TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

IMPLEMENTACE ALGORITMU PRO REDUKCI ŠUMU NA DSP

Bakalářská práce

Studijní program: B2612 – Elektrotechnika a informatika
Studijní obor: 2612R011 – Elektronické informační a řídicí systémy

Autor práce: **Michal Roško**
Vedoucí práce: doc. Ing. Zbyněk Koldovský, Ph.D.

Liberec 2014

TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky a mezioborových studií
Akademický rok: 2013/2014

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Michal Roško**
Osobní číslo:	**M11000171**
Studijní program:	**B2612 Elektrotechnika a informatika**
Studijní obor:	**Elektronické informační a řídicí systémy**
Název tématu:	**Implementace algoritmu pro redukci šumu na DSP**
Zadávající katedra:	**Ústav informačních technologií a elektroniky**

Zásady pro vypracování:

1. Nastudujte metodu pro redukci šumu v zařízení se dvěma mikrofony a způsoby programování signálového procesoru TMS320C6x.
2. Implementujte metodu pro redukci šumu používající banku řeč-potlačujících filtrů na DSP.
3. Vytvořte vhodnou banku řeč-potlačujících filtrů pro model mobilního telefonu.
4. Vylepšete metodu výběrem vhodné metody pro odečítání difúzního šumu.
5. Otestujte navržený postup na zarušených záznamech řeči.

Rozsah grafických prací:	**Dle potřeby dokumentace**
Rozsah pracovní zprávy:	**cca 30 stran**
Forma zpracování bakalářské práce:	**tištěná/elektronická**

Seznam odborné literatury:

[1] **B. Porat, "A Course in Digital Signal Processing", John Wiley & Sons, 1997.**

[2] **Z. Koldovský, P. Tichavský, D. Botka, "Noise Reduction in Dual-Microphone Mobile Phones Using A Bank of Pre-Measured Target-Cancellation Filters," Proc. of ICASSP 2013, pp. 679-683, Vancouver, Canada, May 2013.**

[3] **Rulph Chassaing, "Digital Signal Processing and Applications with the TMS320C6713 and TMS320C6416 DSK", Wiley-IEEE Press, 2008.**

Vedoucí bakalářské práce:	**doc. Ing. Zbyněk Koldovský, Ph.D.** Ústav informačních technologií a elektroniky
Konzultant bakalářské práce:	**Ing. Jiří Málek, Ph.D.** Ústav informačních technologií a elektroniky
Datum zadání bakalářské práce:	**12. září 2013**
Termín odevzdání bakalářské práce:	**16. května 2014**

V. Kopecký

prof. Ing. Václav Kopecký, CSc.
děkan

Technická univerzita v Liberci - 14 -

L.S.

prof. Ing. Zdeněk Plíva, Ph.D.
vedoucí ústavu

V Liberci dne 12. září 2013

Prohlášení

Byl(a) jsem seznámen(a) s tím, že na mou bakalářskou práci se plně vztahuje zákon č. 121/ 2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci (TUL) nezasahuje do mých autorských práv užitím mé bakalářské práce pro vnitřní potřebu TUL.

Užiji-li bakalářskou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti TUL; v tomto případě má TUL právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Bakalářskou práci jsem vypracoval(a) samostatně s použitím uvedené literatury a na základě konzultací s vedoucím bakalářské práce a konzultantem.

15.5.2014

Podpis

Poděkování

Mé poděkování patří především vedoucímu práce, panu doc. Ing. Zbyňku Koldovskému, Ph.D., za rady, věcné připomínky a zapůjčení studijních materiálů a potřebného vybavení. Dále bych chtěl poděkovat panu Ing. Miroslavu Holadovi Ph.D. za rady a zapůjčení vybavení. Děkuji taktéž své rodině za materiální i duševní podporu v průběhu celého studia a své přítelkyni za pomoc při testování algoritmu a při finálních úpravách práce. Díky patří také všem dobrovolníkům, kteří s ochotou vyplnili připravený dotazník.

Abstrakt

Tato bakalářská práce se zabývá zpracováním zvukového signálu - konkrétně algoritmem pro redukci šumu využívající dvou mikrofonů a banky předem naměřených TCF. Tento algoritmus je vhodný zejména k odstranění šumu ze zarušeného zvukového signálu řeči.

Pro svou funkci potřebuje nejprve připravit banku filtrů z nahrávky mluvčího v klidném prostředí. Za použití této banky je poté schopen v reálném čase redukovat šum ve zvukovém signálu obsahujícím hlas mluvčího.

V první kapitole jsou uvedena odvození většiny matematických vztahů potřebných pro výpočet banky filtrů a je rozebrána funkce samotného algoritmu pro redukci šumu v reálném čase. V druhé kapitole je popsáno vývojové prostředí Code Composer Studio, jeho ovládání pomocí doplňku *MATLAB link for CCS* pro MATLAB a systém DSP/BIOS.

Obecné informace z první kapitoly jsou použity v kapitole třetí pro vytvoření programu a jeho následnou implementaci do digitálního signálového procesoru TMS320C6416 od firmy Texas Instruments. Sestavený algoritmus je ve čtvrté kapitole aplikován na testovací nahrávku zarušené řeči. Ta je přehrána několika nezávislým respondentům před i po aplikování algoritmu. Z jejich subjektivního hodnocení funkčnosti algoritmu jsou vyvozeny závěry.

Klíčová slova:

zpracování zvukového signálu, redukce šumu, digitální signálový procesor, programování

Abstract

This bachelor thesis is focused on the processing of an audio signal particularly on the noise reduction algorithm using two microphones and the database of the TCF obtained in advance. This algorithm is suitable especially for the noise reduction of the noisy speech audio signal.

For the right function of the algorithm the database of the filters has to be created from the speaker's speech recording in the quiet environment. The algorithm is then able to reduce the noise in the real-time audio signal of the speaker's speech using these filters.

In the first chapter of this thesis, the majority of the mathematical relationships and equations used for the generation of the filters' database are quoted and the function of the algorithm for the noise reduction in the real-time is studied. In the second chapter, the software development environment of the Code Composer Studio is described together with its control by the *MATLAB link for CCS* add-on for MATLAB and the system DSP/BIOS.

In the third chapter, the general information from the chapter one is used to create the algorithm which is then implemented as the code into the digital signal processor Texas Instruments TMS320C6416. In the fourth chapter, the noise reduction algorithm is applied to the test recording of the noisy speech audio signal. The recordings before and after the application of the algorithm are then played to several respondents. The final conclusions are drawn then from their subjective evaluation of the algorithm functionality.

Keywords:

audio signal processing, noise reduction, digital signal processor, programming

Obsah

Seznam obrázků

Seznam zkratek a symbolů

CCStudio	code composer studio
DFT	diskrétní Fourierova transformace
DSP	digitální signálový procesor
DSP/BIOS	operační systém digitálního signálového procesoru
FFT	rychlá Fourierova transformace
GSM	globální systém pro mobilní komunikaci
iFFT	rychlá inverzní Fourierova transformace
SNR	signal to noise ratio
TCF	target cancellation filtr

Úvod

Rychlý vývoj elektroniky - notebooků, mobilních telefonů, tabletů apod. - přináší mimo jiné nové možnosti do oblasti zpracování signálu v těchto zařízeních. Pomineme-li mnohá další vylepšení dnešních přístrojů, narazíme na použití více mikrofonů. Tato skutečnost společně s rostoucím výpočetním výkonem umožňuje různá využití v této oblasti.

Zaměříme-li se na telefonní hovor, zjistíme, že mnohdy je ze sluchátka slyšet pouze okolní hluk a samotný člověk, se kterým se pokoušíme domluvit, se v tomto hluku ztrácí. Zde se ukazuje, že by bylo vhodné využít výše uvedených vlastností současných přístrojů a snažit se hluk odstranit nebo alespoň zredukovat.

