STRING INSTRUMENTS BASE TONE SIGNAL DETECTION AND ITS INTERPRETATION USING VIRTUAL STUDIO TECHNOLOGY BASED SOUND SYNTHESIZER

Bc. Ondřej Škrabánek

Master's thesis
2017

Tomas Bata University in Zlín
Faculty of Applied Informatics

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
akademický rok: 2016/2017

ZADÁNÍ DIPLOMOVÉ PRÁCE
(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Ondřej Škrabánek**
Osobní číslo:	**A14486**
Studijní program:	**N3902 Inženýrská informatika**
Studijní obor:	**Informační technologie**
Forma studia:	**kombinovaná**
Téma práce:	**Detekce základního tónu signálu strunných nástrojů a její interpretace pomocí syntezátoru v technologii Virtual Studio Technology**
Téma anglicky:	**The Detection of the Base Tone Signals of String Instruments and Their Interpretation Using a Virtual Studio Technology-based Sound Synthesizer**

Zásady pro vypracování:

1. **Popište algoritmy detekce základního tónu signálu a vyberte algoritmus vhodný pro hudební signály (basa a kytara).**
2. **Algoritmus optimalizujte pro zpracování signálu v reálném čase a implementujte jako plug-in modul technologie VST.**
3. **Implementujte oscilátor s digitálním řízením výšky tónu z detektoru základního tónu, digitálně řízený filtr a zesilovač, které budou zpracovávat signál oscilátoru.**
4. **Popište algoritmy detekce začátků a konců tónů (takzvaný onset detektor) pracující v reálném čase a implementujte zvolený algoritmus v plug-in modulu.**
5. **Implementujte dva generátory obálky ADSR řízené onset detektorem pro řízení filtru a zesilovače.**
6. **V plug-in modulu doplňte druhou větev signálu pro filtrování a řízení zesílení vstupního signálu namísto signálu oscilátoru.**

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

Seznam odborné literatury:

1. **OPPENHEIM A. V., WILLSKY A. S.: Signals and Systems, Prentice Hall, New Jersey, 1997.**
2. **VÍCH R., SMEJKAL Z.: Číslicové filtry, Academia, 2000.**
3. **ZAPLATÍLEK K., DOŇAR B.: Matlab – začínáme se signály, BEN, 2006.**
4. **DINIZ P. S., da SILVA E., NETTO S. L.: Digital Signal Processing, Cambridge University Press, 2010.**
5. **NEUKOM M.: Signals, Systems and Sound Synthesis, Peter Lang, 2013.**

Vedoucí diplomové práce:	**doc. Ing. Marek Kubalčík, Ph.D.** Ústav řízení procesů
Datum zadání diplomové práce:	**3. února 2017**
Termín odevzdání diplomové práce:	**16. května 2017**

Ve Zlíně dne 3. února 2017

UNIVERZITA TOMÁŠE BATI VE ZLÍNĚ
Fakulta aplikované informatiky
L.S.
①

[illegible]

doc. Mgr. Milan Adámek, Ph.D.
děkan

[illegible]

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

Jméno, příjmení: Ondřej Škrabánek

Název bakalářské/diplomové práce: String Instruments Base Tone Signal Detection and its Interpretation Using Virtual Studio Technology Based Sound Synthesizer

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 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/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 15. 5. 2017

[illegible]
....
podpis diplomanta

ABSTRAKT

Tato práce se zabývá analýzou zvukového signálu kytary či basy za účelem získání základní frekvence tónu v reálném čase. Pro tento účel jsou zde zkoumány a porovnávány vlastnosti a vhodnost různých algoritmů.

Praktická část řeší implementaci plug-inu v technologii VST, jež produkuje signál s výškou tónu vstupního signálu užitím nejvhodnějšího algoritmu pro extrakci základní frekvence s možností modulace výstupního signálu.

Klíčová slova: VST plug-in, detekce výšky tónu, základní frekvence.

ABSTRACT

This thesis deals with guitar and bass sounds analysis in order to gain fundamental frequency in real-time. For this case, I examine and compare properties and suitability of different algorithms.

The practical part deals with the implementation of a plug-in in VST technology, which produces signal on input signal's fundamental frequency with further options for output signal modulation.

Keywords: VST plug-in, pitch detection, fundamental frequency.

STATEMENT

I hereby declare that this thesis is my own work and effort led by doc. Ing. Marek Kubalčík, Ph.D. I was provided further information by Dr. Jiří Schimmel. Where other sources of information have been used, they have been acknowledged.

I also declare that submitted master's thesis and the electronic version loaded to IS/STAG are identical.

ACKNOWLEDGEMENT

I would like to express my deep gratitude to my master thesis advisor doc. Ing. Marek Kubalčík, Ph.D. and Dr. Jiří Schimmel for their assistance in writing this thesis as well as collecting information. I also thank to prof. Dr.-Ing. Udo Zölzer for providing me his algorithm and related technical materials.

CONTENT

INTRODUCTION

The sound is all around us. Every day from morning when the sound of alarm wakes us, we listen to radio on our way to work, when we converse with someone or when we fall asleep to the sound of wind floating through the trees outside the window.

Even though human do not have this sense so well developed as many other species, it can be trained and enhanced to a certain extent. People, who deal with music and sounds, such as singers, musicians, sound designers, have usually highly developed or trained hearing so they are able to train their ears to recognize tiniest changes, notes or frequencies that an ordinary person would not be able to even notice.

As the progress in computer science started to grow more and more faster, it did not miss the field of signals including sound. Common analogue devices, such as amplifiers, radios, TVs are these days processing the signal digitally, tapes were replaced by CDs or MP3 and music editor software are widely available on the internet either paid or as a freeware and can be used even by less experienced users. However digital sound processing (DSP) can ease and speed up work especially to musicians, who work with signals on a daily basis. Advanced music software can help them not even to record and edit the existing sound, but also create new sounds using different oscillators, modulations and effects, which could not be done in the analogue way.

Music carries huge amount of information. Some of them can be easily obtained without any complex analysis, however some of them require deeper and more complex approach to reach desired information. There are still many unknown practices in the area of music analysis. It is still great unknown, how the human brain processes the sound. While the brain can extract required data practically immediately, it is not so easily achievable this in DSP. There have been proposed many solutions for music extraction. There are thought often limitations in accuracy or they can only be used on pre-recorded sound.

This thesis aims at obtaining of some of the sound information, particularly extraction of fundamental frequency and beginning and end of the guitar or bass tone respectively, while the emphasis is placed on speed of pitch detection algorithms and their usage in real-time. As next goal is to generally dive in the problematics of digital signal processing. In this field I focus especially the one used for sound processing, particularly on the digital filters.

Practical part deals with implementation of the algorithm as the VST plug-in for analysis and modulation of monophonic guitar or bass signal respectively. This can be further used for helping musicians in performances or as a part of more complex system. Such fundamental frequency is then used in oscillator to create new sound with different types of wave and further process the signal with amplifier and filter. Implemented algorithms are proven in Matlab.

I. THEORETICAL PART

1. MUSIC THEORY

First of all, it is important to understand the basics of the music concept. Music is made of individual notes that each has attributes, which human can perceive naturally, such as *loudness*, *timbre*, *duration* and *pitch*. These properties are together called psychological. On the other hand, they are difficult to measure in computer, so there are also physical properties of the sound, such as *pressure*, *frequency*, *spectrum*, *duration* and *envelope*, which can be quite well measured, but are more problematical for human perception. These properties are called physical. Yet there is strong relation between both groups.

		Subjective quality			
		Loudness	**Pitch**	**Timbre**	**Duration**
Physical property	**Sound pressure**	strong	weak	weak	weak
	Frequency	weak	strong	weak	weak
	Spectrum	weak	weak	strong	weak
	Duration	weak	weak	weak	strong
	Envelope	weak	weak	moderate	weak

Table 1 Relations among physical properties and subjective quality

1.1. Psychological properties

Psychological quality also referred to as subjective quality of sound is set of sound properties perceived by human.

Duration determines how long a certain note lasts, and is measured in "beats" or fractions of beats.

Pitch is another human subjective perception of sound. It is how we perceive tone height in comparison with another one. Western music divided pitch into 12 different semi-tones together creating so called octave. While pitch is logarithmic, corresponding semi-tones in octaves have frequency multiplied with 2. So if we take the 49th tone (counted on 88-key piano keyboard) pitch standard A with frequency 440 Hz, we can easily compute frequency of any other note using the following equation.

$$f(n) = 2^{\frac{n-49}{12}} \cdot 440\ [Hz] \tag{1}$$

Loudness refers to the amplitude of the sound so basically determines how strong the sound is. It is measured in phons and is often confused with sound pressure. There is non-linear relation between loudness and sound pressure as shown in Figure 1. [1]

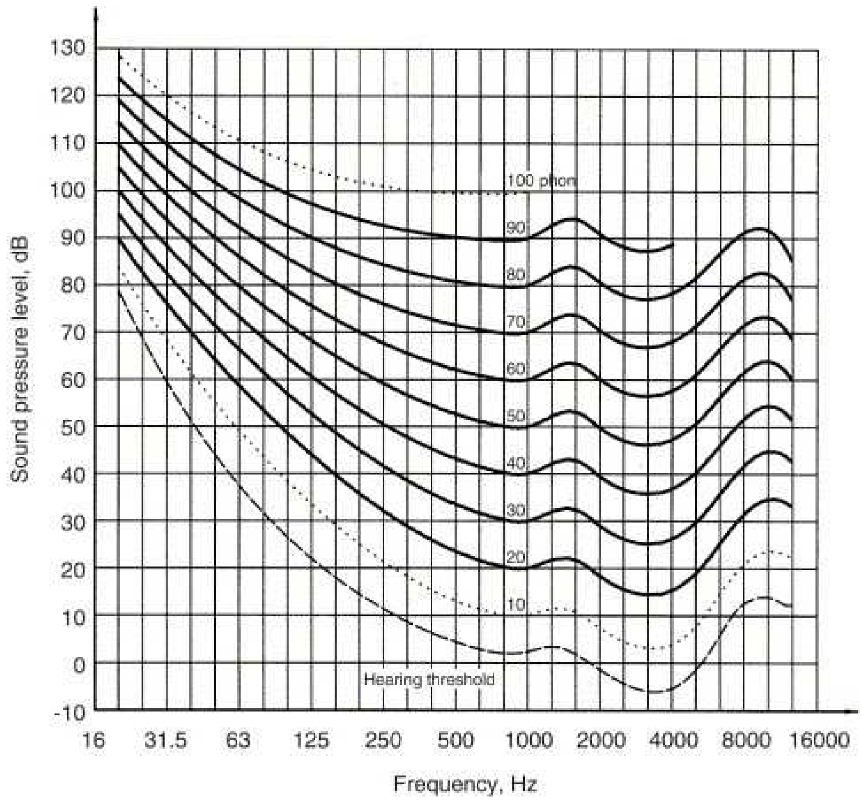


Figure 1: Relation between loudness and sound pressure [2]

Timbre is often referred to as the "colour" of the note. For each instrument is unique and cannot be measured. Even though the pitch is similar, the sound is different. It is given by the additional harmonics with various amplitudes, envelope, attack transients, vibrato, tremolo and others, which depends not only on the material the instrument is made of and the instrument shape, the way it is played, but also on the outer atmospheric conditions. While timbre is easy for humans to distinguish, it is quite difficult to do the same thing using computers, because the principle of human perception of sound is not yet understood. While in computer we are able to distinguish the frequencies, amplitudes and other sound characteristics, it is still mystery, how human are able to perceive the harmonies as one sound even in polyphonic[1] tone. This is an important problem in psychoacoustics[2].

[1] Polyphony represents more tones playing simultaneously

[2] Psychoacoustics studies the relationship between acoustic sound signals, the auditory system physiology, and the psychological perception to sound, in order to explain the auditory behavioral responses of human listeners, the abilities and limitations of the human ear, and the auditory complex processes that occur inside the brain.

1.2. Physical properties

Sound pressure is measured in dB as ratio between the measured pressure and the reference one. Our ears respond to extremely small sound pressure fluctuation. The auditory threshold is used as the reference sound pressure.

$$p_0 = 2 \cdot 10^{-5} Pa \tag{2}$$

Using the reference value and measured sound pressure, the sound pressure level can be obtained using following equation comparing these two squared values in logarithmic scale.

$$L_P = \log_{10}\left(\frac{p^2}{p_0^2}\right)[B] \tag{3}$$

After adjusting the equation and converting to dB units, we get the equation of sound pressure level.

$$L_P = 20 \cdot \log_{10}\left(\frac{p}{p_0}\right)[dB] \tag{4}$$

Below can be seen table of typical sounds and their sound pressure levels.

Jet aircraft, 50m away	140 dB
Threshold of pain	130 dB
Threshold of discomfort	120 dB
Chainsaw, 1m distance	110 dB
Disco, 1m from speaker	100 dB
Diesel truck, 10m away	90 dB
Urban street	80 dB
Vacuum cleaner, distance 1m	70 dB
Conversational speech, 1m	60 dB
Average home	50 dB
Quiet library	40 dB
Quiet bedroom at night	30 dB
Background in TV studio	20 dB
Rustling leaves in the distance	10 dB
Hearing threshold	0 dB

Table 2: Typical sounds and their sound pressure levels

Frequency of the tone is meant as the fundamental frequency. This frequency is the most important and creates the pitch of the sound.

Spectrum of the sound consists of all frequencies the sound is made of. Even though human can perceive sound as one pitch, it may be composed of many various frequencies. Basically it is fundamental frequency, harmonics, inharmonic frequencies[3] and noise[4].

Harmonic frequencies are integer multiples of fundamental frequency. Easily calculable according to following equation, where n is integer:

$$f = n \cdot f_0 [Hz] \tag{5}$$

For example, the note *A3* has fundamental frequency of 220 Hz. The harmonic frequencies occur at 440 Hz, 660 Hz, 880 Hz, etc. Other than fundamental frequencies difficult the analysis, because the fundamental frequency does not always have the biggest magnitude and the first harmonics of the note (in this case 440 Hz) also represent the fundamental frequency of the note one octave higher (in this case *A4*).

Duration is in physics measured time (usually in milliseconds) stating how long the sound resonates.

Envelope is both a physical and psychological property. There are different types of articulations of tones in music terminology such as pizzicato[5], legato[6], staccato[7] and many others, which are given by the playing technique. These techniques are very closely connected with the envelope (for more details about envelope see 3.1.2).

1.3. Guitar and bass

Standard guitars have six strings and usually at least 21 frets. Strings are numbered from the right to the left when looking directly at the fretboard. The first string has the highest pitch and the sixth string has the lowest pitch. Individual notes are played by pressing down on a string above a certain fret and picking (plucking) the string. As the frets get closer to the body of the guitar, the pitch of its corresponding note increases in semi-tone intervals.

One of the characteristics of the guitar is that there are many different ways to play the same note. In standard tuning, the lowest note of the guitar is the *E*, *E2* to be precise, on the open sixth string. Five frets up there is an *A* note which sounds the same as the open note on the fifth string. These notes that have the same pitch but different locations are said to be enharmonic. Some notes may occur as many as five times throughout the neck. The notes of the open strings from the sixth

[3] Inharmonic frequencies are such frequencies, which are not integer multiples of fundamental frequency

[4] Noise in acoustics stands for unwanted sounds, which cause interference of original sound. However, noise in music is usually part of the sound timber.

[5] Pizzicato is playing technique that involves plucking the string of a string instrument.

[6] Legato is playing technique during which the notes are played long and continuously as they were connected.

[7] Staccato designated notes are played shortly. It is opposite to legato.

string to the first string in standard tuning are *EADGBe* with the capital E being two octaves lower than the lowercase *e*. At standard acoustic guitar has the lowest possible note is *E2*, the highest is *D6*, on the other hand, bass has the lowest tone *E1* and the highest *F4*. In addition to standard tuning, guitarists may prefer to use an alternate tuning, where strings are tuned different to notes, for example *DADGAD*. This changes the location of notes throughout the guitar neck, thereby creating different technique for playing the instrument, such as different hand and finger positions for making chords.

To actually produce the sound, acoustic guitars have a hollow body where the vibrations from the strings resonate. In contrast, electric guitars use magnetic coils at the bridge of the guitar to translate the vibrations into an electrical signal that is transmitted to an amplifier. Some acoustic guitars also have a microphone preamp built into them, following them to connect to an amplifier as well. [3]

2. DIGITAL SIGNAL PROCESSING

In this chapter we focus on signals in digital world. How to obtain data from real world, transform them to digital form, process them and reproduce them back.

2.1. Digital signals

Sound is made up of energy in the form of waves. Nowadays the technology doesn't allow us to capture the signal in continuous form, so the state of the signal has to be transferred to discrete form. The process of capturing waveform is called sampling.

2.1.1. Sampling

During this process the analogue signal $x(t)$ is discretized by analogue-to-digital converter (ADC).

The ADC performs sampling in fixed time intervals into a series of samples that describe the amplitude of the waveform at each segment of time. The number of samples received per second is called the *sampling rate* or *sampling frequency*. Sampling interval between two adjacent samples can be calculated using

$$T_s = \frac{1}{f_s} \tag{6}$$

A common sampling rate for music is 44100 samples per second and is the standard for CD quality audio. This value was chosen to prevent aliasing of the signal.

2.1.1.1. Aliasing

Aliasing is a phenomenon, that describes the situation when the signal is converted or displayed with imperfection causing misinterpretation or perceptible difference between original and processed signal.
To prevent from aliasing we have to follow the Nyquist-Shannon sampling theorem[8],

$$f_s \geq 2f_{max} \tag{7}$$

that says the sampling rate must be at least twice as high as the highest frequency to avoid the aliasing.
For example, if we apply this equation to frequency 22050Hz, which is generally stated as the maximum frequency human ear is able to perceive, we obtain the minimum sample rate must be 44100 Hz so the undersampling is prevented.

[8] Nyquist-Shannon sampling theorem was independently discovered by Harry Nyquist, Claude Shannon, E. T. Whittaker, Vladimir Kotelnikov, so the theorem is also known as a composition of their last names.

Undersampling is phenomenon, which occurs if the Nyquist sampling theorem is violated. In such case the signal is misinterpreted with another signal.

