UNIVERSIDADE FEDERAL DE VIÇOSA
CENTRO DE CIÊNCIAS EXATAS E TECNOLÓGICAS
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

FERNANDO CÉSAR PACHECO

REDE NEURAL ARTIFICIAL IMPLEMETADA EM MICROCONTROLADOR DE BAIXO CUSTO: APLICAÇÃO NO RECONHECIMENTO DE PADRÕES DE FALA

VIÇOSA
2010

FERNANDO CÉSAR PACHECO

REDE NEURAL ARTIFICIAL IMPLEMETADA EM MICROCONTROLADOR DE BAIXO CUSTO: APLICAÇÃO NO RECONHECIMENTO DE PADRÕES DE FALA

Monografia apresentada ao Departamento de Engenharia Elétrica do Centro de Ciências Exatas e Tecnológicas da Universidade Federal de Viçosa, para a obtenção dos créditos da disciplina ELT 490 – Monografia e Seminário e cumprimento do requisito parcial para obtenção do grau de Bacharel em Engenharia Elétrica.

Orientador: Prof. Dr. Leonardo Bonato Felix

VIÇOSA

2010

FERNANDO CÉSAR PACHECO

REDE NEURAL ARTIFICIAL IMPLEMETADA EM MICROCONTROLADOR DE BAIXO CUSTO: APLICAÇÃO NO RECONHECIMENTO DE PADRÕES DE FALA

Monografia apresentada ao Departamento de Engenharia Elétrica do Centro de Ciências Exatas e Tecnológicas da Universidade Federal de Viçosa, para a obtenção dos créditos da disciplina ELT 490 – Monografia e Seminário e cumprimento do requisito parcial para obtenção do grau de Bacharel em Engenharia Elétrica.

Aprovada em 03 de Dezembro de 2010.

COMISSÃO EXAMINADORA

Prof. Dr. Leonardo Bonato Felix - Orientador
Universidade Federal de Viçosa

Prof. Dr. Denílson Eduardo Rodrigues - Membro
Universidade Federal de Viçosa

Prof. Ms. Heverton Augusto Pereira - Membro
Universidade Federal de Viçosa

Agradecimentos

À FAPEMIG, pelo apoio financeiro concedido para a execução deste trabalho.

Aos companheiros Aluízio e Jésus pelas dicas e ensinamentos.

Ao Núcleo Interdisciplinar de Análise de Sinais – NIAS, que disponibilizou toda a infra-estrutura.

Ao meu orientador, Leonardo Bonato Felix, que me propôs a idéia desse trabalho.

À minha namorada, pelo apoio e compreensão.

E a todos os meus bons amigos que me acompanharam durante esta grande jornada que é a graduação em Engenharia Elétrica.

Resumo

Neste trabalho desenvolveu-se um protótipo microcontrolado que é capaz de extrair parâmetros caracterizadores de um sinal qualquer, desde que se façam os ajustes de offset necessário, e os utiliza como entrada de uma rede neural, previamente treinada e configurada computacionalmente, a fim de fazer o reconhecimento de padrões através de um dispositivo independente de computadores e de baixo custo.

Para construção desse dispositivo utilizou-se um microcontrolador PIC18F4550, o qual é responsável por todo o processamento realizado: conversão A/D, extração de características e reconhecimento de padrão. Para fins de teste, foram utilizados comandos de voz como sinal de entrada, e alcançou-se uma taxa de acerto de 100% para um banco de dados com dois tipos diferentes de comandos de voz.

Sumário

1 INTRODUÇÃO 9

2 OBJETIVOS 11

3 REVISÃO BIBLIOGRÁFICA 12

3.1 RECONHECIMENTO AUTOMÁTICO DE VOZ 12

3.1.1 Aquisição do sinal de fala 12

3.1.2 Extração de parâmetros caracterizadores do sinal e voz 13

3.1.3 Reconhecimento de padrão 15

3.2 REDE NEURAL ARTIFICIAL 16

3.2.1 Estrutura básica do neurônio artificial 16

3.2.2 Propriedades Das RNAs 18

3.2.3 Multi Layer Perceptron 19

3.3 IMPLEMENTAÇÃO DE RNAS EM SISTEMAS INDEPENDENTES DE COMPUTADORES 21

3.3.1 Ferramentas mais utilizadas 22

3.3.2 Trabalhos correlatos à implementação de RNAs em ambiente não computacional 24

3.3.3 Arquitetura utilizada no presente trabalho 25

4 MATERIAIS E MÉTODOS 26

4.1 AQUISIÇÃO DE DADOS 27

4.2 NORMALIZAÇÃO DOS DADOS 27

4.3 EXTRAÇÃO DE CARACTERÍSTICAS 27

4.4 COMUNICAÇÃO COM COMPUTADOR 29

4.5 CLASSIFICAÇÃO DO COMANDO DE VOZ 29

5 RESULTADOS E DISCUSSÕES 31

6 CONCLUSÕES 34

7 REFERÊNCIAS BIBLIOGRÁFICAS 35

ANEXO 1 37

Lista de Figuras

FIGURA 3.1 PROCESSO DE AQUISIÇÃO DO SINAL DE VOZ 13
FIGURA 3.2 FILTRO ADAPTATIVO NA CONFIGURAÇÃO DE PREDIÇÃO [6]. 14
FIGURA 3.3 ESQUEMA DE UM NEURÔNIO NATURAL [8]. 16
FIGURA 3.4 ESTRUTURA BÁSICA DE UM NEURÔNIO ARTIFICIAL 18
FIGURA 3.5 ALGUMAS FUNÇÕES DE ATIVAÇÃO 18
FIGURA 3.6 TOPOLOGIA DE UMA REDE NEURAL 20
FIGURA 4.1 RESUMO DE OPERAÇÃO DO PROTÓTIPO CONSTRUÍDO 26
FIGURA 4.2 CURVA DE TAXA DE ACERTO POR NÚMERO DE PARÂMETROS PARA RECONHECIMENTO DE CINCO COMANDOS. 28
FIGURA 4.3 ILUSTRAÇÃO DOS VALORES DOS COEFICIENTES PARA DOIS COMANDOS DISTINTOS. OS COEFICIENTES DO COMANDO "PARA" SÃO ENCONTRADOS NAS 100 PRIMEIRAS COLUNAS ENQUANTO AS 100 ÚLTIMAS SE REFEREM AO COMANDO "SIGA". 29
FIGURA 4.4 CURVA DE TAXA DE ACERTO POR NÚMERO DE COMANDOS PARA EXTRAÇÃO DE 20 COEFICIENTES LMS. 30
FIGURA 5.1 DESEMPENHO DA ANN IMPLEMENTADA EM MATLAB PARA O COMANDO “PARA” NO BANCO DE DADOS DE TESTE. NESSE CASO, A REDE DEVERIA APRESENTAR O VALOR DE SAÍDA 1 QUANDO PARÂMETROS LMS ADVINDOS DE UM COMANDO "PARA" FOSSEM USADOS COMO ENTRADA. 32

Lista de Tabelas

TABELA 4.1 RESUMO DO ALGORITMO LMS 28
TABELA 5.1 COMPARAÇÃO COEFICIENTES CALCULADOS PELO MCU E PELO COMPUTADOR. O DESVIO MÉDIO EM RELAÇÃO AO COMPUTADOR FOI DE 2,36X10-4. 31
TABELA 5.2. TEMPO DE PROCESSAMENTO DO DISPOSITIVO PARA COMANDO DE 500MS DE DURAÇÃO. 32

1 Introdução

Os estudos e projetos em redes neurais artificiais, usualmente denominadas “redes neurais” ou RNA, tem sido motivado desde o seu início devido ao fato de que o cérebro humano processa informações de forma completamente diferente e, em certos pontos, de forma bem mais efetiva do que o computador digital convencional. O cérebro é um sistema de processamento de informação altamente complexo, não-linear e paralelo. Ele tem a capacidade de organizar seus constituintes estruturais, conhecidos como neurônios, de forma a realizar processamentos com mais velocidade do que o mais rápido computador hoje existente. Considere como exemplo a visão humana, que se trata de um processo cerebral de reconhecimento perceptivo (reconhecer um rosto familiar inserido em uma cena não-familiar) realizado rotineiramente em aproximadamente 100-200ms, ao passo que tarefas de complexidade inferior demandam mais tempo para serem executadas em um computador convencional[1].

