FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Název:	Využití zvukové karty jako osciloskopu
Student:	Marek Reimer
Vedoucí:	doc. Ing. Ivan Šimeček, Ph.D.
Studijní program:	Informatika
Studijní obor:	Webové a softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	Do konce letního semestru 2018/19

Pokyny pro vypracování

1) Proveďte průzkum existujících řešení využívající zvukovou kartu jako osciloskop [1,2].
2) Proveďte rešerši omezení vzniklých použitím zvukové karty jako osciloskopu.
3) Zvolte vhodný programovací jazyk a technologii pro vizualizaci.
4) Navrhněte, implementujte a otestujte aplikaci.Aplikace bude mít následující funkce:grafické zobrazování sledovaného napětí, výpočet efektivního napětí, měření frekvence.
5) Zhodnoťte použitelnost řešení, navrhněte opravy případných nedostatků.

Seznam odborné literatury

[1] http://engineering.case.edu/lab/circuitslab/sites/engineering.case.edu.lab.circuitslab/files/docs/Oscilloscope_Fundamentals_-_Tektronix.pdf
[2] Christian Zeitnitz: Soundcard Oscilloscope, https://www.zeitnitz.eu/scope_en

Ing. Michal Valenta, Ph.D.
vedoucí katedry

doc. RNDr. Ing. Marcel Jiřina, Ph.D.
děkan

V Praze dne 27. ledna 2018

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Bakalářská práce

Využití zvukové karty jako osciloskopu

Marek Reimer

Katedra softwarového inženýrství
Vedoucí práce: doc. Ing. Ivan Šimeček, Ph.D.

14. května 2018

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů, zejména skutečnost, že České vysoké učení technické v Praze má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona.

V Praze dne 14. května 2018 .....

Odkaz na tuto práci

Reimer, Marek. *Využití zvukové karty jako osciloskopu*. Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2018.

Abstrakt

Tato práce se zabývá problematikou použití zvukové karty jako osciloskopu. Součástí práce je rešerše existujících řešení, analýza omezení vyplývajících z použití zvukové karty za tímto účelem, vývojem a implementací vlastního řešení. V závěru práce je zhodnocena použitelnost a užitečnost vytvořené aplikace a případné vylepšení nedostatků.

Klíčová slova osciloskop, zvuková karta, analýza eletrického signálu, vizualizace dat, grafické rozhraní, Qt, frekvenční analýza

Abstract

This thesis explores the possibility of using a sound card as an oscilloscope. It includes comparison of existing solutions, analysis of limititations of resulting from using a sound card for this purpose and development of our own solution. In the conclusion we assess the usability and usefulness of the developed application.

Keywords oscilloscope, sound card, analysis of electrical signal, data visualisation, graphical interface, Qt, frequency analysis

Obsah

Seznam obrázků

Úvod

V elektrotechnice je osciloskop běžně používaným nástrojem, jehož hlavním účelem je pomoci uživateli provést analýzu elektrického signálu a jeho změn v závislosti na čase. Za tímto účelem je jednou z hlavních vlastností osciloskopů vizualizace zkoumaného signálu, měření napětí, frekvence, a podobně. Komerčně dostupné osciloskopy jsou většinou samostatná zařízení, modernější verze umožňují přesměrovat výstup do počítače pro další analýzu, popřípadě uložit do vlastní paměti pro následné zpracování.

Zvuková karta je zařízení umožňující připojení zvukových periferií, primárně určených k nahrávání čí přehrávání zvuku. Přesněji k zaznamenávání elektrického signálu, který generuje elektromechanický měnič - mikrofon, a nebo v opačném směru k vytváření elektrického signálu například pro sluchátka, nebo malý reproduktor. Dnes je taková karta jednou ze standardních částí drtivé většiny osobních počítačů. Jednou z metod přenosu zvuku mezi zařízeními je analogový signál, a zde vzniká téma této práce. Zvuková karta totiž obsahuje všechnu potřebnou elektroniku pro převod analogového signálu na digitální data. Toto umožňuje vytvořit aplikaci dovolující analýzu takového signálu, neboli osciloskop.

Kapitola 1

Cíle práce

Zastřešujícím cílem této práce je vývoj aplikace poskytující některé funkce osciloskopu za použití zvukové karty jako vstupního zařízení. Komerčně dostupné osciloskopy dnes nabízejí mnoho funkcí, které usnadňují práci jejich uživatelům. Obsahem této práce bude implementace pouze základních z nich. Zejména se bude jednat o grafické vykreslování signálu, výpočet efektivního napětí, měření frekvence, a u signálů střídavého napětí také tzv. peak–to–peak napětí. Za tímto účelem bude proveden průzkum existujících řešení a jejich následné porovnání, což bude více rozvedeno v sekci o existujících řešeních.

Následujícím bodem zadání je analýza omezení vzniklých použitím zvukové karty jako převodníku zkoumaného analogového signálu na digitální hodnoty zpracovatelné v aplikaci. Samozřejmě výrobci zvukových karet optimalizují své produkty pro přehrávání a nahrávání zvuku, neboli člověku slyšitelného vlnění. Tato skutečnost klade zásadní překážky jak přesnosti prováděného měření, tak finální užitečnosti takové aplikace.

Kapitola 2

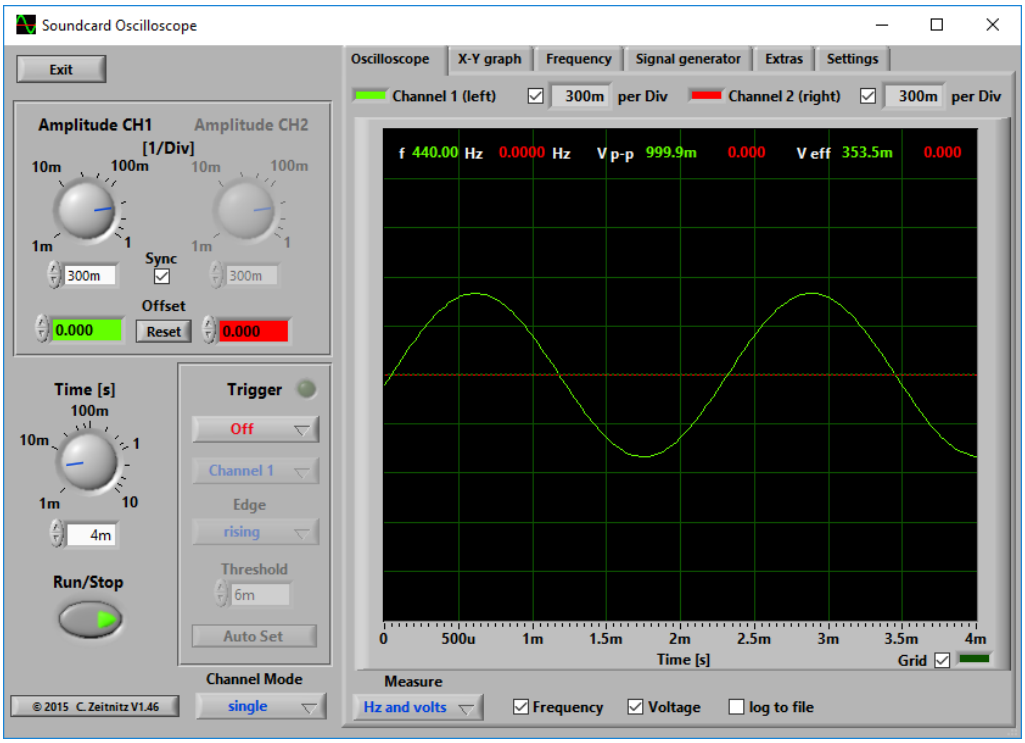
Průzkum existujících řešení

Možnost využití zvukové karty jako osciliskopu není revoluční myšlenka, zvukové karty nabízejí velmi snadnou digitalizaci analogového signálu a tím umožňují prostřednictvím aplikace tento signál analyzovat. Z tohoto důvodu byly zvukové karty za tímto účelem používány alespoň od roku 1996 [1]. Přes 20 let používání má za následek množství existujících řešení, v této práci zmíním ty, které se mi podařilo zprovoznit a zároveň implementují funkcionalitu požadovanou zadáním práce, nepožadují přidaný hardware a nejsou placené. Vzhledem k množství těchto programů, zmíním jen ty, které považuji za kvalitně provedené a nebo se něčím odlišující.

2.1 Soundcard scope

Soundcard scope[2] je program vyvinutý Christianem Zeitnitzem. Pro soukromé a vzdělávací použití je zdarma, pro komerční využití je potřeba zakoupit licenci. Tento program implementuje nejen všechny požadavky stanovené zadáním této práce, ale navíc ještě implementuje zobrazování vícekanálových vstupů na X-Y graf, čímž rozšiřuje možnosti aplikace k vykreslování např. Lissajousových obrazců [3]. Dále pak graf frekvenčního spektra, generátor signálů, uložení signálu jako audio souboru nebo ukládání sejmutého signálu jako tabulkový soubor pro pozdější analýzu. Rovněž také umožňuje výběr vstupního a výstupního zařízení včetně loopbacku zpět do programu a kalibraci měřeného napětí. Grafické rozhraní je jednoduché a plně použitelné, všechny parametry lze snadno nastavit ať už grafickou reprezentací potenciometrů, nebo textovým vstupem.