Pro tento účel existují vhodné algoritmy. Vzhledem k tomu, že by jejich implementace znamenala zásah do operačního systému daného zařízení (v případě GSM hovoru), nebo přímo do aplikace, jež hovor uskutečňuje (např. Skype), není v našich možnostech se jí zabývat. Existuje však jiný druh zařízení, na kterém lze tento problém řešit a je na akademické půdě mnohem dostupnější. Je jím digitální signálový procesor (DSP). Jeho možnosti jsou velmi podobné jako u klasických zařízení, ale práce s ním je jednodušší. Navíc je jeho instrukční sada přizpůsobena zpracování signálu, a proto se v mnoha případech k těmto účelům používá přímo v praxi. Abychom funkci algoritmu pro redukci šumu pouze nesimulovali na počítači (např. v MATLABu), poslouží nám právě DSP jako fyzické zařízení, do kterého budeme algoritmus implementovat.

Algoritmus, jež budeme v této práci používat, nejprve z nahrávky odstraní řeč mluvčího a tím získá informaci o okolním šumu. Tuto informaci následně využije při jeho redukci. Pro odstranění mluvčího (targetu) z nahrávky se používá tzv. target cancellation filtr (TCF), který funguje dobře jen pro určitou pozici mluvčího vůči mikrofonům. Jelikož se mluvčí však může vůči mikrofonům v určitém rozsahu pohybovat, používá algoritmus banku několika takovýchto TCF. Každý z filtrů v této bance odpovídá jiné pozici mluvčího. Tyto filtry je potřeba pro algoritmus předem vypočítat. Důležité je, že nahrávka, ze které se tyto filtry počítají, musí být bez šumu a dávat tedy informaci pouze o mluvčím.

Cílem práce je pochopit funkci výše popsaného algoritmu, implementovat ho do fyzického zařízení a otestovat jeho funkčnost.

V první kapitole se zaměříme na teorii redukce šumu, tedy definujeme vztahy, které zde platí. V další části popíšeme prostředky použité pro vytvoření programu a jeho následné testování. Dále se pokusíme vztahy popsané v první části využít v praxi a sestavit tak algoritmus, který následně implementujeme do DSP. Poslední

kapitola bude věnována vytvoření testovacích nahrávek a subjektivnímu hodnocení několika nezávislými posluchači.

1 Teorie odstraňování šumu

Každý signál by měl nést informaci a může obsahovat šum, díky kterému je tato informace hůře interpretovatelná. Proto je vhodné snažit se ho v signálu zredukovat. Šumem označujeme součást signálu, která pro nás v danou chvíli není užitečná. Často se stává, že část signálu, která je v jedné situaci šumem, může být v jiné situaci užitečnou informací.

Pro redukci šumu v signálu lze využít metodu, která je založena na:

1. Získání informace o šumu - použití TCF.
2. Využití této informace při odstranění šumu ze signálu - použití Wienerova filtru.

Dále budeme hovořit především o zvukovém signálu. Máme-li zdroj zvuku fixně umístěn v prostoru, pak to, jak ho slyšíme velmi závisí na tom, kde se vůči němu nacházíme. Dále také záleží na prostředí, ve kterém se zvuk šíří a samozřejmě na odrazech od okolních objektů. Všechny tyto faktory ovlivňují frekvenční spektrum daného signálu nebo například tvoří ozvěny. Řekněme tedy, že zvuk ze zdroje přichází do daného místa specificky zabarven.

1.1 Target cancellation filtr

Target cancellation filter (TCF) je takový filtr, jež potlačuje užitečný signál a jeho výstupem je tedy šum. V této části si odvodíme, jak vypočítat koeficienty takového filtru.

1.1.1 Nalezení vhodného TCF

Mějme navzorkovaný signál o délce N vzorků odpovídající jedné pozici mluvčího vůči mikrofonům, složený z levého a pravého kanálu (x_L, x_R). Můžeme si představit, že každý z těchto kanálů se skládá z užitečného signálu s specificky zabarveného dle impulzní odezvy levého (h_L) resp. pravého (h_R) kanálu a z šumu v levém (y_L) resp. pravém (y_R) kanálu viz (1), (2).

$$x_L(n) = \{h_L * s\}(n) + y_L(n) \tag{1}$$

$$x_R(n) = \{h_R * s\}(n) + y_R(n) \tag{2}$$

Jestliže zavedeme užitečný signál zabarvený dle pozice levého mikrofonu s_L a relativní zabarvení dle pozice levého mikrofonu vůči pravému h_{rel} podle vztahů (3) a (4),

$$s_L = \{h_L * s\}(n) \tag{3}$$

$$h_{rel} = h_L * h_R^{-1} \tag{4}$$

pak pro jednotlivé kanály můžeme použít vyjádření (5), (6).

$$x_L(n) = s_L(n) + y_L(n) \tag{5}$$

$$x_R(n) = \{h_{rel} * s_L\}(n) + y_R(n) \tag{6}$$

Použijeme-li vstupní signál, záměrně pořízený tak, aby neobsahoval šum, ale pouze užitečný signál, můžeme ho vyjádřit pomocí vztahů (7), (8)

$$x_L(n) = s_L(n) \tag{7}$$

$$x_R(n) = \{h_{rel} * s_L\}(n) \tag{8}$$

a následně z nich vyjádřit rovnost (9).

$$x_R(n) - \{h_{rel} * x_L\}(n) = 0 \tag{9}$$

Zabarvení h_{rel} je vlastně filtr, jehož koeficienty se snažíme najít. V ideálním případě pro všechna n platí rovnice (9). Ve skutečnosti však takovéto situace nelze vždy dosáhnout a snažíme se jí pouze co nejvíce přiblížit. Je tedy potřeba zvolit kritérium, podle kterého posoudíme, jak moc se vypočítaný filtr blíží ideální situaci.

$$\min_{h_{rel}} crit[x_R - \{h_{rel} * x_L\}] \tag{10}$$

Tento problém velmi dobře vystihuje kvadratické kritérium, které je obecně popsáno vztahem (11),

$$crit_{quad}[x] = \sum_i x(i)^2 \tag{11}$$

dosazením výrazu $x_R - \{h_{rel} * x_L\}$ za x dostáváme vztah (12), který lze přepsat také vektorově

$$crit_{quad}[x_R - \{h_{rel} * x_L\}] = \sum_{n=1}^{N} (x_R(n) - \{h_{rel} * x_L\}(n))^2 = \|\mathbf{x}_R - \mathbf{X}_L\mathbf{h}_{rel}\|_2^2, \tag{12}$$

kde

$$\begin{aligned}
\mathbf{x}_R &= [x_R(1), x_R(2), \ldots, x_R(N)]^T, \\
\mathbf{X}_L &= \begin{pmatrix} x_L(1) & 0 & 0 & \ldots & 0 \\ x_L(2) & x_L(1) & 0 & \ldots & 0 \\ x_L(3) & x_L(2) & x_L(1) & \ldots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_L(N) & x_L(N-1) & x_L(N-2) & \ldots & x_L(N-L+1) \end{pmatrix}, \\
\mathbf{h}_{rel} &= [h_{rel}(1), h_{rel}(2), \ldots, h_{rel}(N)]^T.
\end{aligned}$$

Hledáme tedy filtr h_{rel} takový, aby byl výraz $\|\mathbf{x}_R - \mathbf{X}_L\mathbf{h}_{rel}\|_2^2$ minimální. Takovýto nalezený filtr označíme h_{LS}. Viz (13).

$$h_{LS} = arg \min_{h_{rel} \in R^L} \|\mathbf{x}_R - \mathbf{X}_L\mathbf{h}_{rel}\|_2^2 \tag{13}$$

Řešením vztahu (13) je

$$h_{LS} = (\mathbf{X}_L^T\mathbf{X}_L)^{-1}\mathbf{X}_L^T\mathbf{x}_R. \tag{14}$$

Nyní se opět vraťme k situaci, kdy je ve vstupním signálu přítomen šum. Výstup TCF z je v našem případě dán vztahem (15).

$$z(n) = \{h_{LS} * x_L\}(n) - x_R(n) \tag{15}$$

Uvažujeme-li ideální případ a dosadíme do vztahu (15) ideální filtr h_{rel} a signály s šumem (5) a (6), pak dostaneme vztah (16).

$$\begin{aligned} z(n) = \{h_{rel} * x_L\}(n) - x_R(n) = \{h_{rel} * s_L\}(n) + \{h_{rel} * y_L\}(n) \\ -\{h_{rel} * s_L\}(n) - y_R(n) = \{h_{rel} * y_L\}(n) - y_R(n) \end{aligned} \tag{16}$$

Je tedy patrné, že výstupem TCF z je signál, který obsahuje pouze šum. Uvedený postup je též popsaný v prezentaci [1].