Outro exemplo é o sonar de um morcego, sistema ativo de localização por eco, que, além de fornecer informações sobre distância até um alvo, transmite informação sobre velocidade relativa, tamanho, e várias outras características do mesmo. Toda a complexa computação neural necessária pra extrair essa informação do eco ocorre no interior de um cérebro do tamanho de uma ameixa. E de fato, um morcego guiado por eco pode capturar seu alvo com uma taxa de sucesso de causar inveja em um engenheiro de radar ou de sonar[1].

Toda essa impressionante capacidade cerebral é devida, principalmente, à característica de desenvolvimento plástico do cérebro, ou seja, à capacidade de reorganizar os padrões e sistemas de conexões sinápticas com vistas à readequação do crescimento do organismo e às novas capacidades intelectuais e comportamentais do indivíduo. [2] Então, da mesma forma que a plasticidade é essencial para o funcionamento dos neurônios como unidades de processamento de informação do cérebro humano, ela também o é com relação às redes neurais construídas com neurônios artificiais. [1] Essa capacidade plástica do cérebro é mais conhecida como processo de aprendizagem, o qual está presente tanto no cérebro quanto nos métodos de RNA.

Diante das técnicas existentes de redes neurais artificiais e das potencialidades cerebrais que foram apresentadas, como o caso do sonar e da visão humana, a tendência

natural da ciência é tentar reproduzir algumas atividades realizadas pelo cérebro em um sistema dotado de técnicas de RNA. E é objetivando tal fato que se desenvolveu este trabalho, no qual é desenvolvido um *hardware* microcontrolado programado com uma RNA genérica e com um sistema de captura e de extração de características de sinais externos, os quais juntos podem ser utilizados para, primeiramente, extrair informações de um sinal elétrico qualquer (fazendo assim o papel de algum órgão sensitivo) e, em seguida, através de um algoritmo de treinamento computacional, ajustar essa rede para responder a uma saída desejada, conseguindo assim a característica de aprendizado.

Para o caso específico desse projeto, o protótipo foi ajustado para tratar e responder a sinais de voz, servindo assim como um reconhecedor automático de comandos de voz. O que não exclui a possibilidade de utilizá-lo para quaisquer outros fins que demandem de extração de características de um sinal elétrico externo e de reconhecimento de padrões.

2 Objetivos

Construir um classificador de padrões, capaz de operar de forma independente de um computador, através de um protótipo microcontrolado.

O protótipo deve fazer todas as etapas básicas de um algoritmo de classificação de padrões:

- Aquisição de um sinal externo.
- Extração de parâmetros caracterizadores
- Calcular a resposta de uma rede neural artificial previamente treinada e mostrá-la em um LCD

Além disso, o dispositivo deve ser capaz de se comunicar com um computador a fim de enviar informações necessárias para efetuar o treinamento de uma RNA, o qual será efetuado em ambiente computacional.

3 Revisão Bibliográfica

Neste capítulo serão introduzidos alguns conceitos básicos sobre o processo de reconhecimento de voz, abordando as técnicas mais utilizadas e explicando algumas adaptações que foram necessárias.

Em seguida será detalhada a teoria de Redes Neurais Artificiais (RNAs), que são importantes para o desenvolvimento deste projeto. E devido ao escopo do trabalho, o estudo sobre conceitos de RNAs será baseado no modelo Perceptron de Múltiplas Camadas (MLP – *Multi-Layer Perceptro*n).

Uma vez estudada as principais teorias necessárias para o desenvolvimento desse projeto, serão apresentadas as ferramentas que constam na literatura para a implementação de RNAs em sistemas independentes de computadores.

3.1 Reconhecimento automático de voz

Segundo [3], de forma simplificada, o processo de reconhecimento automático de voz (RAV) consiste na extração de informação lingüística no sinal de fala. E normalmente esse processo acontece em três principais passos:

- Aquisição do sinal de fala
- Extração de parâmetros caracterizadores do sinal de voz
- Reconhecimento de padrão

3.1.1 Aquisição do sinal de fala

O primeiro passo consiste em realizar a captação do sinal de fala, e para isso o sinal deve passar pelas etapas de amplificação e filtragem passa-baixa *anti-aliasing* para em seguida ser captado por um conversor analógico digital e ser armazenado em um *buffer*. Tal processo está demonstrado na Figura 3.1.

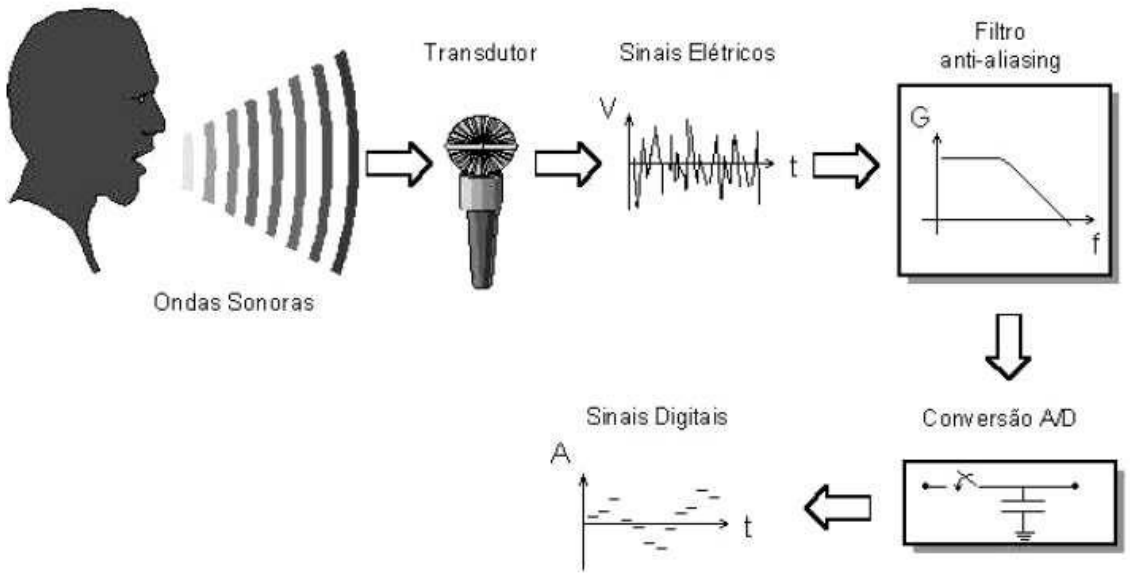

Figura 3.1 Processo de aquisição do sinal de voz

3.1.2 Extração de parâmetros caracterizadores do sinal e voz

Com o sinal armazenado, passa-se para o processo de tratamento digital do sinal de voz, no qual são extraídos parâmetros caracterizadores do mesmo. Nos trabalhos de reconhecimento automático de voz existentes na literatura essa etapa de extração de parâmetros pode ser feita utilizando-se várias técnicas e, dentre essas, as que se destacam mais são os métodos de banco de filtros, transformada rápida de Fourier, análise homomórfica (Cepstrum) e os métodos de codificação por predição linear (LPC – Linear Predictive Coding) [4].

Os métodos de banco de filtros, transformada rápida de Fourier e a de codificação por predição linear foram largamente usados para a extração de informação espectral da fala. Entretanto, elas apresentam restrições. A mais pronunciável é a de não resolver as características do trato vocal. O método cepstrum trata essa restrição. A idéia por trás do cepstrum é a obtenção de uma relação linear entre a excitação da energia do sinal e(n) com o filtro utilizado v(n). A literatura reporta que os melhores resultados, na maioria dos casos, são os obtidos pelo método cepstrum [5]

Todas essas técnicas comuns de extração de parâmetros caracterizadores do sinal de voz são normalmente executadas em ambiente computacional, e como o propósito desse trabalho é realizar todo o processamento digital do sinal de voz através de um hardware microcontrolado, utilizou-se um método de modelagem de sinais, conhecido

como filtragem adaptativa, que possui um algoritmo de baixa complexidade e, por isso, de fácil execução computacional.

