VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

DIPLOMOVÁ PRÁCE

Brno, 2020 Bc. Vojtěch Panenka

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ
DEPARTMENT OF TELECOMMUNICATIONS

SLUCHÁTKA S ADAPTIVNÍM POTLAČENÍM ŠUMU
ADAPTIVE NOISE CANCELLATION HEADPHONE

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. Vojtěch Panenka**
AUTHOR

VEDOUCÍ PRÁCE — **doc. Ing. Petr Sysel, Ph.D.**
SUPERVISOR

BRNO 2020

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

magisterský navazující studijní obor **Audio inženýrství**

Ústav telekomunikací

Student: Bc. Vojtěch Panenka ***ID:*** 174370
Ročník: 2 ***Akademický rok:*** 2019/20

NÁZEV TÉMATU:

Sluchátka s adaptivním potlačením šumu

POKYNY PRO VYPRACOVÁNÍ:

Seznamte se s technologií adaptivního potlačení šumu (ANC) a komerčně nabízenými sluchátky s ANC. Prostudujte používané algoritmy a komerčně dostupná řešení. Následně realizujte buď doplnění běžných sluchátek o dva mikrofony nebo úpravu komerčních sluchátek s ANC tak, aby bylo možné doplnit vlastní externí algoritmus potlačení hluku. Analyzujte používané číslicové algoritmy adaptivního potlačení šumu, vybraný algoritmus implementujte a otestujte na vhodné platformě, např. vývojovém kitu DSK6416.

DOPORUČENÁ LITERATURA:

[1] FARHANG-BOROUJENY, B. Adaptive filters: theory and applications. New York: Wiley, 1998. ISBN 0-47--98337-3

[2] DAVIS, G., M. Noise reduction in speech applications. Boca Raton, Fla.: CRC Press, 2002. ISBN 0-849--0949-2

Termín zadání: 3.2.2020 ***Termín odevzdání:*** 1.6.2020

Vedoucí práce: doc. Ing. Petr Sysel, Ph.D.

prof. Ing. Jiří Mišurec, CSc.
předseda oborové rady

ABSTRAKT

Tato práce se věnuje analýze používaných technologií při návrhu sluchátek s aktivním potlačením okolního hluku a zkoumá možnosti využití adaptivních filtrů pro zjednodušení vývoje a dosažení efektivnějšího útlumu.

KLÍČOVÁ SLOVA

ANC, aktivní potlačení šumu, aktivní potlačení hluku, adaptivní filtrace, sluchátka, DSP, číslicové zpracování zvukových signálů

ABSTRACT

The thesis deals with the analysis of technology used during the design of headphones with integrated active ambient noise cancellation and examines the possibilities of using adaptive filters to simplify development and achieve more effective attenuation.

KEYWORDS

ANC, active noise cancellation, adaptive filtration, headphones, DSP, digital audio signal processing

PANENKA, Vojtěch. *Sluchátka s adaptivním potlačením šumu*. Brno, 2020, 143 s. Diplomová práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací. Vedoucí práce: doc. Ing. Petr Sysel, Ph.D.

PROHLÁŠENÍ

Prohlašuji, že svou diplomovou práci na téma „Sluchátka s adaptivním potlačením šumu“ jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

podpis autora

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu diplomové práce panu doc. Ing. Petru Syslovi, Ph.D. za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci.

Brno ....

podpis autora

Obsah

Seznam obrázků

Úvod

Tato diplomová práce se zabývá technologií aktivního potlačení hluku s využitím adaptivních filtrů. Jedná se o stále celkem novou a atraktivní technologii, která posluchači umožňuje vytvořit výborné poslechové podmínky i tam, kde to jindy není možné, např. ve vlaku, v letadle, na náměstí, v kavárně atp.

I když samotná myšlenka snímání hlukové složky a jejího následného odečtení pomocí generování inverzního průběhu zní jednoduše, praktická implementace je obecně složitější a vcelku komplexní.

Dnešní trend v oblasti mobilního audia nám nahrává do karet, protože oproti dřívějším trendům jsou nyní na trhu značně rozšířena tzv. on-ear sluchátka, tj. sluchátka, která obejmou svým náušníkem celý boltec lidského ucha. Taková sluchátka bývají větších rozměrů a tím pádem je jednodušší je doplnit o další technologii.

Přestože existují na trhu i in-ear sluchátka (do uší) využívající technologii ANC, s konstrukcí in-ear ANC sluchátek se pojí mnoho dodatečných omezení a proto jsou v rámci této práce použitá sluchátka typu on-ear.

1 Používané topologie

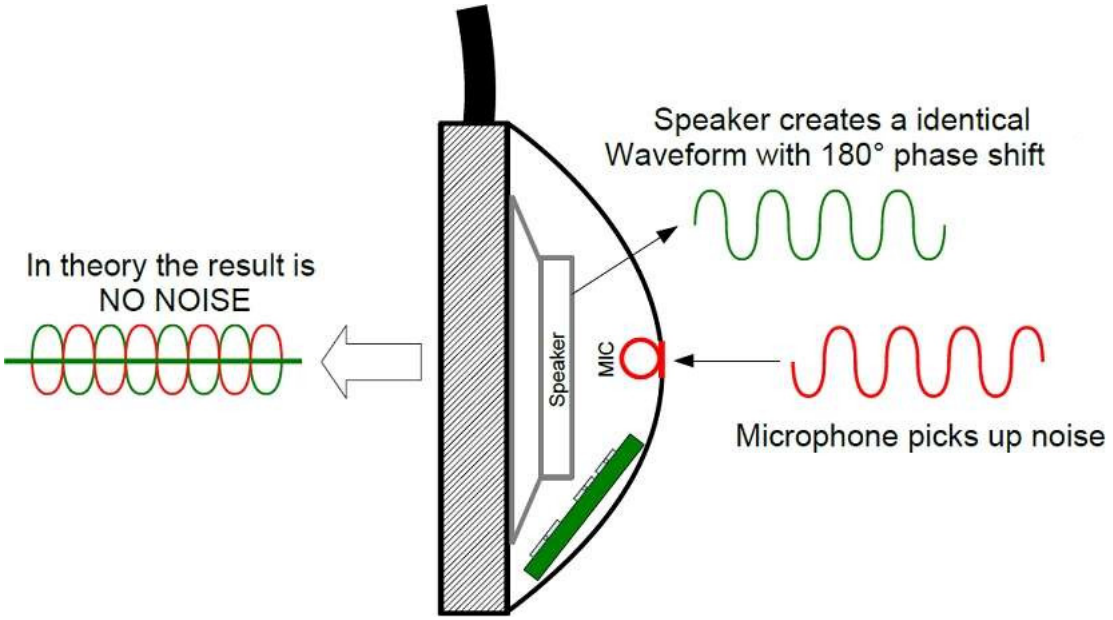
Pokud se podíváme na problematiku ANC (ANC - active noise cancelling) jako na celek, bez ohledu na to, jaká sluchátka právě držíme v ruce, bude možné použitý ANC systém zařadit do jedné z těchto tří základních topologií: dopředná (feed-forward), zpětnovazební (feedback) a hybridní (kombinovaná). Jednotlivé koncepty a jejich plusy i mínusy si popíšeme později, ve všech případech jde však vždy o aplikaci principu superpozice na signál, který je sejmut nějakým senzorem - mikrofonem, následně filtrován a znovu vyzářen do akustického prostoru pomocí reproduktoru.

Cílem použité filtrace je upravit vstupní signál tak, aby se po vyzáření zpět do akustického prostoru odečetl od signálu původního a tím jej potlačil.

Běžné pasivní metody potlačení hluku jsou bohužel rozměrné, drahé a neefektivní na nízkých kmitočtech, proto využijeme toho, že se elektromagnetické vlny šíří o několik řádů rychleji než ty akustické a tím nám poskytují dostatek času na zpracování vstupního signálu a následné vyzáření „antisignálu“.

2 Dopředný systém

Nejběžněji používaná (alespoň dle [28]) je topologie dopředná, fungující na následujícím principu. Jeden referenční mikrofon je umístěn na vnější stěně sluchátek a snímá veškeré ambientní zvuky. Signál z mikrofonu je následně zpracován pomocí ANC obvodu a spolu s potenciálním hudebním signálem je vyzářen do vnitřní dutiny sluchátka pomocí reproduktoru.



Obr. 2.1: Základní princip funkce dopředného ANC systému (převzato z [28])

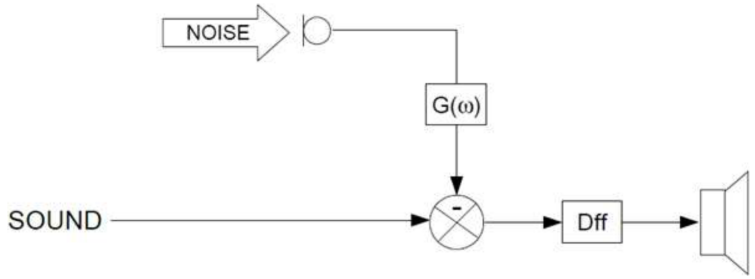
ANC za použití dopředné topologie je efektivní v širokém kmitočtovém pásmu, v praxi dle [28] dosahujícím až ke kmitočtu 3 kHz. I díky tomu je často používán např. v bezdrátových bluetooth headsetech a handsfree zařízeních, kde je srozumitelnost lidského hlasu hlavní prioritou. Kmitočty mezi 1 a 5 kHz jsou totiž v lidské řeči nositelem srozumitelnosti, i proto je dopředná ANC v této oblasti tak hodnotná.

Aby byl dopředný systém efektivní, je třeba zajistit dostatečnou izolaci referenčního mikrofonu od zvuku vyzářeného vnitřním reproduktorem - ideálně vhodně padnoucím náušníkem, což není vždy lehké, zvlášť v případě, kdy se jedná o jiný než over-ear headset. Jednou z dalších limitací je také fakt, že každý člověk má jiný tvar hlavy a uší a pasivní útlum sluchátek se může měnit v závislosti na tom, kdo si sluchátka nasadí - implementovaná přenosová funkce je tedy pouhou aproximací založenou na nějakém obecném modelu hlavy a uší (pokud je systém realizován jen pomocí fixních filtrů).

Návrh ANC s dopřednou topologií je vcelku jednoduchý, není totiž nutné řešit

stabilitu systému, jelikož referenční mikrofon přímo nesnímá zvuk vyzářený reproduktorem. Samotný fakt, že je mikrofon umístěný na vnější stěně sluchátka s sebou však přináší i jednu velkou nevýhodu - mikrofon je náchylný na hluk způsobený větrem.

2.1 Blokový diagram



Obr. 2.2: Blokový diagram dopředného ANC systému (převzato z [28])

Tento blokový diagram představuje základní dopředný ANC systém. Na vstupu je nějaký ambientní hluk, který je po sejmutí mikrofonem transformován pomocí filtru, charakterizovaného přenosovou funkcí $\boldsymbol{G}_{(\mathrm{w})}$, poté je sloučen s přehrávaným signálem a spolu s ním vyzářen pomocí reproduktoru charakterizovaného přenosovou funkcí $\boldsymbol{D}_{(\mathrm{w})}$.

2.2 Návrh filtru

Abychom mohli úspěšně implementovat dopředný ANC systém, je třeba po osazení sluchátek mikrofony definovat, jaké vlastnosti má mít filtr $\boldsymbol{G}_{(\mathrm{w})}$, konkrétně určit jeho amplitudový a fázový přenos v závislosti na kmitočtu.

Každá sluchátka mají jiné akustické vlastnosti, které ovlivňuje mnoho faktorů - rozměry obou akustických dutin (před a za reproduktorem), konstrukce, materiál, síla přítlaku k hlavě atd. Z toho důvodu není možné použít pouze obecnou charakteristiku ANC filtru, ale je třeba ji pro každý typ sluchátek změřit znovu.

2.2.1 Měření přenosu

K měření je třeba základní dvoukanálový FFT měřící systém (viz 8.2), referenční zdroj zvuku ([29] doporučuje koaxiální dvoupásmovou reprosoustavu pro eliminaci

časových problémů spojených se zvukem přicházejícím z více bodů v prostoru najednou) a měřící simulátor ucha - např. G.R.A.S. KEMAR se zabudovaným měřícím mikrofonem.

Určení $\boldsymbol{G}_{(\mathrm{w})}$ probíhá ve třech krocích: měření, výpočet a aproximace filtrem.

Prvním měřením zjistíme kmitočtově závislý pasivní útlum sluchátka.

Přenos lze měřit buď pomocí automaticky přelaďovaného sinusového signálu - tzv. "sine sweep"- od 20 Hz do 20 kHz nebo pomocí širokopásmového šumu, např. růžového.

Druhým měřením určíme přenos samotného referenčního mikrofonu umístěného na vnější stěně sluchátka, měření probíhá podobně jako v prvním případě, ale místo mikrofonu v simulátoru hlavy připojíme na vstup měřícího systému přímo výstup z referenčního ANC mikrofonu.

Třetím měřením určíme přenos zabudovaného reproduktoru. Tentokrát nepoužijeme jako referenční zdroj externí reproduktor, ale měřící signál přivedeme pomocí linkového vstupu přímo do reproduktoru sluchátka. Jako vstup do měřícího systému pak zvolíme stejně jako v případě prvního měření interní mikrofon simulátoru hlavy.

2.2.2 Výpočet výsledné přenosové charakteristiky

Výstupem tří předchozích měření jsou tři přenosové charakteristiky - pozor, nejde o přenosové funkce. Pro získání přenosových funkcí by bylo třeba tyto charakteristiky vhodně proložit křivkou a na základě této aproximace danou přenosovou funkci odhadnout.

Abychom tento časově náročný a nepřesný proces nemuseli podstupovat, [29] doporučuje využít toho, že jsme měřením získali určitý počet zesilovacích činitelů, jeden pro každé pásmo FFT. Tyto zesilovací činitele lze mezi sebou, na rozdíl od přenosových funkcí, jednoduše sčítat [29].

$$A_{\mathrm{G}_{(\mathrm{w})}} = A_{1_{(\mathrm{w})}} - A_{2_{(\mathrm{w})}} - A_{3_{(\mathrm{w})}}$$

$A_{\mathrm{G}_{(\mathrm{w})}}$ ... Výsledné zesílení filtru (závislé na kmitočtu) [dB]
$A_{1_{(\mathrm{w})}}$ ... Zesílení dle prvního měření (závislé na kmitočtu) [dB]
$A_{2_{(\mathrm{w})}}$ ... Zesílení dle druhého měření (závislé na kmitočtu) [dB]
$A_{3_{(\mathrm{w})}}$ ... Zesílení dle třetího měření (závislé na kmitočtu) [dB]

To samé platí i pro fázový průběh, jen místo zesílení pracujeme s časovým posunem.

$$\varphi_{\mathrm{G}_{(\mathrm{w})}} = \varphi_{1_{(\mathrm{w})}} - \varphi_{2_{(\mathrm{w})}} - \varphi_{3_{(\mathrm{w})}}$$

$\varphi_{\mathrm{G}_{(\mathrm{w})}}$ ... Výsledný fázový posun filtru (závislý na kmitočtu) [°]
$\varphi_{1_{(\mathrm{w})}}$ ... Fázový posun dle prvního měření (závislý na kmitočtu) [°]

$\varphi_{2_{(\mathrm{w})}}$ ... Fázový posun dle druhého měření (závislý na kmitočtu) [°]
$\varphi_{3_{(\mathrm{w})}}$ ... Fázový posun dle třetího měření (závislý na kmitočtu) [°]

Na základě tohoto výpočtu je pak jednoduché získat (např. v prostředí programu Excel nebo MATLAB) přenosovou charakteristiku výsledného filtru $\boldsymbol{G}_{(\mathrm{w})}$. Závěrem je na nás, rozhodneme-li se výsledný signál od vstupního odečíst nebo je pro nás výhodnější signály sčítat a posunout fázovou charakteristiku $\varphi_{\mathrm{G}_{(\mathrm{w})}}$ o 180°.

Výše popsaný postup měření a následného výpočtu je univerzální v tom směru, že jej lze aplikovat i v případě, pokud máte k dispozici pouze dvoukanálový analyzér, který neumožňuje volbu externího referenčního signálu - získává referenční signál interně přímo z výstupu. Pokud nás však toto omezení netrápí, měření je možné zjednodušit tím, že při měření jako referenční signál použijeme výstup vnějšího mikrofonu a tím efektivně eliminujeme vliv přenosu tohoto mikrofonu na toto měření. Při výsledném výpočtu pak stačí odečíst přenos reproduktoru od pasivního útlumu.

2.3 Ověření funkčnosti

Pro ověření efektivity výsledného systému je nejvhodnější dvakrát opakovat měření útlumu jako při prvním měření, jednou bez zapnutého ANC systému a jednou se zapnutým ANC.

Výsledek je opět kmitočtově závislý:

$$A_{\mathrm{ANC}_{(\mathrm{w})}} = A_{\mathrm{active}_{(\mathrm{w})}} - A_{\mathrm{passive}_{(\mathrm{w})}}$$

$A_{\mathrm{ANC}_{(\mathrm{w})}}$ ... Rozdíl potlačení hluku (závislý na kmitočtu) [dB]
$A_{\mathrm{active}_{(\mathrm{w})}}$ ... Aktivní potlačení hluku (závislé na kmitočtu) [dB]
$A_{\mathrm{passive}_{(\mathrm{w})}}$ ... Pasivní potlačení hluku (závislé na kmitočtu) [dB]

2.4 Monitor mód

Zajímavou doplňkovou funkcí, kterou lze realizovat pomocí dopředného systému, je tzv. monitor mód [26], kdy je signál z referenčního mikrofonu bez ANC filtrace přiveden přímo do reproduktoru, odkud je vyzářen přímo do lidského ucha a signál přicházející z hudebního vstupu je buď zeslaben nebo úplně odpojen. Touto cestou lze dosáhnout zesílení okolního zvuku tak, aby bylo i přes nasazená sluchátka dobře rozumět, například při rozhovoru.

Přenos okolního zvuku do vnitřního prostoru sluchátka při aktivním monitor módu je zobrazen v sekci 12.3.3.

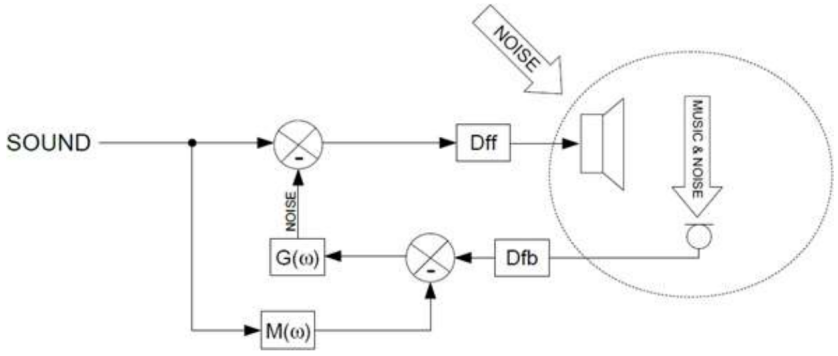
3 Zpětnovazební systém

Druhou topologií, běžně používanou při konstrukci ANC systémů je topologie zpětnovazební (feedback). Při její implementaci jsou použity stejné stavební bloky, ale narozdíl od dopředné topologie je referenční mikrofon umístěn uvnitř sluchátka, ve vnitřní akustické dutině, tvořené z jedné strany náušníkem a z druhé strany hlavou posluchače.

Umístění mikrofonu uvnitř sluchátka je řešením největší nevýhody dopředné topologie, a to náchylnosti na hluk v důsledku působení větru. Zároveň je tento systém méně citlivý na změnu tvaru hlavy posluchače, protože dokáže do určité míry kompenzovat mírně netěsnící náušníky díky zavedené zpětné vazbě [28].

Jedním z hlavních rozdílů mezi dopřednou a zpětnovazební topologií je výsledné kmitočtové pásmo, v němž je nejúčinnější. Zpětnovazební systém může být daleko efektivnější na nízkých kmitočtech (dle [28] typicky pod 100 Hz) než dopředný systém - ale většinou do jisté míry funguje až k 1 kHz. Výsledný útlum (mimo zmíněné nízké kmitočty) je sice mírnější, ale zato rovnoměrně rozložen. Oproti tomu dopředný systém vykazuje ve špičkách daleko silnější útlum, ale pouze v některých částech spektra - jeho distribuce napříč celým kmitočtovým spektrem má charakter hřebenového filtru [28].

3.1 Blokový diagram



Obr. 3.1: Blokový diagram zpětnovazebního ANC systému (převzato z [28])

Z blokového diagramu je patrné, že zdroj zvuku je přiveden přímo do reproduktoru. Referenční mikrofon a reproduktor se nacházejí ve stejném uzavřeném prostoru, mikrofon tudíž mimo hluku snímá i zvuk z reproduktoru - což se někdy může hodit, ale zároveň je tato skutečnost největší nevýhodou zpětnovazební topologie. Bohužel,

ANC mikrofon nerozlišuje mezi signálem žádaným a signálem hlukovým. Mix obou signálů je tedy nasnímán a následně zpracován filtrem $\boldsymbol{G}_{(\mathrm{w})}$. Protože se do zpětnovazební smyčky dostává hluková i hudební složka, ANC systém se pokusí neutralizovat obě. Dle [28] se jedná o běžný jev a mluvíme o tomto jevu jako o ztrátě hlubokých frekvencí v rámci zpětnovazebních ANC systémů.

Tuto ztrátu úrovně v oblasti cca 20 Hz až 1 kHz je třeba nějakým způsobem kompenzovat, to lze v praxi realizovat dvěma způsoby, z nichž každý má své výhody a nevýhody.

Jednou z možností (viz. blokový diagram) je odečíst vstupní hudební signál od signálu z chybového mikrofonu. Před odečtením je na vstupní hudební signál ještě aplikován filtr, který aproximuje charakteristiku přenosu ze vstupu reproduktoru do výstupu z mikrofonu, aby byl odečet hudebního signálu efektivnější. V ideálním případě tak zůstane ve zpětnovazební cestě pouze chybový signál (zbytkový hluk).

Na papíře toto řešení vypadá celkem jednoduše, ale dle [28] není jeho implementace vůbec jednoduchá, proto se v praxi více používá možnost druhá. Tou je jednoduchý ekvalizér na vstupu hudebního signálu, který předem zesílí tu část kmitočtového spektra, která bude následně odečtena. Tato implementace je sice jednodušší, ale připraví celý systém o ne zcela zanedbatelnou část dynamické rezervy (tzv. headroom), což není vždy žádoucí.

3.2 Návrh filtru

Abychom určili přenosovou charakteristiku ANC filtru zpětnovazebního systému, stačí změřit charakteristiku otevřené smyčky reproduktor-mikrofon. Neměli bychom u toho zapomenout rozpojit signálovou cestu mezi mikrofonem a reproduktorem, aby nám už při měření nevznikala nežádoucí zpětná vazba.

Pro měření si vystačíme s dvoukanálovým FFT analyzérem a generátorem měřícího signálu. Výstup generátoru přivedeme přímo na vstup zabudovaného reproduktoru a měříme přímo výstup z vestavěného mikrofonu.

Výslednou cílovou amplitudovou charakteristiku zpětnovazebního filtru pak již získáme pouhou inverzí naměřené charakteristiky (hledáme inverzní filtr).

$$A_{\mathrm{G}_{(\mathrm{w})}} = A_{1_{(\mathrm{w})}} * (-1)$$

$A_{\mathrm{G}_{(\mathrm{w})}}$ ... Výsledné zesílení filtru (závislé na kmitočtu) [dB]
$A_{1_{(\mathrm{w})}}$ ... Zesílení dle prvního měření (závislé na kmitočtu) [dB]

Obdobně lze postupovat i při výpočtu fázové charakteristiky tohoto filtru:

$$\varphi_{\mathrm{G}_{(\mathrm{w})}} = \varphi_{1_{(\mathrm{w})}} * (-1)$$

$\varphi_{\mathrm{G}_{(\mathrm{w})}}$ ... Výsledný fázový posun filtru (závislý na kmitočtu) [°]
$\varphi_{1_{(\mathrm{w})}}$ ... Fázový posun dle prvního měření (závislý na kmitočtu) [°]

3.2.1 Odolnost vůči zpětné vazbě

Již z principu se jedná o zpětnovazební systém, proto je kritické zajistit, aby systém nikdy nezačal oscilovat. Toho lze docílit pouze tak, že pomocí zpětnovazebního filtru vhodně upravíme nejen amplitudové spektrum, ale hlavně to fázové. Ideální je, pokud se neodchýlíme o více než 60° od vypočteného přenosu. V některých kmitočtových oblastech, zvlášť v oblasti vyšších kmitočtů, však může být tento úkol náročný, často až nemožný, proto je důležité v těchto oblastech zamezit oscilaci zavedením co nejsilnějšího útlumu.

3.3 Ověření funkčnosti

Efektivitu výsledného systému lze ověřit stejným způsobem, jako u dopředného systému.

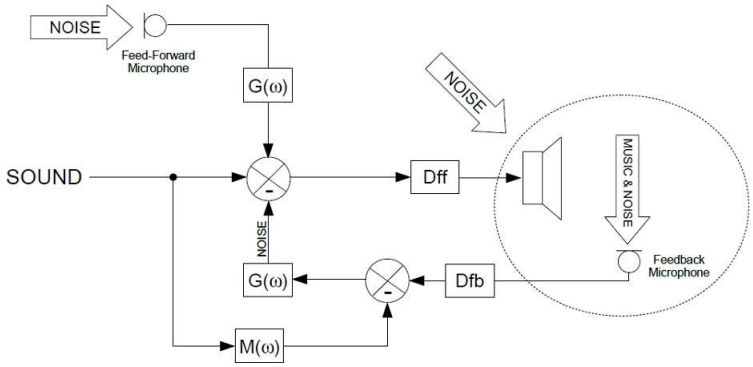
$$A_{\mathrm{ANC}_{(\mathrm{w})}} = A_{\mathrm{active}_{(\mathrm{w})}} - A_{\mathrm{passive}_{(\mathrm{w})}}$$

$A_{\mathrm{ANC}_{(\mathrm{w})}}$ ... Rozdíl potlačení hluku (závislý na kmitočtu) [dB]
$A_{\mathrm{active}_{(\mathrm{w})}}$ ... Aktivní potlačení hluku (závislé na kmitočtu) [dB]
$A_{\mathrm{passive}_{(\mathrm{w})}}$ ... Pasivní potlačení hluku (závislé na kmitočtu) [dB]

4 Hybridní systém

Jestliže dopředná a zpětnovazební topologie přicházely každá s novým, unikátním konceptem, hybridní topologie nedělá nic jiného, než že kombinuje obě topologie předchozí do jednoho hybridního systému.

4.1 Blokový diagram



Obr. 4.1: Blokový diagram hybridního systému (převzato z [28])

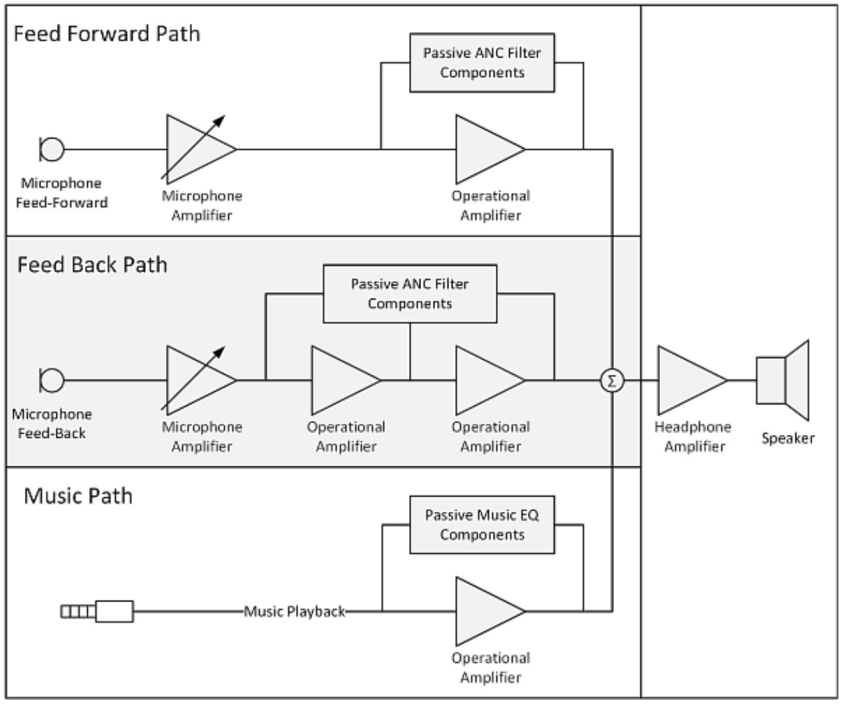
Dohromady oba systémy tvoří jeden celek, kde jeden systém kompenzuje nedostatky druhého a naopak. Touto výhodnou kombinací tak je možné docílit rovnoměrného a silného útlumu od nízkých kmitočtů až do cca 3 kHz [28]. Nevýhodou pak je vyšší cena, protože je potřeba dvakrát tolik součástek.

5 ANC s využitím DSP

Historicky byla ANC technologie určena hlavně pro profesionální užití, například v leteckém průmyslu. Jako taková byla konstruována na zakázku, většinou pomocí diskrétních analogových obvodů, často dokonce jako externí dedikované zařízení [31].

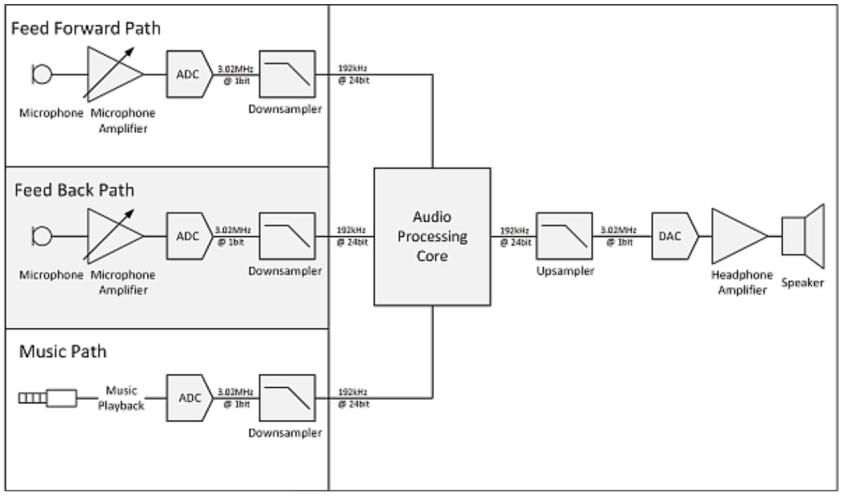
S nástupem nových technologií se postupně technoloige ANC stala dostupnější i pro běžné, konzumní uživatele a výrobci těchto technologií začali řešit nové problémy - technologii ANC bylo třeba vyrábět ve větším množství, sériově a ideálně ji zabudovat přímo do sluchátek tak, aby posluchači nijak nepřekážela při jejím užívání v každodenním koloběhu. Je proto logické, že konstruktéři začali uvažovat o využití výhod, které přináší nově nastupující digitální technologie.

Pokud chceme využít stávajících analogových znalostí konstrukce ANC sluchátek, je ideální vycházet z blokového diagramu analogového hybridního zapojení:



Obr. 5.1: Blokový diagram analogového hybridního systému (převzato z [31])

Takové zapojení lze snadno transformovat vložením vhodných A/D a D/A převodníků a DSP čipu.

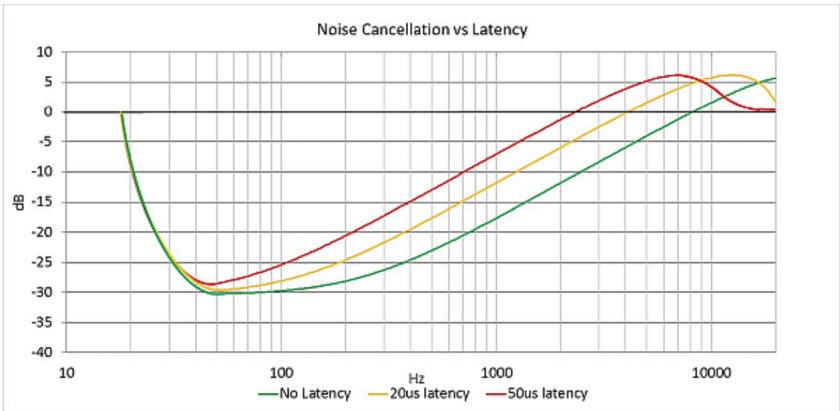


Obr. 5.2: Blokový diagram digitálního hybridního systému (převzato z [31])

5.1 Porovnání analogového a číslicového řešení

Jak už to bývá, použití digitální technologie s sebou přináší výhody i nevýhody, proto nemusí být digitální řešení vždy to nejlepší - pojďme se podívat na některá pro a proti, která s sebou tato technologie přináší.

Základní limitací každého číslicového procesu je fakt, že často potřebuje pro dokončení dané operace (např. A/D nebo D/A převodu) několik vzorků najednou - tím vnáší do systému časové zpoždění, tzv. latenci. Toto zpoždění pro nás vlastně znamená kmitočtově závislý fázový posun, který pak hraje kritickou roli při konstrukci ANC filtru a snižuje jeho účinnost (tento jev je popsán v sekci 11.1.2). Jako vedlejší produkt, zvlášť u zpětnovazebních systémů, tato přidaná latence také zvyšuje riziko nestability systému. Dle [31] se latence ve výsledku může projevit zhoršením účinku až o 10 dB při latenci 50 μs.



Obr. 5.3: Vliv latence na efektivitu systému (převzato z [31])

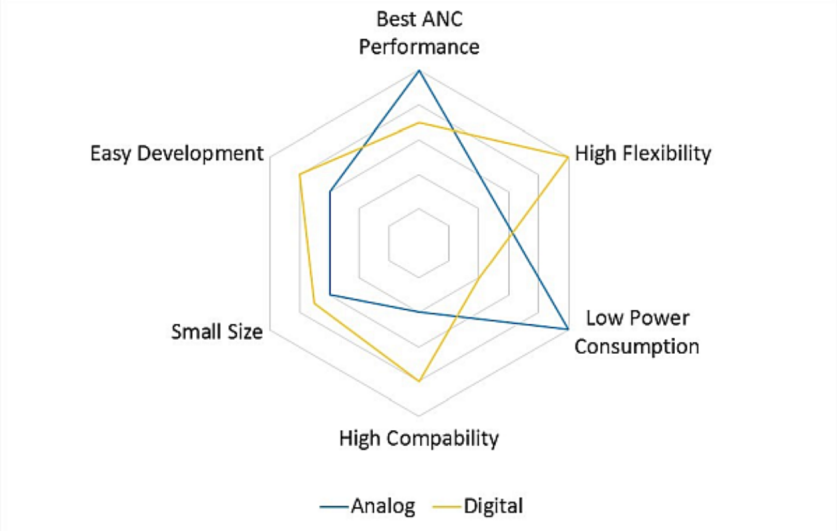
Dalším důležitým faktorem jsou i zvýšené nároky na napájení celého systému a tím snížení provozní doby zařízení, které je napájeno z baterií. Dle [31] lze u modelového analogového headsetu očekávat příkon cca 30 mW a u digitálního v závislosti na vzorkovací frekvenci cca $50 - 120$ mW. Zvýšený odběr (a s ním spojená kratší výdrž zařízení) však nutně nemusí znamenat, že si zákazník takové zařízení nezakoupí, protože většina lidí je dnes zvyklá nabíjet svá osobní zařízení i několikrát denně.

5.2 Potenciál číslicového signálového zpracování

Oproti tomu přináší signálové zpracování pomocí DSP čipu nové možnosti predikce, adaptace a optimalizace - pokud nalezneme vhodný kompromis mezi efektivitou daného algoritmu, cenou a snesitelným příkonem, můžeme naplno ocenit výhody, které s sebou integrace číslicového systému přináší.

S realizací ANC pomocí číslicových filtrů totiž odpadá při vývoji potřeba upravovat desku plošných spojů, nechávat si volné místo pro záložní komponenty atd. Digitální zařízení umožňuje vývoj ANC filtrů čistě pomocí aktualizace programového řešení a tím značně ulehčuje výrobu i vývoj daného zařízení - je možné masově vyrábět standardizovaný model a veškeré úpravy a adaptace obvodů pak již aplikovat v digitální doméně - pozorovat a optimalizovat efektivitu filtrů v reálném čase a měnit jejich parametry pouhým kliknutím. Díky digitálním systémům je dokonce možné tuto optimalizaci výsledných filtrů plně automatizovat.

[31] nabízí zajímavé porovnání možností analogové a digitální konstrukce a v několika kategoriích je hodnotí dle následujícího diagramu:



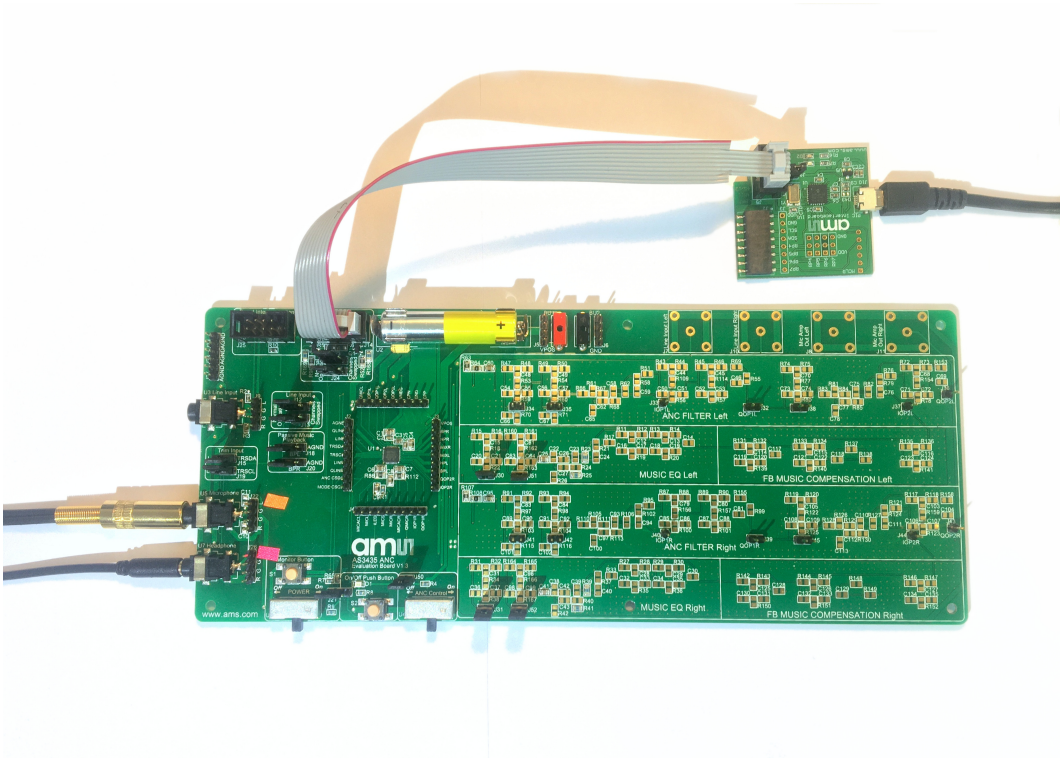
Obr. 5.4: Porovnání možností analogové a digitální konstrukce ANC zařízení (převzato z [31])

6 Analogový HW pro vývoj ANC systémů

6.0.1 Výběr platformy

Aby bylo možné vyzkoušet a otestovat různé možnosti realizace ANC systémů, bylo třeba zvolit platformu, vhodnou pro stavbu prototypu. Již ze začátku bylo zavrženo čistě diskrétní řešení, které by bylo zbytečně drahé, rozměrné a výsledek by byl značně nejistý. Při rozhodování hrála nemalou roli tato kritéria:

- možnost napájení z běžně dostupné baterie (např. AA, AAA, 6LR61)
- velikost, vhodná k zabudování teoretického finálního produktu do vnitřní dutiny sluchátka
- dostatečná flexibilita, umožňující nenáročnou změnu ANC topologie a jednotlivých ANC filtrů
- přítomnost mikrofonního předzesilovače, jehož zesílení je možné diskrétně ovládat
- integrovaný sluchátkový zesilovač
- možnost přemostění obvodu při nedostatečném (nebo odpojeném) napájecím napětí



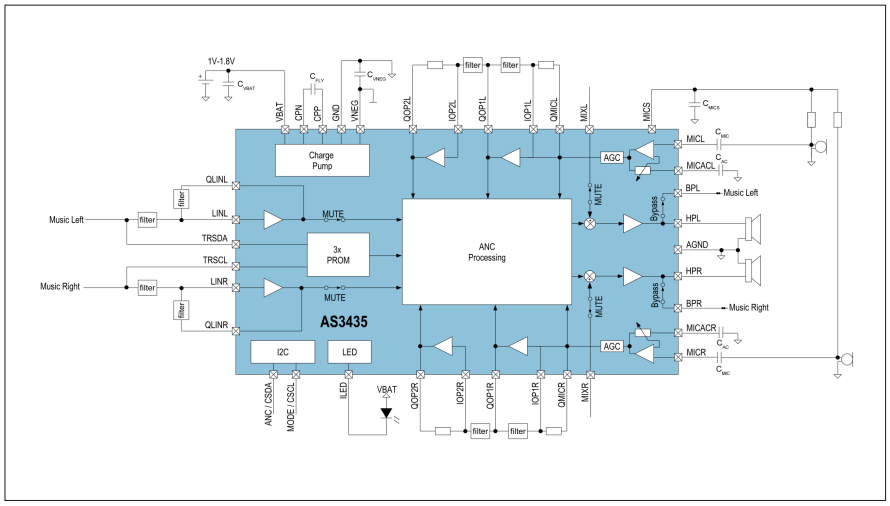
Obr. 6.1: Vývojová deska ams A3435 spolu s USB → I2C převodníkem

6.1 ams AS3435 - integrovaný předzesilovač

Pro realizaci byl vybrán produkt z dílny rakouské firmy ams, která se specializuje na výrobou integrovaných obvodů pro konstrukci senzorů různých veličin. Divize ams, jenž se věnuje vývoji audio senzorů, nabízí produkt s obchodním označením AS3435, který splňuje většinu uvedených požadavků.

Jedná se o soubor několika integrovaných obvodů umístěných v jednom pouzdře o velikosti cca 5x5 mm, což je i po doplnění o malé množství nutných diskrétních součástek rozměr vhodný pro zabudovaní i do malých in-ear sluchátek, natož pak do realizovaného over-ear designu.

