# Estudo da técnica de identificação de sistemas implementada em microcontroladores Arduino Due e Teensy 3.6

**Zorzo, A.*; Fonseca, W. D'A.***

* Engenharia Acústica, Universidade Federal de Santa Maria, Santa Maria, RS, {artur.zorzo, will.fonseca}@eac.ufsm.br

**Resumo**

A técnica de identificações de sistema é uma ferramenta poderosa na caracterização de sistemas dinâmicos. Uma das aplicações mais significativas desse algoritmo é em sistemas de controle ativo de ruído. Este trabalho tem por objetivo expor as teorias, a implementação e os resultados experimentais dessa técnica que faz o uso de filtros adaptativos e do algoritmo LMS. Dois microcontroladores, o Arduino Due e o Teensy 3.6, foram utilizados na implementação dos algoritmos e as velocidades de processamento e precisão dos resultados foram comparados. Para a parte experimental, filtros RC foram utilizados e os resultados das estimativas de resposta em frequência foram comparados com simulações numéricas. Além disso, uma estimativa da planta de um sistema de controle ativo de ruído em fones de ouvido foi obtida e comparada com uma resposta da medição dos transdutores e simulação dos filtros analógicos. Os resultados mostraram que o Arduino Due não foi capaz de lidar com altas frequências de amostragem devido às limitações do processador. O Teensy, por outro lado, foi capaz de gerar resultados satisfatórios até para frequências altas. As evidências obtidas revelaram que a precisão das estimativas dependem fortemente do número de coeficientes do filtro adaptativo utilizado e do parâmetro de convergência do algoritmo.

**Palavras-chave:** Função resposta em frequência, AD/DA, Otimização, Filtros adaptativos.

## 1. INTRODUÇÃO

Dispositivos ou instrumentos que são usados comummente possuem respostas impulsivas (RI) que geralmente não variam com o tempo. Para esses sistemas estáticos, existem diversos métodos bem fundamentados para medir suas respostas impulsivas e respostas em frequência (RI e FRF). Contudo, alguns sistemas são dinâmicos, significando que suas RIs são inconstantes e variam com o tempo. Assim, as análises em tempo e frequência utilizadas em sistemas estáticos não são adequadas para essas situações.

Um dos meios mais utilizados para a estimativa dessas funções é o *método de identificação de sistemas*. Esse método utilizada filtros de resposta ao impulso finitas (FIR) e um algoritmo de mínimos quadrados (MMQ ou LMS) para reduzir o erro de uma *função de custo*. Sendo que essa função é a diferença entre um sinal convoluído com o filtro FIR e a resposta do mesmo sinal aplicado ao sistema de interesse.

Existem diversas aplicações para essa técnica. No campo do controle ativo de ruído, ela é utilizada para estimar a RI da planta composta pelas respostas do alto-falante, conversores AD/DA, filtro *anti-aliasing*[1], microfone e do espaço de ar entre o alto-falante e o microfone. Esse filtro é, posteriormente, utilizado para corrigir um sinal de referência e melhorar a eficiência do sistema.

O tamanho do filtro e sua respectiva frequência de amostragem obtida dependem fortemente da capacidade de processamento do *hardware*. Filtragem adaptativa em tempo real usualmente exige *processadores digitais de sinais* (DSPs). Porém, como a tecnologia de processadores cresce consideravelmente todos os anos, a linha divisória entre DSPs e microcontroladores se tornou tênue. Uma grande parte dos microcontroladores atuais possuem capacidade de processamento e conversores suficientes para processamento digital de sinais.

[1] Evita o efeito do dobramento espectral.

 DOI: 10.55753/aev.v32e49.91

O principal objetivo do presente trabalho é analisar a performance de dois microcontroladores e o comportamento do algoritmo implementado para diferentes bandas de frequência e para diferentes parâmetros do algoritmo, como tamanho de passo e tamanho do filtro. Para isso, o algoritmo foi utilizado para estimar as respostas de alguns sistemas estáticos e os resultados foram comparados com as respostas simuladas ou medidas.

## 2. Identificação de sistemas

De acordo com Morgan e Kuo, a ideia básica por trás da técnica de identificação de sistemas é a construção de um modelo baseado na medição de um sinal modificado pelo sistema. O diagrama fundamental da técnica pode ser visto na Figura 1

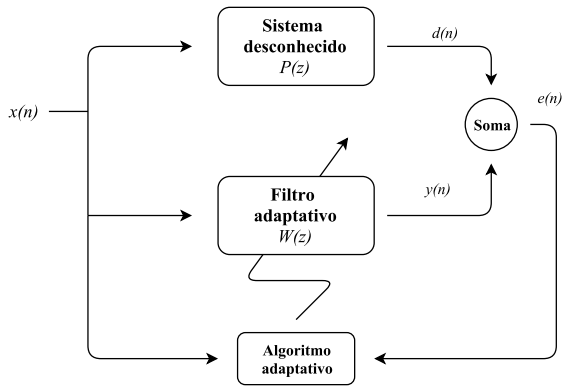


**Figura 1:** Diagrama da técnica de identificação de sistemas.

Um sinal de entrada $x(n)$, normalmente de banda larga como o ruído branco por exemplo, é gerado pelo processador e funciona como entrada tanto para o sistema em estudo quanto para o filtro adaptativo.