Všechny výše popsané operace se signálem platí v časové oblasti. Do ní se analogový signál dostane pouhým navzorkováním určitou vzorkovací frekvencí f_S. Samotný proces odstraňování šumu však bude probíhat v oblasti frekvenční.

1.2 Frekvenční oblast

Pro konverzi z časové do frekvenční oblasti je nejprve potřeba signál rozdělit na bloky o délce N. Vezmeme-li poté každý tento blok a aplikujeme na něj diskrétní Fourierovu transformaci (DFT), dostaneme blok komplexních čísel. Každé komplexní číslo z lze vyjádřit ve tvaru (17).

$$z = Ae^{j\phi} \tag{17}$$

Výsledná komplexní čísla poté vypovídají o amplitudě A a fázi ϕ určité frekvence v daném úseku signálu. To, o jaké frekvenci f_n dané číslo vypovídá, je dáno jeho pořadím n v daném bloku, a to podle vztahu (18)

$$f_n = \frac{f_S}{N}n, \tag{18}$$

kde $n = 0, 1, \ldots, (N-1)$.

Signál ve frekvenční oblasti je tedy reprezentován čísly, jež říkají, v jaké míře se v určitém časovém rozmezí vyskytují určité diskrétní frekvence [2].

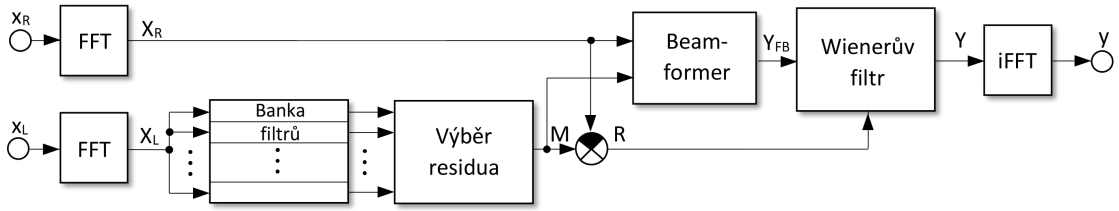
Pro přechod z časové do frekvenční oblasti je mnohem výhodnější použít rychlou Fourierovu transformaci (FFT) namísto DFT. Jedná se o velmi sofistikovaný algoritmus popsaný např. v [3, str. 255], který dokáže transformaci provést rychleji, je-li N mocninou čísla dvě.

Existuje také inverzní FFT (iFFT), která převádí signál z frekvenční oblasti do časové. Tento algoritmus lze jednoduše sestavit z následujících dílčích kroků:

1. Převedení všech komplexních čísel na čísla k nim komplexně sdružená.
2. Aplikování FFT.
3. Podělení všech čísel číslem N.

1.3 Odstranění šumu

Odstranění šumu bude probíhat podle diagramu na obrázku 1. Na začátku oba vstupní kanály převedeme pomocí FFT do frekvenční oblasti. Frekvenční obraz levého kanálu je použit jako vstup do každého filtru v předpřipravené bance. Výstupem z těchto filtrů jsou residua (16), která poskytují informaci o šumu. Z nich je jedno vybráno a po odečtení frekvenčního obrazu pravého kanálu je přivedeno do Wienerova filtru. Frekvenční obrazy obou kanálů jsou také přivedeny na vstup bloku Beamformer. Výstup z tohoto bloku je poté napojen na vstup Wienerova filtru. Výstupem z něho je signál s odfiltrovaným šumem, který vstupuje do bloku iFFT.



Obrázek 1: Struktura filtrování

Některé bloky z obrázku 1 si nyní popíšeme podrobněji.

1.3.1 Výběr residua

Vstupem do tohoto bloku je K residuí R_k o délce N vzorků vzniklých po paralelním filtrování všemi (K) filtry v bance. Blok nejprve vypočítá energie E_k všech těchto residuí podle vztahu (19) a následně na výstup pošle residuum s nejmenší energií.

$$E_k = \sum_{n=1}^{N} \mid R_k(n) \mid^2 \tag{19}$$

1.3.2 Beamformer

Vstupem do Beamformeru je signál z pravého vstupních kanálů X_R a signál M. Tento blok má za úkol vytvořit ze dvou vstupních signálů jeden signál. Pro tento účel lze použít například sečtení vstupu X_R a signálu M. Signál M je výsledkem násobení vstupu X_L a impulzní odezvy filtru H_{rel}, jehož residuum bylo v danou chvíli vybrané (20).

$$Y_{FB} = X_R + M = X_R + X_L H_{rel} \tag{20}$$

Výstupem je spojený a zesílený vstupní signál Y_{FB}.

1.3.3 Wienerův filtr

Vstupem do Wienerova filtru je signál Y_{FB}, který obsahuje užitečný signál i šum a signál R, který nese informaci o šumu. Úkolem tohoto bloku je využít informace o šumu R pro jeho zredukování ve výsledném signálu Y oproti vstupnímu Y_{FB}. Toho je dosaženo pomocí vztahu (21), kde τ je konstanta udávající sílu potlačení.

$$Y = Y_{FB} \frac{\mid Y_{FB} \mid^2}{\mid Y_{FB} \mid^2 + \tau \mid R \mid^2}, \tag{21}$$

V následující kapitole provedeme výčet prostředků, které budeme při práci používat.

2 Použité prostředky

V této části popíšeme prostředky použité pro tvorbu výsledného kódu. Jedná se především o samotné zařízení, na kterém bude algoritmus realizován a vývojové prostředí pro něj určené. Popsány jsou však i další nezbytné prostředky.

2.1 Digitální signálový procesor

Realizace veškerého zpracování signálu bude probíhat na signálovém procesoru. Digitální signálový procesor (DSP) je procesor, který je svými vlastnostmi a instrukční sadou přizpůsoben co nejefektivnějšímu zpracování diskrétního signálu v reálném čase. Jelikož přiváděný signál bývá zpravidla analogový, stejně jako očekávaný výstup, je potřeba použít A/D a D/A převodníky, jak je naznačeno na obrázku 2 [4].

Obrázek 2: Struktura zpracování analogového signálu DSP

V této práci je použit balíček TMS320C6416 Starter kit, který obsahuje:

- procesor TMS320C6416 osazený na desce společně se všemi potřebnými periferiemi (DSK),
- USB kabel pro propojení s počítačem,
- zdroj napájení,
- CD s instalací vývojového prostředí,
- zkušební verzi programu Matlab,
- stručný manuál.

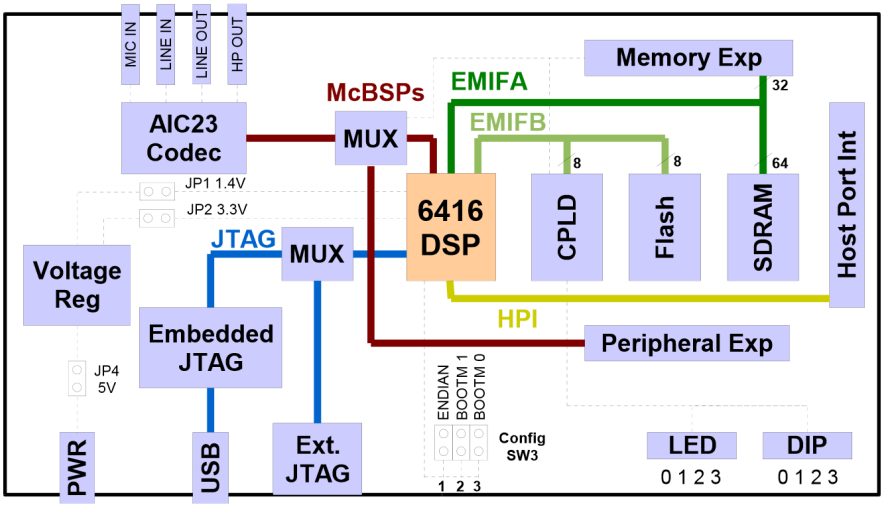
Balíček je tedy plně připraven pro vývoj aplikací pro daný signálový procesor [5].