Zásadním nedostatkem tohoto programu je, že používá pouze základní 44,1 kHz vzorkovací frekvenci a 16 bitovou hloubku vzorkování i v případě, že zvuková karta podporuje vyšší frekvenci, nebo hloubku. Z těchto ome-



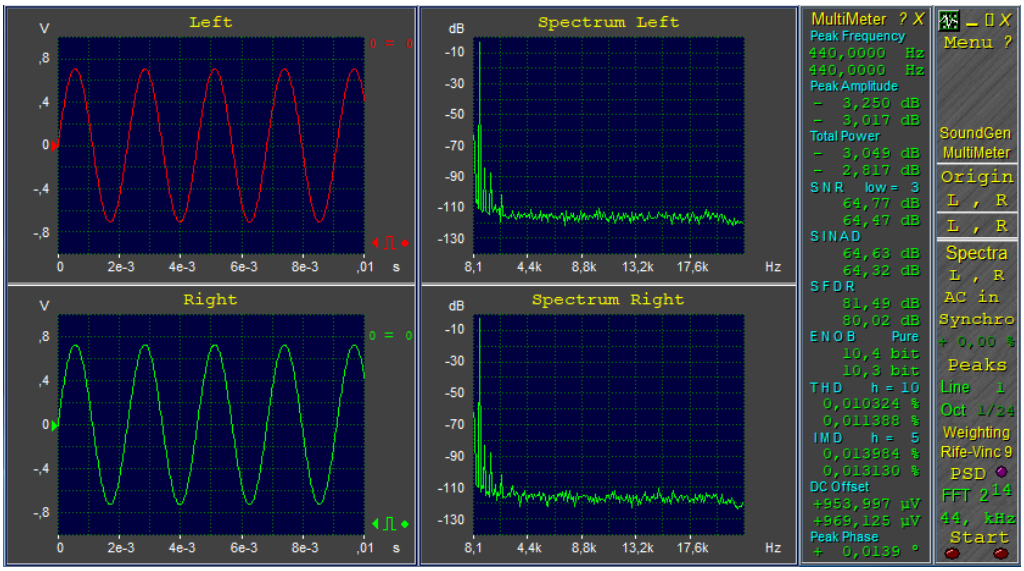
Obrázek 2.1: Soundcard scope

zení má nízká vzorkovací frekvence vyšší dopad, neboť výrazně omezuje schopnosti a následně i užitečnost programu pouze na frekvence do cca 20 kHz. Bohužel je tento program pouze pro platformu Windows, ale to je dle mého názoru menší nedostatek než ty které byly zmíněny výše.

Pokud výše zmíněná omezení nepředstavují překážku zamýšlenému využití, pak tento program poskytuje poměrně rozsáhlé množství funkcí, které jsou k nalezení v komerčně dostupných osciloskopech a proto jej považuji za velmi zdařilý, a z řešení, která jsou v této práci popsaná bych jej považoval za nejlepší.

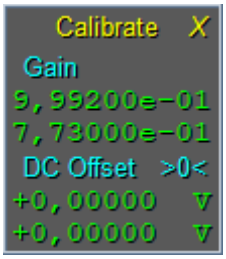
2.2 OscillometerXZ

Dalším existujícím řešením je *OscillometerXZ*[4]. Tento program je více zaměřený na analýzu z pohledu radioamatéra a proto je mnoho údajů uváděno v decibelech. Podobně jako *Soundcard scope* podporuje standardní možnosti vykreslování signálu, výpočet frekvence a efektivního napětí a graf frekvenčního spektra. Na rozdíl od *Soundcard scope* podporuje i výběr vzor-



Obrázek 2.2: OscillometerXZ

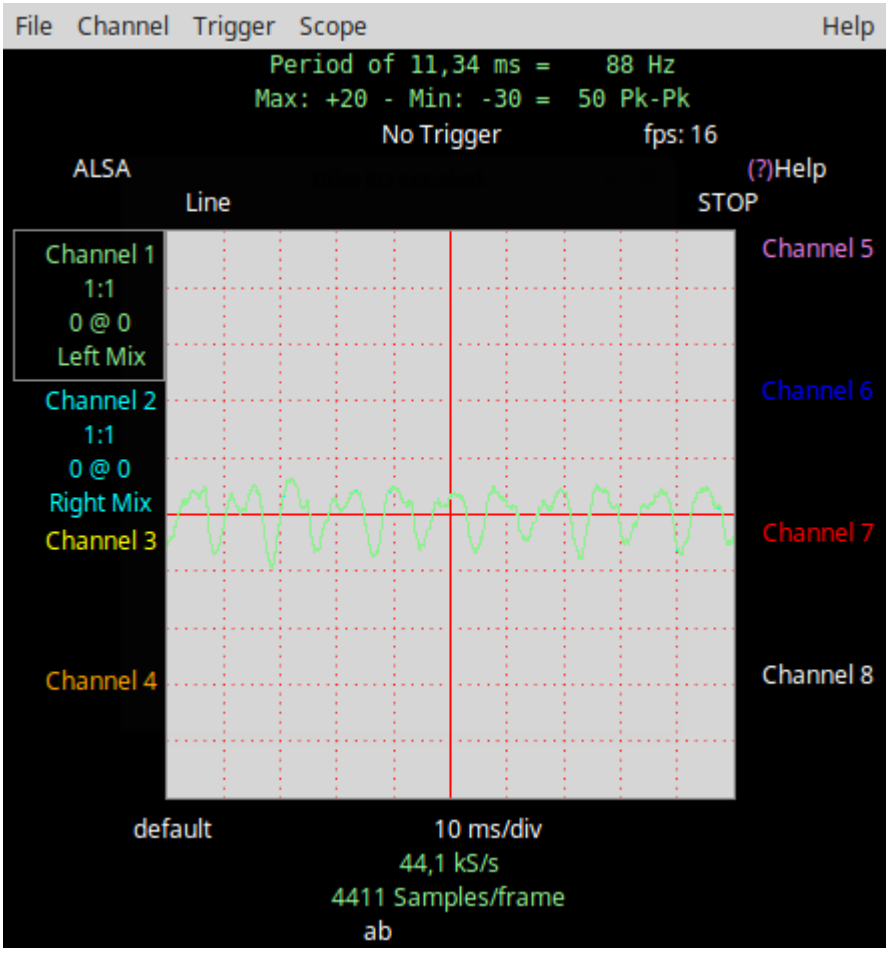
kovací frekvence a bitové hloubky naměřených vzorků signálu, odstraňující tento nedostatek. Avšak okamžitě jej nahrazuje nedostatky jinými. Ve výběru vstupního zařízení jsou na výběr i hodnoty vzorkovací frekvence a bitové hloubky vzorků, které dané zařízení nepodporuje a dojde tak ke zkreslení celého měření. Dalším nedostatkem je grafické rozhraní, které je poměrně nešťastně řešené protože je rozděleno na mnoho nezávislých oken, a chvíli trvá než se v něm uživatel zorientuje, a v některých případech je velmi nepraktické. Například při kalibraci citlivosti vstupu zvukové karty, se hodnota napětí nastavuje tak, že uživatel kliká na jednotlivé cifry desetinného čísla kde levý klik danou cifru sníží o jedna, zatímco pravý klik tuto cifru zvýší 2.3. Podobně jako *Soundcard scope* je *OscillometerXZ* také pouze pro platformu Windows a jeho nejnovější verze je z roku 2005. Jde tedy o poměrně zastaralou aplikaci bez dalšího vývoje a podpory.



Obrázek 2.3: Kalibrace OscillometerXZ

2.3 Xoscope

Xoscope[5] je jedním z existujících řešení které nativně podporují platformu Linux, a je open–source. Podobně jako výše zmíněná řešení je *Xoscope* schopen graficky zobrazit průběh signálu a vypočítat frekvenci, bohužel neprovádí výpočet efektivního napětí, pouze měří napětí peak–to–peak. Zmiňuji ho ve svém průzkumu i přes nedostatek chybějícího výpočtu efektivního napětí z důvodu, že *Xoscope* je jediný program tohoto typu který je stále udržovaný.



Obrázek 2.4: Xoscope

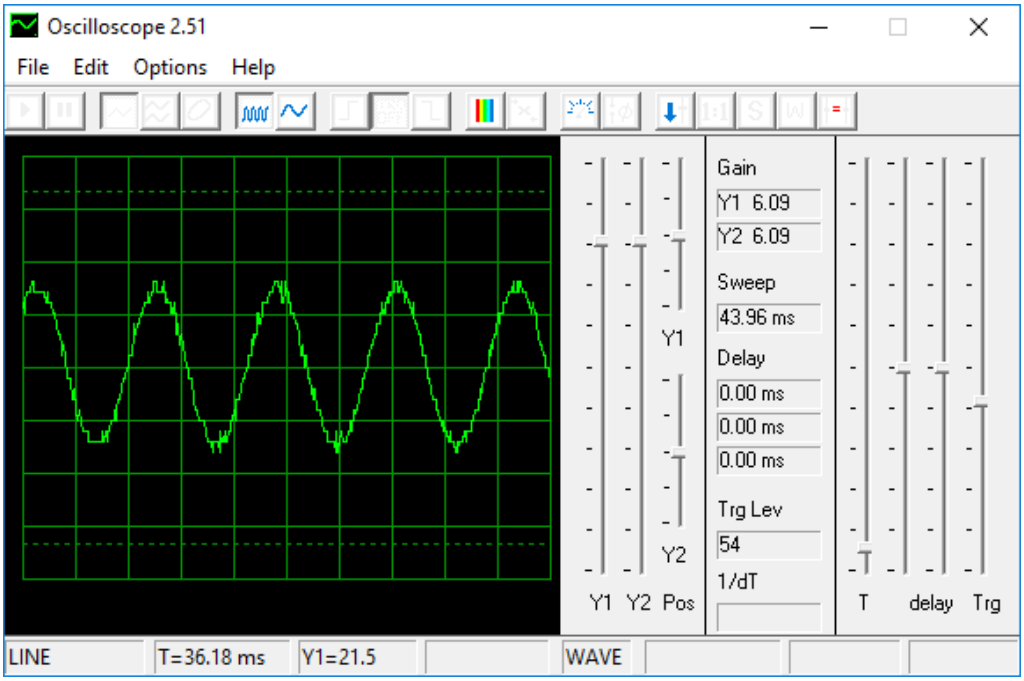
Xoscope implementuje opravdu pouze základní funkce jako již zmíněné měření peak–to–peak napětí, výpočet frekvence, změnu časové osy. Naopak nepodporuje například kalibraci měřeného napětí nebo posun signálu podél svislé osy. Z tohoto důvodu je užitečný pouze na zkoumání průběhu signálu

a ne ke zjišťování a zpracování konkrétních hodnot parametrů zkumaného signálu. Další nevýhodou je volba barev uživatelského prostředí 2.4, pozadí kreslící plochy pro graf je světle šedé a barva prvních dvou kanálů je světle zelená, respektive světle modrá, toto značně ztěžuje používání a nenašel jsem žádnou možnost změny konfigurace barev.

