Rafael Rodrigo Pertum

IMPLEMENTAÇÃO, AVALIAÇÃO DE DESEMPENHO E ANÁLISE DE COMPLEXIDADE COMPUTACIONAL DE UM SISTEMA DE CANCELAMENTO ADAPTATIVO DE RUÍDO

TOLEDO

2020

Rafael Rodrigo Pertum

IMPLEMENTAÇÃO, AVALIAÇÃO DE DESEMPENHO E ANÁLISE DE COMPLEXIDADE COMPUTACIONAL DE UM SISTEMA DE CANCELAMENTO ADAPTATIVO DE RUÍDO

Trabalho de conclusão de curso apresentado ao Curso de Engenharia Eletrônica da Universidade Tecnológica Federal do Paraná, Câmpus Toledo, como requisito parcial para a obtenção do título de Bacharel em Engenharia Eletrônica.

Orientador: Prof. Evandro Marcos Kolling

TOLEDO

2020

TERMO DE APROVAÇÃO

Título do Trabalho de Conclusão de Curso Nª 105

Implementação, avaliação de desempenho e análise de complexidade computacional de um sistema de cancelamento adaptativo de ruído

Rafael Rodrigo Pertum

Esse Trabalho de Conclusão de Curso foi apresentado como requisito parcial para a obtenção do título Bacharel em Engenharia Eletrônica. Após deliberação da Banca Examinadora, composta pelos professores abaixo assinados, o trabalho foi considerado APROVADO.

Toledo, 6 de julho de 2020.

_____

Prof. Felipe Walter Dafico Pfrimer
Coordenador do Curso de Engenharia Eletrônica

Banca Examinadora:

_____

Prof. Evandro Marcos Kolling
UTFPR - Toledo

_____

Prof. Alberto Vinicius de Oliveira
UTFPR - Toledo

_____

Khaled Jamal Bakri,
Profissional externo, UFSC - Florianópolis

O termo de aprovação assinado encontra-se na coordenação do curso.

Dedico este trabalho aos meus pais Luis e Nilza.

AGRADECIMENTOS

Deixo aqui registrados meus sinceros agradecimentos a todos que contribuíram de alguma forma para a realização deste trabalho, em especial:

Aos meus pais pelo incentivo, apoio, dedicação, compreensão e, principalmente, pela paciência ao longo dos últimos anos.

Ao Prof. Eduardo Vinicius Kuhn pela orientação, amizade, motivação e dedicação, fundamentais na minha formação acadêmica e profissional.

Ao Prof. Evandro Marcos Kolling pelas contribuições acadêmicas, colaboração e suporte na realização deste trabalho.

Ao Prof. Alberto Vinicius de Oliveira pelas valorosas contribuições durante a minha formação acadêmica, seja ensinando, oferecendo conselhos e conversando sobre os mais diversos assuntos.

Aos membros da banca examinadora pelas pertinentes considerações, contribuições e sugestões.

Ao Prof. Rui Seara, Daniel Zanco, Rogério Paludo, João Paulo Acosta Luz, Artur Falkovski, Ênio dos Santos Silva e Elton Luiz Fontão pela infraestrutura disponibilizada, pelo apoio e acolhimento durante o período em que estive no LINSE-Laboratório de Circuitos e Processamento de Sinais da Universidade Federal de Santa Catarina, Campus Florianópolis, para a realização de parte deste trabalho.

Não menos importante, aos meus amigos Jackson Lewandowski e Danielly Loureiro, pelas valiosas conversas e pelo apoio psicológico no decorrer dos últimos anos.

Estendo ainda os agradecimentos a todos que de alguma forma contribuíram, direta ou indiretamente, para a minha formação acadêmica e profissional.

RESUMO

Atualmente, sistemas de cancelamento adaptativo de ruído (ANC, *adaptive noise cancelling*) vêm sendo utilizados em diversas aplicações práticas, como, por exemplo, na supressão de ruído em aparelhos auditivos, de áudio-conferência e celulares. Nessas aplicações, visto que as fontes de ruído podem variar com o tempo, técnicas de filtragem adaptativa são empregadas para sintetizar (em tempo real) uma estimativa do ruído que corrompe o sinal de interesse para, então, atenuá-lo (por subtração) no domínio elétrico. No entanto, apesar dos esforços despendidos até então, tem-se investido pouco em metodologias de implementação prática, em comparações de desempenho entre os algoritmos adaptativos empregados (frente à capacidade de atenuação de ruído alcançada), bem como em análises de complexidade computacional. Diante disso, o presente trabalho trata da implementação, avaliação de desempenho e análise de complexidade computacional de um sistema de ANC, operando com relevantes algoritmos adaptativos da literatura. Tal implementação foi realizada em linguagem C utilizando a placa de desenvolvimento Cortex-FM4 *Starter* da *Cypress Semiconductors*®, a qual possui recursos para aquisição, síntese e processamento de sinais de áudio. Para avaliar o desempenho dos algoritmos adaptativos implementados (frente a sinais de fala contaminados por ruído), metodologias objetivas de qualidade e inteligibilidade foram realizadas. Ainda, tendo como base o tempo de execução de cada função, uma comparação da complexidade computacional dos algoritmos é apresentada, evidenciando assim os requisitos computacionais e algumas características importantes dos algoritmos considerados.

Palavras-chave: Controle de ruído, filtros adaptativos, processamento digital de sinais.

ABSTRACT

Nowadays, adaptive noise cancelling (ANC) systems have been used in many different practical applications, such as in noise suppression in hearing aids, audio-conference devices and cellphones. In these applications, since sources of noise can change over time, adaptive filtering techniques are used to synthesize (in real-time) an estimate of noise that corrupts the signal of interest to, then, mitigate it (by subtraction) in the electrical domain. However, only a few efforts have been invested on the practical implementation of adaptive algorithms, on methodologies for performance comparison (with respect to the achieved noise attenuation), as well on computational complexity analyzes. In this context, the present research work deals with implementation, performance evaluation, and computational complexity analysis of an ANC system operating with relevant adaptive algorithms from the literature. Such implementation was carried out in C language using the Cortex FM4 Starter development board from Cypress Semiconductors®, which has acquisition, synthesis, and signal processing resources. In order to assess the performance of the implemented algorithms (vis-à-vis speech signals corrupted by noise), objective methodologies of quality and intelligibility were used. Still, based on the execution time of each function used, a comparison between the computational complexity of the algorithms is presented, thus evidencing computational requirements and important characteristics of the algorithms considered.

Keywords: Noise control, adaptive filters, digital signal processing.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ANC	*Adaptive noise cancelling*
API	*Application programming interface*
CMSIS-DSP	*Cortex microcontroller software interface standard – digital signal processing*
DMA	*Direct memory access*
I2S	*Inter*-IC *sound*
IDE	*Integrated development environment*
ITU	*International Telecommunication Union*
ITU-T	ITU – *Telecommunication Standardization Sector*
LMS	*Least-mean-square*
LQO	*Listen quality objective*
MDK	*Microcontroller development kit*
MOS-LQO	*Mean opinion score* – LQO
NLMS	*Normalized* LMS
PDL	*Peripheral driver library*
PESQ	*Perceptual evaluation of speech quality*
SNR	*Signal-to-noise ratio*
STOI	*Short-time objective intelligibility measure*
UART	*Universal Asynchronous receiver/transmitter*
USB	*Universal serial bus*
VSS-NLMS	*Variable step-size* – NLMS

LISTA DE SÍMBOLOS

n	Instante de tempo discreto
$x(n)$	Sinal de referência
$e(n)$	Sinal de erro
$\hat{d}(n)$	Estimativa do ruído
$d(n)$	Sinal de ruído
$s(n)$	Sinal de interesse
$\mathbf{w}(n)$	Vetor de coeficientes do filtro adaptativo
M	Ordem do filtro adaptativo
$\mathbf{x}(n)$	Vetor contendo as amostras mais recentes do sinal de referência
μ	Passo de adaptação
ε	Parâmetro de regularização
$\mu(n)$	Valor do passo de adaptação dos algoritmos VSS-NLMS
C	Constante positiva inversamente proporcional a SNR utilizada no algoritmo VSS-NLMS de Shin
$\mu_{máx}$	Valor máximo do passo de adaptação no algoritmo de Shin
β	Constante (positiva) de suavização
$\mathbf{p}(n)$	Estimativa da correlação entre o sinal de referência e sinal de erro para o algoritmo VSS-NLMS de Shin
$\gamma(n)$	Variável auxiliar relacionada ao passo de adaptação do algoritmo VSS-NLMS de Benesty
δ	Constante de regularização do algoritmo VSS-NLMS de Benesty
$\hat{\sigma}_e^2(n)$	Estimativa da variância do sinal de erro
λ_1	Constante (positiva) de suavização utilizada no algoritmo VSS-NLMS de Benesty
K	Constante utilizada no algoritmo VSS-NLMS de Benesty
$\hat{\sigma}_v^2(n)$	Estimativa da variância do ruído de medição
$\hat{\sigma}_x^2(n)$	Estimativa da variância do sinal de referência
$\hat{\mathbf{r}}_{ex}(n)$	Estimativa da correlação cruzada entre o sinal de entrada e o sinal de erro
λ_2	Constante (positiva) de suavização utilizada no algoritmo VSS-NLMS de Benesty
$p(n)$	Estimativa de correlação do sinal de erro
$q(n)$	Estimativa da potência do sinal de erro

