pro uložení programu aplikace je 1 MB, velikost RAM 192 kB. Deska je také osazená FPU koprocesorem pro vykonávání matematických operací s čísly s pohyblivou desetinnou čárkou. [33]

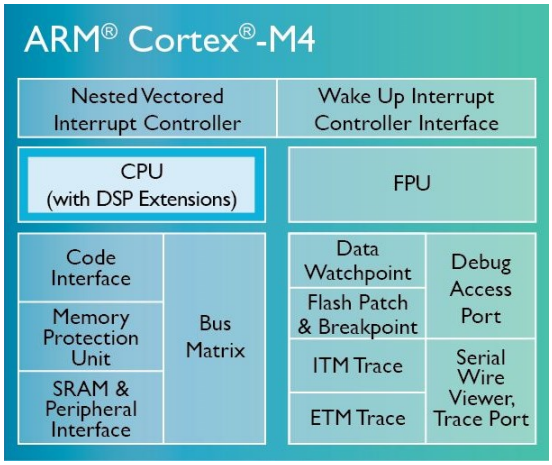


Obrázek 18 Vývojová deska STM32F4 Discovery [34]

Součástí desky je programátor ST-LINK dostupný přes USB Mini port. USB Mini port na desce může sloužit také pro ladění aplikace či jako virtuální sériový port. Druhý port typu Mikro USB poskytuje možnost aplikacím fungovat jako hostitelské zařízení pro jiné USB zařízení. Z desky je možné napájet další zařízení pomocí 5 V, nebo 3 V. [33]

4.1.1 Procesor

Procesor osazený na této desce je STM32F407VGT6, schopný pracovat o frekvenci až 168 MHZ. Jedná se o 32 bitový procesor s jádrem ARM Cortex-M4. M označuje procesory určené pro vestavěné aplikace, zaručující předvídatelné chování. Architektura tohoto procesoru je založena na Harvardské architektuře, to znamená, že je zde fyzicky oddělena paměť pro data a pro program. [35]



Obrázek 19 Diagram jádra Arm Cortex-M4 [36]

Instrukce procesoru jsou založeny na typu instrukcí RISC, redukované instrukční sady, to znamená, že jednotlivé instrukce vykonávané hardwarovou součástí procesoru jsou jednodušší, složitější jsou pak vykonávané softwarově. V důsledku užití RISC je vyžadován složitější překladač. Je zde užit také pipeline mechanismus, kdy je v jednom hodinovém cyklu procesoru načtena instrukce, předchozí je dekódována a předchozí vykonávána. [35]

4.1.2 D/A převodník CS43L22

Tento čip přítomný na desce obsahuje stereo D/A převodník schopný poskytnout až 24 bitové rozlišení. Je v něm integrován také zesilovač třídy D schopny pracovat při nízkém napětí. Součástí jsou také instrukce pro zpracování signálu. Data pro čip jsou přenášena sériově pomocí protokolu I^2S. Přímo na desce se nachází 3,5 milimetrový audio konektor, ke kterému je možné připojit sluchátka nebo reproduktor. [37]

4.1.3 USB On The Go

Na desce je dostupný konektor USB micro, který je zde jako OTG konektor. OTG znamená, že zařízení může fungovat jako hostitel, ke kterému připojíme periferií. V případě aplikace jako je syntezátor tak lze k desce připojit MIDI zařízení jako například MIDI klávesy, nebo jiný MIDI kontrolér, přes USB.

4.2 Vývojové prostředí

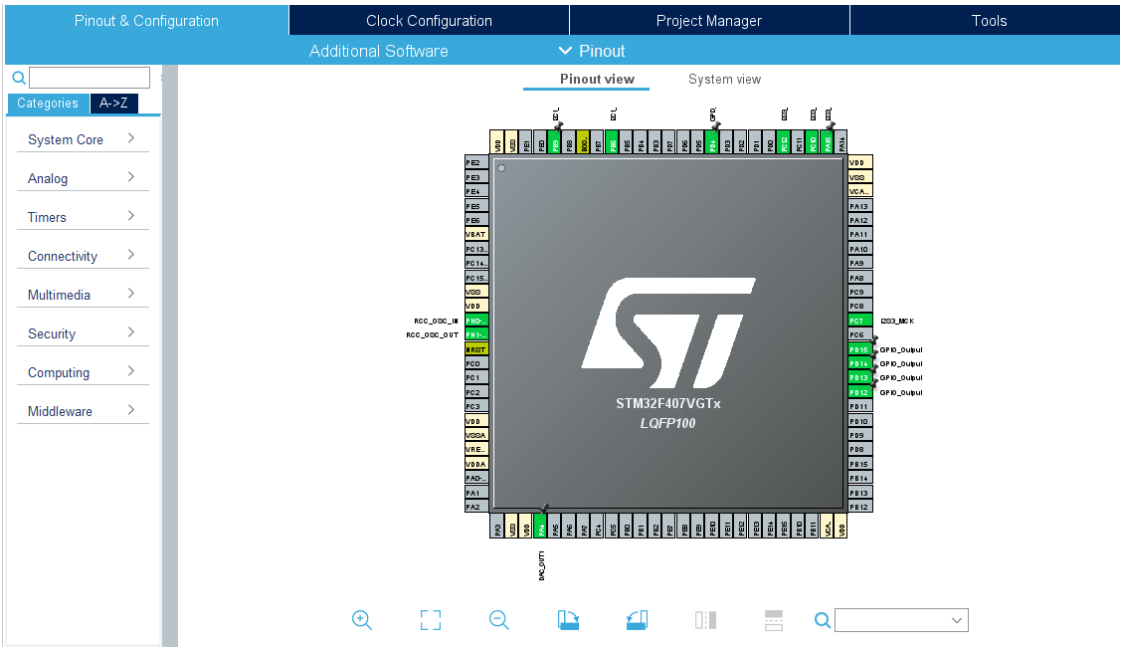
Pro usnadnění vývoje aplikací existují vývojové prostředí. Základní funkcí vývojového prostředí je editace zdrojového kódu. Díky automatickému napovídání či poskytnutí popisu z dokumentace je urychleno psaní aplikaci, zvýraznění syntaxe zase napomůže ke zpřehlednění kódu. Kromě toho vývojové prostředí obsahuje zpravidla také nástroje pro sestavení aplikace, jeho ladění, případně upozorňuje na chyby v kódu. [38]

4.2.1 Eclipse

Eclipse je multiplatformní vývojové prostředí určené především pro vývoj v jazyce Java, díky filozofii návrhu prostředí je možnost jej rozšířit o pluginy, které umožní vývoj i v dalších jazycích jako PHP, HTML, či C a C++. Dnes je dostupných několik verzí Eclipse pro vývojáře, které jsou zaměřeny přímo pro daný jazyk. Prostředí je vytvořeno pomocí programovacího jazyku Java. [39]

4.2.2 STM32CubeIDE

Multiplatformní vývojové prostředí pro mikropočítače STM32 s podporou jazyka C a C++. Je založeno na prostředí Eclipse. Spojuje v jedno dva samostatné nástroje, vývojové prostředí pro psaní a editaci kódu, nahrávání a ladění aplikací TrueStudio a nástroj pro správu periferií a generování příslušného kódu CubeMX. Při vytvoření projektu se zvolí mikropočítač, pro který bude aplikace vyvíjena a provede se nastavení příslušných periferií, po dokončení se vygeneruje kód, který odpovídá příslušnému nastavení. Poté se lze přesunout k implementaci aplikace. Ke správě periferií se lze kdykoliv vrátit. [40]



Obrázek 20 Nástroj pro správu periferií v STM32CubeIDE

4.2.3 ST Link Utility

ST Link Utility není vývojové prostředí pro editaci kódu, ale nástroj určený pro programování a nahrávání binárních souborů do pamětí mikropočítače. Kromě toho lze také pomocí tohoto nástroje paměť mazat, provádět kontrolní výpočty.

II. PRAKTICKÁ ČÁST

5 VLASTNOSTI SYNTEZÁTORU

Výstupem praktické části je implementace zvukového syntezátoru s mikropočítačem.
Pro HW část implementace jsem vybral mikropočítač STM32F4 – Discovery, především díky jeho vysokému výkonu a také dostupným periferiím přímo na vývojové desce. Zejména byl vybrán kvůli kvalitnímu D/A převodníku s audio výstupem a OTG USB konektoru.
K implementaci SW části jsem použil knihovnu MIOS32, kterou využívám k ovládání periferií a správu procesů s tím spojených, knihovna totiž funguje také jako operační systém, založený na Free RTOS. Knihovna je ale určena především pro MIDI kontroléry a sekvencéry, neobsahuje žádné funkce pro syntézu zvuku, tj. generování signálu, filtry či další běžné součásti syntezátorů. Implementace SW architektury zvukového syntezátoru včetně všech potřebných komponent proto tvoří většinu praktické části mé práce.
Syntéza zvuku v mém syntezátoru se nejvíce podobá subtraktivním syntezátorům, používaným především na přelomu 70. a 80. let. Oproti těmto ovšem nabízí polyfonii, konkrétně šest hlasů. Ty jsou tvořeny celkově třemi oscilátory, z nichž dva nabízí celkem pět různých průběhů signálu, lze je také mírně rozladit, čímž lze docílit tzv. chorus efektu, který dodává na bohatosti zvuku a zní jako více nástrojů najednou. Také tím lze simulovat analogové syntezátory, které znějí lehce rozladěně, kvůli nestabilitě, zejména teplotní, elektronických součástek. Paradoxně tato nedokonalost je velmi žádaná. Třetí oscilátor potom přidává basy, jeho frekvence je vždy o polovinu, tedy o oktávu nižší, než u ostatních dvou oscilátorů, signál je obdélníkový, ovládat lze jen jeho hlasitost.