O filtro adaptativo desenha-se de forma automática, baseando-se num algoritmo recursivo para ajuste dos seus parâmetros, com o intuito de minimizar uma função de custo. Ele começa com um conjunto de parâmetros iniciais arbitrários, o que demonstra desconhecimento do ambiente, e caso esse sistema seja estacionário os parâmetros do filtro convergem, a cada iteração, para a solução ótima do filtro de Wiener. Já em um ambiente não estacionário o algoritmo possibilita ao filtro a capacidade de seguimento, conseguindo acompanhar as variações estatísticas ao longo do tempo, desde que suficientemente lentas [6].

Sendo assim, por o sinal de voz ser não estacionário, os parâmetros de ajustes do filtro adaptativo, que neste caso serão os parâmetros caracterizadores do comando de voz, apenas acompanham as variações estatísticas do mesmo, o que pode ser utilizado como uma espécie de "assinatura" desse sinal. Para melhor compreensão desse processo, na Figura 3.2 é demonstrada a configuração de um filtro adaptativo na configuração de predição.

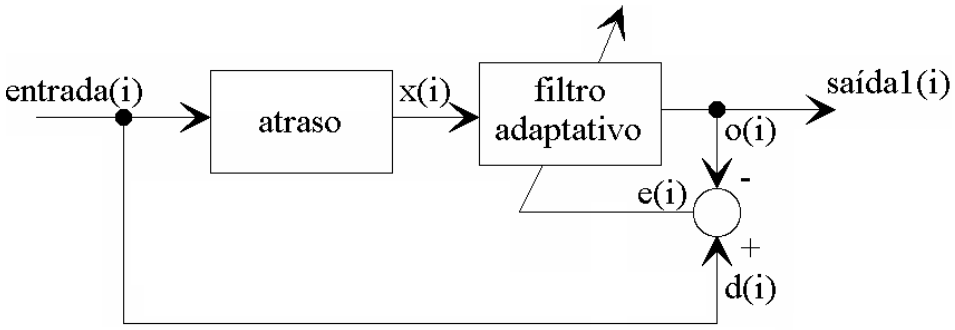

Figura 3.2 Filtro adaptativo na configuração de predição [6].

A função do filtro adaptativo nesta situação consiste em disponibilizar a melhor predição do valor presente de um sinal aleatório. O valor presente do sinal serve de resposta desejada para o filtro adaptativo. É importante notar que o sinal de referência do filtro consiste apenas nos valores passados do processo. Em alguns casos o sinal de saída do filtro adaptativo serve como saída do sistema, o que lhe traz a característica de preditor.

Como pode ser observado na Figura 3.2, a cada iteração o filtro adaptativo é ajustado, através do valor do erro de predição e(i), pelo algoritmo de adaptação. Como existem vários algoritmos, eles são divididos em duas classes[6]:

- Algoritmos de gradiente
- Algoritmos de mínimos quadrados

Os algoritmos de gradiente baseiam-se no filtro de Wiener, estimando o gradiente da superfície da função de custo. Tem como principal vantagem a baixa complexidade. São bons na exploração de superfícies pouco complicadas.

Os algoritmos de mínimos quadrados são baseados no filtro de Kalman. De forma determinística minimizam a soma dos quadrados dos erros parciais. Tem como vantagem a baixa sensibilidade a mínimos locais da superfície da função de custo e a maior velocidade de convergência comparativamente com os algoritmos de gradiente. Apresentam como principais desvantagens as maiores exigências computacionais e problemas de estabilidade.

Tendo em vista a necessidade de um algoritmo de baixa complexidade, será utilizado nesse trabalho o algoritmo de gradiente estocástico *Least Mean Squares* (LMS), que está detalhadamente explicado no Anexo 1.

3.1.3 Reconhecimento de padrão

Uma vez obtido os parâmetros caracterizadores do sinal de voz passa-se para a etapa de reconhecimento de padrões. Para essa etapa existem dois algoritmos de métodos estatísticos de maior destaque [7]: as Redes Neurais Artificiais (*Artificial Neural Networks*) e os Modelos Ocultos de Markov (*Hidden Markov Models*).

Através de um algoritmo de RNA é possível criar um modelo que, ao receber como entrada os parâmetros caracterizadores de um sistema qualquer, ele reconhece um padrão nesses dados de entrada e responde um valor numérico que identifica uma determinada característica nesse sistema dentre as demais que foram previamente informadas para o algoritmo. E trabalhando com um sistema de sinal de voz, por exemplo, pode-se obter como resposta o reconhecimento de um comando dentre os demais que foram previamente informados para o algoritmo.

Na próxima sessão será detalhado esse algoritmo de reconhecimento de padrão (RNA) e, como o foco deste trabalho é a implementação de Redes Neurais Artificiais em Hardware, não será estudado aqui o método Modelos Ocultos de Markov.

3.2 Rede neural artificial

Na sua forma mais geral, uma rede neural artificial é uma máquina projetada para modelar a forma como o cérebro realiza uma tarefa particular. Normalmente é implementada utilizando-se componentes eletrônicos (implementação em hardware) ou é simulada por programação em um computador digital (implementação em software) [1]. Uma rede neural é um processador maciçamente e paralelamente distribuído, formado de unidades de processamento simples, que tem propensão para armazenar conhecimento experimental e torná-lo disponível para o uso. Ela se assemelha ao cérebro em dois aspectos [1]:

a) O conhecimento é adquirido pela rede a partir de seu ambiente através de um processo de aprendizagem.

b) Forças de conexão entre neurônios, conhecidas como pesos sinápticos, são utilizadas para armazenar o conhecimento adquirido.

Uma das características mais importantes das redes neurais é sua capacidade de generalização. Isto é, a rede neural produz saídas adequadas para entradas que não estavam presentes durante o treinamento. A generalização e a aprendizagem tornam possíveis que as redes neurais possam resolver problemas complexos (de grande escala) que são atualmente não tratáveis [1]

3.2.1 Estrutura básica do neurônio artificial

Nesta seção será discutida, brevemente, a estrutura básica de um neurônio artificial. Para introduzir os conceitos básicos, a Figura 3.3 apresenta o esquema de funcionamento de um neurônio natural.

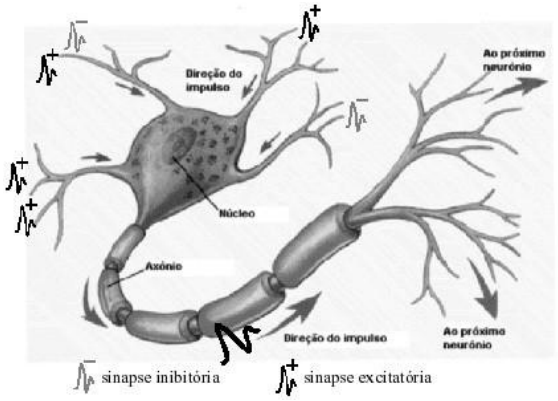

Figura 3.3 Esquema de um neurônio natural [8].

Os neurônios são divididos em três seções: o corpo da célula, os dendritos e o axônio, cada um com funções específicas, porém complementares. O corpo do neurônio mede apenas alguns milésimos de milímetros, e os dendritos apresentam poucos milímetros de comprimento. O axônio, contudo, pode ser mais longo e, em geral, tem calibre uniforme. Os dendritos têm por função receber as informações, ou impulsos nervosos, oriundas de outros neurônios e conduzi-las até o corpo celular. Aqui, a informação é processada e novos impulsos são gerados. Estes impulsos são transmitidos a outros neurônios, passando através do axônio até os dendritos dos neurônios seguintes. O ponto de contato entre a terminação axônica de um neurônio e o dendrito de outro é chamado de sinapse. É pelas sinapses que os nodos se unem funcionalmente, formando redes neurais. As sinapses funcionam como válvulas, e são capazes de controlar a transmissão de impulsos - isto é, o fluxo da informação - entre os nodos na rede neural. [2].