SUMÁRIO

1. INTRODUÇÃO

Sistemas de cancelamento adaptativo de ruído (ANC, *adaptive noise cancelling*) são empregados em diferentes aplicações práticas, tais como, na supressão de ruído em aparelhos auditivos, de áudio-conferência ou celulares (PANAHI; KEHTARNAVAZ; THIBODEAU, 2016; SUGIYAMA; MIYAHARA; OOSUGI, 2019; VANDEN BERGHE; WOUTERS, 2005), como também em equipamentos biomédicos (WIDROW; STEARNS, 1985; YELDERMAN et al., 1983). Nessas aplicações, um algoritmo adaptativo atualiza uma estrutura de filtragem que sintetiza uma estimativa do ruído que corrompe o sinal de interesse, visando, então, atenuá-lo (por subtração) no domínio elétrico. Dentre os diferentes algoritmos usados para adaptar a estrutura de filtragem, se destacam os algoritmos LMS (*least-mean-square*) e NLMS (*normalized* LMS), tanto pela facilidade de implementação quanto pela boa capacidade de adaptação independente do ambiente de operação considerado (HAYKIN, 2014). Ainda nesse contexto, os algoritmos VSS-NLMS (*variable step-size* NLMS) se mostram uma alternativa interessante, já que proporcionam um melhor desempenho em termos de velocidade de convergência e reduzido erro em regime permanente; sobretudo, quando comparados com os algoritmos LMS e NLMS (SAYED, 2008).

Apesar de resultados de simulação atestarem usualmente o bom desempenho dos algoritmos adaptativos mencionados, um sistema de ANC está sujeito a diversos outros fatores práticos de projeto impostos pela aplicação considerada. Como exemplo, pode-se citar a capacidade de processamento do microcontrolador, as características dos sinais envolvidos, os cenários de operação, o número de coeficientes da estrutura de filtragem, ou ainda a taxa de amostragem utilizada. Negligenciar isso no momento do projeto de um sistema de ANC pode prejudicar seu desempenho e, consequentemente, afetar a qualidade final do sinal processado. Por isso, em se tratando sobretudo de aplicações envolvendo o processamento de sinais de fala, metodologias para avaliações subjetivas (ITU-T, 1996) e/ou objetivas (TAAL et al., 2011) relacionadas à qualidade e à inteligibilidade são fundamentais para a avaliação de sistemas de ANC. Nessas metodologias, o sinal original e aquele obtido como resultado de algum processamento são avaliados por um grupo de pessoas ouvintes (em avaliações subjetivas) (ITU-T, 1996) ou através de métodos computacionais (em avaliações objetivas) (LOIZOU, 2017; TAAL et al., 2011), resultando em uma pontuação que indica a qualidade ou inteligibilidade do sinal processado. Contudo, visto que metodologias para avaliações subjetivas são excessivamente demoradas e custosas, avaliações objetivas vêm sendo preferencialmente utilizadas (LOIZOU, 2017).

Neste contexto, os trabalhos de (PANAHI; KEHTARNAVAZ; THIBODEAU, 2016), (SUGIYAMA; MIYAHARA; OOSUGI, 2019) e (VANDEN BERGHE; WOUTERS, 2005) apresentam implementações práticas envolvendo sistemas de ANC. Especificamente, (VANDEN BERGHE; WOUTERS, 1998) descreve um sistema de ANC para um aparelho auditivo utilizando dois microfones próximos, o qual foi avaliado por usuários de aparelhos auditivos com audição normal e com perda de audição moderada. Já (PANAHI; KEHTARNAVAZ; THIBODEAU, 2016) implementa um sistema de ANC em um *smartphone* para melhorar a percepção de sinais de fala por usuários de aparelho auditivo. Em tal trabalho, os autores avaliaram o sistema implementado por meio de metodologias de avaliação subjetivas e objetivas, considerando sinais de fala corrompidos por ruído de balbúrdia, de máquinas em funcionamento e de um automóvel em movimento para valores de razão sinal ruído (SNR, *signal-to-noise ratio*) de -5 dB e 0 dB. Por sua vez, (SUGIYAMA; MIYAHARA; OOSUGI, 2019) apresenta a implementação de um dispositivo (microfone) com um sistema de ANC embarcado. Esse trabalho considera um dispositivo específico para o processamento digital de sinais, conduz testes subjetivos de inteligibilidade envolvendo sinais de fala e compara o número de ciclos de *clock* gastos com outra solução comercializada pela empresa Qualcomm®. Contudo, os trabalhos mencionados não trazem comparações de desempenho envolvendo diferentes algoritmos adaptativos da literatura, não apresentam uma metodologia de avaliação padronizada e não consideram diferentes cenários de ruído e de SNR.

1.1. Objetivos

O presente trabalho tem por objetivo a implementação, a avaliação de desempenho e a análise de complexidade computacional de um sistema de ANC, operando com relevantes algoritmos da literatura, voltado ao tratamento de sinais de fala contaminados por ruído. Especificamente, busca-se

i) implementar um sistema de ANC utilizando a placa de desenvolvimento Cortex-FM4 *Starter* da *Cypress Semiconductors*®, fornecendo assim uma metodologia de implementação unificada;

ii) realizar a implementação de quatro algoritmos adaptativos da literatura, isto é, o algoritmo NLMS, bem como os algoritmos VSS-NLMS introduzidos por Shin (SHIN; SAYED; SONG, 2004), Benesty (BENESTY et al., 2006) e Zipf (ZIPF; TOBIAS; SEARA, 2010);

iii) prover estratégias para estimação de variáveis desconhecidas e requeridas para a operação dos algoritmos considerados;

iv) avaliar o desempenho dos algoritmos implementados, por meio de metodologias objetivas, em termos de qualidade e inteligibilidade dos sinais de fala considerando diferentes cenários de ruído (tais como de ruído branco gaussiano, ruído de balbúrdia e ruído de obras em construção); e

v) analisar a complexidade computacional dos algoritmos em termos do tempo de execução das funções utilizadas e do tempo requerido para processar um quadro do sinal.

Vale salientar que o desenvolvimento do presente trabalho de conclusão de curso teve início através de um projeto de iniciação científica em 2017, o qual se estendeu até 2019. Ao longo desse período, partes deste trabalho foram publicadas em relevantes eventos da área de processamento digital de sinais (conforme devidamente indicado na Seção 6), as quais são também reproduzidas aqui a fim de tornar o documento autocontido. Apesar disso, é importante enfatizar que o presente trabalho estende os resultados até então publicados da seguinte forma:

a) são agora considerados cenários de ruído mais realistas (além do caso de ruído branco gaussiano);

b) é adicionada uma estratégia de *anti-clipping* na síntese dos sinais utilizados;

c) são introduzidas modificações necessárias na implementação do algoritmo VSS-NLMS de Benesty; e

d) é realizada uma análise de complexidade computacional dos algoritmos adaptativos considerados.

2. FUNDAMENTAÇÃO TEÓRICA

Nesta seção, a topologia de um sistema de ANC bem como as expressões do algoritmo NLMS (HAYKIN, 2014) e daqueles descritos em (BENESTY et al., 2006; SHIN; SAYED; SONG, 2004; ZIPF; TOBIAS; SEARA, 2010) são brevemente revisitadas. Vale salientar que a notação matemática adotada segue a prática padrão de usar letras minúsculas em negrito para vetores, enquanto letras romanas em itálico e letras gregas são utilizadas para denotar escalares.

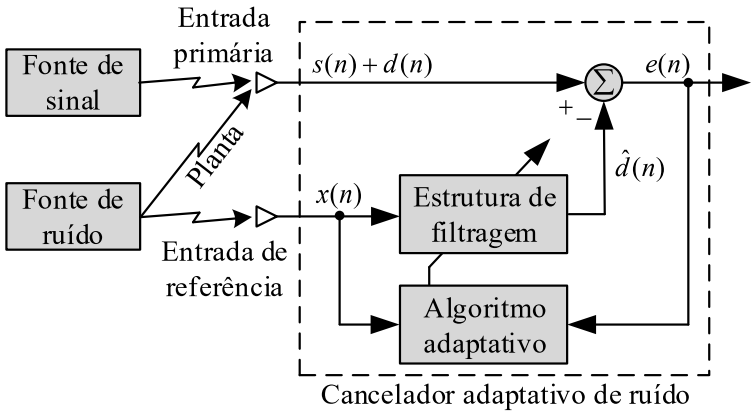
2.1. Topologia de um sistema de ANC

Em um sistema de ANC (conforme ilustrado na Figura 1), um algoritmo adaptativo [utilizando um sinal de referência $x(n)$ e o sinal de erro $e(n)$ no instante de tempo n] ajusta os coeficientes de uma estrutura de filtragem de forma a produzir uma estimativa do ruído $d(n)$ que corrompe o sinal de interesse $s(n)$. Então, realizando a subtração entre o sinal corrompido por ruído [observado na entrada primária como $s(n)+d(n)$] e a estimativa do ruído $\hat{d}(n)$, obtém-se o sinal de erro como

$$e(n) = s(n) + d(n) - \hat{d}(n). \tag{1}$$

Dessa forma, conforme o algoritmo converge para a solução em regime permanente [isto é, $\hat{d}(n) \to d(n)$ conforme $n \to \infty$], o sinal de erro tende para o sinal de interesse [isto é, $e(n) \to s(n)$ para $n \to \infty$], produzindo assim a atenuação do ruído na saída do sistema (como desejado).