Jak již bylo zmíněno, tento program je open–source a je distribuován pouze ve formátu zdrojového kódu a každý uživatel si ho tedy musí sám zkompilovat. Pro běžného uživatele operačního systému Linux je toto jen malá překážka, ale na platformě Windows to může klást mnoha uživatelům značné problémy.

2.4 Winscope

Winscope [6] je, jak již název napovídá, dalším z řešení pouze pro platformu Windows. Implementuje všechny funkce požadované zadáním práce a navíc ještě XY vykreslování a zobrazení rozkladu frekvencí pomocí Fourierovy transformace. Jednou ze zajímavých funkcí je nastavení úrovně snímání které umožňuje odstranění šumu ve vykresleném signálu.



Obrázek 2.5: Winscope

Bohužel je nastavování parametrů řešeno pomocí táhel a tak je například kalibrace vstupního napětí obtížná. Podobně jako *Soundcard scope* je i u *Winscope* vzorkovací frekvence pevně nastavená na 44,1 kHz a tedy má stejné omezení ve snímání vstupního signálu do přibližně 20 kHz i v případě, že použitá zvuková karta podporuje vyšší vzorkovací frekvenci.

Kapitola 3

Omezení způsobená použitím zvukové karty

Zvukové karty jsou, jak již název napovídá, zařízení používaná pro vstup a výstup zvuku v počítačích, nebo přesněji řečeno, ke vstupu a výstupu elektrického signálu reprezentujícího zvuk. Zvukové karty totiž neobsahují elektromechanické převodníky, tedy mikrofony či reproduktory. Toto zaměření na zvuk má za následek několik omezení při jejich použití jako osciloskop. Rozsah těchto omezení se liší dle dané zvukové karty, některá z těchto omezení jsou řešitelná přidáním externí elektroniky, v některých případech je ale problém tato omezení před použitím zjistit. V této práci budeme předpokládat, že uživatel bude pravděpodobně používat integrovanou zvukovou kartu, tedy v podstatě kartu s nejnižšími použitelnými parametry pro reprodukci zvuku.

3.1 Měřitelné frekvence

Zásadním neřešitelným omezením je rozsah měřitelná frekvence. Při nahrávání zvuková karta převádí vstupní analogový signál na řadu diskrétních hodnot metodou nazývanou vzorkování. Vzorkování přečte a uloží hodnotu signálu ze vstupu s četností nazývanou vzorkovací frekvence. Zaměření zvukových karet na slyšitelný zvuk má za následek, že mnoho běžných zvukových karet podporuje pouze minimální potřebnou vzorkovací frekvenci pro nahrávání či přehrávání lidským uchem slyšitelné zvuky. Lidské ucho je schopno detekovat zvuky v rozsahu přibližně v rozsahu 20 Hz – 20 kHz[7], tento rozsah se samozřejmě liší mezi jednotlivci a zároveň se zužuje následkem stárnutí či poškozením sluchové soustavy.

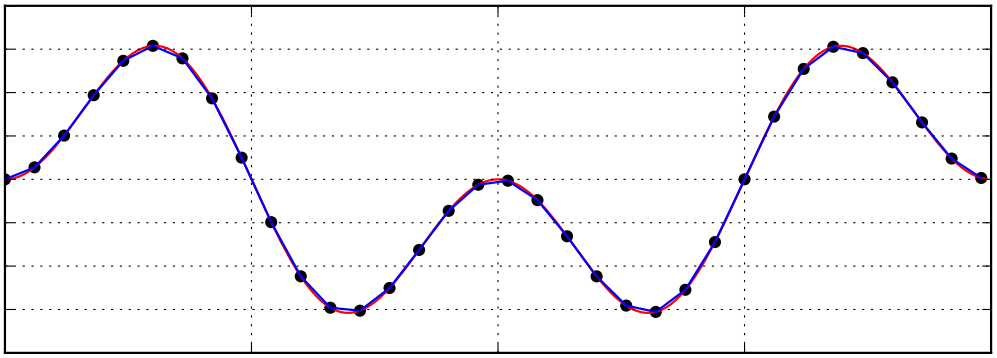
Nyquistův teorém[8] říká, že přesná rekonstrukce signálu je možná pouze pokud je vzorkovací frekvence dvojnásobek nejvyšší frekvence obsažené v daném signálu. Důsledek tohoto teorému v audio průmyslu byl například takový, že pro CD byla zvolena frekvence 44,1 kHz jako dvojnásobek slyšitelných 20 kHz a rezerva. Pro účely této práce to znamená, že aplikace bude schopna analyzovat signály maximálně do frekvence rovné polovině vzorkovací frekvence dané zvukové karty. Lze předpokládat, že každá zvuková karta v dnešní době bude podporovat vzorkovací frekvenci aspoň 44,1 kHz.

Při převodu analogového signálu na posloupnost vzorků dojde ke ztrátě informací, protože počet vzorků je konečný, přičemž rozsah této ztráty je daný granularitou vzorkování. V případech, kde je vzorkovací frekvence mnohonásobně vyšší než frekvence sledovaného signálu, je ztráta nízká až zanedbatelná (pro účely poslechu) jak je znázorněno na obrázku 3.1. Zatímco v případech, kde se frekvence signálu blíží zlomku vzorkovací frekvence už je ztráta informace vysoká. Na obrázku 3.2 je znázorněna nepřesnost vzorkování v situaci, kde vzorkovací frekvence je sedmínásobkem frekvence signálu, i přestože mezi signálem a vzorkováním není posun ve fázi. Posun ve fázi mezi měřeným signálem může měření ovlivnit oběma směry v závislosti na okolnostech. Například pokud vzorkovací frekvence bude dvojnásobkem frekvence signálu a nedojde k posunu fáze, nastane situace, kde vzorkování proběhne v místě, kde měřený signál protíná vodorovnou osu a následně bude ztracena všechna informace. Zatímco v situaci, kdy budou fáze posunuté dojde ke ztrátě pouze většiny informací. Se zvyšujícím se poměrem vzorkovací frekvence vůčí frekvenci signálu se vliv posunu fáze snižuje, až zanikne.

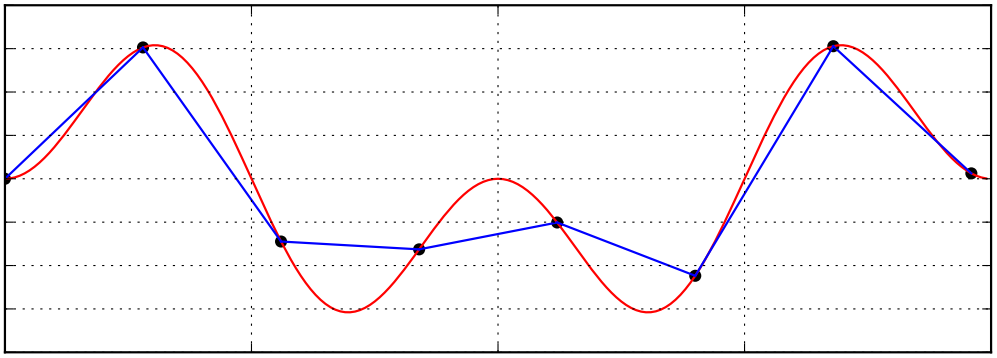
V případě, že dojde pouze k částečné ztrátě informací je možné je přibližně zrekonstruovat, například pomocí interpolace polynomem vysokého stupně, který prochází naměřenými vzorky.

3.2 Rozsah napětí

Schopnost měřit napětí je řešitelné omezení. I komerčně dostupné osciloskopy nabízí možnost jednoduchého děliče napětí již v sondě, a to nejčastěji v poměru 1/10. Komplikace se nacházejí především v neznalosti parametrů převodu hodnoty napětí na vstupu zvukové karty na diskrétní hodnoty, které následně vytváří analog–to–digital převodník. Tuto situaci lze řešit připojením referenční hodnoty napětí, a podle výstupů AD převodníku zobrazených prostřednictvím aplikace nastavit převodní koeficient, tak, aby zobrazované údaje byly shodné se vstupními referenčními. Tento problém nastává z neznalosti daného AD převodníku. Teoreticky každá zvuková



Obrázek 3.1: Dostatečný počet vzorků



Obrázek 3.2: Nedostatečný počet vzorků

karta může používat zcela jiný převodník a proto nejsem schopen zajistit přesnost měření.

V případě, že nebude použit žádný přidaný hardware (jako např. již zmíněná sonda s děličem napětí), je potřeba zohlednit limitace dané zvukové karty. Standardně mají karty vstup pro mikrofon a pokud je prostor pak mají i tzv. „line–in“ vstup který, se používá pro přenos audio signálu mezi zařízeními.

U běžných zvukových karet je vstup pro mikrofon specializován na levné mikrofony, které jsou většinou dynamického typu. U dynamických mikrofonů se signál vytváří pohybem cívky v magnetickém poli poskytovaném permanentním magnetem, tento pohyb generuje v cívce elektrické napětí v rozsahu jednotek až desítek milivoltů (dle konkrétního zařízení). Následkem této skutečnosti je, že mikrofonní vstupy na zvukových kartách jsou optimalizovány pro takto nízká napětí a proto vstupní signál projde

předzesilovačem než je AD převodníkem digitalizován.

Oproti tomu se na některých zvukových kartách, pravděpodobně většině zvukových karet ve stolních počítačích, nachází již zmíněný linkový vstup. Tady vzniká problém, že přenos audio signálu mezi zařízeními není přesně definován, ať už průmyslem samořízeně, nebo skutečnou normou. Pro běžného uživatele toto nevytváří problém, audio signál lze na jedné nebo obou stranách přenosu zesílit nebo zeslabit na požadovanou hlasitost, při použití jako osciloskop to ale vytváří komplikace. Nejčastěji používaná peak–to–peak hodnota linkového analogového signálu je 1V[9].