Bloková struktura tohoto obvodu vypadá následovně:



Obr. 6.2: Bloková struktura AS3435 (převzato z [26])

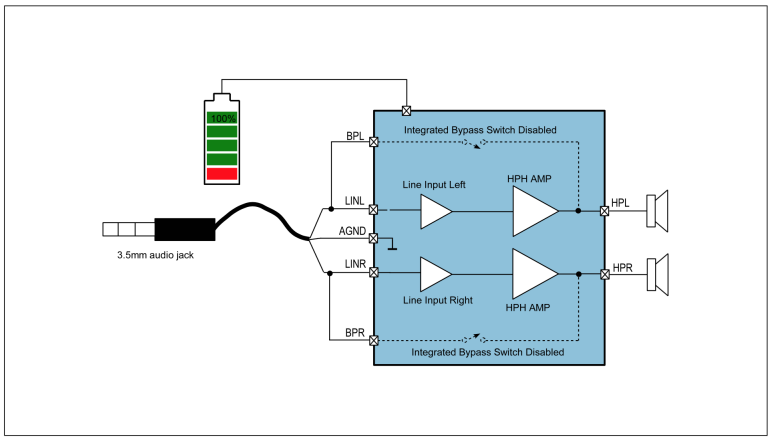
Obvod se skládá z prvků, nacházejících se v přímé signálové cestě - mikrofonní předzesilovač, vstupní linkový předzesilovač, několik operačních zesilovačů pro univerzální použití (primárně určených pro realizaci analogových ANC filtrů), přepínací matice pro výběr odpovídajícího zdroje ANC signálu a sluchátkový zesilovač.

Tyto „primární“ obvody jsou doplněny dalšími, pomocnými obvody, zajišťujícími různé doplňkové funkce, jako například bypass obvodu při nedostatečném napájení, napájecí zdroj včetně několika DC/DC převodníků, díky kterým může i obvod napájený jednou AAA baterií využívat výhod symetrického napájení a samozřejmě i zdroj napájení pro impedanční převodník, nacházející se v pouzdru elektretových mikrofonů.

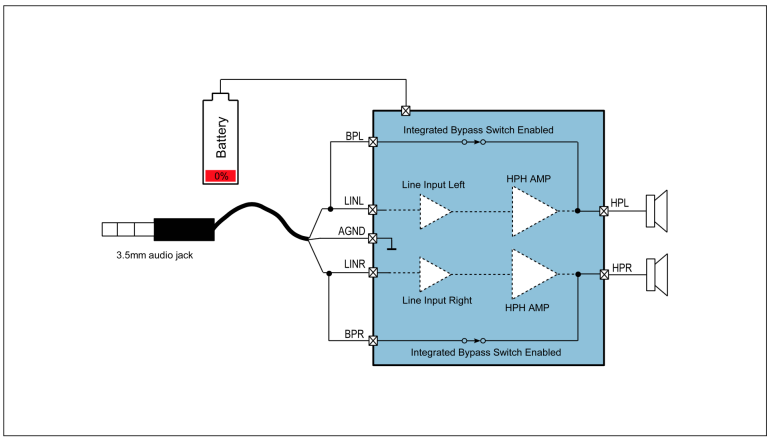
Veškeré parametry obvodu AS3435 lze samozřejmě dohledat v [26], ale přesto je vhodné se zmínit o funkci některých stavebních bloků.

6.1.1 Integrovaný bypass

Jedním z hlavních požadavků na ANC headset je, aby byl použitelný i jako běžná pasivní sluchátka, a to i ve chvíli, kdy není k dispozici vhodný zdroj napájení. Jako vhodné řešení se nabízí elektricky řízený spínač, jenž je v základním stavu sepnut a ve chvíli, kdy je přítomné potřebné napětí se spínač rozpojí. Tento spínač je umístěn mezi linkový vstup obvodu a výstup sluchátkového zesilovače tak, aby se spínač ve chvíli, kdy je vyjmuta (nebo je vybitá) baterie, sepnul a tím přemostil veškeré aktivní prvky.



Obr. 6.3: Blokový diagram integrované funkce bypass v rozepnutém stavu (převzato z [26])



Obr. 6.4: Blokový diagram integrované funkce bypass v sepnutém stavu (převzato z [26])

6.1.2 Zdroj napájení pro mikrofon

Vzhledem k tomu, že jedna baterie AAA je schopná pracovat v rozsahu napětí cca 1 až 1,6 V, není vhodným zdrojem pro napájení elektretového mikrofonu - ty jsou běžně napájeny napětím cca 1,3 až 3 V, nevadí však ani napětí o něco vyšší. Proto obsahuje AS3435 integrovaný DC/DC převodník, který generuje napětí vyšší, typicky 2,7 V. Má to však jednu nevýhodu, tento typ zdroje je schopen dle [26] dodat proud maximálně 300 až 500 mA v závislosti na úrovni napájecího napětí. Tato limitace tak značně omezuje výběr vhodného mikrofonu a celkový počet těhto mikrofonů použitelných s tímto zapojením najednou.

Zdroj napájení pro mikrofony lze pomocí diskrétního ovládání zcela vypnout nebo je možné v případě napájení celého obvodu externím zdrojem napájet mikrofony přímo z tohoto externího zdroje - [26] to však doporučuje pouze tehdy, je-li napětí externího zdroje vyšší než 1,8 V.

Realita je bohužel taková, že i když je myšlenka integrovaného mikrofonního napaječe příjemná, pro napájení mikrofonů zvolených pro tuto práci není takový zdroj dostatečný. Proto bylo zapojení doplňeno o jednoduchý diskrétní předzesilovač, který vhodné napájení pro použité mikrofony zajistí - viz. následující sekce 6.2.

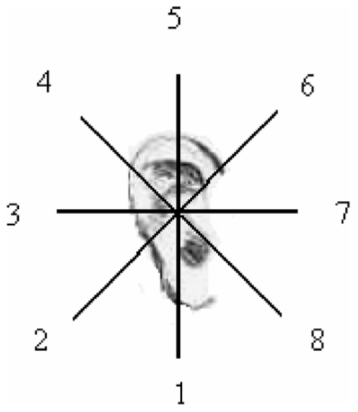
6.1.3 Nastavení zesílení vstupních mikrofonních předzesilovačů

Vzhledem k tomu, že všechny běžně dostupné mikrofony jsou vyráběny s určitou tolerancí (citlivost se může dle [28] lišit o ± 3 dB), je vhodné (a u analogového řešení až kritické) mít možnost precizně nastavovat zesílení předzesilovačů. Plynulé nastavení pomocí rezistoru a potenciometru není příliš vhodné, protože často není replikovatelné a opakovatelné a hlavně je náchylné na lidskou chybu. Navíc by pro nastavení potenciometru při kalibraci mikrofonních zesilovačů bylo třeba akustickou dutinu sluchátka otevřít, provést změnu nastavení potenciometru, dutinu následně opět uzavřít a ověřit vliv této změny. Není to úplně efektivní způsob práce, proto je zde změna zesílení řešena trochu jiným způsobem.

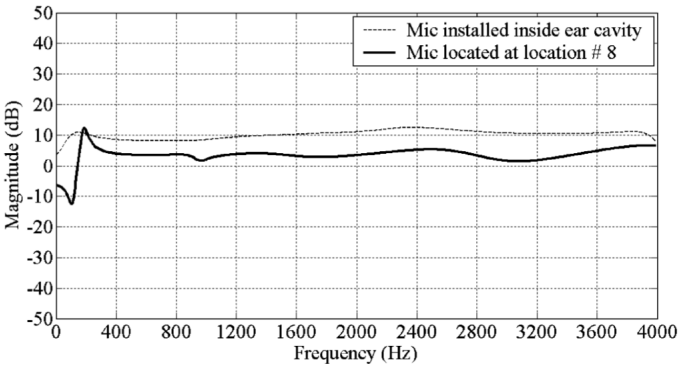
Každý kanál obsahuje množství integrovaných dvojic rezistorů, jenž určují zesílení vstupního předzesilovače, realizovaného pomocí neinvertujícího zesilovače s OZ. Výběr vhodné dvojice je ovládán diskrétně pomocí datových registrů, je možné nastavit celkem 63 různých úrovní zesílení - od 0 dB do 31 dB s krokem 0,5 dB. Diskrétní ovládání tohoto zesílení s sebou tedy přináší jednu obrovskou výhodu, zesílení je možné při kalibraci upravit bez otevření akustické dutiny sluchátka - tedy bez změny akustických podmínek. Zároveň umožňuje při komerční výrobě použít pro finální kalibraci obvodu automatizované měření.

6.1.4 Určení pozice zpětnovazebního mikrofonu

Pro optimální funkci referenčního mikrofonu je třeba se zamyslet nad jeho umístěním uvnitř sluchátek, protože jeho umístění je hned po správném nastavení předzesilovače druhým nejdůležitějším faktorem ovlivňujícím vstupní signál před následným zpracováním pomocí ANC filtru. Pokud jde o vnější mikrofon, není jeho umístění příliš kritické, protože je fakticky umístěn v relativně objemném poloprostoru. Pokud však jde o mikrofon vnitřní, ten je umístěn uvnitř malé zvukové komory tvořené z jedné strany náušníkem a z druhé strany hlavou posluchače. Vzhledem k nepravidelnému tvaru ušního boltce se při posunutí mikrofonu změní i charakter jím nasnímaného signálu. Podrobně se tímto fenoménem zaobírá literatura [24]. Jako hlavní kritérium při hledání optimální pozice mikrofonu byl zvolen předpoklad, že by se kmitočtová charakteristika signálu zaznamenaného vnitřním mikrofonem měla co nejvíce blížit kmitočtové charakteristice signálu zaznamenaného ušním bubínkem, potažmo měřícím mikrofonem umístěným na konci zvukovodu simulátoru ucha a hlavy - v místě bubínku. Experimentální metodou bylo v literatuře [24] dosaženo zjištění, že nejvýhodnější je pozice číslo 8 - tedy umístění mikrofonu v přední dolní oblasti přímo naproti zvukovodu:



Obr. 6.5: Nákres mikrofonních pozic porovnávaných v [24]

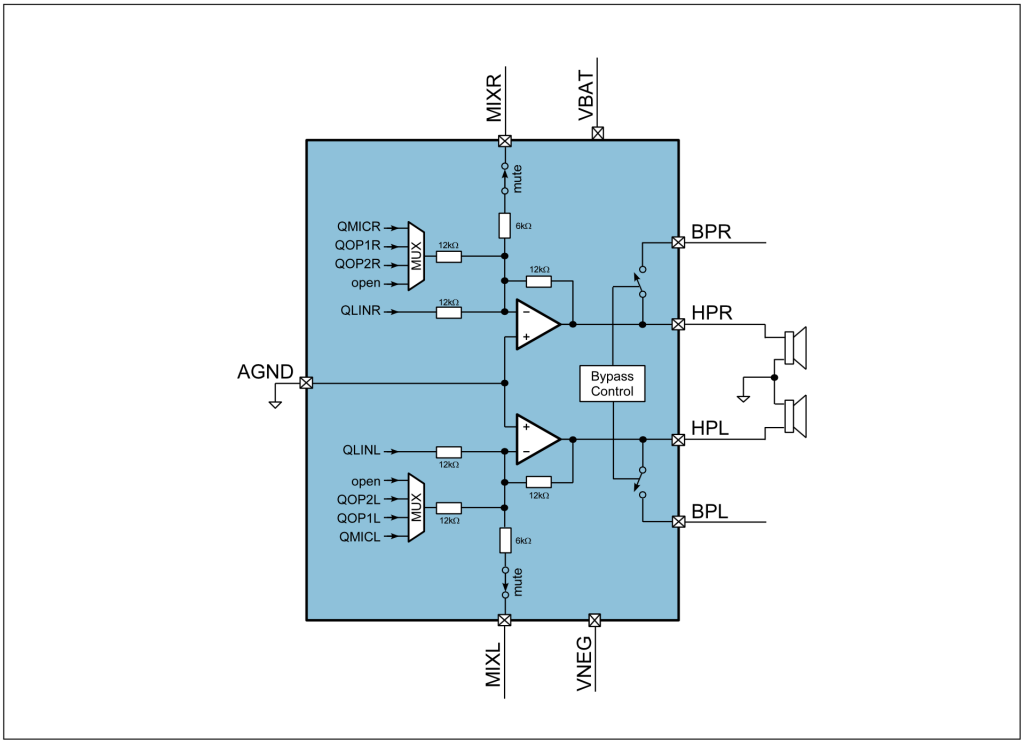


Obr. 6.6: Kmitočtový průběh signálu zaznamenaného mikrofonem uvnitř zvukovodu porovaný s kmitočtovým průběhem signálu zaznamenaného mikrofonem na pozici číslo 8 (převzato z [24])

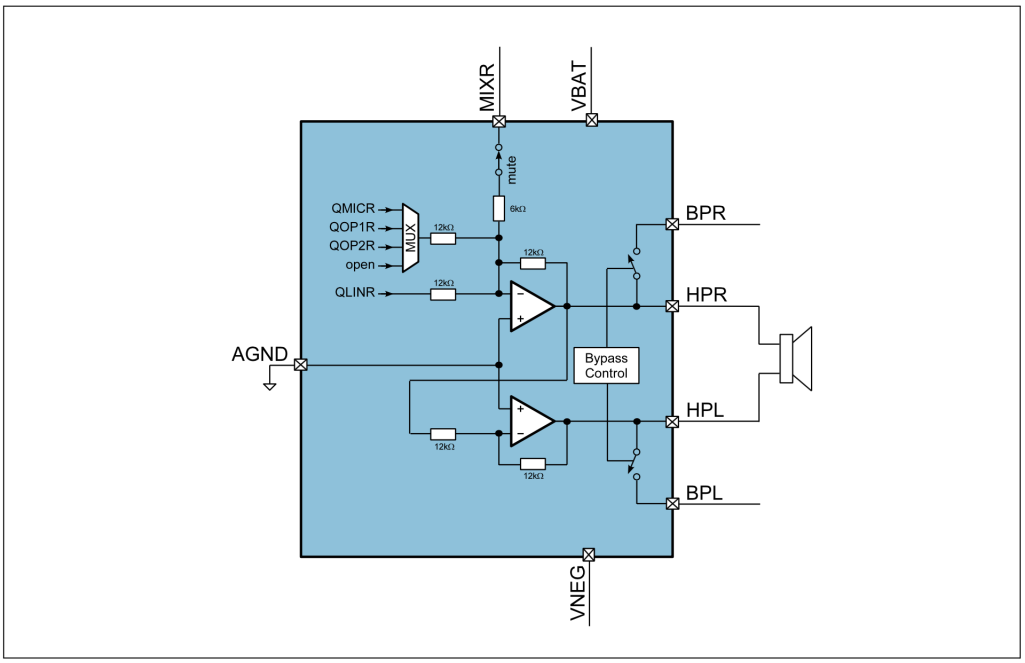
Na základě této studie byla zvolena pozice zpětnovazebního mikrofonu, viz 6.3.2.

6.1.5 Sluchátkový zesilovač

Každý integrovaný obvod AS3435 obsahuje jeden stereo sluchátkový zesilovač. Je tomu tak, aby bylo pomocí jednoho čipu možné realizovat stereo headset s jedním mikrofonem v každém sluchátku. Pro případ, že se designer rozhodne realizovat hybridní zapojení a osadit do jednoho sluchátka mikrofony dva, výrobce umožňuje přepnout sluchátkový zesilovač (opět diskrétně pomocí registru) do diferenčního módu, což má zároveň tu příjemnou vlastnost, že místo výkonu 2x34 mW do 32 Ω je zesilovač zapojený do můstku schopný dodat výkon až 1x120 mW (při napájecím napětí 1,8 V).



Obr. 6.7: Sluchátkový zesilovač v běžném módu (převzato z [26])



Obr. 6.8: Sluchátkový zesilovač v diferenčním módu (převzato z [26])

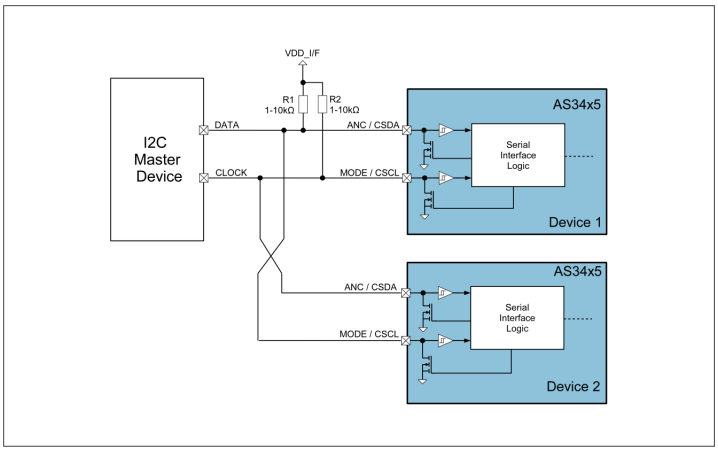
6.1.6 Matice pro výběr zdroje

Pro testování a vývoj je naprosto klíčové mít plnou kontrolu nad tím, jaký signál je přiveden na vstup sluchátkového zesilovače. K tomuto účelu slouží přepínací matice (v blokovém diagramu označena jako MUX), pomocí které lze zvolit jednu ze čtyř možností - signál přímo z mikrofonního předzesilovače, signál z jednoho ze dvou operačních zesilovačů a nebo žádný signál. Volbu v matici lze opět realizovat pomocí změny hodnoty v registru.

Pro náročnější konstrukce je k dispozici vstup MIXL/MIXR, jenž je oddělený od matice MUX, má pevně nastavené vstupní zesílení +6 dB a lze jej pomocí diskrétního řízení pouze připojit nebo odpojit (funkce mute). Jak je zřejmé z blokového diagramu pro diferenciální zapojení zesilovače, separátní vstup je pro hybridní topologii nutný, protože signál z obou mikrofonů je (po filtraci) třeba přivést na vstup pravého kanálu zesilovače.

6.1.7 Rozhraní I2C

Jak již bylo v textu několikrát zmíněno, velká část parametrů integrovaného obvodu je ovládána diskrétně. K tomu lze použít jedno ze dvou rozhraní. To první je vhodné pro použití při návrhu a optimalizaci - jedná se o rozhraní I2C, což je sériové komunikační rozhraní, které pro svůj provoz potřebuje dva vodiče. Jedním z vodičů protéká pouze časovací signál (clock), druhým pak samotná přenášená data. Zajímavou funkcí AS3435 je, že umí automaticky rozpoznat prohozené signály mezi vodiči a na základě toho změnit komunikační adresu tohoto rozhraní. Co to znamená? Že je možné pomocí jednoho I2C rozhraní ovládat dvě zařízení najednou, jedno - to s neprohozenými piny - se pak stane zařízením hlavním (master) a to s prohozenými piny se stane zařízením doplňkovým (slave).



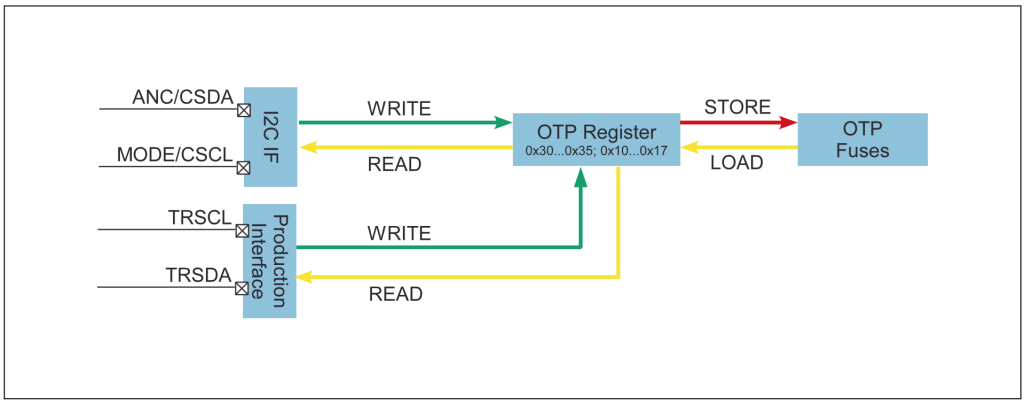
Obr. 6.9: Ovládání dvou zařízení najednou pomocí jednoho I2C rozhraní (převzato z [26])

Druhou, alternativní možností, je pak použití doplňkového nástroje, kterému výrobce říká „production toolbox“, který umožňuje komunikovat s integrovaným obvodem pomocí 3,5 mm jack linkového vstupu. Výhodou tohoto řešení v komerčním prostředí je, že pro přesnou kalibraci a finální nastavení koncového produktu není třeba žádné jiné fyzické rozhraní než to, které je již na místě přítomné pro zvukový vstup.

6.1.8 Paměť OTP a interní registry

Veškeré dostupné diskrétně řízené nastavení je uložené pomocí interních registrů. Tyto registry jsou složeny z volatilní (energeticky závislé) paměti, tzn. při každém vypnutí nebo restartu zařízení se tyto registry resetují do původního nastavení. Pokud je aktivní připojení pomocí I2C rozhraní, získává prioritu a všechno nastavení se zapisuje do registrů pomocí tohoto rozhraní. Je důležité také poznamenat, že data uložená v těchto volatilních registrech je možné neomezeně měnit a přepisovat.

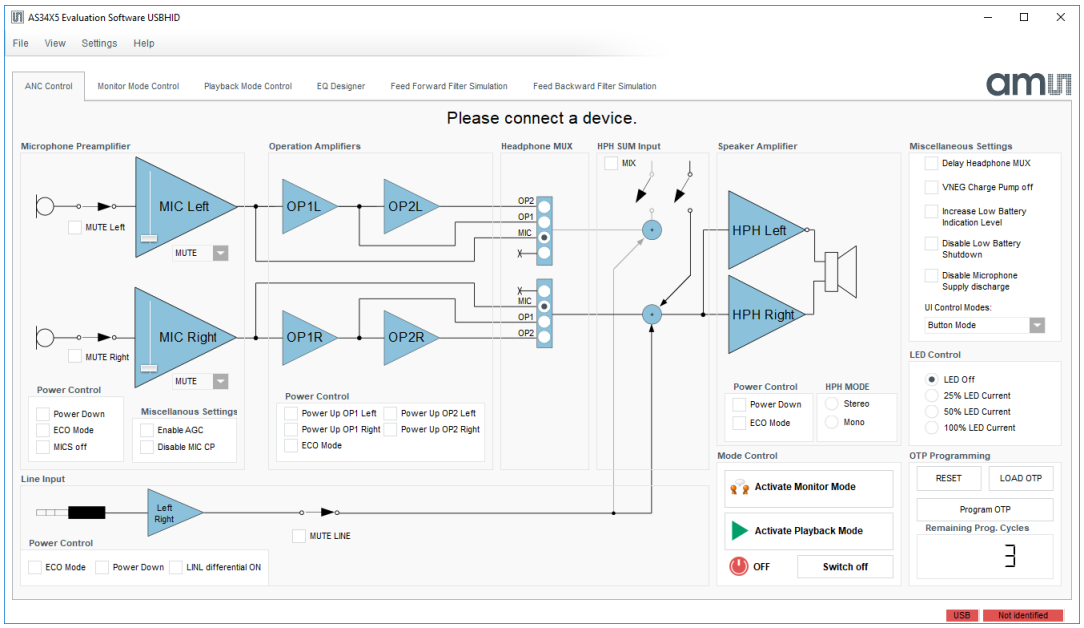
Ve výchozím stavu se vždy po zapnutí čipu vrátí jeho nastavení do původního továrního nastavení. Pokud si přeje designer uložit nějaké nastavení takovým způsobem, aby se po restartu opět vyvolalo, musí k tomu využít část paměti, které se říká „OTP fuses“ (fuse = pojistka). Zde přichází ke slovu konečně zkratka OTP - One Time Programmable, protože jakmile se jednou zapíše nějaká hodnota do OTP paměti, jedná se o nevratný proces. Každý čip má k dispozici tři kolekce těchto OTP bloků, což znamená, že po prvním uložení nastavení jej lze ještě dvakrát přepsat a aktualizovat. Je však důležité mít na paměti, že počet přepsání není nekonečný a tomuto faktu přizpůsobit výrobní a kalibrační procesy.



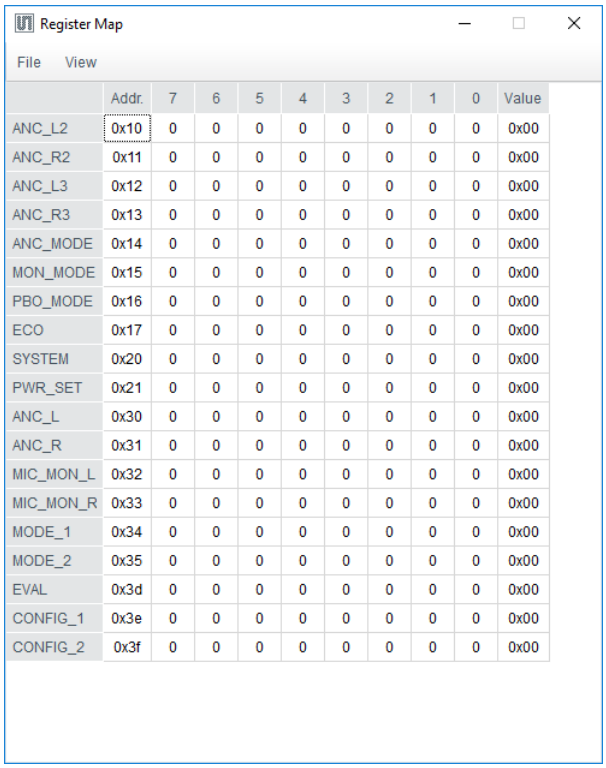
Obr. 6.10: Schéma architektury registrů a OTP paměti (převzato z [26])

6.1.9 GUI pro vývoj

I když je možné implementovat vlastní ovládací zařízení a adresovat dané registry přímo přes I2C, pokud si zakoupíte AS3435 ve verzi Evaluation Board (= vývojová deska), výrobce ams vám k ní dodá i rozhraní USB → I2C a hlavně aplikaci, která umí díky celkem přehlednému GUI (grafické uživatelské rozhraní) zapisovat data přímo do volatilních registrů. Program sice nemá žádný manuál a je určen dle slov výrobce k užívání dle vlastního úsudku - bez záruky, ale zjednodušuje komunikaci s integrovaným obvodem a jeho nastavení.



Obr. 6.11: Ovládací software pro AS3435



	Addr.	7	6	5	4	3	2	1	0	Value
ANC_L2	0x10	0	0	0	0	0	0	0	0	0x00
ANC_R2	0x11	0	0	0	0	0	0	0	0	0x00
ANC_L3	0x12	0	0	0	0	0	0	0	0	0x00
ANC_R3	0x13	0	0	0	0	0	0	0	0	0x00
ANC_MODE	0x14	0	0	0	0	0	0	0	0	0x00
MON_MODE	0x15	0	0	0	0	0	0	0	0	0x00
PBO_MODE	0x16	0	0	0	0	0	0	0	0	0x00
ECO	0x17	0	0	0	0	0	0	0	0	0x00
SYSTEM	0x20	0	0	0	0	0	0	0	0	0x00
PWR_SET	0x21	0	0	0	0	0	0	0	0	0x00
ANC_L	0x30	0	0	0	0	0	0	0	0	0x00
ANC_R	0x31	0	0	0	0	0	0	0	0	0x00
MIC_MON_L	0x32	0	0	0	0	0	0	0	0	0x00
MIC_MON_R	0x33	0	0	0	0	0	0	0	0	0x00
MODE_1	0x34	0	0	0	0	0	0	0	0	0x00
MODE_2	0x35	0	0	0	0	0	0	0	0	0x00
EVAL	0x3d	0	0	0	0	0	0	0	0	0x00
CONFIG_1	0x3e	0	0	0	0	0	0	0	0	0x00
CONFIG_2	0x3f	0	0	0	0	0	0	0	0	0x00

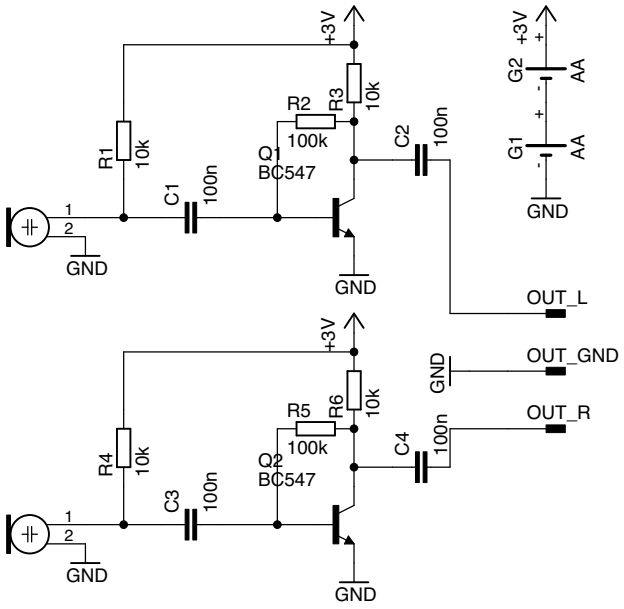
Obr. 6.12: Registry dostupné z ovládacího softwaru

6.2 Dodatečný diskrétní předzesilovač

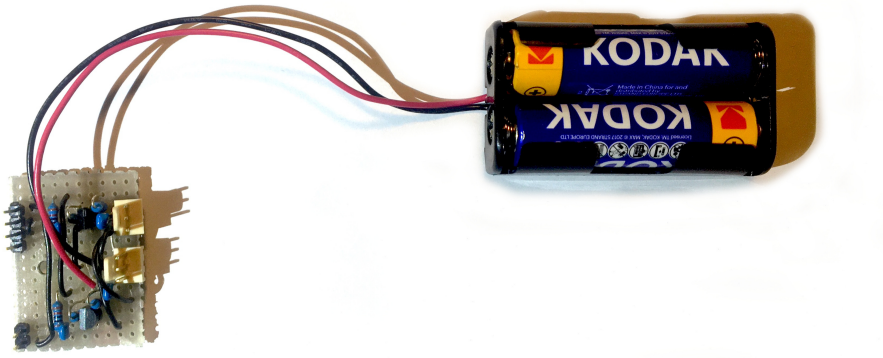
Jak již byl v předchozím textu zmíněno, ač by bylo příjemné jej využít, integrovaný zdroj pro napájení elektretových mikrofonů se ukázal jako nedostatečný a nebyl schopen konstantně poskytovat dostatečný proud potřebný pro provoz mikrofonů.

Z toho důvodu byl navržen a zkonstruován dodatečný diskrétní mikrofonní předzesilovač s integrovaným napájením pro elektretové mikrofony, tentokrát však takový, aby byl schopný dodat potřebný napájecí proud.

Jako první, nejjednodušší řešení, se nabízelo jednoduché zapojení pomocí NPN tranzistoru, napájené dvěma AA články, celkem tedy napětím cca 3 V.



Obr. 6.13: Diskrétní předzesilovač s NPN tranzistorem - schéma

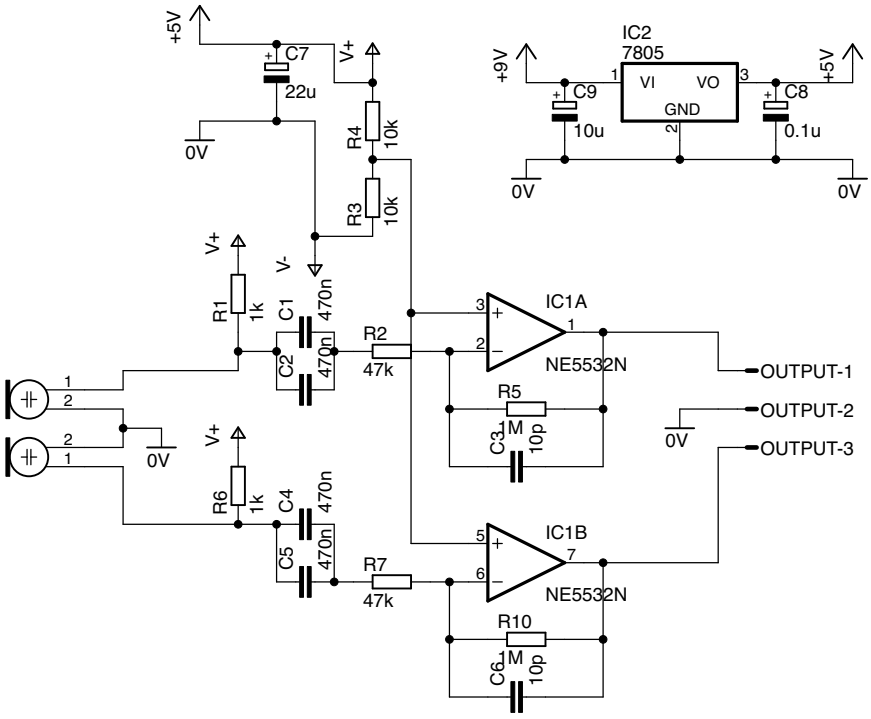


Obr. 6.14: Diskrétní předzesilovač s NPN tranzistorem

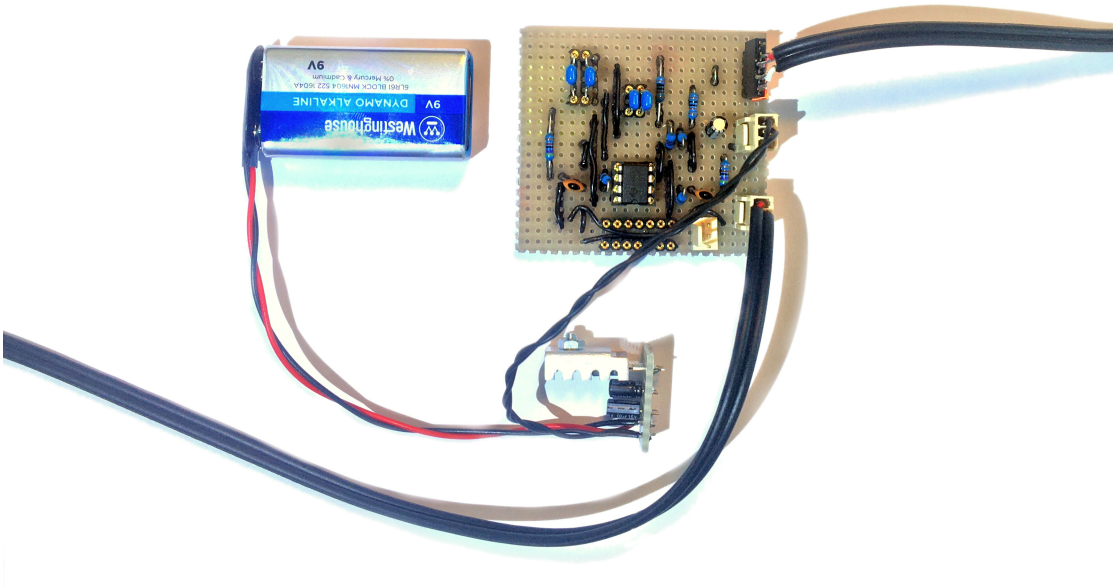
Toto zapojení fungovalo dostatečně, co se týče patřičného zesílení a napájení mikrofonů, ale bohužel mělo jeden velký neduh - kmitočtově nevyrovnaný přenos.

Původním záměrem bylo kompenzovat tuto skutečnost později pomocí digitálního filtru, ale ukázalo se, že je to náročné a nevhodné.

Bylo tedy třeba hledat nové řešení, tentokrát s důrazem na pokud možno lineární kmitočtový přenos. Řešením, které se samo nabízelo, se tedy stal předzesilovač s využitím operačního zesilovače, který díky zavedené zpětné vazbě dokáže z velké části svůj přenos linearizovat.



Obr. 6.15: Diskrétní předzesilovač s OZ - schéma



Obr. 6.16: Diskrétní předzesilovač s OZ

Pro lepší funkci OZ bylo vhodné zvýšit napájecí napětí, ne však zbytečně příliš vysoké, aby jím bylo možné přímo napájet elektretové mikrofony. Pro napájení obvodu je vhodné použít běžně dostupnou baterii, zvolena byla standardní 9 V baterie připojená pomocí lineárního regulátoru napětí 7805 pro stálé napájecí napětí 5 V.

6.3 Sluchátka osazená mikrofony

Pro následnou realizaci prototypu sluchátek s ANC bylo možné postupovat dvěma různými cestami. Buď zakoupit sluchátka s již zabudovaným ANC obvodem a modifikovat jeho zapojení tak, aby bylo možné doplnit vlastní algoritmus pro potlačení okolního šumu, a nebo zakoupit standardní pasivní sluchátka a ta následně doplnit o dva mikrofony.

Jelikož koupě sluchátek s již zabudovaným ANC a jejich následná modifikace by byla zbytečně nákladná a daná sluchátka nebo alespoň zabudovaný ANC obvod by u toho byl pravděpodobně zničen, druhá možnost je pro tento projekt výhodnější - koupit vhodná pasivní sluchátka a šetrně je doplnit o dva mikrofony.

6.3.1 Parametry pro výběr sluchátek

Během výběru sluchátek bylo třeba vzít do úvahy následující požadavky:

- Sluchátka musí být „on-ear“ konstrukce, jejich náušníky tedy musí kompletně obepínat ušní boltec
- Sluchátka by měla mít díky své konstrukci co nejlepší pasivní útlum
- Zároveň by měl zvuk přehrávaný do sluchátek co nejméně unikat do okolí
- Sluchátka musí dobře a pevně sedět na hlavě
- Sluchátka musí být možné nedestruktivně doplnit o dva mikrofony
- Sluchátka by měla mít pokud možno co nejvyrovnanější přenos
- Sluchátka by měla být schopná věrně reprodukovat i velmi nízké kmitočty, aby byla schopná přehrát odečítací signál i v nižších částech kmitočtového spektra

Těmto kritériím výborně odpovídají sluchátka Audio-Technika ATH-M50x, která navíc nejsou přehnaně drahá, jsou tedy relativně dostupná a poskytují vhodnou platformu pro experimentální využití.

Obr. 6.17: Sluchátka ATH-M50x, vnější mikrofon

6.3.2 Osazení sluchátek mikrofony

Součástí těchto sluchátek je navíc drobná dutina pod vnějším povrchem sluchátka, které je oddělena od vnitřní dutiny za reproduktorem. Tato malá dutina je ideálním prostorem, do kterého lze umístit vnější mikrofon (viz foto). Takovým umístěním mikrofonu není nijak narušena akustická konstrukce sluchátek a ta si tak zachovávají své původní vlastnosti i po instalaci tohoto mikrofonu.

Vnitřní mikrofon byl umístěn do prostoru sluchátka přímo pod ochranný akustický molitan (viz foto). Pozice mikrofonu vzhledem k ušnímu boltci byla zvolena dle doporučení [24] přímo naproti vstupu do zvukovodu. Tato pozice zároveň vyhovuje i ergonomicky, protože se mikrofon při správném nasazení sluchátek nedotýká žádné části ušního boltce a nesnižuje tak schopnost sluchátka pasivně tlumit okolní šum.

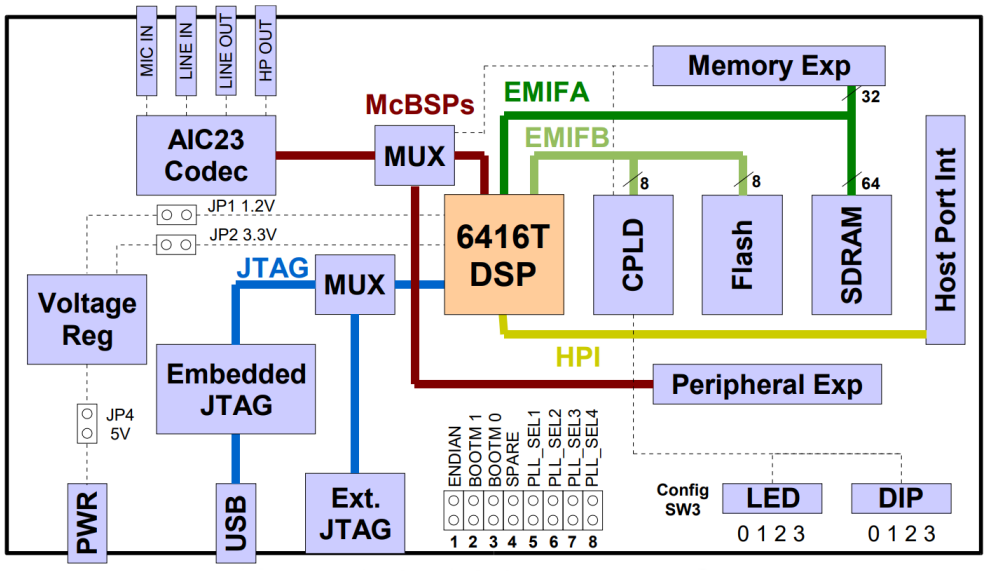
Obr. 6.18: Sluchátka ATH-M50x, vnitřní mikrofon

7 Vývojová deska s DSP čipem

Vývojová deska C6416T DSK (DSK = DSP Starter Kit) je cenově dostupná vývojová platforma pro vývoj aplikací na procesorech řady C64xx firmy Texas Instruments. Její základní stavební bloky a doplňkové integrované obvody popisuje blokový diagram.