Samotná deska obsahuje:

- procesor TMS320C6416 pracující na frekvenci 1 GHz,
- AIC23 stereo kodek,
- 16 MB synchronní DRAM,
- 512 kB non-volatilní paměti FLASH,

- 4 uživatelsky programovatelné LED,
- 4 uživatelsky programovatelné DIP přepínače,
- rozšiřující konektory pro připojení externích periferií,
- USB konektor a JTAG emulátor pro komunikaci s počítačem,
- konektor pro připojení napájecího zdroje (+5V DC).

Diagram vystihující vazby mezi jednotlivými komponentami desky je vidět na obrázku 3. Podrobný popis všech bloků na tomto diagramu je uveden v publikaci [6].



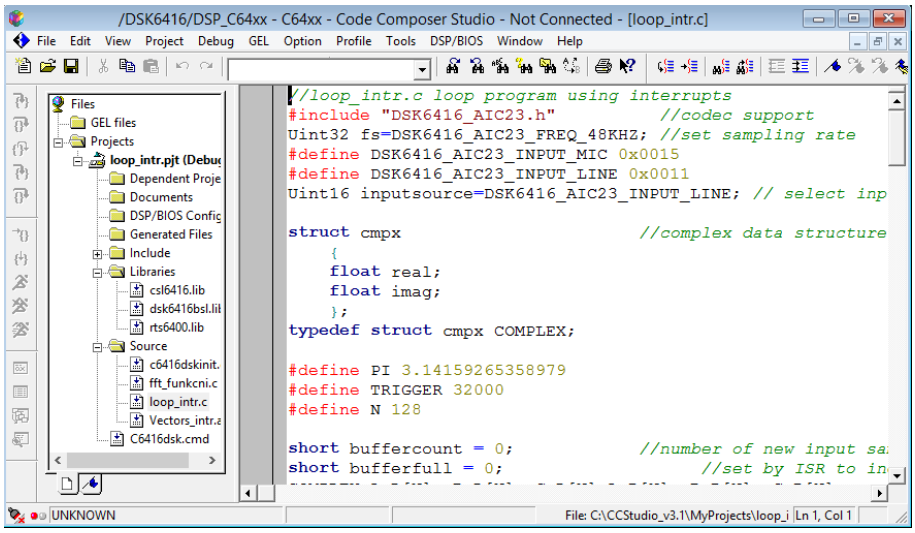
Obrázek 3: Blokový diagram DSK [6]

2.2 Code Composer Studio

Pro vývoj aplikací určených pro DSP od Texas Instruments se používá prostředí Code Composer Studio (CCStudio). V současné době je dostupné ve verzi 5. V této práci je používáno CCStudio verze 3.1, vydané v roce 2005. Důvodem k použití starší verze je lepší kompatibilita s výše uvedeným procesorem, jak bylo zjištěno během programování.

CCStudio umožňuje vytvářet projekty, psát zdrojový kód se zvýrazněním syntaxe, kompilovat, nahrávat kód do desky a ladit program za běhu. Na obrázku 4 je screenshot z tohoto vývojového prostředí. Je na něm vidět manažer projektů (vlevo), prostor pro psaní kódu (vpravo), ukazatel připojení desky a běhu programu (vlevo dole).

Pro vytváření programu v CCStudiu lze použít jazyk C a Assembler. V rámci jednoho projektu je možné používat jak soubory **.C* (s kódem v jazyce C), tak



Obrázek 4: Ukázka prostředí CCStudio v3.1

**.ASM* (s kódem v jazyce Assembler). Navíc **.C* soubory mohou obsahovat části kódu napsané v jazyce Assembler [7].

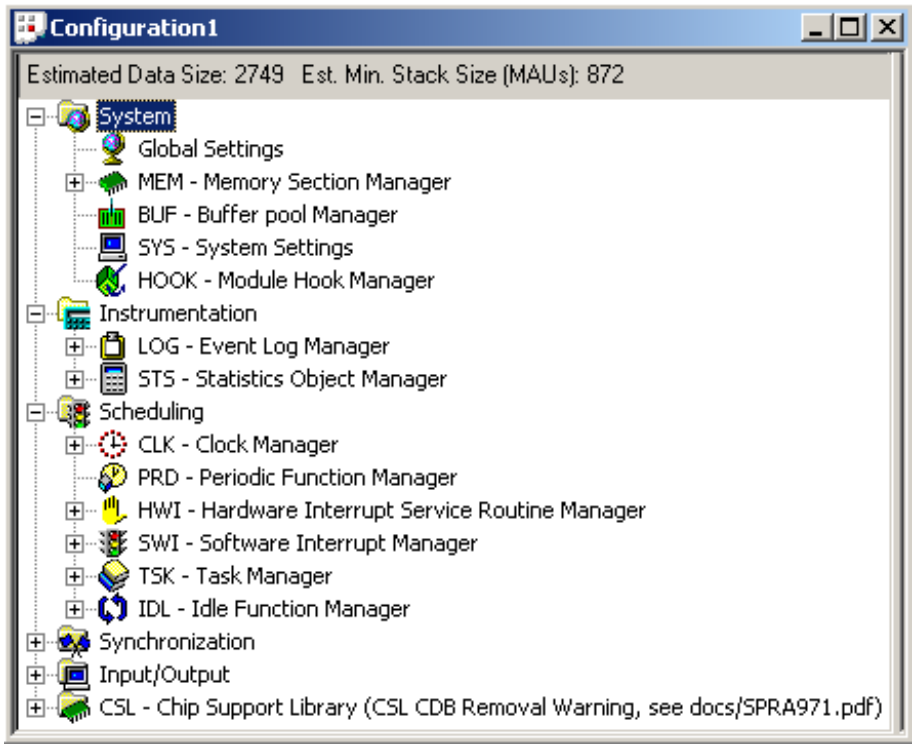
Instalační soubory CCStudia v3.1 jsou umístěny na CD obsaženém ve Starter kitu. S instalací tohoto prostředí se současně do stejné složky rozbalí i ukázkové projekty. Tyto projekty obsahují všechny potřebné hlavičkové soubory a knihovny pro práci s deskou. Obdobné projekty je také možné nalézt na CD přiloženého ke knize [3]. Zde jsou obsaženy i ukázky zpracování signálu v reálném čase apod. Z těchto ukázkových projektů budeme následně vycházet při sestavování kódu.

2.2.1 Systém DSP/BIOS

DSP/BIOS je operační systém, který se stará o předávání řízení programu v reálném čase. CCStudio obsahuje nástroj (na obrázku 5), pomocí kterého lze jednoduše nastavovat všechny parametry tohoto operačního systému. Je zde například vidět možnost nastavení rozložení paměti, logování, nastavení softwarových a hardwarových přerušení apod. Použití vlastního nastavení DSP/BIOS v projektu není nutné, jeho výchozí verze je generována automaticky, není-li přítomen vlastní soubor s nastavením. Použití tohoto souboru a vygenerovaní vlastní verze operačního systému je však velmi vhodné v projektech, které vyžadují složitější předávání řízení mezi jednotlivými funkcemi [8],[3, str. 374].

2.3 Matlab

Program Matlab využijeme pro externí řízení CCStudia a dále také jako výpočetní nástroj pro sestavení banky filtrů.



Obrázek 5: Nástroj pro nastavení systému DSP/BIOS v CCStudiu

CCStudio je možné řídit pomocí různých doplňků i z jiných programů. Například použijeme-li doplněk *MATLAB link for CCS*, můžeme použít Matlab pro spuštění kompilace, nahrání kódu do desky, výměnu dat za běhu programu apod. [3]. Je však potřeba nainstalovat verzi R2006b, kvůli kompatibilitě se starší verzí CCStudia.

2.4 VirtualBox

Oracle VM VirtualBox je opensource virtualizační nástroj umožňující na jednom fyzickém hostitelském počítači spustit jeden i více virtuálních hostovaných počítačů. O využití virtuálního počítače je vhodné uvažovat, nechceme-li aby aplikace, kterou se chystáme použít, ovlinila hostitelský systém a aplikace instalované v něm, nebo naopak. Dalším možným důvodem je nekompatibilita aplikace a hostitelského operačního systému [9].