A saída do sistema desejado $P(z)$, expressa no diagrama por $d(n)$, e a saída do filtro $W(z)$, expressa por $y(n)$, são subtraídas de forma a gerar um sinal de erro $e(n)$.

O sinal de erro, assim como o sinal de entrada, são utilizados pelo algoritmo de minimização para ajustar os coeficientes do filtro de forma a minimizar a diferença das saídas.

Quando o erro atinge seu mínimo, a resposta impulsiva do filtro adaptativo está emulando a resposta de $P(z)$ da melhor forma possível. A medida que o sistema de interesse se modifica, o sinal de erro aumenta e o algoritmo volta a modificar os coeficientes do filtro de forma a modelar a nova RI.

Um dos algoritmos mais utilizados para o controle ativo de ruído, o algoritmo FXLMS realimentado, faz o uso dessa técnica para estimar a resposta de um planta composta por diversos subsistemas presentes no caminho do sinal. Esses subsistemas podem ser observados na Figura 2. Após a obtenção da estimativa, o filtro obtido é utilizado para filtrar um sinal de referência no algoritmo de controle principal.

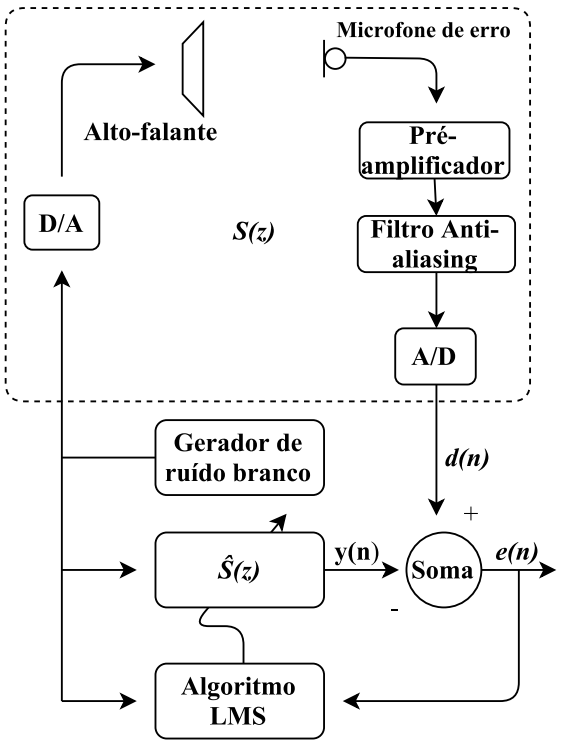


**Figura 2:** Utilização da técnica para a medição do caminho secundário em um sistema de controle ativo de ruído.

## 3. O algoritmo LMS e sua implementação

As equações relativas ao filtro e ao algoritmo adaptativo serão mostradas nesta secção (mais informações e respeito do algoritmo LMS podem ser encontradas em (CLARKSON, 1993)).

Aqui, os vetores e matrizes serão indicadas por letras maiúsculas e as quantidades escalares por letras minúsculas. O filtro FIR é definido como um vetor $W(n)$ com $L$ coeficientes e a memória de valores de entrada é definida pelo vetor $X(n)$, com o mesmo número de termos, no qual $x(n)$ representa o valor atual da entrada, $x(n-1)$ o valor imediatamente anterior e assim por diante. Os vetores citados podem ser definidos por:

$$W(n) = [w_0(n)\ w_1(n)\ ...\ w_{L-1}(n)] \quad (1)$$

e

$$X(n) = [x(n)\ x(n-1)\ ...\ x(n-L+1)]\ . \quad (2)$$

Para cada instante de tempo $n$, o erro pode ser computado pela subtração dos *samples* (amostras) relacionados à saída do filtro e à saída do sistema de interesse, resultando em

$$e(n) = d(n) - y(n)\ . \quad (3)$$

A saída do filtro pode ser computada pela convolução em tempo real entre a resposta impulsiva do filtro e o vetor de entrada (Equação 4). Essa convolução também pode ser expressa pelo produto vetorial entre a transposta de $W(n)$ e o vetor de entrada. Logo, esse produto pode ser denotado por:

$$y(n) = \sum_{i=0}^{L-1} w_i(n)x(n-i) = W^T(n)X(n)\ . \quad (4)$$

A Equação 5 é responsável pela atualização dos coeficientes do filtro adaptativo de modo a minimizar o quadrado do erro. O tamanho de passo, representado por $\mu$, coordena a taxa na qual o algoritmo converge (KUO; MORGAN, 2009), assim,

$$W(n+1) = W(n) + \mu X(n)\, e(n)\ . \quad (5)$$

Na prática, o algoritmo não é capaz de atingir exatamente a solução ideal, porém, o resultado obtido se aproxima do ideal. A medida do quão próximo a solução obtida chega da solução ideal é chamada de desajuste. Se o tamanho de passo é pequeno, o algoritmo toma mais tempo para convergir, porém, o resultado final se aproxima mais do ideal, tornando o desajuste pequeno. Em contrapartida, se $\mu$ é grande, o contrário acontece, o algoritmo converge rápido mas a solução final se distancia do valor desejado (Apolinário Jr, 2009).