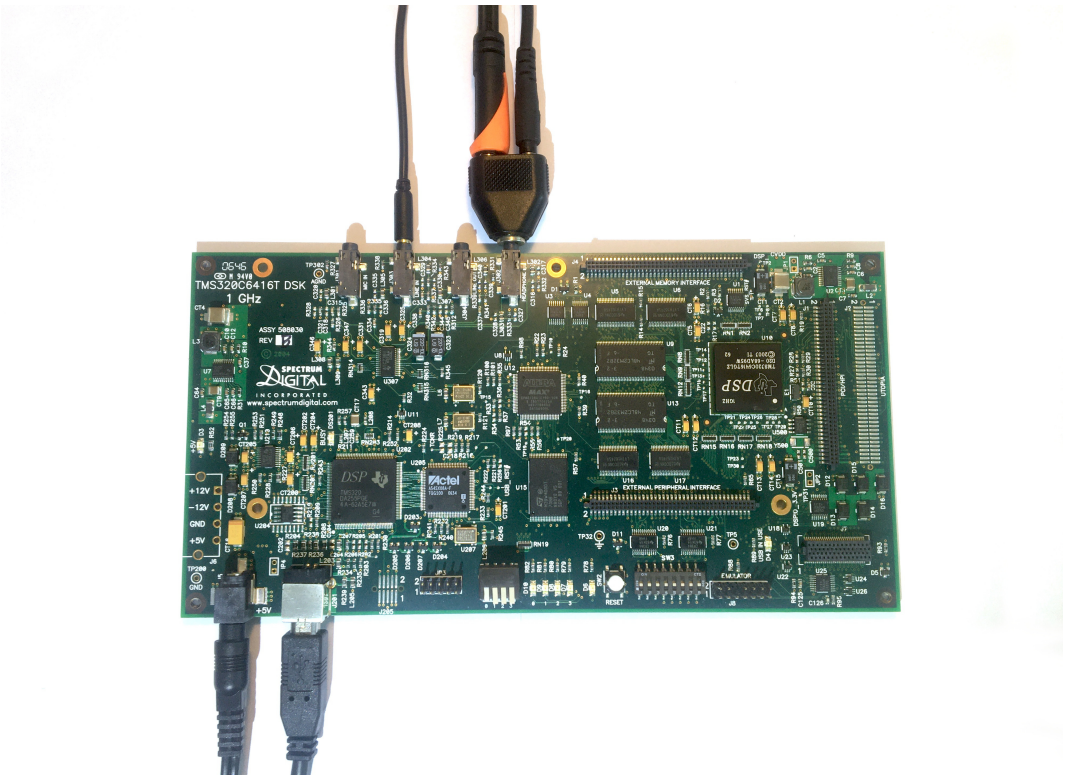
Jde zejména o:

- DSP čip TMS320C6416T s pracovní frekvencí 1 GHz
- Stereo zvukový kodek AIC23
- Synchronní pamět DRAM o kapacitě 16 MB
- Nevolatilní paměť Flash o kapacitě 512 kB
- 4 LED diody a 4 DIP přepínače pro univerzální použití
- Možnost volby zdroje časovacího signálu a úpravy bootovací sekvence
- Standardizované rozhraní pro rozšiřující přídavné moduly (konektory jsou na standardizovaných pozicích pro použití univerzálních rozšiřujících dceřiných karet)
- Integrovaný JTAG emulátor s USB rozhraním s možností připojení externího JTAG emulátoru
- Integrovaná distribuce potřebných napájecích napětí z jediného napájecího zdroje +5 V



Obr. 7.1: Blokový diagram vývojové desky (převzato z dokumentace [13])

Ovládání desky je plně integrované s rozhraním programu Code Composer Studio (viz 8.3), který je používán pro překlad zdrojového kódu a nahrání hotového kódu do paměti cílového procesoru.



Obr. 7.2: Vývojová deska TMS320C6416T DSK

7.1 Signálový procesor C6416T

Procesor TMS320C6416T je DSP čip, který pracuje v 16-bitové aritmetice s pevnou řádovou čárkou (tzv. fixed-point) postavený na patentované architektuře VelociTI. Architektura VelociTI je velmi výkonná pokročilá architektura typu VLIW (very-long-instruction-word), která je ideální volbou pro realizaci vícekanálových, multifunkčních a výpočetně náročných programů a aplikací [15].

Tento procesor poskytuje oproti dalším v dané kategorii obrovskou rychlost a výpočetní výkon, jeden jeho pracovní cyklus trvá cca 1 ns a během jednoho cyklu dokáže vykonat 8 instrukcí zároveň [14].

Procesor je maximálně optimalizován pro vykonávání operace multiply-accumulate (MAC), což je nejběžnější operace v oblasti DSP, která by se dala popsat pomocí rovnice: $accu = accu + (k[i] \cdot x[i])$. Procesor této třídy dokáže vykonat až 4000 MMAC/s (4000 milionů MAC za vteřinu) [14].

7.2 Zvukový kodek AIC23

Součástí této vývojové desky je i výkonný zvukový stereo kodek TLV320AIC23B. Jeho integrované A/D a D/A převodníky používají multibitovou sigma-delta architekturu včetně digitálních interpolačních filtrů se zabudovaným nadvzorkováním (tzv. oversampling).

Tento kodek podporuje množství vzorkovacích kmitočtů od 8 kHz do 96 kHz a datový přenos s délkou slova 16, 20, 24 a 32 bitů. Pro tento projekt byl zvolen vzorkovací kmitčet 96 kHz. Vysoký vzorkovací kmitočet umožňuje realizovat signálové zpracování v extrémně nízkém čase (0,19 ms).

Softwarové ovládání kodeku je realizované pomocí McBSP sériového portu (multichannel buffered serial port) a pomocí stejného sériového portu jsou přenášena i vstupní a výstupní zvuková data. Tímto sériovým portem je zvukový kodek propojen s DSP procesorem.

Vstupní A/D sigma-delta modulační převodník je realizován pomocí multibitové architektury třetího řádu a dosahuje odstupu signálu od šumu až 90 dBA při vzorkovacím kmitočtu 96 kHz.

Výstupní D/A sigma-delta modulační převodník je realizován pomoc multibitové architektury druhého řádu a dosahuje odstupu signálu od šumu až 100 dBA při vzorkovacím kmitočtu 96 kHz.

Součástí kodeku jsou také další integrované analogové obvody, které se podílejí na analogovém zpracování zvukového signálu před a za převodníky. Jde o stereo linkový vstup s digitálně řízeným nastavením zisku, stereo linkový výstup s digitálně řízenou výstupní úrovní a výkonný lineární sluchátkový zesilovač pro přímé připojení sluchátek na výstup kodeku. Všechny vstupy a výstupy lze také odpojit pomocí funkce mute. Kodek také obsahuje kompletní jednokanálový mikrofonní předzesilovač včetně zdroje napájení pro elektretový mikrofon s nastavitelnou citlivostí.

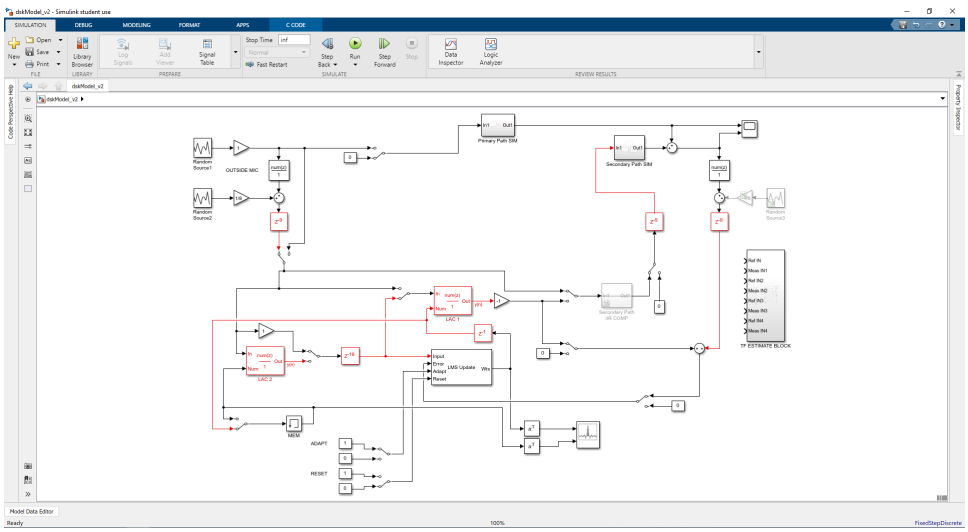
Obvod zvukového kodeku obsahuje také jeden 12 MHz krystal, který může být použit jako zdroj časovacího signálu pro kodek samotný, DSP čip i USB rozhraní.

8 Softwarové nástroje

8.1 MATLAB & Simulink

MATLAB® je programovací platformou speciálně navrženou pro analýzu dat a návrh procesů používanou inženýry a vědci po celém světě. Základním stavebním kamenem je programovací jazyk „MATLAB language“, který je navržený tak, aby jím bylo možné jednoduše a přehledně popisovat a řešit matematické a výpočetní úlohy. Tento programovací jazyk je maximálně optimalizovaný pro zpracování maticových dat a signálů.

Simulink® je grafické simulační prostředí založené na blokové analýze algoritmů, systémů a modelů. Toto prostředí je plně integrováno s prostředím MATLAB, je možné mezi nimi jednoduše přenášet větší množství vstupních i výstupních dat a v rámci modelu v Simulinku je možné implementovat algoritmy navržené v MATLABu.



Obr. 8.1: Vývojové prostředí Simulink

MATLAB i Simulink poskytují rozsáhlé knihovny optimalizovaných funkcí a výpočetních bloků používaných při návrhu, analýze a modelaci algoritmů signálového zpracování. Celé vývojové prostředí i rozšiřující knihovny doplňuje podrobná a přehledná dokumentace včetně příkladů a vzorových programů.

Mimo základní funkce těchto prostředí byly během této práce použity funkce a výpočetní bloky z následujících rozšřujících knihoven.

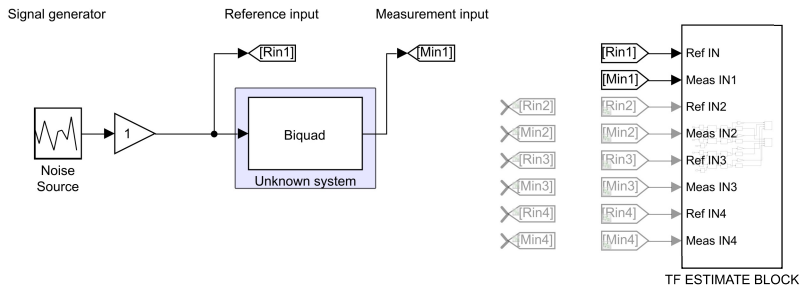
Audio toolbox poskytuje nástroje pro zpracování audio signálů. Součástí tohoto toolboxu jsou funkce pro návrh a realizaci ekvalizačních filtrů, dynamických procesorů a analýzu akustických měření, například odhad impulsní odezvy daného bloku. Zároveň tento toolbox poskytuje nástroje pro zpracování zvukových signálů v reálném čase, podporuje externí zvuková a MIDI rozhraní (včetně ASIO ovladačů) a nástroje pro tvorbu zásuvných VST modulů pro široké spektrum audio aplikací.

DSP system toolbox rozšiřuje vývojové prostředí o funkce a algoritmy pro pokročilé signálové zpracování. Pomocí tohoto toolboxu je možné modelovat FIR, IIR i adaptivní filtry. Také poskytuje nástroje pro analýzu a pozorování signálů v časové nebo kmitočtové doméně.

Filter Designer je samostatná aplikace zjednodušující návrh a analýzu číslicových filtrů. Pokud je k dispozici DSP toolbox, umožňuje Filter Designer provádět také kontrolovanou a kompenzovanou kvantizaci koeficientů výsledných filtrů, což je velmi důležitá funkce při návrhu filtrů pro výpočetní systém s aritmetikou pracující v pevné řádové čárce.

8.1.1 Měřící blok TF Estimate

V rámci modelace v prostředí Simulink byl pro účely této práce implementován měřící blok TF Estimate, který umožňuje zobrazit amplitudový a fázový přenos téměř kterékoli části modelu. Příklad použití je možné vidět na následujícím modelu, kde je měřen přenos neznámého systému simulovaného pomocí bloku „Biquad“.



Obr. 8.2: Použití bloku TF Estimate - model

Aby měřící cesty zbytečně nesnižovaly přehlednost daného modelu, jsou pro získávání referenčních a měřících signálů z různých bodů v modelu použity bloky „From“ a „Goto“.

Jak je možné vidět na modelu, měřený systém je nutné pro správnou funkci měření vybudit širokopásmovým zvukovým signálem a jsou-li v tomto signálu zastoupeny všechny části kmitočtového spektra, na dalším charakteru tohoto signálu nijak zvlášť nezáleží.

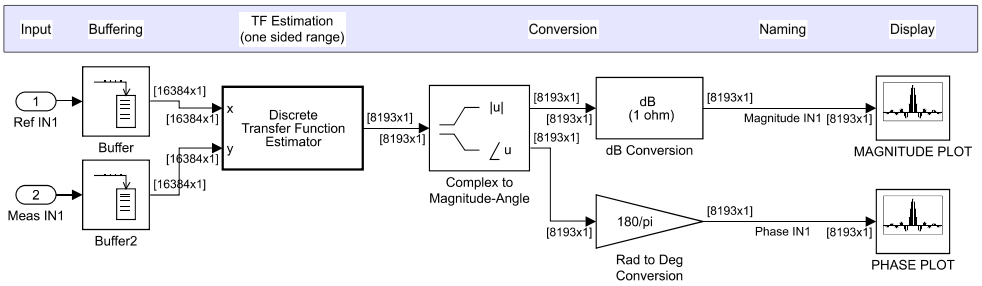
Pro měření potřeba získat dva signály, jeden referenční, který reprezentuje výchozí bod a druhý měřící, který reprezentuje kmitočtově zkreslený signál po průchodu měřeným systémem. Oba tyto signály jsou přivedeny na vstup bloku „TF Estimate“.

Vnitřní struktura jednoho kanálu

Vstupní vzorky jsou pro zvýšení přesnosti na nižších kmitočtech uloženy ve vstupním zásobníku a pro další zpracování je signál reprezentován jednotlivými bloky většího počtu vzorků.

Jádrem výpočtu je blok „Discrete Transfer Function Estimator“, který je součástí DSP system toolboxu a pomocí Welchovy metody průměrovaných periodogramů počítá vyhlazený spektrální odhad měřeného přenosu.

Výstupní komplexní jednostranné spektrum je následně konvertováno na spektrum amplitudové a fázové a to je ve vhodném formátu zobrazeno pomocí bloku „Array plot“.

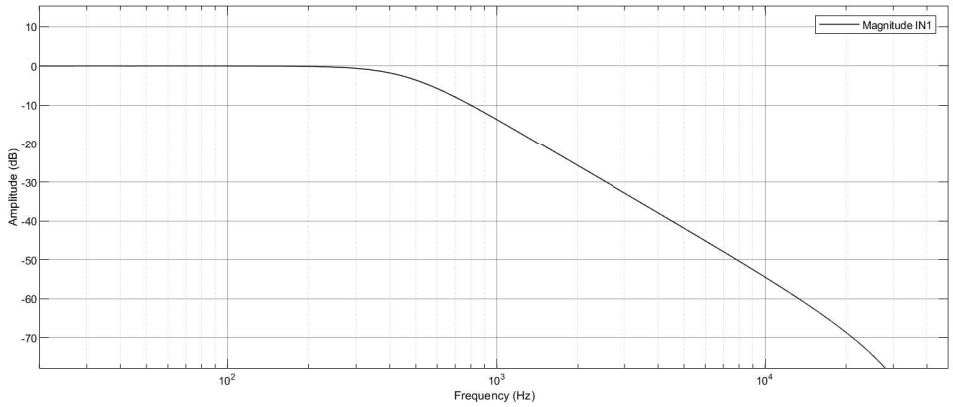


Obr. 8.3: Jeden kanál bloku TF Estimate

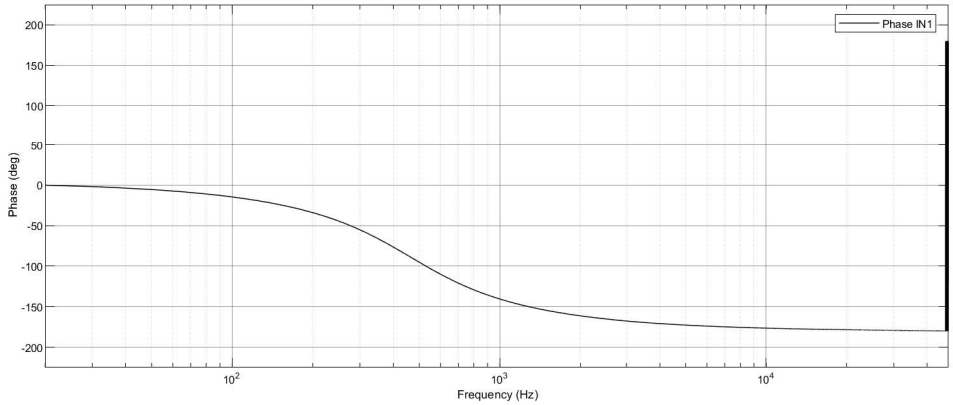
Zobrazení výsledného měření

Výsledné měření je zobrazeno pomocí bloku „Array plot“, který podporuje i zobrazení více signálů najednou. Jednotlivé měřené signály je pak možné zobrazit najednou a odlišit pomocí čtyř základních kontrastních barev.

Legenda pro popis jednotlivých signálů je automaticky generována podle názvu vstupního signálu.



Obr. 8.4: Měření přenosu neznámého systému - amplituda



Obr. 8.5: Měření přenosu neznámého systému - fáze

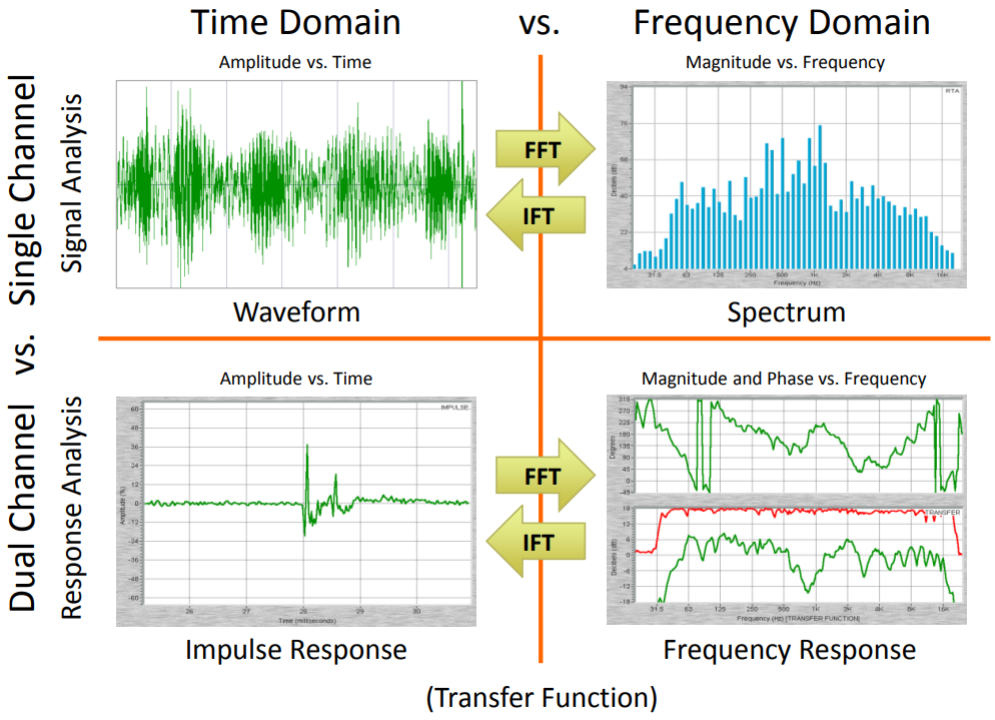
8.2 SMAART

Rational Acoustics Smaart® je dvoukanálový softwarový analyzér akustických signálů, fungující na bázi FFT, který umožňuje jednokanálovou i dvoukanálovou analýzu zvukových signálu v reálném čase, stejně tak jako offline analýzu změřené impulsní odezvy. Jde o nástroj pro měření a analýzu časových a kmitočtových vlastností zvukových systémů.

8.2.1 Analýza v časové a kmitočtové oblasti

Pro základní časovou analýzu zvukového signálu postačí běžný osciloskop, pokud se však zajímáme o analýzu kmitočtového spektra daného signálu, je třeba časovou reprezetaci tohoto signálu transformovat do kmitočtové oblasti.

K tomuto účelu slouží algoritmus FFT (fast Fourier transform - rychlá Fourierova transformace), případně jeho diskrétní alternativa DFT (discrete Fourier transform - diskrétní Fourierova transformace), případně algoritmus inverzní, je-li třeba signál převést z kmitočtové oblasti zpět do oblasti časové.



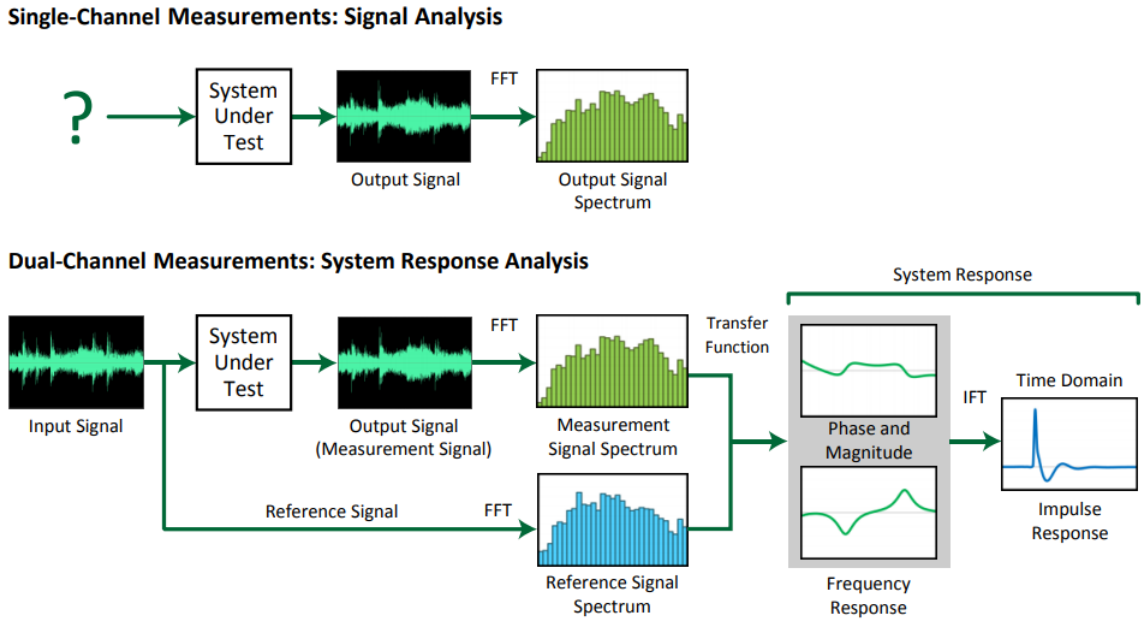
Obr. 8.6: Porovnání analýzy v časové a kmitočtové oblasti (převzato z [9])

8.2.2 Jednokanálová vs. dvoukanálová analýza

Pro měření zvukového signálu v reálném čase podporuje Smaart dva základní typy měření - jednokanálové a dvoukanálové.

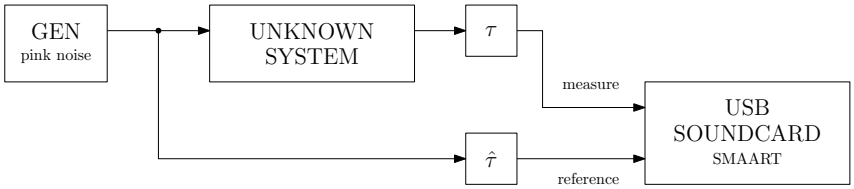
Jednokanálová analýza je pouze měřením spektrálních vlastností vstupního signálu, poskytuje pouze informaci o jeho amplitudovém spektru v závislosti na kmitočtu. Měřící nástroje RTA (Real Time Analyzer) a Spektrograf jsou založeny na jednokanálové analýze vstupního signálu. Dalším příkladem jednokanálového měření může být měření hladiny zvuku.

Dvoukanálová analýza oproti té jednokanálové porovnává dva vstupní (korelované!) signály a zobrazuje jejich vzájemné podobnosti a rozdíly. Dvoukanálového měření se využívá při měření impulsní odezvy a měření přenosu testovaného zvukového systému. Při dvoukanálové analýze totiž neměříme vlastnosti vstupního signálu ale reakci měřeného systému na daný vstupní signál. Jde o stejný princip jako v sekci 8.1.1, pokud vstupní signál splňuje podmínku, že se jedná o signál širokospektrální, na jeho ostatních charakteristikách nijak zvlášť nezáleží.



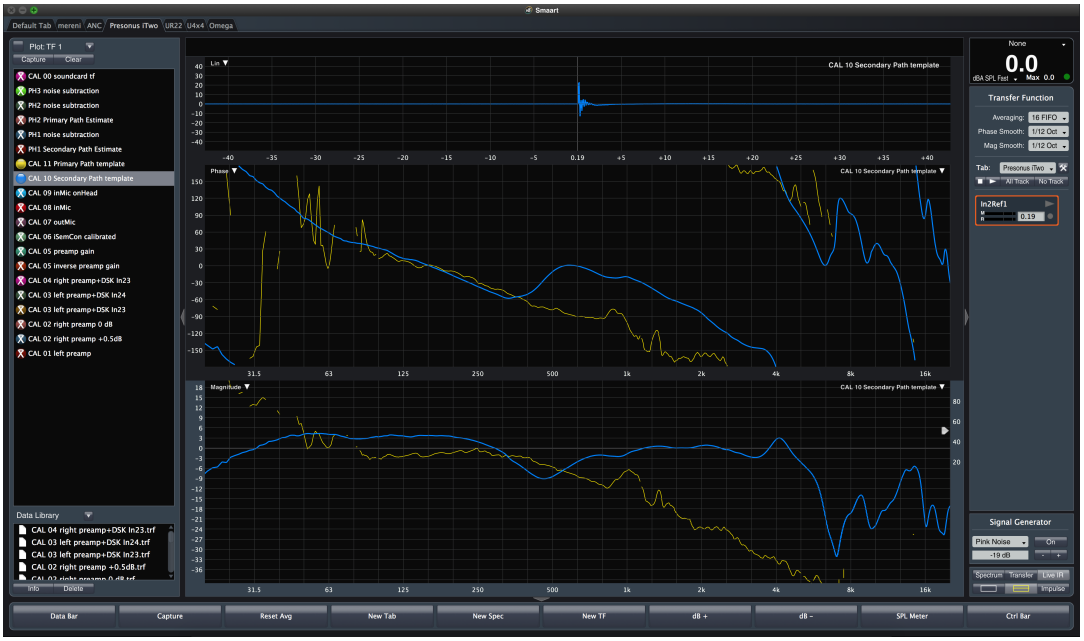
Obr. 8.7: Porovnání jednokanálové a dvoukanálové analýzy (převzato z [9])

Měřící analyzér Smaart také umožňuje kompenzovat nadbytečnou latenci, která mohla během daného měření vzniknout. Je důležité rozlišovat mezi časovým zpožděním, které je součástí měřeného systému a mělo by být i součástí výsledného měření, a nadbytečnou latencí, která vzniká pouze zvolenou metodou měření. Takouvou latencí může být například akustické zpoždění při cestě zvuku z měřeného reproduktoru do měřícího mikrofonu nebo vstupní a výstupní latence digitálního procesoru. Zapojení pro dvoukanálové měření přenosu neznámého systému včetně kompenzace nežádoucí latence vypadá následovně:



Obr. 8.8: Zapojení pro dvoukanálovou analýzu

Výsledné měření přenosu měřeného systému může vypadat následovně:

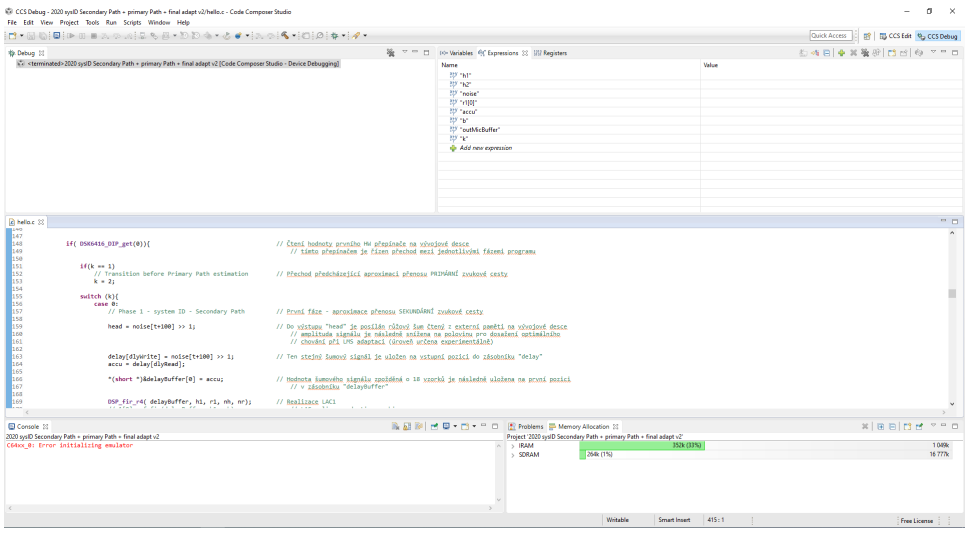


Obr. 8.9: Měřící software SMAART - dvoukanálová analýza

Více informací o funkcích tohoto softwaru je možné získat v uživatelském manuálu [9].

8.3 Code Composer Studio

Code Composer Studio je proprietárním vývojovým prostředím zaměřeným na vývoj programů pro mikrokontroléry a signálové procesory firmy Texas Instruments. Poskytuje souhrn nástrojů pro vývoj a ladění integrovaných aplikací pro tyto procesory a obsahuje běžný překladač jazyka C/C++, editor zdrojového kódu, debugger s možností procházet běžící program krok po kroku a další pomocné nástroje.



Obr. 8.10: Vývojové prostředí Code Composer Studio 6.2.0.00050

Integrace s těmito signálovými procesory zároveň umožňuje analýzu dostupných prostředků - např. kontrolu vytížení jednotlivých bloků paměti, zobrazení stavu a hodnoty jednotlivých proměnných atd.

Code Composer Studio komunikuje s vývojovou deskou s DSP čipem (7) pomocí USB rozhraní a JTAG programátoru integrovaného přímo na vývojové desce.

9 Nastavení a kontrola použitých komponent

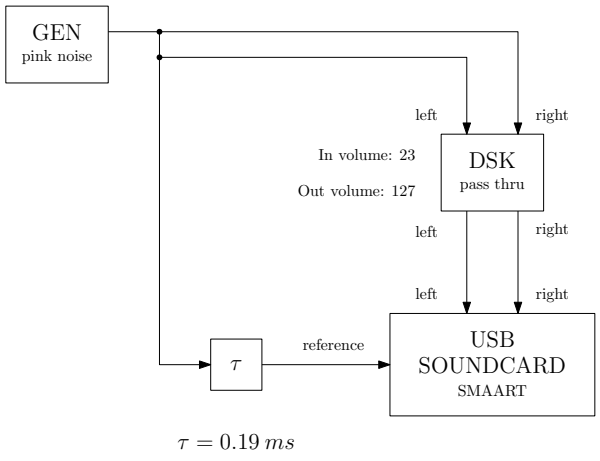
Předtím, než bude možné se pustit do dalšího vývoje, je třeba správně zapojit, nastavit a ověřit vlastnosti zvoleného hardwaru. V rámci realizace našeho zapojení používáme následující komponenty:

- sluchátka doplněná o dva mikrofony (6.3.2)
- diskrétní mikrofonní předzesilovač, zajišťující prvotní zesílení signálu a v neposlední řadě poskytuje mikrofonům požadované napájecí napětí (6.2)
- digitálně řízený integrovaný analogový mikrofonní předzesilovač ams A3435 (6.1)
- vývojová deska s DSP čipem TMS320C6416T (7)

9.1 Přenos vývojové desky s DSP čipem

Zkratka DSK (DSP Starter Kit) označuje vývojovou desku s DSP čipem TMS320C6416T, v následujícím textu je tato zkratka ve vhodných případech použita pro označení této vývojové desky, včetně všech komponent, které obsahuje. Tato deska je zde použita nejen pro implementaci adaptivního algoritmu, vykonává veškeré dodatečné číslicové zpracování zvukového signálu.

Pokud si však v tuto chvíli toto signálové zpracování odmyslíme a naprogramujeme desku tak, aby pouze opakovala vstupní signál na výstup, můžeme změřit její přenos a zjistit tak jednak latenci, kterou do signálové cesty přináší a jednak potenciální kmitočtové zkreslení, kterým může signálovou cestu zatížit.

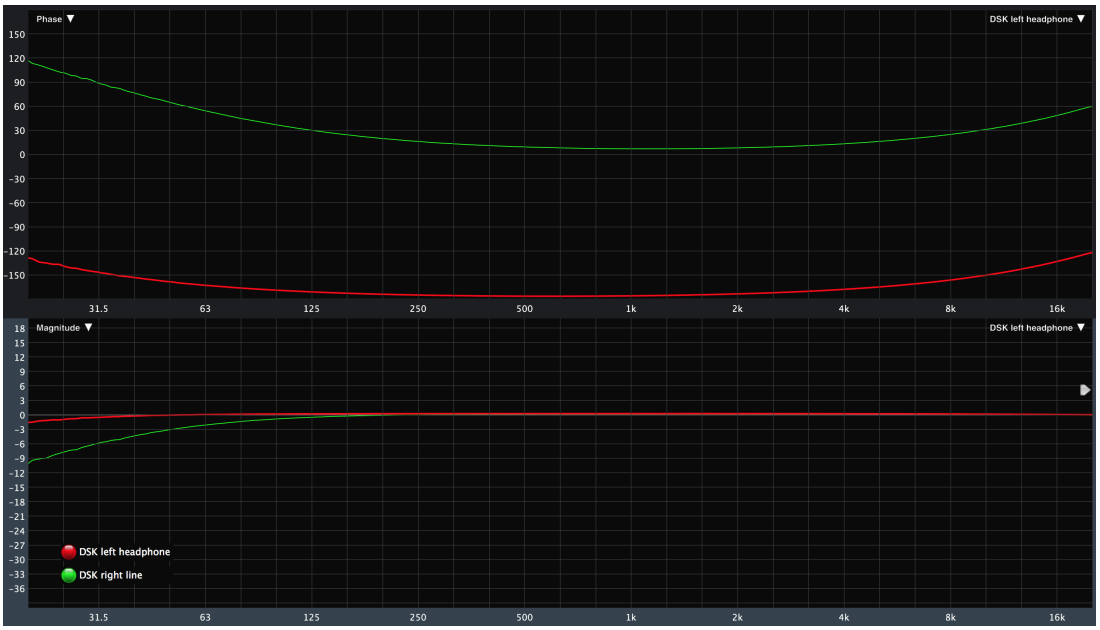


Obr. 9.1: Měření přenosu vývojové desky s DSP čipem

Na vstup DSK je přiveden širokopásmový šum z generátoru a výstupní signál z této desky je připojen přímo na odpovídající vstupy měřící zvukové karty.

Pro měření přenosu je třeba ještě vhodně zvolit referenční signál. Tento signál získáme přímo z výstupu generátoru šumu. Pro měření přenosu je vždy důležité, aby referenční a měřený signál dorazili do měřící jednotky ve správný čas, jinak by bylo výsledné měření fázové charakteristiky zkreslené touto chybou (viz 8.2). Z toho důvodu je referenční signál ještě před vstupem do měřící zvukové karty zpožděn o latenci desky DSK $0,19$ ms.

Výsledné měření přenosu DSK vypadá následovně:



Obr. 9.2: Přenos vývojové desky s DSP čipem

Při měření byly změřeny dva různé výstupy této vývojové desky, linkový výstup a sluchátkový výstup. I když to není úplně logické, měření ukazuje, že linkový výstup sice dodrží správnou polaritu výstupu, ale obsahuje integrovaný filtr typu horní propusti, a to tak vysoko, že to pro naše použití není přípustné. Oproti tomu sluchátkový výstup (pro účely měření byla srovnána výstupní hlasitost obou výstupů) sice invertuje výstupní signál, ale jeho přenos je na nejnižších kmitočtech o mnoho přijatelnější.

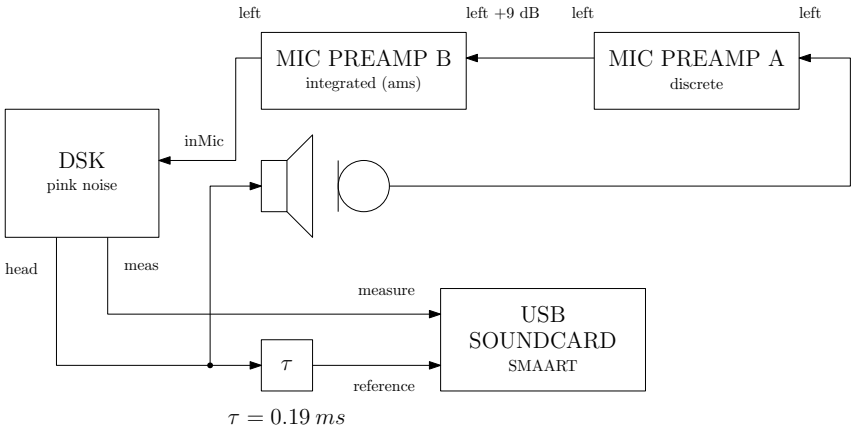
Mimo nejnižších kmitočtů je přenos této vývojové desky ideální.

Jelikož je pro realizovanou konstrukci výhodnější použít sluchátkový výstup - zesílením i kmitočtovým přenosem - bude na konci programu výstupní hodnota invertována pro kompenzaci inverze sluchátkového výstupu.

9.2 Určení zesílení mikrofonního předzesilovače

Mikrofonní předzesilovač se skládá ze dvou oddělených částí - diskrétního předzesilovače s pevně nastaveným zesílením (viz 6.2) a digitálně řízeného integrovaného předzesilovače ams A3435 (viz 6.1), jehož zesílení lze přesně nastavit pomocí obslužného programu. Celkové zesílení předzesilovače je tedy nastavitelné pomocí zesílení ams A3435.

Toto zesílení určíme pomocí následujícího diagramu:



Obr. 9.3: Zapojení pro určení zesílení mikrofonního předzesilovače

Při tomto měření je deska DSK použita pro přehrávání širokopásmového růžového šumu z externí paměti do výstupu označeného jako „head“. Tento výstup je přímo napojen na reproduktor sluchátka. Ten daný signál vyzáří do vnitřního prostoru sluchátka, kde jej následně zaznamená vnitřní mikrofon. Signál z tohoto mikrofonu je následně zesílen předzesilovačem a přiveden na vstup DSK označený jako „inMic“.

DSK tento signál bez dalšího zpracování zopakuje na měřící výstup (označen jako „meas“).

Oba výstupní signály („head“ a „meas“) jsou následně pomocí měřící zvukové karty přivedeny do měřícího softwaru SMAART (viz 8.2). Signál z výstupu „head“ je před vstupem do měřícího prostředí zpožděn o délku latence 0, 19 ms, která vznikla při D/A a A/D převodu signálu „inMic“ při vstupu do a výstupu z desky DSK.

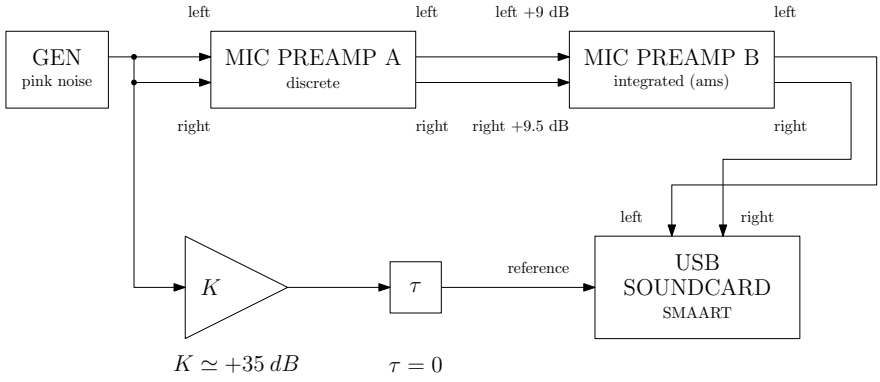
Zesílení předzesilovače je teoreticky nutné nastavit tak, aby změřený přenos odpovídal hodnotě 1 (0 dB). Není to však tak jednoduché, protože v rámci tohoto zapojení měříme přenos soustavy reproduktor-mikrofon a ten není vůči kmitočtu lineární (viz následující sekce 9.4), proto je potřeba zvolit kmitočtovou oblast, jejíž přenos nastavíme na 1. S ohledem na vybraná sluchátka byla tato oblast zvolena na rozsah cca $1 - 4$ kHz.

Výsledné zesílení, nastavené v integrovaném zesilovači pro tento kanál je +9 dB. Tento kanál použijeme jako referenční pro nastavení kanálu druhého.

Pro úspěšné měření je klíčové, aby oba vstupy měřící zvukové karty měly shodnou vstupní citlivost!

9.3 Relativní přenos mikrofonního předzesilovače

Pro kontrolu přenosu mikrofonního předzesilovače a nastavení zesílení druhého kanálu použijeme následující zapojení:



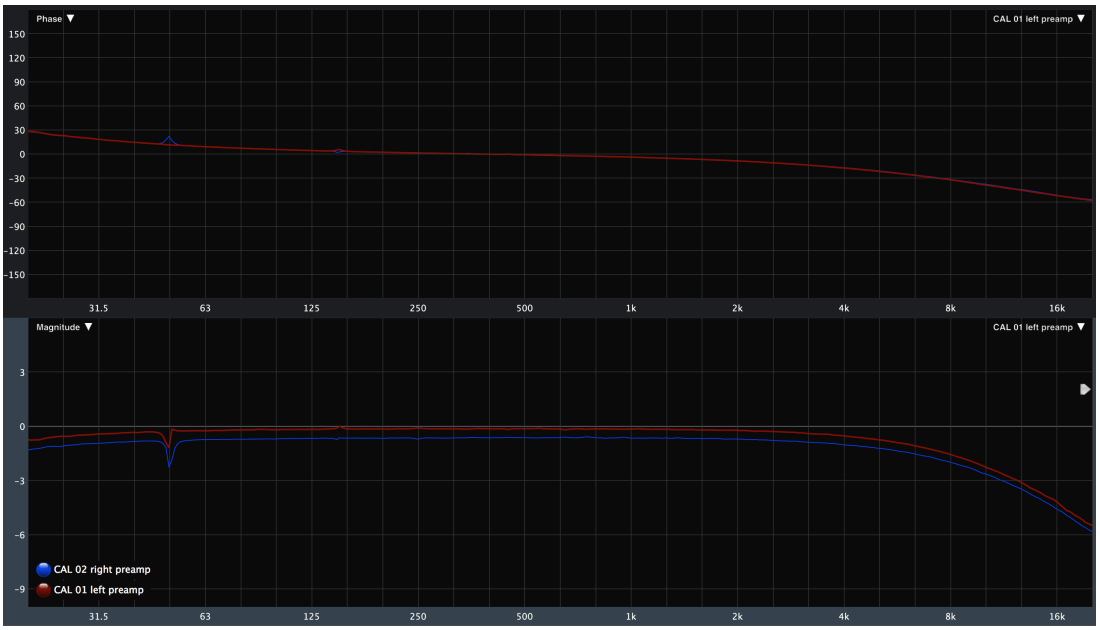
Obr. 9.4: Měření relativního přenosu mikrofonního předzesilovače

Z předchozího měření již víme, že levý kanál musí mít nastavené zesílení +9 dB, jako výchozí bod pro měření přenosu pravého kanálu můžeme zvolit stejné zesílení.