Pro každý hlas se počítá obálkový generátor, ovládající amplitudu signálu. Tuto hodnotu lze také využít pro úpravu mezní frekvence filtru.

V syntezátoru je obsažen také zpožďovací efekt, vytvářející dojem ozvěny.

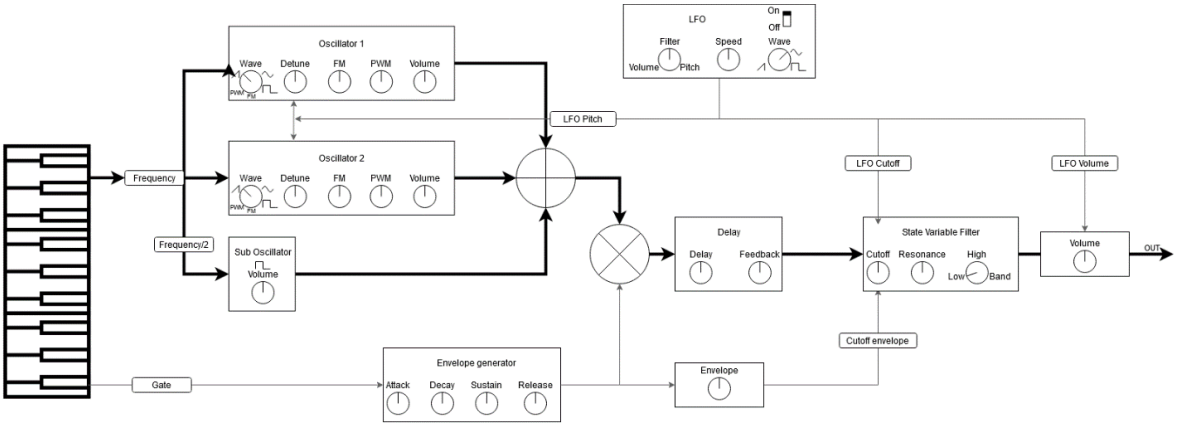
Nakonec projde signál přes filtr, který může pracovat jako dolní, horní nebo pásmová propust.

Poslední součástí je nízkofrekvenční oscilátor, který může průběžně upravovat mezní frekvenci filtru, celkovou hlasitost nebo výšku tónu.

Architektura syntezátoru je vyobrazena na Obr. 21. Zvýrazněná čára zobrazuje cestu signálu, ostatní představují parametry, které signál modulují a zobrazují, v kterých místech do signálu zasáhne.

Generování signálů pro syntézu zvuku je podrobněji popsáno v kapitole 7.2.

Kapitola 7.1 se věnuje zpracování MIDI zpráv, jak CC zpráv, tak i hlasových zpráv. Kapitola 8 popisuje hardwarové řešení připojení dvou potenciometrů pro ovládání parametrů a LCD displeje, kterým je syntezátor vybaven.



Obrázek 21 Architektura syntezátoru

6 PŘÍPRAVA PROSTŘEDÍ PRO VÝVOJ

Pro vývoj syntezátoru byla zvolena knihovna MIOS32 Midibox Operating System. Pro vývoj aplikací pod touto knihovnou je potřeba si připravit několik nástrojů a připravit vývojové prostředí Eclipse pro překlad kódu na binární soubory. Vývoj lze provádět jak na platformě Windows, tak Linux či Mac OS. Kromě toho lze udělat i jednoduché úpravy na vývojové desce.

6.1 Základní nástroje

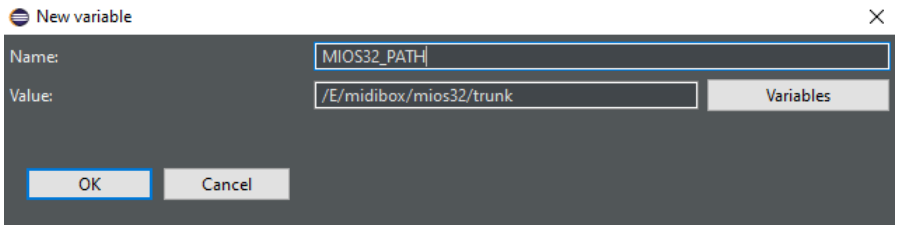
Všechny nástroje pro vývoj jsou volně dostupné. Patří mezi ně Java, potřeba pro běh prostředí Eclipse, MSYS pro spouštění skriptů pro sestavení aplikace, které využívají UNIX příkazů a sada GCC nástrojů s překladačem pro procesory ARM Cortex M3. Přestože je STM32F4 Discovery vývojová deska vybavena procesorem Cortex M4, spustí i aplikace pro Cortex M3. Nevýhodou je odstavení FPU, která u Cortex M3 nebyla běžná.

6.1.1 Instalace nástrojů

Pro běh prostředí vývojového prostředí Eclipse je potřebné, aby bylo nainstalováno Java JRE. Verze Javy by měla být 8. Stabilní verze Eclipse pro vývojáře C a C++, vhodná pro potřeby knihovny MIOS32, je verze nazvaná Luna. Eclipse není potřeba instalovat, stačí jej rozbalit na požadované místo. Pro MSYS je provedena standartní instalace. Užitečný je také nástroj pro správu verzí git, pomocí kterého lze jednoduše stáhnout knihovnu z repositáře. [41]

6.1.2 Konfigurace Eclipse

V Eclipse je nutné nejprve nalinkovat knihovnu MIOS32 a následně další proměnné pro sestavení aplikace. V preferencích, v záložce C/C++, sestavení, prostředí se vytvoří proměnná s prefixem pro překladač.



Obrázek 22 Vytvoření nové proměnné v Eclipse

V následující tabulce jsou vyplněny všechny potřebné proměnné pro sestavení aplikací MIOS32. Je důležité pro proměnnou MIOS32_PATH použít klasické lomítko, přestože je práce prováděna ve Windows, proměnná je volána skripty knihovny, používající UNIXový styl pro cesty k adresářům. [41]

Tabulka 5 Proměnné pro knihovnu MIOS32

Název proměnné	**Hodnota**
MIOS32_GCC_PREFIX	arm-none-eabi
MIOS32_FAMILY	STM32F4xx
MIOS32_PROCESSOR	STM32F407VG
MIOS32_BOARD	MBHP_CORE_STM32
MIOS32_LCD	Universal
MIOS32_BIN_PATH	${MIOS32PATH}/bin
MIOS32_PATH	/E/midibox/mios32/trunk
PATH	E:\msys\1.0\bin;E:\mios32_toolchain\bin

Pro otestování funkčnosti lze načíst prázdný projekt z místa, kde byla uložena knihovna a projekt sestavit. Úspěšné sestavení vypíše na konzoli v Eclipse zprávu o tom, pro kterou desku, procesor a displej byla aplikace sestavena a také informace o velikosti binárního souboru určeného pro nahrání do mikropočítače.

```
Application successfully built for:
Processor: STM32F407VG
Family:    STM32F4xx
Board:     MBHP_CORE_STM32F4
LCD:       Universal
--------------------------------------------------------------------------------
arm-none-eabi-size project_build/project.elf
   text    data     bss     dec     hex filename
  85270     216   50608  136094   2139e project_build/project.elf
20000000 D __ram_start
2000c688 B __ram_end
```

Obrázek 23 Zpráva o úspěšném sestavení aplikace

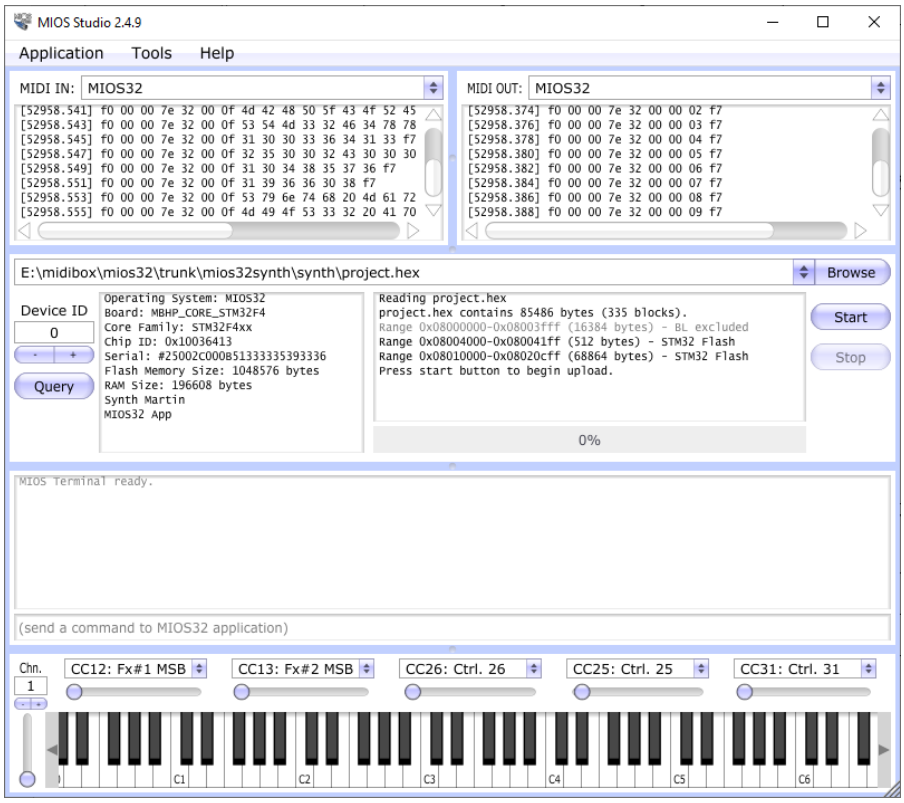
6.2 Programování mikropočítače

Před nahráním sestavené aplikace je nutné nejprve do mikropočítače nahrát zavaděč. Zavaděč umožní nahrávání aplikace do paměti mikropočítače pomocí MIDI sysex zpráv. Nahrání zavaděče se provádí pouze jedenkrát, je pro to použit USB Mini port s programátorem ST-LINK. Pro nahrání lze použít program ST LINK utility.