As equações mostradas acima são implementadas a cada *sample*. Desta forma, a cada leitura do conversor, o valor de saída deve ser calculado antes da próxima leitura. Assim, o período de amostragem deve ser maior que o tempo que o processador leva para computar as equações.

Neste trabalho, a frequência de amostragem não foi fixada por meio de interrupções. O código foi programado de forma que cada leitura do AD é obtida assim que o valor de saída é convertido no DA, obtendo assim, a maior frequência de amostragem possível para a configuração e para o processador. Por esse motivo, o tamanho do filtro utilizado está diretamente relacionado com a máxima frequência de amostragem obtida. Existe então um compromisso entre a frequência de amostragem e o tamanho do filtro FIR. Se o filtro for muito pequeno, o sistema de estudo não será modelado de forma aceitável, especialmente se o sistema for complexo. Em contrapartida, se o filtro for muito grande, a frequência de amostragem e, por consequência, a máxima frequência de análise, será reduzida.

## 4. Descrição do *hardware*

Dois microcontroladores foram utilizados neste estudo, o Arduino Due (Arduino.cc, 2017) e o Teensy 3.6 (PJRC, 2017). As duas placas possuem processadores ARM cortex de 32 bits (mais informações podem ser encontradas em (Arm, 2017)). O Teensy possui um frequência de *clock* consideravelmente maior e possui mais memória RAM. Porém, a diferença mais relevante entre os dois microcontroladores é que o Arduino não possui unidade de ponto flutuante (UPF). Os parâmetros mais importantes

das duas placas podem ser vistos na Tabela 1.

Para este algoritmo escolhido, a UPF faz uma grande diferença no tempo de processamento visto que os coeficientes do filtro FIR tomam forma de ponto flutuante. A unidade é capaz de lidar com aritimética de pontos flutuantes em *hardware* (*hard float*), o que leva poucos ciclos de *clock*. A falta dessa unidade leva o processador a lidar com os números em *software* (*soft float*), processo que leva vários ciclos para ser completado.

As duas placas funcionam com uma tensão de operação de 3,3 V e possuem conversores AD/DA de 12 bits. O conversor DA do Arduino não é capaz de gerar valores analógicos em toda a faixa de tensão (0,0 a 3,3 V), sua saída é limitada de 0,55 a 2,75 V. O Teensy não possui essa limitação e os valores de saída podem assumir quaisquer valores entre 0,0 e 3,3 V.

**Tabela 1:** Tabela de comparações entre o Teensy 3.6 e o Arduino Due.

| | **Arduino Due** | **Teensy 3.6** |
|---|---|---|
| **Frequência de clock** | 84MHz | 180MHz |
| **RAM** | 96KB | 256KB |
| **FPU** | Não | Sim |
| **Processador** | ARM cortex-M3 | ARM cortex-M4 |
| **Profundidade** | 32 bits | 32 bits |

## 5. Descrição do experimento

Um estudo inicial, para testar o funcionamento do algoritmo foi conduzido pela conexão da saída dos microcontroladores diretamente na entrada (vide Figura 3). Teoricamente, como não há alterações relevantes no sinal de medição (ruído branco), o filtro deve convergir para um impulso unitário ideal sem atraso.

Subsequentemente, um filtro RC passa-baixa e um passa-banda (Figuras 5 e 6) foram modelados. Um material detalhado sobre filtros analógicos pode ser visto em *Analog Devices Handbook* (Analog Devices; Zumbahlen, H. (Ed.), 2008). Os filtros foram montados em uma *protoboard* e suas respectivas entradas e saídas foram conectadas diretamente nas entradas e saídas dos microcontroladores, como pode ser visto na Figura 4.

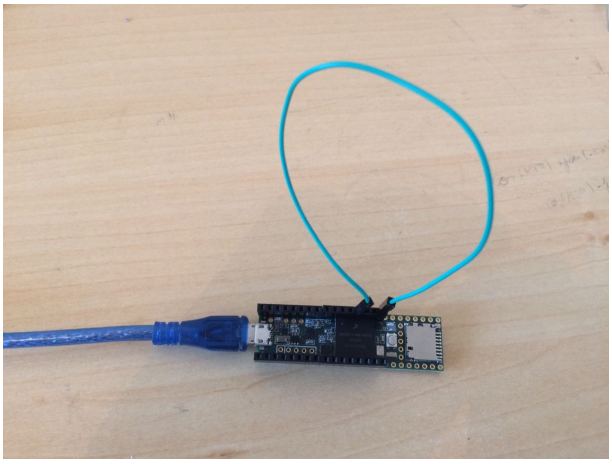

**Figura 3:** Teensy 3.6 na configuração de ligação direta entre a saída e a entrada.

Para os experimentos com os filtros analógicos, um filtro *anti-aliasing* não foi projetado e utilizado devido à variação brusca de frequências de amostragem entre os experimentos. Assim, erros devido a *aliasing* podem ter ocorrido nas estimativas (TAN; JIANG, 2013).