Figura 1 – Topologia de um sistema de ANC.



Fonte: Adaptado de Widrow (1985).

2.2. Algoritmo NLMS

Assumindo uma estrutura transversal de filtragem, a equação de adaptação do algoritmo NLMS pode ser definida como (HAYKIN, 2014)

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu \frac{\mathbf{x}(n)e(n)}{\mathbf{x}^{\mathrm{T}}(n)\mathbf{x}(n) + \varepsilon} \tag{2}$$

onde $\mathbf{w}(n) = [w_0(n) \cdots w_{M-1}(n)]^{\mathrm{T}}$ denota o vetor de coeficientes do filtro adaptativo de ordem M, $(\cdot)^{\mathrm{T}}$, o operador de transposição, $\mathbf{x}(n) = [x(n) \cdots x(n-M+1)]^{\mathrm{T}}$, um vetor contendo as amostras mais recentes da entrada de referência, $0 < \mu < 2$, o passo de adaptação e $\varepsilon > 0$, um parâmetro de regularização que visa evitar divisão por zero e estabilizar a solução.

2.3. Algoritmos VSS-NLMS considerados

Em algoritmos de passo variável, o passo de adaptação em (2) é feito variante no tempo, isto é, μ é substituído por $\mu(n)$. Dessa forma, (2) pode ser reescrita como (HAYKIN, 2014)

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu(n) \frac{\mathbf{x}(n)e(n)}{\mathbf{x}^{\mathrm{T}}(n)\mathbf{x}(n) + \varepsilon} \tag{3}$$

onde $\mu(n)$ caracteriza o passo de adaptação variável, o qual é ajustado iterativamente de acordo com alguma regra específica. Note que diferentes estratégias são utilizadas na literatura para derivar regras de ajuste para o passo de adaptação $\mu(n)$, dando assim origem a um grande número de algoritmos adaptativos de passo variável (alguns deles mostrados a seguir).

2.3.1. Algoritmo VSS-NLMS de Shin

O ajuste do passo de adaptação do algoritmo VSS-NLMS de Shin (SHIN; SAYED; SONG, 2004) é realizado de acordo com

$$\mu(n) = \mu_{\text{máx}} \frac{\mathbf{p}^{\mathrm{T}}(n)\mathbf{p}(n)}{\mathbf{p}^{\mathrm{T}}(n)\mathbf{p}(n) + C} \tag{4}$$

onde

$$\mathbf{p}(n) = \beta\mathbf{p}(n-1) + (1-\beta)\frac{\mathbf{x}(n)e(n)}{\mathbf{x}^{\mathrm{T}}(n)\mathbf{x}(n)} \tag{5}$$

com $0 < C \ll 1$ representando uma constante positiva inversamente proporcional a SNR, $\mu_{máx}$, o valor máximo que o passo de adaptação pode assumir, β, uma constante positiva de suavização e $\mathbf{p}(n)$, um vetor introduzido por Shin para caracterizar uma estimativa da correlação entre o sinal de referência e o sinal de erro [detalhes sobre o funcionamento do algoritmo são discutidos em (SHIN; SAYED; SONG, 2004)].

2.3.2. Algoritmo VSS-NLMS de Benesty

No algoritmo VSS-NLMS de Benesty (dito não paramétrico) (BENESTY et al., 2006), o passo de adaptação é determinado através de

$$\mu(n) = \begin{cases} \gamma(n), & \text{se } \hat{\sigma}_e(n) \geq \hat{\sigma}_v \\ 0, & \text{caso contrário} \end{cases} \tag{6}$$

com

$$\gamma(n) = 1 - \frac{\hat{\sigma}_v}{\delta + \hat{\sigma}_e(n)} \tag{7}$$

onde δ define uma constante de regularização, $\hat{\sigma}_e^2(n)$ representa uma estimativa da variância do sinal de erro obtida de

$$\hat{\sigma}_e^2(n) = \lambda_1 \hat{\sigma}_e^2(n-1) + (1-\lambda_1) e^2(n) \tag{8}$$

para

$$\lambda_1 = 1 - \frac{1}{KM}, \; K \geq 2 \tag{9}$$

e $\hat{\sigma}_v^2$ caracteriza uma estimativa da variância do ruído de medição. Vale salientar que (CIOCHINĂ; PALEOLOGU; BENESTY, 2016) sugerem que essa estimativa seja obtida de acordo com (IQBAL; GRANT, 2008), isto é,

$$\hat{\sigma}_v^2(n) = \hat{\sigma}_e^2(n) - \frac{1}{\hat{\sigma}_x^2(n)} \hat{\mathbf{r}}_{ex}^{\mathrm{T}}(n) \hat{\mathbf{r}}_{ex}(n) \tag{10}$$

com

$$\hat{\sigma}_x^2(n) = \lambda_2 \hat{\sigma}_x^2(n-1) + (1-\lambda_2) x^2(n) \tag{11}$$

e

$$\hat{\mathbf{r}}_{ex}(n) = \lambda_2 \hat{\mathbf{r}}_{ex}(n-1) + (1-\lambda_2) \mathbf{x}(n) e(n). \tag{12}$$

Note que $0 \ll \lambda_1 < 1$ e $0 \ll \lambda_2 < 1$ representam constantes de suavização distintas e, portanto, podem assumir valores diferentes.

2.3.3. Algoritmo VSS-NLMS de Zipf

No algoritmo VSS-NLMS de Zipf (ZIPF; TOBIAS; SEARA, 2010), a regra de ajuste para o passo de adaptação em (3) é descrita através de

$$\mu(n) = \frac{p^2(n)}{q^2(n)} \tag{13}$$

com

$$p(n) = \beta p(n-1) + (1-\beta)e(n)e(n-1) \tag{14}$$

denotando uma estimativa de correlação do sinal de erro e

$$q(n) = \beta q(n-1) + (1-\beta)e^2(n) \tag{15}$$

representando uma estimativa da potência do sinal de erro, sendo $0 \ll \beta < 1$ uma constante positiva de suavização [detalhes sobre o funcionamento do algoritmo são discutidos em (ZIPF; TOBIAS; SEARA, 2010)].

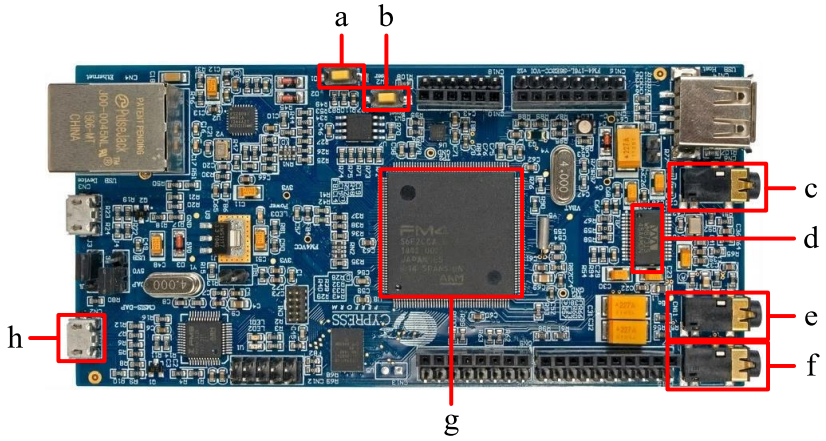
3. MATERIAIS E MÉTODOS

Nesta seção, aspectos pertinentes a implementação do sistema de ANC são apresentados. Especificamente, a Seção 3.1 apresenta uma breve introdução da placa de desenvolvimento e da IDE (*integrated development environment*) utilizadas; a Seção 3.2 traz uma descrição dos sinais de fala e tipos de ruído considerados; a Seção 3.3 mostra a estratégia de avaliação adotada; a Seção 3.4 revisita as metodologias objetivas de avaliação utilizadas; e, por fim, a Seção 3.5 explica o método adotado para realizar a análise da complexidade computacional dos algoritmos considerados.

3.1. Placa de desenvolvimento e IDE

Na implementação do sistema de ANC, uma placa de desenvolvimento Cortex-FM4 *Starter* da *Cypress Semiconductors* (mostrada na Figura 2) foi empregada, a qual possui um microcontrolador ARM® Cortex-M4 S6E2CC operando a 200 MHz (CYPRESS SEMICONDUCTORS, 2020). Essa placa de desenvolvimento utiliza um codec (*coder-decoder*) Wolfson® WM8731 para realizar a interface entre a entrada/saída de áudio (disponíveis através de conectores do tipo P2 estéreo) e o microcontrolador por meio do protocolo de comunicação I2S (*inter*-IC s*ound*). Além disso, a placa de desenvolvimento conta com um dispositivo específico para depuração via USB (*universal serial bus*), permitindo assim acompanhar a execução do código bem como acessar as variáveis de interesse armazenadas na memória. Por sua vez, para o desenvolvimento dos códigos, foi considerada a IDE Keil µVision® MDK (*microcontroller development kit*) versão 5 fornecida pela ARM® (ARM KEIL, 2020), a qual possibilita a programação em *assembly*, linguagem C e C++. Ainda, a IDE oferece suporte a biblioteca CMSIS-DSP (*cortex microcontroller software interface standard – digital signal processing*) versão 1.7.0, contendo funções otimizadas para realizar operações de processamento de sinais em microcontroladores do tipo Cortex-M.