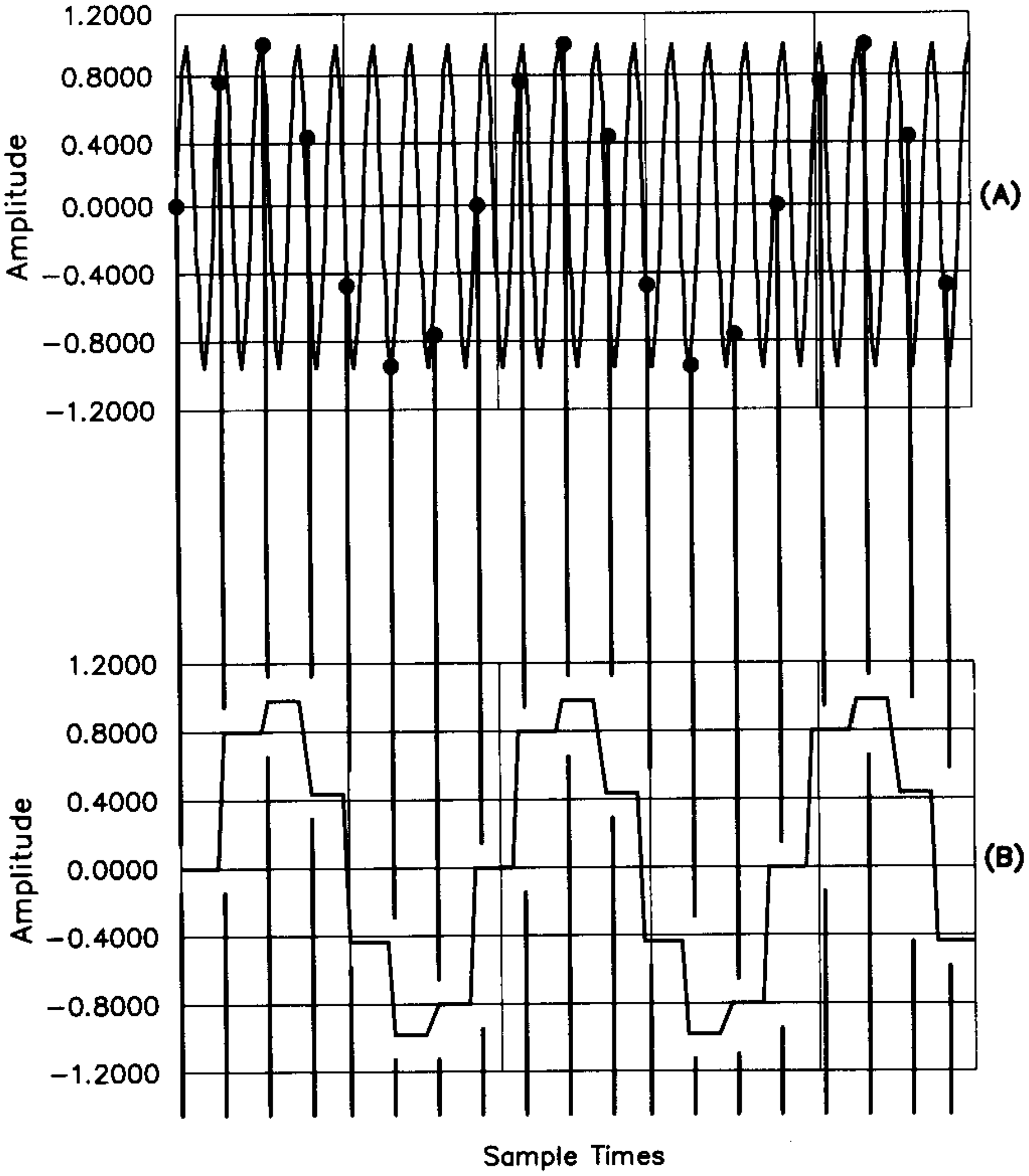


Figure 2: (A) Sine wave of frequency greater than sampling frequency.
(B) Harmonically sampled sine wave. [4]

Oversampling is the opposite phenomenon, which states the signal has been sampled with frequency higher than the frequency stated by the Nyquist sampling theorem. This improves resolution, reduces noise and helps avoid aliasing and phase distortion by relaxing anti-aliasing filter performance requirements. On the other hand it's performance is decreased and takes more resources for sampling and saving the signal. The signal oversampling factor is N where N states how many times the sampling frequency is higher than Nyquist rate.

2.1.2. Quantization

As the samples have been obtained, the value has to be converted to digital data. This process is part of ADC and is called **quantization**. ADC measures the value of the signal in given frequency and this value is then stored as the velocity of the signal in exact time. The value is equally distributed to assigned word, where the accuracy is highly dependent on the word-size of the value. For example, we have signal with limit values from 0V to 1V. 3-bit word provides 8 values with resolution 125mV whereas the 4-bit word provides 16 values with resolution 62,5mV and 5-bit word with 32 values can reach resolution of 31,25mV.

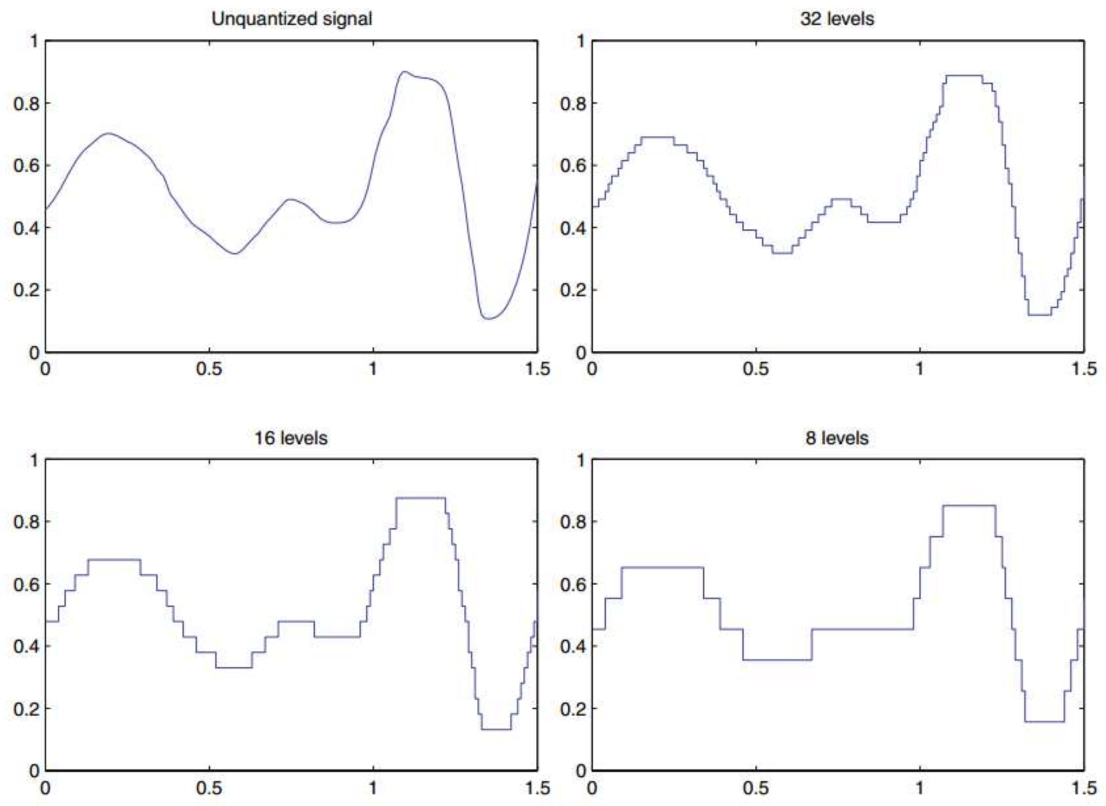


Figure 3: Example of analogue signal quantization with different word size [5]

To process this data for spectral analysis, computers typically take a window of samples (for example, 1024 samples), do desired processing, and then wait for the next window. Window size is important as larger windows contain more information about the signal but are received less often than smaller windows. With a window size of 1024, windows will be received every 23 milliseconds, whereas 4096-sample windows would be received every 93 milliseconds.

However, there is solution to this problem using principles of sliding window with a hop size parameter. Last input windows are stored and used for completing upcoming windows. For example, if the actual buffer size is 512 samples, the computer can save the last few windows and do calculations on a larger window of say 4096 samples. This introduces overlap which can help reduce artefacts and improve resolution in the frequency domain. On the other hand, larger window covers larger time period, so the changes in frequency domain will be more delayed. [3]

2.1.3. Causality

Causality in digital signals defines that the output *y(n)* at time *n* depends only on the current input *x(n)* at time *n* and its past input sample value such as *x(n-1)*, *x(n-2)*,.... Simply put, causal system reacts only to its past and present input. System, which react to future input values, such as *x(n+1)*, *x(n+2)*,..., the system is noncausal. This terms are very closely related to real-time processing, because noncausal systems cannot be realized in real time.

2.2. Fourier Transform

Fourier transform is one of the basic building blocks of DSP. Within DPS it is used to obtain information about frequency spectrum of a signal and vice versa. It is defined as follows

$$F(\omega) = \int_{-\infty}^{\infty} g(t) e^{-2\pi i \omega t} dt \tag{8}$$

and inverse

$$g(t) = \int_{-\infty}^{\infty} F(\xi) e^{2\pi i \omega t} d\xi. \tag{9}$$

For discrete signals there is also discrete Fourier transform

$$F(k) = \sum_{n=0}^{N-1} f(n) e^{\frac{-2\pi i k n}{N}} \tag{10}$$

and inverse discrete Fourier transform

$$f(n) = \frac{1}{N} \sum_{n=0}^{N-1} F(k) e^{\frac{2\pi i k n}{N}} \tag{11}$$

2.2.1. Window functions

When we apply Fourier transformation to the sampled data, we theoretically expect following assumptions:

- Sampled data are periodic to themselves
- Sampled data are continuous to themselves and band limited to folding frequency

The second assumption is often violated, which leads to undesired harmonic frequencies in final spectrum. This unwanted effect is called **spectral leakage**. We cannot fully prevent it, unless abide both assumptions stated above. But we can alleviate this using **window functions**.

Applying window function to the data sequence creates new data sequence with beginning and end heading to zero, which soften transition between them and minimize the spectral leakage.

Common window functions are listed as follows:

- The rectangular window (no window function):

$$w_R(n) = 1, \tag{12}$$

- The triangular window:

$$w_{tri}(n) = 1 - \frac{|2n-N+1|}{N-1}, \tag{13}$$

- The Hamming window:

$$w_{hm}(n) = 0.54 - 0.46\cos\left(\frac{2\pi n}{N-1}\right), \tag{14}$$

- The Hanning window:

$$w_{hn}(n) = 0.5 - 0.5\cos\left(\frac{2\pi n}{N-1}\right), \tag{15}$$

- The Bartlett window:

$$w_b(n) = \frac{2}{N-1}\left(\frac{N-1}{2} - \left|n - \frac{N-1}{2}\right|\right), \tag{16}$$

- The Bartlett-Hann window:

$$w_{bh}(n) = 0.62 - 0.48\left|\frac{n}{N-1} - \frac{1}{2}\right| - 0.38 \cdot cos\left(\frac{2\pi n}{N-1}\right), \tag{17}$$

- The Blackman window:

$$w_{bm}(n) = \frac{1-\alpha}{2} - \frac{1}{2}\cos\left(\frac{2\pi n}{N-1}\right) + \frac{\alpha}{2} cos\left(\frac{4\pi n}{N-1}\right),$$

$$\alpha = 0.16 \tag{18}$$

- The Cosine window:

$$w_c(n) = \cos\left(\frac{\pi n}{N-1} - \frac{\pi}{2}\right) = sin\left(\frac{\pi n}{N-1}\right), \tag{19}$$

- The Lanczos window:

$$w_l(n) = \text{sinc}\left(\frac{2n}{N-1} - 1\right), \tag{20}$$

- The Gaussian window:

$$w_g(n) = e^{-\frac{1}{2}\left(\frac{n-(N-1)/2}{\alpha(N-1)/2}\right)^2},$$

$$\alpha \leq 0.5 \tag{21}$$

The n is defined in interval $0 \leq n \leq N-1$

2.3. Key DSP operations

There are plenty of DSP algorithms. However, all these algorithms, including the most complex, require similar basic operations.

2.3.1. Convolution

Convolution is one of the most frequently used operation in DSP. It is a mathematical operation on two functions creating third function, that is typically viewed as a modified version of one of the original functions. The definition of continues convolution is

$$(f * g)(t) \overset{\text{def}}{=} \int_{-\infty}^{\infty} f(\tau)\, g(t-\tau)d\tau \tag{22}$$

However, in DSP we work with discrete signals, so we need discrete convolution. Given two finite and causal sequences, x(n) and h(n), of length N_1 and N_2, respectively, their convolution is defined as

$$y(n) = h(n) * x(n) = \sum_{k=-\infty}^{\infty} h(k)x(n-k) = \sum_{k=0}^{\infty} h(k)x(n-k),$$

$$n = 0, 1, \dots, (M-1) \tag{23}$$

where the symbol $*$ is used to denote convolution and $M = N_1 + N_2 - 1$.

2.3.2. Auto-Correlation

Autocorrelation refers to the correlation of a time series with its own past and future values.

$$R_{ff}(\tau) = \left(f(t) * \bar{f}(-t)\right)(\tau) = \int_{-\infty}^{\infty} f(t)\bar{f}(t-\tau)dt \tag{24}$$

Positive autocorrelation might be considered a specific form of “persistence”, a tendency for a system to remain in the same state from one observation to the next. Autocorrelation complicates the application of statistical tests by reducing the number of independent observations. Autocorrelation can also complicate the identification of significant covariance or correlation between time series (e.g., precipitation with a tree-ring series). Autocorrelation can be exploited for predictions: an autocorrelated time series is predictable, probabilistically, because future values depend on current and past values.

Autocorrelation can be also estimated using FFT using Wiener-Khinchin theorem as follows

$$R(\tau) = \int_{-\infty}^{\infty} S(\omega)e^{j2\pi\omega\tau}\, df \tag{25}$$

$$S(\omega) = \int_{-\infty}^{\infty} R(\tau)e^{-j2\pi\omega\tau}\, d\tau \tag{26}$$

2.4. Pitch Detection

"We can define pitch detector (PD) or pitch estimator as a software or hardware device that takes sound signal as input and attempts to determine the *fundamental pitch period* of that signal. That is, it attempts to find the frequency that a human listener would agree to be the same pitch as the input signal (assuming there is one such pitch)." [6]

The pitch detection can be used only under certain conditions on limited set of sounds. Not all sounds have clear pitch, such as cymbal crash, brief impulse, or complex sound masses, so it would be pointless to determine pitch.

The principles of human pitch perception are not fully understood. Human can perceive pitch even in very noisy environment, follow multiple pitches simultaneously and also detect tiniest pitch changes, such as vibrato[9]. In such case, the goal of PD is to filter the signal from noisy surroundings, recognize multiple different tones and locate the centre pitch respectively.

In summary, the requirements for PD are quite high and very much depend on current scenario. In some cases, it is desired to be precise and accurate, in others we need the pitch detector to tolerate small deviations.

There are many methods of detecting pitch. In this thesis I focus only on the real-time ones.

2.4.1. Difficulties and Problems

As mentioned before, pitch detection is not easy and straightforward process and it is accompanied by many problems.

If the tones are played in legato or just in rapid sequence, the **tone transition** can cause make analysis unclear and short-time PD confusion.

Attack transient is high amplitude, short-duration sound at the beginning of the waveform. This can cause confusion during detection process. Detailed analysis of the waveform of many instruments reveals chaotic and instable attack. If a fundamental frequency is present in the attack, it is probably obscured by noise and inharmonic partials. For example, with string instruments it is caused by the initial finger motion until the string find and settle in the stable pitch. With some instruments this may take 100 ms or more. During this period the detection is impossible.

PDs using spectrum analysis algorithms usually have problems with acquiring **low frequencies**. It is useful to combine such PD with time-domain one. It is difficult for any PD to identify low pitches in real time. In order to determine the low-frequency fundamental pitch, at least three cycles of the steady-state waveform should be sampled before the analysis can begin. For example,

[9] Vibrato is musical effect of regular, pulsating changes of pitch of the vocal, string, or wind instruments.

an A at 55 Hz, three cycles take 54 ms to sample. If such sound contains also the attack transient, adding the delay of PD algorithm makes the evaluation delay inevitable.

As the low frequency pitch cause problem to frequency-domain algorithms the **high frequencies** can also cause problems to some real-time PDs in time-domain. As the frequency rises, one pitch period is represented by fewer samples. The resolution with which pitch can be determined in the time domain is directly affected by the length of the pitch period or the number of samples of delay used in comparing a signal with its past.

All PDs start with an analysis of narrow time segment that lasts from about 20 to 50 ms. In contrast, human pitch perception is not so time-localized. Since PDs work only from local details they may myopically track irrelevant details that were produced unintentionally, such as unsteadiness at the beginning of a note or excessive vibrato.

The **acoustical ambience** within which a sound is heard affects the PD accuracy. A closely miked and compressed studio recording may exaggerate incidental playing noises, such as bow scraping, key clicking, or breathing sounds, overlapping the signal heard by the PD. By contrast, tones bathed in reverberation and echoes smear early notes over the start of new notes. Provided that the analysis is carried out in non-real time, an attempt at ambience removal may help the PD. [6]

2.4.2. Time-Domain Methods

There are two main approaches of detecting the pitch of a sound. The first is by analysing information in the *time domain* (the explicit data contained in the samples), and other is by analysing the frequency domain (the distribution of the different frequencies spread throughout the frequency spectrum). Fundamental period methods look at the input signal as a fluctuating amplitude in the time domain, like the signal that appears on the screen of an oscilloscope. They try to find repeating patterns in the waveform that give clues as to its periodicity. Perhaps a more suitable term for these types of pitch detectors would be "periodicity detector". [3] [6]

In the time domain, the note envelope is the shape of the amplitude or loudness of a musical note throughout its entire duration. The envelope is made up of four main parts. The first part, called the attack, is when the note is first picked, and can be seen as a sharp increase in amplitude from zero to the peak. After the attack, the decay is the period during which amplitude falls down from the peak to the main portion of the note called the sustain. During the sustain, the note gradually falls until the release, when the note is silenced or becomes inaudible. By paying attention to the relative amplitudes, note onsets can be inferred by sudden increases in amplitude. While this can suggest timing and temporal location of a note, it does not show us which pitch or how many notes are being played at a time. [3]

Pre-processing by filters may improve the accuracy of time domain PDs and make them very accurate, especially in lower frequencies, where frequency-domain PDs cannot reach such accuracy. For example, in case of zero-crossing algorithm the lowpass filter can significantly reduce higher frequencies, which can cause multiple crossing of the axis in one crossing point.

2.4.2.1. Simple time-domain methods

One type of pitch detector tries to find periodicities in the waveform by looking for repeating zero-crossings. A **zero-crossing** is a point where the waveform's amplitude goes from positive to negative or vice versa. For example, a sine wave crosses the zero amplitude threshold at the middle and end of its cycle. By measuring the interval between zero-crossings and comparing successive intervals, the PD infers a fundamental frequency.

A variation on zero-crossing detection is peak rate detection, involving measuring the distances between peaks. In general, zero-crossing and peak PDs are relatively simple and inexpensive, but they are also less accurate than more elaborate methods. This is because other frequencies that are not the pitch frequency may also generate waveforms that cross the zero point or exhibit peaks.