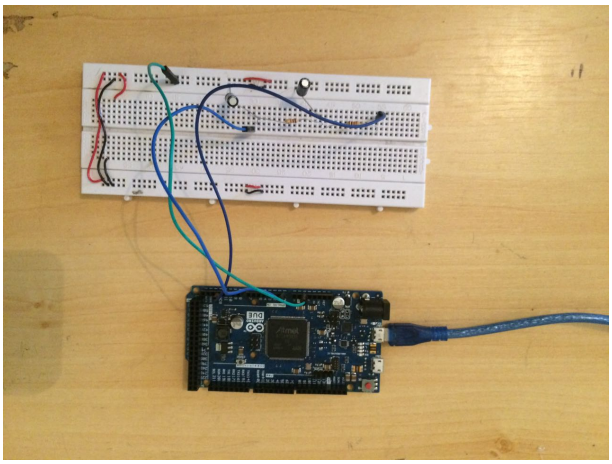

**Figura 4:** Arduino Due conectado ao filtro passa-baixa.

As estimativas foram feitas com filtros de 10, 25, 50, 75, 100, 125 e 150 coeficientes e as variações nas respostas em frequência foram observadas. Em uma etapa anterior, as frequências

de amostragem para cada número de coeficientes foram calculadas por meio das funções de temporizador dos próprios microcontroladores.

Para cada filtro, o algoritmo foi inicializado, e após o tempo necessário para garantir a convergência (com um tamanho de passo igual a $2,5 \cdot 10^{-11}$), a resposta impulsiva do filtro foi coletada e analisada. Em seguida, a resposta em frequência foi obtida a partir da resposta impulsiva e comparada com a resposta em frequência simulada dos filtros analógicos.

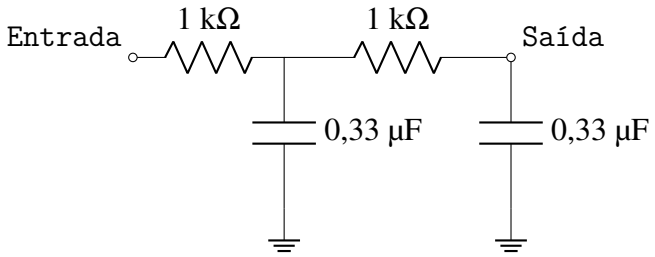


**Figura 5:** Filtro RC passa-baixa.

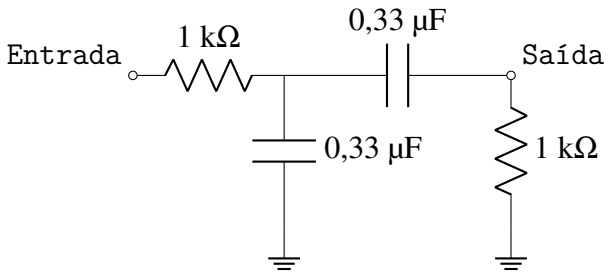


**Figura 6:** Filtro RC passa-banda.

O filtro passa-baixa foi modelado com um filtro de 100 coeficientes e diferentes tamanhos de passo foram computados com o objetivo de comparar os erros devido ao desajuste. Para este experimento, o algoritmo foi finalizado a um valor fixo de 5000 iterações, cerca de 2 segundos de tempo de convergência.

Por fim, para a obtenção de um resultado inicial qualitativo de um sistema de maior complexidade, um sistema de controle ativo de ruído em fones de ouvido foi estimado utilizando a plataforma Teensy e comparado com a medição do fone de ouvido convoluída com a resposta da simulação do filtro *anti-aliasing* utilizado. A estimativa seguiu o mesmo molde apresentado na Figura 2. A partir da análise da resposta em frequência do microfone e do pré-amplificador utilizados no projeto do sistema de controle, foi constatado que esses sistemas possuem uma resposta em frequência relativamente plana na faixa de frequências analisada. Portanto, os sistemas que alteram o sinal de forma mais significativa são o alto-falante do fone de ouvido e o filtro *anti-aliasing* utilizado. Desta forma, uma medição do fone de ouvido foi feita utilizando um simulador de cabeça e torso (veja Figura 7) e a resposta obtida foi multiplicada pela resposta em frequência da simulação do filtro.

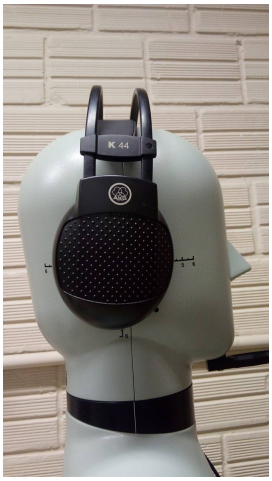

**Figura 7:** Medição do fone de ouvido AKG K44 em simulador de cabeça e torso.

Posteriormente, o sistema de controle foi montado no simulador de cabeça e torso e a estimativa por meio do método de identificação de sistemas foi obtida para filtros de 25, 50, 100 e 150 coeficientes, utilizando o mesmo tamanho de passo que nos experimentos anteriores.

As respostas foram então comparadas na faixa de frequências relativas às frequências de amostragem obtidas.