Figura 2 – Placa de desenvolvimento ARM Cortex-M4 *Starter* da *Cypress Semiconductors*. (a) Botão de propósito geral. (b) Botão de *reset*. (c) Conector P2 de entrada de áudio (*line in*). (d) Codec de áudio Wolfson WM8731. (e) Conector P2 para microfone. (f) Conector P2 de saída de áudio (*headphone*). (g) Microcontrolador ARM® Cortex-M4 32-bit. (h) Conector USB.



Fonte: Adaptado de *Cypress Semiconductors*® (2020).

3.2. Sinais de áudio da entrada de referência e da entrada primária

No intuito de avaliar o desempenho do sistema de ANC frente à atenuação de ruído em sinais de fala, arquivos de áudio para a entrada de referência e entrada primária foram sintetizados. Dezesseis sentenças em português do Brasil (ITU-T, 2000) são utilizadas, para compor os sinais de fala, e três gravações da base *Noise Recordings* (LOIZOU, 2017), para compor os cenários de ruído. Especificamente, o ruído de balbúrdia é obtido de *cafeteria_babble.wav* e o de obras em construção, a partir de *Construction_Crane_Moving.wav* e *Construction_Jackhammer2.wav*. Adicionalmente, um arquivo de áudio com ruído branco gaussiano foi gerado para compor um terceiro cenário de ruído. Esses sinais de ruído são filtrados por uma planta obtida de (ITU-T, Modelo 4, 2015) e adicionados aos sinais de fala utilizando a rotina *addnoise_asl.m*[1] disponível em (LOIZOU, 2017). Tal rotina adiciona um sinal de ruído a um sinal de fala limpo (com um valor predefinido de SNR) considerando a parte ativa do sinal de fala, ou seja, levando em conta apenas o período em que o sinal de fala está presente. Destaca-se ainda que foram sintetizados neste trabalho arquivos de áudio para uma faixa de SNR de -20 dB até 20 dB, com intervalo de 2 dB. No entanto, para valores de SNR baixos (menores do que 0 dB), o sinal de áudio sintetizado apresentou o fenômeno de *clipping*. Tal fenômeno representa a saturação do sinal sintetizado, acarretando assim a perda da integridade dos sinais envolvidos e da característica

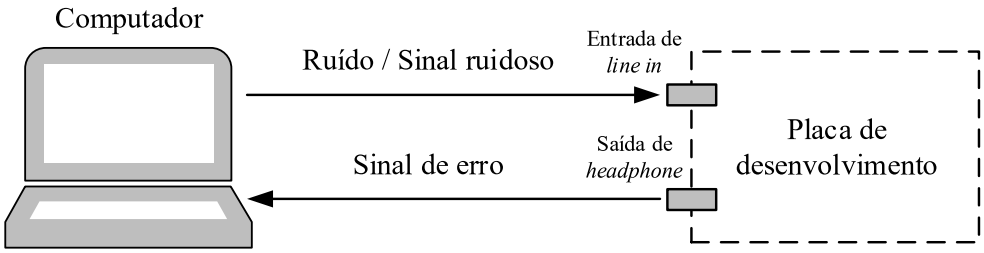
[1] Para otimizar a síntese dos sinais de áudio, a rotina *addnoise_asl.m* foi aqui transcrita para linguagem Python.

de SNR previamente desejada. Para evitar isso, foi adotada a estratégia de dividir os sinais de fala e ruído por um divisor comum, conforme apresentado no fluxograma da Figura 4. Essa estratégia consiste em verificar a ocorrência de *clipping* para todos os sinais sintetizados no intervalo de SNR analisado; então, na ocorrência de *clipping*, a variável *div* é incrementada e os processos de dividir os sinais de fala e ruído, filtrar o sinal de ruído pela planta e adicionar o ruído ao sinal de fala são repetidos. Quando a ocorrência de *clipping* não é mais identificada, os sinais resultantes são arquivados para o uso nas avaliações de desempenho do sistema de ANC, garantindo, portanto, a integridade e a característica de SNR desejada. Cabe destacar que todos os arquivos de áudio foram sintetizados com uma frequência de amostragem de 8 kHz.

3.3. Estratégia de avaliação

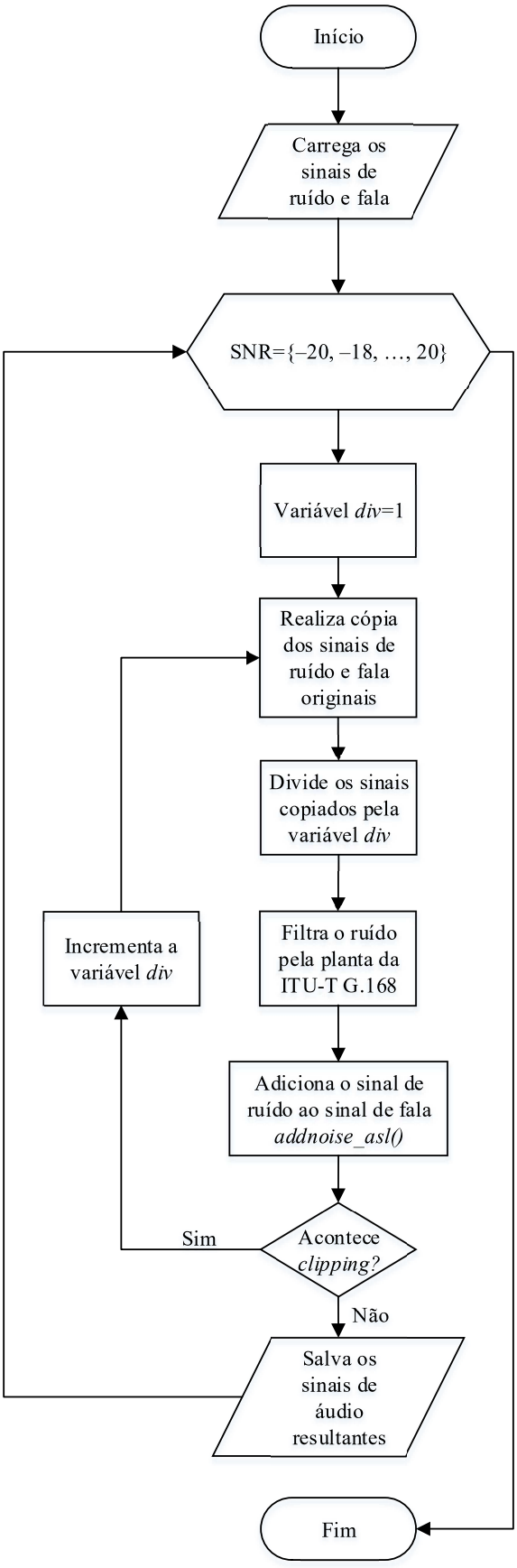
Conforme ilustrado na Figura 3, os sinais da entrada primária (sinal ruidoso) e da entrada de referência (ruído) foram reproduzidos em um computador, o qual estava conectado à entrada de *line in* (via cabo P2 estéreo) da placa de desenvolvimento. O canal esquerdo da entrada de *line in* foi utilizado como entrada de referência e o direito como entrada primária. Simultaneamente, o sinal de erro observado na saída do sistema (saída de *headphone*) foi capturado pelo computador, visando gerar os resultados experimentais. Note que a reprodução e captura foi realizada para os diferentes sinais de teste utilizados.

Figura 3 – Diagrama de blocos da estratégia de avaliação.



Fonte: Autoria própria.

Figura 4 – Fluxograma da estratégia de *anti-clipping*.



Fonte: Autoria própria.

3.4. Metodologias objetivas de avaliação

Três metodologias foram utilizadas para avaliar o desempenho do sistema, a saber: i) *perceptual evaluation of speech quality* (PESQ) (ITU-T, 2005) para avaliar a qualidade, bem como ii) *short-time objective intelligibility measure* (STOI) (TAAL et al., 2011) e iii) *application programming interface* (API) *speech-to-text* da Google Inc. (GOOGLE INC., [s.d.]) para avaliar a inteligibilidade.

3.4.1. Considerações sobre a avaliação de qualidade

A Recomendação P.862 da *International Telecomunication Union-Telecommunication Standardization Sector* (ITU-T) aponta a PESQ como um método de avaliação objetiva de qualidade de voz. Tal método consiste em comparar um áudio de referência (sinal original) com um áudio resultante de algum tipo de processamento em meio digital, o qual pode ter sofrido degradação. Essa avaliação é realizada via processamento computacional e tem como resultado uma pontuação entre $-0,5$ e $4,5$ (quanto menor a pontuação, mais baixa a qualidade do áudio avaliado). Então, para representar a pontuação na escala MOS (*mean opinion score*) (ITU-T, 1996), usualmente empregada em avaliações subjetivas, recorre-se à Recomendação P.862.1 da ITU-T. Tal recomendação descreve uma função de mapeamento entre a pontuação obtida através da PESQ e aquela da escala MOS, resultando assim na escala denominada PESQ (MOS-LQO) (MOS-*listening quality objective*). Dessa forma, a faixa de valores de $-0,5$ e $4,5$ é convertida para a faixa de 1 a 4,5, sendo 1 atribuído aos áudios de qualidade percebida mais baixa e 4,5 aos de qualidade percebida mais alta. Portanto, dispondo da implementação da PESQ dada em (ITU-T, 2005), as pontuações obtidas para cada áudio foram computadas e a média foi então contabilizada para cada valor de SNR utilizado.