Protože CCStudio verze 3.1 i Matlab R2006b pocházejí z doby Windows XP, je vhodné použít pro práci s nimi tento systém. Instalovat ho přímo na fyzický počítač už však není kvůli jeho zastaralosti a ukončené podpoře příliš bezpečné [10]. Použitím nástroje VirtualBox lze však systému například znemožnit přístup k internetu a tím ho ochránit před případnými hrozbami zvenčí. Zároveň je celé vývojové prostředí nainstalováno na čistém systému a vyhýbáme se tak případné

interferenci s ostatními aplikacemi.

2.5 Mikrofony a předzesilovač

Pro účely testování použijeme dva aktivní směrové stolní mikrofony AKG C680BL (blíže specifikované v příloze B)[11] připojené do předzesilovače M-AUDIO Audio-Buddy (viz přílohu C)[12]. Ten zajistí dostatečnou citlivost mikrofonů, která je navíc nastavitelná pomocí potenciometrů na jeho přední straně. Předzesilovač navíc disponuje funkcí PHANTOM, která zajistí přivedení stejnosměrného napětí do aktivních mikrofonů.

V další kapitole se budeme věnovat sestavení programu za použití popsaných prostředků.

3 Praktická část

V této práci se budeme zabývat odstraněním šumu z řeči snímané dvěma mikrofony, které jsou vůči sobě ve fixní poloze. Pozice mluvčího vůči těmto mikrofonům se však v čase mění. Použijeme k tomu metodu využívající banky předem vypočítaných filtrů, které vycházejí z nahrávky v klidném prostředí (bez šumu). Mluvčí se při tvorbě této nahrávky pohybuje vůči mikrofonům v oblasti, ve které chce, aby následně fungovalo filtrování šumu.

3.1 Zpracování v počítači

Nejprve je potřeba vytvořit banku filtrů. Tento proces bude probíhat v počítači a veškeré potřebné výpočty zajistí scripty pro program MATLAB. Banka filtrů se musí skládat z několika TCF (viz kapitolu 1.1), z nichž každý odpovídá jiné pozici mluvčího vůči mikrofonům. Máme dvě možnosti, jak banku sestavit:

1. Nahrát určitý počet kratších nahrávek, každou pro jinou pozici mluvčího vůči mikrofonům, přičemž se mluvčí v rámci jedné nahrávky nehýbe. Z každé nahrávky poté vypočítat jeden filtr.
2. Nahrát jednu delší nahrávku, přičemž mluvčí se plynule pohybuje vůči mikrofonům v určité oblasti. Tuto nahrávku poté rozdělit na kratší části a pro ty vypočítat jednotlivé filtry [13].

Vzhledem k tomu, že první způsob je pro mluvčího méně pohodlný, použijeme způsob druhý.

Připojíme výstup předzesilovače do vstupu počítače. Vytvoříme MATLAB skript, který provede následující operace:

1. Nahrání zvuku z mikrofonů o délce 5 sekund pomocí komponenty *audiorecorder*.
2. Vytvoření banky filtrů z této nahrávky pomocí vytvořené níže popsané funkce *CFBlearning*.
3. Uložení koeficientů vypočítaných filtrů do souboru ve správném formátu tak, aby je bylo možné načíst jako dvourozměrné pole v jazyce C.
4. Otevření projektu v CCStudiu, jeho zkompilování, nahrání do připojené desky a spuštění.

3.1.1 Nahrání zvuku

Pro nahrání zvuku z mikrofonu existuje v MATLABu komponenta `audiorecorder`. Její inicializaci vyvoláme tímto příkazem:

```
recorder = audiorecorder(sampling_freq, bit_depth, channels)
```

Prvním parametrem je vzorkovací frekvence `sampling_freq`, druhým bitová hloubka `bit_depth` a třetím počet kanálů `channels`. Nahrání zvuku probíhá pomocí příkazu:

```
recordblocking(recorder, length)
```

Parametr `length` určuje délku nahrávání v sekundách. Vložení nahraných dat do matice `matrix` na pracovní ploše MATLABu zajišťuje příkaz [14]

```
matrix = getaudiodata(recorder)
```

3.1.2 Vytvoření banky filtrů

Pro vytvoření banky filtrů sestavíme samostatnou funkci pro MATLAB:

```
function [res g] = CFBlearning(x, L, tau_1, tau_2, wlength, shift)
```

Ta rozdělí vstupní signál na jednotlivé úseky určité délky specifikované vstupním parametrem `wlength`. Pro ně poté počítá jednotlivé TCF dle vztahů uvedených v kapitole 1.1.1. Pohyb mluvčího nemusí být rovnoměrný, pozice kterými prochází se mohou opakovat a některé filtry nemusejí být vůbec vypovídající, jelikož se mluvčí například na okamžik odmlčí. Je tedy potřeba vybírat z vypočítaných filtrů jen ty opravdu použitelné. Proto script počítá průběžně také útlumy jednotlivých filtrů a zjistí-li, že je útlum menší, než vstupní parametr `tau_1`, je takový filtr vyřazen. Stejně tak je filtr vyřazen, nemá-li alespoň o `tau_2` lepší útlum, než předchozí přijatý filtr (`tau_2` je též vstupní parametr). Dalšími parametry funkce `CFBlearning` jsou vstupní signál `x`, délka výsledných filtrů `L` a překryv při rozdělování signálu na části `shift`.

3.1.3 Ovládání CCStudia

Pomocí příkazů v MATLABu je díky doplňku *MATLAB link for CCS* možné ovládat CCStudio. Následujícím příkazem vybereme číslo desky (`boardNum`) a procesoru (`procNum`), se kterými budeme dále pracovat:

```
[boardNum, procNum] = boardprocsel
```

Oba získané údaje využijeme pro vytvoření spojení s CCStudiem:

```
cc = ccsdsp('boardnum', boardNum, 'procnum', procNum)
```

Poté můžeme například otevřít projekt příkazem, jehož prvním parametrem je cesta k souboru *.pjt* a druhým klíčový řetězec *project*:

```
cc.open(pjt_file_path,'project');
```

Celý projekt lze následně zkompilovat příkazem:

```
cc.build('all');
```

Výsledný soubor *.out*, vzniklý po zkompilování je prvním parametrem následující funkce, která načte program do desky. Druhým parametrem je časový limit v sekundách pro provedení této operace:

```
cc.load(out_file_path,timeout);
```

Nestihne-li se program nahrát v uvedeném čase, je vrácena chyba. Spuštění programu v desce lze provést příkazem [15]:

```
cc.run('run');
```

3.2 Odstranění šumu v reálném čase

O odstranění šumu se stará DSP popsaný v kapitole 2.1. Celý proces probíhá v reálném čase dle diagramu 1. Na konektor *line-in* přichází vstupní stereo signál. Ten se v desce zpracuje a na konektor *headphones* je poslán výstupní mono signál s odfiltrovaným šumem.

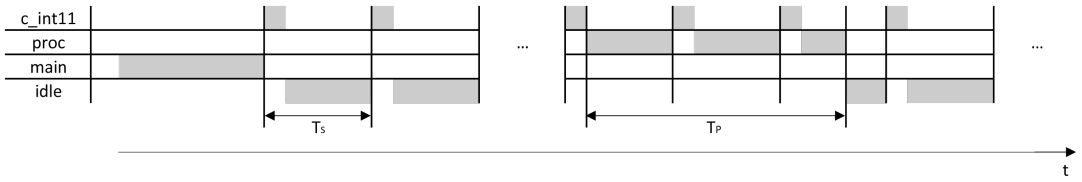
Zpracování signálu probíhá po blocích vzorků o délce určené konstantou *PTS*. Pracuje se se třemi takovýmito bloky, které jsou uloženy v jednotlivých bufferech. První buffer je postupně se vzorkovací frekvencí naplňován vzorky vstupního signálu. Druhý buffer je zpracováván. Třetí buffer je postupně se vzorkovací frekvencí odesílán na výstup. Každý blok signálu postupně prochází všemi těmito třemi buffery. Tento způsob zpracování je popsán v [3, str. 386].

Kód programu pro DSP má následující strukturu. Po spuštění proběhne jednou funkce *main*. Poté začne program reagovat na hardwarové přerušení od kodeku a s každým příchodem tohoto přerušení vyvolá funkci *c_int11*. Ta provede načtení jednoho vzorku vstupního signálu z prvního bufferu a odeslání jednoho vzorku výstupního signálu ze třetího bufferu. Dojde-li na konec bufferu, vyvolá softwarové přerušení (funkci *proc*), které se stará o zpracování druhého bufferu a rotaci všech tří bufferů.

