Simplificações e adaptações para redução do custo computacional do pré-processamento de voz na Platraforma Arduino

Pedro Ítalo Ribeiro Albuquerque [1], Emerson Barbosa da Cunha [2], Daniella Dias Cavalcante da Silva [3], César Rocha Vasconcelos [4]

[1]pedroitaloifpb@gmail.com, [2]emersonbarbosa.ifpb@gmail.com, [3]daniella.silva@ifpb.edu.br, [4]cesarocha@ifpb.edu.br - Instituto Federal de Educação, Ciência e Tecnologia da Paraíba – Campus Campina.

RESUMO

Atualmente, existe um crescente interesse por aplicações em que a interação homem-máquina seja realizada via voz humana. No entanto, alguns equipamentos, como telefones celulares e eletrodomésticos, possuem limitações de armazenamento e processamento, dificultando a implementação de sistemas de reconhecimento automático da fala. Neste trabalho, foram implementadas simplificações matemáticas e estratégias de programação em duas etapas típicas de um sistema de reconhecimento da fala: a pré-ênfase e a divisão em quadros e janelamento. O objetivo dessa implementação foi analisar o impacto dessas adaptações no desempenho e, consequentemente, no custo computacional das referidas etapas. Ao final deste trabalho, o tempo de execução foi reduzido para 1/5 do tempo original da pré-ênfase e para 1/10 no caso da divisão em quadros e janelamento.

Palavras-chave: Processamento de sinais de voz. Pré-ênfase. Janelamento. Arduino. Custo computacional.

ABSTRACT

Currently, there is a growing interest in applications where human-computer interaction be performed by speech recognition. However, some devices such as smartphones and domestic appliances have limited resources for storage and processing. This limitation hinders the implementation of speech recognition systems in this type of device. In this work mathematical simplifications and programming strategies have been implemented in two typical stages of a speech recognition system: pre-emphasis, frame blocking and windowing. The goal was to analyze the impact of this implementation on the performance and computational cost of stages above. Given the adaptations made in this work, the runtime of pre-emphasis was reduced to 1/5 of the original and the frame blocking and windowing one to 1/10.

Keywords: ***Speech signal processing. Pre-emphasis. Windowing. Arduino. Computational cost.***

1 Introdução

Historicamente, o ser humano sempre buscou desenvolver métodos ou formas de comunicação variadas – obviamente, sempre de acordo com o nível de conhecimento de cada época. Uma dessas formas é a fala, que tem um importante papel na difusão e manipulação da informação, desde a simples decodificação de palavras por uma criança até mecanismos tecnológicos cada vez mais sofisticados.

Com o aumento do interesse por aplicações de *software* e equipamentos que "compreendam", reconheçam e simulem a voz humana, pesquisas têm sido realizadas na área de Processamento Digital de Sinais de Voz (PDSV) (KLEIJN; PALIWAL, 1995; LIMA et al., 2000; CIPRIANO, 2001; BENZEGHIBA; BOURLARD, 2003; CUNHA; VELHO, 2003; SILVA, 2006). Atualmente, equipamentos eletrônicos que permitem realizar a interação homem-máquina por meio do reconhecimento da fala são cada vez mais comuns, seja na área de segurança (como forma de controle de acesso), mobilidade (para indivíduos com limitações físicas), ou na busca por praticidade (no acesso a funções de dispositivos como *smartphones, tablets, mídia center* de veículos automotores, etc.). Como se pode ver, existe uma grande diversidade de aplicações na utilização desse tipo de tecnologia. No entanto, alguns equipamentos possuem um *hardware* simples e, consequentemente, limitações na sua capacidade de processamento e armazenamento. Diante disso, torna-se necessário o desenvolvimento de *software* adaptado para esse ambiente. Surge, então, o conceito de sistema embutido (ou embarcado), que é a combinação de *hardware* e *software* e, algumas vezes, peças mecânicas, desenvolvidos para realizar uma função específica (BARR, 1999; FRANCIA III, 2001; SILVA, 2006). Por meio da implementação em *hardware*, é possível alcançar maior eficiência e rapidez na execução de determinadas tarefas e, a partir do *software*, reduzir o tempo de desenvolvimento do sistema.

Em sistemas de reconhecimento de fala, o maior desafio é aliar o bom desempenho ao baixo consumo de energia, o que não é tão simples ou barato de se alcançar. Para permitir a implementação de sistemas de PDSV nesse contexto, alguns trabalhos sacrificam a eficiência do processo de reconhecimento em nome da redução da área e das exigências computacionais. No entanto, algumas técnicas de programação permitem a geração de códigos mais eficientes no que se refere à quantidade de operações a serem executadas pelo processador, o que, consequentemente, também proporciona menor consumo de energia. Um exemplo simples de estratégia é a substituição de operações de multiplicação ou divisão por deslocamento de bits. Sendo assim, a partir da otimização de algoritmos, da utilização de estruturas de *hardware* específicas e de um paralelismo de instruções eficiente, é possível aumentar a velocidade e reduzir os recursos necessários, sem diminuir a taxa de reconhecimento (SILVA, 2006).