3.4.2. Considerações sobre as avaliações de inteligibilidade

A STOI e a API *speech-to-text* da Google Inc. foram empregadas para conduzir avaliações objetivas de inteligibilidade. A partir de um áudio de referência (assim como na PESQ), a STOI é utilizada para avaliar quanto a inteligibilidade de um áudio obtido como resultado de algum processamento foi degradada, resultando em uma pontuação de 0 a 1 (0 indica pior inteligibilidade e 1, a melhor). Considerando a implementação da metodologia STOI disponível em (PARIENTE, 2020), resultados experimentais foram computados e o

desempenho médio foi obtido para cada valor de SNR avaliado. Por sua vez, a API *speech-to-text* da Google Inc., disponível em (GOOGLE INC., [s.d.]), foi utilizada para contabilizar a taxa de acerto do número de palavras corretamente reconhecidas pelo conversor fala-texto. Para tal, áudios degradados resultantes do processamento foram submetidos à API no intuito de se obter uma transcrição do sinal, comparada então com a transcrição original das sentenças presente na Recomendação P.50 da ITU-T (ITU-T, 2000). Em seguida, a taxa de acerto foi calculada como a razão entre o número total de palavras reconhecidas corretamente e o número total de palavras presentes nas transcrições das sentenças de referência. Vale destacar que as palavras corretamente reconhecidas possuíam mais de três letras e pertenciam à sentença de referência analisada. Dessa forma, os resultados experimentais foram computados para cada valor de SNR e o desempenho foi então caracterizado.

3.5. Considerações sobre a análise de complexidade computacional

Com o intuito de compreender melhor os requisitos computacionais dos algoritmos considerados, um contador decrescente de 32-/16-bit (isto é, um *dual timer* disponível no microcontrolador) foi utilizado para contabilizar o tempo que cada algoritmo leva para processar um quadro de 128 amostras e o tempo requerido para executar as principais funções utilizadas. Para isso, o contador foi iniciado antes da execução do código de interesse (definindo assim um valor de tempo inicial) e finalizado após a execução do código (resultando então em um valor de tempo final). Nesse processo, o tempo transcorrido foi determinado (como a diferença entre o tempo inicial e o tempo final) e encaminhado via comunicação *universal asynchrounous receiver/transmiter* (UART) para o computador, onde foi armazenado, comparado e analisado.

4. ROTINAS IMPLEMENTADAS E RESULTADOS OBTIDOS

Nesta seção, trechos de código desenvolvidos para o sistema de ANC e os resultados obtidos ao submeter o sistema implementado as diferentes metodologias de avaliação são apresentados. Particularmente, a Seção 4.1 apresenta uma descrição das rotinas implementadas em linguagem C para o microcontrolador ARM® Cortex-M4 S6E2CC, enquanto a Seção 4.2 apresenta os resultados das metodologias objetivas de avaliação e a análise de complexidade computacional. Os scripts criados, áudios utilizados e resultados obtidos estão disponíveis em (PERTUM; KUHN, 2020).

4.1. Rotinas implementadas em linguagem C

Para a implementação do sistema de ANC na placa de desenvolvimento Cortex-FM4 *Starter* da *Cypress Semiconductors*®, três rotinas em linguagem C foram implementadas, a saber: a rotina principal onde os periféricos utilizados são inicializados (codec, UART e *dual timer*); a rotina responsável por gerenciar o processamento do *buffer* contendo as amostras advindas do codec; e o conjunto com as diferentes rotinas referentes aos algoritmos adaptativos considerados.

4.1.1. Rotina principal

Na rotina principal (observada na Figura 5) são inicializados os periféricos da placa de desenvolvimento tais como o temporizador *dual timer*, o módulo de comunicação UART e o codec Wolfson® WM8731. As funções que inicializam o *dual timer* e a UART foram adaptadas de *templates* da biblioteca *peripheral driver library* (PDL) (CYPRESS SEMICONDUCTORS, 2020). Já a função `audio_init()`, adaptada de (ARM, 2020), inicializa o codec Wolfson® WM8731 para operar com uma frequência de amostragem de 8 kHz, por meio da entrada *line in* e com o uso do modo *direct memory access* (DMA)[2]. Por conveniência de implementação, foi empregado aqui um *buffer* de 128 amostras que é transmitido e recebido pelo codec via protocolo I2S. Após a inicialização dos periféricos, a rotina principal aguarda o *buffer* de recepção estar cheio (sinalizado através da *flag* `rx_buffer_full`) e o *buffer* de transmissão estar vazio (sinalizado por meio da *flag* `tx_buffer_full`) para, então, chamar a função `process_buffer()` que gerencia o processamento das amostras do *buffer*.

[2] Detalhes sobre o funcionamento do modo DMA utilizando a estratégia de *ping-pong buffer* são discutidos em (NEVES, 2016).

Figura 5 – Trecho de código da rotina principal.

```
int main (void) {
   //inicializa a comunicação UART
   Uart_Io_Init();

   //inicializa o dual timer
   if (my_dt() != 0){
         uart_printf("erro!\n");
         while(1);
   };

   //inicializa o CODEC
   audio_init(hz8000,line_in,dma,DMA_HANDLER);

   while(1){
         while(!(rx_buffer_full&&tx_buffer_empty)){};
         // processa o buffer
         proces_buffer();
   }
}
```

Fonte: Autoria própria.

4.1.2. Rotina de processamento e algoritmos implementados

A rotina `process_buffer()` gerencia o processamento das amostras mais recentes. Nessa rotina, tal como apresentada na Figura 6, é também contabilizado o tempo de execução dos algoritmos adaptativos implementados na função `run_filter()` (como descrito na Seção 3.5). Destaca-se ainda que, para cada algoritmo adaptativo implementado no sistema de ANC, foi empregado uma função `run_filter()` diferente. Trechos de código relacionados a implementação dos algoritmos [executados através da função `run_filter()`] são apresentados nas Figuras 7-10. Especificamente, a Figura 7 mostra a implementação do algoritmo NLMS (descrito na Seção 2.2), a Figura 8 apresenta a implementação do algoritmo VSS-NLMS de Shin (revisitado na Seção 2.3.1), a Figura 9 mostra a implementação do algoritmo VSS-NLMS de Benesty (discutido na Seção 2.3.2), enquanto a Figura 10 apresenta a implementação do algoritmo VSS-NLMS de Zipf (dado na Seção 2.3.3). Em tais implementações, foram utilizadas funções de processamento de sinais fornecidas na biblioteca CMSIS-DSP, visando otimizar o código e facilitar a implementação. Note que a estrutura de filtragem tem 128 coeficientes e que os parâmetros dos diferentes algoritmos foram ajustados conforme indicado na Tabela 1. Tais ajustes foram realizados com base no melhor desempenho observado para um áudio contaminado com ruído branco gaussiano de 0 dB, como também seguindo recomendações dadas em (HAYKIN, 2014) para o algoritmo NLMS, (SHIN; SAYED; SONG, 2004) para o algoritmo VSS-NLMS de Shin, (BENESTY et al.,

2006) para o algoritmo VSS-NLMS de Benesty e em (ZIPF; TOBIAS; SEARA, 2010) para o algoritmo VSS-NLMS de Zipf.

Figura 6 – Trecho de código da rotina de processamento.

```
void proces_buffer(void){
...
   // Escreve no timer o tempo inicial
   // da contagem decrescente
   Dt_WriteLoadVal(tempo_inicial, DtChannel0);

   // Habilita a contagem
   Dt_EnableCount(DtChannel0);

   // realiza o processamento pelo algoritmo implementado
   run_filter(txbuf,rxbuf,DMA_BUFFER_SIZE);

   // Desabilita a contagem
   Dt_DisableCount(DtChannel0);

   // Converte para string a diferença entre o tempo
   // inicial e o tempo final.
   sprintf(buffer,"\n%i",tempo_inicial - Dt_ReadCurCntVal
(DtChannel0) - tempo_de_medicao);

   //Encaminha o tempo decorrido via UART para o
computador
   uart_printf(buffer);
}
```

Fonte: Autoria própria.

Tabela 1 – Valores utilizados para os parâmetros dos algoritmos considerados.

NLMS	VSS-NLMS de Shin	VSS-NLMS de Benesty	VSS-NLMS de Zipf
$\mu = 0,05$	$C = 0,001$	$\lambda_1 = 0,97$	$\beta = 0,99$
$\varepsilon = 1,0$	$\beta = 0,9961$	$\lambda_2 = 0,9987$	$\varepsilon = 1,0$
	$\mu_{máx} = 1,5$	$\delta = 1,0$	
	$\varepsilon = 1,0$	$\varepsilon = 20\hat{\sigma}_x^2$	

Fonte: Autoria própria.

Figura 7 – Trecho de código da implementação do algoritmo NLMS.

```
//cálculo da energia de x[]
arm_power_f32(x,M,&energia);

//produto interno entre w[] e x[]
arm_dot_prod_f32(w,x,M,&d_chapeu);

//determinação do erro
e=d-d_chapeu;

//cálculo do fator de adaptação
arm_scale_f32(x,mu*e/(energia+ep),fator,M);

//atualiza os coeficientes do filtro
arm_add_f32(w,fator,w,M);
```

Fonte: Autoria própria.