Zpracování druhého bufferu je v sestaveném programu časově nejnáročnější procedurou, není však tak důležité, kdy probíhá, jedinou podmínkou je, aby byla dokončena dříve, než dojde k naplnění dalšího vstupního bufferu. Načtení vstupního vzorku a odeslání výstupního vzorku je naopak velmi rychlé, ale je potřeba ho provést přesně ve chvíli, kdy přijde přerušení od kodeku. Tím, že má hardwarové přerušení vyšší prioritu, než softwarové, je zajištěno, že není zmeškán žádný vstupní ani výstupní

vzorek signálu a současně nedochází k žádným zbytečným časovým prodlevám, kdy by procesor pouze čekal na vstupní vzorek.

Předávání řízení mezi jednotlivými funkcemi je vidět na diagramu 6, kde T_S je perioda vzorkování a T_P doba průběhu funkce *proc*. Jednotlivé zabarvené části značí, která funkce se v danou chvíli vykonává. Z diagramu je patrné, že nejprve se jednou vykoná funkce *main*. Poté začne vzorkování signálu. Než se načte první blok vstupních dat (o délce PTS vzorků), je procesor v módu *idle*, kde pouze čeká na přerušení. Po každých PTS přerušeních dojde k naplnění vstupního bufferu a řízení programu je předáno funkci *proc*, která tento buffer následně zpracovává. I během zpracování je však nutné stále se vzorkovací frekvencí přijímat vzorky signálu ze vstupu. To je zajištěno vyšší prioritou funkce *c_int11* oproti funkci *proc*.



Obrázek 6: Diagram předávání řízení jednotlivých částí programu

Všechny zdrojové kódy jsou na přiloženém CD, jehož obsah je popsán v příloze A. V následující kapitole se budeme věnovat testování sestaveného programu.

4 Testování

Funkčnost sestaveného programu bude vyhodnocena několika nezávislými posluchači. Těm bude předvedena sada zarušených testovacích nahrávek mluveného slova před a po aplikování algoritmu. Svůj subjektivní názor vyjádří výběrem vhodného tvrzení v dotazníku.

V této kapitole popíšeme přípravu testovacích nahrávek, průběh testování a výsledky ankety.

4.1 Vytvoření testovací nahrávky

Pro účely testování byly vytvořeny čtyři nahrávky o délce 5 sekund:

1. nahrávka ženského hlasu v klidném prostředí,
2. nahrávka ženského hlasu s luskáním.
3. nahrávka ženského hlasu s několika dalšími hlasy v pozadí [16].
4. nahrávka ženského hlasu se zapnutým vysavačem v pozadí.

První z uvedených byla použita jako vstup `x` do funkce `predvedeni` v MATLABu, pomocí které se sestavila banka filtrů. Ta byla předána jako součást zkompilovaného projektu DSP.

Do vstupu desky *line in* byly po spuštění programu předány z počítače postupně všechny ostatní nahrávky. Výstup desky *headphones* byl propojen s audio vstupem počítače, který zaznamenal příchozí zvukovou stopu. Vznikly tedy další tři nahrávky se stejným obsahem, avšak s redukovanými okolními ruchy.

4.2 Průběh testování

Bylo vybráno 20 dobrovolníků, kterým byly postupně přehrány všechny tři dvojice nahrávek před a po aplikování algoritmu. Každý respondent slyšel každou z nahrávek pouze jednou, aby bylo testování co nejobjektivnější. Po každé přehrané dvojici nahrávek respondent vyplnil hodnocení do dotazníku v příloze D. První sloupec dotazníku se týká kvality odstranění ruchu, druhý srozumitelnosti hlasu řečníka. Každé tvrzení v obou sloupcích je současně označeno známkou 1 (nejlépe) až 5 (nejhůře).

4.3 Výsledky testování

Z dat získaných z dotazníků byly vytvořeny dva grafy. První graf vychází z prvního sloupce dotazníku a vypovídá o názoru respondentů na míru odstranění ruchu

v jednotlivých nahrávkách. Druhý graf odpovídá druhému sloupci a vystihuje názor respondentů na srozumitelnost řečníka. Oba grafy vyjadřují absolutní četnost známek, které získaly jednotlivé dvojice nahrávek.

4.3.1 Míra odstranění ruchu v nahrávkách

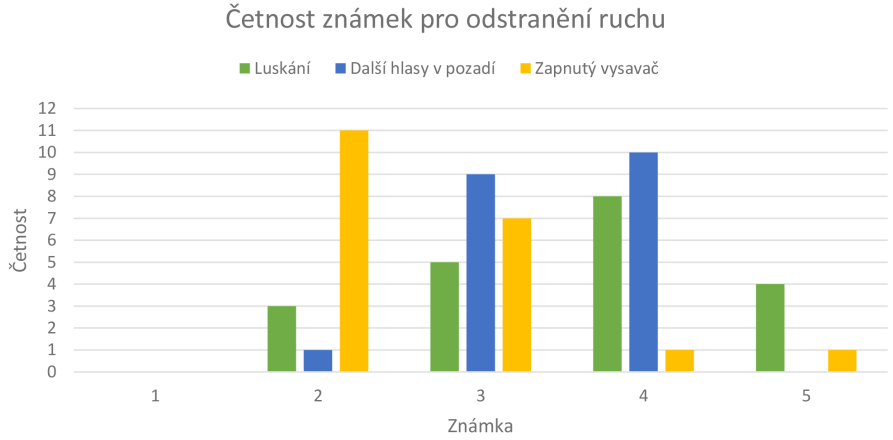
Pohlédneme-li na graf na obr. 7, zjistíme, že známku 1 (ruch byl zcela odstraněn) neudělil žádný z respondentů.

Zaměříme-li se na nahrávky s luskáním, je patrné, že většina respondentů si myslí, že ruch v obou nahrávkách je stejně hlasitý. Čtvrtina se domnívá, že byl ruch ve druhé nahrávce mírně zredukován. Zbylí dotazovaní udělili s téměř shodnou četností známky 2 (ruch byl velmi výrazně zredukován) a 5 (ruch je ve druhé nahrávce hlasitější než v první).

Dvojice nahrávek s dalšími hlasy v pozadí je téměř shodným počtem respondentů ohodnocena známkou 3 (ruch byl mírně zredukován) a 4 (ruch je stejně hlasitý v obou nahrávkách). Počty ostatních známek pro tuto dvojici jsou zanedbatelné.

Nahrávky se zapnutým vysavačem ohodnotila většina respondentů známkou 2 (ruch byl velmi výrazně zredukován). Přibližně třetina dotazovaných si myslí, že ruch byl mírně zredukován. Počty ostatních známek pro tuto dvojici jsou zanedbatelné.

Nejlépe ze všech tří dvojic tedy vyšly nahrávky se zapnutým vysavačem (průměrná známka 2,6), následují nahrávky s dalšími hlasy v pozadí (průměrná známka 3,45) a nakonec nahrávky s luskáním (průměrná známka 3,65)

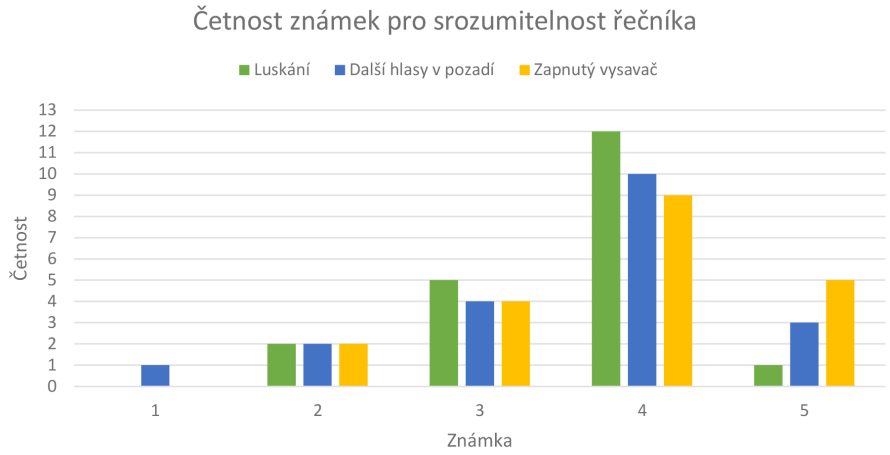


Obrázek 7: Graf vyjadřující četnost známek pro odstranění ruchu

4.3.2 Srozumitelnost řečníka v nahrávkách

Z grafu na obr. 8 je vidět, že názory respondentů na srozumitelnost řečníka se pro jednotlivé nahrávky neliší nijak významně. Známka 4 (řečník je ve druhé nahrávce méně srozumitelný než v první) převládá pro všechny dvojice nahrávek nad ostatními známkami. Necelá čtvrtina odpovídajících si myslí, že srozumitelnost řečníka je v obou nahrávkách stejná. Čtvrtina respondentů se také domnívá, že řečník je v odfiltrované nahrávce se zapnutým vysavačem velmi špatně srozumitelný oproti neodfiltrované nahrávce.