Jednou z komplikací je neznalost referenční hodnoty. Řekněme, že zvuková karta vzorkuje s rozlišením 16 bitů, hodnota 0 odpovídá napětí 0 V, hodnota – 32 768 odpovídá nejnižšímu měřitelnému napětí a hodnota 32 767 odpovídá nejvyššímu měřitelnému napětí. Protože neznáme konkrétní parametry aktuálně používané zvukové karty a tedy použitého AD převodníku, nejsme schopni zjistit, která hodnota z rozsahu odpovídá napětí například 1 V. Tato komplikace je řešitelná za předpokladu, že AD převodník se chová lineárně. Pokud je tento předpoklad splněný, je možno provést kalibraci pomocí známého vstupního signálu, který uživatel připojí na vstup a do aplikace zadá napětí daného signálu. Z toho lze vypočítat koeficient mezi hodnotou která reprezentuje naměřený vstup a skutečným napětím. Jiné řešení by bylo umožnit uživateli nastavit tento koeficient manuálně, kde uživatel zadá aplikaci takový koeficient, aby se hodnota napětí měřeného signálu rovnala hodnotě vstupního signálu.

Další komplikací je neznámost maximálního vstupního napětí kterou výrobce často neuvádí. Tato situace nastává především u integrovaných zvukových karet. V případě, že napětí měřeného signálu překročí hranice možností zvukové karty, dojde k ořezávání signálu a znemožní jeho analýzu. Pokud se uživatel pokusí o měření signálu s napětím výrazně vyšším než schopnosti karty, může dojít i k jejímu poškození, a proto bude potřeba uživatele varovat, aby na vstup zvukové karty nepřipojovali signály neznámých napětí. Obecně je možno považovat napětí do 2 V bezpečné pro linkové vstup, pokud dokumentace od výrobce neuvádí jinak.

3.3 Bitová hloubka

Posledním evidentním omezením je bitová hloubka AD převodníku, neboli „rozlišení“ jednoho vzorku. V praxi to je rozsah, hodnot kterých může jeden vzorek nabývat, přičemž tento rozsah koresponduje rozsahu napětí na vstupu ADC převodníku dané zvukové karty. Na vstupu je analogový signál omezený shora i zdola dle vlastností zařízení, které tento signál generuje, a

tento signál tedy může v daném rozsahu dosahovat nekonečného množství hodnot. V praxi samozřejmě neexistuje AD převodník s nekonečným množstvím hodnot na daném intervalu, což vede k nepřesnosti měření. Dopad této nepřesnosti záleží jak na schopnostech použitého AD převodníku, tak na požadované přesnosti měření. Historicky začínaly zvukové karty s bitovou hloubkou 6 bitů, poskytující 64 hodnot, dnes jsou již běžně dostupné zvukové karty s 16, 24 nebo i vyšší bitovou hloubkou. Pokud vezmeme spodní hranici moderních zvukových karet, tedy 16 bitů, získáme k dispozici celkem 65 535 diskrétních hodnot pro pokrytí otevřeného intervalu. Zda-li je to dostatek záleží na konkrétním použití, ale myslím si, že vzhledem k ostatním omezením výplývajících z použití zvukové karty má toto omezení nejnižší dopad na finální užitečnost aplikace.

Kapitola 4

Výpočet efektivního napětí

Efektivní hodnota střídavého napětí je vlastnost střídavého napětí vycházející z přenášené energie, nebo–li výkonu. Výkon elektrického zdroje je definován vztahem

$$p(t) = \frac{u^2(t)}{R} \tag{4.1}$$

kde $p(t)$ je výkon v čase t a $u(t)$ je napětí také v čase t. Vztah 4.1 určuje závislost mezi napětím a výkonem v jednom okamžiku, ale toto není v praxi moc užitečné, proto se používá průměrný výkon v časovém intervalu definován nasledovně

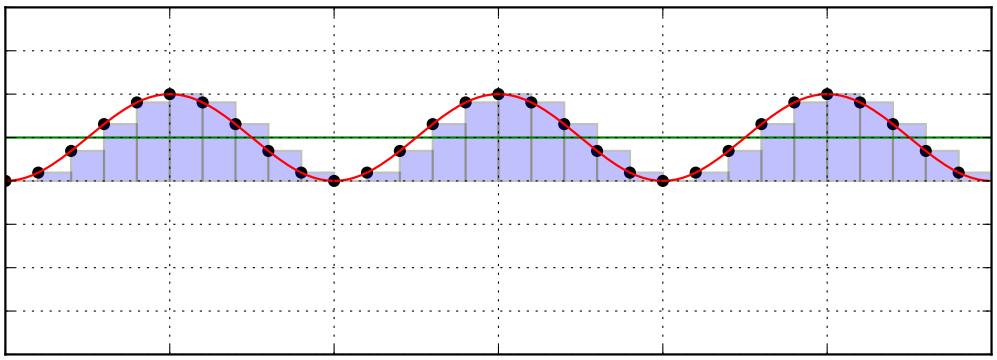
$$P_{average} = \frac{U_{ef}^2}{R} \tag{4.2}$$

V případě stejnosměrného zdroje napětí je velikost napětí konstantní a tedy je výkon velmi snadné vypočítat.

V případě, že situaci zjednodušíme tím, že budeme předpokládat, že výkon měřeného napětí bude realizován na zátěži 1 Ω, tak se efektivní napětí vypočte z předpokládaného přenášeného výkonu jako obsah daného střídavého signálu, pokud ho zobrazíme jako kvadratickou funkci napětí (výkonu), kde horizontální osa bude odpovídat času a vertikální osa bude okamžitá hodnota kvadrátu signálu napětí (výkonu) v daném čase. Následně je takto vypočítaný obsah (reprezentace výkonu) odmocněn, a dostaneme tak hodnotu efektivního napětí.

Jeden ze zdrojů nepřesnosti při tomto výpočtu efektivního napětí bude již zmiňované vzorkování. A to především v případech kde se frekvence signálu blíží technickým limitům použitého AD převodníku. V této situaci už nebude vzorkovaný signál věrně reprezentovat vstupní analogový signál a tedy dojde k odchylce při výpočtu.

Vzhledem k tomu, že pomocí zvukové karty bude mít aplikace k dispozici řadu diskrétních hodnot, se nabízí několik možných způsobů, jak vypočítat obsah který křivka vymezuje. Pravděpodobně nejsnadnějším způsobem je sečíst obsah obdélníků, kde jedna strana obdélníku bude hodnota naměřeného vzorku a druhá bude určena vzrokovací frekvencí. Tato metoda může přinést značné nepřesnosti v měření v případě, kde se frekvence signálu bliží limitu AD převodníku a obdélníky pak nevyplňují plochu mezi signálem a vodorovnou osou, jak je znázorněno na obrázku 4, kde je znázorněn i průměrný výkon.

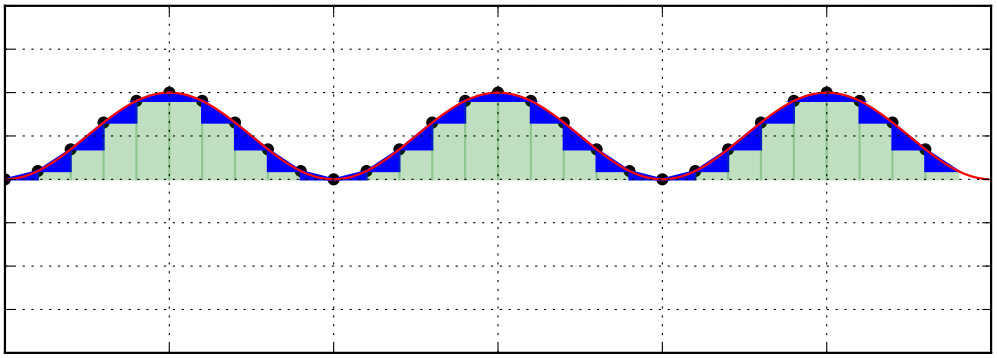


Obrázek 4.1: Obdélníková metoda pro vztah 4.2

Další metodou je metoda obdélníku s trojúhelníkem, kde se použije menší z obdélníků vzniklých dvěma po sobě jdoucími vzorky podobně jako v metodě obdelníkové a k jeho obsahu se přičte obsah pravoúhlého trojúhelníku, který vznikl stejnými vzorky a intervalem mezi nimi. Tato metoda je přesnější než použití pouze obdélníku, ale rozdíl je zanedbatelný, pokud je vzorkování signálu dostatečně granulární. V případě, že se frekvence blíží maximální měřitelné frekvenci daného AD převodníku, je ale odchylka nezanedbatelná a proto je tato metoda vhodnější. Při samotném výpočtu není potřeba počítat obsah trojúhelníku, stačí vypočítat obsah obdélníků vytvořenými dvěma po sobě jdoucími vzorky a vydělit jej dvěma, nebo vypočítat obsah menšího z obdélníků, a přičíst polovinu obsahu obdélníku vzniklého rozdílem dvou sousedních vzorků.

Další metoda výpočtu aproximace obsahu funkce ze vzorků je statistická aproximace nazývaná root–mean–square[10]. Název této metody vychází z její definice:

$$x_{rms} = \sqrt{\frac{\sum_{k=1}^{n} x_k^2}{n}} \tag{4.3}$$



Obrázek 4.2: Metoda obdélníku s trojúhelníkem

Tato metoda je matematicky identická s výše uvedenou metodou aproximace výpočtu plochy prostřednictvím vzorkovacích obdélníků. Nevychází sice striktně z definice hodnoty efektivního napětí, ale je matematicky naprosto neutrální, protože metoda RMS sečte kvadráty naměřeného napětí, vydělí tento součet bezjednotkovou hodnotou a odmocní. Fyzikální veličina napětí, o jejíž výpočet se jedná, tedy zůstane v celém průběhu výpočtu jako jediná veličina s rozměrem.

Existují samozřejmě další metody pro výpočet efektivního napětí, například pokud předem víme, že signál je tvaru sinusoidy, můžeme jeho efektivní napětí spočítat pomocí vzorce

$$V_{ef} = \frac{V_{peak}}{\sqrt{2}} = \frac{V_{p\text{-}p}}{2 * \sqrt{2}} \tag{4.4}$$

Podobně jako pro sinusové signály existují předem zjednodušené výpočty jiných běžných druhů střídavých signálů, jako je například tvar signálu trojúhelník, nebo pila. Ale vzhledem k tomu, že nejsme schopni předem zjistit tvar signálu, je potřeba výpočet provádět metodou, která bude počítat efektivní napětí i u šumu, přestože to nedává smysl.