Úspěšné nahrání zavaděče do mikropočítače oznámí dvojité probliknutí LED diody na vývojové desce.

6.2.1 Nahrávání aplikací

Do zavaděče se aplikace nahrávají aplikace pomocí programu MIOS Studio, který je dostupný od vývojářů knihovny MIOS32. Binární soubor lze nahrát přímo přes MIOS Studio, nebo pomocí příkazové řádky.



Obrázek 24 MIOS Studio s vybraným binárním souborem

7 PROGRAMOVÉ ŘEŠENÍ SYNTEZÁTORU

Knihovna MIOS32 používá pro své aplikace vlastní strukturu souborů. Základní a nutné soubory jsou mios32_config.h, který obsahuje základní konfigurace funkcí, využití pinů pro analogové vstupy, použití periferií jako I^2S pro audio výstup či sériovou komunikaci pro přijímání MIDI zpráv.

Soubor app.c obsahuje základní úlohy, které jsou volány s různou prioritou. Některé z nich například reagují na příchozí MIDI zprávy, analogové vstupy nebo se zde nachází rutina, která provádí operace na pozadí, může například zasílat informace o aktuálním čase na displej. Zde se také mohou definovat vlastní úlohy.

Makefile obsahuje direktivy a příkazy pro sestavení aplikace. Z pohledu vývojáře MIOS32 se zde především navazují moduly knihovny, jejichž funkce následně aplikace využívá. Je důležité, aby jako poslední modul byl zahrnut common.mk, jinak by se aplikace nesestavila.

7.1 Zpracování MIDI zpráv

MIDI zprávy jsou přijímány callback funkcí App_MIDI_NotifyPackage, která se nachází v souboru app.c. Přijímá je ze všech dostupných a aktivovaných portů určených pro MIDI, jak USB, tak UART. Funkce následně předá obsah zprávy a port, ze kterého přišla funkci SYNTH_HandleMIDI, která se nachází v souboru synth.c. Aplikace přijímá pouze zprávy z kanálu jedna, ostatní ignoruje. Nejdříve zkontroluje, o jakou zprávu se jedná, zda se jedná o hlasovou Note On či Noteoff nebo CC zprávu. Poté předá zprávu k dalšímu zpracování. Pokud se jedná o CC zprávu, podle její hodnoty se provede příslušná změna parametru, například pro nastavení tvaru signálu, mezní frekvence filtru, přeladění oscilátoru a další. Každý měnitelný parametr má funkci pro nastavení, do níž je vstupem hodnota v rozsahu od 0 do 127, což odpovídá rozsahu hodnotové části MIDI zprávy. Po nastavení tato funkce taky aktualizuje LCD displej.

Tabulka 6 Seznam CC MIDI zpráv

Číslo CC zprávy	**Funkce**
1	Modulační kolečko, mezní frekvence filtru
7	Celková hlasitost
9	Hlasitost sub oscilátoru
10	Hlasitost prvního oscilátoru

11	Hlasitost druhého oscilátoru
12	Tvar signálu prvního oscilátoru
13	Tvar signálu druhého oscilátoru
14	Množství FM modulace
15	Šířka pulsního signálu PWM
21	Náběh obálkového generátoru/filtru
22	Útlum obálkového generátoru/filtru
23	Podržení obálkového generátoru/filtru
24	Doznívání obálkového generátoru/filtru
25	Rozladění prvního oscilátoru
26	Rozladění druhého oscilátoru
28	Mezní frekvence filtru
29	Q faktor filtru, rezonance
30	Přepínání dolní, horní, pásmová propust
32	Ovlivnění filtru obálkovým generátorem
33	Zapnutí, vypnutí LFO
34	Rychlost LFO
35	Přepínání hlasitost, filtr, výška tónu u LFO
36	Tvar signálu LFO
37	Maximální hodnota LFO
38	Délka zpožďovacího efektu
39	Zpětná vazba zpožďovacího efektu

7.1.1 Zpracování Note On

V případě, že se jedná o Note On zprávu se provede následující:

1. V cyklu For o šesti průchodech hledá v poli struktur voices volná pozice. Identifikuje se hodnotou noty, která je nastavena na číslo 255, v MIDI existuje nota s maximální hodnotou 127, proto hodnota mimo rozsah 0 – 127 značí notu, která nezní. Současně nota musí být v režimu obálkového generátoru ENVELOPE_OFF a nesmí být v režimu doznívání, tedy ENVELOPE_RELEASE.
2. Pokud se nalezne volné místo pro notu, přiřadí se na volné pozici ve struktuře hodnota noty, rychlost stisku noty, čas kdy byla nota stisknuta, režim obálkového generátoru na ENVELOPE_ATTACK.

3. Zavolá se funkce SYNTH_FreqSet_OSC pro každý oscilátor. Tato funkce nastaví fázový akumulátor, tedy krok, se kterým se prochází tabulka, ve které je uložen tvar signálu. Fázový akumulátor acc lze získat následujícím výpočtem.

$$acc = \frac{f}{f_s / l} \tag{7.1.1}$$

   Kde f je požadovaná frekvence, f_s je vzorkovací frekvence, l je délka tabulky.

4. Po přiřazení se procházení cyklu For ukončí, aby nedošlo k přiřazení dvou a více stejných not a zaplnění.

Pokud není nalezena volná nota, MIDI zpráva je ignorována.

7.1.2 Zpracování NoteOff

Při zpracování NoteOff, vypnutí noty, se opět v cyklu For projde pole struktur voices a pokud je nalezena stejná nota, jako je nota MIDI zprávy, je pouze uložen čas a nastaven režim obálkového generátoru na ENVELOPE_RELEASE. Vypnutí a nastavení noty na 255 je potom až na obálkovém generátoru, tak aby mohla nota doznít.

7.2 Generování signálu

Syntezátor může generovat pět různých signálů, z nichž některé lze různými způsoby modulovat. Signály v syntezátoru jsou implementovány metodou vyhledávací tabulky. Výčet dostupných signálů je v datovém typu enumsynth_waveform_t. Nastavení signálů se provádí pomocí funkce SYNTH_WaveformSet, která nastaví hodnotu jednoho ze signálu do dvourozměrného statického pole waveform_select.

O naplnění výstupu vzorky se stará funkce SYNTH_ReloadSampleBuffer, která plní výstupní pole o velikosti 128 se vzorkovací frekvencí 48 kHz. Výstupní datový typ je u32, 32 bitový bezznaménkový integer. Před výpočtem se deaktivují všechny ostatní přerušení, po dokončení se opět aktivují. Mezitím se zavolá funkce SYNTH_MixSample.

1. V cyklu for o šesti průchodech se hledá v poli struktur voices taková, která má členy splňující podmínku, že je zapnuta a není ve stádiu ENVELOPE_OFF

2. Když je taková nalezena, spočítá se pro ni amplitudová obálka pomocí funkce ENVELOPE a také se přičte k pozici každého oscilátoru hodnota akumulátoru. Pokud je pozice větší než velikost tabulek s průběhy signálu, je od pozice oscilátoru odečtena velikost tabulky, tak aby se pozice vracela v kruhu.

3. Poté se pro daný hlas vypočítá vzorek pomocí funkce SYNTH_OSC_output. Funkce přijímá parametry o zvolených průbězích signálu, akumulátory a pozice obou oscilátoru a rychlost stisku dané noty. Podle zvolených průběhů přepínač zvolí, ze kterých průběhů spočítat vzorek. Vzorek je také vynásoben rychlostí stisku noty a hlasitostí daného oscilátoru. Vypočítává se také hodnota sub oscilátoru, který má vždy obdélníkový signál a jeho frekvence je o polovinu, tedy o oktávu, nižší než ostatní oscilátory. Nakonec se vrátí součet vzorků z každého oscilátoru.
4. Vrácená hodnota se vynásobí aktuální hodnotou amplitudové obálky.
5. Následně se hodnota ošetří „soft clipping“ kubickou funkcí, aby se minimalizovalo zkreslení signálu.

$$S(n) = x - \frac{x^3}{3} \tag{7.2.1}$$

   Kde x je ošetřovaná hodnota a $n\ je\ aktuální\ vzorek$.