Výstup z generátoru širokopásmového šumu přivedeme na oba vstupy diskrétního mikrofonního předzesilovače a oba výstupy integrovaného předzesilovače přivedeme na vstup měřící zvukové karty.

Referenční signál získáme přímo z výstupu generátoru šumu. Aby výsledné měření přenosu nebylo ovlivněno celkovým zesílením předzesilovače (cílem je ověřit, je-li přenos předzesilovače vyrovnaný napříč kmitočtovým spektrem), aplikujeme odpovídající zesílení i na signál referenční. Celkové zesílení předzesilovače odpovídá cca +35 dB. Při tomto měření není nutné kompenzovat žádnou latenci, jelikož je předzesilovač kompletně analogový a do obvodu žádnou dodatečnou latenci nevnáší.

Výsledné měření vypadá následovně:



Obr. 9.5: Relativní přenos mikrofonního předzesilovače před korekcí zesílení

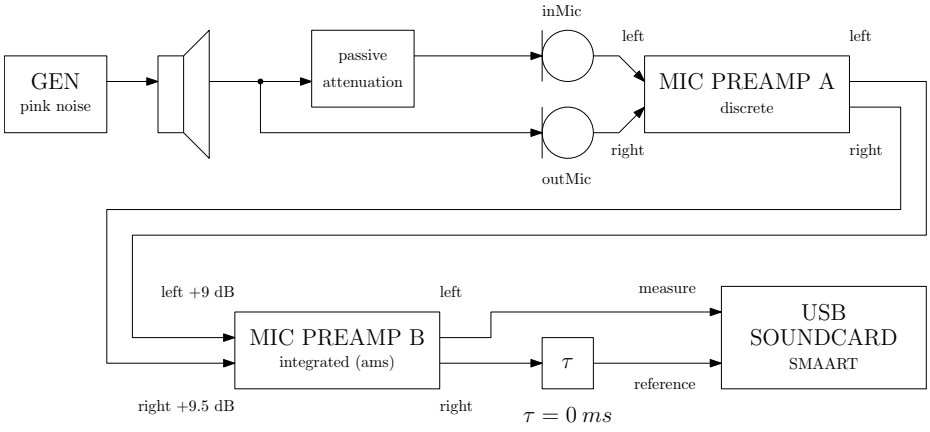
Z tohoto měření vyplývá, že je třeba provést korekci zesílení pravého kanálu, a to přesně o $+0,5\,\mathrm{dB}$ - nastavení zesílení tohoto kanálu tedy bude ve výsledku $+9,5\,\mathrm{dB}$.

Zároveň je možné po provedení tohoto měření konstatovat, že je přenos mikrofonního předzesilovače vyrovnaný na široké části kmitočtového spektra. Přenos mírně klesá na nejvyšších kmitočtech, což je pro tuto aplikaci přijatelné (na hodnotu $-3\mathrm{dB}$ klesne až na kmitočtu $12,27\,\mathrm{kHz}$).

9.4 Přenos sekundární cesty a pasivní útlum sluchátka

Jak již bylo zmíněno, při použití zapojení pro určení zesílení mikrofonního předzesilovače (sekce 9.2) vlastně měříme přenos soustavy reproduktor-mikrofon umístěné uvnitř sluchátka, tento přenos již tedy změřen byl. Tuto soustavu označujeme při konstrukci systému pro odečet okolního šumu jako sekundární zvukovou cestu (viz 11.1.1), jelikož jde o druhou cestu, kterou se okolní šum dostává do vnitřního prostoru sluchátka. Přenos sekundární cesty je jedním z nejdůležitějších faktorů při návrhu systému pro odečet okolního šumu.

Pojem pasivní útlum sluchátka je pojmem pro přenos zvukového signálu z okolního prostředí do vnitřního prostoru sluchátka přirozenou cestou, tedy akusticky. Je ovlivněn primárně fyzickou konstrukcí sluchátka a použitými materiály (viz 2.2). Tento útlum označujeme jako primární zvukovou cestu (viz 11.1.1) a je možné jej změřit pomocí následujícího zapojení:

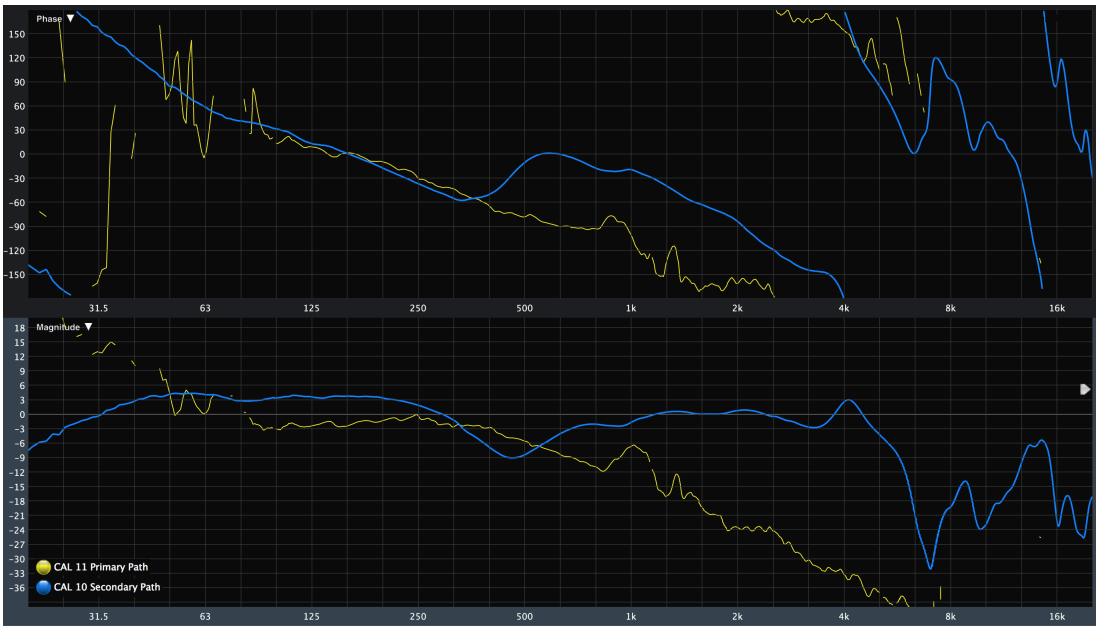


Obr. 9.6: Měření pasivního útlumu sluchátka

Při měření pasivního útlumu sluchátka přehráváme do bezprostředního okolí sluchátka širokopásmový šum. Tento šum je zaznamenán dvěma mikrofony, jednak mikrofonem na vnější stěně sluchátka (označen jako „outMic“) a jednak mikrofonem uvnitř sluchátka (označen jako „inMic“). Vnější mikrofon zaznamenává přímo okolní šum, jeho výstup tedy zvolíme referenčním signálem. Vnitřní mikrofon zaznamenává okolní šum až po útlumu hmotou a rozměry sluchátka, tzn. na jím zaznamenaný signál byl aplikován pasivní útlum sluchátka a jeho výstup se stává signálem měřeným.

Signály z obou mikrofonů jsou následně patřičně zesíleny a přivedeny na vstup měřící zvukové karty. Na referenční signál není aplikováno žádné časové zpoždění, a to přesto, že se oba mikrofony nacházejí na jiném místě v prostoru a tedy čas, kdy mikrofony zaznamenají signál z okolí, se mírně liší. Tento časový rozdíl je však v rámci tohoto obvodu považován za součást měřeného přenosu a proto by nebylo vhodné jej kompenzovat.

Výsledek obou měření je možné vidět na následujícím obrázku. Při dalším návrhu zapojení jsou tyto přenosy použity jako vzor pro modelovou reprezentaci daných částí obvodu i pro kontrolu správné aproximace těchto přenosů.



Obr. 9.7: Přenos sekundární zvukové cesty a pasivní útlum sluchátka

10 Adaptivní filtrace

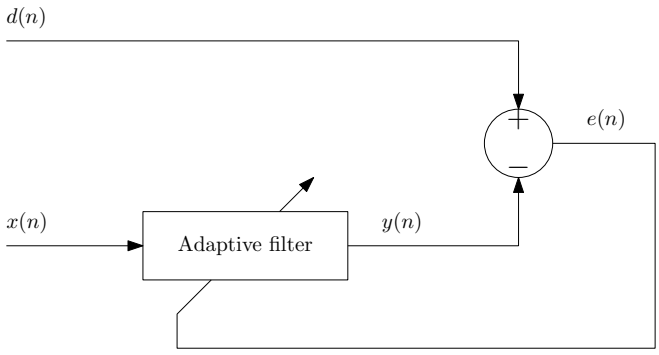
Jednou z výhod, které s sebou přináší použití číslicového zpracování signálů oproti čistě analogovému řešení je možnost použití adaptivních filtrů. Jedná se o filtry typu FIR nebo IIR s časově proměnnými koeficienty. I když je jistě možné implementovat adaptivní filtr typu IIR, použití adaptivních filtrů typu FIR v drtivé většině převládá, narozdíl od IIR filtru je totiž FIR filtr stabilní, ať už jsou jeho koeficienty jakékoli. Při manipulaci s adaptivním filtrem typu IIR je nutné v každém cyklu znovu kontrolovat jeho stabilitu z důvodu změny koeficientů, což je časově i výpočetně náročné a tudíž většinou nevýhodné.

Při návrhu klasických filtrů je nutným předpokladem znalost žádané kmitočtové charakteristiky, které se pomocí těchto filtrů snažíme dosáhnout. V praxi však existují i situace, kdy je nutné realizovat filtraci i v případě, že zadání pro návrh filtru ještě není známé nebo se neustále mění. V takových případech může být velmi užitečný adaptivní filtr, který se dokáže přizpůsobovat novým podmínkám a jejich postupným změnám.

Linear adaptive combiner

Jádrem většiny adaptivních filtrů je filtr typu FIR s proměnnými koeficienty, v zahraniční literatuře nazývaný „linear adaptive combiner“. Základní struktura adaptivního systému se skládá ze dvou zdrojových signálů - vstupního a referenčního. Vstupní signál $x(n)$ je filtrován pomocí adaptivního filtru a následně odečten od referenčního signálu $d(n)$. Výsledný signál po odečtu je nazáván chybovým signálem, protože přímo reprezentuje chybu daného odečtu. Na základně tohoto chybového signálu $e(n)$ pak probíhá adaptace koeficientů adaptivního filtru. Adaptivní filtrace funguje za předpokladu, že je možné ze signálu $x(n)$ získat signál $d(n)$ pomocí filtrace nebo průměrování [2].

Základní zapojení adaptivního filtru vypadá následovně:



Obr. 10.1: Základní zapojení adaptivního filtru

Výpočet chybového signálu:

$$e(n) = d(n) - y(n)$$

Kde $e(n)$ je aktuální vzorek chybového signálu, $d(n)$ je aktuální vzorek referenčního signálu a $y(n)$ je aktuální výstupní vzorek adaptivního filtru.

Koeficienty adaptivního filtru

Výstup adaptivního filtru je získán podle předpisu pro výpočet diskrétní konvoluce:

$$y(n) = \sum_{k=1}^{\mathrm{N}-1} h_{\mathrm{k}}(n)x(n-k)$$

Kde **h** reprezentuje vektor N koeficientů $\mathrm{h_k}$ adaptivního filtru.

Tyto koeficienty jsou v každém cyklu adaptovány takovým způsobem, aby adaptivní filtr směřoval k tzv. optimálnímu řešení. Filtr dosáhne optimálního řešení ve chvíli, kdy je chybový signál po odečtu minimální - často je toto kritérium upřesněno tak, že proměnnou, kterou se snažíme pomocí adaptivní filtrace minimalizovat, je střední hodnota chybového signálu. Sledováním chybového signálu a jeho změn lze jednoduše hodnotit a pozorovat chování a efektivitu adaptivní filtrace.

Při praktickém využití adaptivních filtrů bývá největším oříškem nalezení správného referenčního signálu $d(n)$, jelikož nemusí být na první pohled zřejmé, kde se v celém obvodu s adaptivním filtrem referenční signál nachází [1].

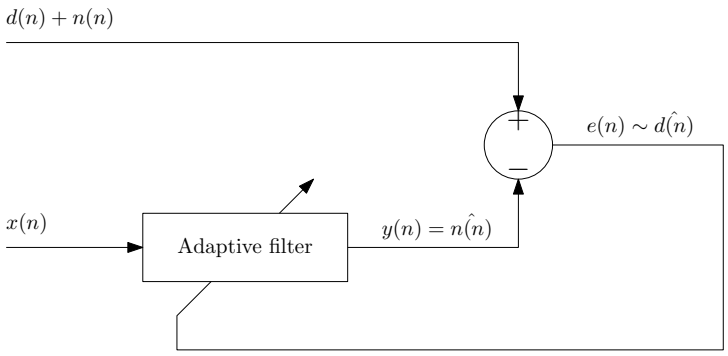
10.1 Struktury adaptivních filtrů

Adaptivní filtraci jde použít v různých situacích při různých aplikacích. Pro každé použití se zapojení a funkce adaptivního filtru mírně liší, avšak princip adaptivní filtrace zůstává stejný. Tři nejběžnější případy použití adaptivního filtru jsou potlačení rušení, identifikace neznámého systému a inverzní filtrace.

10.1.1 Použití adaptivního filtru pro potlačení šumu

Prvním způsobem využití adaptivního filtru je potlačení šumu. Při této aplikaci je refereční signál $d(n)$ rušen nekorelovaným aditivním šumem $n(n)$. Vstupním signálem $x(n)$ adaptivního filtru je šumový signál korelovaný s aditivním šumem $n(n)$ přítomným v referenčním signálu. Tento šum by sice měl pocházet ze stejného zdroje jako šum přítomný v referenčním signálu, ale může být částečně upraven nebo modifikován.

Pokud není signál $d(n)$ korelován s šumovým signálem $n(n)$, vypadá struktura zapojení adaptivního filtru pro potlačení šumu následovně:

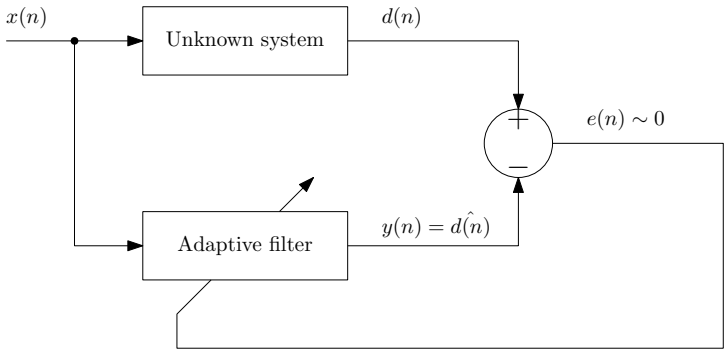


Obr. 10.2: Struktura zapojení adaptivního filtru pro potlačení šumu

Při realizaci tohoto zapojení je důležité, aby referenční signál $d(n)$ opravdu nebyl korelovaný s šumovým signálem $n(n)$ - při splnění této podmínky je minimalizována pouze šumová složka tohoto signálu $n(n)$ a referenční signál $d(n)$ není odečtem nijak ovlivněn [1].

10.1.2 Použití adaptivního filtru pro aproximaci přenosu neznámého systému

Dalším způsobem použití adaptivního filtru je použití pro identifikaci přenosu neznámého systému. Vstupní signál $x(n)$ je společný pro vstup adaptivního filtru i pro vstup neznámého systému. Výstup neznámého systému se stává signálem referenčním $d(n)$, od něhož je následně odečten výstupní signál adaptivního filtru $y(n)$.

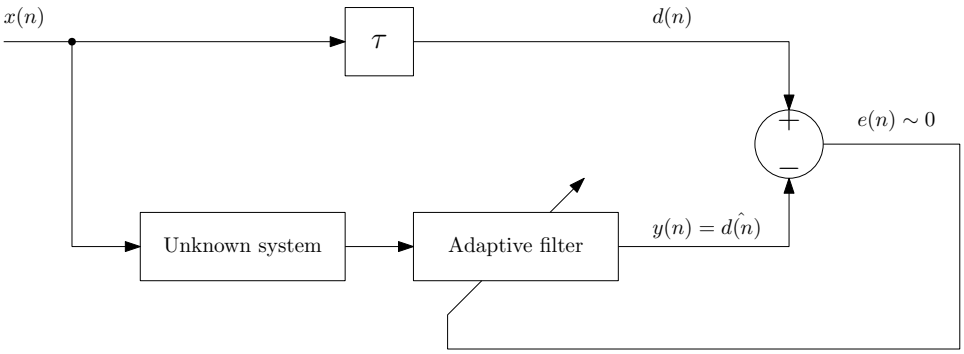


Obr. 10.3: Struktura zapojení adaptivního filtru pro identifikaci neznámého systému

Chybový signál $e(n)$ ovlivňuje změnu koeficientů adaptivního filtru až do chvíle, kdy bude signál $y(n)$ odpovídat signálu $d(n)$, tedy přenos adaptivního filtru bude odpovídat přenosu neznámého systému.

10.1.3 Inverzní filtrace

Adaptivní filtr lze použít i pro kompenzaci přenosu neznámého systému, který svým přenosem zatěžuje vstupní signál. Vstupní signál $x(n)$ je přiveden zároveň na vstup neznámého systému a zároveň na vstup zpožďovacího bloku, který kompenzuje případnou latenci kompenzovaného systému. Výstup tohoto systému je přímo přiveden na vstup adaptivního filtru. Zpožděný vstupní singál se stává signálem referenčním $d(n)$.



Obr. 10.4: Struktura zapojení adaptivního filtru pro inverzní filtraci

Adaptace probíhá tak dlouho, dokud se $y(n)$ a $d(n)$ neshodují, tedy do doby, kdy přenos adaptivního filtru odpovídá inverznímu přenosu neznámého systému.

10.2 Algoritmus pro výpočet nových koeficientů

I když je jádro adaptivního filtru často stejné, efektivitu adaptivního filtru určuje efektivita algoritmu, který počítá a aktualizuje koeficienty tohoto filtru.

10.2.1 Parametry adaptivních algoritmů

Aby bylo možné nějakým způsobem porovnávat jednotlivé adaptivní algoritmy, je třeba si stanovit parametry adaptivních filtrů, které je možné mezi sebou srovnat.

Rychlost konvergence

Rychlost konvergence je udávána jako doba, za kterou dosáhne adaptivní filtr svého optimálního stavu.

Konečná chyba konvergence

Některé adaptivní algoritmmy jsou schopné dosáhnout přímo optimálního řešení, jiné se k němu mohou jen velmi přiblížit. Právě rozdíl mezi optimálním a reálně dosažitelným řešením je nazýván konečnou chybou konvergence.

Robustnost algoritmu

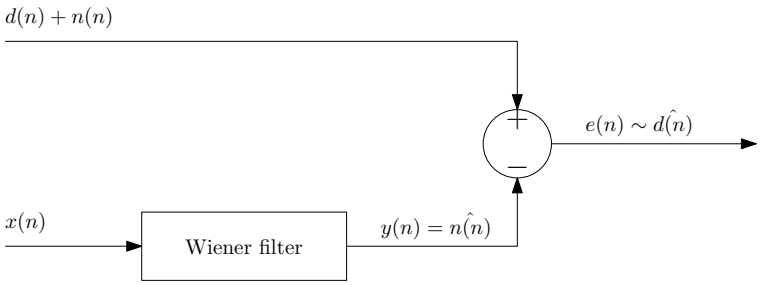
Pokud je algoritmus robustní, přiblíží se optimálnímu řešení i při špatně zadaných počátečních podmínkách. Robustnost algoritmu záleží na mnoha faktorech, hlavně však na požadované rychlosti adaptace a dostupném numerickém rozlišení - čím nižší numerické rozlišení je k dispozici, tím méně je algoritmus robustní. Obecně platí, že zhoršené numerické vlastnosti vlivem kvantovacích chyb mají na robustnost adaptivních filtrů velký vliv a mohou být příčinnou zhoršené stability i přesnosti adaptivní filtrace.

Výpočetní náročnost

Volba algoritmu pro výpočet koeficientů adaptivního filtru je závislá především na dostupných technických prostředcích. Čím výkonnější je použitý výpočetní procesor, tím náročnější a přesnější algoritmus je možné implementovat. Jelikož je však většinou po adaptivním filtru vyžadována funkce v reálném čase, budou dostupné výpočetní prostředky vždy omezené. Výsledné řešení je tedy často kompromisem mezi výpočetní náročností a dostatečnou přesností dosažitelného řešení.

10.2.2 Wienerova filtrace

Na úplném prvopočátku myšlenky adaptivních filtrů se nachází tzv. Wienerova filtrace. Nejedná se však o adaptivní filtraci, při Wienerově filtraci se provádí výpočet koeficientů na základě statistických vlastností vstupního a referenčního signálu a tento výpočet proběhne ještě před filtrací samotnou.



Obr. 10.5: Příklad zapojení Wienerovy filtrace

Považujme $\mathbf{x}$ za vektor vstupního signálu a $\mathbf{h}$ je vektor koeficientů filtru.

$$\mathbf{x} = [x_{\mathrm{k}}\ x_{\mathrm{k}-1}\ x_{\mathrm{k}-2}\ x_{\mathrm{k}-3}\ \ldots\ x_{\mathrm{k}-(\mathrm{N}-1)}]^T$$

$$\mathbf{h} = [h(0)\ h(1)\ h(2)\ h(3)\ \ldots\ h(N-1)]^T$$

Výstup Wienerova filtru lze získat stejným způsobem jako výstup běžného FIR filtru pomocí diskrétní konvoluce:

$$y(n) = \sum_{k=1}^{\mathrm{N}-1} h_{\mathrm{k}}(n)x(n-k)$$

Tento předpis jde zapsat i v maticovém tvaru:

$$y(n) = \mathbf{h}^T\mathbf{x}(n)$$

Výsledný odhad původního signálu po odečtu odhadnutého rušení je vyjádřen:

$$e(n) = d(n) - y(n) = d(n) - n(\hat{n}) = d(n) - \mathbf{h}^T\mathbf{x}(n)$$

Výpočet střední kvadratické chyby

Cílem Wienerovy filtrace je minimalizovat střední kvadratickou hodnotu chybového signálu, která je vyjádřena:

$$E[e(n)^2] = E[\ (d(n) - \mathbf{h}^T\mathbf{x}(n))^2\ ]$$

$$E[e(n)^2] = E[(d(n)^2] - 2E[\ \mathbf{h}^T d(n)\mathbf{x}(n)\ ] + E[\ (\mathbf{h}^T)^2(\mathbf{x}(n))^2\ ]$$

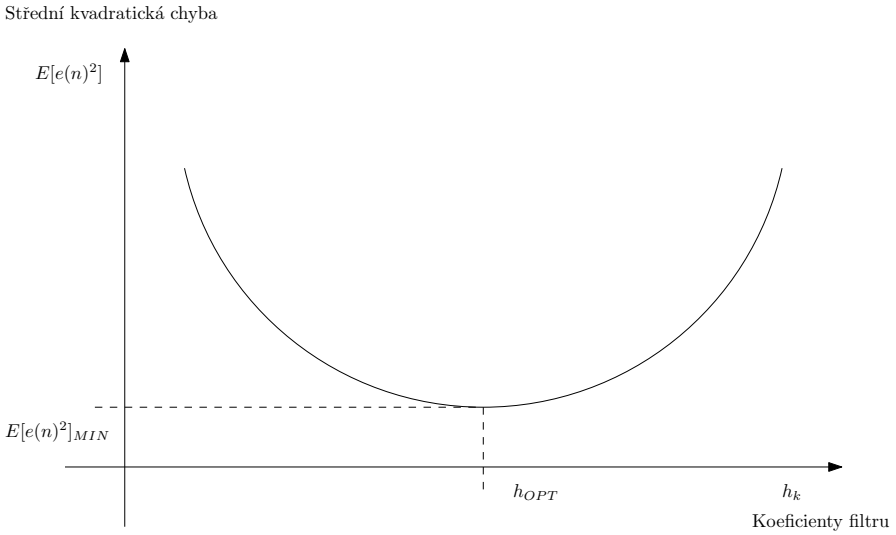
Výsledným vztahem pro výpočet střední kvadratické hodnoty chybového signálu je následující účelová funkce:

$$E[e(n)^2] = E[(d(n)^2] - 2E[\ \mathbf{h}^T\mathbf{R}_{\mathrm{DX}}\ ] + E[\ (\mathbf{h}^T)^2\mathbf{R}_{\mathrm{XX}}\ ]$$

Kde $\mathbf{R}_{\mathrm{DX}}$ je kroskorelační vektor délky N a $\mathbf{R}_{\mathrm{XX}}$ autokorelační matice s rozměrem N × N.

Závislost mezi střední kvadratickou chybou a koeficienty filtru

Dle [3] a [6] má závislost mezi střední kvadratickou chybou a koeficienty filtru geometrický tvar vícerozměrného paraboloidu, jehož vrchol definuje optimální hodnoty koeficientů filtru. Takový tvar pro jeden koeficient je možné vidět na následujícím obrázku. Počet rozměrů paraboloidu je přímo úměrný počtu koeficientů filtru.



Obr. 10.6: Závislost mezi střední kvadratickou chybou a koeficienty filtru

Jestliže se optimální hodnota koeficientů filtru nachází ve vrcholu této závislosti, je možné tuto optimální hodnotu najít pomocí výpočtu gradientu této závislosti - ve vrcholu se totiž gradient rovná nule.

$$\nabla E[e(n)^2] = \frac{\mathrm{d}\ E[e(n)^2]}{\mathrm{d}\ \mathbf{h}}$$

$$\nabla E[e(n)^2] = 0 - 2\mathbf{R}_{\mathrm{DX}} + 2\mathbf{h}\mathbf{R}_{\mathrm{XX}}$$

Pokud nyní položíme $\nabla E[e(n)^2]$ roven nule, získáme vztah pro výpočet optimálních hodnot koeficientů.

$$\nabla E[e(n)^2] = 0 - 2\mathbf{R}_{\mathrm{DX}} + 2\mathbf{h}\mathbf{R}_{\mathrm{XX}} = 0$$

$$2\mathbf{R}_{\mathrm{DX}} = 2\mathbf{h}\mathbf{R}_{\mathrm{XX}}$$

$$\mathbf{h}_{\mathrm{OPT}} = \mathbf{R}_{\mathrm{XX}}^{-1}\mathbf{R}_{\mathrm{DX}}$$

V literatuře bývá zvykem označovat $\mathbf{R}_{\mathrm{DX}}$ jako $\mathbf{P}$ [3], výslednému vztahu se pak říká Wiener-Hopfova rovnice nebo také Wiener-Hopfovo řešení.

$$\mathbf{h}_{\mathrm{OPT}} = \mathbf{R}_{\mathrm{XX}}^{-1}\mathbf{P}$$

10.2.3 Adaptivní algoritmy RLS a LMS

Adaptivní filtr narozdíl od Wienerovy filtrace vychází z nějakých počátečních podmínek a není u něho vyžadována předběžná znalost statistických vlastností zdrojového signálu.

Adaptivních algoritmů, s jejichž pomocí je možné aktualizovat koeficienty adaptivního filtru existuje více (např. Kalmanova filtrace), ale nejpoužívanějšími jsou RLS a LMS.

RLS

Rekurzivní metoda nejmenší kvadratické chyby - „recursive least squares“ - je adaptivní verzí Wienerova filtru. Pro stacionární signály vede metoda RLS ke stejným koeficientům jako Wienerův filtr, pro nestacionární signály se k tomuto řešení přibližuje a iterativně se přizpůsobuje změnám vstupních signálů.

Metoda RLS minimalizuje funkci C (z anglického slova „cost“):

$$C(h_{\mathrm{n}}) = \sum_{i=0}^{n} \lambda^{n-i} e(n)^2$$

Kde h_{n} jsou koeficienty adaptivního filtru, $e(n)$ je chybový signál získaný odečtem výstupního signálu $y(n)$ od referenčního signálu $d(n)$ a λ je konstantou pro zapomínání starších vzorků $e(n)$. Hodnota konstanty λ se nachází v rozsahu $0 < \lambda \leq 1$.

Algoritmus RLS patří do kategorie filtrů, které berou do úvahy celkovou chybu akumulovanou od začátku výpočtu.

Pokud je λ rovna jedné, mají všechny předchozí vzorky chybového signálu vyrovnaný vliv na celkovou chybu. Pokud je λ blízká nule, mají všechny předchozí vzorky

chybového signálu minimální vliv na celkovou chybu. Konstanta λ tedy snižuje význam a váhu starších vzorků chybového signálu při opakovaném výpočtu celkové chyby.

Mezi největší výhody algoritmu RLS patří rychlost konvergence a nízká zbytková chyba po dosažení optimálního stavu. Největší nevýhodou pak je výpočetní náročnost, složitost výpočtu a vysoké nároky na paměť, jelikož uchovává v paměti i minulé hodnoty chybového signálu.

Více o výpočtu a odvození algoritmu RLS je možné dohledat v [2].

LMS

Obecně platí, že algoritmus LMS „least mean squares“ - metoda nejmenší střední kvadratické chyby - velmi efektivně zjednodušuje celý výpočet a patří mezi nejjednodušší a nejsnadněji implementovatelné algoritmy [4].

LMS algoritmy k hledání optimálních koeficientů přistupují tzv. gradientní metodou nejrychlejšího klesání „the steepest descent method“, k optimálnímu řešení se tedy přibližují postupně, krok po kroku a v rámci každého kroku je zvolena na základě gradientu chybového signálu velikost a směr kroku, o který se koeficienty adaptivního filtru posunou směrem k optimálnímu řešení.

Nové koeficienty adaptivního filtru jsou pomocí LMS algoritmu vypočítány následovně, jelikož gradient chybového signálu je závislý i na velikosti signálu vstupního.

$$h_{\mathrm{k}}(n+1) = h_{\mathrm{k}}(n) + 2\mu e(n)x(n) \qquad k = 0, 1, 2...N-1$$

Kde $h_{\mathrm{k}}(n+1)$ je nová hodnota daného koeficientu, $h_{\mathrm{k}}(n)$ je jeho stávající hodnota, $e(n)$ je aktuální hodnota chybového signálu a $x(n)$ odpovídající vzorek signálu vstupního.

Konstanta μ je názývána velikostí adaptačního kroku, pomocí této konstanty je totiž možné regulovat velikost kroku a tím i rychlost, s jakou adaptivní algoritmus konverguje k optimálnímu řešení.

Role této konstanty je dvousečná, protože bude-li velikost adaptačního kroku příliš velká, budou výsledné koeficienty pouze oscilovat okolo optimálního řešení, nikdy jej nedosáhnou a konečná chyba konvergence bude příliš vysoká.

Literatura [2] uvádí, že velikost adaptačního kroku μ se má nacházet v rozsahu $0 < \mu < \frac{2}{\lambda_{\max}}$, kde $\lambda_{\max}$ je největší vlastní číslo autokorelační matice vstupního signálu $x(n)$.

Algoritmus LMS patří do kategorie filtrů, které se řídí pouze aktuální hodnotou chybového signálu.

Mezi největší výhody algoritmu LMS patří jeho jednoduchost, snadná implementace a nenáročnost na výkon a paměť, mezi nevýhody pak pomalejší konvergence a vyšší konečná chyba konvergence po dosažení optimálního řešení.

Výhody algoritmu LMS pro využití v rámci této práce převažují jeho nevýhody a proto byl zvolen pro realizaci následujícího modelu i výsledného programu.

11 Model ANC systému - Simulink

V této kapitole je postupně vytvořen model, na jehož základě je postaven program, který je v rámci této práce realizován. Na jednotlivých simulacích je postupně ukázáno, jak se jeho jednotlivé části a algoritmy chovají a jaký vliv mají na chování celého modelu.

Měření přenosu v rámci modelu

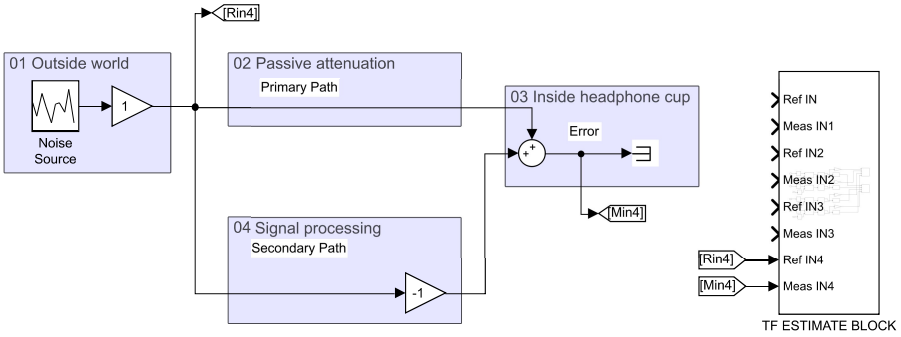
K měření přenosu jednotlivých sekcí modelu využíváme blok TF Estimate, který je popsán kapitole 8.1.1. Tento blok provádí přesný odhad až čtyř různých přenosů a pro každý z měřených přenosů je potřeba poskytnout dva různé signály, referenční a měřící - viz 8.2.

11.1 Základy ANC

Než se vrhneme do návrhu modelu sluchátek s ANC, je nutné si na příkladech ukázat základní stavební bloky takového modelu. I když to bude na první pohled vypadat, že zde realizujeme systém s hybridní topologií (viz 4), není tomu tak. I když zde stejně jako u hybridní topologie používáme dva mikrofony, vnitřní a vnější, jedná se o topologii dopřednou (feed-forward). Vnitřní mikrofon totiž není zdrojem pro odečtový signál, ale slouží pouze jako reference pro úpravu koeficientů adaptivního filtru, realizujícího dopřednou topologii.

11.1.1 SIM 01 - popis jednotlivých částí modelu

Nejjednodušší model dopředného ANC systému vypadá následovně a skládá se ze zdroje šumového signálu, několika signálových cest a vnitřního prostoru sluchátka, v němž se jednotlivé signály setkávají a sčítají.



Obr. 11.1: SIM 01 - popis jednotlivých částí modelu

01 Okolní šum

Cílem ANC obvodu je potlačit okolní šum pronikající do vnitřního prostoru sluchátka. Tento šum je v modelu simulován pomocí zdroje pseudonáhodného, spektrálně vyváženého šumu.

02 Primární zvuková cesta

Pojmem primární zvuková cesta je myšlena akustická cesta, kterou se dostává okolní šum do vnitřního prostoru sluchátka. Tato akustická cesta je v reálném světě také označována jako pasivní útlum sluchátka, protože při průchodu hmotou sluchátka jsou vyšší frekvence okolního šumu utlumeny. V tomto nejjednodušším modelu je však přenos této primární cesty roven jedné.

03 Sekundární zvuková cesta

Snížení výsledné úrovně šumu při realizaci dopředné topologie docilujeme tak, že do obvodu k primární zvukové cestě přidáváme další, sekundární zvukovou cestu. Zdrojem této zvukové cesty by měl být signál, který je korelovaný s okolním šumem, který se snažíme potlačit. Při dopředné topologii je okolní šum snímán vnějším mikrofonem a následně zpracováván sekundární zvukovou cestou. Výstupní signál sekundární zvukové cesty je pak pomocí reproduktoru přehráván do vnitřního prostoru sluchátka. V tomto nejjednodušším modelu volíme přenos sekundární zvukové cesty opět roven jedné.

04 Vnitřní prostor sluchátka

Tím, že jsou sluchátka uzavřená, obepínají ušní boltec a doléhají na hlavu posluchače, je vytvořen oddělený akustický prostor, který nazýváme vnitřním prostorem sluchátka. V tomto prostoru dochází k sumaci dvou akustických signálů, které se do tohoto prostoru dostávají pomocí primární a sekundární zvukové cesty. Je třeba si uvědomit, že z principu jde vždy o sumaci, nikoli o odečet, proto je nutné výstup sekundární zvukové cesty invertovat, aby se v akustické doméně od výstupu primární zvukové cesty odečetl.

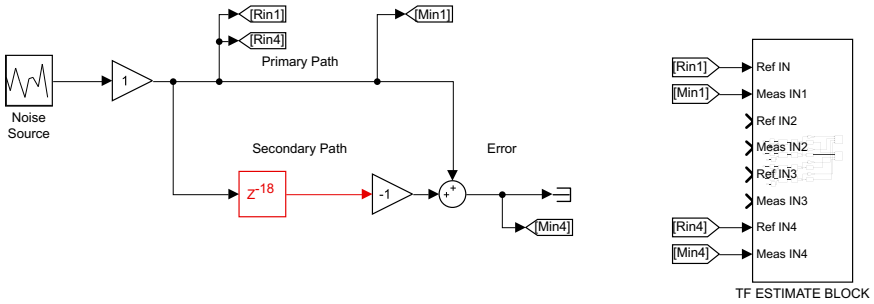
Ve vnitřním prostoru sluchátka se mimo reproduktoru a lidského ucha nachází také vnitřní mikrofon, který snímá zbytkový signál po odečtu obou příchozích signálů (primárního a sekundárního). Tomuto zbytkovému signálu říkáme chybový signál, jelikož reprezentuje chybu výsledného odečtu.

Výsledný odečet

Nikoho jistě nepřekvapí, že pokud zvolíme přenos primární i sekundární cesty roven jedné a výstup sekundární cesty invertujeme, výsledný odečet bude maximální - není jej proto možné ani změřit. Pokud bychom chtěli vyjádřit relativní přenos okolního šumu do prostoru sluchátka, jednalo by se o $-\infty$ dB.

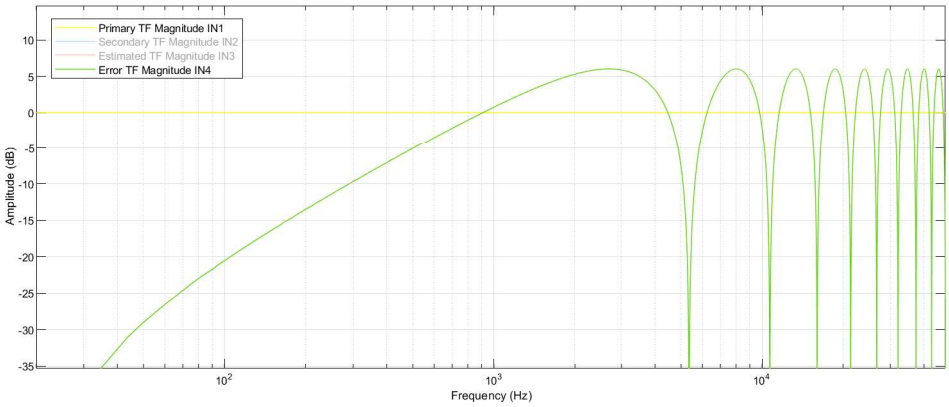
11.1.2 SIM 02 - vliv zpoždění sekundární cesty

Vzhledem k tomu, že pro signálové zpracování v rámci sekundární zvukové cesty používáme digitální DSP čip, vnášíme do této signálové cesty časové zpoždění, které odpovídá latenci vstup-výstup vývojové desky tohoto DSP čipu. Toto zpoždění je při vzorkovacím kmitočtu 96 kHz rovno $0,19$ ms, což odpovídá 18 vzorkům. Vliv tohoto zpoždění na výsledný odečet je naprosto zásadní a znázorňuje jej následující model.

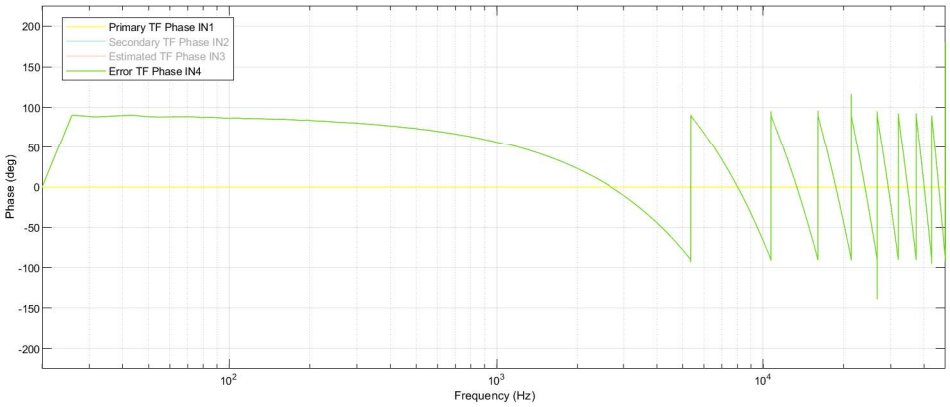


Obr. 11.2: SIM 02 - vliv zpoždění sekundární cesty - model

Pokud bychom výstup sekundární cesty neinvertovali, přenos okolního šumu do prostoru sluchátka (označen jako "Error TF") by v tomto případě měl charakter hřebenového filtru. Jelikož je však výstup sekundární zvukové cesty invertován, tento přenos má charakter inverzního hřebenového filtru, viz následující měření. Z měření je patrné, že při tomto časovém zpoždění sekundární cesty je možné docílit širokopásmového odečtu pouze v kmitočtové oblasti do cca 1-2 kHz a na vyšších kmitočtech bude výsledný odečet mít vždy charakter hřebenového filtru.



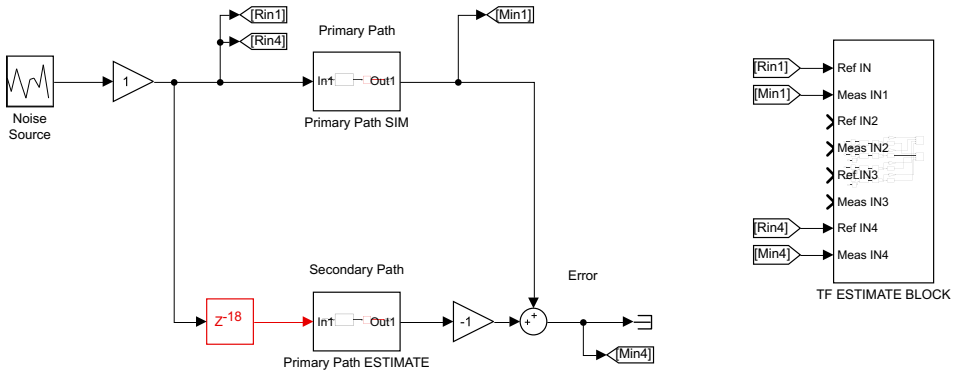
Obr. 11.3: SIM 02 - vliv zpoždění sekundární cesty - amplituda



Obr. 11.4: SIM 02 - vliv zpoždění sekundární cesty - fáze

11.1.3 SIM 03 - přenos primární zvukové cesty

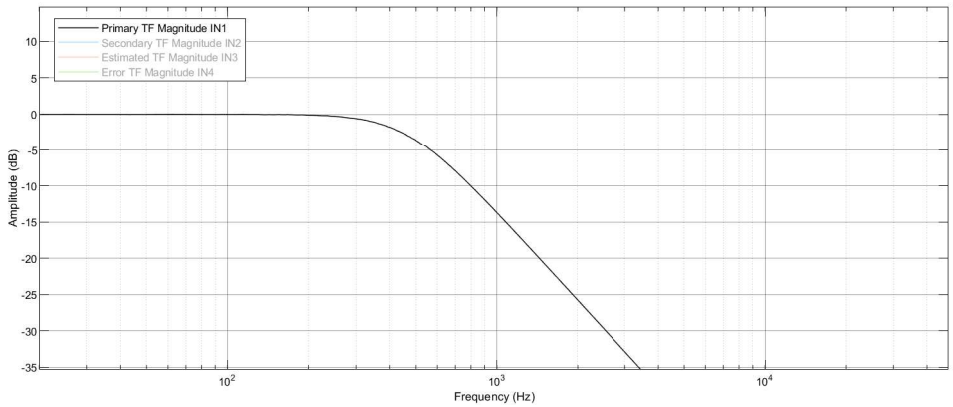
Pro další práci s tímto modelem je nutné již pracovat s reálným přenosem primární zvukové cesty. Tento přenos byl změřen v sekci 9.4 a pro účely modelu jej lze aproximovat vhodně zvoleným filtrem typu dolní propusti.