Isso possibilita a passagem dos sinais oriundos dos neurônios pré-sinápticos para o corpo do neurônio pós-sináptico, onde são comparados com os outros sinais recebidos pelo mesmo. Se o sinal elétrico transmitido ao corpo celular, em um intervalo curto de tempo, é suficientemente alto, a célula "dispara" e produz um impulso que é transmitido para as células seguintes (nodos pós-sinápticos). Este sistema simples é responsável pela maioria das funções realizadas pelo nosso cérebro. E a capacidade de realizar funções complexas surge com a operação em paralelo de todos os 10^{11} nodos do cérebro[9].

Tendo em vista essa configuração do neurônio biológico, Warren McCulloch, psiquiatra e neuroanatomista, e Walter Pitts, matemático, propuseram em 1943 um modelo matemático para o mesmo. O modelo em si era uma simplificação do neurônio biológico até então conhecido na época e pode ser visualizado na Figura 3.4. Para representar os dendritos, o modelo constou de *n* terminais de entrada de informações x_1, x_2, ..., x_n e simplesmente um terminal de saída *y*, para representar o axônio. Cada entrada apresenta um coeficiente ponderador que faz o papel das sinapses, sendo que estes coeficientes são valores reais. De forma análoga ao neurônio biológico, a resposta só ocorre quando a soma ponderada dos sinais de entrada ultrapassa um limiar pré-definido, realizando, portanto, uma atividade semelhante a do corpo celular. No modelo proposto, o limiar foi definido de forma Booleana, dispara ou não dispara, o que é está representado na Figura 3.4 pela função de ativação φ [9].

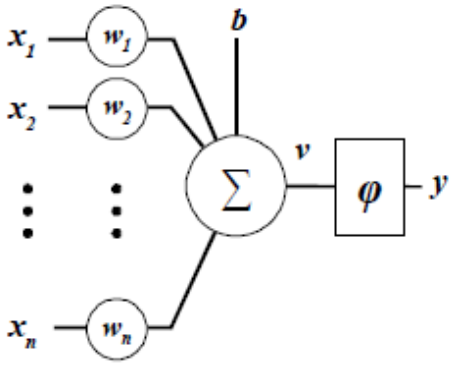

Figura 3.4 Estrutura Básica de um Neurônio Artificial

A partir do modelo proposto por McCulloch e Pitts, foram derivados vários outros modelos que permitem a produção de uma saída qualquer, não necessariamente booleana, e com diferentes funções de ativação. A Figura 3.5 ilustra graficamente quatro funções de ativação φ diferentes: a) função linear, b) função rampa, c) função degrau *(step)* e d) função tangente hiperbólica *(tansig)*.

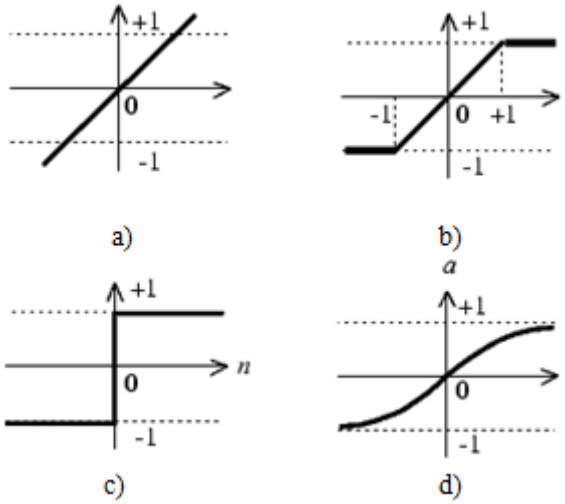

Figura 3.5 Algumas funções de ativação

Uma entrada adicional do neurônio artificial é o *bias*. A palavra *bias*, em inglês, significa "viés, linha oblíqua ou diagonal; movimento oblíquo" em sentido figurado: "tendência, propensão; preferência, preconceito; predisposição". Sendo assim, a função do *bias é* aplicar uma tendência ao campo local induzido, produzindo um deslocamento a seus valores.

3.2.2 Propriedades Das RNAs

Nesta seção estão listadas algumas vantagens das RNAs para a solução de problemas computacionalmente complexos.

O uso de redes neurais oferece as seguintes propriedades uteis e capacidades [1]:

1. **Não-linearidade**: Um neurônio artificial pode ser linear ou não-linear. Uma rede composta de neurônios não-lineares é, por definição, não-linear.

2. **Mapeamento de Entrada-Saída**: Um conjunto de treinamento consiste de uma serie de dados que são usados como entrada de uma rede neural e outra série de dados que correspondem às saídas desejadas da rede para cada entrada. Com essas informações, a RNA pode ser treinada para fornecer as saídas desejadas para os mesmos padrões de entrada. Entretanto, o mais importante é que a rede neural treinada fornecera saídas equivalentes para entradas que não sejam iguais, mas que apresentem semelhança com as utilizadas no seu treinamento. Essa característica e muito útil no reconhecimento de padrões.

3. **Adaptabilidade**: As redes neurais têm a capacidade de adaptar seus pesos sinápticos sob influência de modificações do meio ambiente. Em particular, uma rede neural treinada para operar em um ambiente específico pode ser facilmente re-treinada para lidar com pequenas modificações nas condições operativas do ambiente.

4. **Resposta a Evidências**: No contexto de classificação de padrões, uma rede neural pode ser projetada para fornecer informação não somente sobre qual padrão particular selecionar, mas também sobre a confiança ou crença na decisão tomada.

3.2.3 Multi Layer Perceptron

As RNAs com uma camada de neurônio são capazes de resolver problemas linearmente separáveis, porém, apesar de resolver uma vasta gama de problemas, existe, por outro lado, outra vasta coleção de problemas não linearmente separáveis. Este problema foi proposto por Minsky e Pappert na década de 70, quando, em suas publicações, depreciaram a habilidade das RNA de encontrar soluções para simples problemas, como por exemplo, a modelagem do “Ou Exclusivo” da lógica digital. A solução encontrada para contornar este problema e como conseqüência retomar as pesquisas sobre RNA, até então desacreditas por Minsky e Pappert, foram as estruturas neurais de múltiplas camadas, também conhecida como redes MLP (Multi Layer Perceptron), na década de 80.

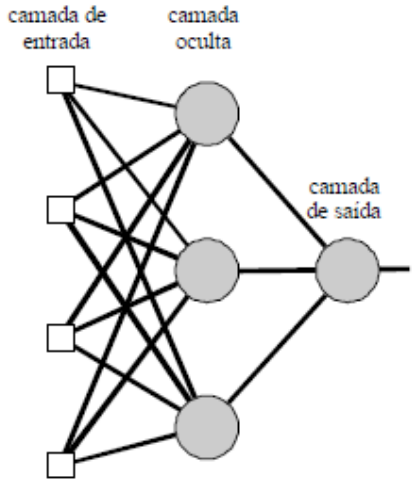

Figura 3.6 Topologia de uma rede neural

As redes MLP apresentam a arquitetura mostrada na Figura 3.6, onde se encontram a camada de entrada, as camadas intermediárias (ou ocultas) e a camada de saída. O número de variáveis da camada de entrada depende diretamente do número de características agrupadas no vetor das amostras. O número de neurônios das camadas intermediárias depende da complexidade do problema. E a camada de saída contém o número de neurônios necessário para executar a codificação das amostras de entrada.

O número de neurônios das camadas intermediárias é determinado de forma empírica, atentando para o caso de *overfitting* (ou superajuste), que é o caso onde existe uma grande quantidade de neurônios e a estrutura ao invés de generalizar as informações, acabar por memorizar os padrões apresentados, não sendo capaz de classificar padrões semelhantes. Outro efeito do superajuste é que a RNA além de armazenar as características relevantes extraídas das amostras, guarda em seus pesos informações de ruídos que a princípio não revelam interesse. Por outro lado, caso o número de neurônios seja inferior ao desejado, pode ocorrer um *underfitting* e a RNA não convergir para uma resposta devido a uma sobrecarga de informações a serem armazenadas em poucos pesos.