Uma plataforma bastante utilizada para o desenvolvimento, de maneira rápida, de sistemas embarcados é o Arduino. O objetivo deste trabalho foi analisar o impacto no desempenho, para essa plataforma de *hardware*, ao se utilizar simplificações matemáticas e estratégias de programação em subetapas do pré-processamento de um sistema de reconhecimento de fala.

O restante deste documento está organizado da seguinte maneira: na Seção 2, é realizada uma breve apresentação da plataforma Arduino; na Seção 3, são apresentadas as principais etapas de um sistema de reconhecimento de fala; na Seção 4, é detalhada a etapa de pré-processamento, objeto de estudo deste trabalho; na Seção 5, são descritas as simplificações implementadas, como também as limitações encontradas durante o desenvolvimento; na Seção 6, são apresentados os resultados obtidos e sua discussão; por fim, na Seção 7, são feitas as considerações finais.

2 Arduino

A plataforma de *hardware* adotada neste trabalho foi o Arduino. Este consiste em uma placa fabricada como plataforma de prototipagem eletrônica e torna a robótica mais acessível a todos. O projeto italiano iniciado em 2005 tinha, primeiramente, cunho educacional e interagia com aplicações escolares. No entanto, atingiu proporções muito maiores ao longo dos anos, justamente por sua facilidade de uso. Com o Arduino não é necessário ter tanto conhecimento do *hardware*, como ocorre com outros microcontroladores, como o PIC (MCROBERTS, 2011).

É possível também aumentar a capacidade do *hardware* do Arduino, acoplando a ele os chamados *shields* (escudos). Estes consistem em placas que são acopladas à placa original, agregando-lhe funcionalidades. Há diversos tipos de *shields*, com as mais variadas funções – por exemplo, comunicação com

rede *Ethernet,* transmissão via *Bluetooth*, módulos Zigbee, entre outras.

No entanto, essa vantagem de trabalhar em alto nível pode se tornar uma grande desvantagem quando é necessário ter um controle maior no funcionamento do *hardware* – por exemplo, quando é preciso utilizar soluções de otimização, como o uso de registradores específicos e *pipeline*.

O modelo utilizado neste trabalho foi o Arduino UNO (Figura 1), que é equipado com o microcontrolador ATmega328 da ATMEL®. Ele tem um *clock* base de 16 MHz e vários pinos de entrada e saída, tanto digitais quanto analógicas. Também possui módulo PWM (*Pulse Width Modulation*) (ATMEL CORPORATION, 2014), que vem implementado nativamente e com funções específicas na própria IDE. Esse módulo permite realizar conversão A/D e D/A, facilitando a implementação de aplicações para processamento de áudio.

Figura 1 - Arduino UNO.

Fonte: MCROBERTS, 2011

3 Sistemas de Reconhecimento de Fala

Nesta seção, será realizada uma breve descrição das etapas típicas de um sistema de reconhecimento de fala (Figura 2), de modo que seja possível compreender o papel do pré-processamento nesse tipo de sistema.

Sistemas de reconhecimento de fala realizam uma tarefa de reconhecimento de padrões – nesse caso, padrões de fala – que sempre inclui duas fases: treinamento e reconhecimento. Com base nos dados de treinamento (palavras ou frases), são gerados os modelos de referência (modelos acústicos), aos quais são atribuídos rótulos que identificam cada padrão (palavra ou frase). Na fase de reconhecimento, a partir dos dados de teste (sinais de voz), são obtidos padrões de teste que, em seguida, são comparados com os modelos gerados durante o treinamento. Assim, utilizando-se uma regra de decisão, é identificado o modelo que mais se assemelha ao padrão de entrada desconhecido.

A etapa de pré-processamento é responsável pelo tratamento do sinal de voz com relação ao ambiente de gravação e ao canal de comunicação utilizado. O objetivo desse tratamento é reduzir efeitos indesejados incorporados ou presentes no sinal de voz, além de prepará-lo para as etapas seguintes do processo de reconhecimento. Dentre os vários aspectos que podem ser tratados nessa etapa, podem ser citados (RABINER; SCHAFER, 1978; FURUI, 1981; SHAUGHNESSY, 2000):

- Variações relacionadas ao estilo do falante;
- Ruído no ambiente ou no meio de comunicação;

Figura 2. Sistema de reconhecimento de padrões da fala.

Fonte: SILVA, 2006

- Variações no momento de gravação do sinal (por exemplo, distância entre o locutor e o microfone);
- Considerações quanto ao tipo de unidade da fala a ser processada nas etapas posteriores.