Kapitola 5

Výpočet frekvence

Výpočet frekvence bude prováděn pomocí diskrétní Fourierovy transformace[11][12], která umožňuje rozklad posloupnosti reprezentující vzorkovaný signál na jednotlivé frekvence, ze kterých se tento signál skládá.

5.0.1 Definice diskrétní Fourierovy transformace

Fourierova diskrétní transformace převede posloupnost N komplexních čísel $\{x_n\} := x_1, x_2, \ldots, x_{N-1}$ na jinou posloupnost komplexních čísel $\{X_k\} := X_1, X_2, \ldots, X_{N-1}$, která je definována následovně[12]

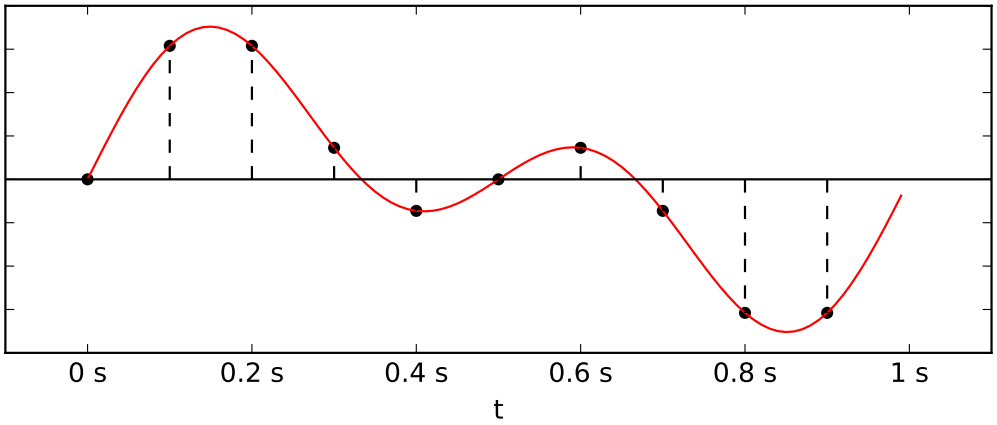
$$X_k = \sum_{n=0}^{N-1} x_n \cdot e^{-\frac{2\pi i}{N} kn} = \sum_{n=0}^{N-1} x_n \cdot \left(cos\left(\frac{2\pi k\ n}{N}\right) - i \cdot sin\left(\frac{2\pi k\ n}{N}\right)\right) \quad (5.1)$$

5.0.2 Popis postupu výpočtu

Vzhledem k tomu, že postup určování frekvence pomocí Fourierovy transformace je poměrně obtížné vysvětlit slovy, rozhodl jsem se celý postup projít na příkladu. V tomto příkladě budeme mít signál složený ze dvou sinusoid (harmonický signál), jedna o frekvenci 1 Hz a druhá o frekvenci 2 Hz, vzorkovací frekvence bude 10 Hz a délka měření bude 1s. Na obrázku 5.0.2 je znázorněn signál s naměřenými vzorky, posloupnost naměřených hodnot je tedy

$$\begin{aligned}\{x_k\} = \{&0i; 1,539i; 1,539i; 0,363i; -0,363i;\\ &0i, 0,363i; -0,363i; -1,539i; -1,539i\}\end{aligned} \quad (5.2)$$

Následně se pomocí diskrétní Fourierovy transformace provede výpočet transformované řady dle definice 5.1. Vzhledem k tomu, že se jedná o dosa-



Obrázek 5.1: Vzorkování signálu

zení do vzorce, nebudu ukazovat jednotlivé výpočty ale pouze transformovanou řadu.

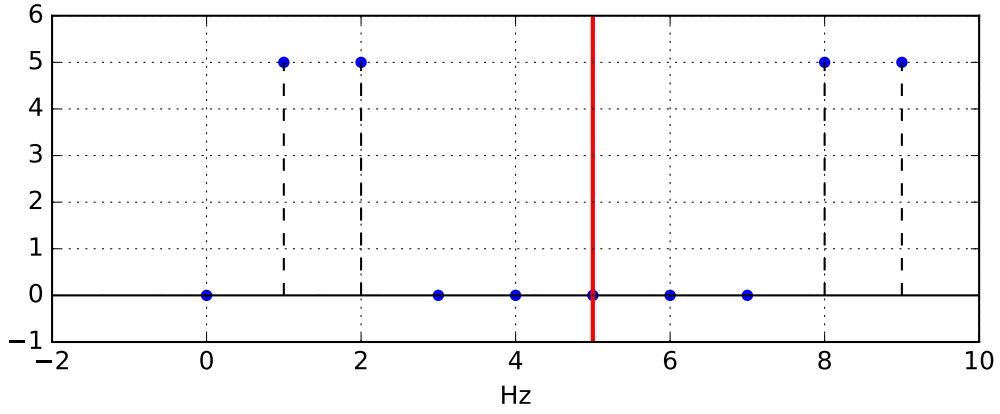
$$\{X_k\} = \{0+0i; 0-5i; 0-5i; 0+0i; 0+0i; 0+0i; 0+0i; 0+5i; 0+5i\} \tag{5.3}$$

Nyní vypočteme absolutní hodnotu členů této posloupnosti a získáme posloupnost

$$\{M_k\} = \{0; 5; 5; 0; 0; 0; 0; 0; 5; 5\} \tag{5.4}$$

Právě tato posloupnost již téměř vyjadřuje frekvence obsažené v daném signálu, kde index prvku v posloupnosti vyjadřuje frekvenci. Téměř ale znamená, že ještě není hotovo. Na obrázku 5.2 je znázorněna posloupnost $\{M_k\}$, a je z něj ihned zjevné, že hodnoty jsou symetrické podle svislé červené čáry. Tato svislá čára reprezentuje tzv. Nyquistův limit [8] vycházející z jeho teorému, který říká, že vzorkovací frekvence musí být alespoň dvojnásobkem nejvyšší frekvence obsažené ve vzorkovaném signálu. Tento limit je tedy roven polovině vzorkovací frekvence, v tomto případě 5 Hz. Symetrie signálů dle tohoto limitu se nazývá folding[13] a pro účely výpočtu má za následek, že hodnoty za ním vyřadíme, zatímco hodnoty před ním vynásobíme dvěma („překlopíme druhou polovinu na první a sečteme“). Ozbrojeni touto znalostí jsme schopni urychlit implementaci tím, že nebudeme počítat Fourierovu transformaci pro celou posloupnost N vzorků, ale pouze pro první polovinu vzorků a výsledek zdvojnásobíme. Tím ušetříme polovinu výpočtů a zdvojnásobíme tak rychlost.

Obrázek 5.2: Posloupnost $\{M_k\}$

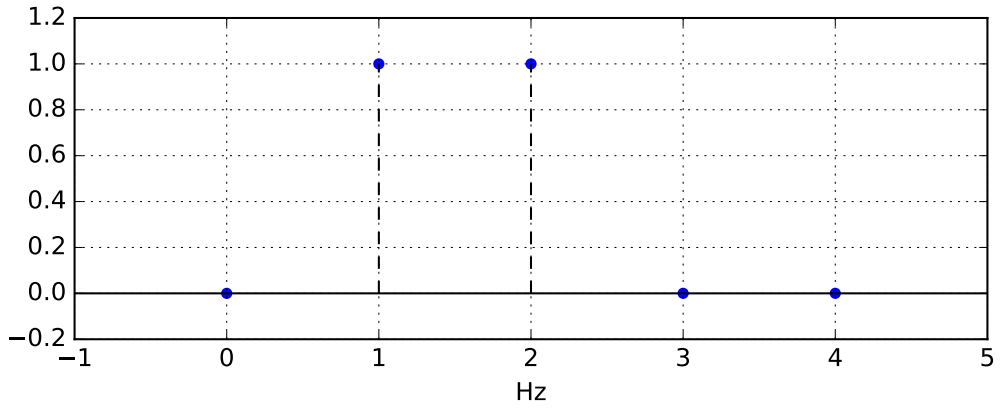


Posledním krokem je vydělení každé hodnoty posloupnosti počtem vzorků, tím získáme posloupnost hodnot v intervalu $< 0, 1 >$, výsledkem je tedy posloupnost

$$\{M_k\} = \{0; 1; 1; 0; 0\} \tag{5.5}$$

znázorněna na obrázku 5.3. Z posloupnosti je zjevné, že vstupní signál je složený ze dvou signálů o frekvenci 1 Hz, respektive 2 Hz, což odpovídá signálu ze zadání.

Obrázek 5.3: Posloupnost $\{M_k\}$ po přeložení



Pokud vynecháme zkrácení výpočtu znalostí symetrie výsledné posloupnosti, je složitost Fourierovy transformace $O(N^2)$ pro N vzorků. V případě,

že vzorkovací frekvence je 44,1 kHz, algoritmus provede 19,36 milionu operací, pokud jedna operace zabere 1 ms, což je dle mého názoru rozumný odhad vzhledem k tomu, že 1 operace je v tomto kontextu vynásobení a následné sečtení dvou komplexních čísel, bude výpočet jedné řady trvat necelých 20 s. To je dvacetinásobek délky tohoto signálu a je tedy zjevné, že tento algoritmus je sice použitelný, ale není optimální.

5.0.3 Rychlá Fourierova transformace

Jedno z existujících řešení je tzv. rychlá Fourierova transformace[11] (dále FFT) která má složitost $O(N \cdot \log_2 N)$ pro N vzorků. Pokud parametry zůstanou stejné jako v předchozím odhadu, FFT provede přibližně 534 tisíc operací, počet výrazně nižší než u standardní Fourierovy transformace. Výpočet výsledné posloupnosti by tedy trval cca půl sekundy, což výrazné zlepšení především proto, že jsme nyní schopni vypočítat rozklad na jednotlivé frekvence v čase kratším, než je trvání signálu, samozřejmě za předpokladu, že operace trvá 1 ms.