Průměrné známky jednotlivých dvojic nahrávek se téměř shodují. Nahrávky se zapnutým vysavačem dosáhly průměru 3,85. Nahrávky s luskáním a s dalšími hlasy v pozadí mají shodnou průměrnou známku 3,6.



Obrázek 8: Graf vyjadřující četnost známek pro srozumitelnost řečníka

4.4 Zhodnocení výsledků testování

Podívejme se nejprve na výsledky druhého sloupce dotazníku, tedy porovnání srozumitelnosti řečníka v obou nahrávkách. Zde zcela jasně převládá názor, že řečník je po aplikaci algoritmu méně srozumitelný, než před ní. Toto je způsobeno především těmito faktory:

1. vysoké vytížení DSP při provádění algoritmu,
2. vysoký koeficient τ ve Wienerově filtru.

DSP je velmi vytížen prováděním algoritmu v reálném čase - tedy dochází k porušení podmínky, že zpracování bufferu nesmí trvat déle, než naplnění dalšího vstupního bufferu popsané v kapitole 3.2. To způsobuje, že po aplikování algoritmu je

nahrávka „trhaná“ (je slyšet „praskání“). Tomuto problému jsme se snažili při vytváření testovacích nahrávek co nejvíce předejít tím, že jsme použili banku pouze o jednom filtru. Počet filtrů v bance je totiž přímo úměrný době zpracování.

Dále je v hlase řečníka po aplikaci algoritmu slyšet tzv. hudební šum. Ten je způsoben zvoleným vyšším koeficientem τ ve Wienerově filtru (21). Volíme ho takto, jelikož čím vyšší tento koeficient je, tím více je šum odstraněn. Současně se však znepříjemní poslech hlasu řečníka. Je tedy potřeba zvolit vhodný kompromis.

Z výsledků prvního sloupce dotazníku lze usuzovat, že mnoha posluchačům se zdá šum v nahrávkách po aplikaci algoritmu alespoň mírně zredukován. Je však také vidět, že záleží, o jaký ruch se jedná. Nejhůře dopadly nahrávky s luskáním, u nichž se dokonce necelé čtvrtině respondentů jevilo, že je ruch po aplikaci algoritmu hlasitější než před ní. Šlo však současně o dvojici nahrávek s nejrozporuplnějšími výsledky, tedy získala i mnoho kladných hodnocení. Očividně nejlépe dopadla dvojice nahrávek se zapnutým vysavačem, kde téměř všichni respondenti slyšeli zredukování ruchu.

V článku [13] byl otestován stejný algoritmus, avšak jiným způsobem. Autoři provedli porovnání koeficientu SNR (signal to noise ratio) před a po aplikaci algoritmu. Výsledky byly rovněž kladné.

Závěr

V práci jsme se seznámili s principem fungování algoritmu pro redukci šumu za použití dvou mikrofonů a banky předměřených TCF. Naučili jsme se také programovat a využívat digitální signálový procesor TMS320C6416 a popsali jsme prostředky k tomu potřebné. Mnoho úsilí bylo vynaloženo při pokusech o vývoj aplikace za použití soudobých prostředků (Windows 8.1, CCStudio v5, Matlab R2013a). Ukázalo se však, že jedinou možností pro správné fungování DSP je použití staršího prostředí (Windows XP, CCStudio v3.1, Matlab R2006b). Tento fakt by nebyl tolik limitující, nebýt ukončení podpory Windows XP. Tímto se stává jakékoliv využití tohoto systému potenciálně nebezpečné pro počítač, na kterém vývoj aplikace pro DSP probíhá. Naštěstí lze problém řešit separací vývojového prostředí na samostatný virtuální počítač a snížit tak rizika jím způsobená, čehož jsme v této práci využili.

V praktické části práce se nám podařilo sestavit kód výše uvedeného algoritmu v jazyce C. Při jeho testování jsme se však potýkali s nedostatečně rychlým zpracováním signálu pomocí DSP. Bylo nutné snížit počet filtrů v bance na jeden, aby procesor lépe stíhal provádět algoritmus v reálném čase. Nepodařilo se tedy vyzkoušet, jak je algoritmus schopen zredukovat šum při pohybu řečníka, jelikož k tomu je potřeba více filtrů v bance. Problém pomalého zpracování je způsoben tím, že některé procedury programu pro DSP napsané v jazyce C nejsou překladačem zcela efektivně převedeny do strojového kódu. Jedná se tedy o nedostatek CCStudia, který lze řešit pouze identifikováním, o které procedury se jedná a jejich přepsáním do jazyka Assembler. Tento proces může být časově i myšlenkově velmi náročný.

Abychom mohli konstatovat, zda algoritmus dokáže šum v nahrávce zredukovat, sestavili jsme sadu tří dvojic nahrávek před a po aplikování algoritmu a dotazník, o jehož vyplnění jsme požádali 20 dobrovolníků. Z výsledků uspořádané ankety se nám podařilo zjistit, že algoritmus pro jednu pozici mluvčího funguje. Subjektivní názory dotazovaných osob se sice liší v závislosti na odstraňovaném šumu, ale v průměru vypovídají o tom, že je šum po aplikaci algoritmu na zarušený signál řeči zredukován. Současně byli respondenti dotázáni také na porovnání srozumitelnosti mluvčího před a po aplikování algoritmu na nahrávku. Mezi odpověďmi převládal názor, že se po aplikování algoritmu srozumitelnost řečníka zhoršila. Tento výsledek je však stále především důsledkem problému pomalého zpracování popsaného v předchozím odstavci. Svůj podíl na této skutečnosti má i vybraný způsob odstranění šumu - Wienerův filtr. Ten zpravidla způsobuje zhoršení srozumitelnosti hlasu řečníka v nahrávce. Vhodným řešením by bylo zaměřit se na jiné filtry, které více zohledňují poslechovou kvalitu výsledných signálů.

Použitá literatura

[1] KOLDOVSKÝ, Zbyněk. *Audio filters with sparse impulse response* [prezentace]. Tehcnická univerzita v Liberci, Fakulta mechatroniky, informatiky a mezioborových studií, 14.11.2013.

[2] UHLÍŘ, Jan; SOVKA, Pavel. *Číslicové zpracování signálů*. Praha: Vydavatelství ČVUT, 1995. ISBN 80-01-01303-0.

[3] CHASSAING, Rulph; REAY, Donald. *Digital Signal Processing and Applications with the TMS320C6713 and TMS320C6416 DSK*. Wiley-IEEE Press, 2008 ISBN 978-0-470-13866-3.