## 6. Resultados e discussões

Para cada tamanho de filtro, os processadores foram capazes de computar uma frequência de amostragem máxima. Uma comparação entre essas frequências para os dois processadores e

os diferentes filtros pode ser vista na Tabela 2. O Teensy foi capaz de calcular as equações de forma rápida o suficiente para gerar frequências de amostragem relativamente altas. O Arduino, por outro lado, não foi capaz de atingir frequências altas. Por esse fato, não foram testados filtros com mais de 100 coeficientes

**Tabela 2:** Frequência de amostragem obtida para cada tamanho de filtro.

| Tamanho do filtro (L) | Arduino Due | Teensy 3.6 |
|---|---|---|
| **10** | 16000 Hz | 66600 Hz |
| **25** | 6500 Hz | 52600 Hz |
| **50** | 3500 Hz | 38400 Hz |
| **75** | 2500 Hz | 30300 Hz |
| **100** | 1700 Hz | 24350 Hz |
| **125** | - | 21250 Hz |
| **150** | - | 18500 Hz |

Uma interpolação cúbica foi feita com os valores da Tabela 2 de forma a obter uma aproximação das frequências de amostragem. Os resultados das interpolações podem ser vistos nas Figuras 8 e 9.

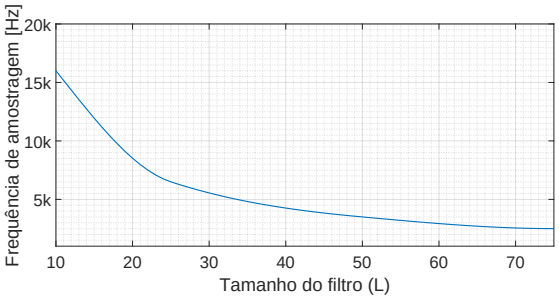


**Figura 8:** Interpolação cúbica entre as frequências de amostragem obtidas para cada tamanho de filtro - Arduino Due.

Os resultados obtidos pela conexão da saída diretamente na entrada dos microcontroladores podem ser observados na Figura 10. Como esperado, os resultados convergiram para um impulso ideal aproximado. A menor amplitude do pico principal da RI obtida pelo Arduino pode ser explicada pela limitação de tensão no conversor DA (que age como um atenuador), visto que o valor obtido de $0,66$ no primeiro *sample* é exatamente o valor obtido pela divisão entre faixas de tensão de entrada e saída:

$$\frac{2,75-0,55}{3,3-0,0} = 0,666. \qquad (6)$$

Além disso, os coeficientes subsequentes não convergiram a zero, como era esperado. Por outro lado, os resultados obtidos pela plataforma Teensy (em azul) se mostraram muito próximos do impulso ideal.

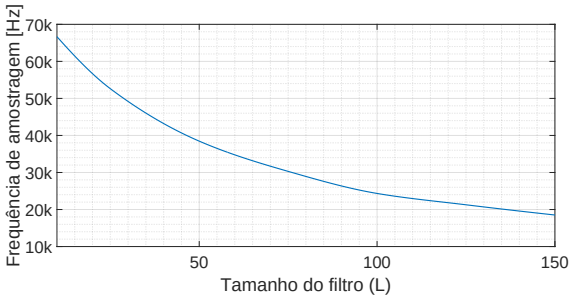


**Figura 9:** Interpolação cúbica entre as frequências de amostragem obtidas para cada tamanho de filtro - Teensy 3.6.

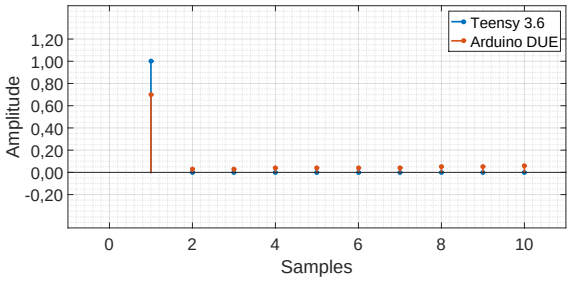


**Figura 10:** Respostas impulsivas para a configuração de ligação direta, Arduino em vermelho e Teensy em azul.

As Figuras 11 e 12 mostram os resultados obtidos pelas estimativas do filtro passa-baixa obtidas pelo Arduino. Cada curva foi plotada até a metade de sua respectiva frequência de amostragem. Pode-se perceber que para o filtro de 10 coeficientes, embora o resultado se aproxima de um filtro passa-baixa, o resultado não convergiu de forma satisfatória para o filtro testado. Para filtros maiores, melhores resultados foram obtidos na faixa de frequências disponível. Contudo, as baixas frequências de amostragem obtidas não são adequadas para a modelagem de

sistemas que operam em frequências mais altas, como no caso dos filtros adotados para esse estudo.