2.4.2.2. Autocorrelation Function

One popular time domain algorithm is called autocorrelation. It looks for the highest peaks of each wave in the sample window, and measures the distances between them. With these distances, the period of the wave can be inferred, and the frequency can be detected. This method works well for individual pitches or notes, but when there is more than one note sounding at a time, the algorithm is unable to classify them.

Autocorrelation is not so fast as previous time-domain methods, because it involves applying autocorrelation to input signal. Unlike the methods above having linear time complexity, the fastest known autocorrelation algorithm has time complexity $\theta(N \cdot \log(N))$ (see 2.3.2).

2.4.3. Frequency-Domain Methods

To analyse a sample window in *the frequency domain* (FD), a common technique is to run the samples through a function called the Fourier transform. This function takes in an array of amplitudes at discrete points in time, and outputs an array of frequency bins that contain the magnitude and phase of different parts of the frequency spectrum. By locating the bins with the largest magnitudes, it is possible to see which frequencies are present. In the case of a guitar note, the bin containing the fundamental frequency as well as any bins containing the frequencies of any harmonics will have significant magnitudes. [3]

As mentioned previously, the Fourier transform, (specifically the discrete Fourier transform) takes in a vector of audio samples and produces a vector of frequency bins with both magnitude and phase information corresponding to different ranges of the spectrum.

$$c_k = \frac{1}{N}\sum_{n=0}^{N-1} x(n)e^{-i\frac{2\pi kn}{N}} \qquad -\infty < k < \infty \tag{1.27}$$

Because calculating the DFT is on the order of $\theta(N^2)$, a fast Fourier transform (FFT) is used instead to calculate the same result in $\theta(N.\log(N))$ operations. When the FFT is taken repeatedly over short periods of time (often using overlapping windows), the result is called a short-time Fourier transform or STFT. The STFT can be visualized to show the change in local frequencies over time. [3]

The Fourier transform result gives us information about equal spaced frequency bands from 0 Hz up to the Nyquist frequency which is equal to half of the sampling rate. The number of useful bands is equal to half of the number of samples in the input window. So, for example, if we are using a window of 1024 samples and using a sampling rate of 44.1 kHz, we will end up with 512 frequency bands spanning the total range of frequencies between 0 and 22050 Hz. [7]

In addition to there being many notes with lots of harmonics, different notes may have overlapping harmonics. For example, the first four harmonics of the note A2 are 110, 220, 330 and 440. For E2, they are 82.4, 164.8, 247.2, and 329.6. At 44.1 kHz, it would take a very long window size to compute an FFT with high enough resolution for 329.6 and 330 to end up in separate bins. As more notes are sounding simultaneously, there is a larger chance of harmonic overlap. Also, this shows that bandwidth of the FFT bins is important. For example, with a window size of 1024 samples, the bandwidth of FFT bins will be 43.1 Hz. The lowest notes on the guitar have fundamental frequencies that differ by less than 10 Hz, so at this resolution, many of the first harmonics will overlap with each other, making it difficult to distinguish between notes.

Another issue with the FFT output is that during the initial attack, there is a lot of noise associated when the finger or pick strikes the strings. Not until the actual sustain are frequency bins generally representative of the true harmonics of the sounding notes. While this attack period is very short, it must be taken into account. [3]

Before FFT is computed, we apply the Hanning windowing function to the sample buffer. Window functions are used to scale the samples in the time domain, and help to reduce edge effects that cause spectral leakage in the resulting FFT. [3]

2.4.3.1. Strongest frequency

The easy and fast method of finding the pitch is to select the strongest frequency as the fundamental. More sophisticated way might be find harmonics and subtracting adjoining harmonics will lead to the fundamental frequency. Depending on sound nature and spectral distribution of harmonics power the strongest frequency does not always have to be fundamental. This can be enhanced by autocorrelation of the signal. This method increases the power of lower frequencies and suppress he high ones. Applying Wiener-Khinchin theorem we can compute autocorrelation using two FFTs accordingly:

$$\boldsymbol{F_R}(\omega) = \boldsymbol{FFT}[\boldsymbol{x}(\boldsymbol{t})] \tag{28}$$

$$\boldsymbol{S}(\omega) = \boldsymbol{F_R}(\omega) \cdot \overline{\boldsymbol{F_R}(\omega)} \tag{29}$$

$$\boldsymbol{R}(\tau) = \boldsymbol{IFFT}[\boldsymbol{S}(\omega)] \tag{30}$$

As we want the spectrum of autocorrelated function, adding another FFT would be redundant, since we can omit the last step instead and attain the same effect.

$$\boldsymbol{FFT}(\boldsymbol{IFFT}[\boldsymbol{S}(\omega)]) = \boldsymbol{S}(\omega) \tag{31}$$

3. SYNTHESIZER

A sound synthesizes is an electronic instrument or a software capable of producing wide range of sound. It can both imitate other instruments as well as create new sounds. Synthesizers are built from blocks, that every one of them generates or modulates the signal.

3.1. Basic synthesis controls

3.1.1. Oscillator

The tones produced by synthesizer are generated by oscillator. The oscillator can generate three types of wave. The sine wave

$$y(n) = A \cdot sin\left(2\pi n \frac{f}{f_s} + \emptyset\right) \tag{32}$$

square wave

$$y(n) = \begin{cases} 1 & 0 \leq n < \frac{N}{2} \\ -1 & \frac{N}{2} \leq n < N \end{cases} \tag{33}$$

and saw-tooth wave

$$y(n) = 2 \cdot \left(n - \frac{N}{2}\right)\frac{f}{f_s} \tag{34}$$

3.1.2. ADSR Envelope Generator

The loudness produced by many instruments is not always the same for all the time it sounds. The *attack* and *decay* have a great effect on the instrument's sonic character. Sound synthesizer often applies *envelope generator* that controls a sound's parameters over the duration. Most often is an ADSR (attack decay sustain release) envelope. The envelope can be applied to control not only amplitude, but also filter frequency, oscillator frequency and other synthesizer parameters. The contour of an ADSR envelope is specified using four parameters:

- **Attack time** is the time taken for initial run-up of level from nil to peak, beginning when the key is first pressed.
- **Decay time** is the time taken for the subsequent run down from the attack level to the designated sustain level.
- **Sustain level** is the level during the main sequence of the sound's duration, until the key is released.

- **Release time** is the time taken for the level to decay from the sustain level to zero after the key is released.

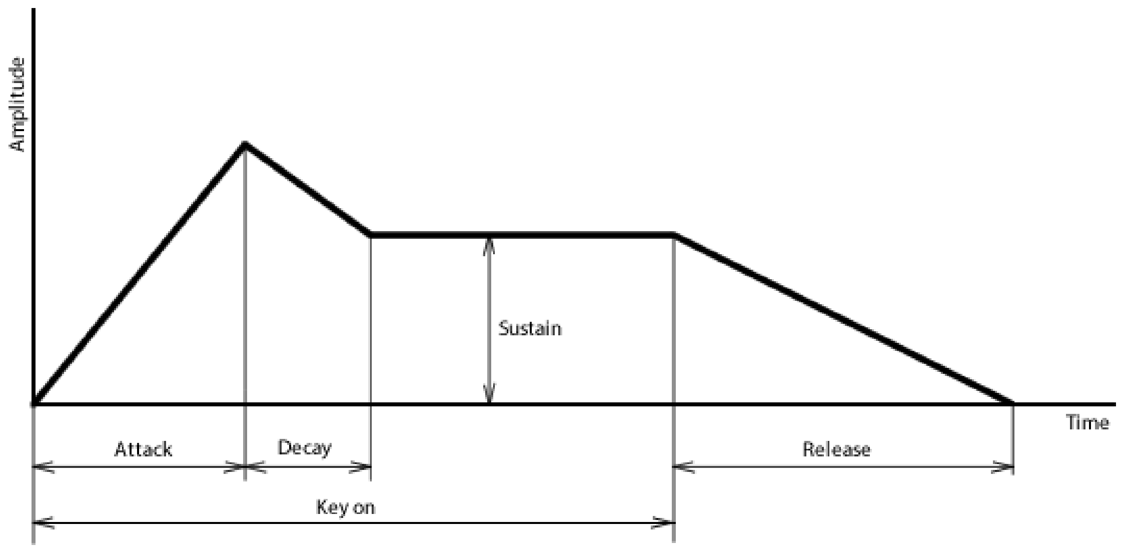


Figure 4: Schema of ADSR envelope

3.1.3. Filter

The various types of filters can be defined according to the following classification (see Figure 5):

- **Lowpass (LP)** filters select low frequencies up to the cut-off frequency f_c and attenuate frequencies higher than f_c. Additionally, a resonance may amplify frequencies around f_c.
- **Highpass (HP)** filters select frequencies higher than f_c and attenuate frequencies below f_c, possibly with a resonance around f_c.
- **Bandpass (BP)** filters select frequencies between a lower cut-off frequency f_{cl} and a higher cut-off frequency f_{ch}. Frequencies below f_{cl} and frequencies higher than f_{ch} are attenuated.
- **Bandreject (BR)** filters attenuate frequencies between a lower cut-off frequency f_{cl} and a higher cut-off frequency f_{ch}. Frequencies below f_{cl} and frequencies higher than f_{ch} are passed.
- **Allpass** filters pass all frequencies, but modify the phase of the input signal.

The lowpass with resonance is very often used in computer music to simulate an acoustical resonating structure; the highpass filter can remove undesired very low frequencies; the bandpass can produce effects such as the imitation of a telephone line or of a mute on an acoustical instrument; the bandreject can divide the audible spectrum into two bands that seem to be uncorrelated.

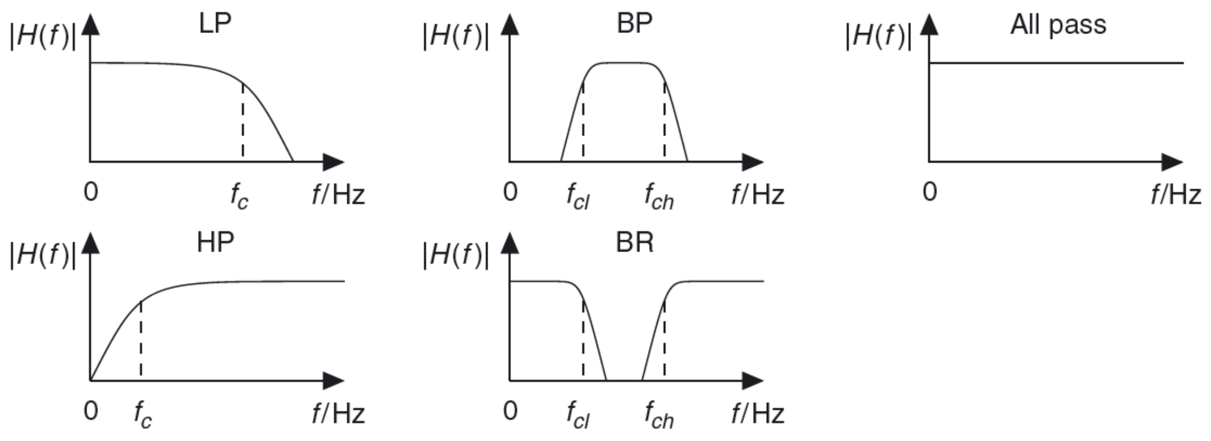


Figure 5: Filter classification [8]

3.1.3.1. Canonical filters

There are various ways to implement a digital filter. The simplest being the canonical filter, as shown in Figure 2.2 for a second-order filter, which can be implemented by the difference equations

$$x_h(n) = x(n) - a_1 x_h(n-1) - a_2 x_h(n-2) \tag{35}$$

$$y(n) = b_0 x_h(n) + b_1 x_h(n-1) + b_2 x_h(n-2) \tag{36}$$

Table 3: Coefficients for first-order filters. [8]

	b$_0$	**b_1**	**a_1**
Lowpass	$\frac{K}{(K+1)}$	$\frac{K}{(K+1)}$	$\frac{(K-1)}{(K+1)}$
Highpass	$\frac{1}{(K+1)}$	$\frac{-1}{(K+1)}$	$\frac{(K-1)}{(K+1)}$
Allpass	$\frac{(K-1)}{(K+1)}$	1	$\frac{(K-1)}{(K+1)}$

and leads to transfer function

$$H(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-1}}{1 + a_1 z^{-1} + a_2 z^{-1}}. \tag{37}$$

By setting $a_2 = b_2 = 0$, this reduces to a first-order filter which, can be used to implement an all-pass, lowpass or highpass with the coefficients of Table 3 where *K* depends on the cut-off frequency f_c by

$$K = tan\left(\pi \frac{fc}{fs}\right) \tag{38}$$

For the allpass filter, the coefficient K likewise controls the frequency fc when $-90°$ phase shift is reached.

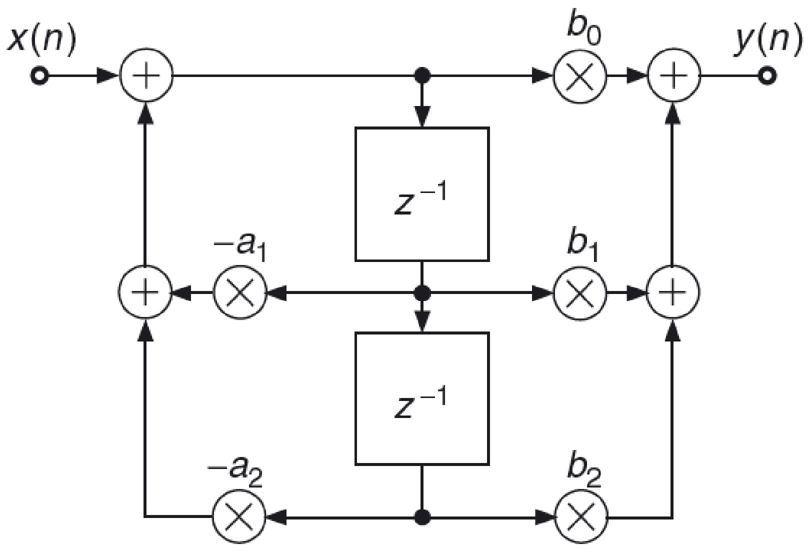


Figure 6: Canonical second-order digital filter [8]

For the second-order filters with coefficients shown in Table 4, in addition to the cut-off frequency (for lowpass and highpass) or the centre frequency (for bandpass, bandreject and allpass) we additionally need the Q factor with slightly different meanings for the different filter types:

- For the lowpass and highpass filters, it controls the height of the resonance. For $Q = \frac{1}{\sqrt{2}}$, the filter is maximally flat up to the cut-off frequency; for lower Q, it has higher pass-band attenuation, while for higher Q, amplification around fc occurs.
- For the bandpass and bandreject filters, it is related to the bandwidth f_b by $Q = \frac{f_b}{f_c} fb$, i.e., it is the inverse of the relative bandwidth $\frac{f_b}{f_c}$.
- For the allpass filter, it likewise controls the bandwidth, which here depends on the points where ±90° phase shift relative to the −180° phase shift at fc are reached.

While the canonical filters are relatively simple, the calculation of their coefficients from parameters like cut-off frequency and bandwidth is not. In the following, we will therefore study filter structures that are slightly more complicated, but allow for easier parameterization.

	b₀	**b₁**	**b₂**	**a₁**	**a₂**
Lowpass	$\frac{K^2Q}{K^2Q+K+Q}$	$\frac{2K^2Q}{K^2Q+K+Q}$	$\frac{K^2Q}{K^2Q+K+Q}$	$\frac{2Q(K^2-1)}{K^2Q+K+Q}$	$\frac{K^2Q-K+Q}{K^2Q+K+Q}$

Highpass	$\frac{Q}{K^2Q+K+Q}$	$-\frac{2Q}{K^2Q+K+Q}$	$\frac{Q}{K^2Q+K+Q}$	$\frac{2Q(K^2-1)}{K^2Q+K+Q}$	$\frac{K^2Q-K+Q}{K^2Q+K+Q}$
Bandpass	$\frac{K}{K^2Q+K+Q}$	0	$-\frac{K}{K^2Q+K+Q}$	$\frac{2Q(K^2-1)}{K^2Q+K+Q}$	$\frac{K^2Q-K+Q}{K^2Q+K+Q}$
Bandreject	$\frac{Q(1+K^2)}{K^2Q+K+Q}$	$\frac{2Q(K^2-1)}{K^2Q+K+Q}$	$\frac{Q(1+K^2)}{K^2Q+K+Q}$	$\frac{2Q(K^2-1)}{K^2Q+K+Q}$	$\frac{K^2Q-K+Q}{K^2Q+K+Q}$
Allpass	$\frac{K^2Q-K+Q}{K^2Q+K+Q}$	$\frac{2Q(K^2-1)}{K^2Q+K+Q}$	1	$\frac{2Q(K^2-1)}{K^2Q+K+Q}$	$\frac{K^2Q-K+Q}{K^2Q+K+Q}$

Table 4: Coefficients for second-order filters. [8]

3.1.4. Amplifier

In practical implementation, the signal coming out of the filter circuitry must be brought up in level so that the output signal shall be strong enough to drive the line level inputs. This is done by an amplifier. The function of amplifier is

$$y(n) = k \cdot x(n) \tag{39}$$

where *k* is the gain level. If *k<1* the signal is reduced and for *k>1* is strengthened. It usually consists of a knob for setting the overall gain level, and one input and one output port. Most importantly, it has also a control input for temporal amplitude level change for connecting signal from ADSR envelope generator.

3.2. On-Set filter

To activate ADSR envelope attack and release, we need detector, which indicates the beginning of the note and the end, respectively. For this purpose, is used On-Set filter (OS). OS use *envelope detector*, sometimes also called *envelope follower*, and one or two thresholds to detect note start and end.

3.2.1. Envelope detector

Signal's envelope is equivalent to its outline and an envelope detector connects all the peaks in this signal. Envelope detection has numerous applications in the fields of signal processing and communication, one of which is amplitude modulation (AM) or On-Set filter. There are multiple methods of finding the envelope.

3.2.1.1. Squaring and Lowpass Filtering

This envelope detection method involves squaring the input signal and sending this signal through a lowpass filter. Squaring the signal demodulates the input by using the input as its own carrier wave. This means that half the energy of the signal is pushed up to higher frequencies and half is shifted down toward DC. You then downsample this signal to reduce the sampling frequency. You can do downsampling if the signal does not have any high frequencies which could cause aliasing. Otherwise an FIR decimation should be used which applies a low pass filter before downsampling the signal. After this, pass the signal through a minimum-phase, lowpass filter to eliminate the high frequency energy. Finally, you are left with only the envelope of the signal.

To maintain the correct scale, you must perform two additional operations. First, you must amplify the signal by a factor of two. Since you are keeping only the lower half of the signal energy, this gain matches the final energy to its original energy. Second, you must take the square root of the signal to reverse the scaling distortion that resulted from squaring the signal.

This envelope detection method is easy to implement and can be done with a low-order filter, which minimizes the lag of the output.