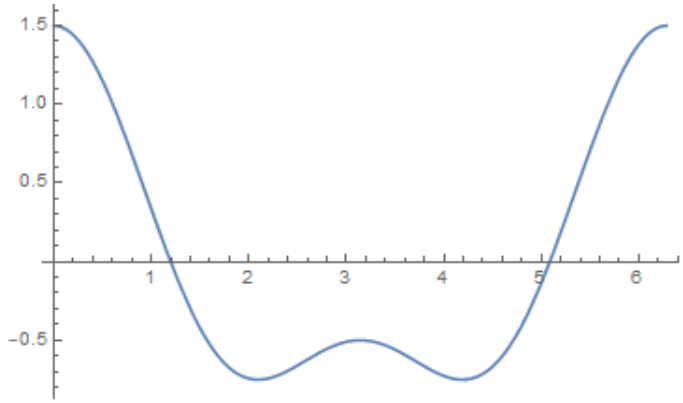
6. Výsledek se přičte k výstupní hodnotě. Ta po dokončení cyklu for projde přes efektovou zpožďovací linku delay a následně filtr řízený manuálně, nebo nízkofrekvenčním oscilátorem, nebo obálkovým generátorem.
7. Výsledek těchto operací se vrátí do funkce SYNTH_ReloadSampleBuffer a uloží se do výstupního pole, které se po naplnění odešle na D/A převodník. Ten je obsluhován ovladači knihovny MIOS32.

V další části jsou popsány jednotlivé algoritmy, které generují dané signály. Naivní implementací např. obdélníkového signálu, přepínáním mezi dvěma hodnotami, by docházelo při vyšších frekvencích ke zkreslení signálu, kvůli překročení Nyquistovi frekvence harmonickými frekvencemi v signálu. Tento jev je zde potlačen a téměř odstraněn pomocí metody přechodové tabulky. Proto každá funkce pro generování signálu přijímá dva parametry, fázový akumulátor acc a aktuální pozici pos pro výběr z tabulky.

Základem je v paměti uložená tabulka sStep o velikosti 2048 vzorků, obsahující hodnoty přechodové funkce. [22]

$$S(n) = (\cos\frac{2\pi n}{l}) + (\cos 2(\frac{2\pi n}{l}))/2 \tag{7.2.2}$$

Kde n je n-tá pozice v tabulce a l je velikost tabulky. Přechodová tabulka je tak složena ze dvou harmonických frekvencí.



Obrázek 25 Tvar signálu přechodové tabulky

Všechny algoritmy byly implementovány a jejich vlastnosti ověřeny v programu Wolfram Mathematica, než došlo k realizaci na vývojové desce. Lze zde snadno zobrazit, jak se projevuje přechodová část a zda tvar signálu odpovídá. Wolfram Mathematica také umožňuje pomocí funkce ListPlay přehrát zvuk tvořený pomocí algoritmu pro generování signálu, takže pro ověření správné funkčnosti bylo také možné poslechem zjistit, zda algoritmus ve vyšších frekvencích odstraňuje zkreslení způsobené aliasingem.

7.2.1 Obdélníkový signál

Funkce SYNTH_Pulse vrací ošetřený obdélníkový signál. Algoritmus pro získání vzorku vypadá následovně.

1. Nejprve se podle akumulátoru spočítá délka přechodu stepLen, vynásobením akumulátoru počtem kroků, které budou získány z přechodové tabulky. V syntezátoru jsou při každém přechodu použity čtyři kroky pro přechod.
2. Poté se podle pozičního parametru pos určí způsob, jakým bude zpracována výstupní hodnota.
    a. Pokud je pozice menší než polovina délky tabulky a zároveň je pozice menší než délka přechodu a tudíž se nachází v přechodové části, je index pro výběr z přechodové tabulky spočítán podle této rovnice.

$$i = \frac{pos}{stepLen} * \left(\frac{1}{2}\, l\right) \qquad (7.2.1.1)$$

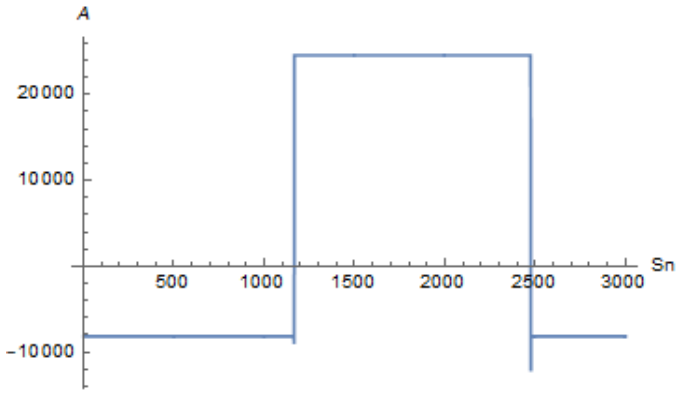
Kde i je index, pos je pozice v tabulce, stepLen je délka přechodu a l je délka tabulky. Vybírá se tak vzorek z první poloviny přechodové tabulky.

b. Pokud je pozice menší než polovina délky tabulky a větší než délka přechodu, vrací se záporná hodnota ze dvou, kterých může obdélníkový signál nabývat. Případně může být hodnota modulována přičtením hodnoty ze sinusové tabulky sinTab.

c. Pokud je pozice větší než polovina a menší než součet poloviny a délky přechodu, je v přechodové části, hodnota se vybírá z druhé poloviny přechodové tabulky. Index je spočítán následovně.

$$i = \left(\frac{1}{2}\ l\right) + \frac{pos - (0{,}5)}{stepLen} * \left(\frac{1}{2}\ l\right) \qquad (7.2.1.2)$$

d. Poslední fáze je v druhé polovině, kdy je pozice větší než součet poloviny a délky přechodu, pak se vrací kladná hodnota obdélníkového signálu. Opět zde může být hodnota modulována.



Obrázek 26 Ošetřený obdelníkový signál

Na obrázku je ošetřený obdélníkový signál, vykreslený pomocí programu Wolfram Mathematica.

7.2.2 Pilový signál

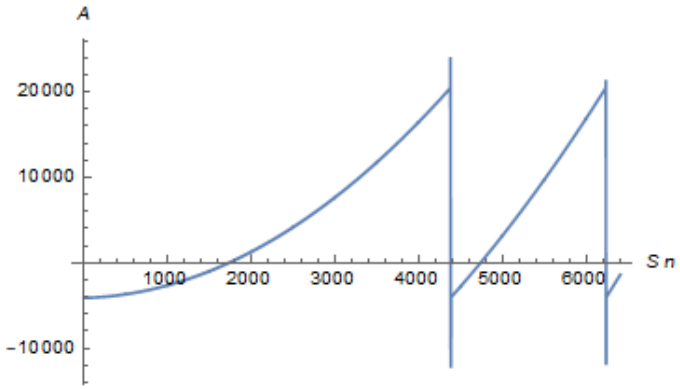
Signál je generován pomocí funkce SYNTH_Saw. Také tento signál je ošetřen vůči zkreslení. Pro tento signál je použita navíc v paměti uložena tabulka sawTab, která představuje lineárně rostoucí tvar pilového signálu.

1. V prvním kroku se spočítá délka přechodu stepLen. Oproti obdélníkovému signálu je zde osm kroků pro přechod.
2. Druhým krokem je určení pozice v tabulce a získání vzorku.

a. Pokud je pozice menší než polovina délky tabulky a také je menší než délka přechodu stepLen, nachází se v přechodové části. Index se získá následovně.

$$i = \frac{pos}{stepLen} * \left(\frac{1}{2}\ l\right) \qquad (7.2.2.1)$$

b. Ve všech ostatních případech, ve kterých se pozice může nacházet, je vždy vzorek získán výběrem z tabulky sawTab, přímo podle parametru pozice.



Obrázek 27 Ošetřený pilový signál

7.2.3 Pulsní šířkově modulovaný signál

Tento signál je generován ve funkci SYNTH_PWM. Tato funkce přijímá ještě třetí parametr PWM, který určuje šířku signálu. Samotný algoritmus pro získávání vzorku je velmi podobný jako pro obdélníkový signál. Indexy pro přechody se počítají naprosto stejně jako u funkce pro obdélníkový signál, rozdíl je v tom, že oproti obdélníkovému signálu se hodnota nepřepíná v polovině, ale přepnutí se řídí podle hodnoty PWM, čímž se změní šířka signálu. Pokud se nenachází v přechodové fázi, je vzorek získáván pomocí tabulky pro pilový signál.

$$S(n) = sawTab(pos) - sawTab(pos - PWM) \qquad (7.2.3.1)$$

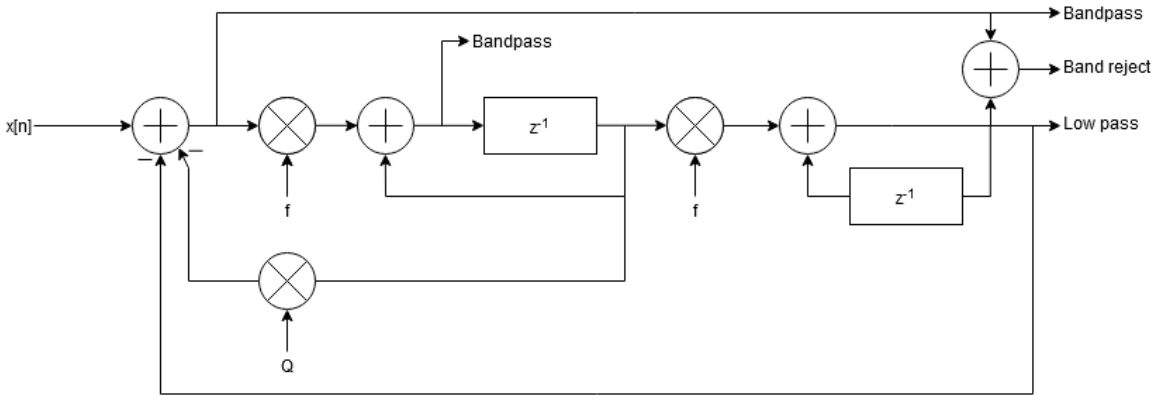
Kde n je aktuální vzorek, parametr pos představuje index pro výběr z tabulky, PWM je parametr pro pulsní šířkovou modulaci, nabývající hodnot od nuly do jedné, přičemž jedna odpovídá délce tabulky. Výsledkem odečtením hodnot pilového signálu je obdélníkový signál, pokud se v hodnotě, která je odčítána, provede fázový posun, změní se šířka tohoto signálu.