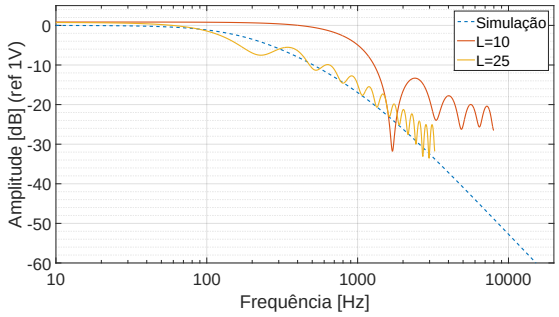


**Figura 11:** Estimativa da resposta em frequência do filtro passa-baixa (simulação e L= $\{10, 25\}$) - Arduino Due

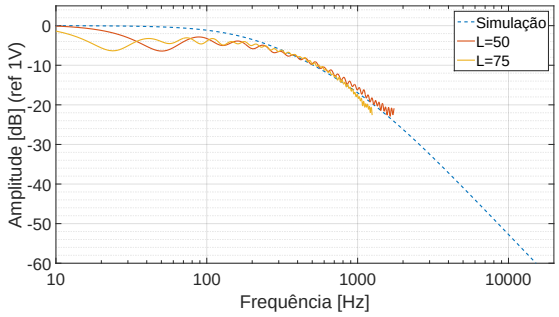


**Figura 12:** Estimativa da resposta em frequência do filtro passa-baixa (simulação e L= $\{50, 75\}$) - Arduino Due.

Com o Teensy, por outro lado, bons resultados foram obtidos para o filtro analógico. Esses resultados podem ser vistos nas Figuras 13 e 14. Para filtros com poucos coeficientes, as estimativas não tomaram forma do resultado esperado e várias oscilações apareceram em altas frequências. Contudo, o erro entre a curva esperada e a estimada reduz drásticamente com o aumento do tamanho do filtro. Para filtros com mais de 100 coeficientes as respostas foram satisfatórias.

As fases para os maiores filtros obtidas com o Teensy também foram comparadas com as simulações. Os resultados podem ser vistos na Figura 15. Os resultados também convergiram para valores satisfatórios embora alguns erros tenham ocorrido para frequências altas.

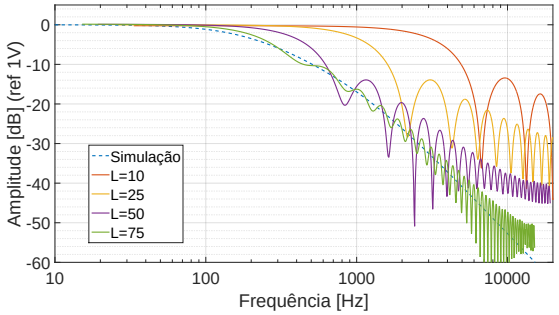


**Figura 13:** Estimativa da resposta em frequência do filtro passa-baixa (simulação e L= $\{10, 25, 50, 75\}$) - Teensy 3.6.

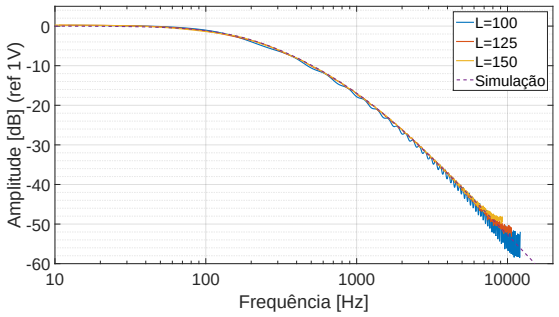


**Figura 14:** Estimativa da resposta em frequência do filtro passa-baixa (simulação e L= $\{100, 125, 150\}$) - Teensy 3.6.

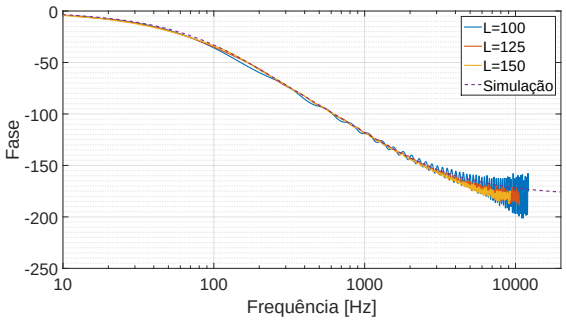


**Figura 15:** Estimativa da fase do filtro passa-baixa (simulação e L= $\{100, 125, 150\}$) - Teensy 3.6.

Para a configuração passa-banda, o Arduino não foi capaz de gerar bons resultados (vide Figuras 16 e 17). A porção passa-baixa do filtro possui uma frequência de corte muito alta para o processador. Assim, a melhor estimativa foi obtida pelo menor filtro já que ele foi o único que conseguiu modelar o decaimento em altas frequências.