3.2.1.2. Envelope follower with different attack and release

This envelope detection method involves LPF to smoothen the signal and then transposition of negative values to the positive ones, which can be done by simply by taking the absolute value of the sample. If the value of current sample is higher than the previous one, set the envelope current sample proportionally by the attack coefficient, otherwise decrease the amplitude using release coefficient.

$$e(t) = \begin{cases} c_A(e(t-1) - x(t)), & x(t) > e(t) \\ c_R(e(t-1) - x(t)), & x(t) \leq e(t) \end{cases} \tag{40}$$

3.2.1.3. The Hilbert Transformation

This envelope detection method involves creating the analytic signal of the input using the Hilbert transform. An analytic signal is a complex signal, where the real part is the original signal and the imaginary part is the Hilbert transform of the original signal.

Mathematically the envelope *e(t)* of a signal *x(t)* is defined as the magnitude of the analytic signal as shown by the following equation.

$$e(t) = \sqrt{x(t)^2 + \hat{x}(t)^2} \tag{41}$$

where $\hat{x}(t)$ is the Hilbert transform of *x(t)*.

You can find the Hilbert transform of the signal using a 32-point Parks-McClellan FIR filter. To form the analytic signal, you then multiply the Hilbert transform of the signal by sqrt(-1) (the imaginary unit) and add it to the time-delayed original signal. It is necessary to delay the input signal because the Hilbert transform, which is implemented by an FIR filter, will introduce a delay of half the filter length.

You find the envelope of the signal by taking the absolute value of the analytic signal. The envelope is a low frequency signal compared to the original signal. To reduce its sampling frequency, to eliminate ringing and to smooth the envelope, you downsample this signal and pass the result through a lowpass filter.

3.2.2. Thresholds

We need the thresholds to indicate, where the signal envelope level is so high or low, the note starts or ends, respectively. On-Set Filter can implement either one threshold for both, start and end of the note, or hysteresis[10], for start and end independently. One to set the output and another, set a few dB below, to reset the output. This means that once a signal has dropped below the close threshold, it has to rise to the open threshold, so that a signal that crosses over the close threshold regularly does not set the filter and cause chattering.

[10] Hysteresis is the dependence of a system not only no its current state or environment but also on its past one. The dependency arises because the system can be in more than one internal state. To predict its future development, either its internal state or its history must be known.

II. PRACTICAL PART

4. VST PLUG-IN

As a practical part of this thesis is to create plug-in in VST technology implementing suitable algorithm to analyse string instrument sound, extract the fundamental frequency and further process it with voltage-controlled amplifier (VCA) and voltage-controlled filter (VCF).

4.1. Virtual studio technology

Virtual studio technology is software interface developed by Steinberg Media Technologies for audio synthesizers and effects to simulate traditional recording studio techniques. VST plug-in cannot run by itself and relies on other software using the VST interface. Such software is mostly the digital audio workstation.

"From the host application's point of view, a VST Plug-In is a black box with an arbitrary number of inputs, outputs (Midi or Audio), and associated parameters. The Host needs no knowledge of the Plug-In process to be able to use it." [9]

The VST provides interface for inputs and outputs as well as for the parameters, that can be automatized using the host application. The input and output values can move within the range from -1.0 to 1.0 inclusive and the parameters can reach values from 0.0 to 1.0 inclusive.

4.2. Digital audio workstation

Digital audio workstation (DAW) is a computer software or electronic device providing functionality for creating, recording and edition sound. It can use multiple software plug-ins as well as the hardware components, of which both of them can also be partly or fully controlled from the inside of the DAW.

4.2.1. Interface

DAW can have several user interfaces of which the most common nowadays applies the multitrack tape recorder metaphor. This is based on the technique when the sounds are recorded separately and then played and recorded together allowing adding delay, changing the level and tone or applying effect to each sound without affecting the others. All tracks are then mixed together.

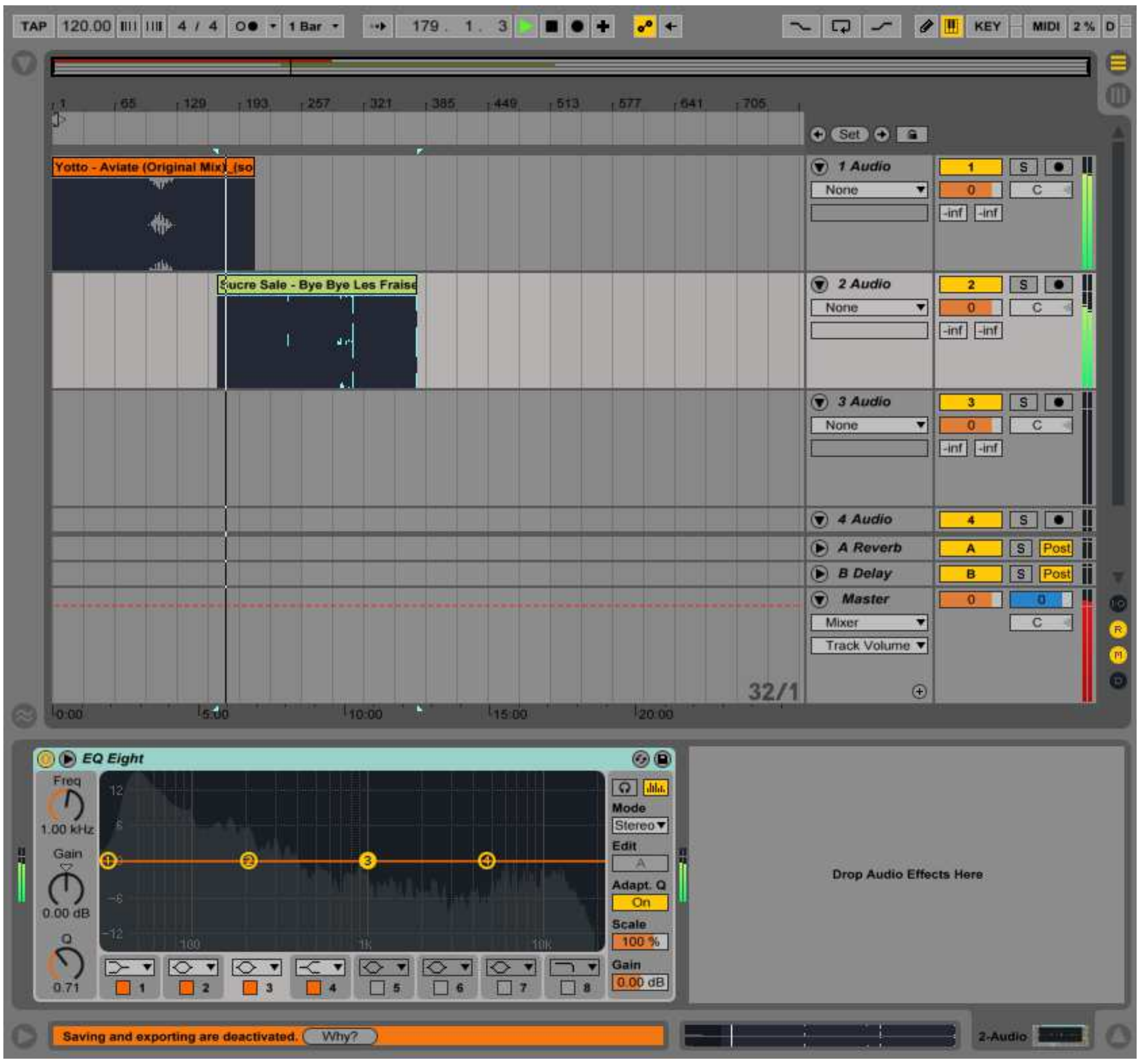


Figure 7: Ableton Live 9

4.2.2. Functionality

Each track[11] has its own parameters, such as the sound level, pan (for stereo sounds) and slots for additional effects. In case of midi also the sound-generating device has to be added.

The multitrack metaphor requires a mixer to put the tracks together. Mixer is the fundamental part of DAW since it is the place, where the tracks are compiled together. Each track can have set the amplitude, pan and sometimes other parameters, such as solo parameter.

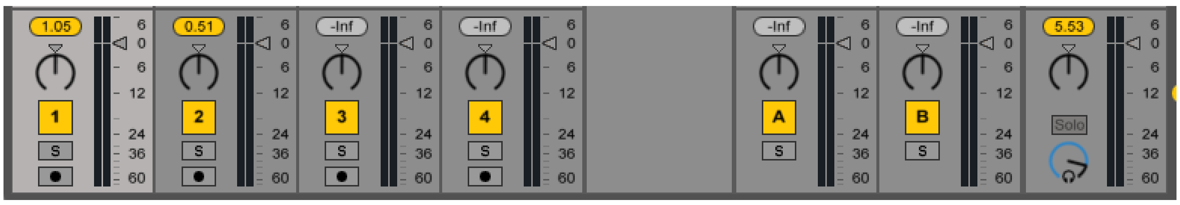


Figure 8: Ableton Live mixer

Automation is user defined function of the parameter of effect. The process can be set using the breakpoints. Between each breakpoint there is defined function stating the value for the parameter in each moment. The function can be linear or exponential or logarithmic depending on the DAW. For each parameter can be set only one automation.

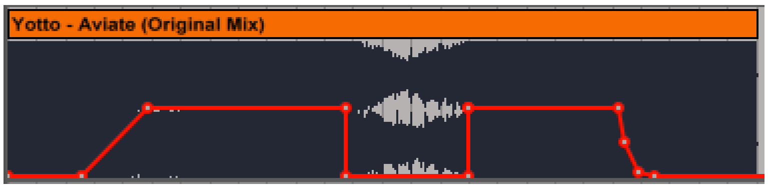


Figure 9: Ableton Live automation

4.3. Prerequisites

The plug-in is written in C# language using .NET Framework 2.0. To open the solution in Visual Studio it is required to have installed .NET minimum version 2.0 and VST host program. For editing and compilation, the source is required Visual Studio 2012 and ILMerge. VST.NET v1.0 and Exocortex.DSP libraries are included in the solution.

4.4. Implementation

The goal of the practical part is to implement VST plug-in for guitar or bass signal consisting of two switchable signal paths as shown in Figure 10. While first path leaves the input signal for further modulation, second way consist of pitch detector, which analyse the input signal and sets the fundamental frequency.

[11] Track is referred to as a single sound in the multitrack tape recorder metaphor technique

The signal from either of these ways is then modulated by both VCF and VCA. The intensity is controlled by separate ADSR envelope. On-Set filter (On-Set) indicates the beginning and end of the tone from input signal. This signal is used to control bypass of VCO and setting ADSR envelopes.

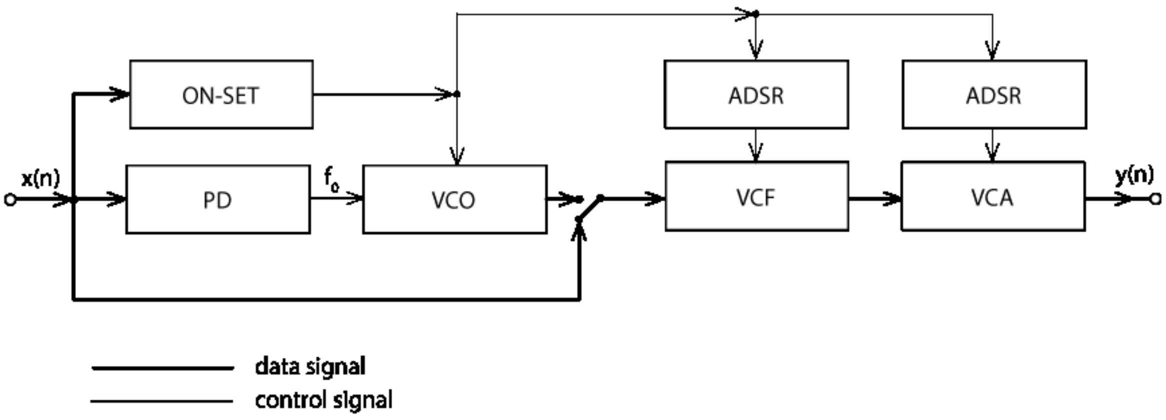


Figure 10: VST plug-in schema

This schema can be described with following equations

$$y(n) = k\left(b_0 x_h(n) + b_1 x_h(n-1) + b_2 x_h(n-2)\right) \tag{42}$$

where $x_h(n)$ is canonical filter described in equation

$$x_h(n) = x(n) - a_1 x_h(n-1) - a_2 x_h(n-2)$$ (35).

All modules are stand-alone modules and can separately. In case of need they can be reorganized or completely switched off, so the order and connection and dependency of the modules can be changed by the developer inside the source code.

4.4.1. Pitch detector

Pitch detector is implemented as algorithm taking strongest frequency of autocorrelated input signal described in 2.4.2.2. Implementation of this algorithm is done by autocorrelation of the buffered samples and getting the power spectral density of the autocorrelation result. Here the strongest frequency is considered to be the fundamental frequency.

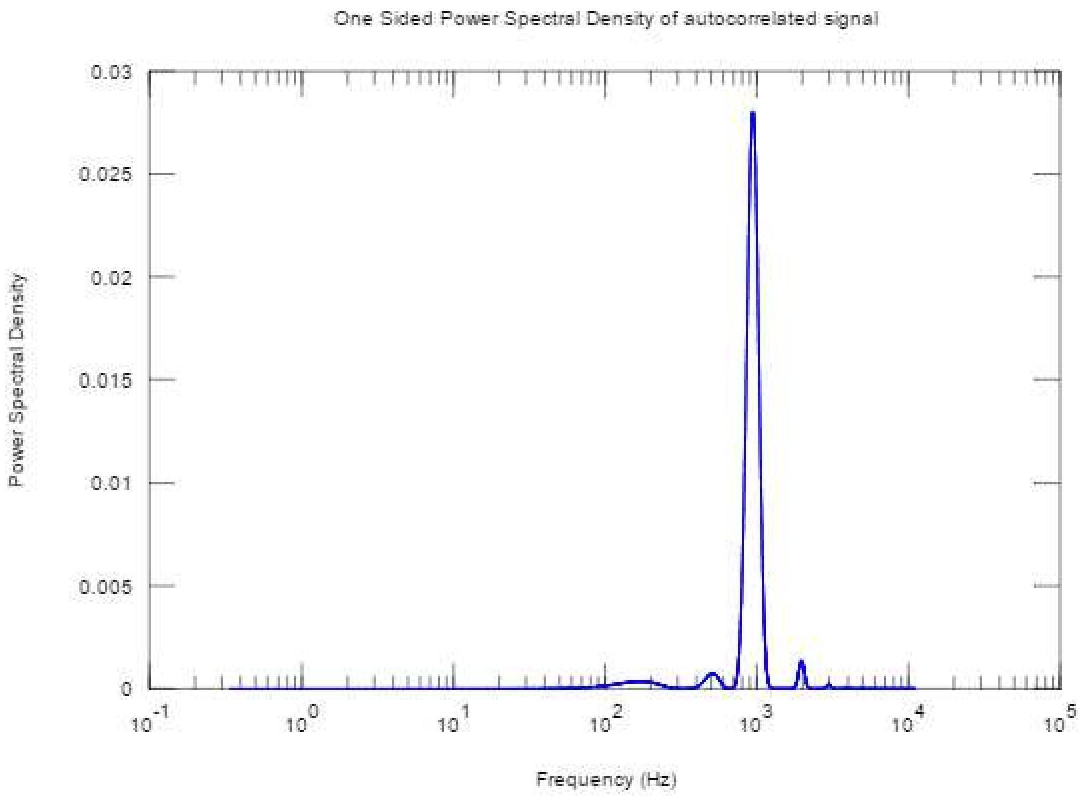


Figure 11: Power spectral density of autocorrelated signal with the peek in fundamental frequency

For increased performance, the enhancements are implemented, such as frequency range to have better control over the process and processing frequency that sets number of detection cycles within second. The own pitch detection uses buffer, over which the FFT is computed with use of Hann window.

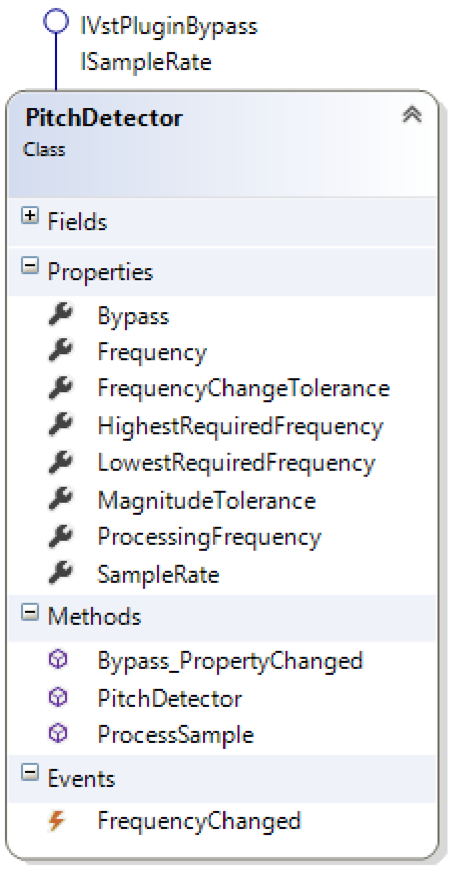


Figure 12: PitchDetector class

For still sound there can be set the frequency change tolerance, so if the oscillates within this toleration, the event *FrequencyChanged* is not triggered.

Another optimization improves performance by setting the processing gap every nth sample. Since the FFT is computationally expensive operation it is better not to process whole buffer each step.

4.4.1.1. Window

Windows for FFT are defined in its own separate class, where are defined various types of windows. The implemented windows are as follows:

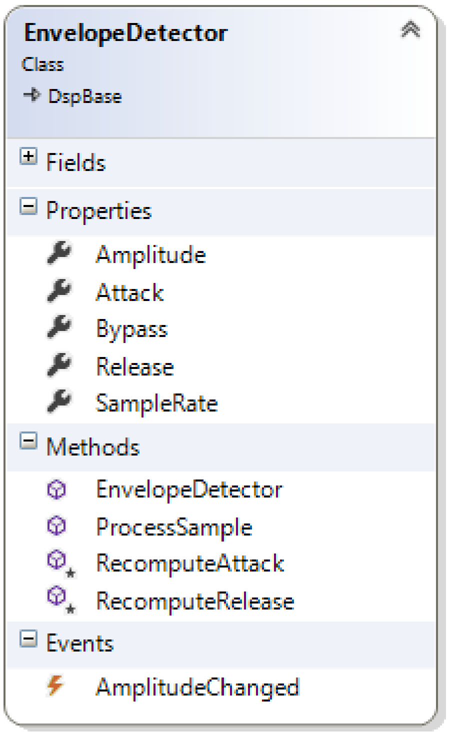
- Bartlett
- Bartlett-Hann
- Blackman
- Cosine
- Gaussian
- Hamming
- Hann
- Lanczos
- Rectangular

- Triangular

  The definition of the windows can be found in 2.2.1.