O treinamento das redes MLP é normalmente realizado pelo algoritmo de retro-propagação do erro (ou *back-propagation*), um algoritmo supervisionado que realiza o ajuste dos pesos, a partir do erro existente entre os pares de amostra de dados de entrada e saída da RNA. Este algoritmo apresenta duas fases denominadas *forward* e *backward*. A fase *forward* é utilizada para que seja encontrada uma saída a partir dos valores de entrada. A fase *backward* compara esta saída com a saída desejada e retorna atualizando os valores dos pesos das conexões dos neurônios da estrutura. Por isso, para o

treinamento de uma rede é necessário dispor de um banco de dados com os valores de entrada de um sistema, e o valor, ou valores, de saída correspondente a cada conjunto de valores de entrada. Não existe um tamanho fixo desse banco de dados, porém quanto maior ele seja maiores são as chances de se obter uma RNA que solucione de maneira mais eficiente um determinado problema.

Basicamente, o processo de treinamento da rede MLP tem como objetivo, durante o treinamento, a minimização da função de erro quadrático médio (mse – *mean square error*). O treinamento da RNA deve ocorrer até que se complete um número predeterminado de iterações para atualização dos pesos ou quando o erro quadrático médio encontrar-se abaixo de um valor pré-estabelecido.

Além desses critérios de parada do treinamento, existem outros que visam evitar o overfitting, já que, infelizmente, é difícil saber de antemão o tamanho ideal da rede que evite tal acontecimento. E dentre esses o mais utilizado é o parada precoce (*early stopping)*.

A parada precoce é um método de melhoria da generalização das RNAs, e oferece modificação durante a etapa de treinamento. Esta técnica fraciona as amostras em três grupos distintos. O primeiro grupo são as amostras de treinamento, que são utilizadas para ajustar os pesos. O segundo grupo são as amostras de validação. O erro de validação é monitorado durante o processo de treinamento. Normalmente, no início do treinamento, tanto o erro de treinamento quanto o erro de validação decrescem no decorrer das iterações, entretanto, à medida que começa a ocorrer o *overfitting*, o erro de validação começa a crescer e a partir desse momento, passado um pré-determinado número de iterações, o treinamento é parado e os pesos e os *bias* obtidos na iteração de menor erro de validação são retornados. O terceiro grupo é constituído das amostras de teste, as quais não são utilizadas no treinamento, mas são uteis para medir o desempenho do modelo obtido e, também, para comparar diversos modelos [10].

3.3 Implementação de RNAs em sistemas independentes de computadores

Com o intuito de melhor entender o tema de construção de RNAs fora do ambiente computacional, esta sessão apresentará, resumidamente, os principais métodos utilizados para esse fim e abordará as principais características da ferramenta atualmente mais utilizada para esse propósito: o FPGA.

Em seguida, a fim de entender as potenciais aplicações desse tema, serão apresentados alguns exemplos de trabalhos publicados nessa área, e, por fim, será abordada a tecnologia utilizada nesse trabalho juntamente com os motivos da utilização da mesma.

3.3.1 Ferramentas mais utilizadas

Como já foi visto, Redes Neurais Artificiais (RNAs) são arquiteturas massiçamente paralelas que podem resolver uma série de complexos problemas de otimização, classificação e de modelagem. Seu paralelismo juntamente com a capacidade de aprendizado podem oferecer uma maneira rápida, flexível e alternativa de baixo custo aos paradigmas tradicionais de computação[11].

Sua implementação normalmente é realizada por meio de construção em hardware ou de *softwares* computacionais, porém, a característica de velocidade de processamento proporcionada pelo paralelismo dessas redes não pode ser aproveitada no caso de execução em software, pois a arquitetura do hardware que executa tal software é de natureza seqüencial, baseado na arquitetura de *Von Neumann,* e isso torna essencial a construção de RNAs em *hardware* quando se tem o desempenho como uma necessidade crítica de um sistema[11].

Portando, foi mantendo o foco no desempenho que na bibliografia os Neuro-processadores (NPs) e os ASICs (*Application Specific Integrated Circuits* - Circuitos Integrados de Aplicação Específica) foram os mais populares meios de implementação de redes neurais em hardware. No entanto, os algoritmos de NPs tende a ser mais específico, o que leva a um ciclo de projeto mais complexo e demorado (maior *time-to-market*). Como resultado, os neuro-processadores não correspondem ao ritmo de avanço visto em computação, e tendem a ficar desatualizados mais rapidamente. E a falta de flexibilidade e o alto custo de ASICs também se revelaram pontos antieconômicos para a implantação de redes neurais em sistemas de produção de baixo volume [11].

Sendo assim, desde o início 1990, dispositivos programáveis como o FPGA (*Field Programmable Gate Arrays*) tem sido utilizado como uma alternativa para implementações tradicionais de redes neurais. FPGAs fornecem uma alternativa de baixo custo e, ainda, junta a eficiência e velocidade dos ASICs com a flexibilidade dos neuro-processadores [11].

A programação dos FPGAs pode ser feita por diversas ferramentas de projeto e síntese fornecidas por companhias como a Altera, Xilinx, a Atmel, entre outras. Os circuitos podem ser especificados utilizando-se linguagens de descrição de hardware de alto nível (HDLs – *Hardware Description Languages*), como a VHDL ou a Verilog [12]. Ferramentas de *software* realizam a síntese dos arquivos com as descrições e produzem um código de configuração binário: o arquivo de *bitstream*. Este arquivo é utilizado para configurar o FPGA.

FPGAs permitem a implementação de algoritmos diretamente em *hardware*, em vez de em *software* (como, por exemplo, os programas utilizados nos microcontroladores). A implementação em *software*, como já foi dito, tem limitações oriundas da natureza sequencial das arquiteturas de Von Neumann que executam os programas [13]. Por outro lado, o paralelismo existente no *hardware* pode ser explorado nas implementações em FPGA, em busca da melhoria de desempenho dos projetos. Além disso, a flexibilidade desses dispositivos permite que se configure o sistema mesmo em tempo de execução, o que é conhecido como Reconfiguração Dinâmica [14],

Esses Sistemas Reconfiguráveis são normalmente baseados em um microcontrolador e um FPGA funcionando em conjunto. Uma parte do sistema é implementada em *hardware*, no FPGA, e a outra parte roda em *software*, no microcontrolador.

Então, entendendo que a aplicação de RNAs em hardware almeje alcançar a mais importante característica da rede neural biológica, que é a aprendizagem, é natural que se busque uma arquitetura que seja capaz de adaptar as conexões entre neurônios de acordo com as necessidades do sistema externo. E é a característica de Reconfiguração Dinâmica que permite tal propriedade, passando assim a ser um quesito importante para se atingir algum nível de aprendizagem em *hardware*.

Mesmo que utilização de arquiteturas não seqüenciais seja predominante na implementação de redes neurais fora do ambiente computacional, ainda encontram-se alguns trabalhos que utilizam a arquitetura sequencial, como será constatado na sessão 3.3.2, uma vez que não são muitos os processos que necessitam criticamente do alto desempenho dos FPGAs.

Serão apresentados a seguir alguns trabalhos encontrados na literatura que conseguiram avanços significativos no aproveitamento das características de

paralelismo e reconfiguração dinâmica ao implementar uma RNA fora do ambiente computacional.