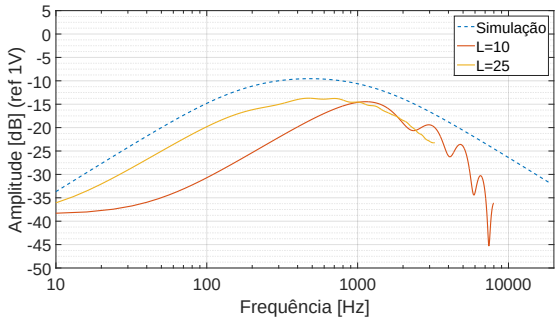


**Figura 16:** Estimativa da resposta em frequência do filtro passa-banda (simulação e L= $\{10, 25\}$) - Arduino Due.

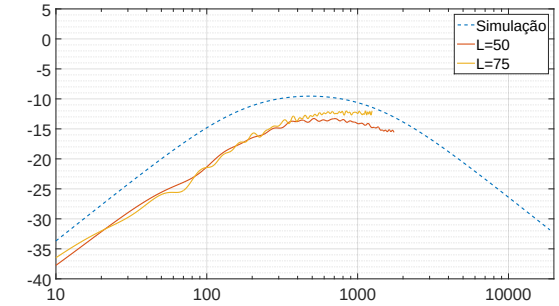


**Figura 17:** Estimativa da resposta em frequência do filtro passa-banda (simulação e L= $\{50, 75\}$) - Arduino Due.

Os resultados para a estimativa do filtro passa-banda obtidos pelo Teensy podem ser vistos nas Figuras 18 e 19. Para filtros com poucos coeficientes, a placa não foi capaz de modelar o filtro analógico. Para filtros com mais coeficientes, os resultados convergiram para os esperados na maior parte do espectro. É possível observar que nos extremos da faixa de frequências alguns erros ocorreram. Esse fator pode ser explicado pela baixa amplitude da resposta do filtro analógico nessas áreas, visto que o algoritmo funciona pela minimização do quadrado do erro, e valores de baixas amplitudes, como no caso dessas faixas do espectro, contribuem pouco para o valor global do erro.

A resposta de fase para os maiores filtros pode ser vista na Figura 20. Novamente pode-se observar que nos extremos do espectro a estimativa divergiu do resultado esperado.

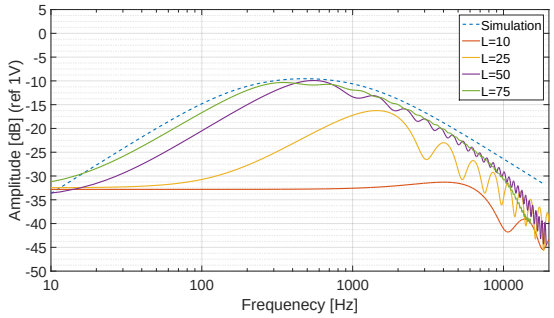


**Figura 18:** Estimativa da resposta em frequência do filtro passa-banda (simulação e L= $\{10, 25, 50, 75\}$) - Teensy 3.6.

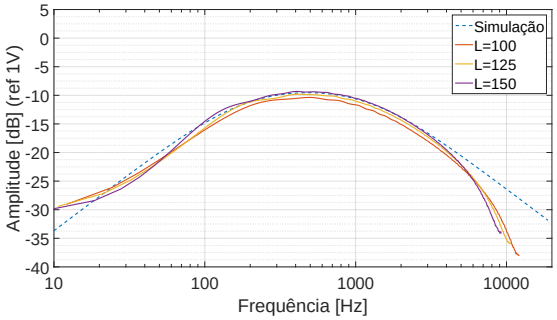


**Figura 19:** Estimativa da resposta em frequência do filtro passa-banda (simulação e L= $\{100, 125, 150\}$) - Teensy 3.6.

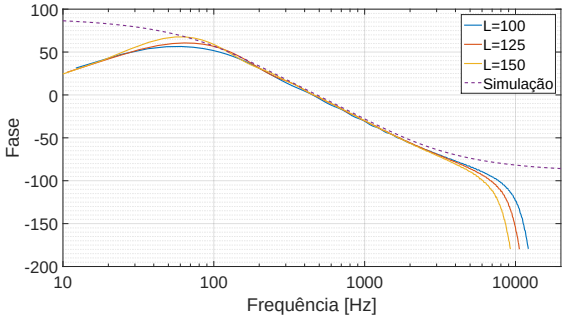


**Figura 20:** Estimativa da fase do filtro passa-banda (simulação e L= $\{100, 125, 150\}$) - Teensy 3.6.

O estudo da variação do tamanho de passo pode ser visto na Figura 21. Para valores menores que $3,57 \cdot 10^{-}9$ a estimativa não convergiu, foi possível observar que tamanhos de passo menores produzem menores desajustes e, por consequência, melhores resultados as custas de um tempo maior de convergência. Valores maiores que $2,5 \cdot 10^{-11}$ também não convergiram no tempo de convergência determinado.

Os resultados obtidos para a comparação entre a estimativa utilizando a técnica de identificação de sistemas e os resultados da medição dos fones de ouvido convoluídos pela resposta da simulação do filtro podem ser vistos nas Figuras 22 e 23.

Pode-se observar que os dois menores filtros não foram capazes de estimar de forma satisfatória os sistemas em baixas frequências. Porém, as curvas seguem o formato geral da resposta da medição até frequências mais altas.

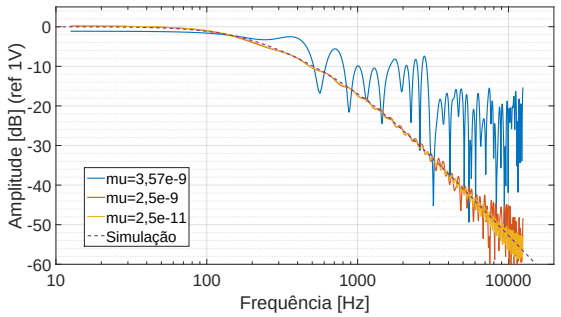


**Figura 21:** Variação do tamanho de passo para um filtro de 100 coeficientes - Teensy 3.6.

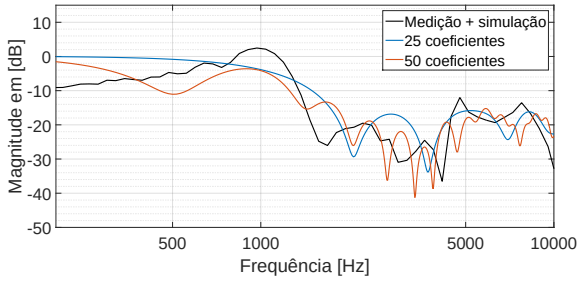


**Figura 22:** Comparação entre as resposta medida + simulada e estimada do caminho de sinal do sistema de controle ativo de ruído.