7.2.4 Sinusový signál

Pro vytvoření sinusového signálu existuje funkce SYNTH_Sin, která využívá tabulku sinTab a na základě pozičního parametru získává vzorek z této tabulky. Sinusový signál se skládá pouze z jedné harmonické frekvence, té základní, proto i při nejvyšším možném tónu nepřekročí daná frekvence Nyquistovu frekvenci, a tak není nutné řešit ošetření vůči zkreslení signálu. Frekvence může být modulována opět sinusovou tabulkou, tím že je k fázi přičtena hodnota z této tabulky.

7.3 Stavově proměnný filtr

Stavově proměnný filtr nabízí čtyři výstupy, lze jej použít jako dolní, horní, pásmovou propust a pásmovou zádrž. Filtr je přímo odvozen od svého analogového protějšku. V digitální verzi byl navržen Halem Chamberlinem takto:



Obrázek 28 Schéma Stavově proměnného filtru

Jedná se o filtr s nekonečnou impulsní odezvou (IIR). Proto může být filtr nestabilní při vyšších frekvencích. Výpočet koeficientu mezní frekvence filtru vypadá následovně: [21]

$$f = 2\sin\frac{\pi F_c}{F_s} \tag{7.2.4.1}$$

Kde f koeficient mezní frekvence filtru, F_c je mezní frekvence a F_s je vzorkovací frekvence. Okolo hodnoty $f = 1$ začíná nestabilita filtru, tato hodnota odpovídá $\frac{1}{6}$ vzorkovací frekvence. Řešením je zřetězení více filtrů za sebe, což zvýší stabilitu. Koeficient q představuje rezonanční chování filtru. Koeficient lze získat tímto výpočtem. [21]

$$q = \frac{1}{Q} \tag{7.2.4.2}$$

Kde Q nabývá hodnoty od $0{,}5$ do nekonečna, čím je hodnota q vyšší, tím spíše začne být filtr nestabilní a začne oscilovat.

Algoritmus filtru si uchovává a předává dvě hodnoty spojené s výpočty dolní a pásmové propusti. Nachází se ve funkci FILTER_StateVariableResonance. Výsledek lze získat následujícím kroky. [21]

1. Úprava a normalizace mezní frekvence$f$ a rezonančního koeficientu q tak aby se předešlo nestabilitě.
2. Dolní propust se vypočítá vynásobením mezní frekvence f a pásmové propusti, která je při prvním průchodu rovna nule. Tato hodnota se při průchodech přičítá k předchozí.
3. Horní propust se vypočítá odečtením dolní propusti a pásmové propusti vynásobenou s koeficientem q od vstupní filtrované hodnoty. Tato hodnota se při každém průchodu přepisuje.
4. Pásmová propust se vypočítá vynásobením mezní frekvence f a horní propusti. Tato hodnota se při průchodech přičítá k předchozí.
5. Po dokončení prvního průchodu se uloží do výstupních proměnných jednotlivé výsledky propustí.
6. Opakování průchodu od kroku číslo dvě. Po těchto průchodech se výsledky přičítají k prvnímu, nepřepisují se
7. Přepínač, podle parametru mode, rozhodne, který výstup vrátí.

7.4 Obálkový generátor

Obálkový generátor je implementován ve funkci SYNTH_Envelope. Hodnota, kterou vrací, upravuje amplitudu signálu. Ta má čtyři fáze, náběh, útlum, podržení, doznívání. Každá fáze má parametr, který upravuje její délku trvání. Funkce je volána s dvěma parametry, hlasem, pro který je funkce volána a aktuálním časem, získaným pomocí funkce xTaskGetTickCount, která je součástí Free RTOS. Funkce je volána ve funkci SYNTH_MixSample. Po příjmu MIDI zprávy Note On je nastaven na příslušné pozici ve struktuře voices čas stisknutí noty a při každém volání funkce SYNTH_Envelope je tato hodnota použita k výpočtu. Také je zde nastavena fáze na ENVELOPE_ATTACK. Ve

funkci ENVELOPE se v této fázi se počítá délka doby života noty, tedy celková doba kdy je nota stisknuta, odečtením času stisknutí od parametru funkce představující aktuální čas. Pokud je klávesa uvolněna a je přijata zpráva Note Off, je fáze nastavena na ENVELOPE_RELEASE. Zde se počítá délka doby života noty odečtením času stisknutí od času uvolnění noty.

1. Pokud je hodnota doby života noty menší nebo rovna hodnotě určující délku náběhu noty, je výstupní hodnota počítána následovně:

$$A = \frac{l}{a} * s_1 \tag{7.4.1}$$

   Kde A je výstupní hodnota představující amplitudu singálu, l je doba života noty, a je délka náběhu noty a s_1 je maximální hodnota, ke které postupně amplituda narůstá, v tomto případě je to hodnota 128.

2. Pokud je doba života noty větší než hodnota určující délku náběhu noty, ale je také menší než součet délky náběhu noty a délky útlumu od maximální hodnoty, je amplituda postupně klesající počítána následovně.

$$A = \left(\frac{(l-a)}{d} * (s - s_1)\right) + s_1 \tag{7.4.2}$$

   Kde d je délka útlumu od maximální hodnoty a s je hodnota podržení noty, ke které v této fázi amplituda postupně klesá.

3. Pokud doba života přesáhne součet hodnot délky náběhu a útlumu, je výstupní hodnota rovna hodnotě podržení s. V tento moment se také hodnota podržení chová jako celková hlasitost.

4. Po uvolnění noty dojde k doznívání noty. Podle délky doby života, v případě doznívání odečtením času stisknutí noty od uvolnění noty, se vypočítá pomocná hodnota pomocí jedné ze tří předchozích možností – záleží, ve které fázi byla nota uvolněna. Následně je výsledná hodnota amplitudy získána tímto způsobem,

$$A = \left(\frac{l_1}{r} * (0 - A_1)\right) + A_1 \tag{7.4.3}$$

   Kde l_1 čas od uvolnění noty, získaný odečtením času uvolnění od aktuálního času, který do funkce vstupuje jako parametr. r je délka doznívání noty. A_1 je pomocná hodnota.

Poté, co doba od uvolnění noty přesáhnou délku doznívání, je ve struktuře voices nota odstraněna, její fáze nastavena na ENVELOPE_OFF a hodnota noty je nastavena na 255, tedy takovou, kterou nelze zaměnit s hodnotou noty přicházející MIDI protokolem. Zde je rozsah možný od 0 do 127. Pokud je některý z parametrů roven nul, je jeho fáze automaticky přeskočen.

Hodnota obálky pro každý hlas může také ovládat hodnotu mezní frekvence filtru, a to stejným způsobem jako amplitudu. Parametr filter_envelope určuje, jak moc je mezní frekvence filtru ovládána obálkou. Čím je jeho hodnota větší, tím se zmenšuje vliv hodnoty filter_cutoff na mezní frekvenci filtru.

7.5 Nízkofrekvenční oscilátor

Tento oscilátor je použit pro průběžné modulování různých parametrů. Využívá modul knihovny MIOS32 sequencer. Z tohoto modulu je využita funkce pro generování BPM. To pak v pravidelných intervalech spouští příslušnou funkci provádějící specifickou událost. Pro nízkofrekvenční oscilátor jsou vytvořeny tabulky o 128 hodnotách, představující průběhy sinusového, obdélníkového a pilového signálu, v souboru lfoTab.h

1. Při inicializaci syntezátoru se vytvoří úloha pomocí funkce xTaskCreate, nazvaná TASK_LFO. Ta je volána systémem periodicky každou milisekundu. [42]

   V této funkci se zavolá funkce LFO_Handler.

2. Funkce LFO_Handler v cyklu do-while kontroluje požadavky ze sequencer modulu ohledně BPM. Jedná se o požadavky na zastavení generování BPM, spuštění generování, pokračování, nastavení pozice z externího MIDI signálu a požadavek na hodinový signál, který vrací hodnotu bpm_tick, tedy moment spuštění podle BPM signálu. Rychlost BPM signálu se nastaví pomocí funkce SEQ_BPM_Set, například pomocí MIDI zprávy. V momentě, kdy je vrácena hodnota bpm_tick, se zavolá funkce LFO_Tick.

3. Ve funkci LFO_Tick se podle hodnoty LFO_target vybere, který parametr bude modulován. Lze tak zvolit z modulování hlasitosti, mezní frekvence filtru a výšky tónu.

4. Zavoláním funkce LFO_Out se získá aktuální hodnota, která se poté použije pro nastavení zvoleného parametru. Funkce je volána se dvěma parametry, LFO_mode, který určuje, průběh signálu nízko frekvenčního oscilátoru a LFO_phase, který

určuje jeho fázi, tedy pozici ve vyhledávací tabulce, podle které se vrátí aktuální hodnota.

5. Následně se inkrementuje hodnota iterátoru LFO_phase o jednu a zkontroluje se, zda nepřekračuje velikost LFO tabulek. Pokud ano, hodnota iterátoru se nastaví na nulu, aby mohl nízkofrekvenční oscilátor plynule opakovat svou činnost.