A etapa de extração de características é de extrema importância em sistemas de reconhecimento, uma vez que, nessa etapa, são obtidos elementos que possibilitam a geração de padrões. Esses elementos também podem ser parâmetros obtidos a partir de modelos de produção de voz. Assim, essa etapa também pode ser chamada de extração de parâmetros (SILVA, 2006).

Durante a fase de treinamento, com base nas características extraídas, são gerados padrões de referência, os quais serão comparados com o padrão de teste na fase de reconhecimento.

Vale salientar que as etapas de pré-processamento e extração de características das fases de treinamento e de reconhecimento devem ser equivalentes, para que seja possível a correta comparação entre o padrão testado e o(s) padrão(ões) já conhecido(s) pelo sistema (SILVA, 2006).

4 Pré-processamento Digital de Sinais de Voz

Os sinais de voz são compostos de uma sequência de sons que servem como uma representação simbólica da mensagem produzida pelo locutor para o ouvinte (RABINER; SCHAFER, 1978). O sinal de voz produzido pelo homem é naturalmente analógico; para que seja possível seu processamento por computadores digitais, é necessário que seja realizada sua conversão analógico-digital (A/D). Essa conversão inclui a amostragem e quantização do sinal.

A aquisição do sinal de voz é realizada por meio de um microfone, o qual converte as variações que a fala causa na pressão do ar em variações de tensão elétrica. Em seguida, é realizada a amostragem do sinal, seguindo o teorema de Nyquist. Esse teorema demonstra que um sinal contínuo pode ser completamente representado e recuperado com base em algumas de suas amostras igualmente espaçadas no tempo (LATHI, 1998).

O passo seguinte à amostragem é a quantização do sinal amostrado. Essa etapa consiste na discretização da intensidade (amplitude) do sinal, permitindo a sua representação por uma quantidade finita e pré-definida de bits, obtendo-se então um sinal digital. Quanto maior a quantidade de bits utilizada nesse processo, maior o número de valores permitidos e menor o de aproximações necessárias, o que torna o sinal quantizado mais fiel ao original.

Após a aquisição e digitalização do sinal de voz, é realizado o pré-processamento das amostras, a fim de prepará-las para a extração de suas características. Estas são utilizadas no algoritmo de reconhecimento de padrões da fala. O pré-processamento a ser implementado neste trabalho inclui as etapas de pré-ênfase e divisão em quadros e janelamento.

4.1 Pré-ênfase

A distorção provocada pelos lábios produz uma queda na envoltória espectral de, aproximadamente, 6 dB/oitava. Uma vez que o sinal de voz apresenta baixas amplitudes nas altas frequências, essa tendência espectral torna essas frequências especialmente vulneráveis ao ruído, comprometendo o processo de reconhecimento (PETRY; ZANUZ; BARONE, 2000).

Para solucionar esse problema, é aplicado um filtro, cuja resposta é de aproximadamente +6dB/oitava, que ocasiona um nivelamento no espectro (PETRY; ZANUZ; BARONE, 2000). A esse processo de tratamento do sinal de voz dá-se o nome de pré-ênfase.

A função de transferência da pré-ênfase consiste em um sistema de primeira ordem fixo, cuja função é apresentada na Equação 1:

$$N(z) = 1 - \alpha.z^{-1},\ 0 \leq \alpha \leq 1 \qquad (1)$$

Nesse caso, a saída da pré-ênfase $S_p(n)$ está relacionada à entrada $S(n)$ pela Equação 2, dada por (PETRY; ZANUZ; BARONE, 2000):

$$S_p(n) = S(n) - \alpha.S(n-1) \qquad (2)$$

Sendo:

- $S_p(n)$ - amostra pré-enfatizada;
- $S(n)$ - amostra original;
- α - fator de pré-ênfase, $0{,}9 \leq \alpha \leq 1$

Um valor típico usado é $\alpha = 0{,}95$, o que significa 20 dB de amplificação para as mais altas frequências.

4.2 Divisão em quadros e janelamento

Um sinal é dito estacionário quando suas características estatísticas não variam com o tempo (LATHI, 1998). A segmentação, ou divisão em quadros, consiste em particionar o sinal de voz em segmentos,

selecionados por janelas ou quadros (*frames*) de duração perfeitamente definida. O tamanho desses segmentos é escolhido dentro dos limites de estacionariedade do sinal, com duração média de 16 a 32 ms (RABINER; SCHAFER, 1978; SHAUGHNESSY, 2000). Os tipos de janelas mais comumente utilizados em sistemas de processamento de voz são (SHAUGHNESSY, 2000):