3.3.2 Trabalhos correlatos à implementação de RNAs em ambiente não computacional

Em [15] descreve-se um tipo de rede neural, baseada num algoritmo chamado *Pattern Shaper*, que gera sinais elétricos para estimular músculos em indivíduos com lesões na espinha dorsal. O FPGA faz parte de uma placa chamada Pattern *Shaper Board*. Ele gera um sinal digital que passa por um conversor DA (digital para analógico) e é enviado para os músculos. Uma série de sensores são utilizados para verificar se o movimento produzido pelo estímulo foi o esperado. Se houver divergência, as saídas desses sensores alimentam um sistema de adaptação dos pesos da rede. Dessa forma, o equipamento pode se adequar automaticamente diante de duas situações comuns: "cada músculo de cada individuo possui diferentes características de resposta" e "as propriedades dos músculos podem variar com o tempo por causa da fadiga muscular". Outro ponto digno de nota e que os FPGAs neste sistema são configurados automaticamente no inicio da operação. Os dados de configuração dos dispositivos ficam armazenados em uma SPROM (*Serial Programmable Read-Only Memory* – Memória Serial Programável Somente de Leitura) e são transferidos para os FPGAs quando os circuitos são ligados.

Em [16], duas implementações de controladores para um motor SRM (*Switched Reluctance Motor* – Motor de Relutância Chaveado) foram realizadas. Uma RNA implementada em um DSP (*Digital Signal Processor* – Processador Digital de Sinais), *Texas Instruments* TMS320C25, foi apropriada para controlar o motor em baixa e media velocidades. O tempo de resposta desta implementação era de 84 microssegundos. Entretanto, para controlar o motor em alta velocidade, o tempo de resposta deve ser de no Maximo 39 microssegundos. Uma implementação de rede neural em um FPGA Xilinx XC4000 alcançou um tempo de resposta de dois microssegundos.

Em [17] é apresentada uma arquitetura chamada FAST (*Flexible Adaptable-Size Topology* – Topologia Flexível de Tamanho Adaptável), que permite a alteração do tamanho das redes em tempo de execução, aumentando ou diminuindo a quantidade de neurônios. Foi construída em uma máquina especializada, que, dentre outros recursos, possui quatro FPGAs Xilinx XC4013-6, dos quais apenas dois foram utilizados. Realizou-se um teste prático onde a rede neural implementada reconheceu e agrupou

pontos de uma imagem por similaridade de cor. Estudos estão sendo feitos pelos autores para aproveitar a reconfigurabilidade dinâmica dos novos FPGAs.

[18] Apresenta neurônios construídos em placas com vários FPGAs, dando ênfase a velocidade do processamento obtida. Três sistemas foram implementados nesta arquitetura. Comparando-os com suas versões em software, que rodaram em um DEC5100, os autores informam que as versões em *hardware* foram aproximadamente 600 vezes mais rápidas.

3.3.3 Arquitetura utilizada no presente trabalho

Como pôde ser constatado, o FPGA tem sido a principal ferramenta utilizada para aproveitar o desempenho oriundo da característica de paralelismo das RNAs. Porém, como a velocidade de execução da RNA, não é uma necessidade crítica, esse trabalho não aproveitará os desempenhos oferecidos por essa arquitetura e, para o propósito da construção da RNA em ambiente não computacional, será utilizado um microcontrolador de natureza sequencial, arquitetura Harvard, o qual terá, internamente, um algoritmo que executa uma RNA pré-determinada.

O algoritmo da RNA foi programado de forma a permitir uma fácil reconfiguração da estrutura da rede, permitindo assim que o usuário encontre sem maiores dificuldades a melhor rede que solucione o problema em questão, facilitando assim o processo de aprendizagem do dispositivo.

4 Materiais e Métodos

Na Figura 4.1 está apresentado o resumo de operação do sistema que foi construído. Como pode ser observado, o início da operação é representado pelo microfone, onde o sistema fica aguardando a chegada de algum comando de voz. Uma vez pronunciado algum comando, o microcontrolador inicia o processo de conversão A/D e armazena o sinal digital em uma memória externa. Em seguida, estando o sinal armazenado na memória, inicia-se o algoritmo de extração de características, o qual transforma todo o sinal em apenas alguns coeficientes caracterizadores. A partir desse momento, se o protótipo estiver funcionando no modo de execução, ele irá calcular a resposta da RNA e mostrar o resultado em um LCD, senão, caso esteja no modo de configuração, ele irá enviar os coeficientes caracterizadores para um computador pessoal (PC) via USB, onde será realizado o treinamento da RNA para futura reconfiguração do microcontrolador.

Nas próximas sessões serão explicadas todas as etapas desse ciclo de operação, mostrando detalhes de cada algoritmo e modo de operação.

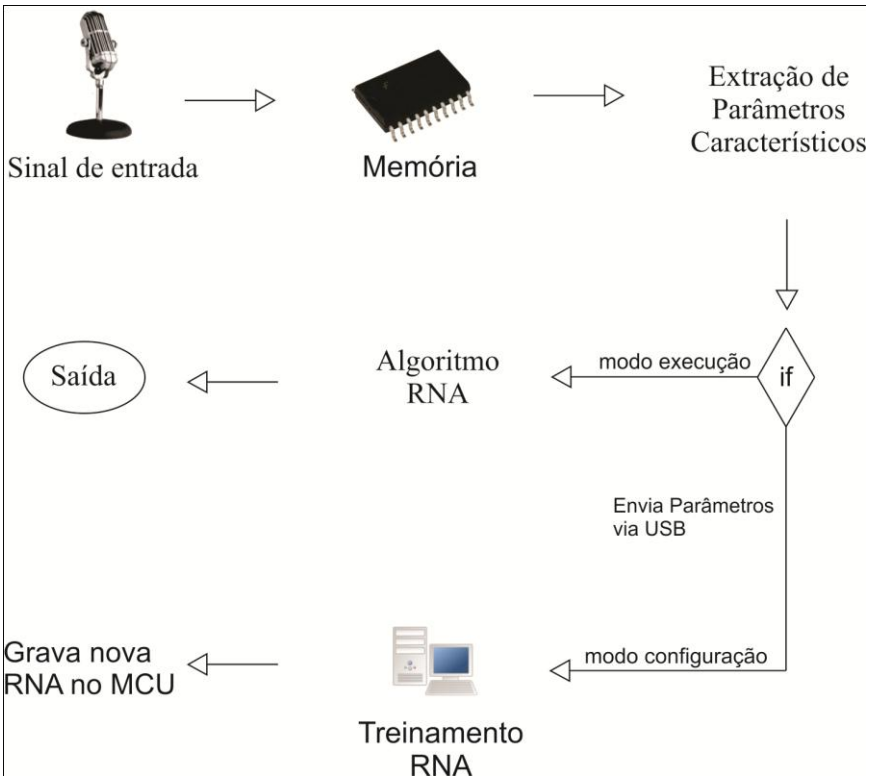

Figura 4.1 Resumo de operação do protótipo construído.

4.1 Aquisição de dados

Para a captação do sinal de voz utilizou-se um microfone de eletreto conectado a um amplificador operacional com ligação não inversora. Para evitar o efeito *aliasing*, utilizou-se um filtro passa-baixa *Butterworth* de quarta ordem com freqüência de corte de 2kHz entre o circuito amplificador e a entrada do conversor analógico-digital do microcontrolador PIC18F4550. Como o corte desse filtro não é abrupto a conversão A/D foi realizada na freqüência de 5kHz, garantindo assim o cumprimento do teorema de amostragem de *Nyquist*. É importante destacar que essa taxa de amostragem foi escolhida de forma a minimizar o tempo de processamento, porém sem diminuir a taxa de acerto do dispositivo.

Na conversão analógico-digital trabalhou-se com o microcontrolador (MCU) realizando conversões A/D com resolução de oito bits. Os dados obtidos do conversor A/D foram armazenados em uma memória S-RAM externa de 32kbytes. O acesso ao endereçamento dessa memória foi feito de forma direta, o que permitiu a utilização de apenas um pino do MCU. Isso pôde ser obtido ligando em cascata dois registradores 74164 diretamente aos 14 pinos de endereçamento da memória, assim o MCU pôde enviar o endereço de memória bit por bit para o registrador e o mesmo os disponibilizou de forma paralela à memória. Para todas as operações do MCU foi utilizada a freqüência de *clock* máxima do mesmo que é de 48MHz.