[4] TIŠNOVSKÝ, Pavel. *Od mikrořadičů k digitálním signálovým procesorům (DSP)*. [online]. [cit. 2014-04-02]. Dostupné z: http://www.root.cz/clanky/od-mikroradicu-k-digitalnim-signalovym-procesorum-dsp/

[5] TMS320C6416 DSP Starter Kit (DSK). TEXAS INSTRUMENTS INCORPORATED. *Texas Instruments* [online]. [cit. 2014-04-02]. Dostupné z: http://www.ti.com/tool/tmdsdsk6416

[6] SPECTRUM DIGITAL, INC. *TMS320C6416 DSK Technical Reference.*[online]. [cit. 2014-04-02]. 505945-0001 Rev. A. Dostupné z: http://c6000.spectrumdigital.com/dsk6416/V1/docs/dsk6416_TechRef.pdf

[7] Using Inline Assembly in C/C++. *CodeProject* [online]. 2006 [cit. 2014-05-03]. Dostupné z: http://www.codeproject.com/Articles/15971/Using-Inline-Assembly-in-C-C

[8] DSP/BIOS Real-Time Operating System (RTOS) - DSPBIOS. TEXAS INSTRUMENTS INCORPORATED. *Texas Instruments* [online]. [cit. 2014-04-16]. Dostupné z: http://www.ti.com/tool/dspbios

[9] Oracle VM VirtualBox. ORACLE. *Oracle Technology Network* [online]. [cit. 2014-05-13]. Dostupné z: http://www.oracle.com/technetwork/server-storage/virtualbox/overview/index.html

[10] Konec podpory pro Windows XP. MICROSOFT. *Microsoft Windows* [online]. [cit. 2014-05-13]. Dostupné z: http://windows.microsoft.com/cs-cz/windows/end-support-help

[11] AKG C 680 BL: Table microphone. AKG ACOUSTICS. *AKG* [online]. [cit. 2014-04-07]. Dostupné z: http://www.akg.com/media/media/download/9444

[12] M-AUDIO - Audio Buddy - Budget Microphone Preamp: Table microphone. INMUSIC BRANDS, Inc. M-AUDIO. *M-AUDIO* [online]. [cit. 2014-04-07]. Dostupné z: http://www.m-audio.com/products/en_us/AudioBuddy.html

[13] KOLDOVSKÝ, Zbyněk, Petr TICHAVSKÝ a David BOTKA. *Noise Reduction in Dual-Microphone Mobile Phones Using A Bank of Pre-Measured Target-Cancellation Filters*. [online]. Proc. of ICASSP 2013, pp. 679-683, Vancouver, Canada, May 2013 [cit. 2014-05-03]. Dostupné z: http://itakura.ite.tul.cz/zbynek/pubs/icassp2013ZK.pdf

[14] MATLAB Function Reference: audiorecorder. NCI NATIONAL FACILITY. *National Computational Infrastructure* [online]. [cit. 2014-05-15]. Dostupné z: http://nf.nci.org.au/facilities/software/Matlab/techdoc/ref/audiorecorder.html

[15] POLÁK, Kamil. *Realizace digitálních audio efektů na signálovém procesoru TMS320C6416* Technická univerzita v Liberci, Fakulta mechatroniky, informatiky a mezioborových studií, 2011.

[16] Talking people noise. YOUTUBE. *Youtube* [online]. [cit. 2014-05-15]. Dostupné z: https://www.youtube.com/watch?v=GWxy38DCeqw

[17] AKG ACOUSTICS. *C 680 BL: Table microphone* [online]. [cit. 2014-05-08]. Dostupné z: http://www.alectrosystems.com/shop/data/C680BL.pdf

[18] M-AUDIO. *Audio buddy: Dual mic preamp / direct box* [online]. [cit. 2014-05-08]. Dostupné z: http://edmullen.net/manuals/MAudio_AudioBuddy_user_guide.pdf

A Obsah přiloženého CD

- Text bakalářské práce - soubor *bakalarska_prace_2014_Michal_Rosko.pdf*
- Zdrojové kódy - adresář *zdrojove_kody*
  soubor *fastconv.pjt* - projekt CCStudia
  soubor *predvedeni.m* - spouštěcí MATLAB skript
- Testovací nahrávky - adresář *nahravky*
  soubory končící *_v* - vstupní nahrávka
  soubory končící *_m* - vybraný jeden kanál vstupní nahrávky
  soubory končící *_r* - residuum po aplikování TCF na vstupní nahrávku
  soubory končící *_o* - nahrávka se zredukovaným ruchem

B Specifikace mikrofonů

Výrobce:	AKG
Typ:	C680BL
Frekvenční rozsah:	60 - 20 000 Hz
Poměr signál/šum:	>68 dB
Elektrická impedance:	<200 Ω
Napájení:	9 - 52 V
Konektor:	3-pin XLR
Délka kabelu:	3 m
Rozměry:	97 x 67 x 20 mm
Datasheet:	zdroj [17]

Obrázek 9: Mikrofony AKG C680BL

C Specifikace předzesilovače

Výrobce:	M-AUDIO
Typ:	Audio Buddy
Vstupy:	2 nezávislé konektory XLR
	2 nezávislé konektory 6,35 mm jack (alternativní)
Výstupy:	2 konektory 6,35 mm jack
Ovládání:	2 ovládání vybuzení
	vypínací tlačítko
	tlačítko pro zapínání přídavného mikrofonů (phantom)
Vybuzení mikrofonů:	max. 60 dB
Impedance mikr. vstupů:	1 kΩ
Napájení:	9 V AC, 500 mA
Rozměry:	140 x 84 x 44 mm
Datasheet:	zdroj [18]

Obrázek 10: Předzesilovač M-AUDIO AudioBuddy

D Testování algoritmu - dotazník

Postupujte prosím podle následujících pokynů:

1. Přečtěte si nadpis, abyste věděli, jakého ruchu si máte všímat.
2. Poslechněte si dvojici nahrávek hlasu (každou smíte slyšet pouze jednou).
3. V každém sloupečku zaškrtněte jednu možnost, která nejvíce vystihuje Váš pocit z dané dvojice nahrávek.
4. Přejděte k další dvojici.

Ruch - luskání:

☐ 1 Ruch ve druhé nahrávce byl zcela odstraněn.	☐ 1 Řečník je ve druhé nahrávce mnohem srozumitelnější, než v první.
☐ 2 Ruch ve druhé nahrávce je oproti první nahrávce velmi výrazně zredukován.	☐ 2 Řečník je ve druhé nahrávce mírně srozumitelnější, než v první.
☐ 3 Ruch ve druhé nahrávce je oproti první nahrávce mírně zredukován.	☐ 3 Řečník je ve druhé nahrávce stejně srozumitelný, jako v první.
☐ 4 Ruch ve druhé nahrávce je slyšet stejně hlasitě jako v první nahrávce.	☐ 4 Řečník je ve druhé nahrávce méně srozumitelný, než v první.
☐ 5 Ruch ve druhé nahrávce je hlasitější, než v první.	☐ 5 Řečník je ve druhé nahrávce velmi špatně srozumitelný, oproti první.

Ruch - další hlasy v pozadí:

☐ 1 Ruch ve druhé nahrávce byl zcela odstraněn.	☐ 1 Řečník je ve druhé nahrávce mnohem srozumitelnější, než v první.
☐ 2 Ruch ve druhé nahrávce je oproti první nahrávce velmi výrazně zredukován.	☐ 2 Řečník je ve druhé nahrávce mírně srozumitelnější, než v první.
☐ 3 Ruch ve druhé nahrávce je oproti první nahrávce mírně zredukován.	☐ 3 Řečník je ve druhé nahrávce stejně srozumitelný, jako v první.
☐ 4 Ruch ve druhé nahrávce je slyšet stejně hlasitě jako v první nahrávce.	☐ 4 Řečník je ve druhé nahrávce méně srozumitelný, než v první.
☐ 5 Ruch ve druhé nahrávce je hlasitější, než v první.	☐ 5 Řečník je ve druhé nahrávce velmi špatně srozumitelný, oproti první.

Ruch - zapnutý vysavač:

☐ 1 Ruch ve druhé nahrávce byl zcela odstraněn.	☐ 1 Řečník je ve druhé nahrávce mnohem srozumitelnější, než v první.
☐ 2 Ruch ve druhé nahrávce je oproti první nahrávce velmi výrazně zredukován.	☐ 2 Řečník je ve druhé nahrávce mírně srozumitelnější, než v první.
☐ 3 Ruch ve druhé nahrávce je oproti první nahrávce mírně zredukován.	☐ 3 Řečník je ve druhé nahrávce stejně srozumitelný, jako v první.
☐ 4 Ruch ve druhé nahrávce je slyšet stejně hlasitě jako v první nahrávce.	☐ 4 Řečník je ve druhé nahrávce méně srozumitelný, než v první.
☐ 5 Ruch ve druhé nahrávce je hlasitější, než v první.	☐ 5 Řečník je ve druhé nahrávce velmi špatně srozumitelný, oproti první.