4.4.2. Envelope follower

Envelope follower creates envelope from the signal. The basic idea is to create the positive envelope following the course of the wave. The basic is to get the absolute values of the wave. To finish the envelope, we need to set the parameters of attack and release. Attack and release influences the envelopes ascension or descent respectively.



EnvelopeDetector
Class
DspBase

Fields

Properties
- Amplitude
- Attack
- Bypass
- Release
- SampleRate

Methods
- EnvelopeDetector
- ProcessSample
- RecomputeAttack
- RecomputeRelease

Events
- AmplitudeChanged

Figure 13: EnvelopeDetector class

The attack and release are calculated using attack and release coefficient c_{att} or c_{rls}, where the value of the attack or release is given in milliseconds.

$$c_{att} = e^{\frac{\log(0.01)}{att f_s \cdot 0.001}} \tag{43}$$

The final value of the amplitude is then calculated using following equation

$$y(n) = c_{att} \cdot (y(n-1) - |x(n)|) + |x(n)| \tag{44}$$

The decision if is used attack or release coefficient is given by condition $|x(n)|>y(n-1)$. If absolute value of current input sample is higher than the previous output sample, the attack is used, otherwise is used release.

For demonstration is example of the envelope for sine wave with 1000Hz, attack 1ms and release 3ms.

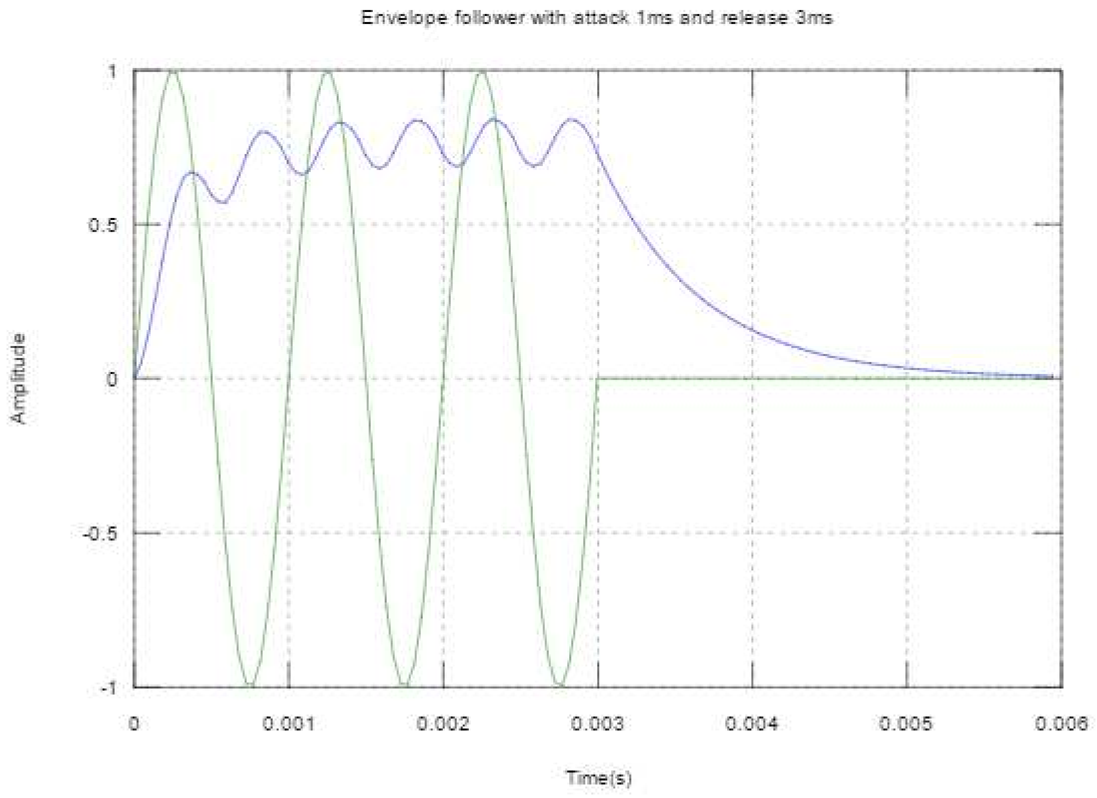


Figure 14: Envelope follower with attack 1ms and release 3ms

In current implementation we want to notify the rising edge immediately and also eliminate the oscillation of output as much as possible but with acceptable delay at the tone's end, so the attack is set to 0ms and release to 10ms. With this setting the envelope looks as follows.

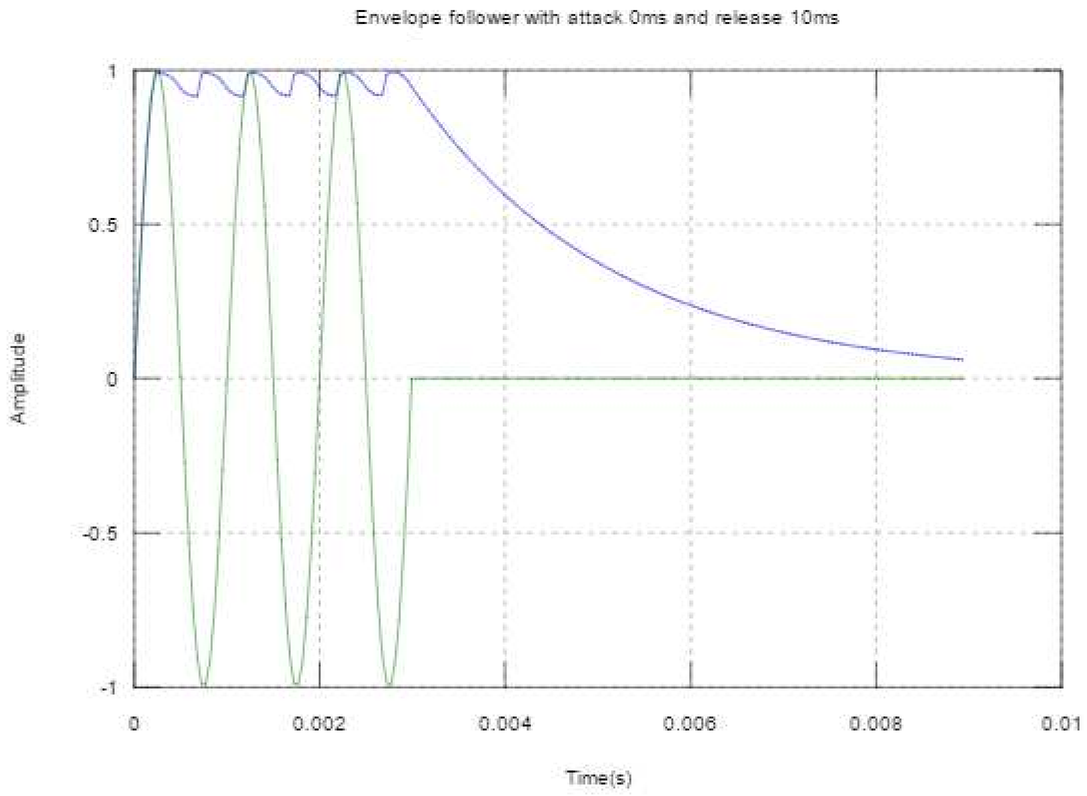
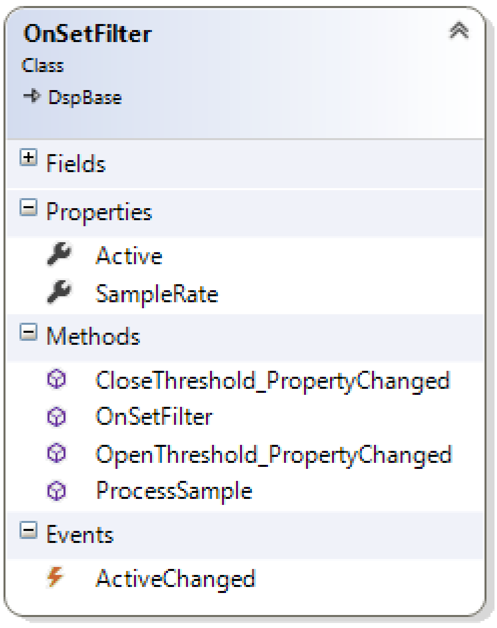


Figure 15: Envelope follower with attack 0ms and release 10ms

4.4.3. On-Set filter

On-Set filter is implemented as envelope follower to detect when the tone is on.


OnSetFilter
Class
DspBase

Fields

Properties
- Active
- SampleRate

Methods
- CloseThreshold_PropertyChanged
- OnSetFilter
- OpenThreshold_PropertyChanged
- ProcessSample

Events
- ActiveChanged

Figure 16: OnSetFilter class

The filter uses the envelope follower with attack 0ms and release 10ms to get the current amplitude. The amplitude is compared to the open and close threshold stating the moment the on-set filter activity is triggered on and off.

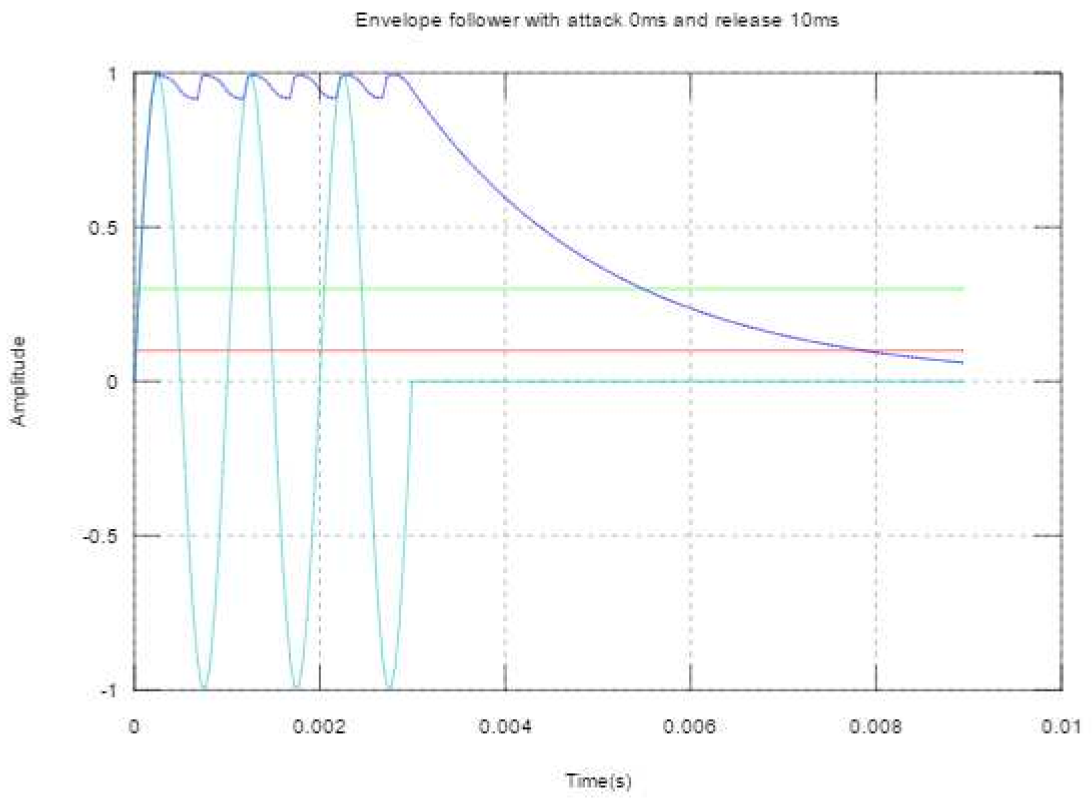


Figure 17: On-set filter with open and close threshold

On the graph can be seen the envelope of sine wave with open threshold marked with green line and close threshold marked with red line. So in this case the filter will be to active state since approximately the beginning at time 0 and deactivated in time 0.008. The delay depends on the setting of the thresholds, which can be set by user.

4.4.4. Oscillator

Oscillator produces the sound at given frequency, which is in this program set to fundamental frequency gained from PD.

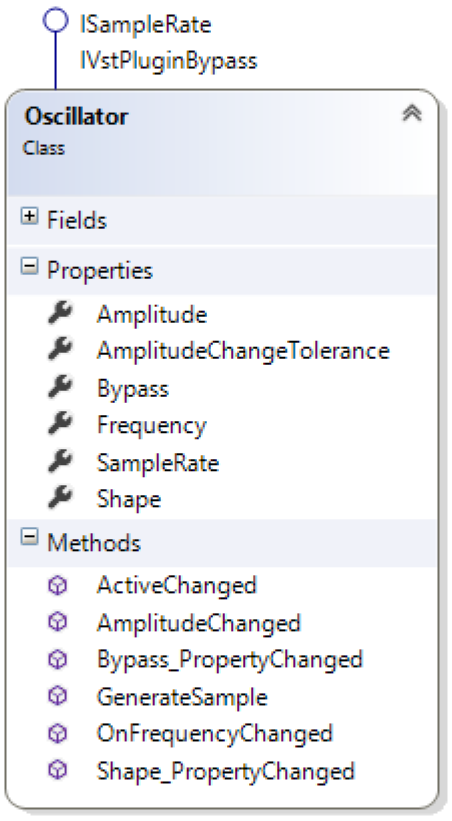


Figure 18: Oscillator class

The Oscillator class consist of multiple properties influencing the generated wave.

4.4.4.1.Sine wave

Sine wave generation is implemented using sine function described in equation $y(n) = A \cdot sin\left(2\pi n \frac{f}{f_s} + \emptyset\right)$ (32) and produces only one frequency without any harmonic nor inharmonic frequencies. The wave looks as follows.