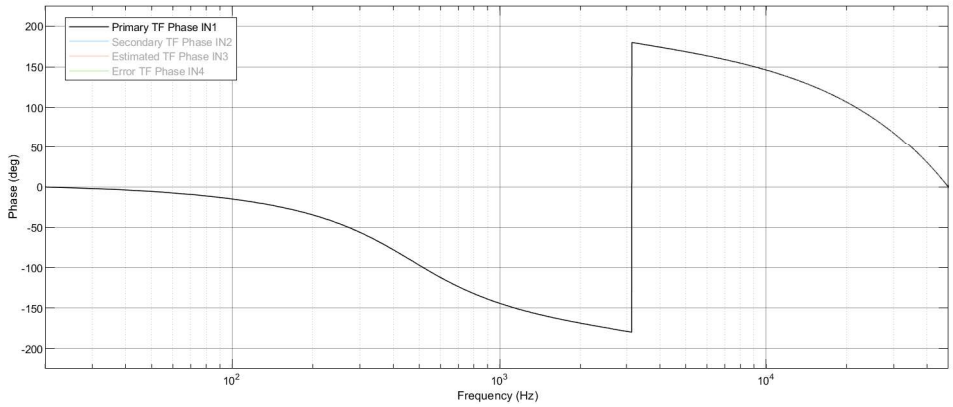
Obr. 11.5: SIM 03 - přenos primární zvukové cesty - model

Pro optimální odečet obou signálu by měl přenos sekundární zvukové cesty odpovídat přenosu cesty primární, jakým způsobem toho lze docílit je popsáno dále.

Přenos modelované primární zvukové cesty je v následujícím měření popsán jako „Primary TF“.



Obr. 11.6: SIM 03 - přenos primární zvukové cesty - amplituda



Obr. 11.7: SIM 03 - přenos primární zvukové cesty - fáze

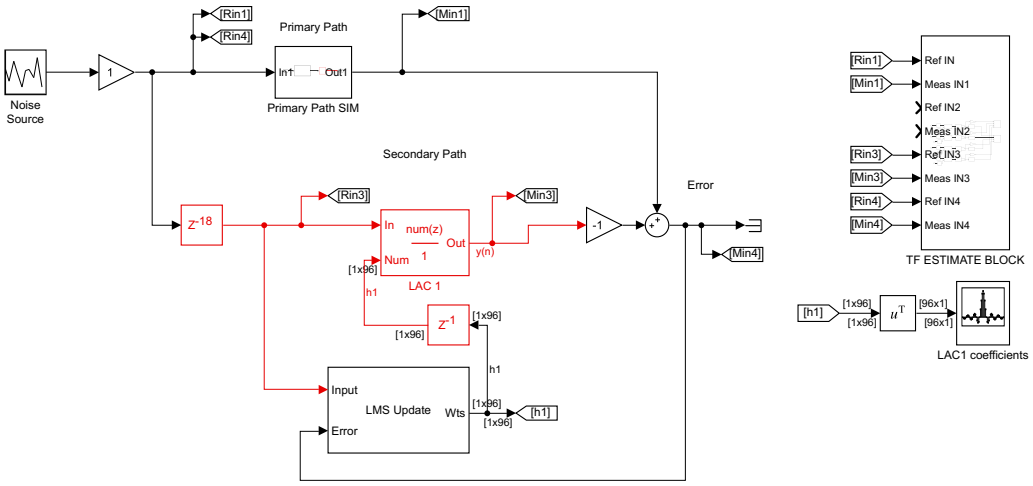
11.1.4 SIM 04 - aproximace přenosu primární zvukové cesty pomocí adaptivního filtru

Jak již bylo řečeno, pro optimální odečet obou signálů je důležité, aby přenos sekundární zvukové cesty odpovídal přenosu cesty primární. Do sekundární zvukové cesty byl tedy vložen FIR filtr, který by měl aproximovat přenos primární zvukové cesty. Tento filtr je v modelu značen jako LAC1, zkratka vychází z anglického pojmu pro adaptivní filtr s proměnnými koeficienty - linear adaptive combiner.

Získání koeficientů LAC1

Nejjednodušším způsobem, jak získat koeficienty LAC1 není přímý návrh filtru pomocí dostupných nástrojů (např. Filter Designer v prostředí MATLAB - viz sekce 8.1). Tyto nástroje totiž navrhují symetrické FIR filtry s lineární fází. Nevýhodou takových FIR filtrů je, že do obvodu vnášejí další dodatečné zpoždění, což pro tuto aplikaci není vhodné. Existují metody, jak přepočítat FIR filtr s lineární fází na FIR filtr s minimální fází, ale tyto metody jsou výpočetně velmi náročné, proto ani tato cesta není vhodným řešením.

Nejjednodušším způsobem, jakým lze získat koeficienty tohoto filtru je použít adaptivní filtr v zapojení pro identifikaci systému (viz 10.1.2). Implementaci tohoto filtru znázorňuje následující model.



Obr. 11.8: SIM 04 - aproximace přenosu primární zvukové cesty - model

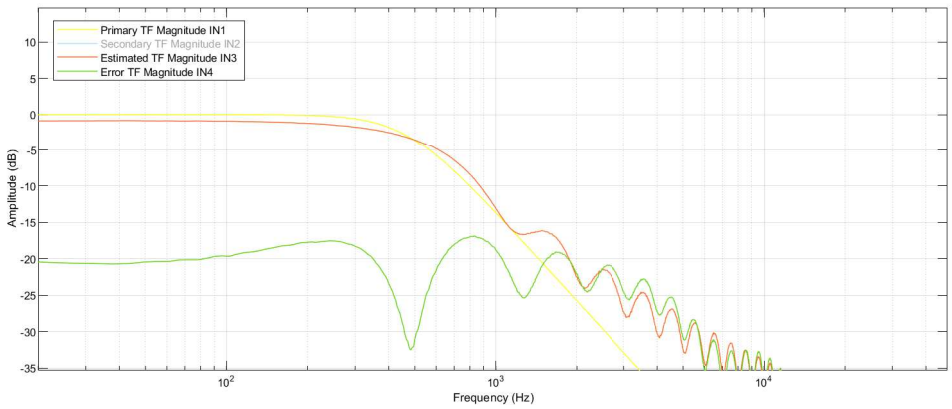
Implementovaný FIR filtr je 95. řádu, je tedy charakterizován 96 koeficienty. Vektor těchto koeficientů je v modelu i v navazujícím programu označen jako „h1“. Množství koeficientů bylo určeno na základě analýzy výpočetních prostředků dostupných v použitém DSP čipu - pokud bychom se snažili o implementaci filtru s více koeficienty, program by již nestačil tyto koeficienty a další příslušné zásobníky číst a zapisovat zpět do paměti. Omezený počet koeficientů s sebou přináší různá další omezení, avšak ta se projeví až později.

Koeficienty jsou v každém cyklu aktualizovány pomocí bloku LMS Update, který reprezentuje implementaci adaptačního algoritmu LMS a na základě vstupního a chybového signálu aktualizuje koeficienty (více o funkci tohoto bloku si lze přečíst v sekci 10.2.3). Zpoždění o jeden vzorek vložené mezi blok LMS Update a LAC1 modeluje fakt, že výpočet nových koeficientů probíhá vždy pro následující cyklus, LAC1 tedy fitruje vstupní signál na základě koeficientů určených na konci minulého cyklu.

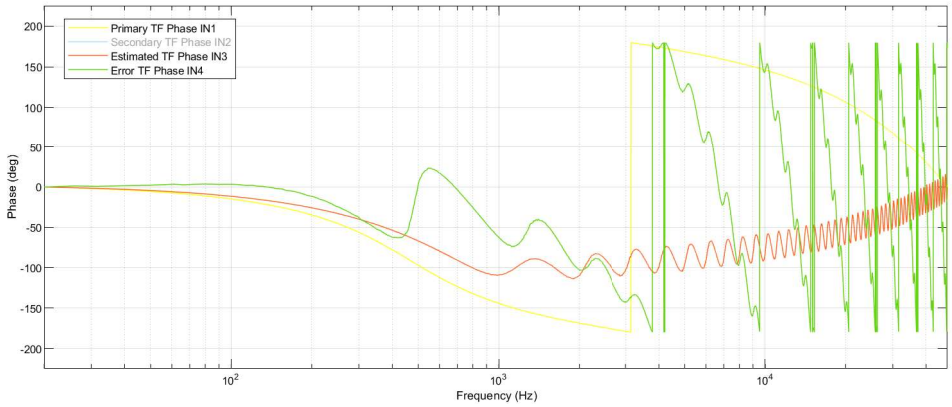
Součástí sekundární zvukové cesty je i nadále zpožďovací blok reprezentující latenci signálového zpracování.

Následující měření znázorňuje přenos primární zvukové cesty (označený „Primary TF“), přenos fitru LAC1, který tento přenos primární zvukové cesty aproximuje (označený „Estimated TF“) a také přenos okolního šumu do vnitřního prostoru sluchátka po ukončení adaptace (označený „Error TF“) - pozor tento výsledný přenos neodpovídá výsledné skutečnosti vlivem latence - viz sekce 11.1.5.

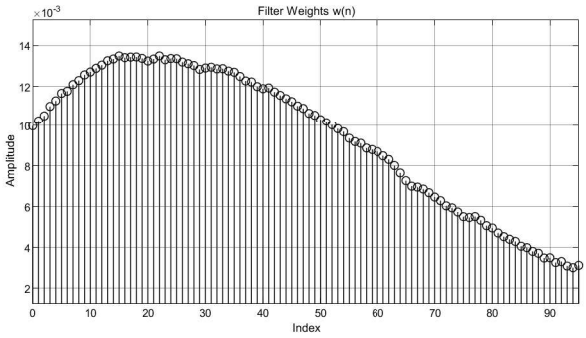
Zároveň se zde nachází graf znázorňující výsledné hodnoty jednotlivých koeficientů LAC1 po identifikaci přenosu primární zvukové cesty s využitím adaptivního filtru.



Obr. 11.9: SIM 04 - aproximace přenosu primární zvukové cesty - amplituda



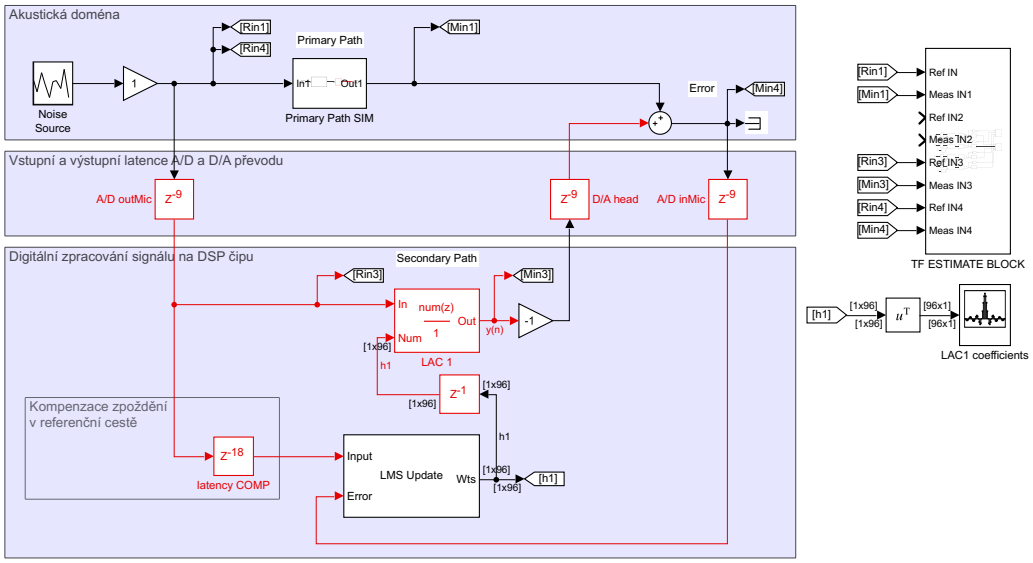
Obr. 11.10: SIM 04 - aproximace přenosu primární zvukové cesty - fáze



Obr. 11.11: SIM 04 - aproximace přenosu primární zvukové cesty - koeficienty LAC1

11.1.5 SIM 05 - distribuce latence a její následná kompenzace

Až do této chvíle jsme uvažovali latenci přítomnou v obvodu jako jeden homogenní blok. Pro další vývoj modelu je však zásadní podrobněji analyzovat, na jakých místech se tato latence nachází a jak je s ní ve skutečnosti nutné dále pracovat. Nejprve je nutné si uvědomit, v jaké doméně se odehrávají jednotlivé části modelu.



Obr. 11.12: SIM 05 - distribuce latence a její následná kompenzace - model

Akustická doména

Jak již bylo zmíněno v 11.1.1, zdroj šumu se nachází v bezprostředním okolí sluchátka - jde tedy o akustický signál. Tento signál prochází hmotou sluchátka do jeho vnitřního prostoru, což je nazýváno primární zvukovou cestou. Do vnitřního prostoru sluchátka je pomocí reproduktoru vyzářen další akustický signál přicházející sekundární zvukovou cestou. Tyto dva akustické signály se zde sečtou a výsledkem je jakýsi zbytkový signál, který vniká do zvukovodu posluchače.

Převod z akustické do digitální domény

Akustický signál vně i uvnitř sluchátka je zaznamenán mikrofonem a jako elektrický signál následně přiveden na vstupní A/D převodník, který jej převede na signál číslicový. Tento převod vnáší do obvodu jistou latenci. Podobným způsobem je výstupní číslicový signál sekundární zvukové cesty převeden výstupním D/A převodníkem

na signál elektrický a ten je následně vyzářen reproduktorem jako signál akustický. Obdobně je i zde při D/A převodu do obvodu vnášena jistá latence.

Naše předchozí měření ukázalo, že součet vstupní a výstupní latence je při vzorkovacím kmitočtu 96 kHz právě výše zmíněných 18 vzorků. Pro účely modelu je tato latence rozdělena na dvě poloviny a aproximována zpožděním 9 vzorků při každém z převodů.

Signál z vnějšího mikrofonu je v modelu označen „outMic“ a signál z vnitřního mikrofonu je v modelu označen „inMic“. Výstupní signál do reproduktoru sluchátka je označen „head“.

Kompenzace zpoždění

Vzhledem k tomu, že na DSP čipu implementujeme blok LMS Update, pro jeho správnou funkci je nutné přidanou latenci obvodu kompenzovat. Proč? Protože blok LMS Update aktualizuje koeficienty LAC1 na základě dvou signálů. Jedním je vstupní signál do LAC1 a druhým je výsledný chybový signál. Problém nastane ve chvíli, kdy by tento algoritmus aktualizoval koeficienty na základě chybového signálu, který by byl zpožděn o 18 vzorků vůči signálu vstupnímu. Taková adaptace by probíhala na základě špatných signálů a velice rychle by se stala nestabilní.

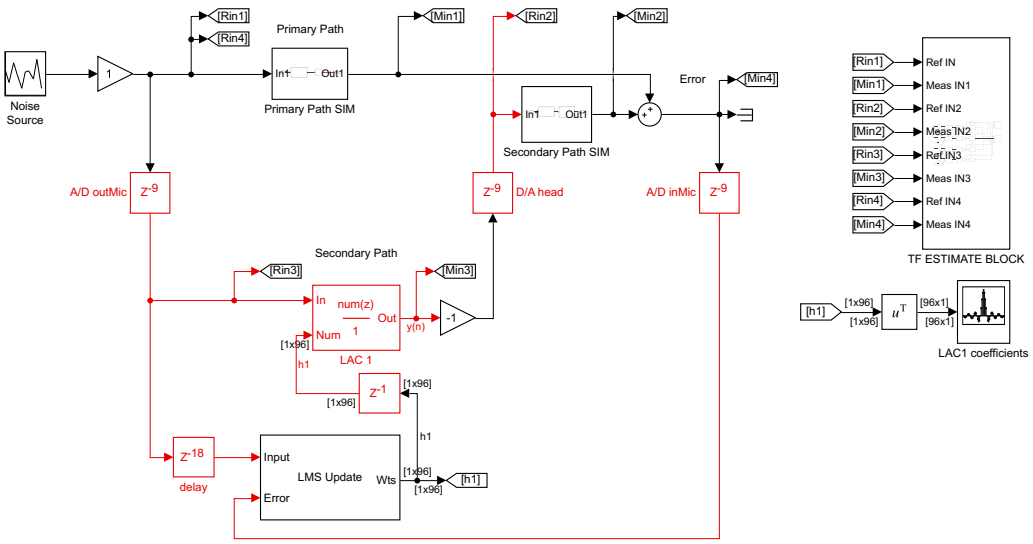
Protože chybový signál bude oproti výstupu LAC1 vždy opožděn o 18 vzorků, je třeba vstupní signál LAC1 pro účely bloku LMS Update také zpozdit o 18 vzorků. Adaptace pak sice probíhá opožděně, avšak na základě správných signálů a tím pádem vede k optimálnímu řešení.

Jednotlivé zpožďovací bloky na finálních pozicích zobrazuje model SIM 05 (11.1.5), jehož fungování je naprosto identické s fungováním modelu SIM 04 (11.1.4).

11.1.6 SIM 06 - přenos sekundární zvukové cesty a jeho aproximace

V předchozích simulacích jsme nebrali v potaz kmitočtově závislý přenos samotného reproduktoru uvnitř sluchátka. Bylo tomu tak, aby bylo možné jednoduše ukázat funkci dalších částí obvodu. Pro další vývoj už je nutné vzít do úvahy i tento kmitočtově závislý přenos reproduktoru sluchátka, jelikož kmitočtové zkreslení, které do obvodu vnáší není zanedbatelné a má zásadní vliv na funkci bloku LMS Update.

Simulovaný přenos sekundární zvukové cesty je modelován dle naměřeného přenosu v sekci 9.4 pomocí kombinace IIR filtrů a grafického ekvalizéru, který je součástí Audio toolboxu dostupného v prostředí Simulink. Simulovaný přenos je v důsledku modelace pomocí grafického ekvalizéru v porovnání s reálným přenosem mírně zvlněn, což však nemá žádný zásadní vliv na názornost a funkci modelu.



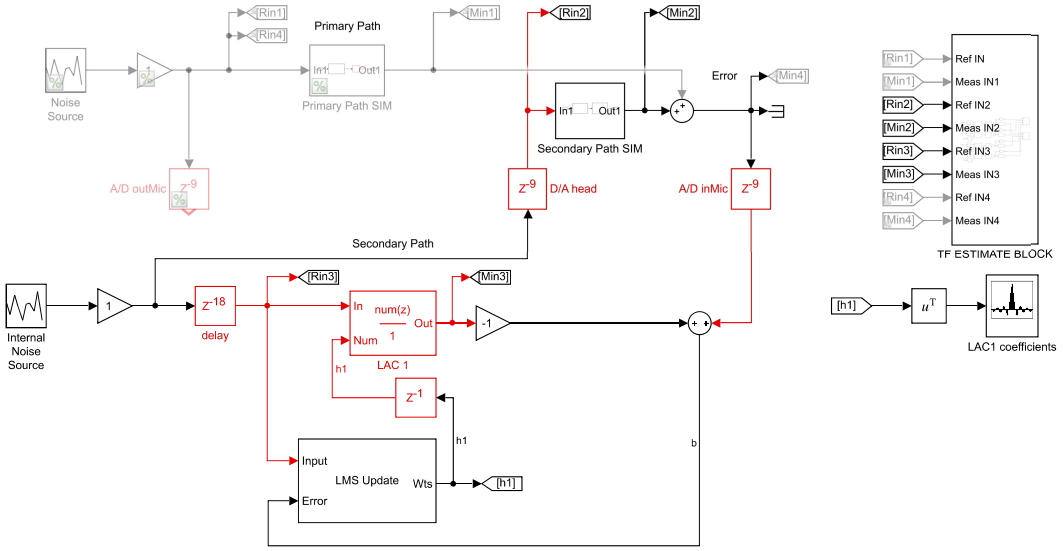
Obr. 11.13: SIM 06 - přenos sekundární zvukové cesty - model

Přenos sekundární zvukové cesty se v modelu nachází na místě přenosu reproduktoru, protože přenos sekundární zvukové cesty je právě přenosem reproduktoru definován. Ve výsledné podobě je tento přenos doplněn o adaptivní FIR filtr, který umožňuje přenos reproduktoru kompenzovat pomocí inverzní filtrace.

Zde je nutné zmínit, že adaptivní filtr zapojený tímto způsobem je nestabilní a nekonverguje k optimálnímu řešení, naopak je náchylný k oscilaci. Řešením je tzv. „filtered-x LMS“ zapojení, viz následující sekce.

Aproximace přenosu sekundární zvukové cesty

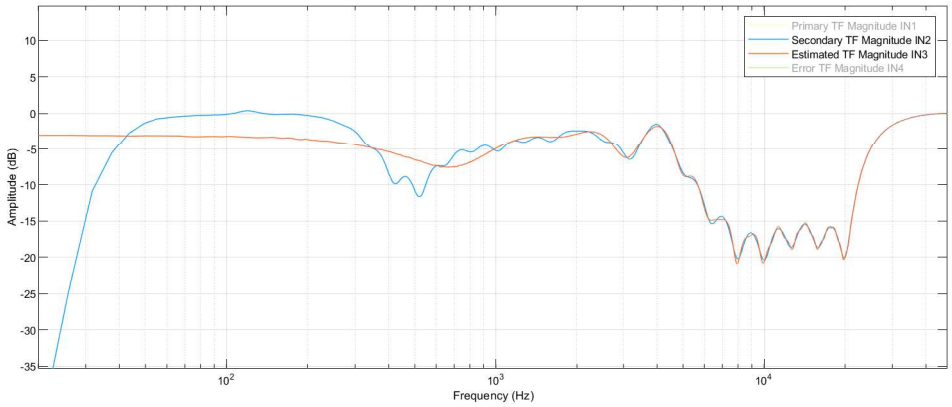
Pro aproximaci přenosu sekundární zvukové cesty je podobně jako v případě aproximace zvukové cesty primární možné použít adaptivní filtr v zapojení pro identifikaci systému (viz 10.1.2). Oproti aproximaci primární zvukové cesty je však měřící širokospektrální šum generován interně v rámci DSP čipu a přehráván do reproduktoru sluchátka jako akustický signál. Tento signál je následně zaznamenán vnitřním mikrofonem a přiveden zpět na vstup DSP jako signál „inMic“. Následně je od signálu „inMic“ odečten výstupní signál LAC1, čímž vzniká chybový signál „b“, který je použitý pro adaptaci koeficientů LAC1. Toto zapojení znázorňuje následující model.



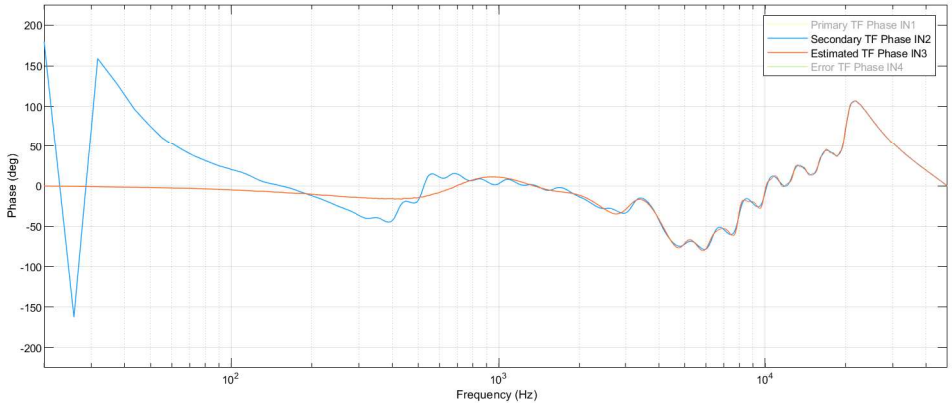
Obr. 11.14: SIM 06 - aproximace přenosu sekundární zvukové cesty - model

Simulovaný přenos sekundární zvukové cesty je v následujícím měření znázorněn modrou křivkou označen jako „Secondary TF“ a jeho aproximace je znázorněna červenou křivkou a označena „Estimated TF“.

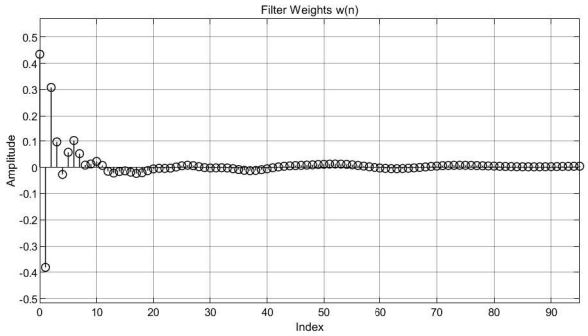
Toto měření také ilustruje vliv nedostatečné délky FIR filtru na přesnost aproximace na nízkých kmitočtech - viz sekce 11.1.8.



Obr. 11.15: SIM 06 - aproximace přenosu sekundární zvukové cesty - amplituda



Obr. 11.16: SIM 06 - aproximace přenosu sekundární zvukové cesty - fáze

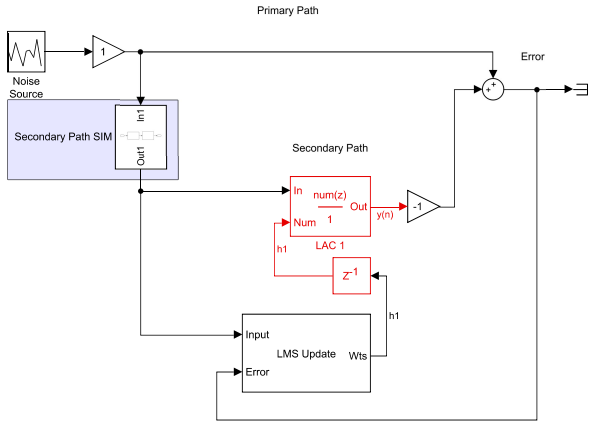


Obr. 11.17: SIM 06 - aproximace přenosu sekundární zvukové cesty - koeficienty LAC1

11.1.7 SIM 07 - filtered-x LMS

Zapojení, které se nazývá dle [19] a [5] filtered-x LMS je modifikací předchozího zapojení, a to takovým způsobem, aby obdobně jako při řešení otázky distribuce latence dostávala funkce LMS Update pro každý chybový vzorek odpovídající sadu vzorků vstupních. V tuto chvíli je však chybový vzorek místo časové chyby zatížen kmitočtovým zkreslením přenosu sekundární zvukové cesty - řešením je obdobně jako v případě řešení přidané latence toto kmitočtové zkreslení aplikovat i na vstupní vzorky funkce LMS Update. K tomuto řešení se lze dobrat následující úvahou.

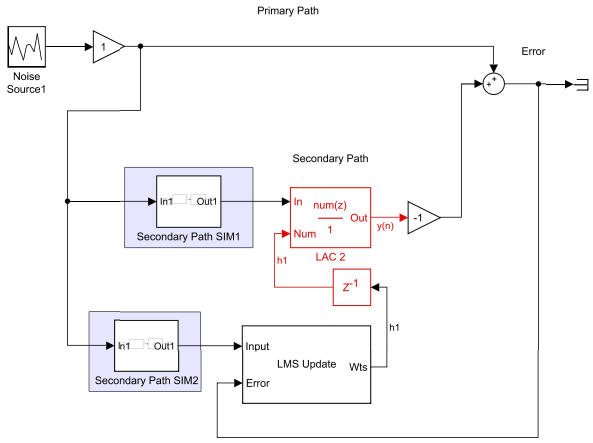
Distribuce přenosu sekundární zvukové cesty



Obr. 11.18: SIM 07 - distribuce kmitočtového zkreslení sekundární cesty - A

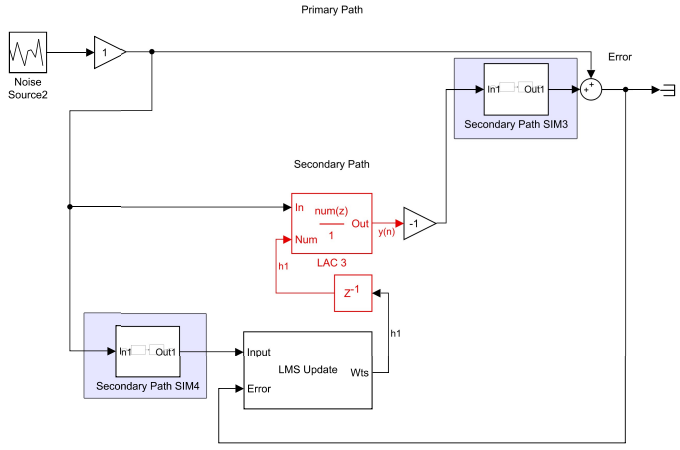
Nejprve si představme situaci, kdy filtr reprezentující přenos reproduktoru umístíme na začátek sekundární cesty. Tento filtr tak sice ovlivňuje přenos celé sekundární cesty, ale nemá žádný vliv na funkci a stabilitu adaptivního filtru, jelikož filtr LAC1 i blok LMS Update mají na svém vstupu stejný signál.

Dalším krokem je rozdělit tento filtr (označený „Secondary Path SIM“) na dva (identické) filtry, jeden z nich umístit na vstup LAC1 a druhý na vstup LMS Update. Platí, že tato změna nemá na funkci obvodu žádný vliv.



Obr. 11.19: SIM 07 - distribuce kmitočtového zkreslení sekundární cesty - B

Za předpokladu, že ani přenos reproduktoru ani přenos filtru LAC1 nemají nezvratný efekt na vstupní signál (např. nepotlačují některé kmitočtové pásmo pod kritickou úroveň šumu v obvodu), jsou oba sériově zapojené filtry komutativní a je tedy možné zaměnit jejich vzájemné pořadí.

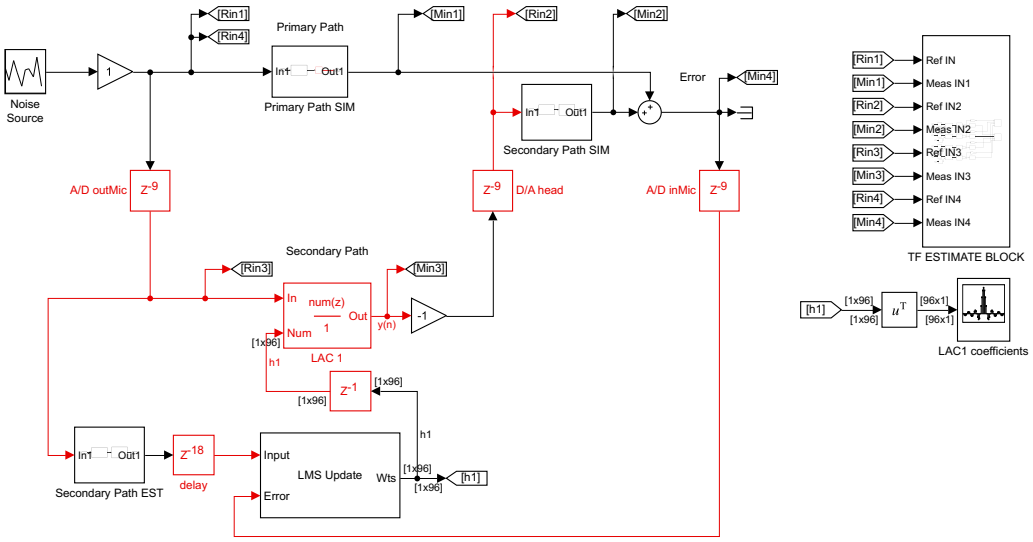


Obr. 11.20: SIM 07 - distribuce kmitočtového zkreslení sekundární cesty - C

Dle [19] je možné dokázat, že za výše uvedených podmínek jsou tyto tři způsoby zakreslení distribuce přenosu sekundární zvukové cesty ekvivalentní a volba kteréhokoli z nich nijak neovlivní funkci zapojení. Na základě této logiky tedy vzniká následující model systému pro potlačení okolního šumu, jenž je v literatuře [19] označován právě jako "filtered-x LMS".

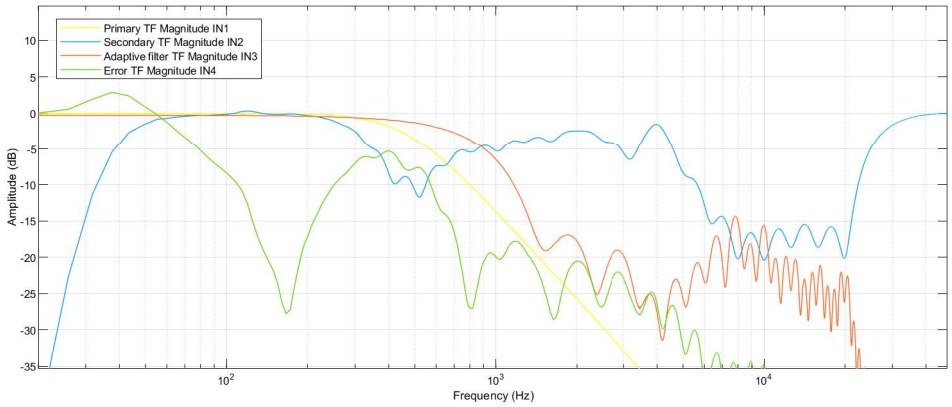
Model filtered-x LMS

Funkci LMS Update je tedy předřazen filtr, jehož přenos je aproximací přenosu sekundární zvukové cesty. Jakým způsobem je možné tento přenos aproximovat je popsáno v SIM 06 11.1.6.

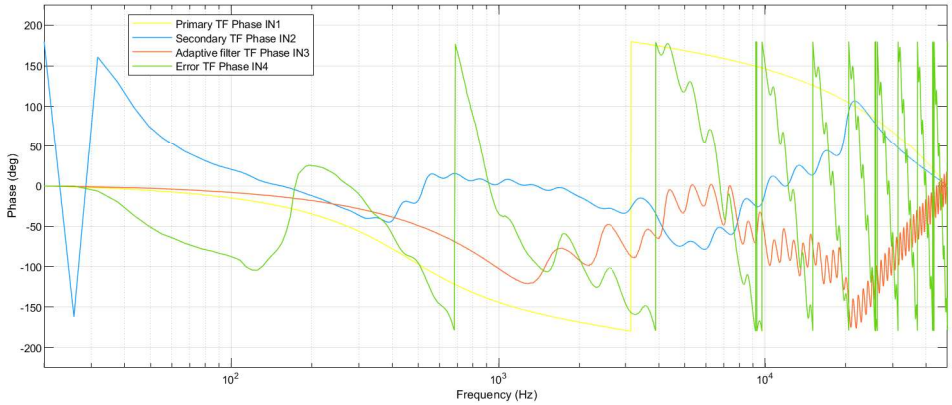


Obr. 11.21: SIM 07 - filtered-x LMS - model

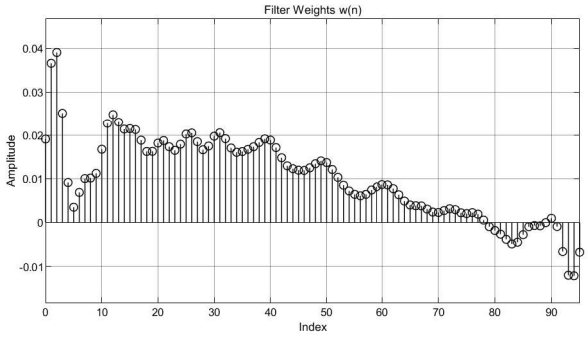
Následující měření zobrazuje přenos LAC1 (červená křivka, označena jako „Adaptive filter TF“). Na základě tohoto měření je zřejmé, že přenos LAC1 je adaptován takovým způsobem, aby zároveň aproximoval přenos primární zvukové cesty a zároveň potlačuje vliv přenosu cesty sekundární pomocí inverzní filtrace (převážně v kmitočtové oblasti nad 1 kHz). Za pozornost také stojí výsledný útlum okolního šumu (označený „Error TF“) a jeho kmitočtové rozložení.



Obr. 11.22: SIM 07 - filtered-x LMS - amplituda



Obr. 11.23: SIM 07 - filtered-x LMS - fáze



Obr. 11.24: SIM 07 - filtered-x LMS - koeficienty LAC1

11.1.8 SIM 08 - IIR kompenzace nedostatečné délky FIR filtru

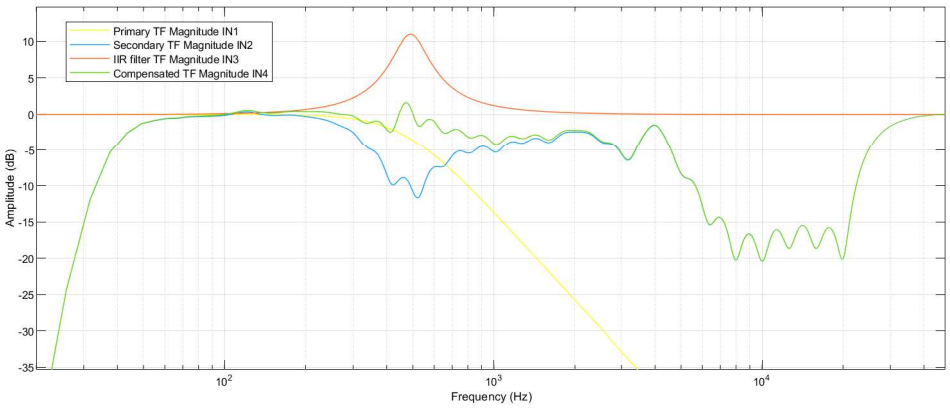
Jak lze vyčíst z předchozích simulací, schopnost FIR filtru s minimální fází, definovaného pomocí 96 koeficientů, kompenzovat a kopírovat přenosovou charakteristiku končí okolo kmitočtu 1 kHz - 96 vzorků totiž odpovídá při vzorkovacím kmitočtu 96 kHz přesně periodě 1 ms, tedy kmitočtu 1 kHz.

Zároveň je názorně vidět, že v kmitočtové oblasti mezi 400 až 600 Hz není odečet tak efektivní, jak by doopravdy být mohl. Je to způsobeno právě tím, že na těchto kmitočtech již FIR filtr s danou délkou není zcela efektivní. Řešením je pak použít fixní IIR filtr, který v dané kmitočtové oblasti pomůže kompenzovat zkreslení kmitočtového přenosu sekundární zvukové cesty.

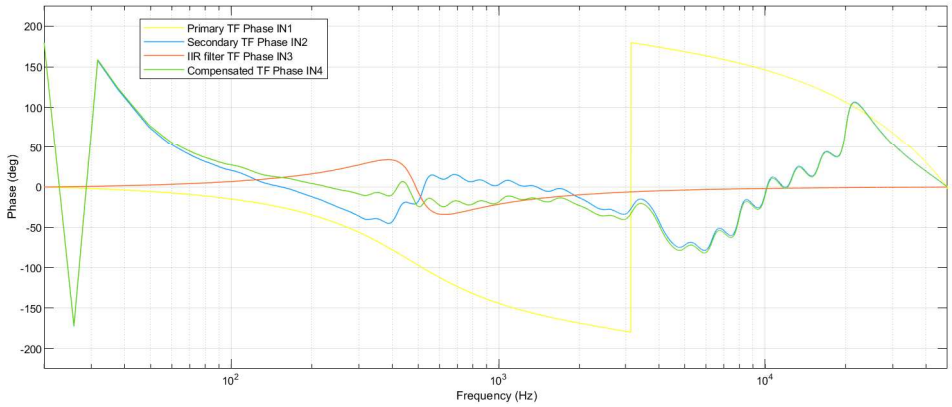
Na následujícím měření vidíte přenos navrhovaného kompenzačního IIR filtru a jeho vliv na výsledný přenos sekundární zvukové cesty. Jedná se o filtr typu peak, navržený v prostředí MATLAB pomocí funkce designParamEQ(), která je součástí Audio toolboxu. Parametry tohoto filtru jsou následující:

- řád filtru: 2
- zesílení filtru: +11dB
- střední kmitočet: 470 Hz (0.0098 pak relativně vůči vzorkovacímu kmitočtu)
- relativní šířka pásma: 0.005

Přenos kompenzačního IIR filtru je zobrazen červenou křivkou, přenos sekundární zvukové cesty modrou křivkou a výsledný přenos (sériová kombinace obou předchozích) je zde zobrazen křivkou zelenou.