- Janela Retangular: o sinal é simplesmente particionado em blocos consecutivos de mesmo tamanho, segundo a Equação 3, na qual NA representa o tamanho da janela:

$$J(n) = \begin{cases} 1, & 0 \le n \le N_A - 1 \\ 0, & \textit{caso contrário} \end{cases} \quad (3)$$

- Janela de *Hamming*: proporciona a manutenção das características espectrais do centro do quadro e a eliminação das transições abruptas das extremidades, segundo a Equação 4, sendo NA o tamanho da janela:

$$J(n) = \begin{cases} 0{,}54 - 0{,}46\cos\left[\frac{2\pi n}{(N_A-1)}\right], 0 \le n \le N_A - 1 \\ 0, \quad \textit{caso contrário} \end{cases} \quad (4)$$

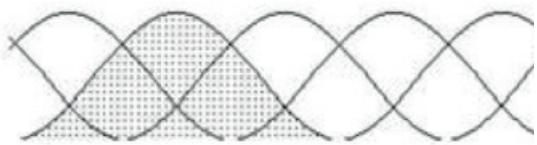
- Janela de *Hanning*: assemelha-se à janela de *Hamming*, porém gera um reforço menor nas amostras do centro e uma suavização maior nas amostras das extremidades, segundo a Equação 5:

$$J(n) = \begin{cases} 2\,\text{acos}\left[\frac{\pi n}{N_A}\right] + b, 0 \le n \le N_A - 1 \\ 0, \quad \textit{caso contrário} \end{cases} \quad (5)$$

sendo 2a + b = 1 (0 ≤ a ≤ 0,25 e 0,5 ≤ b ≤ 1)

As janelas de *Hamming* e *Hanning* apresentam, porém, uma característica nem sempre desejável, que corresponde à atribuição de um peso muito baixo às amostras das extremidades. Essas amostras podem representar eventos importantes de curta duração do sinal de voz, e multiplicá-las por um peso baixo representa pouca atenção no processamento subsequente, realizado no nível de blocos. Para assegurar que a tais eventos seja dado o peso necessário, blocos adjacentes são sobrepostos, de modo que um evento seja "coberto" por outros blocos. Muitos trabalhos utilizam uma sobreposição em torno de 50%. Outra justificativa para a sobreposição das janelas é que ela proporciona uma variação mais gradual das características entre janelas sucessivas. Um exemplo de sobreposição de janelas pode ser visto na Figura 3.

Figura 3. Sobreposição de janelas de *hamming*.



Fonte: Elaborado pelos autores

Para o contexto da produção da voz, as características da janela de *Hamming* se mostram mais eficientes, quando comparadas às janelas Retangular e de *Hanning* (FECHINE, 2000). Diante disso, essa foi a janela utilizada neste trabalho.

5 Simplificações e Implementação

Para analisar o impacto e, consequentemente, a viabilidade da implementação de técnicas de PDSV na plataforma Arduino, foram utilizadas neste trabalho as simplificações matemáticas e adaptações propostas por Silva (2006) e Cipriano (2001) referentes à etapa de pré-processamento. Inicialmente, foi analisada a subetapa de pré-ênfase e, em seguida, a subetapa de divisão em quadros e janelamento. A etapa de conversão A/D não foi avaliada neste trabalho. Por isso, foram utilizadas amostras de sinais de voz já digitalizadas e armazenadas em um cartão de memória, conforme será descrito na seção a seguir.

5.1 Pré-ênfase

Para a função de pré-ênfase, na Equação 2, o fator α foi substituído pelo valor 15/16, correspondente a 0,9375:

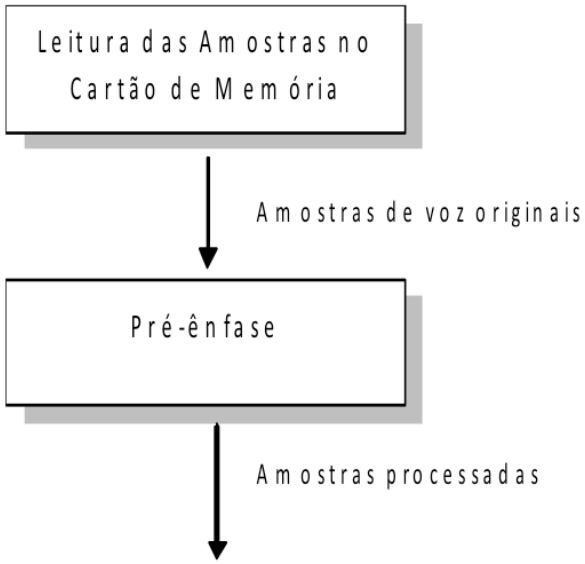
$$\begin{aligned} S_p(n) &= S(n) - \frac{15}{16}.S(n-1) \\ &= S(n) - S(n-1) + \frac{(S(n-1))}{16} \end{aligned} \quad (6)$$

A modificação apresentada na Equação 6 implica a substituição de uma multiplicação por um número 2*m*, sendo *m* inteiro. Essa divisão pode ser realizada

por meio de um simples deslocamento de *m* bits – nesse caso, *m* = 4. Durante a divisão por 16 da Equação 6, é realizado um deslocamento de quatro bits para a direita, repetindo quatro vezes o bit mais significativo à esquerda. Essa repetição é necessária para que seja mantido o sinal do valor correspondente da amostra de voz.