Figura 8 – Trecho de código da implementação do algoritmo VSS-NLMS de Shin.

```
//cálculo da energia de x[]
arm_power_f32(x,M,&energia);

//produto interno entre w[] e x[]
arm_dot_prod_f32(w,x,M,&d_chapeu);

//determinação do erro
e=d-d_chapeu;

//cálculo do vetor p (considerando p = pA + pB)
arm_scale_f32(p,beta,pA,M);
arm_scale_f32(x,(1-beta)*e/energia,pB,M);
arm_add_f32(pA,pB,p,M);

//cálculo da energia do vetor p
arm_power_f32(p,M,&energia_p);

//cálculo do passo variável
mu=mu_max*energia_p/(energia_p+C);

//cálculo do fator de adaptação
arm_scale_f32(x,(mu*e)/(energia+ep),fator,M);

//atualiza os coeficientes do filtro
arm_add_f32(w,fator,w,M);
```

Fonte: Autoria própria.

Figura 9 – Trecho de código da implementação do algoritmo VSS-NLMS de Benesty.

```
//produto interno entre w[] e x[]
arm_dot_prod_f32(x,w,M,&d_chapeu);

//determinação do erro
e=d-d_chapeu;

//estimativa da variância do sinal de erro e de x[]
var_e=lambda1*(var_e)+(1-lambda1)*e*e;
var_x=lambda2*(var_x)+(1-lambda2)*x[0]*x[0];

//define o parâmetro delta
ep=20*var_x;

//correlação entre x[] e o erro (r_ex = A + B)
arm_scale_f32(r_ex,lambda2,A,M);
arm_scale_f32(x,(1-lambda2)*e,B,M);
arm_add_f32(A,B,r_ex,M);
arm_power_f32(r_ex,M,&energia_r_ex);

//variância do sinal de fala (ruído de medição)
var_v=var_e-(energia_r_ex/var_x);

//cálculo do desvio padrão (erro e ruído de medição)
arm_sqrt_f32(var_e,&desv_e);
arm_sqrt_f32(var_v,&desv_v);

//fator gama
gama=(1-desv_v/(delta+desv_e)));

//atualiza os coeficientes do filtro
if(desv_e>=desv_v){
  mu=gama;
  arm_scale_f32(x,mu/(ep+energia_x)*e,fator,M);
  arm_add_f32(w,fator,w,M);
  }
```

Fonte: Autoria própria.

Figura 10 – Trecho de código da implementação do algoritmo VSS-NLMS de Zipf.

```
//cálculo da energia de x[]
arm_power_f32(x,M,&energia);

//produto interno entre w[] e x[]
arm_dot_prod_f32(w,x,M,&d_chapeu);

//determinação do erro
e=d-d_chapeu;

//estimativa da correlação do sinal de erro
p=beta*p_anterior+(1-beta)*e*e_anterior;

//estimativa da variância do sinal de erro
q=beta*q_anterior+(1-beta)*(e*e);

//determinação de mu
mu=(p/q)*(p/q);

//cálculo do fator de adaptação
arm_scale_f32(x,(mu/(ep+energia)*e,fator,M);

//atualização dos coeficientes do filtro adaptativo
arm_add_f32(w,fator,w,M);
```

Fonte: Autoria própria.

4.2. Discussão e análise dos resultados

Tendo em vista os objetivos do presente trabalho, avaliações com respeito à qualidade e inteligibilidade dos sinais de fala processados foram conduzidas (conforme descrito nas Seções 3.3 e 3.4). Também, foi determinada a complexidade computacional dos algoritmos em termos do tempo de execução das principais funções utilizadas e do tempo requerido para processar um quadro de 128 amostras de sinal (tal como descrito na Seção 3.5). Todos esses resultados são apresentados e discutidos a seguir.

4.2.1. Resultados provenientes da avaliação de qualidade

A média dos resultados da metodologia PESQ foi contabilizada para cada valor de SNR e o valor obtido é ilustrado na Figura 11(a) para o cenário de ruído branco, na Figura 11(b) para o cenário de ruído de balbúrdia e na Figura 11(c) para o cenário de ruído de obras em construção. Em linhas gerais, observa-se em tais figuras que o algoritmo VSS-NLMS de Benesty apresentou o melhor resultado global para os três cenários de ruído, seguido pelos algoritmos NLMS, VSS-NLMS de Shin e de Zipf. Destaca-se também que, em certos intervalos de SNR, todos os algoritmos apresentaram resultados superiores de qualidade quando comparados com o sistema de ANC desligado; por exemplo, o algoritmo VSS-NLMS de Zipf entre -20 dB e 6 dB, o algoritmo VSS-NLMS de Shin entre -20 dB até 12 dB e os algoritmos NLMS e VSS-NLMS de Benesty na faixa de -12 dB até 16 dB (veja a Figura 11). Todavia, uma degradação de desempenho do sistema é observada frente a condições de ruído mais realistas [isto é, ruído de balbúrdia e de obras em construção, conforme mostrado na Figura 11(b) e 11(c)], quando comparado com aqueles resultados do cenário de ruído branco gaussiano [Figura 11(a)]. Apesar disso, é possível inferir que os algoritmos considerados propiciam uma importante melhoria de qualidade no sinal de fala processado se comparado ao do sistema de ANC desligado.

4.2.2. Resultados provenientes da avaliação de inteligibilidade

Os resultados experimentais obtidos para a STOI e a API *speech-to-text* da Google Inc. são apresentadas nas Figuras 12 e 13. Nessas avaliações de inteligibilidade, os algoritmos VSS-NLMS de Benesty e NLMS apresentaram os melhores resultados em todos os cenários de ruído. Ainda, para valores de SNR de -20 dB até 8 dB, todos os algoritmos (exceto aquele proposto por Zipf) apresentaram um aumento de inteligibilidade quando comparados

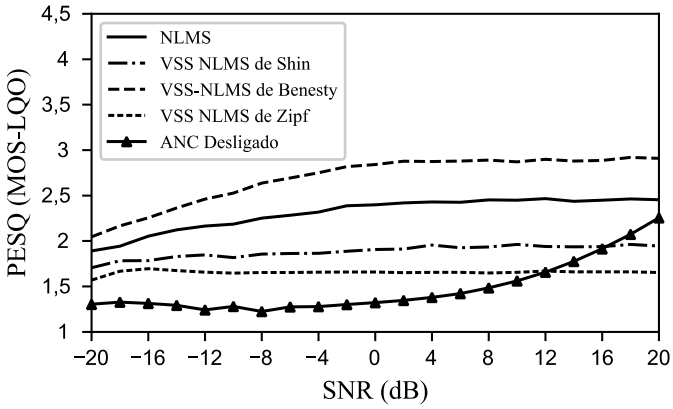
com o sistema de ANC desligado. O pior resultado observado foi do algoritmo VSS-NLMS de Zipf; apesar disso, esse algoritmo ainda é superior ao resultado obtido com o sistema de ANC desligado para valores de SNR abaixo de 2 dB. Por último, verifica-se novamente uma degradação de desempenho do sistema ANC quando os resultados dos cenários de ruído de balbúrdia e de obras em construção [Figuras 12(b), 12(c), 13(b) e 13(c)] são comparados com aqueles obtidos para ruído branco gaussiano [observado nas Figuras 12(a) e 13(a)].

4.2.3. Resultados provenientes da análise de complexidade computacional

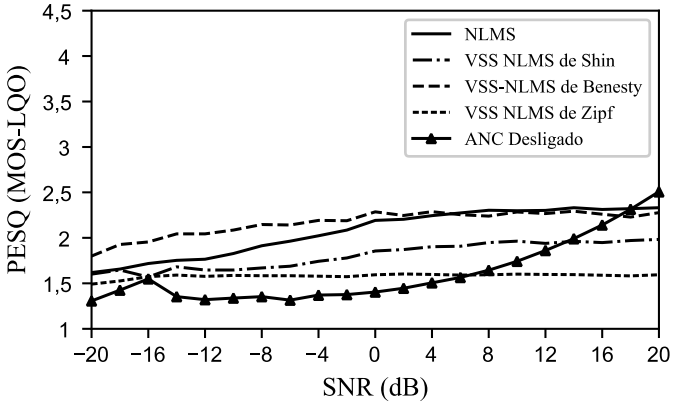
Nas Tabelas 2 e 3 são apresentados os resultados obtidos da análise de complexidade computacional. Na Tabela 2, é expresso o tempo de processamento (em ms) para cada algoritmo considerado, a ocupação (percentual) da capacidade de processamento do microcontrolador e o tempo de execução normalizado em função do algoritmo NLMS. Este último, representa quantas vezes cada implementação é mais complexa (em termos de tempo de execução) quando comparada com o algoritmo NLMS. Esses resultados mostram que os algoritmos NLMS e VSS-NLMS de Zipf possuem o menor tempo de execução (2,77 ms e 2,89 ms, respectivamente), ocupando assim apenas 18% da capacidade de processamento do microcontrolador. Em contrapartida, os algoritmos VSS-NLMS de Shin e de Benesty exibem os maiores tempos de execução (5,12 ms e 5,31 ms, respectivamente) e, consequentemente, maior ocupação do microcontrolador (em torno de 33%); portanto, são aproximadamente 2 vezes mais complexos computacionalmente do que o algoritmo NLMS. Já na Tabela 3, é apresentado o tempo de execução (em μs) para cada função utilizada, o número de vezes que tais funções são empregadas (em cada algoritmo) e o tempo total acumulado para processar um quadro de 128 amostras. Dentre elas, as funções `arm_add_f32()` e `arm_dot_prod_f32()` demandaram o maior tempo de execução, as quais desempenham a soma entre dois vetores (5,32 μs) e o produto interno de dois vetores (5,15 μs). Observa-se ainda que o tempo total acumulado das funções utilizadas (Tabela 3) e o tempo de processamento de cada algoritmo (Tabela 2) é muito próximo, ratificando os resultados obtidos. A pequena diferença entre esses valores pode ser atribuída à outras operações (tais como subtração, divisão e atribuição) necessárias para a operação dos algoritmos. Portanto, a partir da Tabela 3, é possível ainda obter uma estimativa do tempo de execução de um dado algoritmo arbitrário levando em conta o tempo de execução das principais funções utilizadas em sua correspondente implementação, sendo esse um importante critério de projeto.