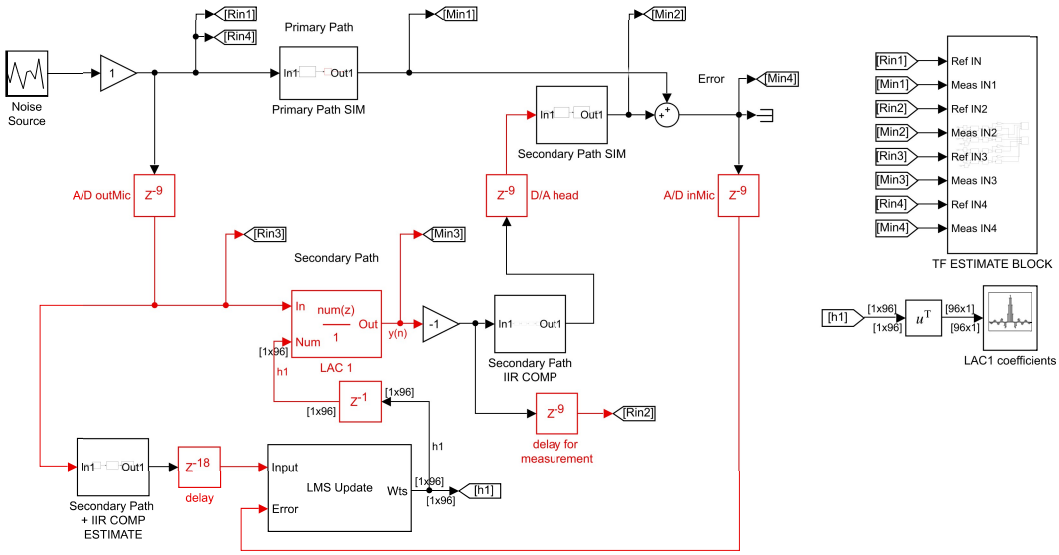
Obr. 11.25: SIM 08 - IIR kompenzace nedostatečné délky FIR filtru - amplituda



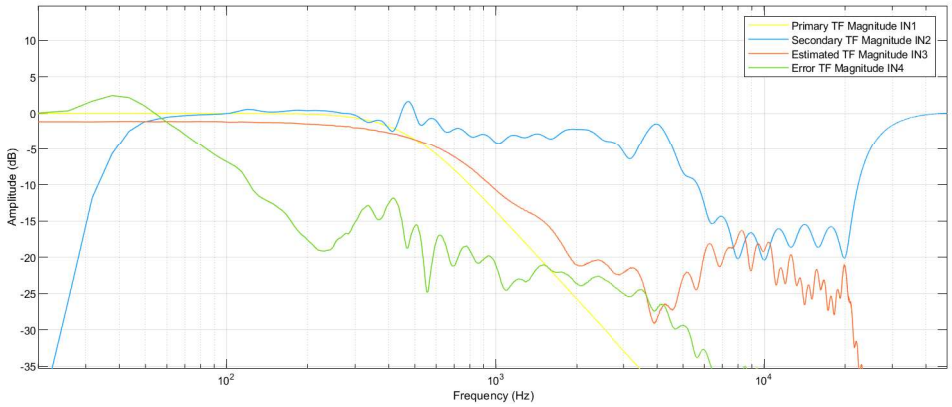
Obr. 11.26: SIM 08 - IIR kompenzace nedostatečné délky FIR filtru - fáze

Výsledný model adaptivního systému filtered-x LMS s IIR kompenzací pak vypadá následovně. V dříve problematické oblasti mezi 400 až 600 Hz je nyní možné pozorovat drastické zlepšení výsledného útlumu okolního šumu.

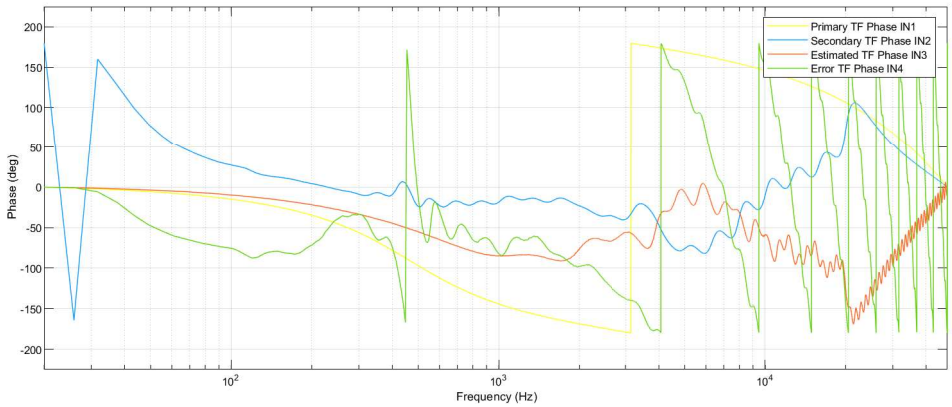
Abychom při měření výsledného přenosu sekundární zvukové cesty včetně kompenzačního filtru eliminovali simulované zpoždění na D/A převodu, srovnatelné zpoždění bylo v tomto modelu pro účely měření vloženo i do cesty referenčního signálu (označeno jako "delay for measurement").



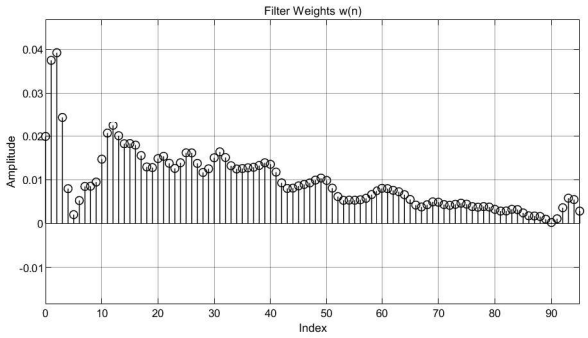
Obr. 11.27: SIM 08 - Filtered-x LMS systém po IIR kompenzaci délky FIR filtru - model



Obr. 11.28: SIM 08 - Filtered-x LMS systém po IIR kompenzaci délky FIR filtru - amplituda



Obr. 11.29: SIM 08 - Filtered-x LMS systém po IIR kompenzaci délky FIR filtru - fáze



Obr. 11.30: SIM 08 - Filtered-x LMS systém po IIR kompenzaci délky FIR filtru - koeficienty LAC1

11.2 Model sluchátek s ANC

Nyní už je jasné, jak funguje algoritmus potlačení šumu filtered-x LMS a je načase tuto metodu aplikovat přímo na námi řešený návrh sluchátek s adaptivním potlačením šumu.

V rámci této práci je konstruován modelový prototyp sluchátek s ANC, proto je realizován pouze jeden kanál za pomoci 2 mikrofonů, 2 předzesilovačů, stereo digitálního DSP čipu a jednoho sluchátka. Přebývá tedy jeden výstup z DSP k volnému použití a při vývoji je možné jej využít pro měření a pozorování jednotlivých fází číslicového zpracování - viz kapitola 12.

Rozdělení do jednotlivých fází

Pro správnou funkci výsledného modelu sluchátek s ANC je nutné provést několik úkonů v předem daném pořadí. Tyto úkony rozdělují fungování výsledného programu na několik samostatných, na sebe navazujících fází. V první fázi je aproximován přenos sekundární zvukové cesty, v druhé fázi je aproximován přenos cesty primární a v průběhu třetí fáze je realizován výsledný model, kde je s použitím adaptivního filtru výsledný odečet okolního šumu optimalizován.

11.2.1 SIM 09 - první fáze - aproximace přenosu sekundární zvukové cesty

Cílem první fáze je identifikovat a aproximovat přenos sekundární zvukové cesty pomocí 96 koeficientů adaptivního filtru. K tomuto účelu lze použít zapojení adaptivního filtru pro identifikaci systému, viz 10.1.2.

V této fázi je nejprve digitálně vygenerován širokopásmový růžový šum, který je přehráván do reproduktoru sluchátka (v modelu označeno jako „head“). Vygenerovaný šum, zpožděný o 18 ms, je dále přiveden na vstup adaptivního FIR filtru (v modelu označen jako „LAC1“) a také na vstup bloku LMS Update.

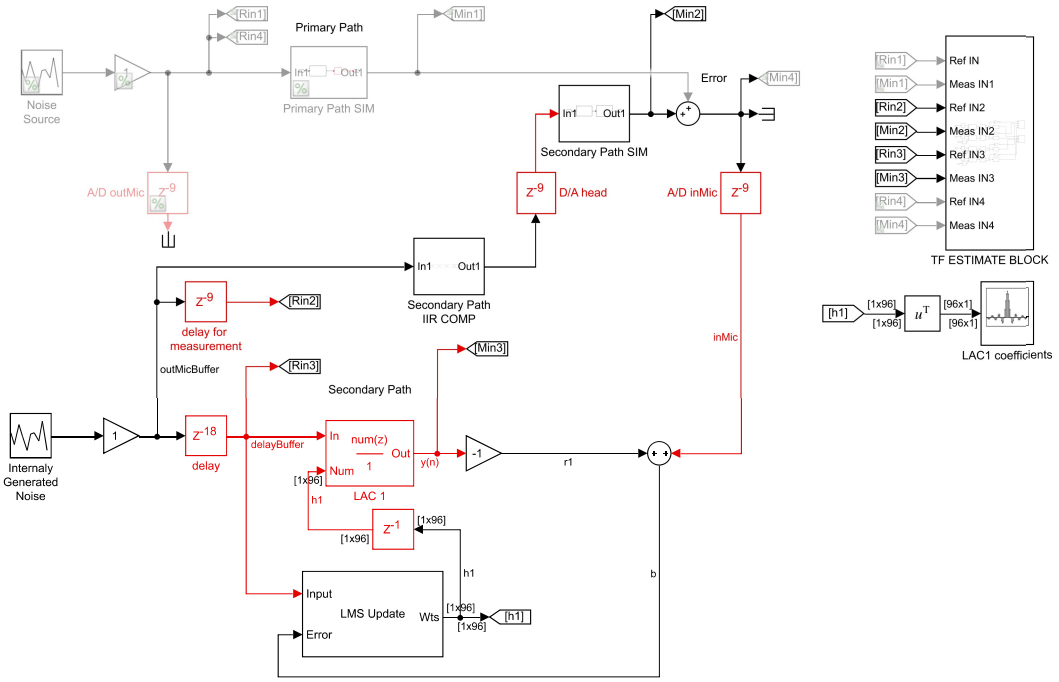
Reproduktor přehrává šum do prostoru sluchátka, kde je tento šum zaznamenán pomocí vnitřního mikrofonu.

Výstupní signál z adaptivního filtru je následně odečten od signálu přicházejícího z mikrofonu uvnitř sluchátka (v modelu označen jako „inMic“) a výsledkem tohoto odečtu je chybový signál (v modelu označen jako „b“). Na základě tohoto chybového signálu pak blok LMS Update provede aktualizaci koeficientů pro budoucí výpočetní cyklus.

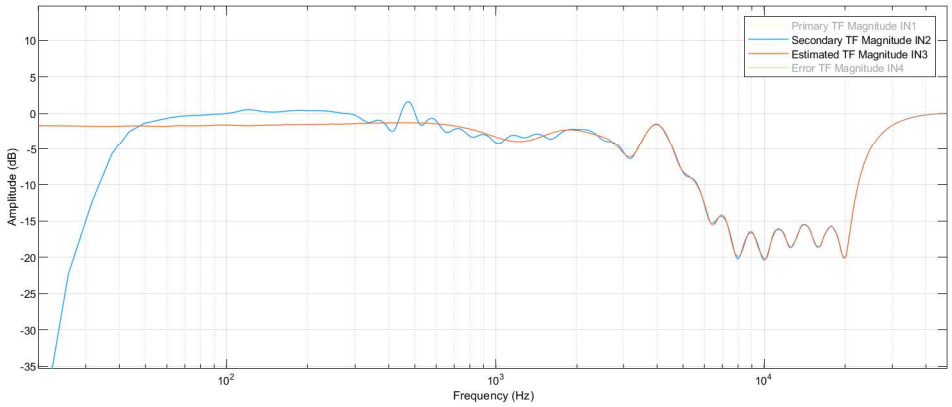
V ideálním případě by měl přenos adaptivního filtru plně aproximovat přenos sekundární cesty. To platí za předpokladu, že je tento filtr definován dostatečným množstvím koeficientů. Jelikož je však v reálném prostředí počet vzorků, které stihne

DSP v požadovaném čase spočítat a uložit do paměti omezen kapacitou datových sběrnic, v rámci optimalizace dostupných zdrojů bylo určeno, že počet koeficientů adaptivního filtru bude 96, jelikož toto číslo poskytuje dostatečný kompromis.

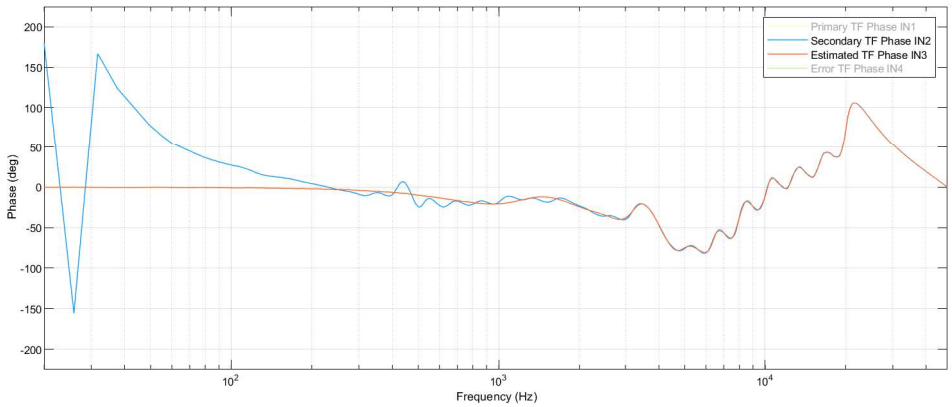
Vektor koeficientů **h1** je po ukončení adaptace uložen k dalšímu, pozdějšímu použití.



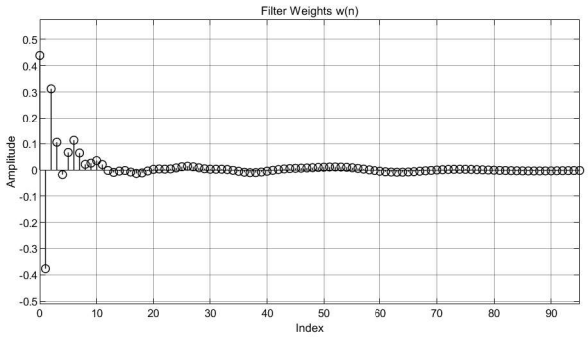
Obr. 11.31: SIM 09 - první fáze - aproximace přenosu sekundární zvukové cesty - model



Obr. 11.32: SIM 09 - první fáze - aproximace přenosu sekundární zvukové cesty - amplituda



Obr. 11.33: SIM 09 - první fáze - aproximace přenosu sekundární zvukové cesty - fáze

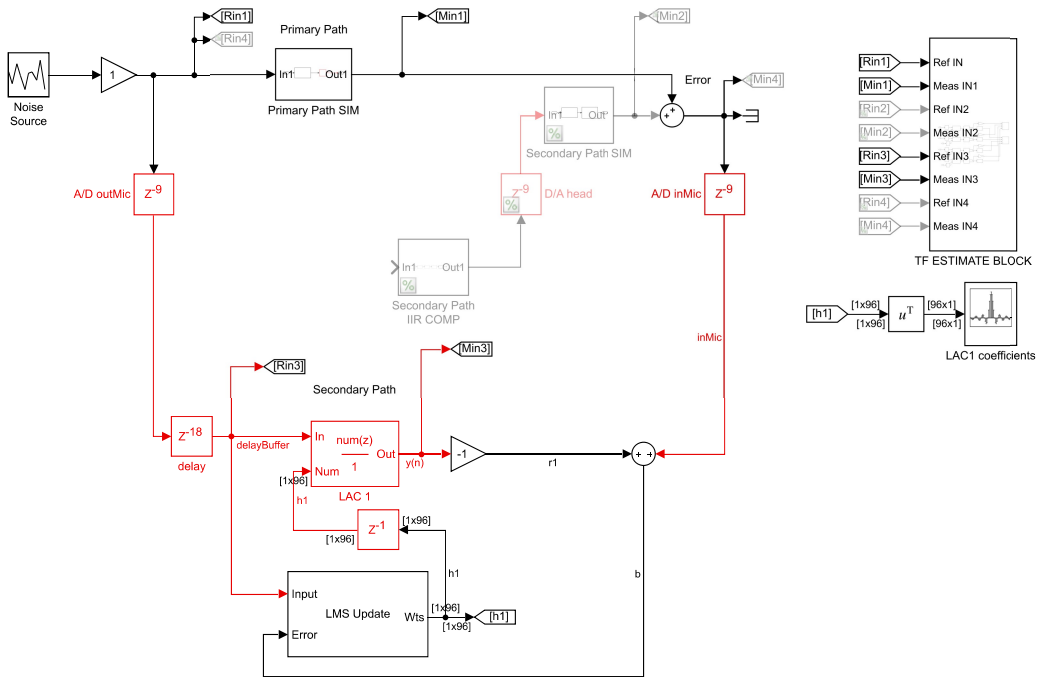


Obr. 11.34: SIM 09 - první fáze - aproximace přenosu sekundární zvukové cesty - koeficienty LAC1

11.2.2 SIM 10 - druhá fáze - aproximace přenosu primární zvukové cesty

Poté, co je první fáze hotová a aproximace přenosu sekundární zvukové cesty je dokončena, následuje fáze druhá - aproximace přenosu cesty primární. Je třeba zmínit, že tuto fázi není vždy nutné do programu zařadit, jelikož za určitých podmínek dokáže její funkci splnit fáze třetí, ale v rámci této práce implementtována je, protože pracujeme s DSP čipem počítajícím v aritmetice s pevnou řádovou čárkou, který není vždy tak flexibilní jako systém pracující s pohyblivou řádovou čárkou. Předběžná aproximace přenosu primární zvukové cesty bez současné realizace odečtu vede k lepší funkci adaptivního filtru v průběhu třetí fáze programu. Během simulace v prostředí Simulink tato fáze nutná není, ale nijak výsledku neuškodí.

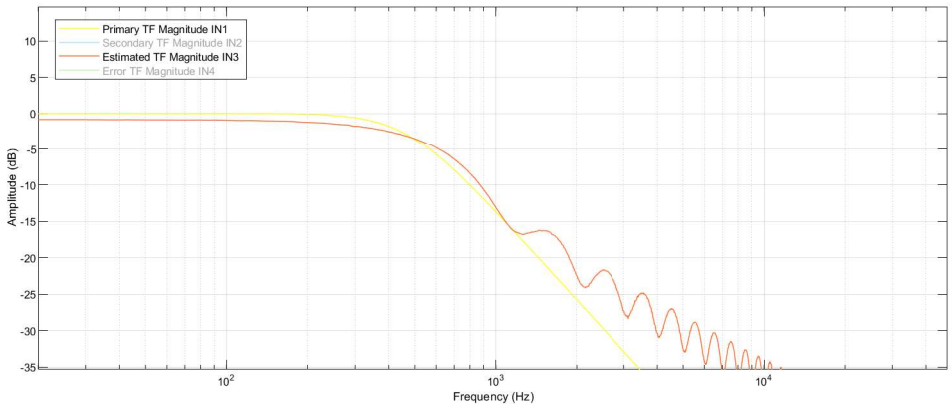
Druhá fáze je velice podobná fázi první, ale narozdíl od ní není do výstupu „head“ přehráván žádný signál. Místo toho je do bezprostředního okolí sluchátka přehráván širokospektrální růžový šum za pomoci externího zdroje. Tento šum je následně zaznamenán pomocí vnějšího a vnitřního mikrofonu. Jejich vzájemný rozdíl je způsoben právě pasivním útlumem sluchátka - tedy primární zvukovou cestou.



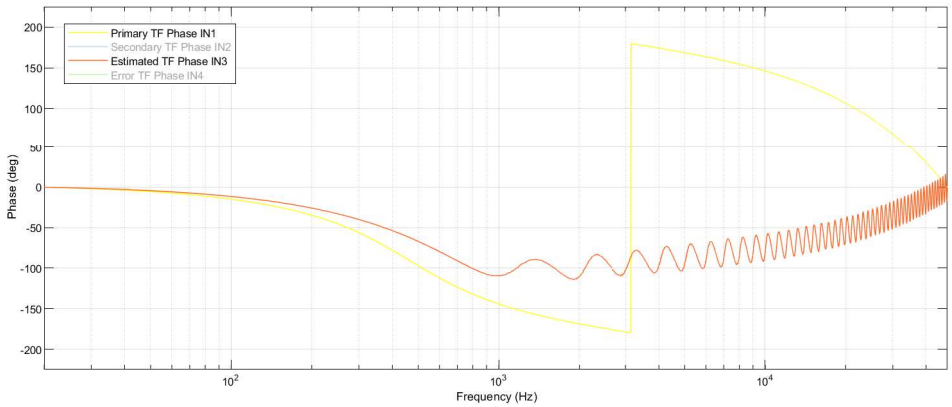
Obr. 11.35: SIM 10 - druhá fáze - aproximace přenosu primární zvukové cesty - model

Za povšimnutí stojí i fakt, že adaptivnímu filtru zůstává i během druhé fáze předřazen zpožďovací zásobník - ve výsledném zapojení (včetně sekundární zvukové cesty) se na tomto místě bude tento zpožďovací zásobník nacházet pro kompenzaci zpoždění v sekundární zvukové cestě a bylo by kontraproduktivní jej při aproximaci přenosu primární zvukové cesty vynechávat. Tento blok nemá na tuto fázi negativní vliv zejména z důvodu, že zmíněný přenos primární zvukové cesty má charakter filtru typu dolní propusti s rozhodným kmitočtem cca 470 Hz, který v nepropustném pásmu klesá se strmostí přibližně 12 dB na oktávu. V přenosovém pásmu takového filtru nehraje zpoždění o pouhých 18 vzorků nijak zásadní roli - výrazněji se projevuje až na vyšších kmitočtech (viz. 11.1.2).

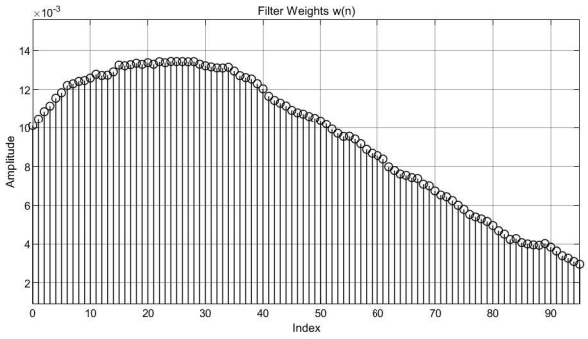
Na následujícím měření je možné vidět reálný a aproximovaný přenos primární zvukové cesty.



Obr. 11.36: SIM 10 - druhá fáze - aproximace přenosu primární zvukové cesty - amplituda



Obr. 11.37: SIM 10 - druhá fáze - aproximace přenosu primární zvukové cesty - fáze



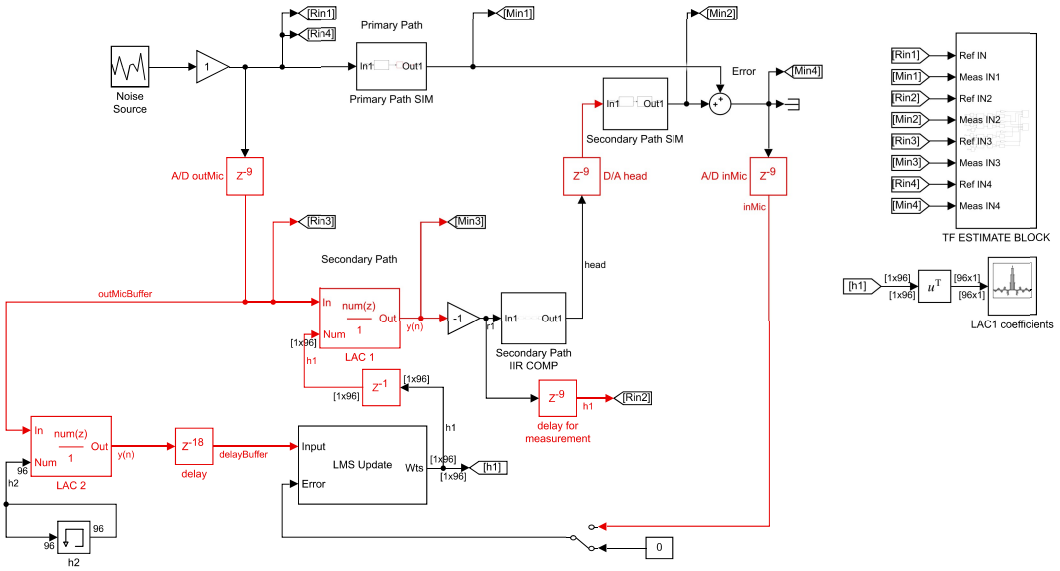
Obr. 11.38: SIM 10 - druhá fáze - aproximace přenosu primární zvukové cesty - koeficienty LAC1

11.2.3 SIM 11 - třetí fáze - adaptivní potlačení šumu

Ve chvíli, kdy známe přenos obou signálových cest - primární i sekundární, můžeme přistoupit k samotnému potlačení šumu. Pro názornost modelu jej realizujeme ve dvou částech, nejprve bez adaptace koeficientů a následní s adaptací. Zastavení adaptace koeficientů lze docílit pomocí toho, že manuálně vynulujeme chybový signál.

Zde je třeba zmínit, že i když je chybový signál roven 0, adaptační algoritmus stále počítá nové koeficienty, ale tyto se nijak neliší od těch původních. Takové řešení má smysl hlavně z toho důvodu, aby byla zachována stálá výpočetní zátěž DSP čipu.

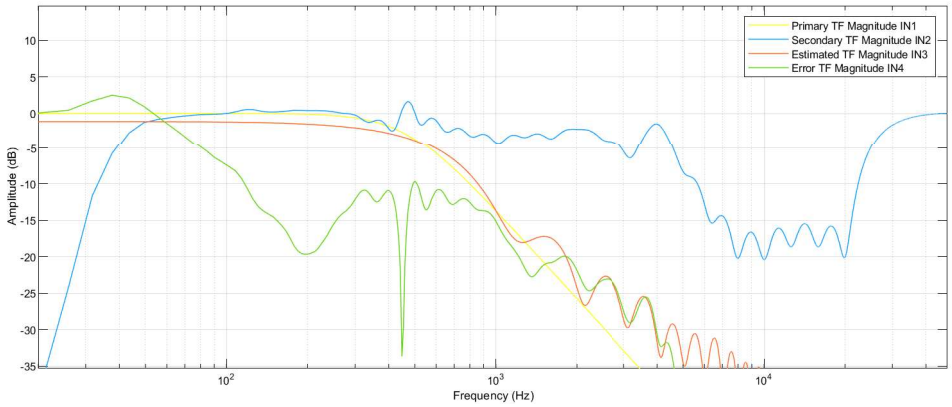
Takový model vypadá následovně:



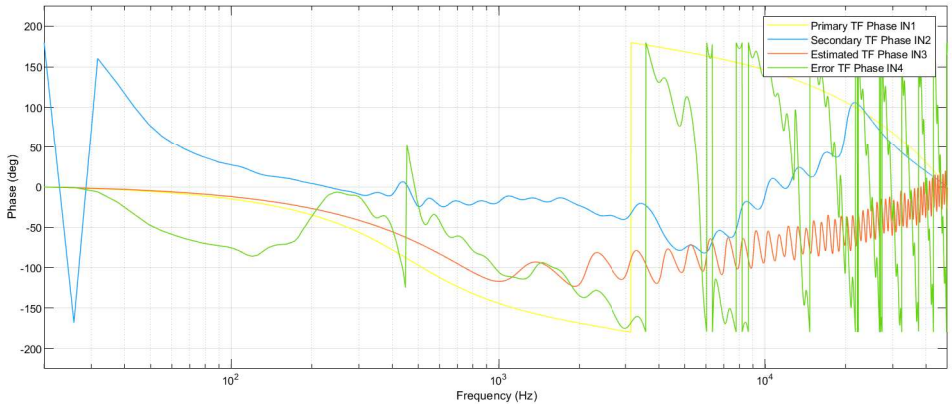
Obr. 11.39: SIM 11 - třetí fáze - potlačení šumu bez adaptace - model

Bylo by vhodné připomenout, že v této chvíli již známe koeficienty FIR filtru LAC2, které jsou uloženy v paměti (v modelu označeny jako „h2“) - tyto koeficienty jsme získali v průběhu první fáze a jak lze vyčíst z 11.1.7, jedná se o aproximaci přenosu sekundární zvukové cesty. Rovněž již známe výchozí koeficienty pro LAC1 (v modelu jsou označeny jako „h1“), které jsme získali ve fázi druhé a reprezentují aproximaci přenosu cesty primární.

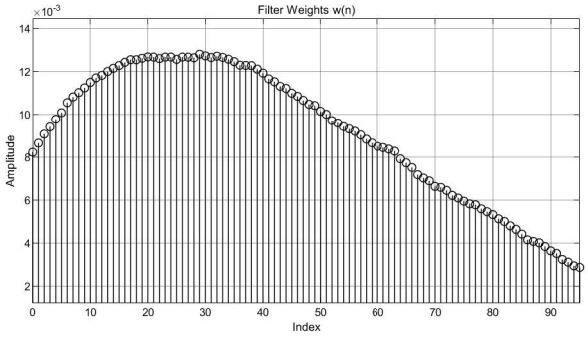
Přenos šumu z okolí do prostoru sluchátka bez finální adaptace, tedy ve chvíli, kdy LAC1 pouze aproximuje přenos primární cesty, je v následujícím měření zobrazen zelenou křivkou (označen „Error TF“).



Obr. 11.40: SIM 11 - třetí fáze - potlačení šumu bez adaptace - amplituda



Obr. 11.41: SIM 11 - třetí fáze - potlačení šumu bez adaptace - fáze

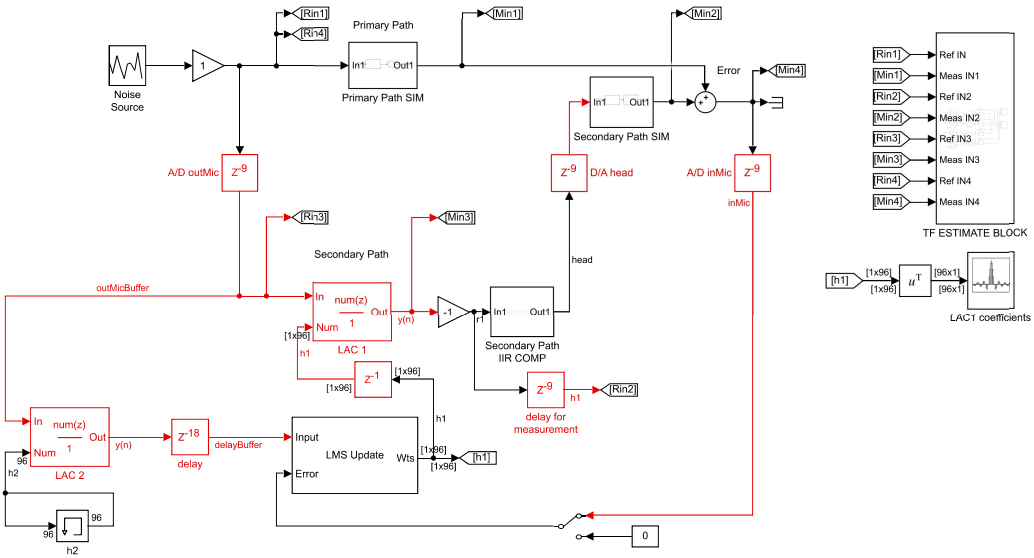


Obr. 11.42: SIM 11 - třetí fáze - potlačení šumu bez adaptace - koeficienty LAC1

Pozorný čtenář si jistě pamatuje, že adaptivní filtr LAC1 by měl po výsledné adaptaci vykonávat dvě úlohy zároveň - aproximovat přenos primární zvukové cesty a zároveň pomocí inverzní filtrace kompenzovat kmitočtové zkreslení způsobené přenosem sekundární zvukové cesty.

V tuto chvíli plní pouze svou první funkci a to aproximaci přenosu primární zvukové cesty. Tato první funkce je bezesporu tou důležitější, ale výsledek je ještě možné náležitě vylepšit, pokud bude LAC1 plnit i svou druhou úlohu.

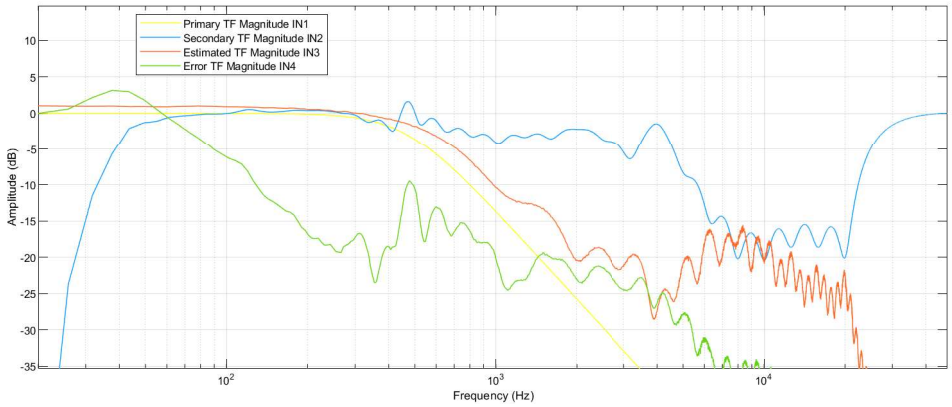
Model s povolenou adaptací bude vypadat následovně:



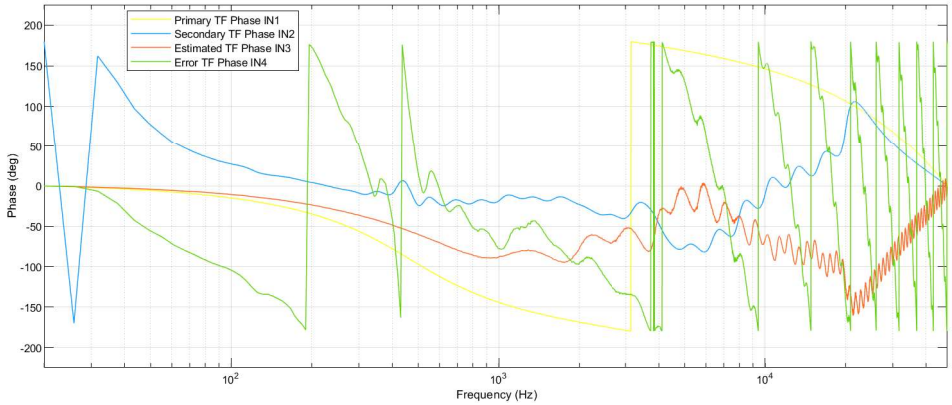
Obr. 11.43: SIM 11 - třetí fáze - potlačení šumu po adaptaci - model

Po dosažení optimálního stavu koeficientů LAC1 může být adaptace zastavena. Výsledný přenos okolního šumu do vnitřního prostoru sluchátka je možné vidět na následujícím měření (označen jako „Error TF“).

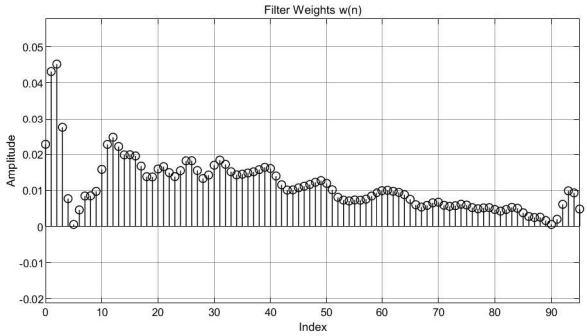
Oproti poslednímu měření je zde možné pozorovat, že se odečet hlavně v oblasti mezi kmitočty 300 a 1000 Hz zlepšil o cca 6 dB, čímž bylo dosaženo širšího pásma odečtu.



Obr. 11.44: SIM 11 - třetí fáze - potlačení šumu po adaptaci - amplituda



Obr. 11.45: SIM 11 - třetí fáze - potlačení šumu po adaptaci - fáze



Obr. 11.46: SIM 11 - třetí fáze - potlačení šumu po adaptaci - koeficienty LAC1

11.2.4 Zhodnocení simulovaného potlačení šumu

Po prostudování posledních simulací lze prohlásit, že se nám pomocí adaptivní filtrace podařilo navrhnout dopředný (Feed-Forward) ANC systém, jehož výsledné potlačení je relativně širokopásmové a vhodné pro praktickou realizaci.

Pro zhodnocení výsledného odečtu je nutné si výsledné měření rozdělit na několik kmitočtových pásem.

V pásmu pod 60 Hz dochází dle modelu k částečné sumaci, nejde však o ideální součet dvou korelovaných signálů (+6dB). Tato částečná sumace je způsobena převážně nevhodně vysoko umístěnými ořezovými filtry na vstupech a výstupech zvukového kodeku na vývojové desce DSP čipu, jejichž přenos je do modelu zahrnut v rámci modelovaného přenosu sekundární zvukové cesty. Jejich optimalizací bychom mohli v této kmitočtové oblasti dosáhnout lepších výsledků, ale toto již není obsahem této práce.

V pásmu od 200 do 1500 Hz dochází k širokopásmovému potlačení okolního hluku s úzkopásmovými odchylkami, způsobenými mírným zvlněním amplitudového přenosu sekundární cesty. Tato lokální zvlnění (např. na kmitočtu 480 Hz) nedokáže adaptivní filtr kompenzovat, jelikož se stále nacházíme v kmitočtové oblasti, kde FIR filtr délky 96 vzorků není zcela efektivní. Lokální zvlnění neumožní na daném kmitočtu přesně nastavit vhodnou amplitudu odečtového signálu a proto není odečet tak efektivní jako na okolních kmitočtech.

V pásmu od 1500 Hz do 8 kHz je pasivní útlum sluchátka efektivnější v potlačování okolního šumu než námi implementovaný algoritmus, a to převážně z důvodu dodatečné latence přítomné v obvodu. Proto je cílem adaptivního filtru přenos sekundární zvukové cesty v tomto kmitočtovém pásmu co nejvíce potlačit a tím se pokusit přiblížit přenosu cesty primární. Celkem však stále dosahujeme v této kmitočtové oblasti silného útlumu přes 15 dB oproti úrovni okolnímu šumu, pouze ne tak silného, jako kdybychom systém potlačení šumu úplně odpojili.

Tato odchylka je jednou z mála nevýhod daného řešení a pokud je pro vaše řešení tato kmitočtová oblast obvzláště důležitá, je třeba vzít tento fakt do úvahy a hledat alternativní řešení. Pokud se však na celou věc podíváme s odstupem, lze říci, že pomocí tohoto systému potlačení šumu dosáhneme rovnoměrnějšího potlačení okolního šumu na širší kmitočtové oblasti než bez jeho použití a v této kritické oblasti dosahujeme pozvolnější změny útlumu, než za použití čistě pasivního útlumu sluchátka. Výsledný dojem z útlumu okolního šumu je tedy vyrovnanější.

12 Realizace a ověření

V předcházející části práce byly popsány jednotlivé fáze, kterými musí navržený program projít a na modelových příkladech byly tyto fáze postupně rozebrány a otestovány. Dalším krokem je tyto modely převést na výsledný program.

Než však bude možné začít programovat samotný algoritmus zpracovávající vstupní a výstupní vzorky, je nutné vytvořit nebo získat zbytek pogramu - jakousi páteřní část, která zajistí obsluhu vývojové desky DSP čipu a na ní přítomných obvodů a zajistí časování, vzájemnou komunikaci a synchronizaci těchto obvodů.

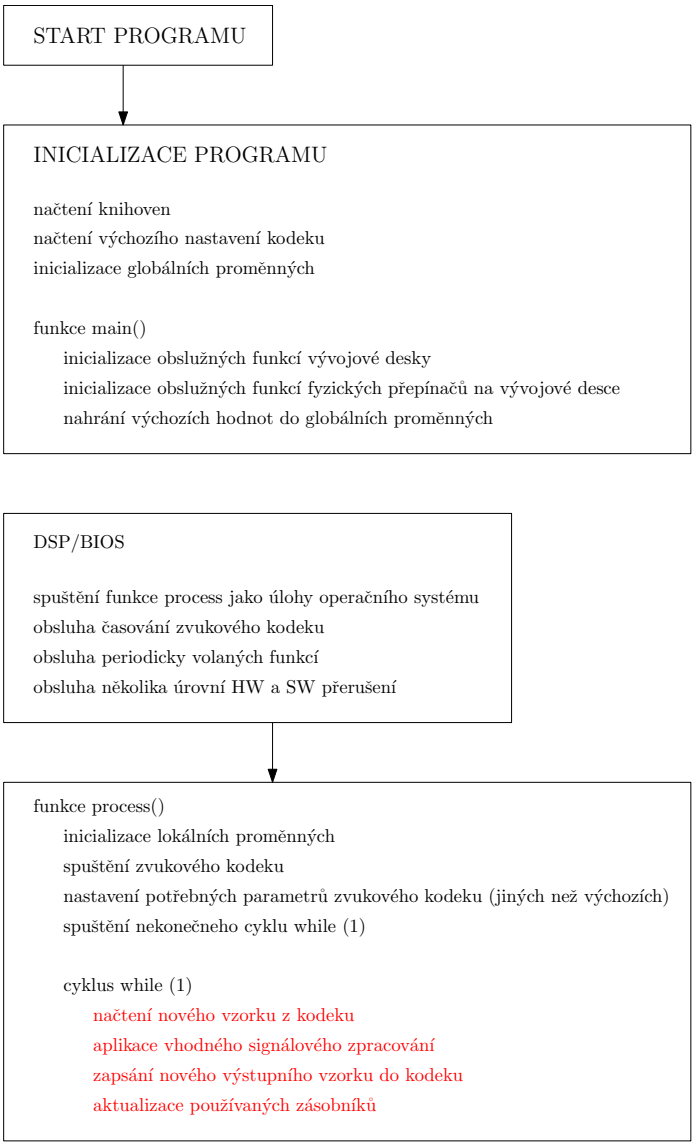
V případě této realizace jde o výchozí projekt napsaný v jazyce C pomocí programu Code Composer Studio 6.2.0.00050, což je vývojové prostředí přímo od výrobce DSP čipu - Texas Instruments. Páteřní část programu se skládá z vhodného výchozího nastavení operačního systému DSP/BIOS, alokace a namapování vhodných částí interní i externí paměti, implementuje funkce pro nastavení a obsluhu integrovaného zvukového kodeku AIC23 a dalších nezbytných součástí, bez kterých by navržený algoritmus nebylo možné implementovat v reálném čase na DSP čipu TMS320C6416T.

Tvorba této páteřní části programu není přímo obsahem této práce, proto byla tato část programu částečně převzata z dostupných materiálů výrobce, viz. [14], [15], [23] a částečně i z podpůrných materiálů určených pro výuku předmětu Signálové procesory na VUT FEKT, viz. [7], [8]. Výchozí páteřní část programu bylo pro potřeby realizace tohoto programu nutné modifikovat a přizpůsobit, ale její hlavní část pochází z výše citovaných zdrojů.

Hlavní částí programu je však samotný algoritmus signálového zpracování, jehož návrh a realizace obsahem této práce samozřejmě je a proto je zde realizace tohoto algoritmu, navrženého v předchozí části práce, podrobně popsána a náležitě otestována.

12.1 Struktura programu

Program je navržen tak, aby fungoval dle následujícího diagramu. Všechny části tohoto programu jsou podrobně popsány a implementovány dále.