Rychlá Fourierova transformace je implementována několika algoritmy, jedním z nich je tzv. Cooley-Tukey [14] algoritmus, který dělí řady na poloviny a tím dosáhne zmiňované složitosti $O(N \cdot \log_2 N)$ pokud $N = 2^k, k \in \mathbb{N}$.

$$X_k = \sum_{m=0}^{N/2-1} x_{2m} \cdot e^{-\frac{2\pi ikm}{N/2}} + \sum_{m=0}^{N/2-1} x_{2m+1} \cdot e^{-\frac{2\pi ik\ (m+\frac{1}{2})}{N/2}} \tag{5.6}$$

Ve druhé sumě si lze všimnout, že v exponentu vznikla konstanta, kterou lze vytknout:

$$\begin{aligned} \sum_{m=0}^{N/2-1} x_{2m+1} \cdot e^{-\frac{2\pi ik(m+\frac{1}{2})}{N/2}} &= \sum_{m=0}^{N/2-1} x_{2m+1} \cdot e^{-\frac{2\pi ikm}{N/2} - \frac{\pi ik}{N/2}} \\ = \sum_{m=0}^{N/2-1} x_{2m+1} \cdot e^{-\frac{2\pi ikm}{N/2}} \cdot e^{-\frac{\pi ik}{N/2}} &= e^{-\frac{2\pi ik}{N}} \cdot \sum_{m=0}^{N/2-1} x_{2m+1} \cdot e^{-\frac{2\pi ikm}{N/2}} \end{aligned} \tag{5.7}$$

Po vytknutí a dosazení do původní rovnice získáme:

$$X_k = \sum_{m=0}^{N/2-1} x_{2m} \cdot e^{-\frac{2\pi ikm}{N/2}} + e^{-\frac{2\pi ik}{N}} \cdot \sum_{m=0}^{N/2-1} x_{2m+1} \cdot e^{-\frac{2\pi ikm}{N/2}} \tag{5.8}$$

Dalším pozorováním zjistíme, že exponenty e se nyní v obou sumách shodují, tento exponent převedeme pomocí Eulerovy formule [15] na goniometrický tvar:

$$e^{-\frac{2\pi ik\ m}{N/2}} = \cos\left(-\frac{2\pi ikm}{N/2}\right) + i \cdot \sin\left(-\frac{2\pi ikm}{N/2}\right) \tag{5.9}$$

Vzhledem k tomu, že sin i cos jsou periodické, jsme schopni určit jejich periodu a následně říci, že $e^{-\frac{2\pi ikm}{N/2}}$ je také periodické.

$$\cos\Big(-\frac{2\pi ikm}{N/2}\Big) = \cos\Big(-\frac{2\pi i(k+N/2)m}{N/2}\Big) \tag{5.10}$$

$$\sin\Big(-\frac{2\pi ikm}{N/2}\Big) = \sin\Big(-\frac{2\pi i(k+N/2)m}{N/2}\Big) \tag{5.11}$$

$$e^{-\frac{2\pi ikm}{N/2}} = e^{-\frac{2\pi i(k+N/2)m}{N/2}} \tag{5.12}$$

$$X_{k+\frac{N}{2}} = \sum_{m=0}^{N/2-1} x_{2m} \cdot e^{-\frac{2\pi im}{N/2}(k+\frac{N}{2})} + e^{-\frac{2\pi i}{N}(k+\frac{N}{2})} \cdot \sum_{m=0}^{N/2-1} x_{2m+1} \cdot e^{-\frac{2\pi im}{N/2}(k+\frac{N}{2})} \tag{5.13}$$

$$= \sum_{m=0}^{N/2-1} x_{2m} \cdot e^{-\frac{2\pi imk}{N/2}} e^{-2\pi im} + e^{-\frac{2\pi ik}{N}-\pi i} \cdot \sum_{m=0}^{N/2-1} x_{2m+1} \cdot e^{-\frac{2\pi imk}{N/2}} e^{-2\pi im} \tag{5.14}$$

$$= \sum_{m=0}^{N/2-1} x_{2m} \cdot e^{-\frac{2\pi imk}{N/2}} - e^{-\frac{2\pi ik}{N}} \cdot \sum_{m=0}^{N/2-1} x_{2m+1} \cdot e^{-\frac{2\pi imk}{N/2}} \tag{5.15}$$

Poslední úprava mezi rovnicemi byla provedena za pomoci znalosti Eulerovy identity [15].

Nyní si lze všimnout, že výrazy 5.8 a 5.15 jsou si velmi podobné, jediný rozdíl je, že ve výrazu 5.8 se sumy sčítají, zatímco ve výrazu 5.15 se odečítají. Následkem je, že pokud máme $N = 2^k$ vzorků, jsme schopni každou ze sum rozdělit na dvě stejně dlouhé sumy a každou z těchto částečných sum vypočítat pouze jednou, opět za pomoci rozdělení na poloviční sumy. Až dojde na situaci kde $N = 1$, tvořící ukončovací podmínku rekurze. Tímto postupem lze tedy dosáhnout složitosti $O(N \cdot \log N)$.

Kapitola 6

Popis implementace

V této sekci budu diskutovat postupy a technologie, které byly použité při vývoji této aplikace. Především se zaměřím na diskusi použitých některých tříd z knihovny Qt, a implementaci výše popsaných metod výpočtu efektivního napětí a frekvence.

6.1 Požadavky

Ze zadání této bakalářské práce přímo vycházejí tři požadavky: vykreslování zkoumaného signálu, výpočet efektivního napětí a výpočet frekvence.

6.2 Volba technologie

Z prvu byl pro implementaci, především z důvodu snadnosti použití, zvolen jazyk Python s knihovnou TkInter pro grafické rozhraní v kombinaci s knihovnou matplotlib pro vykreslování zkoumaného signálu. Po implementaci první beta verze pouze s funkčním zobrazováním zkoumaného signálu se projevilo, že jsem se mýlil. Již od prvního spuštění bylo evidentní, že záznam hodnot a jejich vykreslování trvá mnohem déle, něž zobrazovaný interval. Tato skutečnost v kombinaci s tím, že se signál zobrazoval sekvenčně mělo za následek to, že aplikace zobrazovala signál nejprve o několik desetin sekundy opožděně, ale čím déle běžela, tím se více vzdalovala od aktuálního stavu. Následně tedy aplikace zobrazovala signál o několik sekund později, než byl na vstupu, což mělo za následek absolutní nepoužitelnost aplikace. Z tohoto důvodu bylo evidentní, že je potřeba zvolit jinou technologii pro implementaci aplikace s ohledem na zachování multiplatformnosti aplikace. Samozřejmě byl tento výběr omezen na technologie, se kterými jsem měl

alespoň minimální předchozí zkušenosti, tedy Java, C# a C++ s Qt nebo SFML knihovnou. Vzhledem k tomu, že problém s implementací v Pythonu byla rychlost, jsem se rozhodl pro C++ s Qt[16], neboť jsem si myslel, že tato možnost má nejvyšší pravděpodobnost úspěchu řešení problému s rychlostí generování grafu. Po základní reimplementaci vykreslování signálu v Qt se ukázalo, že můj předpoklad byl správný a rychlost vykreslování je nyní dostatečná. Je nezanedbatelné, že kvůli komplexnosti jazyka C++ je implementace značně složitější oproti implementaci v Pythonu, ale naštěstí obsahuje knihovna Qt velké množství nástrojů vhodných k použití mimo jiné i v rámci této práce.

Velká část tvorby této aplikace spočívá v analýze a vyběru metod pro výpočty, zatímco návrh aplikace je poměrně přímočarý. Vzhledem k použití knihovny Qt verze 5.10 budu diskutovat i některé prvky této knihovny které používám.

6.3 Vstup signálu

Jak již bylo zmíněno výše, zvuková karta signál navzorkuje, a tedy jakýkoliv program bude již pracovat s posloupností vzorků. Bylo tedy potřeba zjistit, jak za použití Qt knihovny získat tyto data, první částí tohoto problému je začít naslouchat na zvukové kartě. K tomu má knihovna Qt připravenou třídu *QAudioInput*. K tomu, aby tato třída začala signál nahrávat potřebuje informace o zařízení na kterém má naslouchat, a také formát ve kterém má tento signál zaznamenávat 6.1. Při spuštění programu používám zařízení které operační systém označuje jako implicitní vstupní audio zařízení. Při běhu aplikace lze vstupní zařízení zvolit ze seznamu, tento seznam je tvořen při spuštění aplikace a z tohoto důvodu není možné zvolit zařízení připojené po jejím spuštění. Seznam vstupních zařízení je získáván z operačního systému, je tedy možné, že v některých případech se v něm vyskytnou zařízení které vstup neumožňují (například z důvodu, že je operační systém špatně označil). Toto chování bylo pozorováno pouze na platformě Linux a nepodařilo se mi najít řešení.

Listing 6.1: Konstruktor třídy QAudioInput

```
QAudioInput(
        const QAudioDeviceInfo &audioDevice,
        const QAudioFormat &format = QAudioFormat(),
        QObject *parent = Q_NULLPTR
);
```

Objekt typu *QAudioFormat* obsahuje parametry nahrávání, pro účely

této aplikace jsou důležité tři: vzorkovací frekvence, bitová hloubka, a počet kanálů. Vzorkovací frekvenci vždy nastavuji na nejvyšší frekvenci podporovanou daným zařízením za účelem dosažení nejvyšší možné přesnosti zobrazení. Bitová hloubka je bohužel pevně nastavena na 8 bitů na vzorek, toto omezení vyvstalo z technických problémů s nahráváním větších vzorků. Ze stejných technických problémů je počet kanálů omezen na jeden.

QAudioInput zapisuje data, která přečte ze vstupu do objektu typu *QIODevice*. Za tímto účelem jsem implementoval třídu *DataReader* která rozšiřuje třídu *QIODevice* a přetěžuje metodu *writeData*. Právě v tomto místě se vyskytl problém s bitovou hloubkou a počtem kanálů. Pokud jsem nastavil objektu *QAudioFormat* bitovou hloubku jinou než 8 bitů, nebo počet kanálů jiný než 1, přestala se tato metoda volat. Vzhledem k tomu, že jsem neznal důvod tohoto chování jsem se rozhodl přetížit ostatní zapisovací metody třídy *QIODevice*, bohužel se mi problém vyřešit nepodařilo.