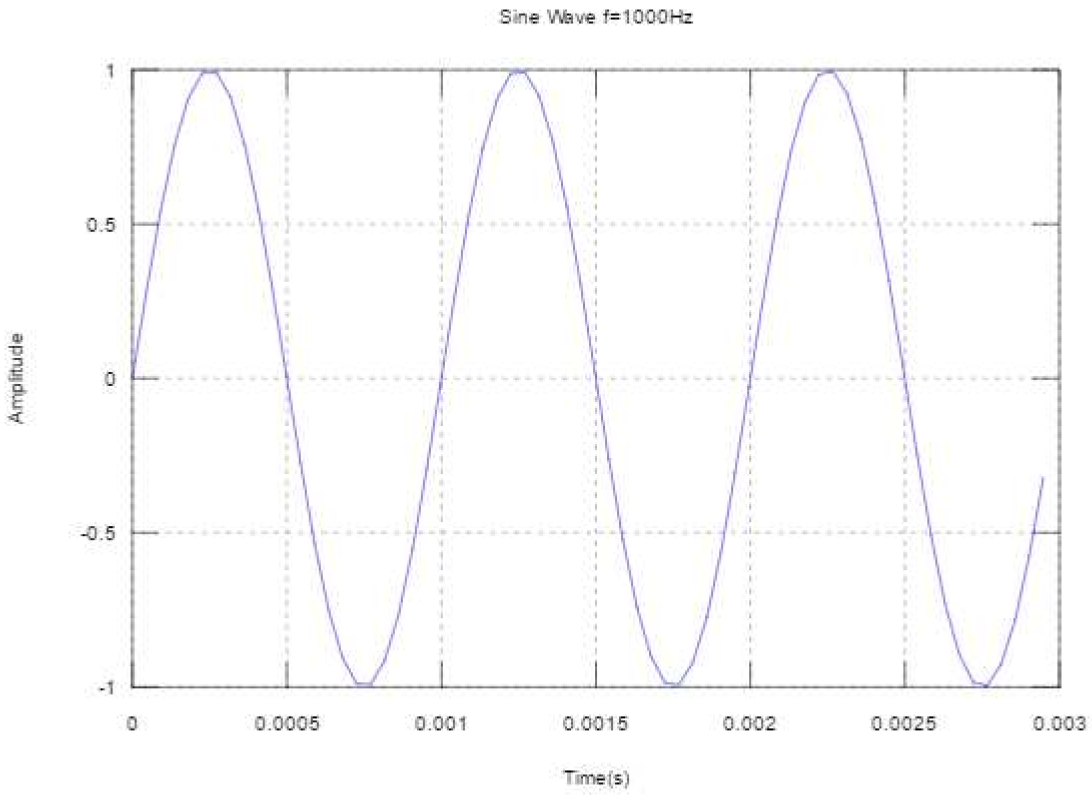


Figure 19: Sine wave

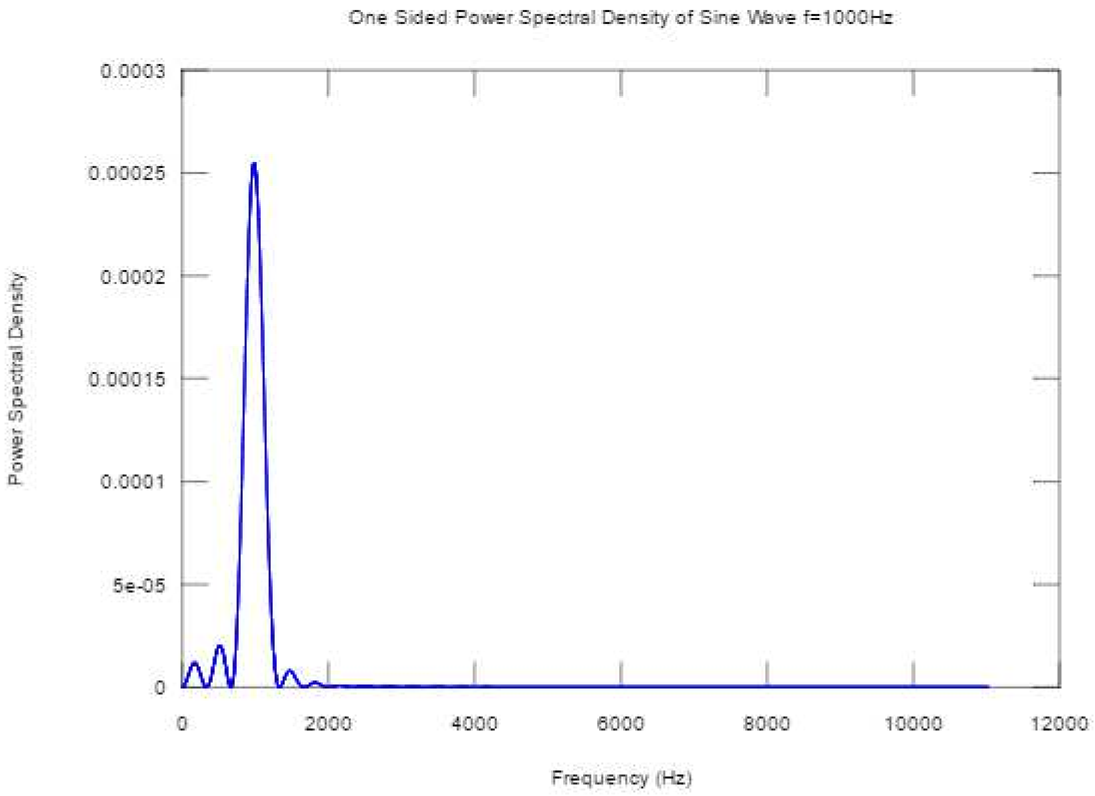


Figure 20: Power spectral density of sine wave

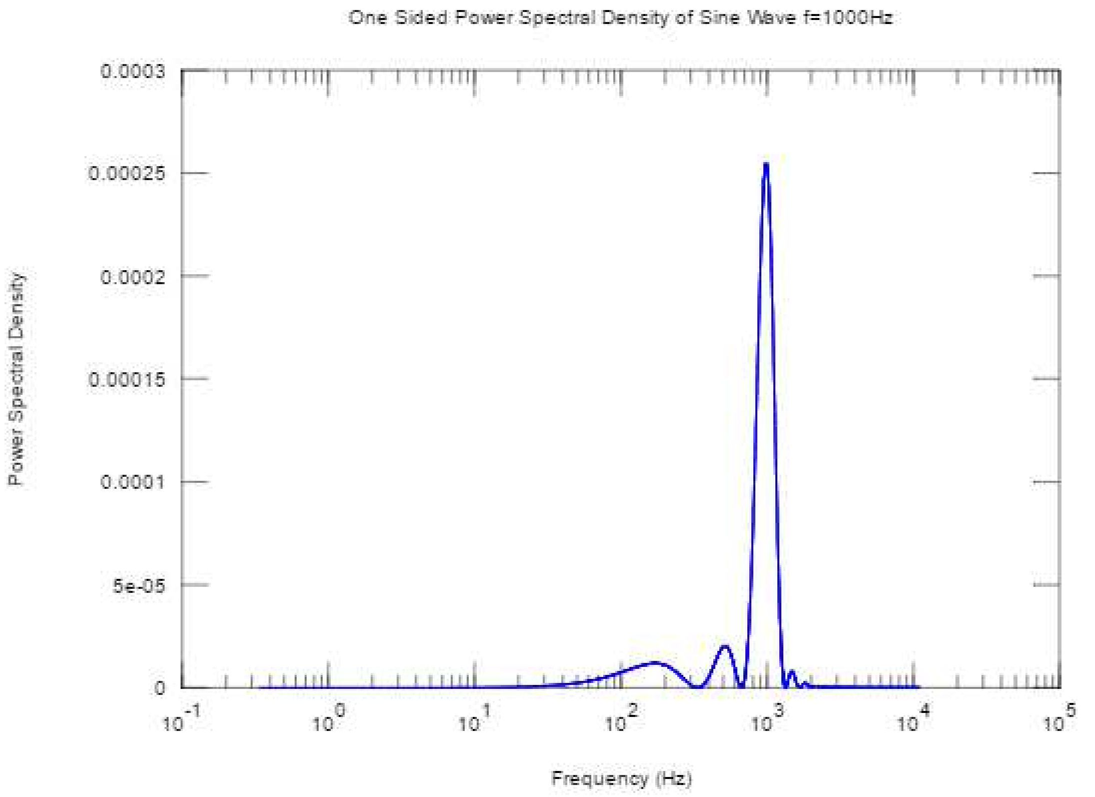


Figure 21: Power spectral density of sine wave in logarithmic scale

4.4.4.2. Square wave

Square wave generation is implemented using square function described in equation $y(n) = \begin{cases} 1 & 0 \le n < \frac{N}{2} \\ -1 & \frac{N}{2} \le n < N \end{cases}$ (33) and produces only one frequency without any harmonic nor inharmonic frequencies. The wave looks as follows.

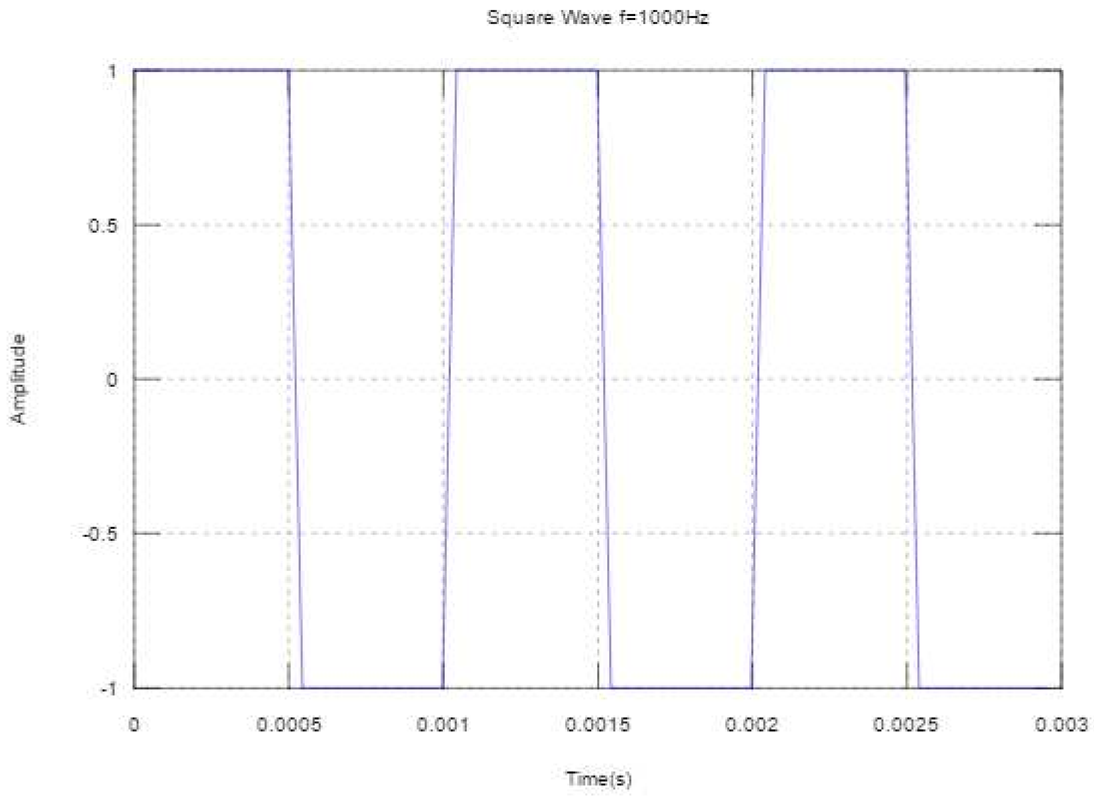


Figure 22: Square wave

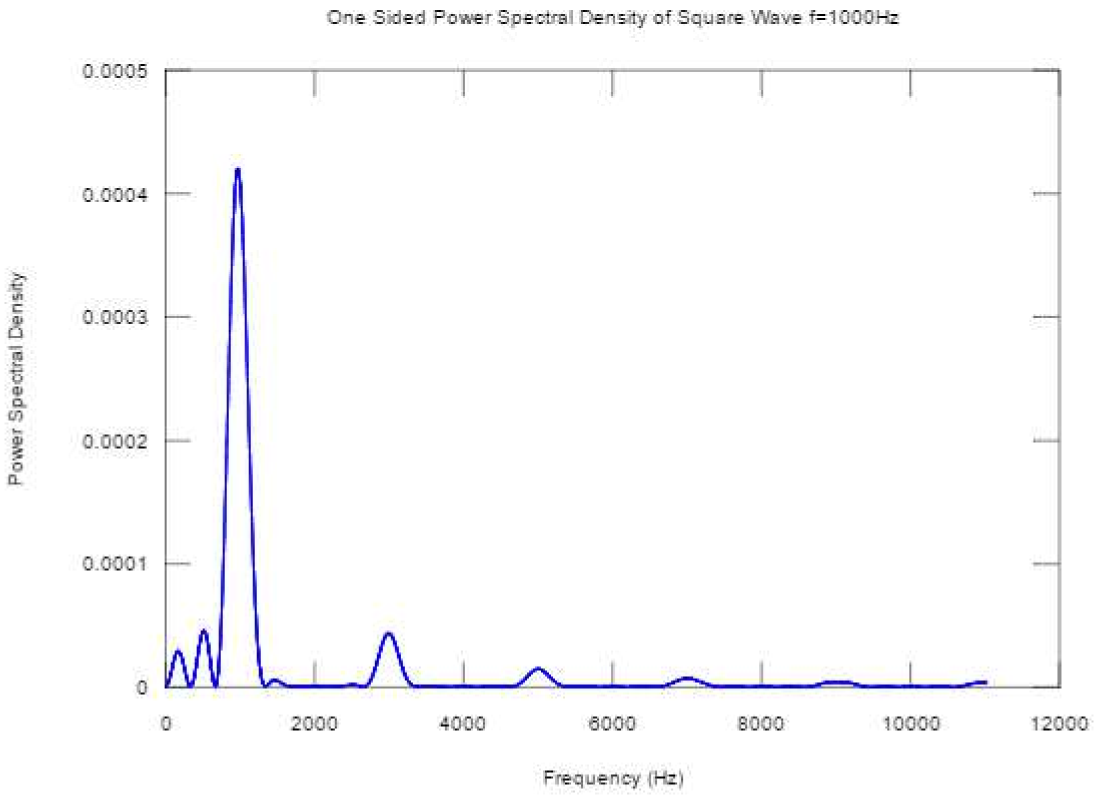


Figure 23: Power spectral density of square wave

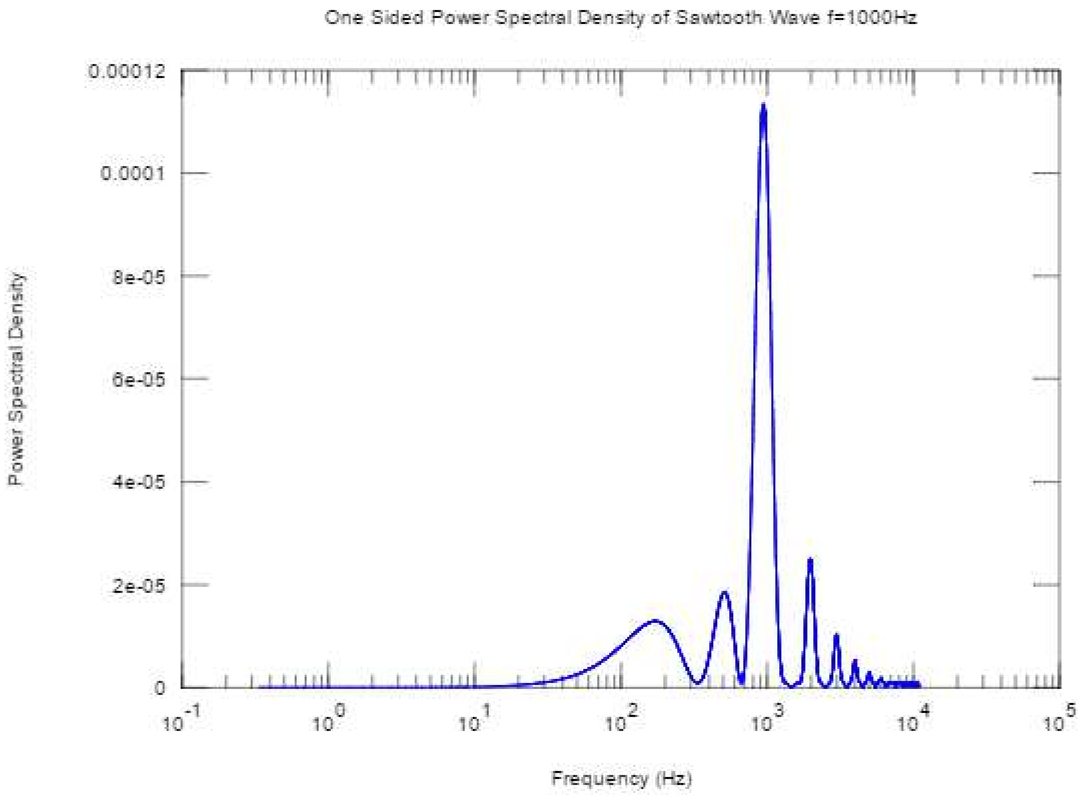


Figure 24: Power spectral density of square wave in logarithmic scale

4.4.4.3. Saw tooth

Saw tooth wave generation is implemented using saw tooth function described in equation $y(n) = 2 \cdot \left(n - \frac{N}{2}\right)\frac{f}{f_s}$ (34) and like a square wave produces multiple harmonic frequencies, just in different power. The wave looks as follows.

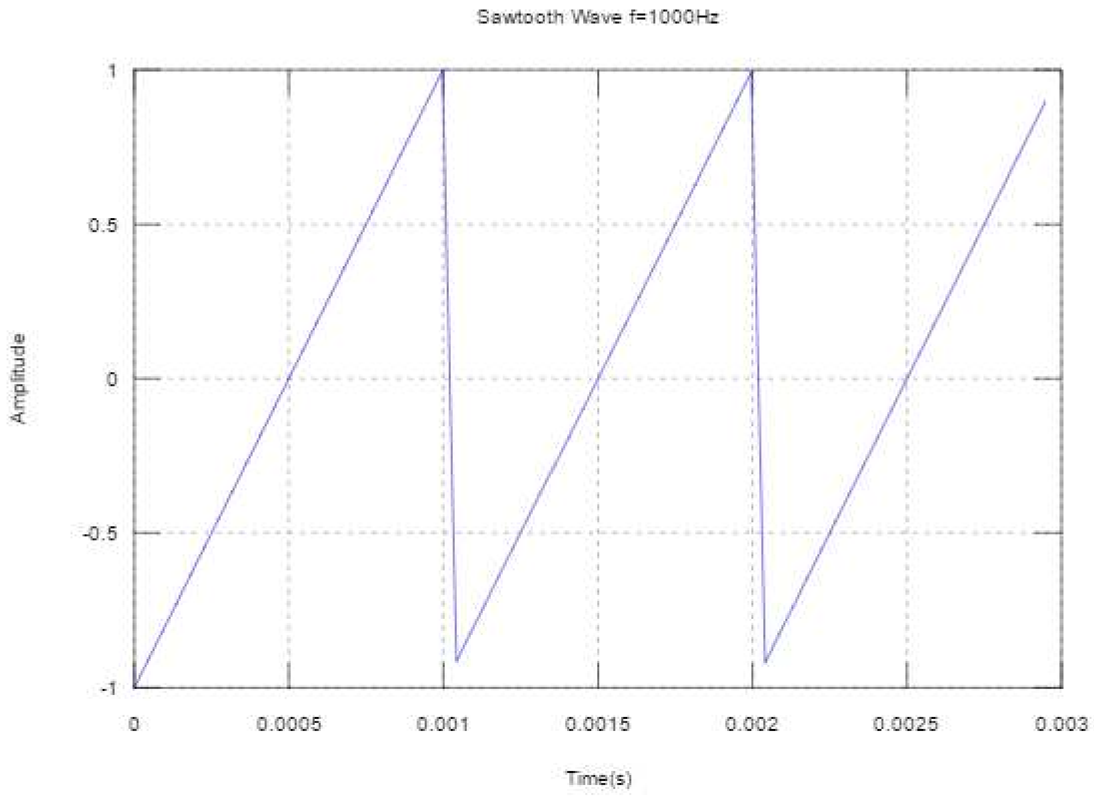


Figure 25: Saw tooth wave

On the power spectral density graph can be seen the harmonic frequencies in the multiplications of the fundamental frequency as described in equation (5)

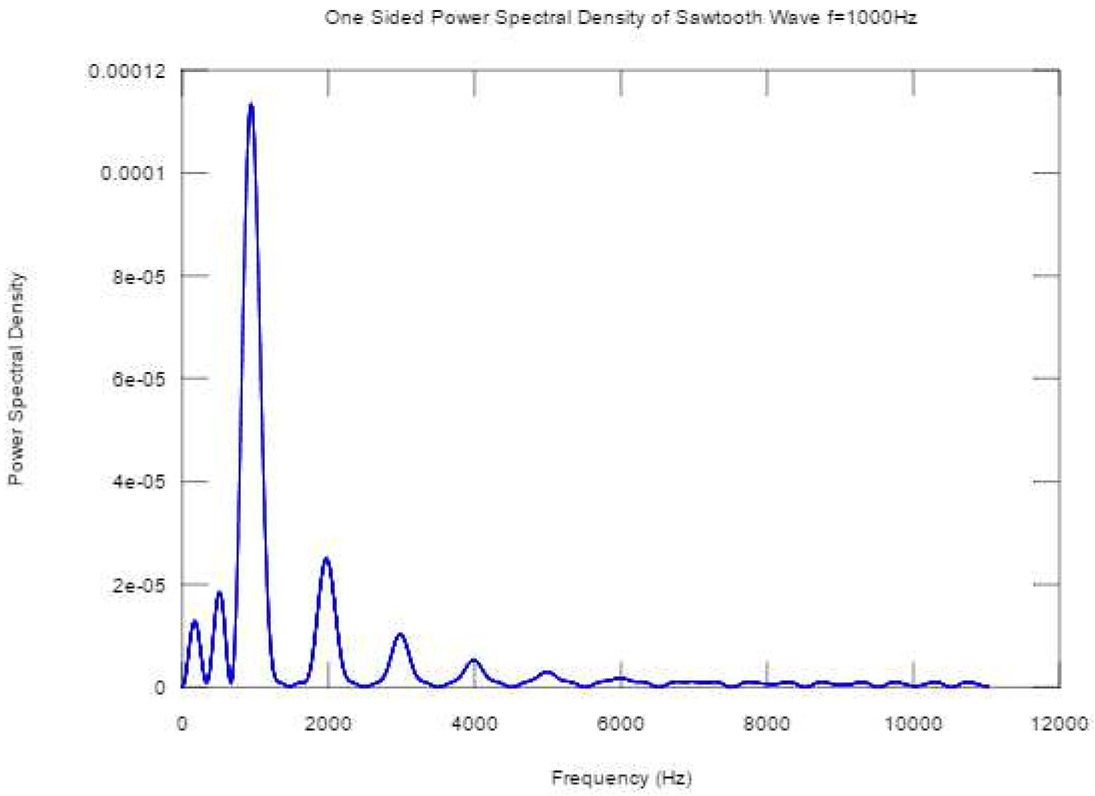


Figure 26: Power spectral density of saw tooth wave

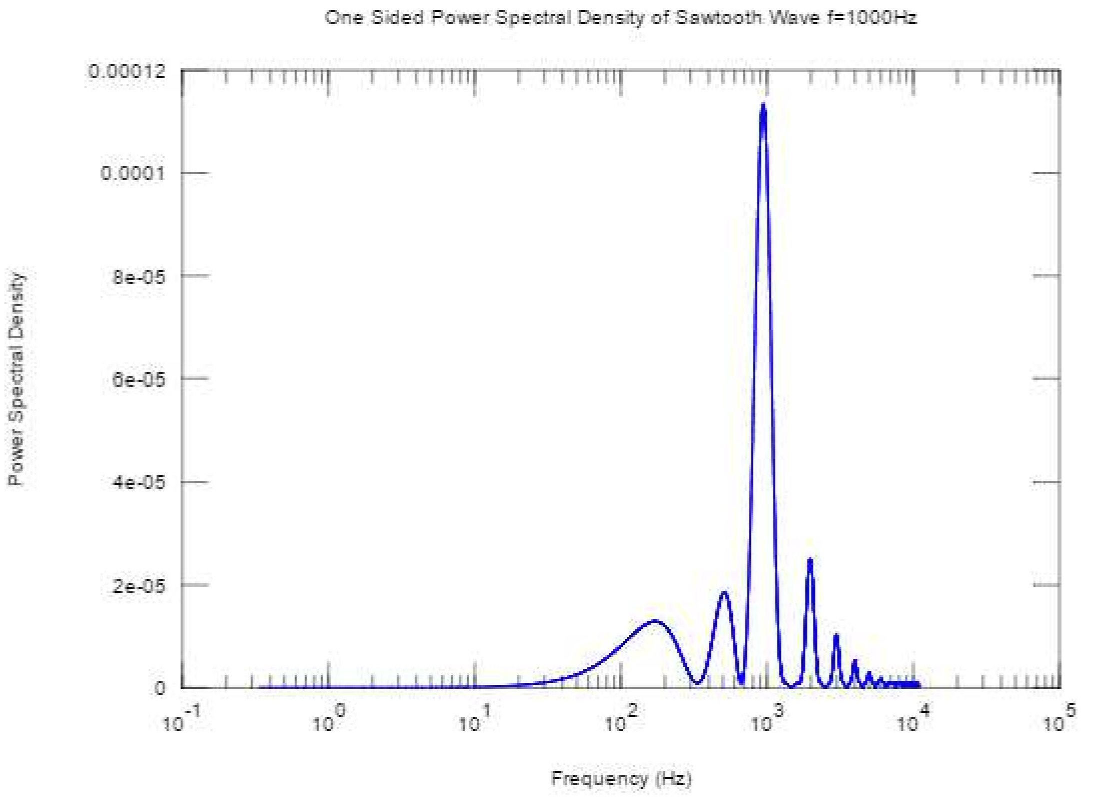


Figure 27: Power spectral density of saw tooth wave in logarithmic scale

4.4.5. ADSR Envelope

The ADSR generates envelope that is used by other DSP effect. The generated ratio is the value of ADSR envelope function in certain time. The ratio is then applied with the effect on the input value.