Na Figura 4, é apresentado o diagrama de blocos do sistema de pré-ênfase implementado.

Figura 4. Diagrama de blocos da pré-ênfase.



Fonte: Elaborada pelos autores

Além das versões tradicional e adaptada, a pré-ênfase também foi implementada com e sem módulos. Os módulos em questão consistem na separação da etapa de leitura das amostras de voz, armazenadas em um cartão SD, da função de pré-ênfase propriamente dita.

A versão sem módulo está totalmente implementada em um único arquivo fonte. Já a versão com módulo possui uma biblioteca para a leitura do cartão SD e outra para a pré-ênfase, sendo essas duas referenciadas em um programa principal. A vantagem da modularização é que, no futuro, caso se deseje captar o sinal de voz diretamente de um microfone, ao invés de lê-lo de uma memória externa, não será necessário efetuar nenhuma mudança no módulo de pré-ênfase. Além disso, o uso de módulos permite uma maior organização do código e facilita a conexão com a próxima etapa implementada, que, no caso do presente trabalho, foi o janelamento. Por outro lado, essa estratégia implica na chamada de funções (ou métodos) de outra biblioteca, gerando assim um retardo no processamento.

5.2 Divisão em quadros e janelamento

A velocidade com que as amostras são geradas pela pré-ênfase é maior do que a velocidade com que são utilizadas pelo janelamento. Isso porque, no janelamento, devido à sobreposição, algumas amostras são utilizadas mais de uma vez, para formar mais de um quadro. Diante disso, surge a necessidade de que esses dois módulos (Pré-ênfase e Janelamento) sejam executados em velocidades diferentes, sendo o primeiro mais lento que o segundo. Uma das otimizações propostas por Silva (2006) e Cipriano (2001) é utilizar um esquema de paralelismo, de modo a se obter um controle maior entre leitura e escrita das amostras. Esse paralelismo proporcionaria um ganho no tempo total de execução, quando comparado a um modelo sequencial das duas etapas.

No entanto, essa proposta e algumas outras simplificações pretendidas para esta etapa não puderam ser implementadas no Atmega328, uma vez que este não permite que o programador controle explicitamente o paralelismo fornecido pelo microcontrolador. Assim, a leitura e a escrita das amostras usadas na pré-ênfase e no janelamento tiveram que acontecer de forma sequencial ou minimamente paralela, por meio do uso do *pipeline* que o próprio Atmega328 oferece.

Outra limitação encontrada está relacionada ao uso de registradores para armazenar os valores utilizados tanto na pré-ênfase quanto no janelamento, o que diminuiria o tempo de acesso às amostras. Esse uso não foi possível porque, mesmo existindo registradores de propósito geral no Atmega328, eles são de apenas 8 bits (ATMEL CORPORATION, 2014). Como as amostras têm valores que ultrapassam muito facilmente os 8 bits, o uso desses registradores se torna inviável. Limitações no número de bits também foram enfrentadas ao se utilizar o PWM do microcontrolador para conversão D/A (digital/analógico) do sinal de voz.

Diante das limitações encontradas, a solução mais viável e que trouxe melhores resultados para a etapa de janelamento foi a utilização de um vetor para armazenar os valores pré-calculados da janela (Equação 4), ao invés de se realizar esse cálculo em tempo de execução. Além disso, devido à propriedade de simetria da janela de *hamming*, só é necessário

calcular e armazenar metade dos valores que formam uma janela.

Para respeitar o critério de estacionariedade citado anteriormente, foi utilizada uma janela de 252 amostras, o que corresponde a um quadro de aproximadamente 23 ms (SILVA, 2006). Para implementar a sobreposição de janelas, cada quadro é formado por 3 blocos de 84 amostras, o que corresponde a 1/3 do quadro e permite uma sobreposição de 66%.