4.2 Normalização dos dados

Com os dados do comando de voz armazenados na memória externa, o MCU faz a normalização dos mesmos através da aplicação da Equação 1. O que homogeneíza a intensidade sonora de cada comando.

$$x(i) = \frac{x(i)}{máximo} \times 255 \quad (1)$$

4.3 Extração de características

Como já explicado, devido às limitações de desempenho computacional que se encontra em um microcontrolador, utilizou-se o algoritmo de gradiente estocástico – *Least Mean Square* [6] para extração de características do sinal de voz. Esse algoritmo, além de acompanhar as variações estatísticas do sinal de entrada, o que pode ser usado

com uma espécie de “assinatura” de um comando de voz, possui um algoritmo de baixa complexidade computacional, como pode ser observado na Tabela 4.1.

Tabela 4.1 Resumo do algoritmo LMS

Inicialização:
$W(0) = 0$
$\mu << 1$
Em cada iteração:
1. $o(i) = X(i)^T W$
2. $e(i) = d(i) - o(i)$
3. $W(i+1) = W(i) + 2\mu e(i)X(i)$

Nota-se que esse algoritmo pode modelar um sinal qualquer com quantos coeficientes preditores for necessário, já que os mesmos são proporcionais ao número de entradas anteriores do vetor X(i). Sendo assim escolheu-se montar o vetor X(i) com 20 valores anteriores, o que resulta em um filtro de 20 coeficientes, pois como pode ser observado na Figura 4.2, para esse número de valores, o gráfico de taxa de acerto por número de coeficientes do filtro se encontra estabilizado, e isso significa que, acima desse número de parâmetros, o algoritmo não consegue aumentar seu grau de representatividade em relação ao sinal de voz.

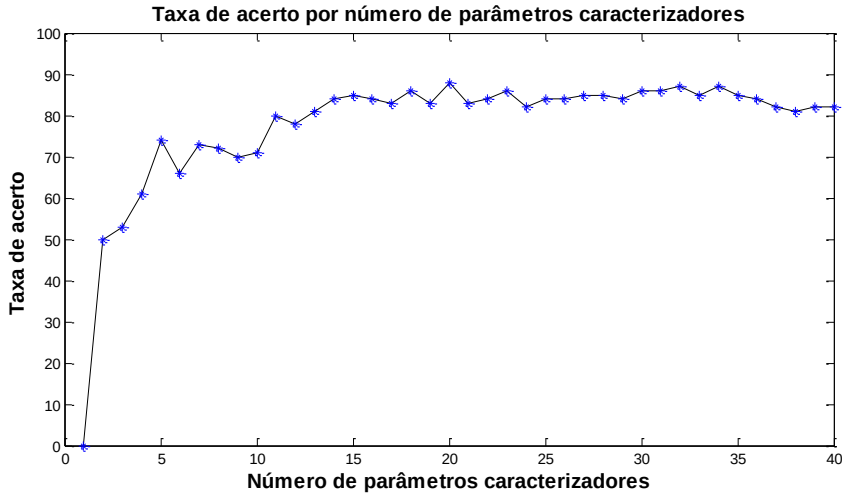

Figura 4.2 Curva de taxa de acerto por número de parâmetros para reconhecimento de cinco comandos.

Na Figura 4.2 cada ponto da curva foi obtido através da resposta da melhor rede neural dentre 300 redes que foram treinadas para cada ponto, totalizando assim 12000

redes distintas treinadas. Cada rede era responsável para classificação de cinco comandos distintos: “direita”, “esquerda”, “siga”, “para” e “trás”.

Com esse algoritmo implementado no MCU, é possível fazer o ajuste dos coeficientes do preditor de forma recursiva para todo o sinal de entrada. E como o objetivo é a extração de característica do comando de voz, passam-se então a entender os coeficientes preditores como parâmetros caracterizadores do comando de fala, ou, simplesmente, parâmetros LMS.

4.4 Comunicação com Computador

Para fazer o algoritmo de classificação de padrões, primeiramente é necessário enviar os parâmetros LMS calculados no MCU para um PC, sendo assim possível analisar os dados e treinar a RNA. Para isso, foi necessário estabelecer uma comunicação do tipo USB entre o MCU e o PC, o que fez o sistema possuir dois modos de operação: o primeiro é o modo de configuração, no qual ao final de cada comando o MCU envia os parâmetros LMS para um PC, e o segundo é o modo de execução, no qual após o cálculo dos parâmetros LMS, o MCU executa o algoritmo de RNA para realizar a classificação.

4.5 Classificação do comando de voz

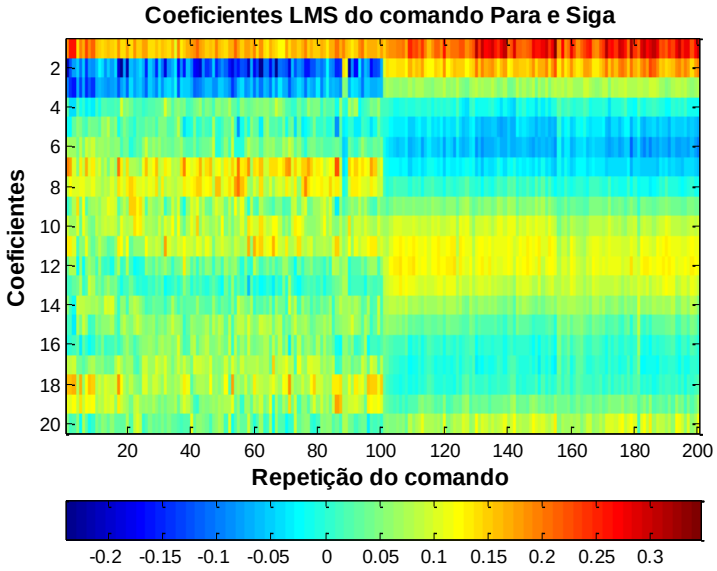

Figura 4.3 Ilustração dos valores dos coeficientes para dois comandos distintos. Os coeficientes do comando "Para" são encontrados nas 100 primeiras colunas enquanto as 100 últimas se referem ao comando "Siga".

De posse do algoritmo de extração de característica do comando de voz fez-se um banco de dados com os parâmetros LMS calculados de 100 repetições do comando "Siga" e mais 100 repetições do comando "Para". Na Figura 4.3 é exibido esse banco de dados na forma de imagem para que se possa perceber a diferença dos valores de cada coeficiente LMS para cada comando.

Foram escolhidos apenas dois comandos para a classificação porque, como pode ser observada na Figura 4.4, a curva de taxa de acerto por número de comandos é decrescente e possui acerto máximo para dois comandos.

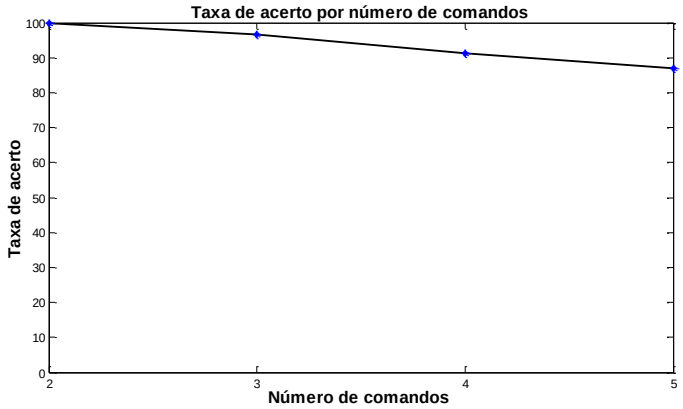

Figura 4.4 Curva de taxa de acerto por número de comandos para extração de 20 coeficientes LMS.

Na Figura 4.4 cada ponto da curva foi obtido pela resposta da melhor rede neural dentre 300 que foram treinadas para cada ponto. O que totalizou 1200 redes treinadas.

Com esse banco de dados treinou-se, em ambiente computacional através do *software* Matlab, duas RNA do tipo *feed-forward backpropagation* com dois neurônios na camada de entrada e um neurônio de saída. A primeira RNA responde nível alto somente no caso de receber coeficientes do comando "Para", e a segunda RNA responde nível alto somente no caso de receber coeficientes do comando "Siga".