7.6 Zpožďovací efekt

Zpožďovací efekt simuluje ozvěnu, kromě toho se jeho implementace dá použít jako základ pro další efekty využívající zpožďovací linku. Tento efekt si průběžně ukládá aktuální hodnotu a poté na základě parametrů delay_val, který udává vzdálenost mezi jednotlivými ozvěnami a feedback_val, který udává dobu, za kterou ozvěna zanikne, vytváří efekt ozvěny. Základem tohoto efektu je pole hodnot delay_buffer o 48000 hodnotách. Zpožďovací efekt je implementován ve funkci EFFECTS_Delay.

1. Při inicializaci syntezátoru se všechny hodnoty v poli delay_buffer nastaví na nulu, aby nedošlo k chybě, kdy by se funkce pokoušela číst z neexistující hodnoty.
2. Funkce EFFECTS_Delay se zavolá ve funkci SYNTH_MixSample. Vstupním parametrem této funkce je dosavadní hodnota získaná z generování signálů.
3. Spočítá se index pro výběr z pole hodnot delay_buffer. Při prvním volání je index nastaven na nultou pozici. Index se získá následovně.

$$idx = i - l \tag{7.6.1}$$

   Kde idx je pomocný index, i je aktuální hodnota indexu a l je vzdálenost mezi ozvěnami, ta může nabývat až hodnoty celkové velikosti delay_buffer, tedy 48000, což představuje délku ozvěn až jednu sekundu.

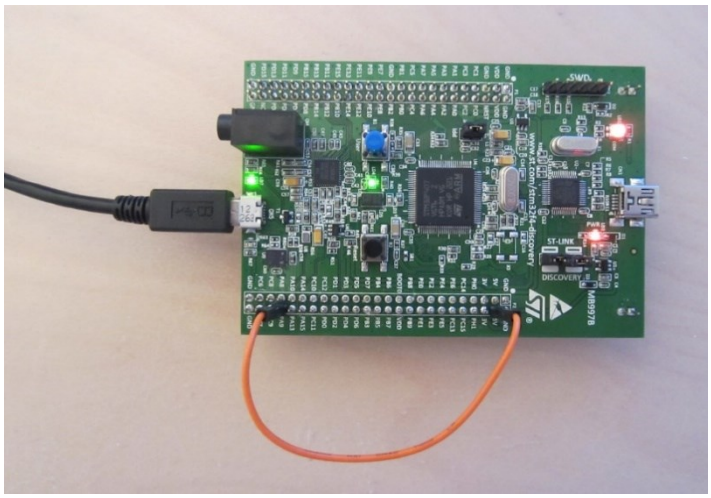
4. Pokud je pomocný index idx menší než nula, přičte se k němu velikost delay_buffer, aby se index nedostal na neexistující pozici.
5. Získaná hodnota pomocí indexu idx z delay_buffer se uloží do pomocné proměnné, ta se přičte ke vstupní hodnotě. Pomocná proměnná je vynásobena hodnotou feedback_val. Pokud je tato hodnota nulová, ozvěna nezní, pokud je rovna jedné, zní nekonečně.
6. Zvýší se aktuální hodnota indexu i o jednu hodnotu, pokud přesáhne velikost delay_buffer, odečte se od indexu i tato hodnota.

7. Funkce vrátí součet vstupní hodnoty a hodnoty získané z delay_buffer pomocí vypočítaného indexu idx.

8 HARDWAROVÉ ŘEŠENÍ SYNTEZÁTORU

Díky vybavenosti vývojové desky STM32F4 Discovery a knihovně MIOS32 může syntezátor pracovat bez jakýchkoliv dalších periferií a napájení odebírat z micro USB konektoru, přes který zároveň lze posílat MIDI zprávy.

Pro umožnění napájení pomocí micro USB, je nutné propojit pin PA9 na desce, s 5V pinem. [43]



Obrázek 29 Zapojení pro napájení pomocí mikro USB[44]

Pro lepší ovládání parametrů bez nutnosti použít kontrolér podporující CC MIDI zprávy jsou k syntezátoru připojeny dva potenciometry a pro přehled dvou řádkový LCD displej o šestnácti znacích. Některé MIDI klávesy totiž nejsou vybaveny programovatelnými ovládacími prvky na vysílání CC zpráv.

K vývojové desce je připojen také klasický MIDI konektor pěti kolíkový DIN. Umožňuje tak nejen připojit starší MIDI klávesy, ale také ovládat syntezátor z více MIDI zdrojů.

Knihovna MIOS32 poskytuje také moduly pro vývojové desky, které umožňují připojení například většího množství periferií, MIDI vstupů a výstupů, různých znakových i grafických displejů a dalších. Pro účely syntezátoru byla využita pouze část pro displej, potenciometry byly připojeny přímo k vývojové desce.

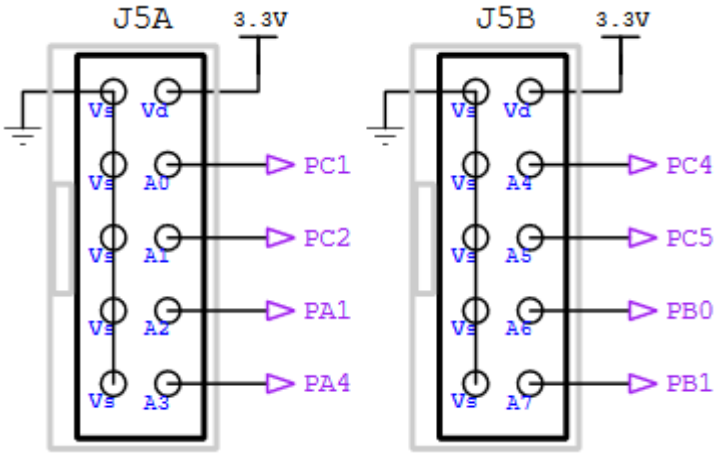
8.1 Ovládání syntezátoru

Kromě MIDI lze syntezátor ovládat pomocí dvou potenciometrů. Jeden slouží k procházení mezi parametry a druhý jako datový vstup pro daný parametr. Poskytované rozlišení A/D

převodníku je 12 bitů, maximální získatelná hodnota je tedy 4095. Pro kompatibilitu s MIDI zprávami je tato hodnota zmenšena na maximální možnou 127. [45]

Využité potenciometry jsou lineární a mají odpor 10 kΩ.

Nastavení pinů se provádí v souboru mios32_config.h. Pomocí direktivy MIOS32_AIN_CHANNEL_MASK 0x0011 jsou nastaveny piny PC1 a PC4.



Obrázek 30 Modul analogových vstupů[46]

Veškeré změny na A/D převodníku jsou převáděny pomocí DMA protokolu, což umožní přímý přístup periferiím na paměť a nezatíží tak zbytečně procesor. [47]

Po dokončení převodu přerušení A/D převodníku přesune získané hodnoty do funkce APP_AIN_NotifyChange, z argumentů pro tuto funkci lze získat hodnotu na A/D převodníku a také pin, na kterém změna nastala. Ta se projeví, jakmile přesáhne hodnotu mrtvého pásma, které zabraňuje náhodným hodnotám způsobených mechanickými zákmity. Pásmo lze nastavit pomocí direktivy MIOS32_AIN_DEADBAND v souboru mios32_config.h. Jeho hodnota je nastavena na 31, podle této definice.

$$2^{12-r} - 1 \tag{8.1.1}$$

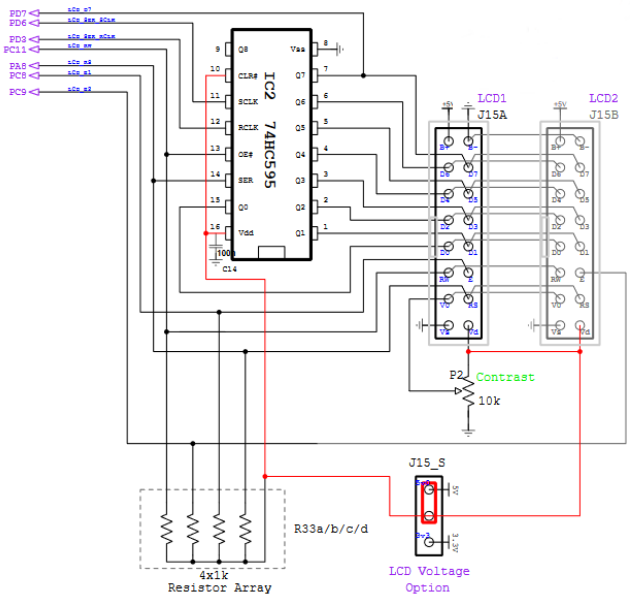
Kde 12 je maximální bitové rozlišení A/D převodníku, r je požadované rozlišení, pro kompatibilitu s MIDI zprávami, tedy 7 bitů.

Na základě pinu 0 se změní hodnota pin_menu, která poté v přepínači volí mezi jednotlivými parametry, viz Tabulka 6. Pin 1 změní hodnotu pin_CC, která představuje datový vstup pro ovládací funkci parametru. Při přepínání mezi parametry se automaticky volá funkce pro aktualizaci LCD displeje, pokud dojde ke změně na datovém vstupu, zavolá se příslušná funkce pro nastavení parametru a ta poté aktualizuje displej.

8.2 LCD displej

Pro lepší přehled v ovládání syntezátoru je využit šestnácti znakový displej o dvou řádcích s ovladačem HD44780. Knihovna MIOS32 podporuje několik dalších displejů, kromě tohoto i další znakové displeje o jiných rozměrech a také několik grafických displejů. Zároveň je možné připojit až dva displeje.