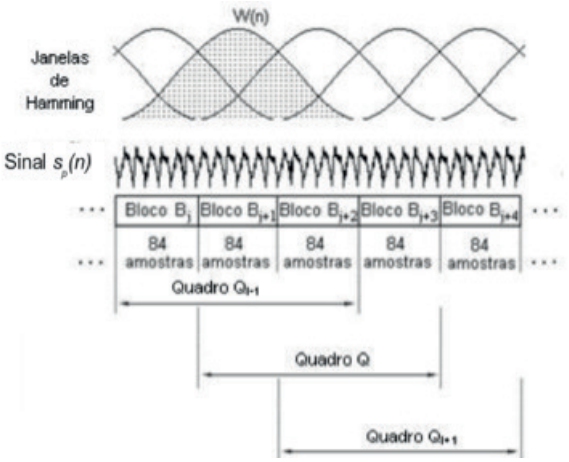
Cada janela utiliza 3 blocos para formar um quadro, sendo que, em quadros consecutivos, 2 blocos podem ser reaproveitados do quadro anterior. Ou seja, é realizado um deslocamento para esquerda entre os blocos a cada quadro, como é ilustrado na Figura 5.

Para implementação do esquema descrito, foram utilizados outros 3 vetores, chamados de B1, B2 e B3, com 84 valores cada, para armazenar os resultados da pré-ênfase.

Cada vetor corresponde a um bloco e armazena os valores da pré-ênfase, sendo multiplicado pelo vetor que armazena os valores da janela *J(n)*. A cada quadro, somente 1 bloco deve ser pré-enfatizado, uma vez que os outros 2 foram reaproveitados do quadro anterior, havendo assim uma redução no tempo de processamento.

Como foi mencionado anteriormente, apesar do tamanho da janela utilizada ser de 252 amostras, *J(n)* só precisa armazenar os primeiros 126 valores da janela de *Hamming* (n = 0, ..., 125). Os últimos 126 valores (n = 126, ..., 251) são obtidos com base em um contador decrescente.

Figura 5. Exemplo da relação entre blocos e quadros.



Fonte: SILVA, 2006

6 Resultados e Discussão

A partir das adaptações propostas e apresentadas anteriormente, foram realizados alguns testes e medições, de modo a quantificar o real ganho de desempenho em comparação à implementação tradicional das duas etapas. Para isso, foram considerados o número de ciclos de *clock* e o tempo de execução para cada uma das implementações (tradicional e adaptada).

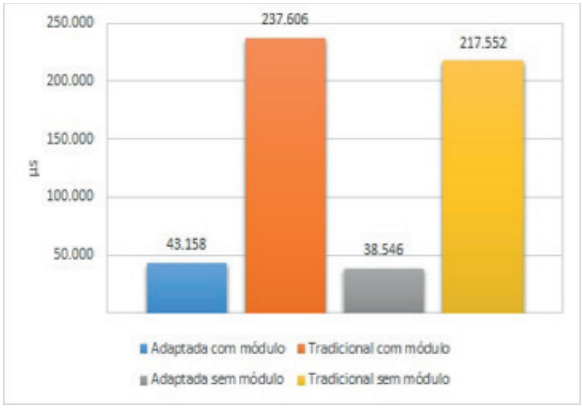
Para execução dos testes, foram utilizadas as amostras do sinal de voz referentes à sílaba "ba". O sinal foi capturado utilizando uma taxa de amostragem de 11 kHz e 16 bits de resolução, totalizando 6.652 amostras, cujos valores variaram de -10225 a 7828.

As amostras foram armazenadas em um cartão de memória SD. Como as amostras eram processadas à medida que eram lidas do cartão SD, o tempo total de execução sofria pequenas variações a cada execução. Isso acontece porque o tempo de leitura do cartão pode sofrer pequenas variações a cada solicitação. Diante desse comportamento, os valores considerados neste trabalho, para o tempo total de execução (Figuras 6 e 7) e número de ciclos de *clock* (Figuras 8 e 9), foram baseados na média de cinco execuções consecutivas.

Os valores apresentados nas figuras foram obtidos a partir da função *micros()* da biblioteca do Arduino. Essa função retorna o tempo de execução desde que o programa foi inicializado. Subtraindo-se o valor retornado por micros() em diferentes pontos do programa, é possível medir o tempo de execução de determinados trechos de código.

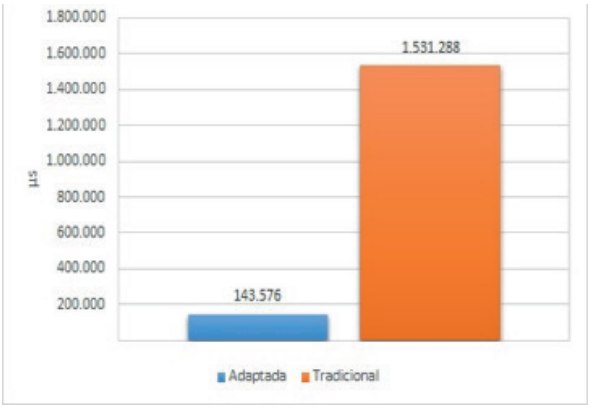
Com base nos tempos obtidos e na frequência de operação do microcontrolador (clock), foi possível calcular o número de ciclos necessários para cada trecho do programa. Essa estratégia foi utilizada para medir o tempo de execução da pré-ênfase isolada, ou seja, sem considerar o tempo para leitura das amostras no cartão de memória. O tempo de leitura do cartão de memória foi desconsiderado porque o objetivo era avaliar apenas as adaptações implementadas, sem nenhuma interferência. O mesmo foi feito para a etapa de divisão em quadros e janelamento.