Obr. 12.1: Zjednodušená struktura programu

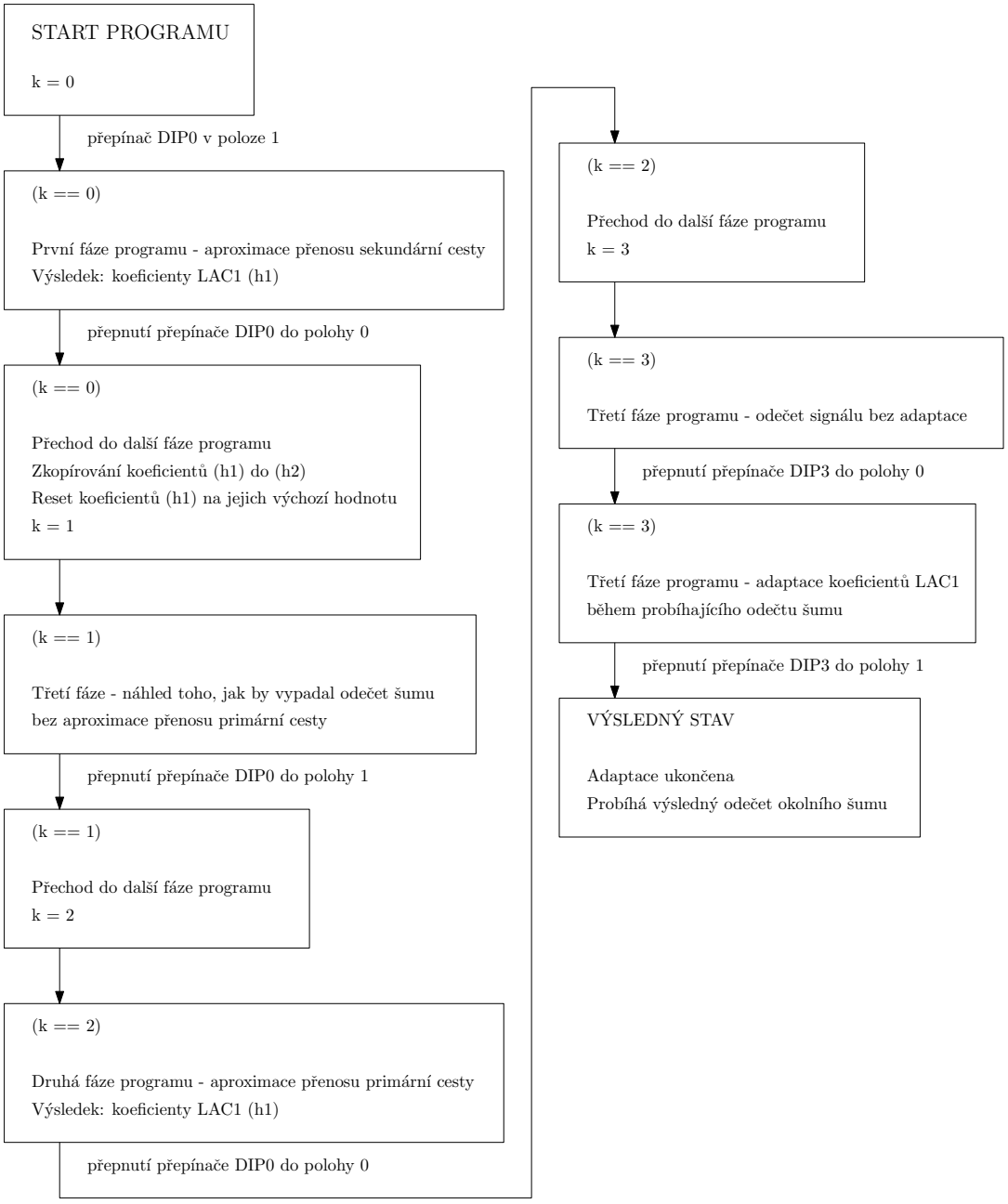
Samotné signálové zpracování probíhá v rámci nekonečného cyklu poté, co jsou všechny potřebné obvody spuštěny a všechny potřebné proměnné jsou iniciovány a připraveny.

Jak již bylo v předchozím textu práce řečeno, program zajišťující zpracování zvukového signálu musí v předem daném pořadí úspěšně provést několik na sebe

navazujících úkonů. Tyto úkony jsou rozděleny do jednotlivých fází programu a každá fáze programu je implementována a popsána zvlášť.

Navigace mezi jednotlivými fázemi programu je řešena pomocí pomocné stavové proměnné "k", přechod mezi jednotlivými fázemi a ovládání programu je vyřešeno ručně pomocí fyzických přepínaču nacházejících se na vývojové desce DSP čipu.

Strukturu jednotlivých fází a stavů programu lze vyčíst z následujícího diagramu:



Obr. 12.2: Jednotlivé stavy a fáze programu

Přílohou této práce jsou dva projekty obsahující zdrojové kódy spustitelné v programu Code Composer Studio 6.2.0.00050. O dva projekty jde proto, že program lze realizovat ve dvou variantách - s IIR filtrací a bez ní. Více o výhodách a nevýhodách obou variant je napsáno v závěrečném zhodnocení (12.4).

Nyní si projdeme jednotlivé bloky programu tak, jak jsou chronologicky realizovány.

12.2 Inicializační část programu

12.2.1 Načtení potřebných knihoven

Nejprve je nutné načíst potřebné zdrojové knihovny pro práci s programem a obsluhu DSP čipu a jeho vývojové desky. načtení těchto knihoven obstarává následující kód.

```
#include <std.h>
#include <log.h>
#include <sts.h>
#include <clk.h>
#include <stdlib.h>
#include <math.h>
#include <stdio.h>
#include "hellocfg.h"

// Obsluha vývojové desky TMS320C6146T DSK
#include "dsk6416.h"
// Obsluha fyzických přepínačů na vývojové desce
#include "dsk6416_dip.h"
// Obsluha fyzických LED diod na vývojové desce
#include "dsk6416_led.h"
// Nastavení a obsluha zvukového kodeku
#include "aic23.h"
// Optimalizovaná interní funkce pro realizaci
// diskrétního FIR filtru
#include "dsp_fir_r4.h"
// Optimalizovaná interní funkce pro realizaci LMS bloku pro
  // výpočet nových koeficientů
#include "dsp_firlms2.h"

#include "noise.h"
```

Výpis 12.1: hello.c

12.2.2 Uložení růžového šumu do externí paměti

Pro optimální výsledek adaptace v první fázi programu je nutné přehrávat do reproduktoru sluchátka širokopásmový šum, který však musí být pro optimální funkci algoritmu nekorelovaný s okolním šumem. I když je možné takový náhodný signál pomocí různých algoritmů generovat, výhodnějším řešením je si takový signál vygenerovat dopředu a uložit si jej do externí paměti SDRAM na vývojové desce (viz [1]), odkud pak stačí pouze číst jednotlivé vzorky a přehrávat je. Toto řeší hlavičkový soubor „noise.h“. Tento šum byl zaznamenán, nakvantován a převeden do číselného rozsahu $< -32768; 32767 >$ pomocí pomocného skriptu v prostředí MATLAB.

Jelikož se jedná o velké množství vzorků, proměnná „noise“ je přesunuta do externí paměti SDRAM na vývojové desce.

```
short noise[132072] = {-2457, -3887, -3975, -2837 ...}
// Přesun proměnné noise do předem definované externí paměti
#pragma DATA_SECTION( noise,".EXTRAM")
```

Výpis 12.2: noise.h

12.2.3 Globální proměnné a výchozí nastavení zvukového kodeku

Po načtení potřebných knihoven je třeba nahrát do zvukového kodeku jeho výchozí konfiguraci (později je možné ji dle potřeby měnit) a provést inicializaci globálních proměnných. Výchozí konfigurace zvukového kodeku je předem definovaná obslužnou knihovnou tohoto kodeku.

```
// Výchozí konfigurace kodeku
static AIC23_ConfigTab CodecCfg = AIC23_DEFAULTCONFIG;

// Předpis pro převod hodnot z rozsahu <-1;1) do rozsahu datového
   typu short <-32768;32767>
//#define FLOAT2FIXED(x)  ((x) >= 0 ? ((Int16)((x)*32768+0.5 )) :
   ((Int16)((x)*32768-0.5)))

// Počet koeficientů adaptivního filtru (řád filtru + 1)
# define N (96)
// Zásobník pro ukládání vstupních vzorků z proměnné "outMic"
short outMicBuffer[N+3];
// Zásobník pro realizaci zpoždění až o 31 vzorků
short delay[32];
// Zásobník pro ukládání vzorků z výstupu zpožďovacího zásobníku
  //"delay"
short delayBuffer[N+3];
// Pole koeficientů určených pro LAC1
short h1[N];
```

```
// Pole koeficientů určených pro LAC2
short h2[N];
// Krátkodobá paměť pro uložení potřebných vzorků
short mem[N];

// Počet koeficientů v "h1" a "h2"
int nh = N;
// Výstupní vzorky LAC1
short r1[4];
// Výstupní vzorky LAC2
short r2[4];
// Počet vzorků v "r1" a "r2"
int nr = 4;
// Aktuální hodnota chyby odečtu/chyby adaptace (chybový signál)
short b = 0;
// Délka zásobníku "delay"
const short nd = 32;
// Index pro čtení ze zásobníku "delay"
short dlyRead = 0;
// Index pro zápis do zásobníku "delay"
short dlyWrite = 18;

// Konstanta pro postupnou navigaci programem
short k = 0;
```

Výpis 12.3: hello.c

Pokud realizujeme program včetně kompenzační IIR sekce, je třeba ještě doplnit proměnné k tomu potřebné.

```
// Paměť pro vstupní a výstupní vzorky IIR filtru
short x[3], y[3];
// Koeficienty pro realizaci IIR filtru ve tvaru
  // {b0, b1, b2, a0, a1, a2}
// Tyto koeficienty jsou sníženy na polovinu, aby splňovaly
  // podmínku, že všechna čísla se musí nacházet v rozsahu
  // <-32768;32767>
// Snížení koeficientů je během výpočtu průběžně kompenzováno
short coef [6] = {16558, -32617, 16075, 16384, -32617, 16248};
// Pole sloužící pro uložení mezivýpočtů při realizaci IIR filtru
short state[5] = {0};
```

Výpis 12.4: hello.c

12.2.4 Knihovna DSPlib

V rámci programu jsou použity dvě optimalizované funkce z knihovny přímo od výrobce daného čipu - DSPlib (viz [16]). Tato knihovna obsahuje funkce, které implementují běžně používané algoritmy, které jsou však ručně optimalizované přímo v assembleru pro daný DSP čip a nepřekládají se tedy z jazyka C.

Z této knihovny jsou v kódu použité dvě funkce, konkrétně funkce „dsp_fir_r4()“ pro výpočet FIR filtru a funkce „dsp_firlms2()“ pro výpočet nových koeficientů pomocí LMS algoritmu.

Tyto konkrétní funkce je sice možné přímo implementovat jako v následujícím kódu a takto implementované funkce v testovacím prostředí fungují, ale ve chvíli, kdy je po nich vyžadováno, aby fungovaly v rámci komplexního programového řešení neposkytují tak přesný a rychlý výsledek jako obě výše zmíněné optimalizované funkce z knihovny DSPlib.

Pro seznámení s danými algoritmy jsou zde uvedeny implementace funkcí vykonávajících výpočet FIR filtrace a aktualizaci koeficientů adaptivního filtru na základě LMS algoritmu, ve výsledném programu jsou však použity jejich optimalizované alternativy.

```
short fxfir( const short x[], short h[], int nh){
  /* Funkce realizující výpočet diskrétního FIR filtru
   *  - x[] = pole vstupních vzorků o délce "nh", na první pozici
      je vždy nejnovější vzorek
   *  - h[] = pole koeficientů pro výpočet FIR filtru
   *  - nh = počet prvků v poli "h" a "x"
   */
  int i;
  int accu = 0;
  short yn = 0;

  for (i = 0; i < nh; i++){
    // accu += (h[i] * x[i]);
    accu = _sadd( accu, _smpy( h[i], x[i] ) );
  }

  yn = _sadd( 0x00008000, accu) >> 16;
  return yn;
}
void fxlms(short h, const short x,  short  b, int nh ){
  /* Funkce realizující výpočet a adaptaci nových koeficientů FIR
     filtru pro následující cyklus
   *  - h[] = pole koeficientů pro výpočet FIR filtru
   *  - x[] = pole vstupních vzorků o délce "nh", na první pozici
      je vždy nejnovější vzorek
```

```
     *  - b = chybový signál, na jehož základě probíhá adaptace
        koeficientů
     *  - nh = počet prvků v poli "h" a "x"
     */
    int i;
    int accu;

    for (i = 0; i < nh; i++){
      //w[i] += b*x[i];
      accu = _sshl( h[i], 16);
      accu = _sadd( accu, _smpy(b, x[i]) );
      h[i] = _sadd( 0x00008000, accu) >> 16;
    }
}
```

Výpis 12.5: hello.c

12.2.5 Inicializace vývojové desky, výchozí hodnoty proměnných

Na začátku programu je vždy spuštěna funkce „main()“. Tato funkce proběhne pouze jednou a v tomto programu je využita pro přepsání všech minulých nebo předem neurčených hodnot globálních proměnných na předem určené výchozí hodnoty.

```
// Hlavní funkce "main()" je spuštěna pouze jednou a to ihned po
   inicializaci globálních proměnných
Void main()
{
  CSL_init();
  // Inicializace samotné vývojové desky a jejích obslužných
     programů
  DSK6416_init();
  // Inicializace obsluhy HW přepínačů na vývojové desce
  DSK6416_DIP_init();
  // Inicializace obsluhy LED diod na vývojové desce
  DSK6416_LED_init();

  // Pomocná proměnná
  int i, accu;
  // Nastavení všech počátečních hodnot v zásobnících
    // "outMicBuffer" a "delayBuffer" na nulu
  for (i = 0; i < nh+3; i++){
    *(short *)&outMicBuffer[i] = 0;
    *(short *)&delayBuffer[i] = 0;
  }
  // Nastavení všech koeficientů v poli "h1" a "h2" na nulu
  for (i = 0; i < nh; i++){
```

```
    *(short *)&h1[i] = 0
    *(short *)&h2[i] = 0;
  }
  // Nastavení výchozích hodnot koeficientů v poli "h1" a "h2"
    // na [1,0,0, ... 0]
  *(short *)&h1[0] = 32767;
  *(short *)&h2[0] = 32767;
  // Vymazání všech počátečních hodnot v zásobníku "delay"
  for (i = 0; i < nd; i++){
    *(short *)&delay[i] = 0;
  }
  // Nastavení všech výchozích hodnot v poli "r1" a "r2" na nulu
  for (i = 0; i < 4; i++){
    *(short *)&r1[i] = 0;
    *(short *)&r2[i] = 0;
  }
  // Nastavení všech výchozích hodnot v poli "x" a "y" na nulu
  for (i=0; i<3; i++){
    *(short *)&x[i] = 0;
    *(short *)&y[i] = 0;
  }
  // Inverze koeficientů "a1" a "a2" pro zjednodušení IIR filtrace
  accu = _smpy(coef[4], -32768);
  coef[4] = _sadd(accu, 0x00008000) >> 16;
  accu = _smpy(coef[5], -32768);
  coef[5] = _sadd(accu, 0x00008000) >> 16;

    return;
}
```

Výpis 12.6: hello.c

12.3 Spuštění funkce process - hlavní část programu

Po vykonání obsahu funkce „main()“ dochází k tomu, že operační systém DSP/BIOS spouští jako úlohu (nebo také vlákno) funkci „process()“, jež skrývá hlavní jádro programu - přijetí nového vstupního vzorku z kodeku, jeho následné zpracování a odeslání nového výstupního vzorku zpět do kodeku.

12.3.1 Lokální proměnné, nastavení kodeku

```
  // Funkce process je spuštěna jako "task" (= úloha, vlákno)
    // operačním systémem DSP/BIOS
void process( void)
{
  // Celý vstupní vzorek (32 bitů pro dva 16 bitové kanály)
  Int32 sample;
  // Vstupní vzorky jednotlivých kanálů
  Int16 left, right, inMic, outMic;
  // Výstupní hodnoty jednotlivých kanálů
    // Označení odvozeno od jejich reálného využití pro lepší
       orientaci napříč programem
  Int16 meas = {0}, head = {0};
  // Dočasná pomocná proměnná pro výpočty s rozlišením 32 bitů
  Int32 accu;
  // Index pro čtení z pole "noise", použitý při cyklickém
    // přehrávání růžového šumu z externí paměti
  Int32 t = 0;
  // Čekání na spuštění zvukového kodeku
  if( AIC23_OpenCodec( &CodecCfg) < 0)
    return;
  // Nastavení vzorkovacího kmitočtu na 96 kHz
  AIC23_SetFreq( AIC23_FREQ_96KHZ);
  // Nastavení vstupního zesílení kodeku na hodnotu 23
    // (z rozsahu 0-33)
  AIC23_InGain(23);
  // Nastavení výstupního zesílení kodeku na hodnotu 127
    // (z rozsahu 0-127)
  AIC23_OutGain(127);
  // Možnost propojit přímo v kodeku signál ze vstupu na výstup,
    // použití pro diagnostiku a vývoj
  AIC23_Loopback(0);
  // Možnost vypnout zvukový výstup přímo v kodeku, použití
    // pro diagnostiku a vývoj
  AIC23_Mute(0);
```

Výpis 12.7: hello.c

12.3.2 Načtení vstupního vzorku

Po inicializaci lokálních proměnných a finálním nastavení a spuštění kodeku následuje spuštění nekonečného cyklu, v rámci něhož probíhá samotné zpracování jednotlivých vzorků.

```
// Hlavní cyklus, který realizuje čtení vstupního, výpočet a ná
   sledný zápis výstupního vzorku
while( 1)
  {
    // Pomocná proměnná
    int i;

// READ INPUT SAMPLE
    // Čtení celého aktuálního vzorku z kodeku AIC23 a jeho uložení
        do proměnné "sample"
    // (32 bitů pro dva kanály po 16 bitech)
    AIC23_Read( &sample);

    // Izolace 16 bitů levého kanálu
    left = _ext( sample, 0, 16);
    // Izolace 16 bitů pravého kanálu
    right = _ext( sample, 16, 16);

// INPUT ROUTING MATRIX
    // Sekce pro uložení správných vstupních vzorků do správných
       proměnných

    // Signál z vnitřního mikrofonu je uložen do proměnné "inMic"
    inMic = left;
    // Signál z vnějšího mikrofonu je uložen do proměnné "outMic"
    outMic = right;
```

Výpis 12.8: hello.c

12.3.3 První fáze - aproximace přenosu sekundární cesty

Cílem první fáze je identifikovat a aproximovat přenos sekundární zvukové cesty. Model této fáze je zpracován v této práci v sekci 11.2.1.

Do reproduktoru sluchátka je přehráván růžový šum z externí paměti na vývojové desce, zvuk uvnitř sluchátka je následně zaznamenán vnitřním mikrofonem a po A/D převodu uložen do vstupní proměnné „inMic“.

Přehrání růžového šumu z paměti

```
// PROCESSING
    // Sekce pro realizaci samotného výpočtu
    // Navigace v této sekci je realizována pomocí proměnné "k"
    // Jednotlivé fáze se přepínají pomocí prvního HW přepínače na
       vývojové desce

    // Čtení hodnoty prvního HW přepínače na vývojové desce
    if( DSK6416_DIP_get(0)){

      if(k == 1)
        // Přechod předcházející aproximaci přenosu PRIMÁRNÍ zvukov
           é cesty
        k = 2;
      if(k == 3)
        // Přechod předcházející opakované aproximaci přenosu PRIMÁ
           RNÍ zvukové cesty
        k = 2;

      switch (k){
        case 0:
          // První fáze - aproximace přenosu SEKUNDÁRNÍ zvukové
             cesty

          // Do výstupu "head" je posílán růžový šum čtený z
            // externí paměti na vývojové desce
          // Amplituda signálu je následně snížena na polovinu
            // pro dosažení optimálního chování při LMS adaptaci
            // (úroveň určena experimentálně)
          head = noise[t+100] >> 1;
```

Výpis 12.9: hello.c

IIR filtrace

Jestliže realizujeme program bez IIR kompenzace, tento signál „head“ je odeslán přímo na výstup. Pokud ale realizujeme program včetně IIR kompenzačního filtru, stane se signál „head“ zdrojem pro tento filtr.

IIR filtr druhého řádu je zde poněkud netradičně realizován pomocí první přímé formy, a to z důvodu aritmetiky s pevnou řádovou čárkou. První přímá forma totiž obsahuje pouze jeden jediný sumační bod a díky tomu je možné využít té vlastnosti dvojkového doplňku, že může při sčítání několika hodnot jeho hodnota krátkodobě přetéct, pokud bude výsledný součet v původním rozsahu. Více o této problematice je možné si přečíst v literatuře [12] a [10].

Pro výpočet jsou zde použité Intrinsic funkce _sshl() a _sadd(). Funkce _sshl() realizuje bitový posun doleva se saturací na 32. bitu, proto je třeba vstupní hodnotu posunout o šestnáct bitů vlevo do horní poloviny 32 bitového slova. Pokud bychom zde saturaci neprovedli, systém by poté v případě přetečení rozsahu 16 bitového čísla byl zatížen velkou chybou. Funkce _sadd() funguje podobně, jen provádí součet namísto bitového posunu. I _sadd() provádí saturaci na 32. bitu a je proto vhodné, když je aplikována na dvě 32 bitové proměnné.

Více o dostupných Intrinsic funkcích je možné si přečíst v [15].

Výsledek obou funkcí je pak nutné převést zpět do 16 bitové reprezentace, což se provádí pomocí zaokrouhlení a následného bitového posunu.

```
// IIR
            // Vstupní vzorek je pro navýšení numerického dynamického
                rozsahu násoben dvěma
            // x = 2*head;
            accu = _sshl(head<<16,1);
            x[0] = _sadd ( accu , 0x00008000 ) >> 16;

            //state[0] = x[0]*b0
            accu = x[0] * coef[0];
            accu = _sshl(accu,1);
            state[0] = _sadd(accu,0x00008000) >> 16;

            //state[1] = x[1]*b1
            accu = x[1] * coef[1];
            accu = _sshl(accu,1);
            state[1] = _sadd(accu,0x00008000) >> 16;

            //state[2] = x[2]*b2
            accu = x[2] * coef[2];
            accu = _sshl(accu,1);
            state[2] = _sadd(accu,0x00008000) >> 16;
```

```
            //state[3] = y[1]*a1
            accu = y[1] * coef[4];
            accu = _sshl(accu,1);
            state[3] = _sadd(accu,0x00008000) >> 16;

            //state[4] = y[2]*a2
            accu = y[2] * coef[5];
            accu = _sshl(accu,1);
            state[4] = _sadd(accu,0x00008000) >> 16;

            // accu = SUM(state0:4);
            accu = state[0] + state[1];
            accu = accu + state[2];
            accu = accu + state[3];
            accu = accu + state[4];

            // Pro kompenzaci snížení všech koeficientů na polovinu
               je výsledný vzorek násoben dvěma
            // y = 2*accu;
            accu = accu + accu;
            accu = _sshl(accu, 16);
            y[0] =  _sadd(accu,0x00008000) >> 16;

            // Pokud byl vstupní vzorek násoben dvěma, výstupní
               vzorek je nutné vydělit dvěma
            //y = accu/2;
            head = _sadd(y[0], 0x00000001) >> 1;

// konec IIR
```

Výpis 12.10: hello.c

Výstup IIR filtru je opět uložen do proměnně „head“, která je odeslána na výstup do reproduktoru sluchátka.

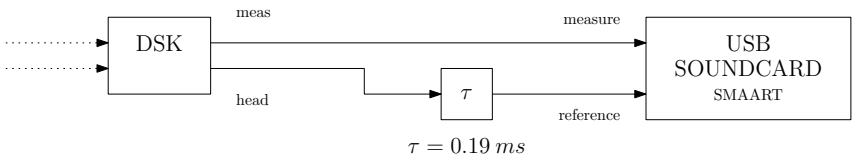
FIR filtrace - LAC1

Dalším krokem je implementace diskrétního FIR filtru. Jak již bylo zmíněno, pro tento účel je zde použita optimalizovaná funkce z knihovny DSPlib ([16]) DSP_fir_r4. Vstupním zásobníkem pro tento FIR filtr je zásobník „delayBuffer“, na jehož první pozici se nachází nejnovější vzorek zpožděný o 18 vzorků, viz model 11.2.1.

```
            // Ten stejný šumový signál je uložen na vstupní pozici
               do zásobníku "delay"
            delay[dlyWrite] = noise[t+100] >> 1;
            accu = delay[dlyRead];
            // Hodnota šumového signálu zpožděná o 18 vzorků je
              // následně uložena na první pozici v zásobníku
              // "delayBuffer"
            *(short *)&delayBuffer[0] = accu;

            // Realizace LAC1
              // LAC = linear adaptive combiner, jde o odborné
              // označení diskrétního FIR filtru s proměnnými
                 koeficienty
            DSP_fir_r4( delayBuffer, h1, r1, nh, nr);
            //r1[0] = fxfir(delayBuffer, h1, nh);

            // Do výstupu "meas" je posílán výstup LAC1 pro možnost
              // měření jeho přenosu
            meas = r1[0];

            // Výstup LAC1 je následně invertován
            accu = _smpy(r1[0], -32768);
            r1[0] = _sadd(accu, 0x00008000) >> 16;
```

Výpis 12.11: hello.c

V předchozím kódu je možné si povšimnout přiřazení výstupu LAC1 do proměnné „meas“, která reprezentuje měřící výstup z vývojové desky DSP čipu. V této fázi programu je možné sledovat chování LAC1 a měřit jeho přenos pomoc následujícího zapojení. Časová konstanta kompenzuje latenci, která vzniká při průchodu signálu D/A a A/D převodníky. Signál „r1[0]“ je vhodné získat ještě před jeho inverzí, protože tímto způsobem odpovídá změřená fázová charakteristika LAC1 předpokládané hodnotě.



Obr. 12.3: Měření přenosu LAC1 v průběhu první fáze programu

Adaptace koeficientů LAC1

Po FIR filtraci je na řadě adaptace jeho koeficientů pro následující cyklus. Tato adaptace probíhá na základě vstupního signálu ze zásobníku „delayBuffer“ a chybového signálu „b“. Opět je zde pro výpočet použita výše zmíněná optimalizovaná funkce DSP_firlms2. Pro zpomalení adaptace a zvýšení přesnosti je chybový signál násoben konstantou 0,25 - literatura tuto konstantu označuje jako μ [2].

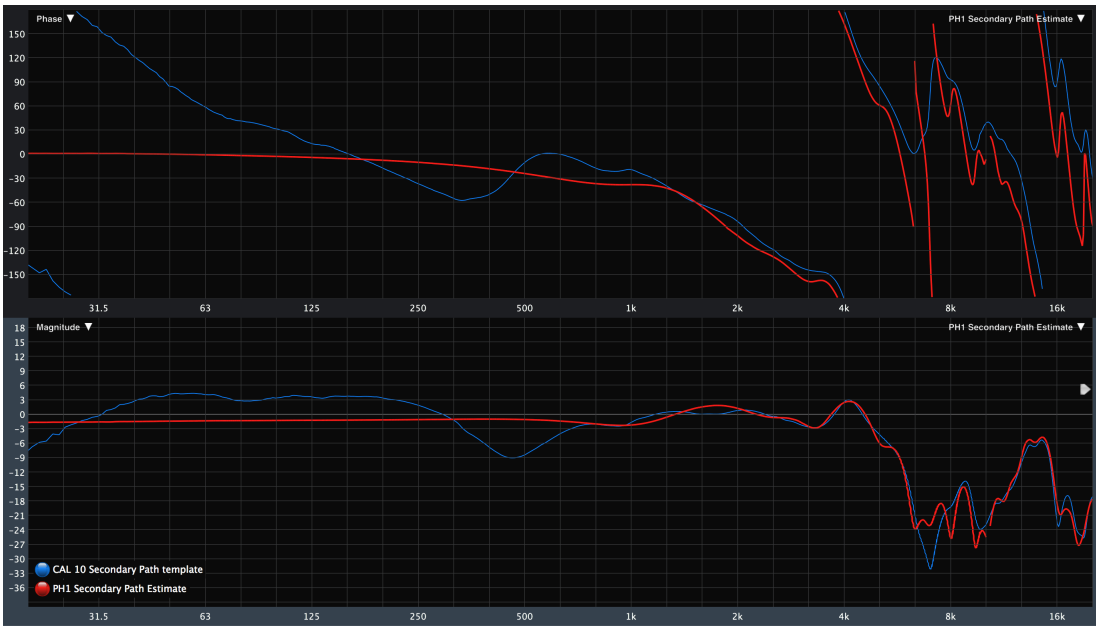
Jelikož v první fázi realizujeme zapojení adaptivního filtru pro identifikaci neznámého systému (viz 10.1.2), probíhá výpočet chybového signálu stále v rámci DSP, proto je po provedení adaptace koeficientů vypočítán chybový signál, který se použije pro adaptaci koeficientů v příštím cyklu.

```
            // Chybový signál je před výpočtem nových koeficientů ná
               soben konstantou 0,25
            b = b >> 2;

            // Funkce aktualizuje koeficienty v poli "h1" na základě
               algoritmu LMS
              // její návratová hodnota není v tomto kódu využita
            accu = DSP_firlms2( h1, delayBuffer, b, nh);

            // Výpočet chybového signálu pro příští výpočet nových
               koeficientů LAC1
            accu = _sadd( inMic << 16, r1[0] << 16);
            b = _sadd(accu, 0x00008000) >> 16;

         break;
```

Výpis 12.12: hello.c

Jak jsem již zmínil, správnou funkci adaptivního algoritmu lze kontrolovat pomocí měření přenosu LAC1. Výsledek tohoto měření je možné vidět na následujícím obrázku.

Na měření je zřetelně vidět vliv délky FIR filtru na výslednou přesnost aproximace, kdy na vyšších kmitočtech (nad 1 kHz) aproximuje filtr velice přesně přenos sekundární cesty, avšak v oblasti nižších kmitočtů již tato aproximace není tak přesná. Omezení délky FIR filtru je možné efektivně kompenzovat právě implementací kompenzačního IIR filtru, viz 11.1.8.



Obr. 12.4: Aproximace přenosu sekundární zvukové cesty

Ukončení první fáze

Ve chvíli, kdy dosáhne adaptivní filtrace svého optimálního stavu - viz předcházející měření, je možné ukončit první fázi programu a připravit se na fázi následující.

Po ukončení první fáze je nutné zkopírovat koeficienty LAC1 do koeficientů LAC2 a obnovit výchozí hodnoty koeficientů LAC1.

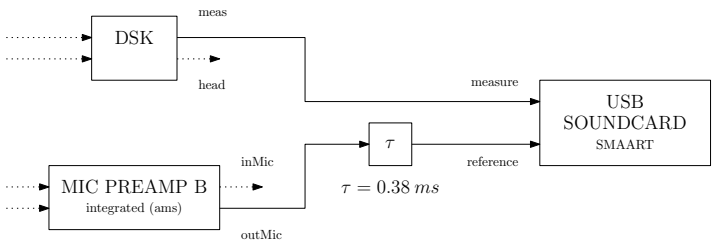
```
    }else{
      if(k == 0){
        // Přechodová akce po aproximaci přenosu SEKUNDÁRNÍ zvukové
            cesty

        // Při aproximaci SEKUNDÁRNÍ zvukové cesty jsou použity
          // koeficienty v poli "h1", proto je nutné je následně
          // překopírovat do pole "h2", kde jsou uloženy a později
          // použity k dalšímu výpočtu
        for (i = N-1; i >= 0; i--){
          h2[i] = h1[i];
          // Zároveň je provedeno nulování původních koeficientů v
              poli "h1"
          h1[i] = 0;
        }
        // První koeficient v poli "h1" je nastaven na výchozí
            hodnotu přibližně +1
        // Z důvodu limitace aritmetiky s pevnou řádovou čárkou
            neexistuje přímo hodnota +1 (32768), nahrazena je tedy
```

```
            nejbližším existujícím číslem
        *(short *)&h1[0] = 32767;
        k = 1;
    }
```

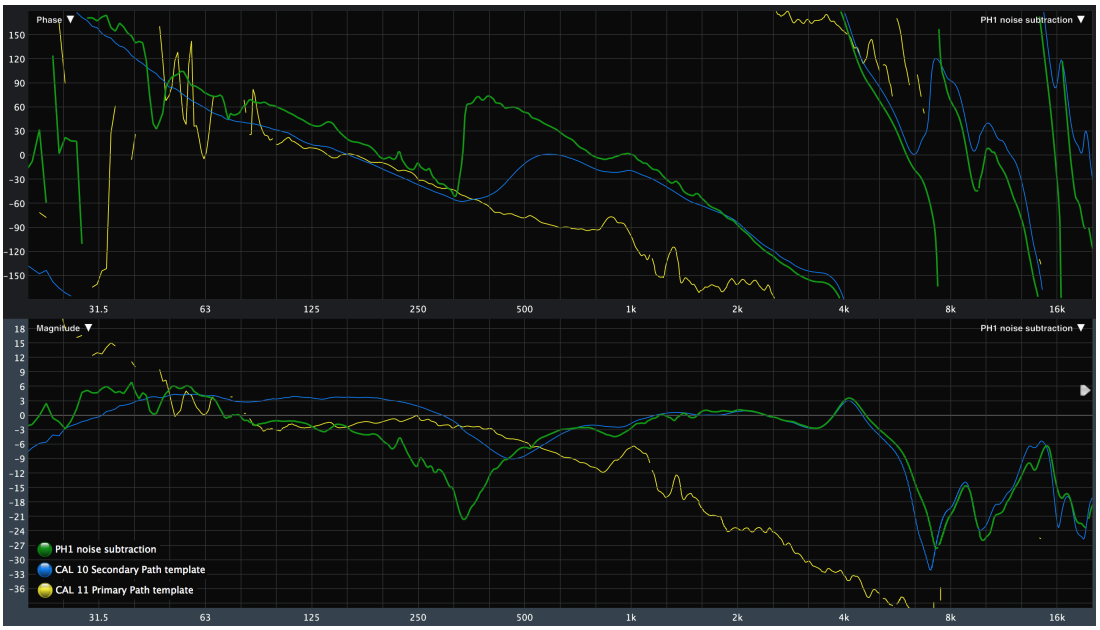
Výpis 12.13: hello.c

Náhled výsledného odečtu bez aproximace primární zvukové cesty

V tuto chvíli je možné změřit, jak by vypadal pokus o odečet okolního šumu ve chvíli, kdybychom vynechali druhou fázi programu. Měření probíhá dle následujícího diagramu:



Obr. 12.5: Měření útlumu sluchátka



Obr. 12.6: Náhled odečtu bez aproximace primární zvukové cesty

Toto měření ukazuje, že odečet signálu bez aproximace primární zvukové cesty není efektivní ani vhodný. Takto by vypadal přenos vnějšího šumu do vnitřního

prostoru sluchátka v takzvaném monitor módu (viz 2.4), jehož cílem je naopak okolní šum téměř bez útlumu přehrávat do sluchátka. Takový mód je možné využít například pro krátkou konverzaci s nasazenými sluchátky.

V této části programu není doporučeno realizovat adaptaci koeficientů LAC1. Při realizaci systému pomocí aritmetiky s pohyblivou čárkou by tato sekce měla s použitím adaptace bez poblému konvergovat k optimálnímu výsledku, při realizaci systému pomocí aritmetiky s pevnou řádovou čárkou tomu tak však není z důvodu omezené dynamiky signálu a bez jakéhokoli výchozího bodu by tato adaptace byla značně nestabilní, způsobovala by nepříjemné zvukové artefakty a nacházela by se na hraně stability.

```
        switch (k){
          case 1:
            // Odečet okolního hluku před aproximací přenosu PRIMÁRNÍ
              // zvukové cesty - jde o náhled toho, jak by vypadal
              // odečet bez realizace druhé fáze programu
            // Vstupní signál z vnějšího mikrofonu je uložen do
              // zásobníku "outMicBuffer"
            *(short *)&outMicBuffer[0] = outMic;
            // Realizace LAC1
            DSP_fir_r4( outMicBuffer, h1, r1, nh, nr);
            //r1[0] = fxfir(outMicBuffer, h1, nh);

            // Výstup LAC1 je následně invertován
            accu = _smpy(r1[0], -32768);
            r1[0] = _sadd(accu, 0x00008000) >> 16;
            // Do výstupu "head" je posílán invertovaný signál
              // z výstupu LAC1
            head = r1[0];
            // V tuto chvíli není doporučeno umožnit adaptaci
               koeficientů LAC1

            // Chybový signál je reprezentován akustickou sumací
               okolního hluku a zvuku přehrávaného z reproduktoru
               sluchátka. Tento signál je zaznamenán vnitřním
               mikrofonem a jeho úroveň je uložena do proměnné "b"
            b = inMic;
            // Do výstupu "meas" je posílán přímo chybový signál pro
               možnost měření přenosu okolního hluku do sluchátka
            meas = b;
          break;
```

Výpis 12.14: hello.c

12.3.4 Druhá fáze - aproximace přenosu primární cesty

Pro plynulejší průběh výsledné adaptace a přesnější výsledek je vhodné realizovat před výslednou adaptací ve třetí fázi programu ještě aproximaci primární zvukové cesty. Ta je realizována podobným způsobem jako aproximace cesty sekundární, ale s tím rozdílem, že měřící širokopásmový šum je generován externě a je přehráván do bezprostředího okolní sluchátka (stejným způsobem, jako tomu bylo při měření pasivního útlumu sluchátka - viz 9.4).

Tento šum je pak sejmut vnějším i vnitřním mikrofonem a jejich vzájemný rozdíl je definován právě přenosem primární zvukové cesty (pasivním útlumem sluchátka).

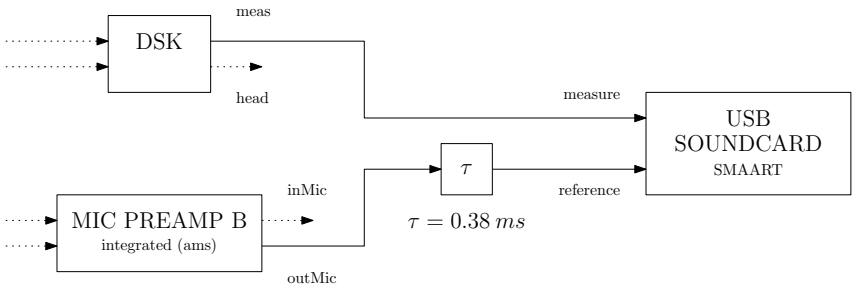
```
        switch (k){
          case 2:
            // Druhá fáze - aproximace přenosu PRIMÁRNÍ zvukové cesty

            // Do výstupu "head" není posílán žádný zvukový signál
            head = 0;

            // Do okolního prostředí je přehráván růžový šum
              // z externího zdroje
            // Vstupní signál z vnějšího mikrofonu je uložen do
              // zásobníku "delay"
            delay[dlyWrite] = outMic;
            accu = delay[dlyRead];

            // Hodnota signálu z vnějšího mikrofonu zpožděná
              // o 18 vzorků je následně uložena na první pozici
              // v zásobníku "delayBuffer"
            *(short *)&delayBuffer[0] = accu;

            // Realizace LAC1
            DSP_fir_r4( delayBuffer, h1, r1, nh, nr);
            //r1[0] = fxfir(delayBuffer, h1, nh);

            // Do výstupu "meas" je posílán výstup LAC1 pro možnost
              // měření jeho přenosu
            meas = r1[0];

            // Výstup LAC1 je následně invertován
            accu = _smpy(r1[0], -32768);
            r1[0] = _sadd(accu, 0x00008000) >> 16;
```

Výpis 12.15: hello.c

```
            // Chybový signál tentokrát není násoben žádnou
               konstantou (nebo konstantou 1)

            // Funkce aktualizuje koeficienty v poli "h1" na základě
               algoritmu LMS, její návratová hodnota není v tomto kó
               du využita
            accu = DSP_firlms2( h1, delayBuffer, b, nh);

            // Výpočet chybového signálu pro budoucí výpočet nových
               koeficientů LAC1
            accu = _sadd( inMic << 16, r1[0] << 16);
            b = _sadd(accu, 0x00008000) >> 16;
          break;
```

Výpis 12.16: hello.c

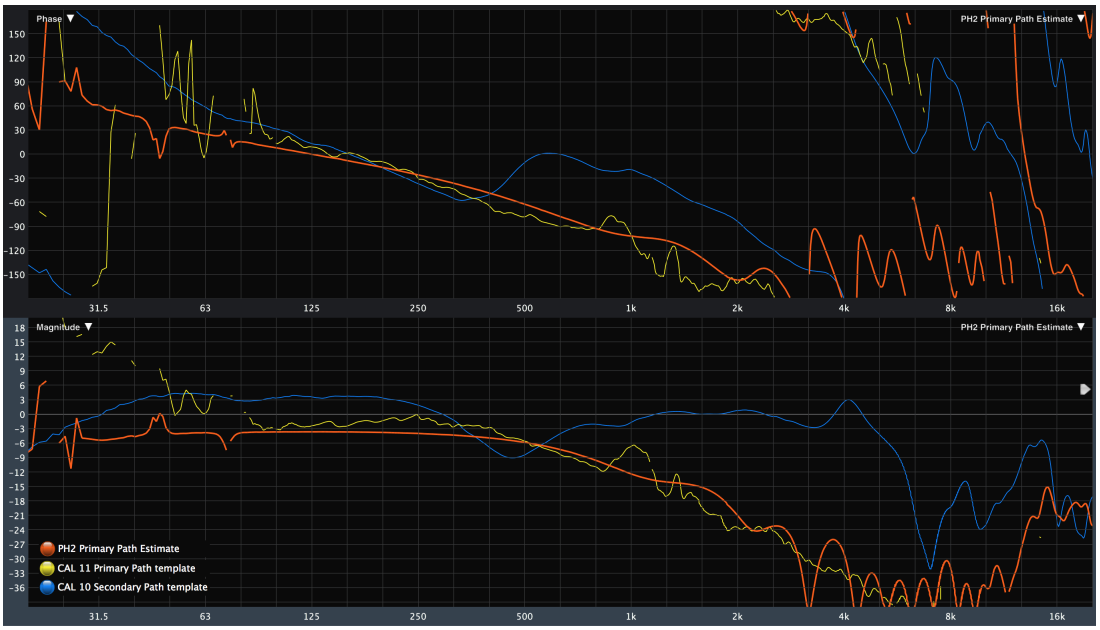
Výsledkem druhé fáze programu jsou koeficienty LAC1, které aproximují přenos primární zvukové cesty. Tento přenos je možné opět měřit s použitím externího měřené přenosu pomocí následujícího zapojení.