K připojení displeje je nutné mít modul pro LCD displeje, který je součástí knihovny MIOS32.



Obrázek 31 Schéma LCD modulu[46]

Proměnná knihovny MIOS32_LCD s hodnotou UNIVERSAL funguje pro znakové displeje, pokud je ale použit například grafický displej, je potřeba tuto proměnou změnit.

Přímo knihovna poskytuje funkce pro vytisknutí na displej, vymazání displeje, nastavení kurzoru a další. Tyto funkce nejsou volány přímo, ale jsou použity ve funkcích v souboru lcd.c, kde jsou vytvořeny funkce, které zobrazují informace o příslušných parametrech na LCD displej.

Například funkce pro informování o tvaru signálu se na základě vstupních argumentů rozhodne, zda se jedná o první nebo druhý oscilátor, a také jaký signál je aktuálně zvolený.

Obrázek 32 Informace o tvaru signálu na LCD displeji

Každá funkce před tisknutím znaků na displej nejprve původní obsah vymaže pomocí funkce MIOS32_LCD_Clear, následně se pak, pomocí funkce MIOS32_LCD_CursorSet, nastaví počátek prvního řádku, kdy obě souřadnice jsou rovny nule a následně se vytiskne pomocí funkce MIOS32_LCD_PrintFormattedString jméno části syntezátoru, ke které aktuální parametr patří. Na druhém řádku pak název parametru a jeho hodnota.

ZÁVĚR

Výstupem této práce je zvukový syntezátor implementovaný na vývojové desce STM32F4 Discovery. Syntezátor je implementován pomocí knihovny MIOS32, která zajišťuje především podporu ovladačů na vývojové desce a řízení syntezátoru pomocí MIDI protokolu.

Tato knihovna se ukázala jako poměrně dobrá volba, neboť podporuje pravděpodobně nejlepší možný mikrokontroler v kategorii podobně výkonných, protože přímo na desce jsou dostupné periferie pro audio výstup s velmi kvalitním D/A převodníkem i řízení pomocí USB. Díky tomu je tak možné vyvíjet syntezátor bez nutnosti připojování jakýchkoliv další periferií pro zobrazování či ovládání.

Přesto je zde k nalezení i několik nevýhod, zejména pak nemožnost debugovat aplikace pomocí na desce dostupného debuggeru, kvůli využití vlastního zavaděče, do kterého jsou aplikace nahrávány pomocí MIDI. Z počátku implementace se jevilo jako nevýhoda odstavení FPU jednotky, kvůli kompatibilitě s dalšími procesory podporovanými knihovnou, kvůli čemuž je nutné použít fixed point aritmetiku, avšak ve výsledku se pak tato technika ukázala jako výhoda, neboť fixed point operace jsou méně náročné na procesor i přes využití FPU. Osobně mi pak chyběla podpora pro I^2C sběrnice pro jednodušší připojení displeje, který byl využit pro lepší ovládání a kontrolu nad parametry syntezátoru.

Přes poměrně vysoký výkon použitého procesoru se ukázalo, že pro lepší výsledky by bylo vhodné rozdělit syntezátor na více modulů – tak, aby se jeden procesor staral o jednu činnost, například generování signálu, další o filtraci, efektovou jednotku a další. I tak ale syntezátor poskytuje poměrně velké možnosti v úpravě a prozkoumávání možností elektronických zvuků.

SEZNAM POUŽITÉ LITERATURY

1. KOPECKÝ, Pavel. *Základy elektronického zvuku a jeho kreativní zpracování*. 2. vydání. Praha: Nakladatelství Akademie múzických umění v Praze, 2017. ISBN 978-80-7331-431-6.
2. KLECL, Martin. *Modulární analogový syntezátor*. Technická 3058/10, 601 90 Brno, 2017. Bakalářská práce. Vysoké učení Technické, Fakulta elektrotechniky a komunikačních technologií.
3. Jak funguje syntezátor. *Elektronická Hudba* [online]. [cit. 2020-03-08]. Dostupné z: http://elektronicka-hudba.telotone.cz/clanky/syntezator
4. PEJROLO, Andrea a Scott B. METCALFE. *Creatingsoundsfromscratch: a practicalguide to music synthesisforproducers and composers* [online]. New York, NY: Oxford University Press, [2017] [cit. 2020-03-06]. ISBN 978-019-9921-904. Dostupné z: https://b-ok.cc/book/2922474/3e8e21
5. Metody zvukové syntézy. *Elektronická Hudba* [online]. [cit. 2020-03-09]. Dostupné z: http://elektronicka-hudba.telotone.cz/clanky/metody-zvukove-syntezy/
6. Basic Synthesizer. In: *Pro Audio Files* [online]. 2018 [cit. 2020-08-03]. Dostupné z: https://theproaudiofiles.com/wp-content/uploads/2018/09/Screen-Shot-2018-09-20-at-9.55.05-AM.png
7. SamplingSynthesis. *Center forComputerResearch in Music and Acoustics* [online]. [cit. 2020-03-11]. Dostupné z: https://ccrma.stanford.edu/~jos/jnmr/Sampling_Synthesis.html
8. KeyboardZone. In: *Wikiwand* [online]. [cit. 2020-08-03]. Dostupné z: https://upload.wikimedia.org/wikipedia/commons/thumb/c/cd/KeyboardZone.jpg/1280px-KeyboardZone.jpg
9. *Synclavier History Tour and The Quantum Potential With Christopher Currell* [online]. Encyclotronic [cit. 2020-07-27]. Dostupné z: https://encyclotronic.com/synthesizers/new-england-digital/synclavier-ii-r957/
10. *New England Digital Synclavier* [online]. Vintage Synth [cit. 2020-07-27]. Dostupné z: http://www.vintagesynth.com/misc/synclav.php

11. Synclavier. In: *Vintage Synth Explorer* [online]. Vintage Synth [cit. 2020-07-27]. Dostupné z: http://www.vintagesynth.com/sites/default/files/2017-05/synclavier.jpg

12. *Yamaha DX7 - historický test* [online]. Muzikus, 2005 [cit. 2020-07-27]. Dostupné z: http://www.muzikus.cz/pro-muzikanty-testy/Yamaha-DX7-historicky-test~21~leden~2005/

13. Yamaha DX7 Digital Programmable Algorithm Synthesizer. In: *KeyboardKountry.com* [online]. [cit. 2020-07-27]. Dostupné z: https://cdn11.bigcommerce.com/s-hqdnxgyg/images/stencil/1280x1280/products/8359/36475/DSC03464__06002.1574549773.JPG?c=2&imbypass=on

14. *Waldorf Blofeld: MINI Synth modul* [online]. Music Store, 2008 [cit. 2020-07-27]. Dostupné z: https://www.music-store.cz/recenze/waldorf-blofeld

15. Waldorf Blofeld. In: *Music Store* [online]. 2008 [cit. 2020-07-27]. Dostupné z: https://www.music-store.cz/sites/default/files/field_image/review/08_05_w_blofeld_1.jpg

16. HUBER, David Miles. *The MIDI manual*. 2nd ed. Carmel, Indiana: SAMS, c1991. ISBN 06-722-2757-6.

17. MIDI voicemessages. *RecordingBlogs* [online]. [cit. 2020-03-11]. Dostupné z: https://www.recordingblogs.com/wiki/midi-voice-messages

18. MIDI systemcommonmessages. *RecordingBlogs* [online]. [cit. 2020-03-11]. Dostupné z: https://www.recordingblogs.com/wiki/midi-system-common-messages

19. MIDI systemrealtimemessages. *RecordingBlogs* [online]. [cit. 2020-03-11]. Dostupné z: https://www.recordingblogs.com/wiki/midi-system-realtime-messages

20. UHLÍŘ, Jan a Pavel SOVKA. *Číslicové zpracování signálů*. Vyd. 2. přeprac. Praha: Vydavatelství ČVUT, 2002. ISBN 80-010-2613-2.

21. CHAMBERLIN, Hal. *Musical applicationsofmicroprocessors*. 2nd ed. HasbrouckHeights, N.J.: HaydenBook Co., c1985. ISBN 08-104-5768-7.

22. GroovyDSP: Transition Splice Oscillators. In: *Youtube* [online]. 29. 10. 2017 [cit. 2020-07-28]. Dostupné z: https://www.youtube.com/watch?v=dQ8I9F-uj3w

23. VÍCH, Robert a Zdeněk SMÉKAL. *Číslicové filtry*. Vydání 1., 2000. Praha: Academia, 2000. Česká matice technická (Academia). ISBN 80-200-0761-X.