Metoda *writeData* obdrží pole jednobytových hodnot které jsou v rozsahu 0 – 255, proto je od nich ještě před uložením odečteno 128. Hodnoty jsou v *DataReader* uloženy dvakrát, jednou pro graf zobrazení signálu, kde je uloženo tolik hodnot, kolik odpovídá zobrazovanému intervalu s ohledem na vzorkovací frekvenci. Například pokud je vzorkovací frekvence 48 kHz, a zobrazovaný interval je 100 ms, bude ukládáno 4800 hodnot. Druhý vektor hodnot je ukládán pro potřeby Fourierovy transformace a proto jeho délka odpovídá použité vzorkovací frekvenci.

6.4 Zobrazování signálu

Pro zobrazování signálu byla zvolena třída *QChart*. Tato třída umožňuje zobrazování různých druhů grafů, pro účely osciloskopu jsem zvolil graf čárový. Zobrazení grafu se dosáhne přiřazením posloupnosti bodů reprezentovanou třídou *QLineSeries*, body jsou v posloupnosti reprezentovány třídou *QPointF*, která body reprezentuje jako dvojici souřadnic v desetinném formátu. Stejný postup byl aplikován na zobrazování frekvenčního spektra.

Vzhledem k tomu, že body jsou ukládány jako jednobytové hodnoty, je potřeba je převést na desetinné číslo. To je prováděno dle vzorce 6.1, kde p je hodnota vzorku získaného ze vstupu, m je násobič a y je výsledná svislá souřadnice daného bodu.

$$y = p/128 * 1000 * m \tag{6.1}$$

Vzorec byl takto zvolen aby hodnoty byly nejprve normalizovány do intervalu $< 0, 1 >$, vynásobení tisícem je zamýšleno jako převod z voltů

Listing 6.2: Ukázka deklarace signálu

```
class MyClass {
        signals:
                void update(char * data);
}
```

na milivolty a nakonec vynásobení kalibračním koeficientem. Jak již bylo zmíněno v sekci o rozsahu měřitelného napětí 3.2, tento koeficient bude muset uživatel určit po každou zvukovou kartu indiviudálně za pomoci referenčního napětí.

6.5 Grafické rozhraní

V Qt je každý prvek grafického rozhraní reprezentován objektem typu *QWidget*, a to včetně okna samotného. Proto jsem vytvořil třídu *ScopeWindow*, která reprezentuje právě okno aplikace a tak obsahuje všechny ovládací prvky. Pro zobrazování grafů byla použita výše zmíněná třída *QChart*, textové vstupy jsou zprostředkovány třídou *QLineEdit* v kombinaci s třidou *QPushButton* pro tlačítka, výstupy jsou provedeny pomocí třídy *QLabel*. S vyjímkou *QChart* jsou všechny výše zmíněné třídy potomky *QWidget*, *QChart* je proto obalen třídou *QChartView* která je potomkem *QWidget* a tedy umožňuje zobrazení tohoto grafu v grafickém rozhraní. Na obrázcích A a A je znázorněno provední aplikace.

Komunikace mezi prvky grafického rozhraní, například změna časové osy v grafu, je provedena asynchronně. Za tímto účelem obsahuje knihovna Qt tzv. signály a sloty [17] jako náhradu callback funkcí (nebo metod). Použití signálů a slotů je velmi jednoduché, v předpisu třídy se vytvoří kategorie „signals“, respektive „slots“ u kterého se ještě nastaví zda–li je public či private, a do této kategorie se zapíší předpisy metod pracujících jako signály, respektive sloty, jak je znázorněno v 6.2 Klasifikace public či private u slotů má stejný dopad jako u tradičních metod, tedy zda je lze vyvolat z kontextu mimo objekt. Signály jsou vždy klasifikace public, umožňující vysílání signálu i z prostředí mimo kontext třídy, tím lze dosáhnout například zřetězení signálů. Stejně jako ostatní metody mohou být signály i sloty vytvořeny s parametry, umožňující předávání informací. Pro použití je potřeba ještě propojit signál se slotem, jak je ukázáno v 6.3.

Listing 6.3: Ukázka propojení signálu a slotu

```
connect(
        button,
        SIGNAL(released()),
        this,
        SLOT(changeTimescale())
);
```

6.6 Výpočet efektivního napětí

Výpočet efektivního napětí byl již diskutován v své vlastní kapitole, tak jej pouze shrnu. Zvolená metoda se nazývá root–mean–square a je definována vztahem 4.3. Implementace v programu je tedy také velmi jednoduchá jak je vidět na 6.4

Listing 6.4: Implementace metody RMS

```
qreal DataReader::calculateRMSVoltage() {
    QVector<QPointF> points = m_scopeSeries->pointsVector();
    qreal sum_of_squares = 0.0;

    foreach (QPointF point, points)
        sum_of_squares += point.y() * point.y();

    return sqrt(sum_of_squares/points.length());
}
```

6.7 Výpočet Fourierovy transformace

Tato práce byla pro mě prvním setkáním s Fourierovou transformací, proto jsem se rozhodl k ní nejprve přistupovat opatrně. Z tohoto důvodu jsem se rozhodl nejprve implementovat její základní verzi dle definice 5.1 6.5.

Hned po prvním spuštění bylo evidentní, že tento výpočet trvá velmi dlouho, a proto jsem se rozhodl jej přesunout do vlastního vlákna aby neblokoval zbytek aplikace. Zvuková karta v mém laptopu podporuje vzorkovací frekvenci 48 kHz a výpočet Fourierovy transformace nad 48 tisíci vzorky zabral přes 30 sekund. Takové trvání výpočtu bylo nepraktické a začal jsem tedy hledat řešení. První nalezenou možností byl Cooley–Tukey [14] algoritmus rychlé Fourierovy transformace ale tento algoritmus funguje pouze pro počty vzorků rovné mocninám dvou. Žádná ze standardních vzorkovacích frekvencí tomuto požadavku neodpovídá, rozhodl jsem se proto nejprve provést malý experiment.

Listing 6.5: Implementace základní Fourierovy transformace

```
QVector<QPointF> frequencies;
int sampleCount = m_samples.length();
for(int i = 0; i < m_sampleRate/2; i++) {
        complex<qreal> bin;
        for(int j = 0; j < sampleCount; j++) {
                std::complex<qreal> mult;
                qreal arg = (2 * PI * i * j)/m_sampleRate;
                mult.real(cos(-arg));
                mult.imag(sin(-arg));

                bin += complex<qreal>(0, m_samples[j]) * mult;
        }
        qreal y = (abs(bin) * 2)/m_sampleRate;
        frequencies.append(QPointF(i, y));
}
```

Tento experiment spočíval v provedení Fourierovy transformace pouze nad poslední desetinou naměřených vzorků. Empirické pozorování ukázalo, že výsledky takto provedené Fourierovy transformace odpovídají jedné desetině skutečné frekvence signálu příčemž výpočet nyní trval pouze 5 s. Toto mělo za následek, že program byl nyní schopen určit frekvenci mnohem rychleji, ale se sníženou přesností na desítky Hertzů místo jednotek.

Následně jsem se pokusil o implementaci rychlé Fourierovy transformace 5.0.3 pomocí Cooley–Tukey algoritmu. Jak již bylo řečeno, tento algoritmus vyžaduje, aby počet vzorků byl roven mocninám dvou. Toto jsem se rozhodl vyřešit nastavením pole naměřených vzorků nulami.

Rozhodl jsem se nejprve implementovat rychlou Fourierovu transformaci rekurzivní metodou 6.6 dle definice 5.0.3 a v případě, že by ani toto nebylo dostatečně rychlé bych přešel na iterativní in–place metodu. Naštěstí se ukázalo, že rekurzivní metoda je dle mého názoru dostatečně rychlá pro účely této aplikace a tak jsem se rozhodl u ní zůstat. Dokonce vyšlo najevo, že se Fourierova transformace provede rychleji, než se vykreslí graf frekvenčního spektra. Implementace vlastního vlákna pro výpočet transformace tedy ztratil smysl, ale rozhodl jsem se tuto část implementace zanechat pro případ, že uživatel použije zvukovou kartu s vysokou vzorkovací frekvencí (192 kHz nebo i více).

Vzhledem k použití čísel s plovoucí desetinnou čárkou v kombinaci s velkým počtem násobení mezi nimy dojde k amplifikaci nepřesnosti vzniklé použitím takto zakódovaných hodnot. Proto může mít měření frekvence malou odchylku od skutečné hodnoty.

Frekvence dominantního harmonického signálu obsaženého ve vstupním

Listing 6.6: Implementace rekurzivní Fourierovy transformace

```
void FrequencyCalculator::FFT(complex<qreal> * data, qint64 len)
{
    if(len <= 1) return;

    qint64 half_len = len/2;
    complex<qreal> * even_series = new complex<qreal>[half_len];
    complex<qreal> * odd_series = new complex<qreal>[half_len];
    for(int i = 0; i < half_len; i++) {
        even_series[i] = data[2 * i];
        odd_series[i] = data[2*i + 1];
    }

    FFT(even_series, half_len);
    FFT(odd_series, half_len);

    for (int i = 0; i < half_len; i++) {
        complex<qreal> mult = polar(1.0, - 2.0 * PI * i / len);
        data[i] = even_series[i] + mult * odd_series[i];
        data[i + half_len] = even_series[i] - mult * odd_series[i];
    }

    delete even_series;
    delete odd_series;
}
```

signálu je určena nejvyšší vypočtenou hodnotou na výstupu Fourierovy transformace.

6.8 Instalace

Pro platformu Windows je připraven binární soubor doprovázený potřebnými `dll` soubory, tento soubor stačí pouze spustit.