Obr. 12.7: Měření přenosu LAC1 v průběhu druhé fáze programu

Výsledné měření vypadá následovně, přenos LAC1 je zde vynesen oranžovou křivkou a označen „PH2 Primary Path Estimate“, pro porovnání jsou zde zobrazeny i dříve změřené přenosy primární i sekundární zvukové cesty.

V tomto měření je možné pozorovat jeden z důsledků nedokonalých vlastností použitých mikrofonů, a to pokles jejich citlivosti nad 8 kHz. Tento pokles totiž ovlivní chování adaptivního algoritmu, který realizuje danou aproximaci. Tento pokles se v našem zapojení nachází dvakrát (používáme dva mikrofony), ale po podrobném pozorování tohoto jevu na modelu lze konstatovat, že vliv na adaptaci má převážně mikrofon vnější, který poskytuje zdrojový signál adaptivnímu filtru LAC1. A v případě, že vstupní signál nemá v dané oblasti dostatečnou úroveň, bude přenos adaptivního filtru v této oblasti stále narůstat.



Obr. 12.8: Aproximace přenosu primární cesty

Tento jev je částečně kompenzován přítomností vlastního šumu mikrofonního předzesilovače, který nedostatečnou úroveň v oblasti vyšších kmitočtů částečně dorovná, jedná se však o nekorelovaný zdroj šumu vůči tomu okolnímu a proto jím není možné zmíněný nedostatek kompenzovat zcela.

Pokud však pomineme tuto oblast nejvyšších kmitočtů a zaměříme se na zbytek kmitočtového spektra, je možné prohlásit, že primární zvuková cesta je tímto způsobem dostatečně přesně aproximována.

Ukončení druhé fáze

Ve chvíli, kdy dosáhne adaptivní filtrace svého optimálního stavu - viz předcházející měření, je možné ukončit druhou fázi programu a připravit se na fázi následující. Na závěr druhé fáze je nutné zkopírovat koeficienty LAC1 do paměti „mem“.

```
        if(k == 2){
          // Přechod po aproximaci přenosu PRIMÁRNÍ zvukové cesty
          for (i = N-1; i >= 0; i--){
            mem[i] = h1[i];
          }

          k = 3;
        }
```

Výpis 12.17: hello.c

12.3.5 Třetí fáze - výsledný odečet okolního šumu

Uživatelské ovládání programu

Pro zhodnocení finální adaptace je nutné znát stav odečtu předtím, než se tato finální adpatace projeví. V programu je tato možnost realizována pomocí fyzického přepínače na vývojové desce DSK, který umožňuje pozastavit adaptaci koeficientů.

Nyní je vhodná chvíle ukázat si, jak je řešeno ovládání programu pomocí těchto přepínačů, jedná se o čtyři fyzické přepínače, z nichž každý může nabývat pouze hodnot 0 a 1.

```
// USER INTERFACE
    // Uživatelské rozhraní je v tuto chvíli reprezentováno čtyřmi
       HW přepínači, které jsou integrovány přímo na vývojové desce

    // Funkce prvního HW přepínače je popsána i implementována
      // v kódu výše, zde jej uvádím pouze jako formalitu pro
      // zlepšení přehlednosti programu. Tento přepínač je používan
      // pro přechod mezi jednotlivými fázemi programu.
/*
if( DSK6416_DIP_get(0)){
    }else{
    }
*/
    // Pomocí druhého HW přepínače je možné krátkodobě ztlumit
      // výstup "head" do reproduktoru sluchátka. Tato funkce
      // umožňuje měření pasivního útlumu sluchátka bez odečtového
         signálu.
    if( !DSK6416_DIP_get(1)){
      head = 0;
    }

    // Třetí HW přepínač umožňuje smazat koeficienty uložené v poli
        "h1" a nahradit je výchozími
    if( !DSK6416_DIP_get(2)){
      if(k != 3){
        for (i = N-1; i >= 0; i--){
          // Pokud nejsme ve třetí fázi programu, výchozí "h1"
            // je [1,0,0,0...0]
          h1[i] = 0;
        }
        h1[0] = 32767;
```

Výpis 12.18: hello.c

```
        }else{
          for (i = N-1; i >= 0; i--){
            // Pokud jsme ve třetí fázi programu, výchozí "h1"
              // odpovídá hodnotám získaným ve fázi druhé a uložené
              // v "mem"
            h1[i] = mem[i];
          }
        }
      }

      // Čtvrtý HW přepínač umožňuje zapínat a vypínat adaptaci
        // koeficientů v poli "h1" pomocí manipulace s chybovým
        // signálem.
      if( !DSK6416_DIP_get(3))
        b = _smpy( b, 32767) >> 16;
      else{
        b = _smpy( b, 0) >> 16;
      }
      // Pro upřesnění je třeba dodat, že adaptace probíhá, i pokud
        // je chybový signál nulový, ale hodnota koeficientů v poli
        // "h1" se nemění. Toto chování je žádané, cílem je, aby
        // program pracoval stále stejně a rovnoměrně, ale bylo možné
        // kontrolovat, kdy jsou a kdy nejsou koeficienty
        // aktualizovány a měněny.
```

Výpis 12.19: hello.c

FIR filtrace - LAC1

V průběhu třetí fáze je program postaven podobně jako ve fázi druhé, ale do výstupu „head“ je přehráván odečtový signál s možností použít kompenzační IIR filtr. Adaptaci je možné pro porovnání výsledného odečtu s odečtem bez adaptace pozastavit pomocí přepínače na vývojové desce DSK, viz předchozí sekce.

```
        switch (k){
          case 3:
            // Odečet okolního hluku po aproximaci přenosu PRIMÁRNÍ
              // i SEKUNDÁRNÍ zvukové cesty
            // Vstupní signál z vnějšího mikrofonu je uložen do
              // zásobníku "outMicBuffer"
            *(short *)&outMicBuffer[0] = outMic;

            // Realizace LAC1
            DSP_fir_r4( outMicBuffer, h1, r1, nh, nr);
            //r1[0] = fxfir(outMicBuffer, h1, nh);

            // Výstup LAC1 je následně invertován
            accu = _smpy(r1[0], -32768);
            r1[0] = _sadd(accu, 0x00008000) >> 16;

            // Do výstupu "head" je posílán invertovaný signál
              // z výstupu LAC1
            head = r1[0];
```

Výpis 12.20: hello.c

IIR filtrace

Pokud realizujeme program včetně IIR filtrace, tato filtrace je v tomto místě aplikována na signál „head“ stejným způsobem, jako tomu bylo v první fázi programu. Kód realizující IIR filtrace se nijak nemění, proto zde není nutné jej znovu uvádět. Tento kód je dostupný v sekci 12.3.3.

FIR filtrace - LAC2

V první fázi programu jsme získali koeficienty FIR filtru, jehož přenos aproximuje přenos sekundární zvukové cesty. Tyto koeficienty jsou důležité, protože pokud bychom v této třetí fázi programu provedli adaptaci koeficientů LAC1 na základě vstupního signálu bez aproximace přenosu sekundární zvukové cesty, tato adaptace by byla nestabilní a nevedla by k optimálnímu řešení. O tomto problému je možné si více přečíst v sekci 11.1.6.

Protože zde realizujeme zapojení pojmenované jako „filtered-x LMS“, viz 11.1.7, před výpočtem nových koeficientů je realizována FIR filtrace pomocí LAC2 a až výstup tohoto filtru je následně použit pro adaptaci koeficientů LAC1.

```
            // Realizace LAC2
            DSP_fir_r4( outMicBuffer, h2, r2, nh, nr);
            //r2[0] = fxfir(outMicBuffer, h2, nh);

            // Výstupní signál z LAC2 je uložen do zásobníku "delay"
            delay[dlyWrite] = r2[0];
            accu = delay[dlyRead];

            // Ten samý signál zpožděný o 18 vzorků je následně
              // zapsán na první pozici zásobníku "delayBuffer"
            *(short *)&delayBuffer[0] = accu;
```

Výpis 12.21: hello.c

Adaptace koeficientů LAC1

Ve chvíli, kdy už jsou dostupné všechny signály potřebné k výpočtu nových koeficientů stačí jen vynásobit chybový signál vynásobit konstantou 0,5 pro zpomalení adaptace a zvýšení přesnosti výsledku.

V následujícím kódu je implementována ještě jedna optimalizace, kterou je omezena adaptace prvních tří koeficientů LAC1. Toto omezení je nutné z důvodu, který je popsán v sekci 12.3.4, jedním z projevů nedokonalých vlastností vnějšího mikrofonu, konkrétně jeho poklesu citlivosti nad kmitočtem 8 kHz, je neustálý nárůst hodnot několika prvních koeficientů adaptivního filtru při implementaci adaptivního odečtu. Hodnota těchto koeficientů stoupá velmi pomalu, ale pokud překročí jistou úroveň, může způsobit nestabilitu této adaptace.

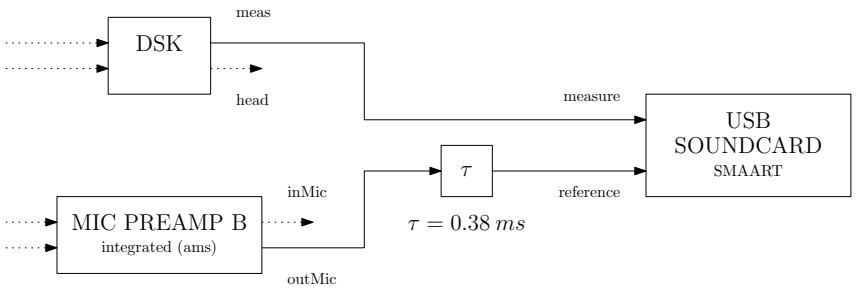
Řešení tohoto problému není systémové a vychází z analýzy a pozorování chování tohoto fenoménu. Počet omezených koeficientů je určen experimentálně. Tato optimalizace může fungovat hlavně z důvodu, že tyto první koeficenty nemají nijak zásadní vliv na přenos nižších kmitočtů a díky implementaci druhé fáze - aproximace přenosu primární zvukové cesty - jsou již nastaveny na hodnoty, které jsou pro výsledný odečet použitelné.

```
            // Chybový signál je před výpočtem nových koeficientů ná
               soben konstantou 0,5
            b = b >> 1;

            // První tři koeficienty z pole "h1" jsou uloženy v
              // krátkodobé paměti "mem"
            // Ty jsou následně použity pro optimalizaci adaptace
            mem[0] = h1[0];
            mem[1] = h1[1];
            mem[2] = h1[2];

            // Funkce aktualizuje koeficienty v poli "h1" na základě
              // algoritmu LMS, její návratová hodnota není v tomto
              // kódu využita
            accu = DSP_firlms2( h1, delayBuffer, b, nh);

            // Pro zajištění větší stability adaptivního filtru není
               několik prvních koeficientů aktualizováno, protože
               tyto jsou nejdůležitější pro nejvyšší kmitočty, na
               kterých nemá funkce DSP_firlms2 dostatečné hodnoty na
               vstupu - kvůli omezenému propustnému pásmu snímacích
               mikrofonů.
            // Jedná se o nestandardní optimalizaci, která vznikla na
                základě analýzy modelu a pozorování adaptivního
               algoritmu
            h1[0] = mem[0];
            h1[1] = mem[1];
            h1[2] = mem[2];

            // Chybový signál je tentokrát reprezentován akustickou
               sumací okolního hluku a zvuku přehrávaného z
               reproduktoru sluchátka. Výsledný chybový signál je
               zaznamenán mikrofonem uvnitř sluchátka a jeho aktuální
                hodnota je uložena do proměnné "b"
            b = inMic;
            // Do výstupu "meas" je posílán přímo chybový signál pro
               možnost měření přenosu okolního hluku do sluchátka
            meas = b;
          break;
        }
      }
```

Výpis 12.22: hello.c

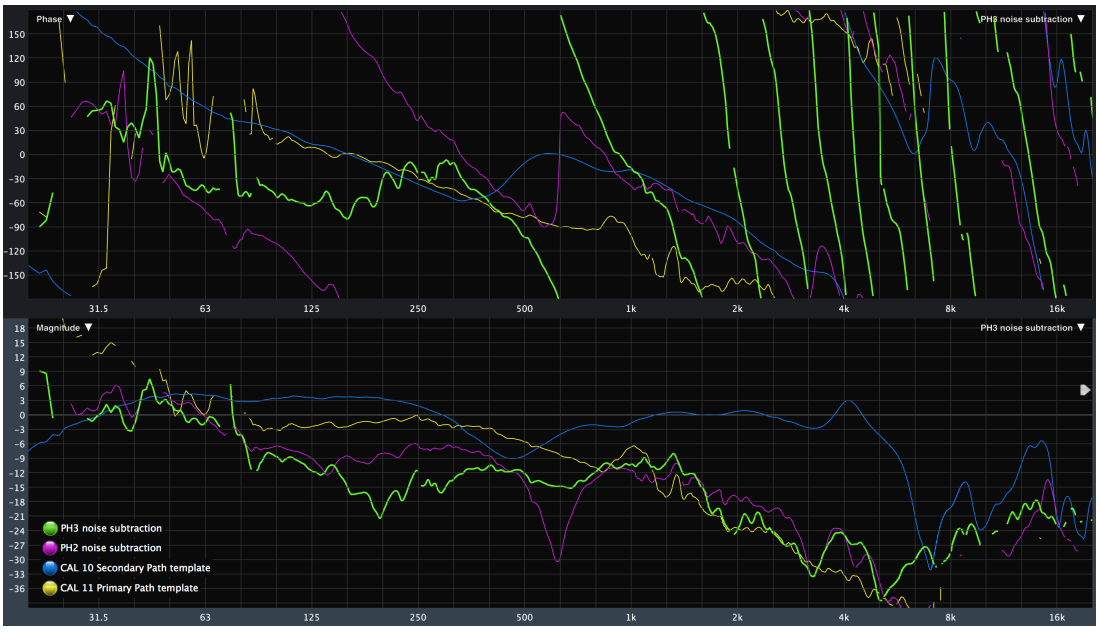
Měření útlumu okolního šumu před a po adaptaci

Měření probíhá pomocí stejného zapojení jako při měření druhé fáze:



Obr. 12.9: Měření výsledného útlumu v průběhu třetí fáze programu

V následujícím obrázku je zobrazena kmitočtová závislost útlumu okolního šumu před adaptací (označen „PH2 noise subtraction“) a po adaptaci (označen „PH3 noise subtraction“). Toto měření zobrazuje výsledek při realizaci programu bez IIR kompenzace.



Obr. 12.10: Měření útlumu okolního šumu před a po adaptaci bez IIR kompenzace

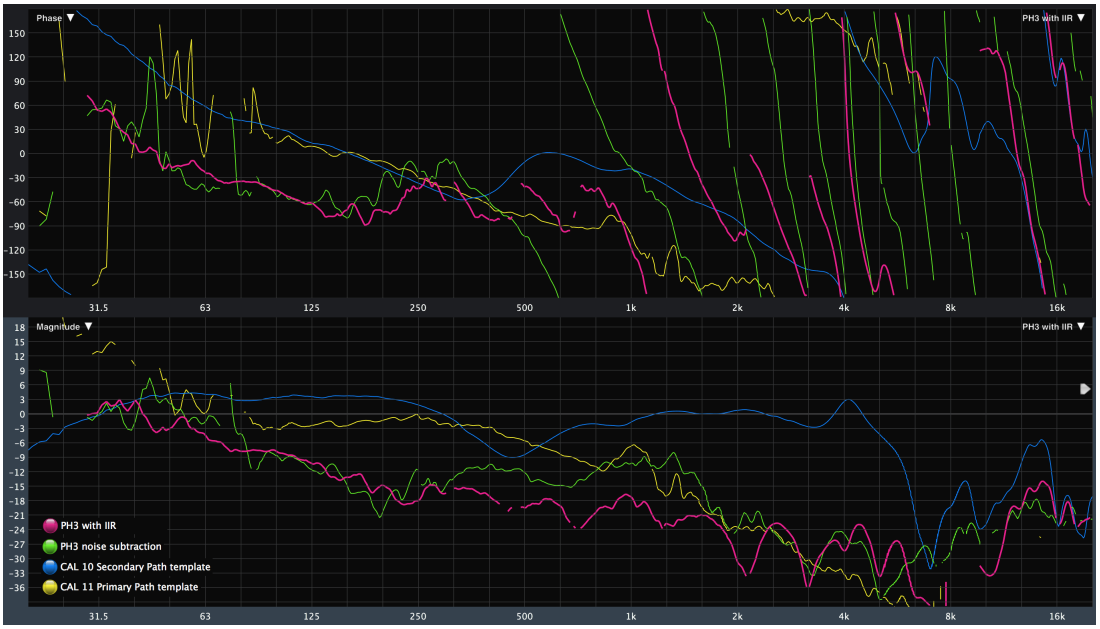
Z tohoto měření vyplývá, že i útlum okolního šumu bez finální adaptace je při mírném snížení nároků zcela použitelný (zobrazen pomocí fialové křivky, označen „PH2 noise subtraction“). Pokud však požadujeme, aby byl systém účinný v širším

kmitočtovém pásmu, je ideální použít k optimalizaci koeicientů adaptivní filtr (výsledný útlum po adaptaci je zobrazen pomocí zelené křivky, označen „PH3 noise subtraction“).

Útlum okolního šumu po adaptaci včetně IIR kompenzační filtrace

V případě, že realizujeme program včetně IIR kompenzační filtrace, získáme následující výsledek. Výsledný útlum bez IIR kompenzace je popsán jako „PH3 noise subtraction“ a je zobrazen pomocí zelené křivky, výsledný útlum včetně IIR kompenzace je popsán jako „PH3 with IIR“ a je zobrazen pomocí růžové křivky.

Z tohoto měření vyplývá, že použít IIR filtr pro kompenzaci nedostatečné délky FIR filtru je vhodné a funkční řešení a díky této kompenzaci se útlum okolního šumu rozšíří na většinu kmitočtového spektra. I když je útlum stále kmitočtově závislý, jedná se o širokopásmový útlum s hodnotami cca -6 *až* -9 dB v kmitočtovém pasmu od 100 *do* 1000 Hz, což lze považovat za výborný výsledek. Proč není možné provést širokopásmový odečet šumu i na vyšších kmitočtech je popsáno v sekci 11.1.2.



Obr. 12.11: Měření útlumu okolního šumu po adaptaci s IIR kompenzací

12.3.6 Odeslání nového výstupního vzorku

Poté, co je získán, je nový výstupní vzorek uložen do proměnné „head“, která je spolu s měřící proměnnou „meas“ odeslána na výstup zvukového kodeku. Před odesláním do výstupu kodeku jsou však ještě jednou invertovány a to z toho důvodu, že sluchátkový výstup vývojové desky s DSP čipem má invertovaný výstup oproti výstupu linkovému. Použití linkového výstupu je zde nevhodné hned ze dvou různých důvodů - jednak proto, že by bylo nutné do obvodu přidávat ještě další sluchátkový zesilovač a jednak proto, že linkový výstup této vývojové desky obsahuje z neznámého důvodu integrovaný filtr typu horní propusti, který je pro naši aplikaci zcela nevhodný.

```
// OUTPUT ROUTING MATRIX
      // Sekce pro odeslání správných výstupných hodnot na správné
         výstupy
      left = meas;
      right = head;

// WRITE OUTPUT SAMPLE
      // "line output" obsahuje integrovaný filtr typu horní
         propusti, výstup má správnou polaritu
      // "headphone output" je v rámci možností lineární, výstup má
          však otočenou polaritu

      // Jelikož volím pro výstup sluchátkový výstup, je třeba
         kompenzovat otočenou polaritu
      left = _smpy(-32768, left) >> 16;
      right = _smpy(-32768, right) >> 16;

      // Složení celého výstupního vzorku o délce 32 bitů ze dvou
         kanálů po 16 bitech
      sample = _extu( right, 16, 16);
      sample = sample | _extu( left, 16, 0);

      // Zapsání celého výstupního vzorku na výstup do zvukového
         kodeku AIC23
      AIC23_Write( sample);
```

Výpis 12.23: hello.c

12.3.7 Aktualizace zásobníků na konci každého cyklu

Na závěr každého cyklu je nutné aktualizovat všechny zásobníky a některé další pomocné proměnné.

```
// UPDATE BUFFERS
      // Zásobníky "outMicBuffer" a "delayBuffer" jsou kvůli
        // vstupním podmínkám používaných optimalizovaných funkcí
        // realizovány jako lineární posuvné zásobníky, proto je
        // třeba každý vzorek posunout o jednu pozici dále.
      // Následující aktuální vzorek bude vložen na první pozici
        // těchto zásobníků na začátku následujícího cyklu.
      for (i = N+3-1; i >= 0; i--){
        *(short *)&outMicBuffer[i] = *(short *)&outMicBuffer[i-1];
        *(short *)&delayBuffer[i] = *(short *)&delayBuffer[i-1]
      }

      // Zásobník "delay" je realizován jako kruhový zásobník,
        // na konci každého cyklu je tedy nutné  pouze zvětšit
        // indexy pro čtení a zápis o 1. Pro zajištění cyklického
        // opakování těchto indexů je využito faktu, že velikost
        // zásobníku "delay" odpovídá mocnině 2 a tím pádem lze
        // cyklické opakování zajistit pomocí bitového součinu
        // (operátor "&") s délkou zásobníku sníženou o 1.
      dlyRead = (dlyRead + 1) & (nd - 1);
      dlyWrite = (dlyWrite + 1) & (nd - 1);

      // Stejným způsobem je zvyšován i index "t" pro cyklické
        // čtení z pole "noise".
      t = (t + 1) & (131072 - 1);

  }
}
```

Výpis 12.24: hello.c

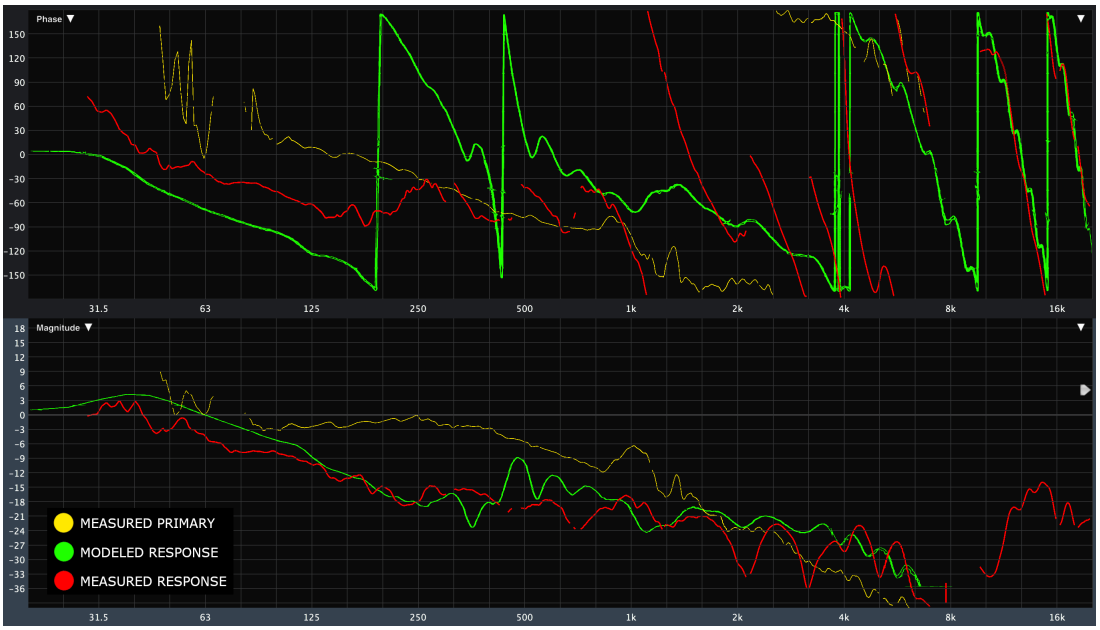
12.4 Zhodnocení výsledků realizovaného obvodu

12.4.1 Srovnání simulovaného a reálného potlačení šumu

Pro zhodnocení výsledného potlačení okolního šumu pojďme je vhodné porovnat simulovaný a reálný naměřený výsledek. Na následujícím měření je vidět:

- **pasivní útlum sluchátka**
  (žlutá křivka, označena „MEASURED PRIMARY“)
- **simulované potlačení šumu** s IIR kompenzací po výsledné adaptaci, realizované a popsané v sekci 11.2.4
  (zelená křivka, označena „MODELED RESPONSE“)
- **reálné potlačení šumu** s IIR kompenzací po výsledné adaptaci, realizované v sekci 12
  (červená křivka, označena „MEASURED RESPONSE“)

Z tohoto měření jasně vyplývá, že pro reálný výsledek platí stejné hodnocení jako pro simulovaný výsledek, který je podrobně popsán v sekci 11.2.4, výsledky obou měření jsou totiž srovnatelné.



Obr. 12.12: Srovnání simulovaného a reálného potlačení šumu

V kmitočtové oblasti pod 50 Hz je možné pozorovat mírný nárůst přenosu, a to hlavně z důvodu nevhodných ořezových filtrů použitých v analogové části obvodu (kombinace fitrů typu horní propust v mikrofonních předzesilovačích a na vývojové desce DSK - viz sekce 9). Tyto filtry se zde kumulují a jejich vliv je možné pozorovat i na přenosu sekundární cesty, naměřeném v sekci 9.4. Celkový nárůst však není tak dramatický jako v původním modelu a celkový zisk v této oblasti nepřesáhne 3 dB.

V oblasti mezi 50 a 150 Hz výsledný útum postupně narůstá od 3 do 9 dB.

V oblasti mezi 150 a 1000 Hz je útlum nejefektivnější a pohybuje se mezi 9 a 15 dB vůči pasivnímu útlumu sluchátka a celkový útlum oproti referenčnímu přenosu 1 (0 dB) se dokonce pohybuje mezi 15 až 20 dB!

V oblasti mezi 1000 a 2500 Hz ještě stále můžeme pozorovat útlum i oproti pasivnímu útlumu sluchátka, ale jeho průběh již získává charakter hřebenového filtru kvůli latenci DSP zpracování (viz 11.1.2).

V oblasti mezi 2,5 a 8 kHz je pak možné pozorovat o něco menší útlum, než jakého by bylo docíleno čistě pomocí pasivního útlumu sluchátka, ale celkový útlum zůstává vyrovnaný okolo 21 až 24 dB, což je možné hodnotit jako velmi dobrý výsledek!

V oblasti nad 8 kHz je opět možné pozorovat vliv kombinace nedostatečného přenosu mikrofonů nad 8 kHz s případnou menší netěsností náušníku, což se projevuje o něco horším útlumem, než na ostatních kmitočtech. Celkový útlum v této oblasti se však stále pohybuje v číslech nad 15 dB, takže ani v této oblasti se nedá výsledný útlum považovat za neúspěch.

12.4.2 Možnost přehrávání hudby

Pokud by to výsledný produkt vyžadoval, po ukončení třetí fáze programu již není využíván vstup pro vnitřní mikrofon, jelikož už neprobíhá žádná adaptace. Tento vstup je možné po finální adaptaci využít pro vstup hudebního signálu, který může být následně přehráván do reproduktoru sluchátka. Sluchátka je však možné používat i čistě pro potlačení okolního šumu bez přehrávání hudby.

12.4.3 Limitace implementace s pevnou řádovou čárkou

Použitý DSP čip TMS320C6416T byl pro realizaci této práce zvolen hlavně z důvodu jeho dostupnosti a výkonu, který pro realizaci výpočetních algoritmů poskytuje. I

když to tak ze začátku nevypadalo, největším limitem, který s sebou tento procesor přináší je fakt, že veškeré výpočty jsou realizovány pomocí aritmetiky s pevnou řádovou čárkou (fixed point).

Pro základní výpočty a strukturu programu tento fakt nepředstavuje žádnou překážku, avšak pro funkci některých algoritmů je velmi důležité numerické rozlišení, které bohužel není zrovna silnou stránkou systémů s pevnou řádovou čárkou.

Během realizace programu se ukázalo, že nejvíce limitované jsou tímto omezením algoritmy pro výpočet nových koeficientů a pro realizaci IIR filtrace. Bohužel jde o klíčové algoritmy pro realizaci sluchátek s adaptivním potlačením okolního šumu.

I když je možné i v těchto podmínkách dosáhnout tíženého výsledku, viz předchozí srovnání 12.4, zpětně lze prohlásit, že by pro implementaci tohoto programu bylo daleko vhodnější použít systém s větším numerickým rozsahem a ideálně systém s plovoucí řádovou čárkou, klidně za cenu snížení výpočetního výkonu.

Limitace při implementaci LMS bloku

Z teoretického úvodu do adaptivních filtrů (viz sekce 10.2.3) víme, že jedním z nejdůležitějších faktorů pro přesnou a stabilní implementaci adaptivního LMS algoritmu je vhodné zvolení konstanty μ, která určuje velikost adaptačního kroku a tím reguluje i rychlost adaptace, naprosto kritická.

Tato teorie předpokládá velké numerické rozlišení a pro LMS algoritmus není ničím nevídaným hodnota této konstanty v řádu 10^{-4} až 10^{-6}. Při práci s numerickým rozlišením 16 bitového čísla s pevnou řádovou čárkou je však nejmenší kladné číslo 2^{-15}, což řádově odpovídá číslu 10^{-5}. Pokud bychom tímto číslem vynásobili jakékoli číslo z dostupného rozsahu $< -32767; 32768)$, téměř vždy bude výsledkem 0, protože v systému s pevnou řádovou čárkou neexistují desetinná čísla. Pro hodnotu konstanty 10^{-3} platí podobná limitace, pouze pro čísla z rozsahu $< -2^{-10}; 2^{1}0 >$ atd.

Role této konstanty je tedy při implementaci v rámci takového systému dvousečná, protože bude-li velikost adaptačního kroku příliš nízká, zastaví se adaptace příliš brzy (chybový signál bude příliš brzy roven 0) a výsledné řešení se dostačně nepřiblíží optimálnímu řešení. Pokud však bude hodnota konstanty μ příliš vysoká, nebude adaptace stabilní nebo bude oscilovat z důvodu neúměrně velkého adaptačního kroku okolo optimálního řešení. Za jiných okolností by se přesnost adaptace dala zvýšit právě snížením konstanty μ, ale to jsme zase zpět na začátku.

Hodnota konstanty μ je při realizaci LMS adaptivních filtrů v systému s pevnou řádovou čárkou zároveň tím nejkritičtějším a zároveň tím nejomezenějším parametrem a nalezení vhodné hodnoty, pro kterou bude algoritmus stabilní a zároveň dostatečně přesný je velmi náročné. V případě použití systému s plovoucí řádovou

čárkou by tento problém neměl prakticky nikdy nastat.

Limitace při implementaci IIR filtrace

Implementace IIR filtrace v systémech s pevnou řádovou čárkou je dlouho diskutovaným tématem, viz literatura [12], protože naprostá většina navržených filtrů je definována pomocí koeficientů, z nichž alespoň jeden je větší než 1. V takovém případě je nutné úměrně snížit všechny koeficienty takového filtru do rozsahu <-1;1) a při implementaci IIR filtru pak tuto skutečnost na vhodném místě kompenzovat, podobně jako v sekci 12.3.3. Tento proces bohužel snižuje již tak nízké numerické rozlišení výpočtu.

Tyto koeficienty jsou však v rámci této transformace nejen transformovány do rozsahu $< -32767; 32768)$, ale také jsou v rámci tohoto rozsahu kvantovány na celá čísla. Při této kvantizaci vzniká relativně závažná chyba, která se mimo možnosti narušit stabilitu filtru posunem pólu mimo rozsah jednotkové kružnice projevuje také zvýšením kvantovacího šumu.

Právě tento zvýšený kvantovací šum pak může při realizaci IIR filtrace v systému s pevnou řádovou čárkou vést ke vzniku tzv. mezních cyklů, které jsou způsobeny zaokrouhlováním výstupní hodnoty filtru, která (již zaokrouhlená) je následně opět použita pro výpočet zpětnovazebních smyček IIR filtrace.

Právě tyto mezní cykly se v této práci při implementaci IIR filtru projevují, ale jejich úroveň se pohybuje přibližně 15 až 20 dB pod běžnou hladinou okolního šumu (tzn. vstupního signálu IIR filtru) - jsou tedy postřehnutelné jen v případě, že je okolní prostředí nezvykle tiché.

Tento fenomén je pevně spojen se systémy s pevnou řádovou čárkou a řešením je opět použít systém s plovoucí řádovou čárkou. Zajímavým řešením by mohlo být také přesunutí kompenzačního IIR filtru z číslicové do analogové domény mezi D/A převodník a reproduktor sluchátka.

13 Závěr

V rámci této práce byly popsány jednotlivé používané topologie ANC systémů, vysvětlen jejich princip, výhody a nevýhody. Následně byla zvolena analogová i číslicová platforma pro realizaci vlastního systému sluchátek s adaptivním potlačením okolního šumu.

Po kontrole a ověření vlastností zvolených komponent a teoretickém úvodu do adaptivních filtrů byl nejprve navržen a realizován adaptivní systém v simulačním programu. V rámci těchto simulací byly popsány funkce jednotlivých stavebních bloků takového systému a následně byla na modelu sestavena struktura budoucího programu.

Na základě této struktury byl pak navržen a realizován výsledný program pro DSP čip, který realizoval jak adaptaci koeficientů adaptivních filtrů, tak i samotnou filtraci.

Oba dosažené výsledky, simulovaný i reálně změřený byly individuálně popsány i vzájemně srovnány. Oba výsledky jsou si velmi blízké a oba systémy, simulovaný i reálný, dosahují velmi dobrých výsledků.

Na závěr práce bylo otevřeno téma, zda-li byl použitý DSP procesor tou nejúspěšnější vlobou a zda-li by nebylo vhodnější zvolit pro realizaci tohoto programu raději aritmetiku s plovoucí řádovou čárkou.

Výsledkem této práce je fungující řešení systému adaptivních sluchátek pro potlačení okolního šumu v rámci aritmetiky s pevnou řádovou čárkou a také nalezení hranic, které se s tímto systémem nedají překonat.

Literatura

[1] CHASSAING, R.: *Digital signal processing and applications with the C6713 and C6416 DSK*, Hoboken, N.J, 2005, Wiley-Interscience, ISBN 978-0-471-70406-5.

[2] VASEGHI, S. *Advanced digital signal processing and noise reduction*, Chichester, West Sussex, England Hoboken, 2006, Wiley, ISBN 0-470-09494-X.

[3] YADAV, G.; KRISHNA, B.: Study of different adaptive filter algorithms for noise cancellation in real-Time environment[online], *International Journal of Computer Applications*, 2014. Dostupné z url: <https://www.researchgate.net/publication/330902011_Study_of_different_adaptive_filter_algorithms_for_noise_cancellation_in_real-Time_environment>.

[4] SIREESHA, N.; KRISHNASWAMY, CH.; SUDHAKAR, T.: Adaptive filtering based on least mean square algorithm, *International Symposium on Ocean Electronics*, 2013 Ocean Electronics (SYMPOL), s. 42-48. Dostupné z url: <https://www.researchgate.net/publication/261449615_Adaptive_filtering_based_on_least_mean_square_algorithm>.

[5] AKHTAR, M.; ABE, M.; KAWAMATA, M.: *Modified Filtered-x LMS Algorithm Based Cross-updating Active Noise Control System with Online Secondary-Path Modeling*[online], 2020. Dostupné z url: <https://www.researchgate.net/publication/265117413_6B3L-3-1_Modified_Filtered-x_LMS_Algorithm_Based_Cross-updating_Active_Noise_Control_System_with_Online_Secondary-Path_Modeling>.

[6] SYSEL, P.; SMÉKAL, Z.: *Číslicové filtry*, skriptum, Brno, 2012, Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací.

[7] SYSEL, P.: *Signálové procesory*, skriptum, Brno, 2015, Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, ISBN 978-80-214-5187-2.

[8] SYSEL, P. *Signálové procesory - laboratorní cvičení*, Brno, 2014, Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, ISBN 978-80-214-5204-6.

[9] *Smaart v8 User Guide [v8.3]* 224 stran. poslední aktualizace 2018 [cit. 2020-05-23]. Dostupné z URL:

<https://www.rationalacoustics.com/download/
Smaart-v8-User-Guide.pdf>.

[10] SMITH, J.: *Introduction to digital filters : with audio applications*, Stanford, California, 2008, Center for Computer Research in Music and Acoustics (CCRMA), Department of Music, Stanford University, ISBN 978-0974560717.

[11] WILLISTON, K.: *Digital signal processing : world class designs*, Amsterdam, Boston, 2009, Newnes/Elsevier, ISBN 978-1-85617-623-1.

[12] CHRISTENSEN M.; TAYLOR F. J.: *Fixed-point-IIR-filter challenges* [online], *EDN Network*, Florida, 2006, UNIVERSITY OF FLORIDA AND THE ATHENA GROUP INC. Dostupné z URL:
<https://api.semanticscholar.org/CorpusID:52245941>.

[13] *Spectrum Digital*: *TMS320C6416T DSK Technical Reference*[online] poslední aktualizace 2004. Dostupné z URL:
<http://c6000.spectrumdigital.com/dsk6416/V3/docs/dsk6416_
TechRef.pdf>.

[14] *Texas Instruments*: *C6000 Integreation Workshop, Revision 3.1a*[online] poslední aktualizace Srpen 2005. Dostupné z URL:
<https://e2e.ti.com/cfs-file/__key/communityserver-discussions-components-fi
791/iw6000_5F00_workshop.pdf>.

[15] *Texas Instruments*: *TMS320C6000 Programmer's Guide*[online] poslední aktualizace Červenec 2011, SPRU198K. Dostupné z URL:
<http://www.ti.com/lit/ug/spru198k/spru198k.pdf>.

[16] *Texas Instruments*: *TMS320C64x DSP Library Programmer's Reference*[online] poslední aktualizace Říjen 2003, SPRU565B. Dostupné z URL:
<http://www.ti.com/lit/ug/spru565b/spru565b.pdf>.

[17] *Texas Instruments*: *TLV320AIC23 Audio Codec Data Manual*[online] poslední aktualizace Únor 2004, SLWS106H. Dostupné z URL:
<https://www.ti.com/lit/ds/symlink/tlv320aic23b.pdf>.

[18] GAN, W. S.; MITRA S.; KUO S.: Adaptive feedback active noise control headset: implementation, evaluation and its extensions. *IEEE Transactions on Consumer Electronics*, 2005, vol. 51, no. 3, s. 975-982. ISSN 1558-4127.

[19] BJARNASO, E.: Analysis of the filtered-X LMS algorithm. *IEEE International Conference on Acoustics Speech and Signal Processing*, 1993. ISSN 1520-6149.

[20] KUO, S. M.; KONG, X.; GAN, W. S.: Applications of adaptive feedback active noise control system. *IEEE Transactions on Control Systems Technology*, 2003, vol. 11, no. 2, s. 216 - 220. ISSN 1558-0865.

[21] *Texas instruments - Design of Active Noise Control Systems with the TMS320 Family* 162 stran. poslední aktualizace 1996 [cit. 2019-05-15]. Dostupné z URL: <http://www.ti.com/lit/an/spra042/spra042.pdf>.

[22] *Texas instruments - Implementing a Noise Cancellation System with the TMS320C31* 23 stran. poslední aktualizace 1996 [cit. 2019-05-15]. Dostupné z URL:
<http://www.tij.co.jp/jp/lit/an/spra335/spra335.pdf>.

[23] *Texas instruments - Implementing a Single Channel Active Adaptive Noise Canceller with the TMS320C50 DSP Starter Kit* 32 stran. poslední aktualizace 1997 [cit. 2019-05-15]. Dostupné z URL:
<http://www.ti.com/jp/lit/an/spra285/spra285.pdf>.

[24] KUO, S. M.; MITRA, S.; GAN, Woon-Seng: Active noise control system for headphone applications. *IEEE Transactions on Control Systems Technology*, 2006, vol. 14, no. 2, s. 331-335. ISSN 1558-0865.

[25] ALVAREZ, A. M.; LEITCH, R. R.: Active attenuation of acoustic noise using adaptive algorithms. *IEEE International Symposium on Circuits and Systems*, Espoo, Finland, 1988, vol. 1, s. 511-514. Dostupné z url:
<https://ieeexplore.ieee.org/document/14976/>.

[26] *AMS datasheet - AS3415/AS3435 [v1-30]* 102 stran. poslední aktualizace 13.6.2014 [cit. 2018-12-11]. Dostupné z URL:
<https://ams.com/documents/20143/36005/AS3415_35_DS000299_1-00.pdf/3cf70528-ebbe-b1c9-7607-04af31b150ae/>.

[27] SIRAVARA, B.; MANSOUR, M.; COLE, R.; MAGORTA, N.: Comparative study of wideband single reference active noise cancellation algorithms on a fixed-point DSP. *2003 IEEE International Conference on Acoustics, Speech, and Signal Processing*, Hong Kong 2003, vol. 2, s. II-473. ISSN 1520-6149.

[28] GETHER, H.: *Active noise cancellation: Trends, concepts, and technical challenges.* [online] *EDN Network*, poslední aktualizace 8.10.2013 [cit. 2018-12-11]. Dostupné z URL:
<https://www.edn.com/design/consumer/4422370/Active-noise-cancellation--Trends--concepts--and-technical-challenges/>.

[29] GETHER, H.: *Design an ANC headset using the AS3415.* [online] *EDN Network*, poslední aktualizace 8.4.2014 [cit. 2018-12-11]. Dostupné z URL: <https://www.edn.com/design/consumer/4429817/Design-an-ANC-headset-using-the-AS3415/>.

[30] GETHER, H.: *Designing a feedback ANC headset using AS3435.* [online] *EDN Network*, poslední aktualizace 26.2.2015 [cit. 2018-12-11]. Dostupné z URL: <https://www.edn.com/design/consumer/4438751/Designing-a-feedback-ANC-headset-using-AS3435/>.

[31] GETHER, H.: *A perspective on digital ANC solutions in a low latency dominated world.* [online] *EDN Network*, poslední aktualizace 19.6.2017 [cit. 2018-12-11]. Dostupné z URL: <https://www.edn.com/design/analog/4458544/A-perspective-on-digital-ANC-solutions-in-a-low-latency-dominated-world/>.

Seznam symbolů, veličin a zkratek

ANC	aktivní potlačení šumu – Active Noise-Canceling
fft	rychlá fourierova transformace - Fast Fourier Transform
A/D	převodník z analogové do digitální domény - Analog to Digital
D/A	převodník z digitální do analogové domény - Digital to Analog
IIR	filtr s nekonečnou impulzní odezvou - Infinite Impuls Response
FIR	filtr s konečnou impulzní odezvou - Finite Impulse Response
DSP	číslicové zpracování signálů – Digital Signal Processing
OZ	operační zesilovač
RLS	Recusrive Least Squares
LMS	Least Mean Squares
DSK	Vývojová deska s DSP čipem TMS320C6416T