24. Analog versus Digital: Co je tedy lepší? *Elektronická Hudba* [online]. [cit. 2020-03-23]. Dostupné z: http://elektronicka-hudba.telotone.cz/clanky/analog-versus-digital

25. TheMIDIboxOperatingSystem. *TheMIDIboxOperatingSystem* [online]. [cit. 2020-03-30]. Dostupné z: http://www.ucapps.de/mios.html

26. MIOS Studio 2. *TheMIDIboxOperatingSystem* [online]. [cit. 2020-03-30]. Dostupné z: http://www.ucapps.de/mios_studio.html

27. Goom. *Quinapalus.com* [online]. [cit. 2020-03-30]. Dostupné z: https://quinapalus.com/goom.html

28. *Teensy USB Development Board* [online]. PJRC [cit. 2020-08-03]. Dostupné z: https://www.pjrc.com/teensy/index.html

29. *Audio Library Development Roadmap* [online]. PJRC [cit. 2020-08-03]. Dostupné z: https://www.pjrc.com/teensy/td_libs_AudioRoadmap.html

30. *Mozzi: audio synthesis library for Arduino* [online]. [cit. 2020-08-03]. Dostupné z: https://sensorium.github.io/Mozzi/

31. *Trinity: Digital Synthesizer, Sequencer, Controller, Arpeggiator or Almost Anything* [online]. Bastl Instruments [cit. 2020-08-03]. Dostupné z: https://bastl-instruments.com/instruments/trinity

32. Trinity drum. In: *Trinity: Digital Synthesizer, Sequencer, Controller, Arpeggiator or Almost Anything* [online]. Bastl Instruments [cit. 2020-08-03]. Dostupné z: https://bastl-instruments.com/thumbs/instruments/trinity/drum_006_02-1000x997.jpg

33. STM32F4DISCOVERY: Discovery kit with STM32F407VG MCU * New order code STM32F407G-DISC1 (replaces STM32F4DISCOVERY). In: *ST life.augmented* [online]. [cit. 2020-08-03]. Dostupné z: https://www.st.com/en/evaluation-tools/stm32f4discovery.html

34. STM32F407G-DISC1 - Discovery kit with STM32F407VG MCU. In: *EVelta* [online]. [cit. 2020-08-03]. Dostupné z: https://cdn11.bigcommerce.com/s-

3fd3md1ghs/images/stencil/1280x1280/products/26563/7505/DISC1-Stm32f407__11873.1561068028.jpg?c=2&imbypass=on

35. VÁŇA, Vladimír. *ARM pro začátečníky*. Praha: BEN - technická literatura, 2009. ISBN 978-80-7300-246-6.
36. ARM® Cortex®-M4 with FPU. In: *Silicon Labs* [online]. Silicon Labs [cit. 2020-08-03]. Dostupné z: https://siliconlabs-h.assetsadobe.com/is/image//content/dam/siliconlabs/images/products/microcontrollers/32-bit_mcus/cortex-m4-chip-diagram.png?$LargeFullContentWidth$
37. Portable Audio DAC with Integrated Class D Speaker Driver. *Cirrus Logic* [online]. [cit. 2020-08-03]. Dostupné z: https://www.cirrus.com/products/cs43l22/
38. WhatIsan IDE? *CodeAcademy* [online]. [cit. 2020-04-08]. Dostupné z: https://www.codecademy.com/articles/what-is-an-ide
39. Eclipse (software). In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): WikimediaFoundation, 2001- [cit. 2020-04-08]. Dostupné z: https://en.wikipedia.org/wiki/Eclipse_(software)
40. STM32CubeIDE. *ST* [online]. [cit. 2020-04-08]. Dostupné z: https://www.st.com/en/development-tools/stm32cubeide.html
41. DUGGLE. Eclipse. *MIDIbox* [online]. Midibox, 2016 [cit. 2020-08-04]. Dostupné z: http://wiki.midibox.org/doku.php?id=eclipse
42. *MIOS32 Tutorial #017: A simple Sequencer* [online]. midibox, 2009 [cit. 2020-07-22]. Dostupné z: https://github.com/midibox/mios32/tree/master/apps/tutorials/017_sequencer
43. MIDIbox Hardware Platform, CORE_STM32F4 Module. *UCApps.de* [online]. 2020 [cit. 2020-07-24]. Dostupné z: http://www.ucapps.de/mbhp_core_stm32f4.html
44. Mbhp_core_stm32f4_standalone. In: *MIDIbox Hardware Platform, CORE_STM32F4 Module* [online]. 2020 [cit. 2020-07-24]. Dostupné z: http://www.ucapps.de/mbhp/mbhp_core_stm32f4_standalone.jpg
45. *MIOS32 Tutorial #011: Scanning 12 analog pots* [online]. midibox, 2009 [cit. 2020-07-24]. Dostupné z: https://github.com/midibox/mios32/tree/master/apps/tutorials/011_ain

46. *MBHP_CORE_STM32F4_V1* [online]. midibox, 2013 [cit. 2020-07-24]. Dostupné z: http://www.ucapps.de/mbhp/mbhp_core_stm32f4.pdf

47. *I/O Interface (Interrupt and DMA Mode)* [online]. GeeksforGeeks [cit. 2020-07-24]. Dostupné z: https://www.geeksforgeeks.org/io-interface-interrupt-dma-mode/

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

FM Frekvenčně modulační syntéza

Baud Rychlost přenosu informací – počet bitů za jednu sekundu

MIDI Music Instrument Digital Interface

FIR Finite Impulse Response

IIR Infinite Impulse Response

Hz Hertz, jednotka frekvence

USB Universal Serial Bus, univerzální sériová sběrnice

RTOS Real Time Operating System, operační systém reálného času

FPU Floating Point Unit, jednotka s pohyblivou desetinou čárkou

CC ControlChange, druh midi zprávy

A/D Převodník analogového signálu na digitální

D/A Převodník digitálního signálu na analogový

V Volt, jednotka napětí

RAM Random acces memory

RISC Reduced Instruction Set Computer, redukovaná instrukční sada

I^2S Inter IC sound, sériový komunikační protokol pro audio zařízení

OTG On The Go

JRE Java Runtime Enviroment, Java běhové prostředí

BPM Beats per minute, počet úderů za minutu

DMA Direct Memory Acces, přímý přístup do paměti

LCD Liquid Crystal Display

Ω Ohm, jednotka odporu

LFO nízkofrekvenční oscilátor

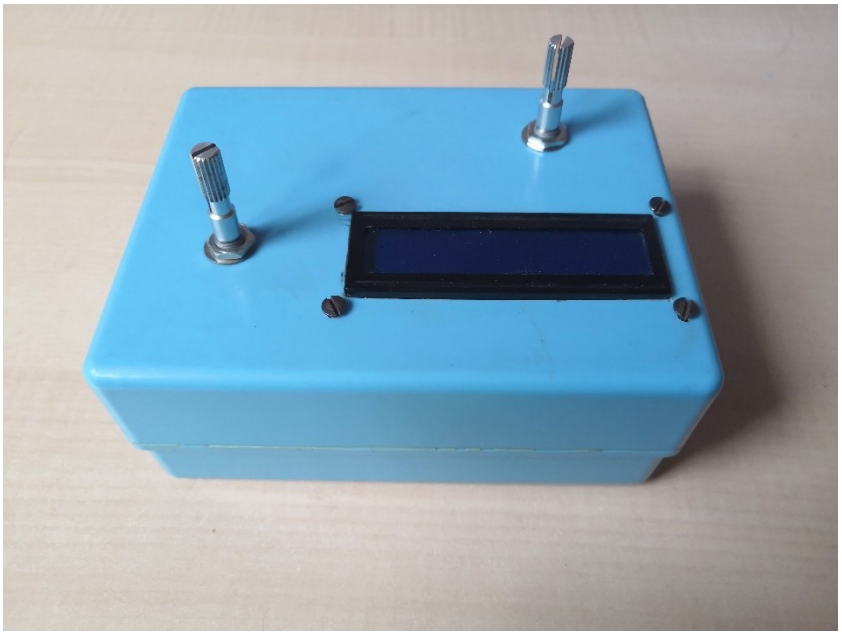
BLIT Band limited impulse train, pásmově omezený impulzní průběh

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM PŘÍLOH

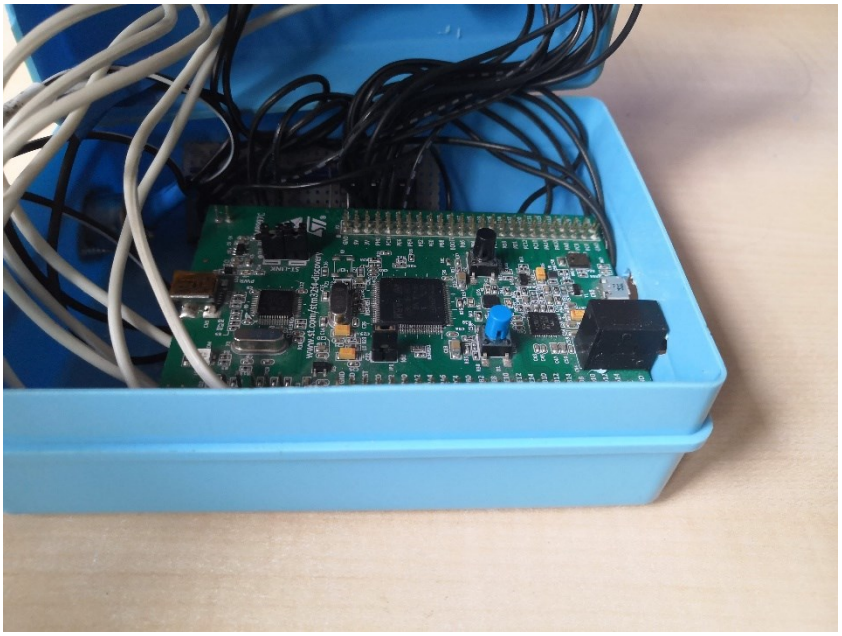
PŘÍLOHA P I: FOTODOKUMENTACE SYNTEZÁTORU



Pohled shora – na syntezátoru lze vidět ovládací prvky, pravý vrchní potenciometr přepíná mezi parametry, levý slouží jako datový vstup

Pohled zprava – DIN vstup pro MIDI

Pohled zleva - Vstup pro mikro USB zde může sloužit jako napajení i jako MIDI vstup



Pohled dovnitř - Uložení vývojové desky v krabičce