Figura 6. Tempo de execução da Pré-ênfase (µs).



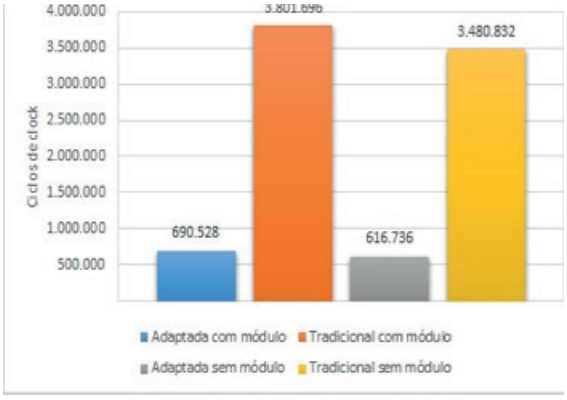
Fonte: Elaborada pelos autores.

Figura 7. Tempo de execução do Janelamento (µs).



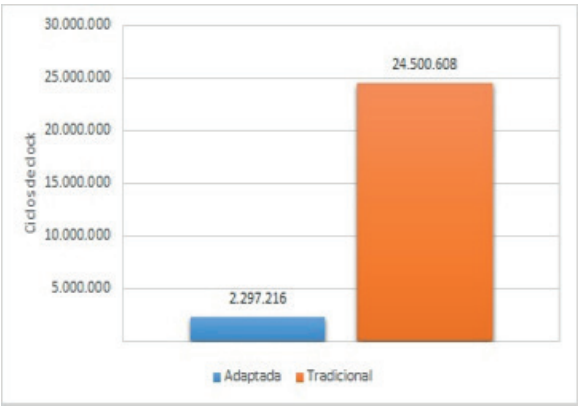
Fonte: Elaborada pelos autores.

Figura 8. Total de ciclos de *clock* da Pré-ênfase.



Fonte: Elaborada pelos autores.

Figura 9. Total de ciclos de clock do Janelamento.



Fonte: Elaborada pelos autores.

Analisando os gráficos das Figuras 6 e 8, é possível observar um ganho considerável de desempenho na função de pré-ênfase adaptada, em relação à implementação tradicional. O tempo e o número de ciclos são reduzidos a praticamente 1/5 dos valores originais. No que se refere ao uso de módulos separados, existe um aumento de aproximadamente 12% do tempo gasto nesse tipo de implementação, em relação à versão sem módulos.

A decisão de optar ou não pelo uso de módulos vai depender das prioridades do sistema a ser desenvolvido. Caso o tempo de resposta seja o mais importante, talvez valha a pena sacrificar todas as vantagens da modularização. Como exemplo dessas vantagens, além da que foi citada anteriormente – facilidade de mudar a origem das amostras de voz para um microfone –, tem-se a possibilidade de conectar, ao mesmo tempo, a saída da pré-ênfase a diferentes módulos, como o de janelamento e o PWM do Arduino. Ou seja, a decisão de usar ou não uma implementação modularizada consiste em um *tradeoff* entre desacoplamento e tempo de resposta.

Em relação à etapa de divisão em quadros e janelamento, como pode ser observado nas Figuras 7 e 9, foi obtida uma redução para aproximadamente 1/10 do tempo de execução e do número de ciclos de *clock* da implementação tradicional, que calcula o valor da janela em tempo de execução.

Embora a redução obtida seja considerável e, a princípio, animadora, é preciso destacar que as duas etapas analisadas são etapas muito iniciais e simples do processo completo de reconhecimento de fala, o qual foi apresentado na Seção 3 deste artigo. Outro ponto que merece ser destacado são as limitações de recursos do microcontrolador, apresentadas ao longo

da Seção 5.2. Além disso, nos testes realizados, foi utilizada uma pequena amostra de voz; à medida que forem utilizadas amostras maiores, como palavras e frases, o tempo de processamento também aumentará e poderá ser considerado demasiado longo.

7 Considerações finais

O crescente interesse por aplicações de reconhecimento de fala faz com que muitas pesquisas estejam sendo realizadas na área de processamento digital de sinais de voz. No entanto, a implementação desse tipo de aplicação em sistemas embarcados com poucos recursos computacionais torna-se complexa, uma vez que essas aplicações exigem certa capacidade de armazenamento e processamento.