Figura 11 – Curvas de qualidade obtidas por meio da metodologia PESQ. (a) Cenário de ruído branco gaussiano. (b) Cenário de ruído de balbúrdia. (c) Cenário de ruído de obras em construção.

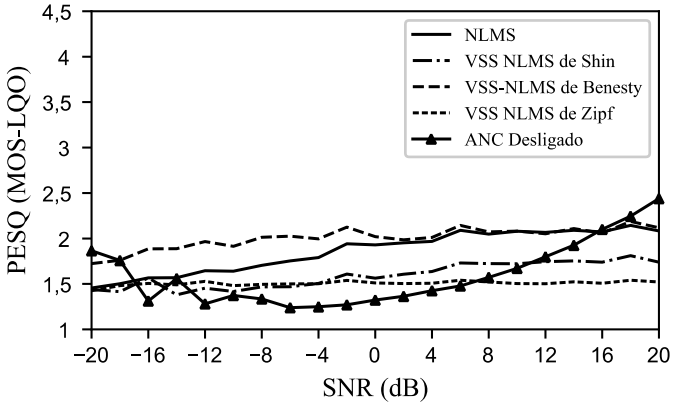
(a)



(b)



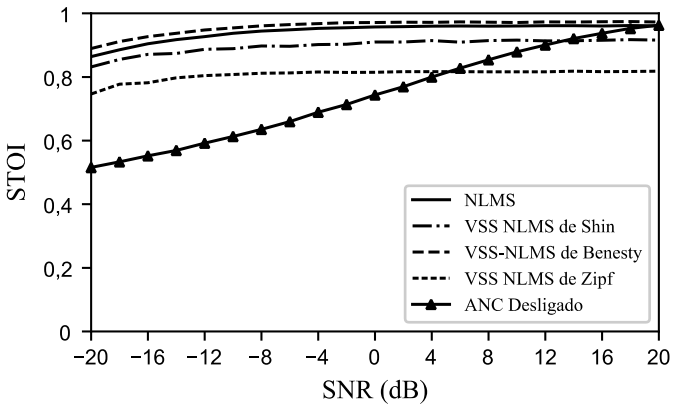
(c)



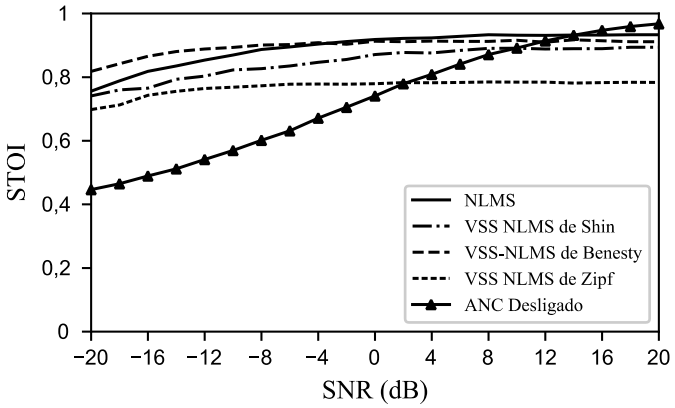
Fonte: Autoria própria.

Figura 12 – Curvas de inteligibilidade obtidas por meio da metodologia STOI. (a) Cenário de ruído branco gaussiano. (b) Cenário de ruído de balbúrdia. (c) Cenário de ruído de obras em construção.

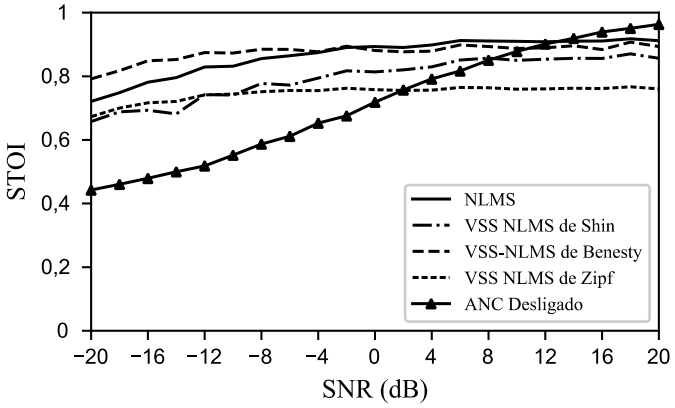
(a)



(b)



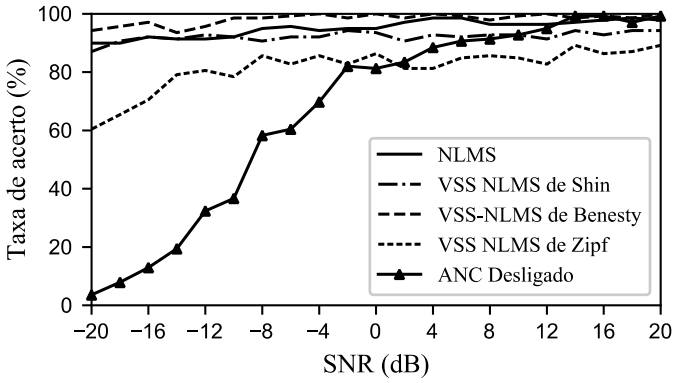
(c)



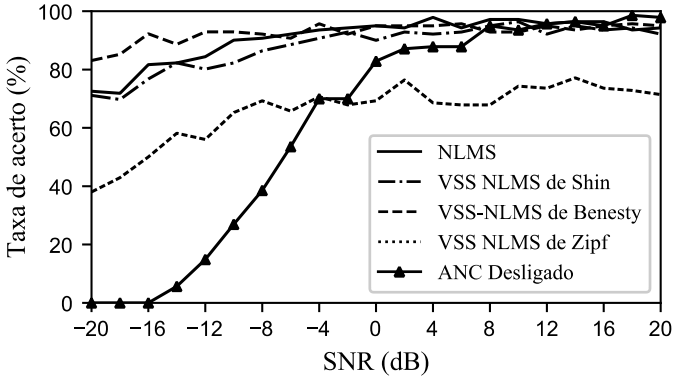
Fonte: Autoria própria.

Figura 13 – Curvas de inteligibilidade obtidas a partir da taxa de acerto no reconhecimento de palavras via API speech-to-text do Google Inc. (a) Cenário de ruído branco gaussiano. (b) Cenário de ruído de balbúrdia. (c) Cenário de ruído de obras em construção.

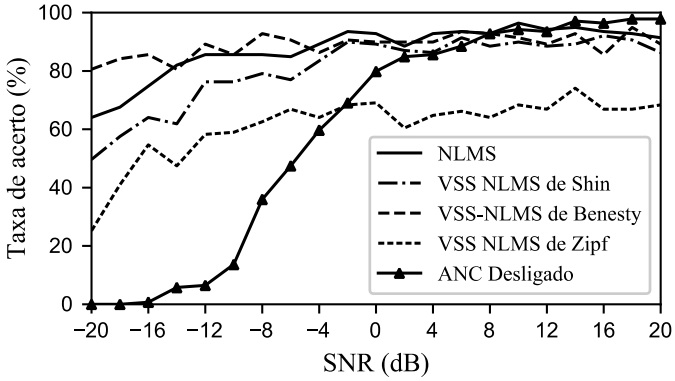
(a)



(b)



(c)



Fonte: Autoria própria.

Tabela 2 – Tempo de processamento de um quadro de 128 amostras.

Algoritmo	Tempo (ms)	% Ocupação	/ NLMS
NLMS	2,77	17,29	1
VSS-NLMS de Shin	5,12	32,00	1,85
VSS-NLMS de Zipf	2,89	18,06	1,04
VSS-NLMS de Benesty	5,31	33,14	1,92

Fonte: Autoria própria.

Tabela 3 – Tempo de execução das principais funções utilizadas por cada algoritmo e o tempo acumulado para processar um quadro de 128 amostras.

Função	Tempo (µs)	Algoritmo			
		NLMS	Shin	Benesty	Zipf
`arm_power_f32()`	3,54	×1	×2	×1	×1
`arm_dot_prod_f32()`	5,15	×1	×1	×1	×1
`arm_scale_f32()`	4,55	×1	×3	×3	×1
`arm_add_f32()`	5,32	×1	×2	×2	×1
`memcpy()`	2,08	×1	×1	×1	×1
Total acumulado (ms)		2,64	4,94	4,49	2,64

Fonte: Autoria própria.