Pro platformu Linux je potřeba stáhnout Qt knihovnu a aplikaci zkompilovat. Kompilace se provede spuštěním programu `make`, kterému je potřeba nastavit parametr `QMAKE`. Tento parametr by měl obsahovat cestu k programu `qmake`, který se nainstaloval spolu s knihovnou Qt.

Spuštění aplikace se provede spuštěním programu `make run`, nebo `./Oscilloscope`. Alternativně je možno aplikaci na všech platformách zkompilovat pomocí nástroje *Qt Creator*, který lze získat při instalaci knihovny Qt.

Závěr

Účelem této práce bylo vytvořit aplikaci využívající zvukovou kartu jako osciloskop. Tato aplikace měla dle zadání být schopna zobrazovat graf průběhu signálu, provádět výpočet efektivního napětí a měření frekvence. Myslím, že i přes jisté nedostatky vytvořená aplikace požadavky stanovené zadáním splňuje.

Budoucí vývoj aplikace by se dle mého názoru měl vydat především směrem odstranění nedostatků vzniklých implementací. Prvním, a dle mého názoru důležitějším, nedostatkem je schopnost snímání pouze jednoho kanálu, druhým takovým nedostatkem je bitová hloubka snímaných vzorků. Vhodnou součástí dalšího vývoje by bylo také rozšíření funkcionality programu například o posun zobrazovaného signálu ve směru os grafu, možnost uložit naměřená data nebo synchronizace zobrazovaného signálu (tzv. triggery). Uživatelé by jistě také ocenily možnost instalace aplikace pomocí instalátoru.

Pokud omezení a nedostatky zmiňované v průběhu této práce nejsou překážkou zamýšlenému použití, považuji vytvořený program za použitelný k hobby aplikacím.

Bibliografie

1. MARTIN, Dennis J. *SOUND TECHNOLOGY REVEALS AUDIO SPECTRUM* [online]. 1996 [cit. 29. 04. 2018]. Dostupné z: `http://www.soundtechnology.com/rta_dm.html`.
2. ZEITNITZ, Christian. *Soundcard Oscilloscope* [online]. 2017 [cit. 11. 05. 2018]. Dostupné z: `https://www.zeitnitz.eu/scope_en`.
3. ASKARI, Hisham A. H. AL-KHAZALI; Mohamad R. Geometrical and Graphical Representations Analysis of Lissajous Figures in Rotor Dynamic System [online]. 2012 [cit. 19. 03. 2018]. Dostupné z: `http://www.iosrjen.org/Papers/vol2_issue5/G025971978.pdf`.
4. MANNINI, Simone. *OsxillometerXZ* [online] [cit. 02. 05. 2018]. Dostupné z: `http://www.iw5edi.com/software/oscillometerxz`.
5. *Xoscope* [online]. 2016 [cit. 15. 04. 2018]. Dostupné z: `http://xoscope.sourceforge.net/`.
6. COLLINSON, Andy. *Winscope* [online] [cit. 10. 05. 2018]. Dostupné z: `http://www.zen22142.zen.co.uk/Prac/winscope.htm`.
7. CHOUDHARY, Rizwan. *Frequency Range of Human Hearing* [online] [cit. 2018]. Dostupné z: `https://hypertextbook.com/facts/2003/ChrisDAmbrose.shtml`.
8. ROUSE, Margaret. *Nyquist Theorem* [online] [cit. 10. 05. 2018]. Dostupné z: `https://whatis.techtarget.com/definition/Nyquist-Theorem`.
9. *Mic Level and Line Level – What do they mean?* [online]. Shure Incorporated, 2017 [cit. 14. 05. 2018]. Dostupné z: `http://www.shure.com/americas/support/find-an-answer/mic-level-and-line-level-what-do-they-mean`.

10. KENNETH V. CARTWRIGHT, Ph.D. Determining the effective or RMS voltage of various waveforms without calculus. 2007. Dostupné také z: http://tiij.org/issues/issues/fall2007/30_Cartwright/Cartwright-Waveforms.pdf.

11. STEVEN W. SMITH, Ph.D. *The Scientist and Engineer's Guide to Digital Signal Processing* [online]. 1997 [cit. 15. 04. 2018]. ISBN 978-0966017632. Dostupné z: http://www.dspguide.com/pdfbook.htm.

12. WEISSTEIN, Eric W. *Discrete Fourier Transform* [online] [cit. 14. 05. 2018]. Dostupné z: http://mathworld.wolfram.com/DiscreteFourierTransform.html.

13. *Folding frequency* [online]. 2012 [cit. 14. 05. 2018]. Dostupné z: http://glossary.ametsoc.org/wiki/Folding_frequency.

14. TUKEY, James W. Cooley; John W. An Algorithm for the Machine Calculation of Complex Fourier Series [online]. 1965 [cit. 13. 05. 2018]. Dostupné z: http://www.ams.org/journals/mcom/1965-19-090/S0025-5718-1965-0178586-1/home.html#References.

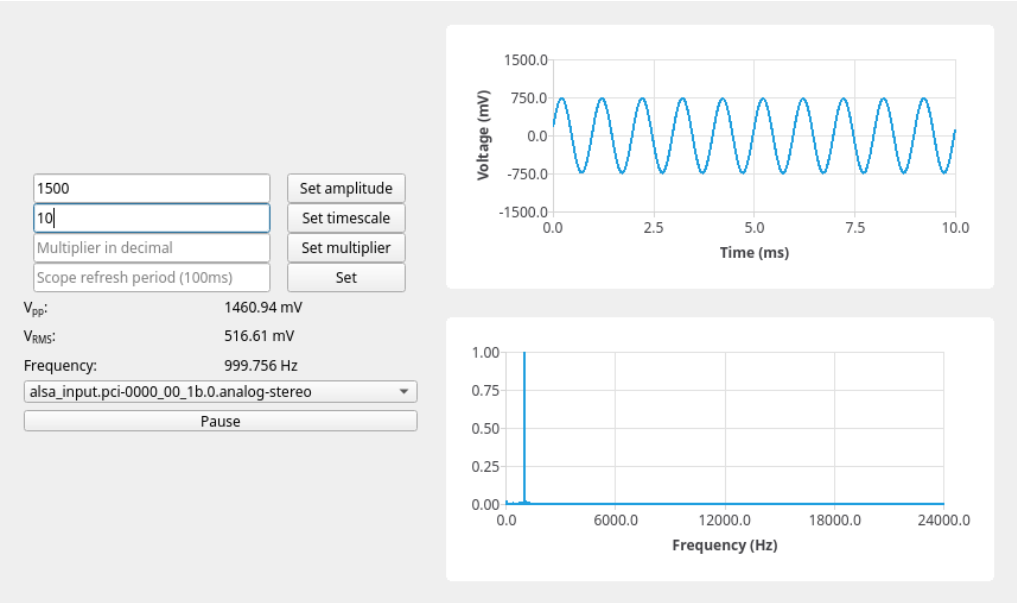
15. DUNHAM, W. *Euler: The Master of Us All*. Mathematical Association of America, 1999. Dolciani Mathematical Expositions, č. v. 22. ISBN 9780883853283. Dostupné také z: https://books.google.cz/books?id=uKOVNvGOkhQC.

16. THE QT COMPANY. *Qt*. 2018. Dostupné také z: https://www.qt.io/.

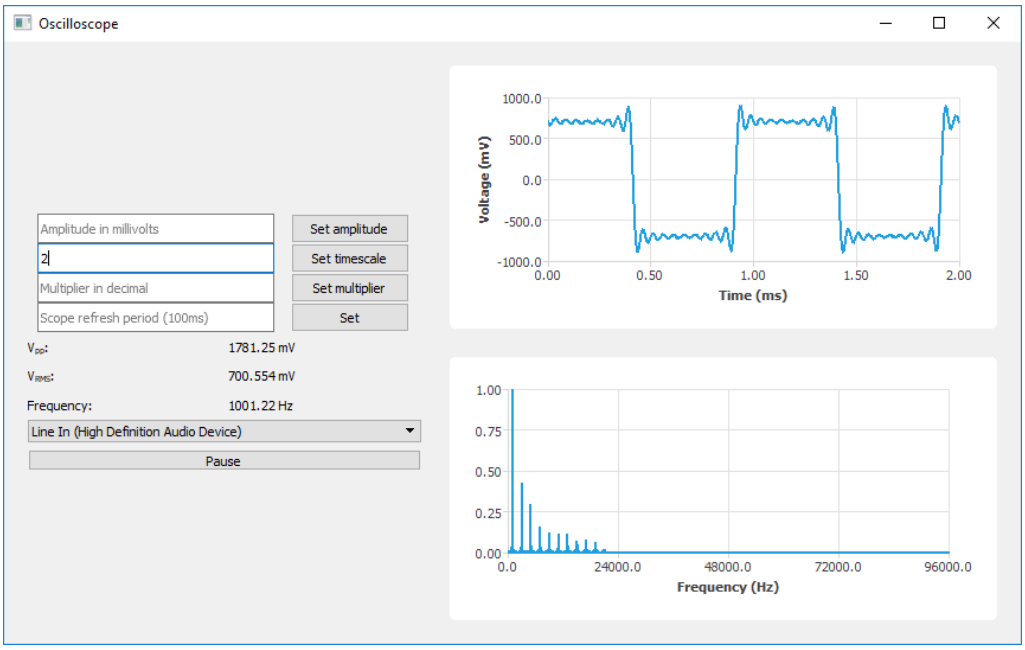
17. *Signals and Slots* [online] [cit. 14. 06. 2018]. Dostupné z: http://doc.qt.io/archives/qt-4.8/signalsandslots.html.

Příloha A

Obrázky



Obrázek A.1: Ukázka aplikace snímající sinusový signál



Obrázek A.2: Ukázka aplikace snímající zkreslený obdélníkový signál

Příloha B

Seznam použitých zkratek

RMS Root mean square

FFT Fast Fourier transform

AD Analog to Digital

Příloha C

Obsah přiložené SD karty

- `readme.txt` ..... stručný popis obsahu CD
- `exe` ..... adresář se spustitelnou formou implementace
- `src`
  - `impl` ..... zdrojové kódy implementace
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `BP-REIMER-MAREK-2018.pdf` ..... text práce ve formátu PDF