Para os maiores filtros, a máxima frequência de análise obtida foi baixa para sistemas de áudio. As curvas obtidas a partir das estimativas foram coerentes para baixas frequências mas não foram capazes de modelar o decaimento após a ressonância em 1000 Hz de forma satisfatória. Porém, com a exceção do pico na ressonância e do decaimento subsequente, as curvas se mostraram próximas da obtida pela medição.

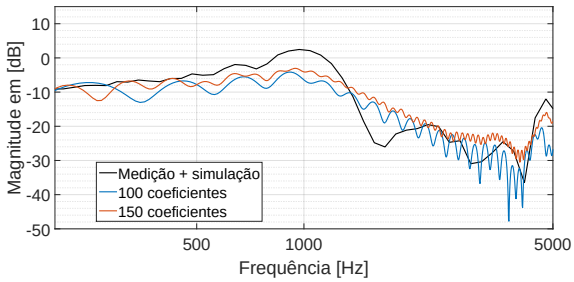


**Figura 23:** Comparação entre as resposta medida + simulada e a estimada do caminho de sinal do sistema de controle ativo de ruído.

## 7. Considerações finais

Ambos os processadores mostraram uma grande diferença no tempo de processamento. O Arduino não foi capaz de estimar os sistemas em uma faixa de frequência como a do áudio para a maior parte dos filtros digitais utilizados. Porém, o autor acredita que para sistemas vibracionais a técnica poderia ser utilizada sem grande problemas com esse processador. Com o Teensy, resultados satisfatórios foram obtidos mesmo para relativas altas frequências.

Para as estimativas dos filtros analógicos, alguns erros foram observados, especialmente nas porções do espectro de menor amplitude. Esse comportamento acontece divido a própria forma como o algoritmo é construído, visto que ele funciona pela diminuição do erro medido no domínio do tempo, e as frequências nas quais a resposta do sistema é muito baixa, contribuem pouco para o valor global do erro.

O valor do tamanho de passo é responsável tanto pelo desajuste quanto pelo tempo de convergência do algoritmo. Se o sistema de interesse varia rapidamente, um grande valor do tamanho do passo deve ser utilizado, para garantir uma convergência rápida. Porém, se esse valor for muito alto, o modelo obtido não será uma boa estimativa do sistema.

Os resultados da comparação para o sistema de controle ativo de ruído revelaram que para sistemas muito complexos, mesmo filtros com grandes coeficientes não são capazes de estimar

de forma adequada as respostas dos transdutores e filtros envolvidos. Além disso, pôde-se notar que filtros com um maior número de coeficientes geram melhores resultados em baixas frequência, o que é desejado para sistemas de controle ativo de ruído, visto que eles só são eficazes para grandes comprimentos de onda. Contudo, de acordo com Kuo e Morgan (KUO; MORGAN, 2009), para sistemas de controle a estimativa do caminho secundário não precisa ser igual a resposta real dos sistemas. A convergência do algoritmo não é afetada de forma muito significativa por diferenças de amplitude na estimativa.

## REFERÊNCIAS

Analog Devices; Zumbahlen, H. (Ed.). *Linear Circuit Design Handbook*. Newnes/Elsevier, 2008. doi: 10.1016/B978-0-7506-8703-4.X0001-6. ISBN 978-0750687034. Disponível em: <https://www.analog.com/en/education/education-library/linear-circuit-design-handbook.html>.

Apolinário Jr, J. A. (Ed.). *QRD-RLS Adaptive Filtering*. [S.l.]: Springer, 2009. doi: 10.1007/978-0-387-09734-3. ISBN 978-0387097336.

Arduino.cc. *What is Arduino?* 2017. <https://www.arduino.cc/en/Guide/Introduction>. Acessado em 12/12/2017.

Arm. *Processors Cortex-M Series*. 2017. <https://www.arm.com/products/processors/cortex-m>. Acessado em 12/12/2017.

CLARKSON, P. M. *Optimal and Adaptive Signal Processing: A Volume in the Electronic Engineering Systems Series: 3*. [S.l.]: CRC-Press, 1993. doi: 10.1201/9780203744925. ISBN 978-0367450076.

KUO, S. M.; MORGAN, D. R. *Active Noise Control Systems – Algorithms and DSP Implementation*. [S.l.: s.n.], 2009. ISBN 978-0471134244.

PJRC. *Teensy USB Development Board*. 2017. <https://www.pjrc.com/store/teensy36.html>. Acessado em 12/12/2017.

TAN, L.; JIANG, J. *Digital Signal Processing: Fundamentals and Applications*. 2. ed. [S.l.]: Academic Press, 2013. ISBN 978-0124158931.