5. CONCLUSÕES

Neste trabalho, a implementação de um sistema de ANC considerando o algoritmo NLMS e outros três algoritmos VSS-NLMS da literatura foi apresentada. Para tal, a placa de desenvolvimento Cortex-FM4 *Starter* da *Cypress Semiconductors* foi utilizada, a qual possui recursos importantes para a aquisição e síntese de sinais de áudio. Adicionalmente, a IDE Keil µVision® MDK foi considerada por oferecer bibliotecas e recursos importantes para o desenvolvimento deste trabalho. O desempenho do sistema de ANC implementado foi avaliado por meio de metodologias objetivas de qualidade e inteligibilidade, como também em termos de complexidade computacional. A partir dos resultados da avaliação de qualidade e inteligibilidade, concluiu-se que o melhor desempenho foi obtido pelo algoritmo NLMS e pelo algoritmo VSS-NLMS de Benesty. Além disso, a eficácia do sistema de ANC foi confirmada em certos intervalos de SNR para todos os algoritmos adaptativos considerados. Ainda nessas avaliações, tornou-se evidente a degradação de desempenho do sistema ANC quando submetido a cenários de ruído mais realistas, tais como para ruído de balbúrdia e de obras em construção. Observou-se também que a metodologia PESQ apresentou resultados de falso-positivo frente a valores de SNR muito baixos, onde o resultado esperado deveria ser inferior ao desempenho obtido para valores de SNR maiores. Com isso, verifica-se a importância de se realizar testes mais extensivos da metodologia PESQ, bem como comparar os seus resultados com os obtidos através de outras metodologias de avaliação de qualidade. Já a partir da análise de complexidade computacional, verificou-se que as implementações dos algoritmos adaptativos de Shin e Benesty foram as mais custosas computacionalmente, ocupando cerca de 33% da capacidade de processamento do microcontrolador para processar um quadro de 128 amostras. Em contrapartida, os algoritmos adaptativos NLMS e VSS-NLMS de Zipf apresentaram o menor custo computacional, ocupando apenas 18% da capacidade do microcontrolador. Portanto, caso a complexidade computacional seja um requisito importante de projeto, recomenda-se a utilização do algoritmo NLMS; caso contrário, o algoritmo VSS-NLMS de Benesty deve ser considerado, já que apresenta um desempenho (em termos gerais de melhoria de qualidade e inteligibilidade do sinal de fala) levemente superior ao obtido com o algoritmo NLMS. Com o intuito de dar continuidade ao presente trabalho, sugere-se agora a implementação de outros algoritmos da literatura, a construção um arranjo de microfones para integrar ao sistema, bem como o uso de outra metodologia [isto é, a *Perceptual objective listening quality analysis* (POLQA)] para determinar a qualidade dos sinais processados.

6. TRABALHOS PUBLICADOS

Os resultados obtidos, ao longo do desenvolvimento do presente trabalho de conclusão de curso (iniciado em 2017 através de um projeto de iniciação científica), deram origem a publicação e/ou submissão dos seguintes artigos científicos:

- PERTUM, R. R.; KUHN, E. V. Implementação de sistema de cancelamento adaptativo de ruído utilizando um algoritmo VSS-NLMS robusto. In: SIMPÓSIO BRASILEIRO DE TELECOMUNICAÇÕES E PROCESSAMENTO DE SINAIS (SBrT), 2018, Campina Grande. Disponível em: <https://biblioteca.sbrt.org.br/articles/1692>. Acesso em: 13 mai. 2020.
- PERTUM, R. R.; KUHN, E. V. Implementação e avaliação de desempenho de algoritmos para cancelamento adaptativo de ruído. In: SIMPÓSIO BRASILEIRO DE TELECOMUNICAÇÕES E PROCESSAMENTO DE SINAIS (SBrT), 2019, Petrópolis. Disponível em: <https://biblioteca.sbrt.org.br/articles/1994>. Acesso em: 13 mai. 2020.
- PERTUM, R. R,; KUHN, E. V. Considerações sobre o desempenho e a complexidade computacional de algoritmos utilizados em sistemas de cancelamento adaptativo de ruído. In: SIMPÓSIO BRASILEIRO DE TELECOMUNICAÇÕES E PROCESSAMENTO DE SINAIS (SBrT), 2020, Florianópolis. (SUBMETIDO).

REFERÊNCIAS

ARM. **ARM University**. Disponível em: <https://www.arm.com/resources/education/education-kits/digital-signal-processing>. Acesso em: 6 jun. 2020.

ARM KEIL. **µVision® IDE**. Disponível em: <http://www2.keil.com/mdk5/uvision/>. Acesso em: 03 jun. 2020.

BENESTY, J. et al. A Nonparametric VSS NLMS Algorithm. **IEEE Signal Processing Letters**, v. 13, n. 10, p. 581–584, out. 2006.

CIOCHINĂ, S.; PALEOLOGU, C.; BENESTY, J. An optimized NLMS algorithm for system identification. **Signal Processing**, v. 118, p. 115–121, jan. 2016.

CYPRESS SEMICONDUCTORS. **ARM Cortex-FM4 Starter kit (S6E2CC-ETH)**. Disponível em: <https://www.cypress.com/documentation/development-kitsboards/sk-fm4-176l-s6e2cc-fm4-family-quick-start-guide>. Acesso em: 03 jun. 2020.

CYPRESS SEMICONDUCTORS. **Peripheral Driver Library**. Disponível em: <https://www.cypress.com/design-guides/peripheral-driver-library-pdl-psoc-creator>. Acesso em: 06 jun. 2020.

GOOGLE INC. **Cloud Speech-to-Text**. [s.d.]. Disponível em: <https://cloud.google.com/speech-to-text/>. Acesso em: 6 jun. 2020.

HAYKIN, S. **Adaptive Filter Theory**. 5 ed. ed. Upper Saddle River: NJ: Prentice-Hall, 2014.

ITU-T. **P.800 : Methods for subjective determination of transmission quality**, 1996. Disponível em: <https://www.itu.int/rec/T-REC-P.800-199608-I/en>. Acesso em: 6 jun. 2020.

ITU-T. **ITU-T Test Signals for Telecommunication Systems**, 2000. Disponível em: <https://www.itu.int/net/itu-t/sigdb/menu.aspx>. Acesso em: 6 jun. 2020.

ITU-T. **Recommendation P.862 - Perceptual evaluation of speech quality (PESQ)**, 2005. Disponível em: <https://www.itu.int/rec/T-REC-P.862-200511-I!Amd2/en>. Acesso em: 6 jun. 2020.

LOIZOU, P. C. **Speech Enhancement: Theory and Practice**. 2nd. ed. Boca Raton: CRC Press, 2017.

IQBAL, M. A.; GRANT, S. L. Novel variable step size nlms algorithms for echo cancellation. In: IEEE INTERNATIONA CONFERENCE ON ACOUSTICS, SPEECHS AND SIGNAL PROCESSING, 2008. Disponível em: <http://ieeexplore.ieee.org/document/4517591/>. Acesso em: 3 jun. 2020.

NEVES, F. Ping pong buffer para sistemas embarcados. **Embarcados**, 2016. Disponível em: <https://www.embarcados.com.br/ping-pong-buffer/>. Acesso em: 6 jun. 2020.

PANAHI, I.; KEHTARNAVAZ, N.; THIBODEAU, L. Smartphone-based noise adaptive speech enhancement for hearing aid applications. **Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBS)**, 2016.

PARIENTE, M. **Python implementation of the Short Term Objective Intelligibility measure (pystoi)**. Disponível em: <https://github.com/mpariente/pystoi>. Acesso em: 3 jun. 2020.

PERTUM, R. R.; KUHN, E. V.; **Scripts e resultados [Online]**. Disponível em: <http://lapse.td.utfpr.edu.br/downloads/artigo_sbrt2020.zip>. Acesso em: 3 jun. 2020.

SAYED, A. H. **Adaptive Filters**. Hoboken, NJ, USA: John Wiley & Sons, Inc., 2008. v. 5

SHIN, H.-C.; SAYED, A. H.; SONG, W.-J. Variable Step-Size NLMS and Affine Projection Algorithms. **IEEE Signal Processing Letters**, v. 11, n. 2, p. 132–135, 2004.

SUGIYAMA, A. K.; MIYAHARA, R.; OOSUGI, K. A Noise Robust Hearable Device with an Adaptive Noise Canceller and Its DSP Implementation. 2019 IEEE INTERNATIONAL CONFERENCE ON CONSUMER ELECTRONICS (ICCE), mai.2019.

TAAL, C. H. et al. An Algorithm for Intelligibility Prediction of Time–Frequency Weighted Noisy Speech. **IEEE Transactions on Audio, Speech, and Language Processing**, v. 19, n. 7, p. 2125–2136, set. 2011.

VANDEN BERGHE, J.; WOUTERS, J. Hearing aid with adaptive noise canceller. **The Journal of the Acoustical Society of America**, v. 118, n. 4, p. 2109, 1998.

WIDROW, B.; STEARNS, S. D. **Adaptive Signal Processing**. [s.l.] Prentice-Hall Signal Processing Series, 1985.

YELDERMAN, M. et al. ECG Enhancement by Adaptive Cancellation of Electrosurgical Interference. **IEEE Transactions on Biomedical Engineering**, v. BME-30, n. 7, p. 392–398, jul. 1983.

ZIPF, J. G. F.; TOBIAS, O. J.; SEARA, R. Non-parametric VSS-NLMS algorithm with control parameter based on the error correlation. In: IEEE INTERNATIONAL TELECOMMUNICATIONS SYMPOSIUM (ITS), p. 1–5, set. 2010.