$$y(n) = r_e \cdot x(n) + (1 - r_e) \cdot x_f(n) \tag{45}$$

The r_e stands for the ratio generated by the envelope in method *GetRatio()* and $x_f(n)$ is filtered sample. This equation is applied in method *Apply()*.

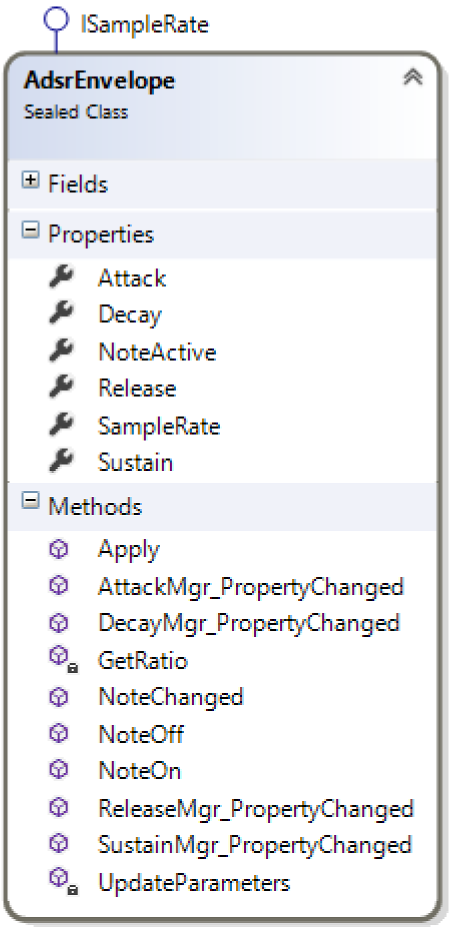
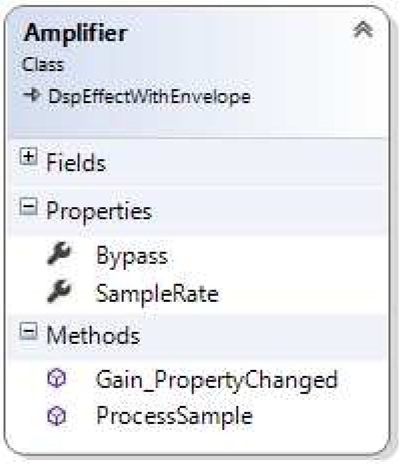


Figure 28: AdsrEnvelope class

4.4.6. Amplifier

The amplifier multiplies the input signal with gain value. The ADSR envelope sets the ratio in which the amplification is applied.



Amplifier
Class
→ DspEffectWithEnvelope

⊞ Fields
⊟ Properties
Bypass
SampleRate
⊟ Methods
Gain_PropertyChanged
ProcessSample

Figure 29: Amplifier class

4.4.7. Basic filter

Basic filter is the simplest implementation of filter removing high frequency noise by using moving average algorithm, which basically takes average of two following samples.

$$y(n) = \frac{x(n)+x(n-1)}{2} \tag{46}$$

4.4.8. Filter

Filter is implemented as canonical second-order filter described in equation $H(z) = \frac{b_0+b_1z^{-1}+b_2z^{-1}}{1+a_1z^{-1}+a_2z^{-1}}$. (37). Can be set to four types of filtering – highpass, lowpass, bandpass and bandreject. Depending on the frequency and Q parameter the input is filtered. Filtered output is also driven through the ADSR envelope. Internally it contains small buffer to keep the last three input samples needed for processing the filter.

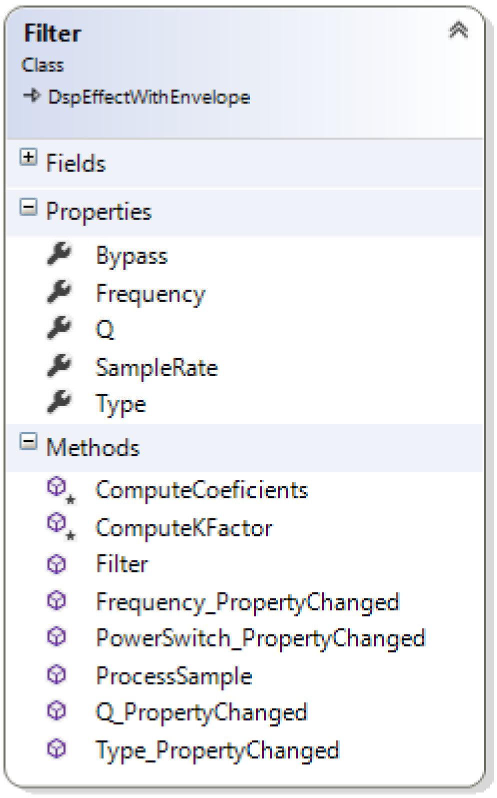

Filter
Class
DspEffectWithEnvelope
Fields
Properties
Bypass
Frequency
Q
SampleRate
Type
Methods
ComputeCoeficients
ComputeKFactor
Filter
Frequency_PropertyChanged
PowerSwitch_PropertyChanged
ProcessSample
Q_PropertyChanged
Type_PropertyChanged

Figure 30: Filter class

The filter was tested on white Gaussian noise:

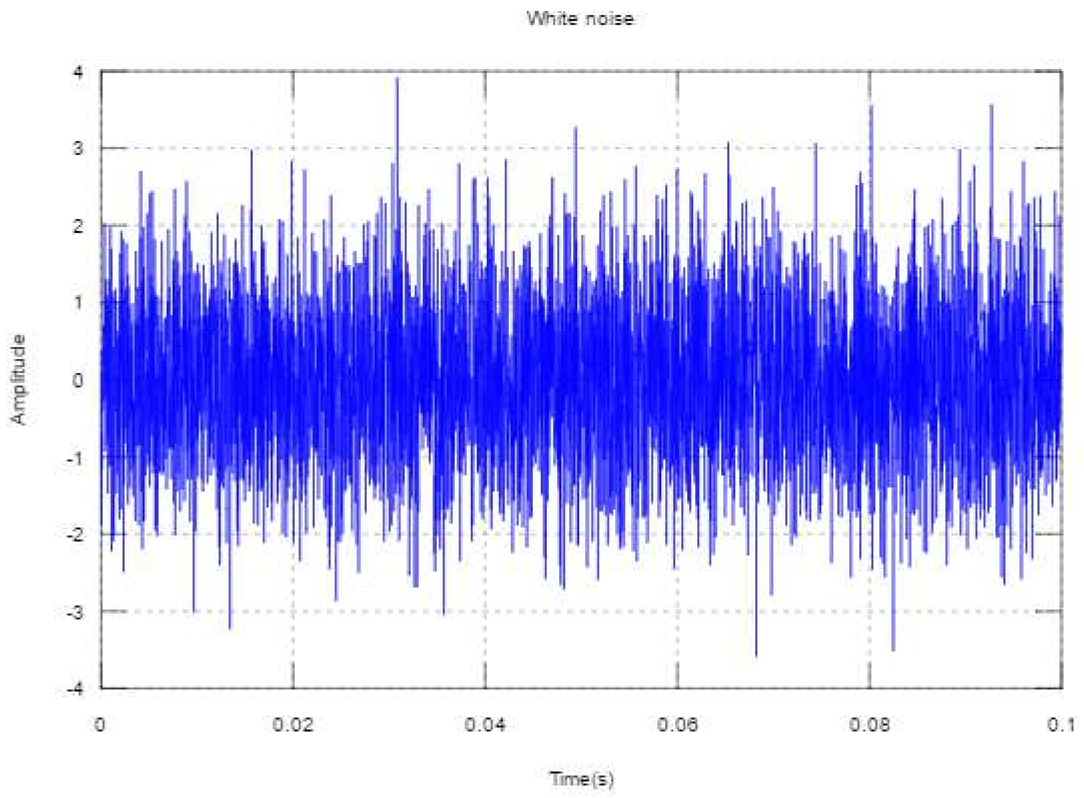


Figure 31: White Gaussian noise

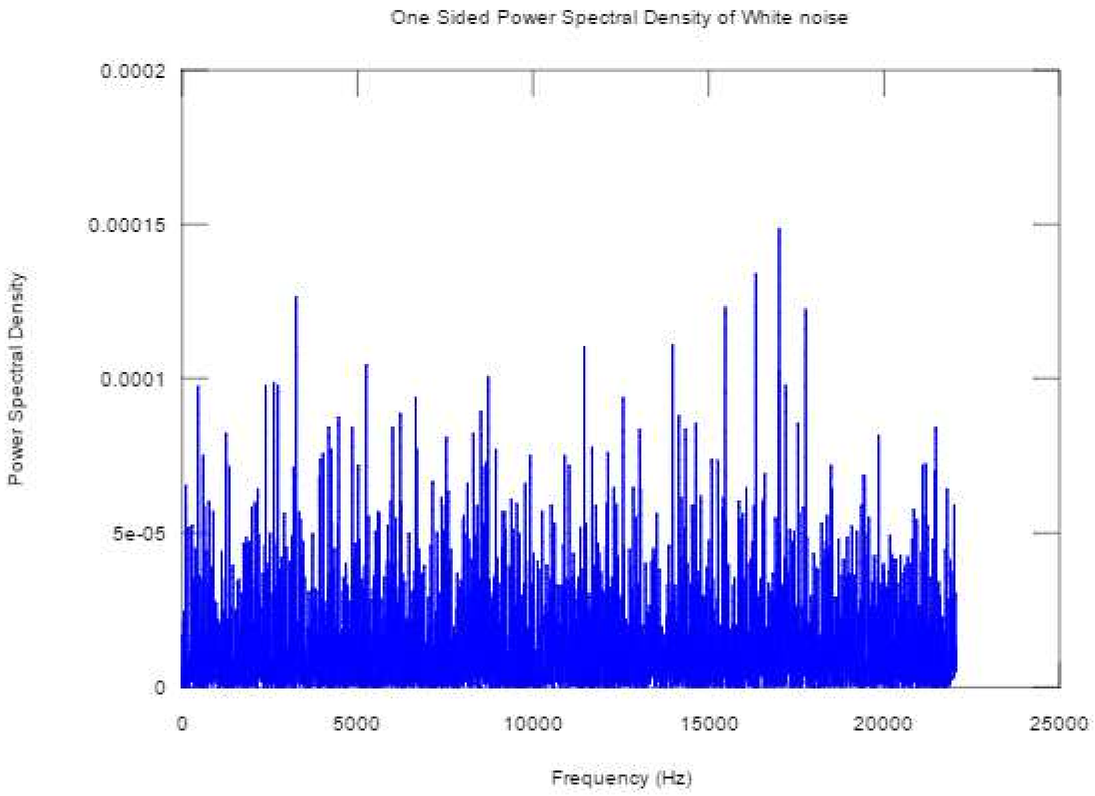


Figure 32: Spectral Density of White Gaussian noise

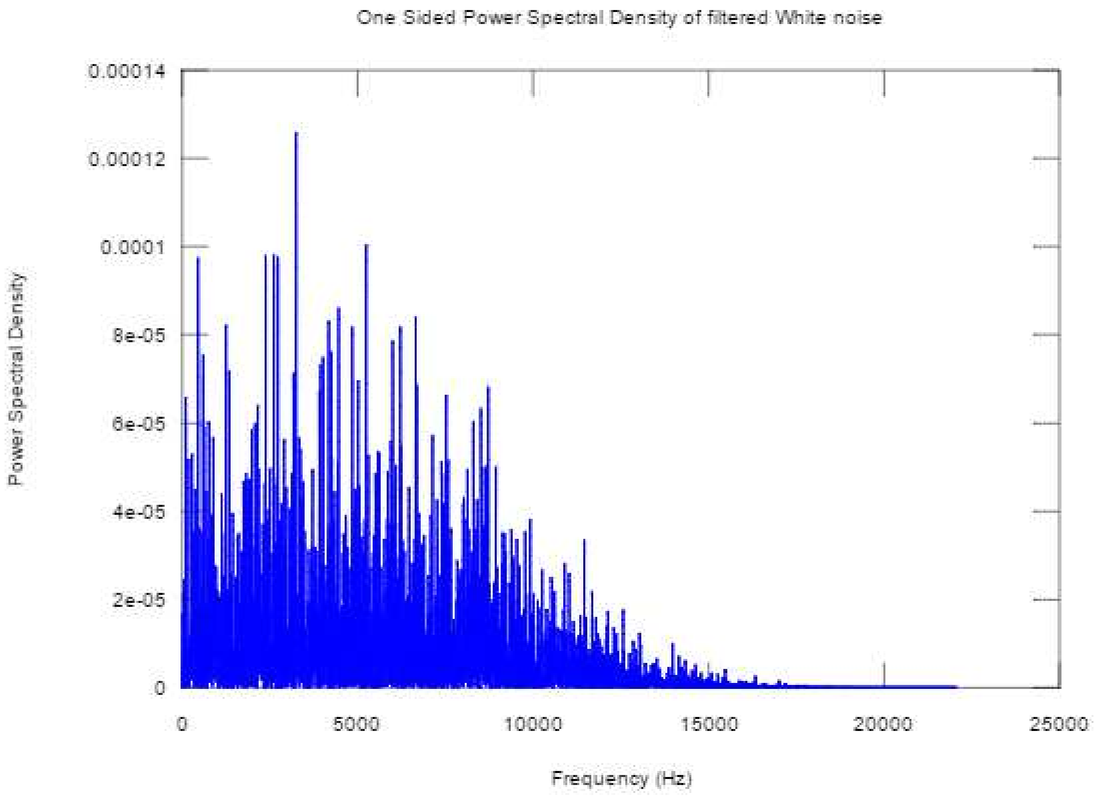


Figure 33: Spectral Density of lowpass filtered White Gaussian noise

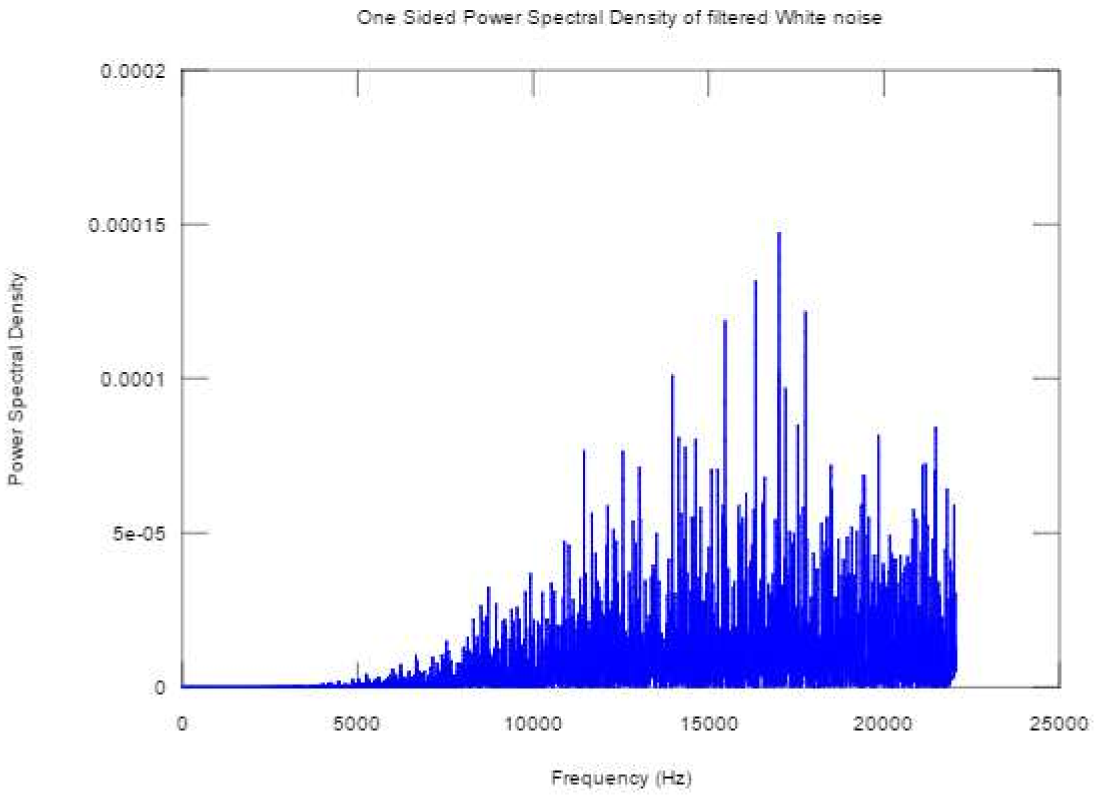


Figure 34: Spectral Density of highpass filtered White Gaussian noise

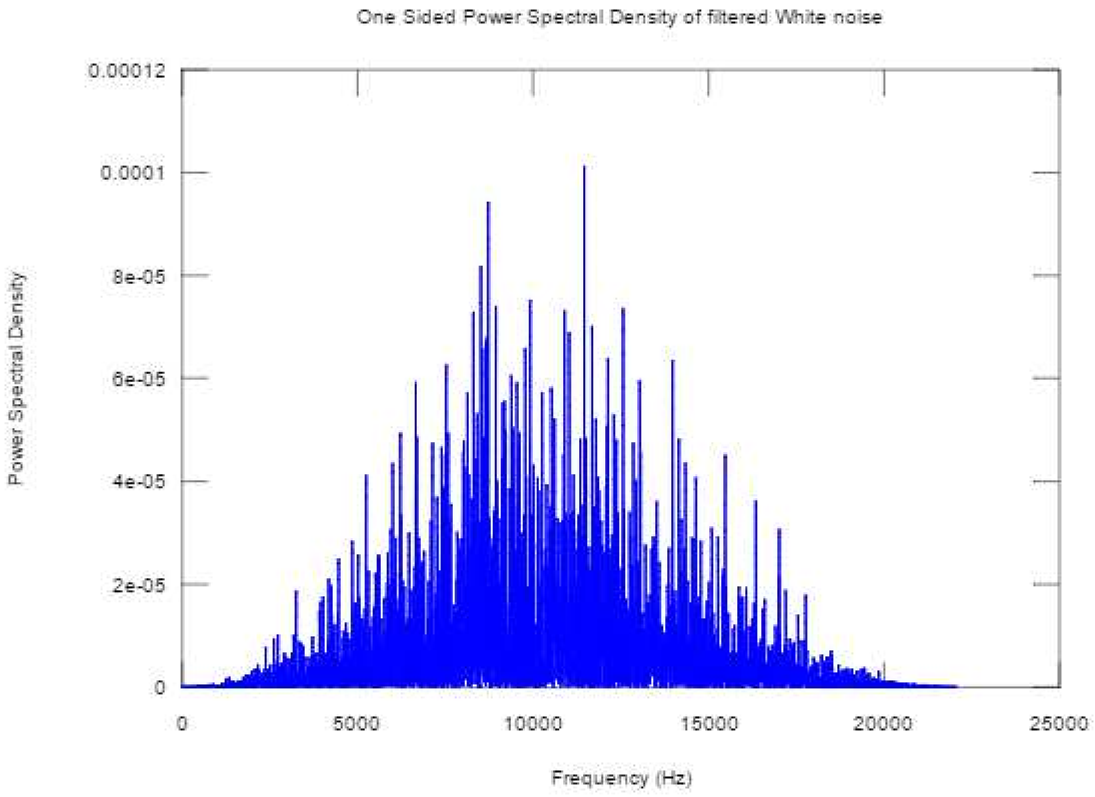


Figure 35: Spectral Density of bandpass filtered White Gaussian noise

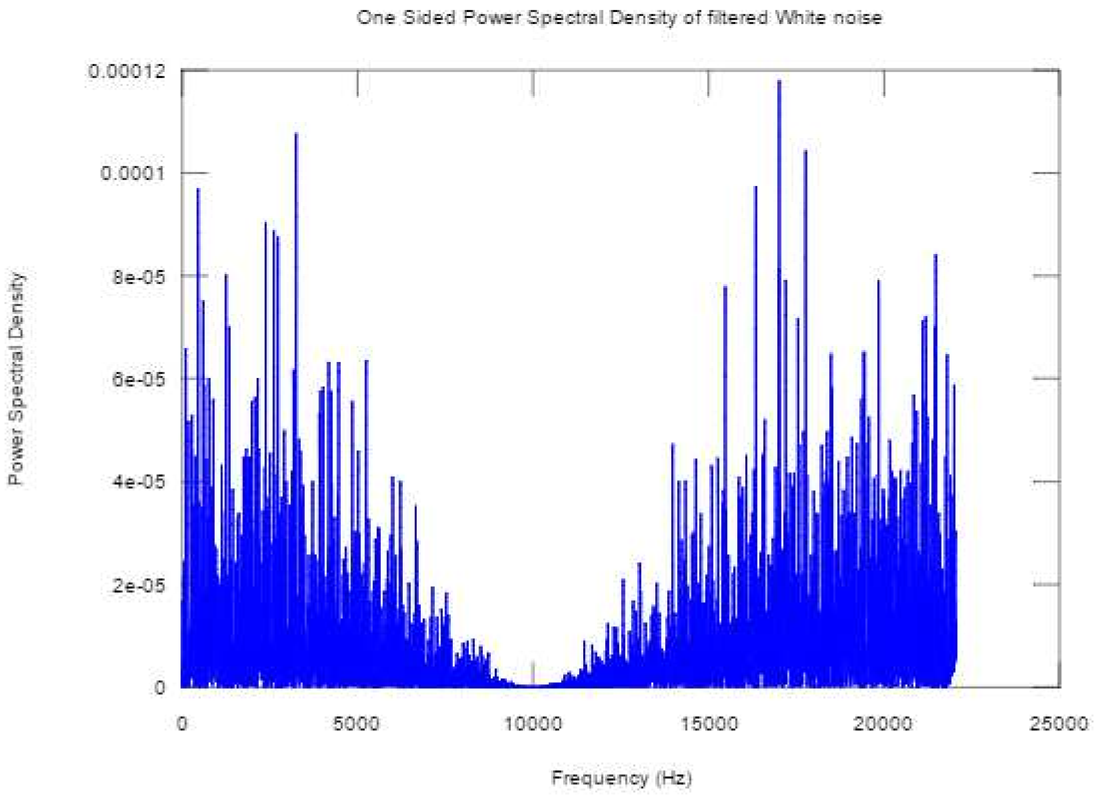


Figure 36: Spectral Density of bandreject filtered White Gaussian noise

4.4.9. Parameter Managers

Parameter managers are classes creating VST automations and binding. Each effect that has a VST automation has its own manager and each instance of the class will create the defined automations. For example, in AdsrEnvelopeParameterManager there are automations for each parameter of Attack, Decay, Sustain and Release.

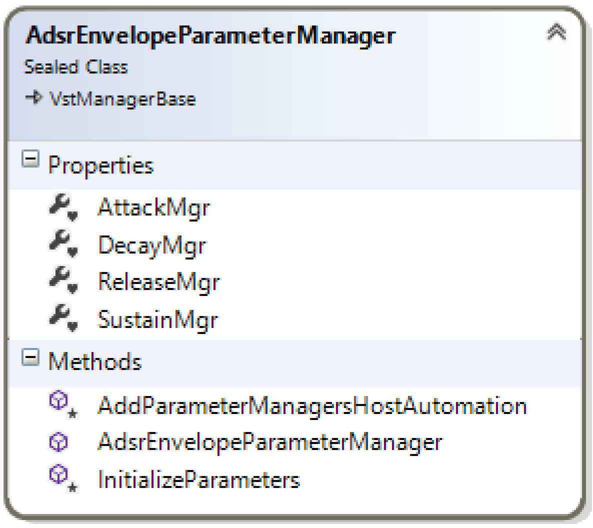
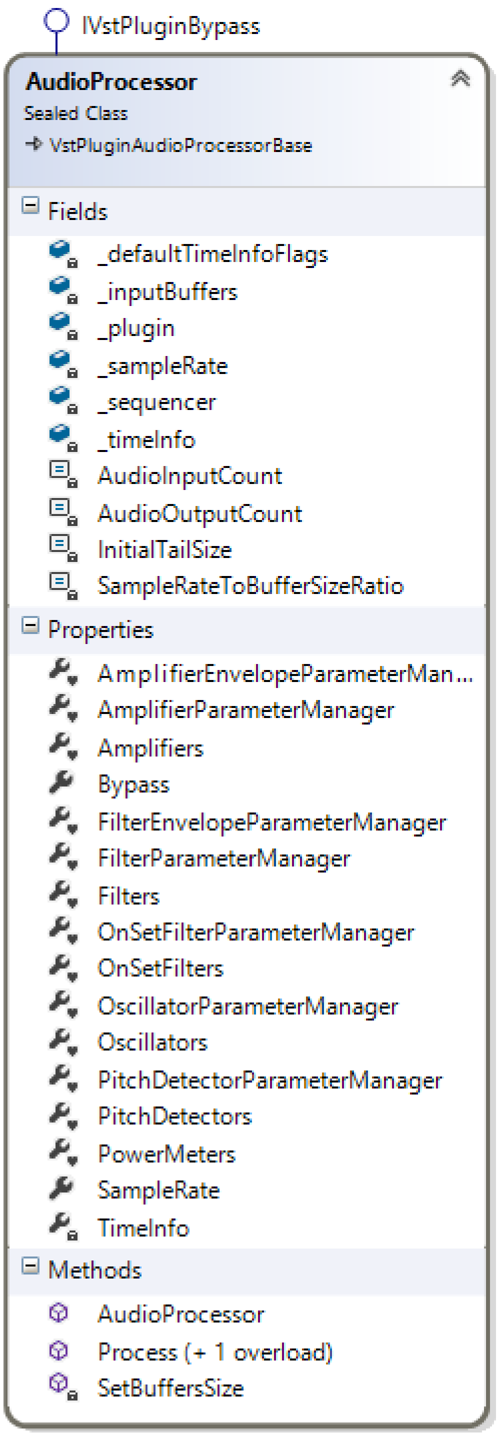


Figure 37: AdsrEnvelopeParameterManager class

4.4.10. Audio processor

Audio processor is the base class defining the processing pipeline. In constructor all the effects and managers are created and event and data bindings are set. Then the *Process()* method receives the input sample and returns output sample. Inside this method the whole processing pipeline is done.


IVstPluginBypass

AudioProcessor
Sealed Class
→ VstPluginAudioProcessorBase

Fields
_defaultTimeInfoFlags
_inputBuffers
_plugin
_sampleRate
_sequencer
_timeInfo
AudioInputCount
AudioOutputCount
InitialTailSize
SampleRateToBufferSizeRatio

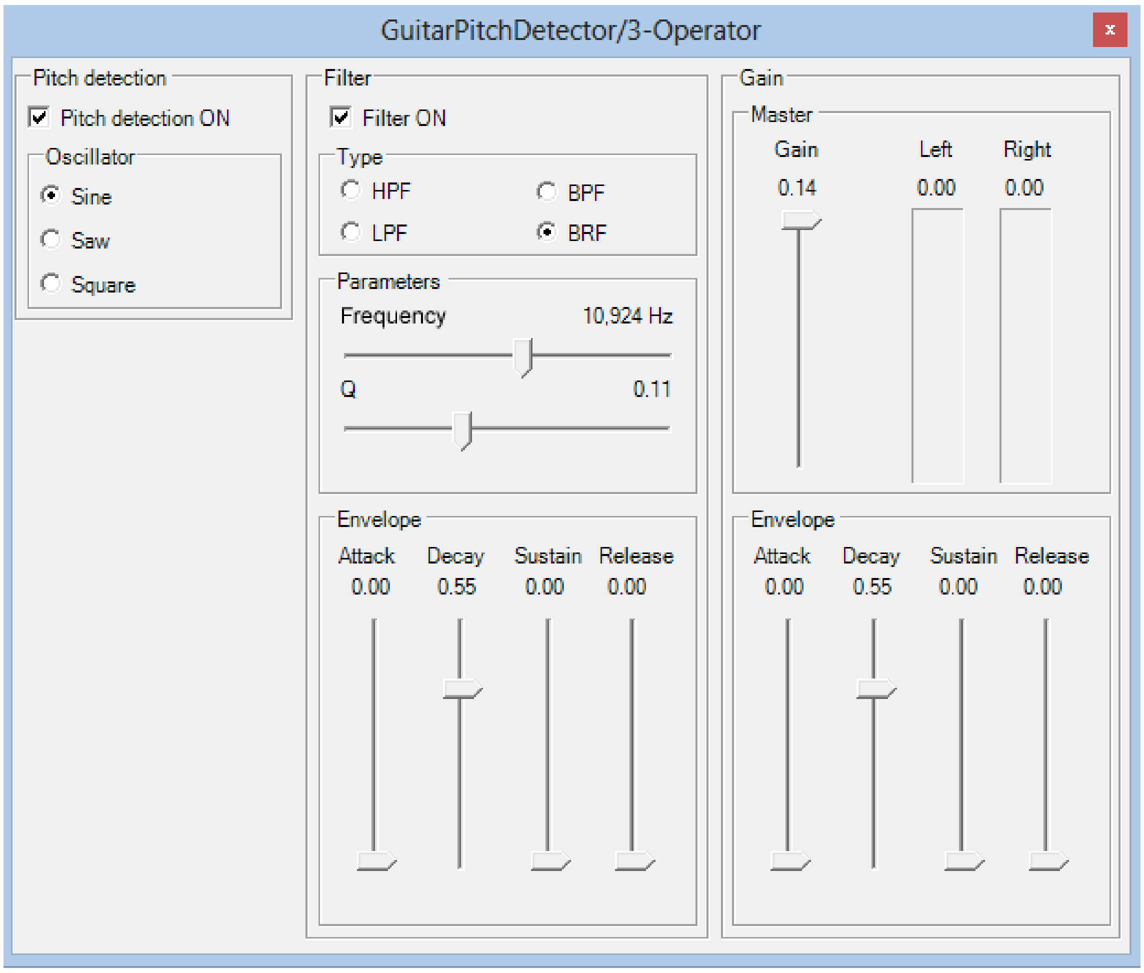
Properties
AmplifierEnvelopeParameterMan...
AmplifierParameterManager
Amplifiers
Bypass
FilterEnvelopeParameterManager
FilterParameterManager
Filters
OnSetFilterParameterManager
OnSetFilters
OscillatorParameterManager
Oscillators
PitchDetectorParameterManager
PitchDetectors
PowerMeters
SampleRate
TimeInfo

Methods
AudioProcessor
Process (+ 1 overload)
SetBuffersSize

Figure 38: AudioProcessor class

4.5. Graphic user interface

To create graphic user interface (GUI) I have used WinForms technology. It provides extensive amount of tools and controls for creating and programming GUI. One of the most useful feature of the controls are the events, which allows me to utilize the observer pattern.


GuitarPitchDetector/3-Operator

Pitch detection
[✓ Pitch detection ON]
Oscillator
[✓ Sine] Saw Square

Filter
[✓ Filter ON]
Type
HPF BPF LPF [✓ BRF]
Parameters
Frequency 10,924 Hz
Q 0.11
Envelope
Attack 0.00 Decay 0.55 Sustain 0.00 Release 0.00

Gain
Master
Gain 0.14 Left 0.00 Right 0.00
Envelope
Attack 0.00 Decay 0.55 Sustain 0.00 Release 0.00

Figure 39: Plug-in GUI

4.6. Tests and Verification

Software was tested on sequencer and VST host program Ableton Live. Algorithms of all synthesizer blocks were verified using Octave or Matlab. Source codes are included on CD.

4.7. Compilation

In post-build actions there are defined merge action for the output and C#.NET library using IL Merge library. The merged library with extension *vstdll* is created and ready for use in DAW.

4.8. 3rd Party libraries

The plugin is based on VST.NET library, which is under LGPL licence. As other library was used Exocortex.DSP containing implementation of complex numbers and FFT.

5. CONCLUSION

Aim of this thesis was to design and implement algorithm for pitch detection of guitar signal in VST technology.

In theoretical part I deal with digital sound processing in general and focus on the methods of pitch extraction, their types, positive and negative sides and suitability for usage in real-time processing and guitar and bass signals and detecting the edges of the notes. For one of these methods I designed solution as the VST plug-in.

In practical part the chosen method of pitch detection was implemented as VST plug-in effect supplied with synthesizer using oscillator generating varies waves, filter and amplifier, both driven by ADSR envelope.

From testing I managed, computationally the most difficult and slowest block was the pitch detection, which can cause processing delay up to 50ms.

5.1. Future Work Perspectives

As mentioned before, PD is not straightforward technique and needs to be researched deeper. Until we know the principles of human pitch perception, we need to try to find optional way of reliable PD. There is possibility for deeper research in the field of monophonic sounds, but for extensive and practical usage will be needed especially algorithms, which are able to extract pitch from polyphonic signals.

5.2. Personal Conclusion

This project brought me lots of new experience, opportunity to get deeper into signal processing, see the way the DSP works, better understand real-time VST technology and new perspectives of viewing signal processing. I have had the opportunity to consult my thoughts and ideas with internationally acknowledged specialist.

BIBLIOGRAPHY

[1] H. F. Olson, „The Measurement of Loudness,“ 1972.

[2] G. Leventhall, „What is infrasound?,“ sv. 93, č. 1–3, 2006.

[3] J. Hartquist, Real-time Musical Analysis of Polyphonic Guitar Audio, 2012.

[4] D. Smith, „Signals, Samples and Stuff: A DSP Tutorial - Part 1,“ Newington, 1998.

[5] „Information Signals,“ 2015. [Online]. Available: https://www.princeton.edu/~cuff/ele201/kulkarni_text/digitizn.pdf. [Přístup získán 4 3 2016].

[6] C. Roads, The computer music tutorial, The MIT press, 1996.

[7] L. Tan, Digital Signal Processing: Fundamentals and Applications, location unknown: Academic Press, 2008.

[8] U. Zölzer, DAFX: Digital Audio Effects, 2nd editor, location unknown: John Wiley & Sons, 2011.

[9] „VST Plug-Ins SDK Documentation,“ [Online]. Available: http://ygrabit.steinberg.de/~ygrabit/public_html/vstsdk/OnlineDoc/vstsdk2.3/. [Přístup získán 17 4 2016].

LIST OF ABBREVIATIONS

ADC	Analog-to-Digital converter
ADSR	Attack-Decay-Sustain-Release Envelope
BPF	Bandpass Filter
BRF	Bandreject Filter
DAC	Digital-to-Analog converter
DAW	Digital audio workstation
DSP	Digital Sound Processing
DTF	Discrete Fourier Transformation
FFT	Fast Fourier Transformation
GUI	Graphical User Interface
HPF	Highpass Filter
LFO	Low Frequency Oscillator
LPF	Lowpass Filter
OS	On-Set Filter
PD	Pitch Detection
STFT	Short-Time Fourier Transformation
VCA	Voltage-Controlled Amplifier
VCF	Voltage-Controlled Filter
VCO	Voltage-Controlled Oscillator
VST	Virtual Studio Technology

LIST OF FIGURES

LIST OF TABLES

LIST OF ATTACHMENTS

CD containing VST plug-in, MatLab sources and result images