O objetivo deste trabalho foi analisar o impacto no desempenho, resultante da implementação de simplificações matemáticas e estratégias de programação em duas etapas típicas de um sistema de reconhecimento de fala. Para isso, foram considerados o tempo e o número de ciclos de *clock* na execução das etapas de pré-ênfase e de divisão em quadros e janelamento de voz, com e sem as implementações propostas.

Mesmo com resultados tão positivos para as etapas implementadas – como a redução para 1/5 e 1/10 dos tempos originais da pré-ênfase e do janelamento, respectivamente –, muitas limitações foram encontradas durante a realização deste trabalho, as quais foram discutidas ao longo deste artigo.

Diante do que foi exposto, pode-se concluir que a implementação de um sistema de reconhecimento automático de fala eficiente e com um tempo de resposta aceitável em um dispositivo com recursos de *hardware* tão limitados é um processo complexo e difícil de alcançar. Isso não significa que seja inviável, até porque já existem no mercado alguns módulos de reconhecimento de fala para a plataforma Arduino. No entanto, a maioria desses módulos utiliza um *hardware* extra e realiza um reconhecimento limitado, o que corrobora a conclusão apresentada.

REFERÊNCIAS

ATMEL CORPORATION. **Atmel 8-bit microcontroller with 4/8/16/32kbytes in-system programmable flash:** datasheet summary. San Jose, CA: Atmel Corporation, 2014.

BARR, M. **Programming Embedded Systems in C and C++**. Sebastopol, CA: O'Reilly & Associates, 1999.

BENZEGHIBA, M. F.; BOURLARD, H. On the combination of speech and speaker recognition. In: EUROPEAN CONFERENCE ON SPEECH, COMMUNICATION AND TECHNOLOGY (EUROSPEECH" 03), 8., 2003, Geneva, Switzerland, **Proceedings...** Geneva: ISCA, 2003. p. 1361-1364.

CIPRIANO, J. L. G. **Desenvolvimento de arquitetura para sistemas de reconhecimento automático de voz baseados em modelos ocultos de Markov.** 2001. 123 f. Tese (Doutorado em Ciência da Computação) – Universidade Federal do Rio Grande do Sul, Porto Alegre, 2001.

CUNHA, A. M.; VELHO, L. **Métodos Probabilísticos para Reconhecimento de Voz.** Relatório Técnico. Rio de Janeiro: Laboratório VISGRAF – Instituto de Matemática Pura e Aplicada, 2003.

FECHINE, J. M. **Reconhecimento automático de identidade vocal utilizando modelagem híbrida:** Paramétrica e Estatística. 2000. 212 f. Tese (Doutorado em Engenharia Elétrica) – Universidade Federal de Campina Grande, Campina Grande, 2000.

FRANCIA III, G. A. Embedded Systems Programming. **Journal of Computing Sciences in Colleges**, v. 17, n. 2, p. 217-223, 2001.

FURUI, S. Cepstral Analysis Technique for Automatic Speaker Verification. **IEEE Transactions on Acoustics, Speech and Signal Processing,** v. 29, n. 2, p. 254-272, Apr. 1981.

KLEIJN, W. B.; PALIWAL, K. K. **Speech Coding and Synthesis**. New York: Elsevier Science, 1995.

LATHI, B. P. **Modern Digital and Analog Communication Systems.** 3rd ed. New York: Oxford University Press, 1998.

LIMA, A. A.; FRANCISCO, M. S.; LIMA NETTO, S.; RESENDE JUNIOR., F. G. V. Análise Comparativa de Parâmetros em Sistemas de Reconhecimento de Voz. In: SIMPÓSIO

BRASILEIRO DE TELECOMUNICAÇÕES, 18., 2000, Gramado. **Anais...** Gramado: SBrT, 2000.

MCROBERTS, Michael. **Arduino Básico.** São Paulo: Novatec, 2011.

PETRY, A.; ZANUZ, A.; BARONE, D. A. C. Reconhecimento automático de pessoas pela voz através de técnicas de processamento digital de sinais. In: SEMANA DA COMPUTAÇÃO – SEMAC, 11., 2000, São José do Rio Preto. **Anais...** São José do Rio Preto: UNESP, 2000.

RABINER, L.; SCHAFER, R. W. **Digital processing of speech signals.** New Jersey: Prentice Hall, 1978.

SHAUGHNESSY, D. O. **Speech Communications:** human and machine. New York: Wiley-IEEE Press, 2000.

SILVA, D. D. C. **Desenvolvimento de um IP CORE de Pré-Processamento Digital de Sinais de Voz para Aplicação em Sistemas Embutidos.** 2006. 108 f. Dissertação (Mestrado em Informática) – Universidade Federal de Campina Grande, Campina Grande, 2006.