Com as redes treinadas gravaram-se os pesos e os *bias* das RNAs na memória de programa do MCU, os quais são utilizados pelo algoritmo de classificação para calcular a respostas das RNAs programadas no MCU. No algoritmo de classificação que foi criado, caso as duas redes respondam nível alto para um mesmo comando, ou as duas respondam nível baixo, o sistema informa, através de um LCD, que o comando não foi identificado.

5 Resultados e discussões

Para testar o algoritmo de cálculo dos coeficientes LMS, gravou-se um vetor de 900 dados na memória de programa do MCU para simular um sinal de voz de entrada e, para esse mesmo sinal, calculou-se 20 coeficientes LMS através do MCU e do Matlab. O resultado é exibido na Tabela 5.1.

Tabela 5.1 Comparação coeficientes calculados pelo MCU e pelo Computador. O desvio médio em relação ao computador foi de 2,36x10-4.

Parâm. MCU	**Parâm. Computador**
0,243832251	0,24366165
0,210463354	0,210280836
0,178023706	0,177828784
0,146889685	0,146683343
0,117326672	0,117110082
0,089655133	0,089429388
0,064149091	0,063915174
0,041058716	0,04081653
0,020617279	0,020368936
0,003030534	0,00277707
-0,01151817	-0,011775755
-0,022913335	-0,023173991
-0,031026355	-0,03128803
-0,035766916	-0,036028619
-0,037080444	-0,037341071
-0,034950558	-0,035209206
-0,029422137	-0,029676632
-0,020543851	-0,020794258
-0,008396764	-0,008641014
0,006868068	0,006631003

Para testar a precisão dos cálculos do algoritmo de classificação foram usados os coeficientes da Tabela 5.1 calculados pelo MCU como entrada desse algoritmo, e foi calculada a resposta das RNAs no MCU e no PC. O resultado do MCU apresentou um desvio de 1,82x10-7 em relação ao computador.

No treinamento da RNA separou-se 50% dos dados para o treino, 20% para a validação e mais 30% para teste. O resultado do treinamento é ilustrado na Figura 5.1. Nota-se que as redes alcançaram um acerto de 100% para os dois comandos, o que é devido à grande diferença entre os coeficientes LMS dos dois comandos como pôde ser visto na Figura 4.3.

De posse da estrutura das redes e dos valores dos pesos, ambos obtidos no Matlab, construíram-se as mesmas redes neurais no MCU, as quais são utilizadas pelo algoritmo que controla o modo de execução. Nesse algoritmo, depois de calculado a resposta da RNA, o mesmo ainda faz um arredondamento dos valores para o inteiro mais próximo.

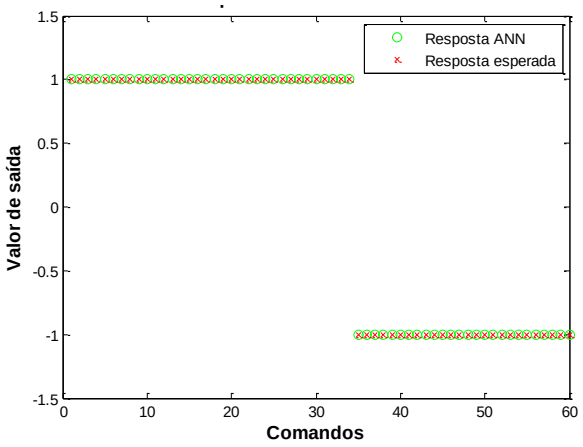

Figura 5.1 Desempenho da ANN implementada em Matlab para o comando “Para” no banco de dados de teste. Nesse caso, a rede deveria apresentar o valor de saída 1 quando parâmetros LMS advindos de um comando "Para" fossem usados como entrada.

Com o protótipo pronto, testaram-se as respostas da rede do MCU e obteve-se um acerto de 100% para classificação de dois comandos de voz. Esses resultados estão consistentes com aqueles obtidos pelo Matlab que estão exibidos na Figura 5.1.

Foi medido, também, o tempo de processamento dos algoritmos de cada etapa de reconhecimento que foram programados no MCU. O resultado está exibido na Tabela 5.2

Tabela 5.2. Tempo de processamento do dispositivo para comando de 500ms de duração.

Tempo de processamento p/ comando de 500ms	
Etapa	**Tempo (s)**
Captação do comando de fala	tempo real
Cálculo dos parâmetros LMS	6,52833
Cálculo da resposta da RNA	0,011

É importante lembrar que os algoritmos implementados no MCU foram programados usando variáveis do tipo Ponto Flutuante (32 bits). Por isso existe diferença entres os valores calculados pelo PC e o MCU, pois em ambiente

computacional os cálculos são feitos com variáveis de 64 bits, o que traz uma maior precisão de cálculo.

Outra ponderação importante é que o desempenho da RNA está diretamente ligado à qualidade dos parâmetros extratores de característica do sinal de fala, sendo assim, justifica-se o acerto de 100% da RNA devido à boa diferenciação entre os comandos de voz promovida pelo alto número de parâmetros LMS calculados. Mas é importante destacar que tal nível de diferenciação foi obtido à custa de um maior tempo gasto no cálculo dos parâmetros LMS, pois o número de parâmetros calculados é diretamente proporcional ao tempo de processamento.

Comparando esse trabalho com aqueles mostrados na sessão 3.3.2, percebe-se que haveria um grande avanço na velocidade de resposta do dispositivo caso fosse utilizado um dispositivo FPGA ao invés de um microcontrolador, uma vez que, utilizando um FPGA, a etapa de extração de características poderia ser processada paralelamente à etapa de conversão A/D, o que resultaria em uma resposta dos parâmetros LMS imediatamente após o final do pronunciamento de um comando de voz, assim o dispositivo demandaria apenas do tempo de cálculo de resposta da RNA para atuar sobre uma saída desejada.

O dispositivo apresenta ainda uma ligeira vantagem em relação a alguns trabalhos executados em FPGA, que é a facilidade de reconfiguração da RNA. Como a RNA é implementada em software, é preciso apenas mudar a matriz de pesos e *bias* gravada na memória de programa do MCU para reconfigurar a RNA, já em um dispositivo FPGA é necessário montar, trabalhosamente, cada um dos neurônios modificados. Porém, essa vantagem passa a não existir quando se tem uma ferramenta como a que é demonstrada em 3.3.2, a qual permite que os usuários descrevam arquiteturas de rede em um alto nível de abstração e gera códigos VHDL sintetizáveis em sistema CAD para FPGAs. Então, para esse caso específico, talvez, a reconfiguração se torne ainda mais fácil que aquela efetuada em software.

6 Conclusões

O reconhecimento de padrões através de um protótipo microcontrolado é uma alternativa de baixo custo, pois, em modo de execução, o mesmo independe de computadores e ainda dispensa componentes dispendiosos como os DSPs. E possui, ainda, um grande potencial de aplicabilidade, pois permite a aplicação dos métodos de RNA em situações em que não seja viável, ou possível, destinar um computador para cumprir essa função.

Outro resultado interessante é a capacidade de se obter um hardware de extração de características de sinais de áudio, com comunicação USB, com um alto nível de representatividade do sinal de entrada a partir do décimo quinto parâmetro calculado, o que pôde ser constatado na Figura 4.3, e com um baixo desvio médio em relação ao PC.

A fim de continuar as pesquisas nessa área de aplicação, seria interessante, para os trabalhos futuros, explorar a extração de caracteristicas de sinais e o reconhecimento de padrões utilizando um dispositivo FPGA, já que, como foi demonstrado, o mesmo possui diversas características que permite um produto final de maior desempenho.

7 Referências Bibliográficas

1. HAYKIN, S. **Redes Neurais Princípios e Práticas**. São Paulo: Prentice Hall, 1999.
2. PINHEIRO, M. Fundamentos de Neuropsicologia - O Desenvolvimento Cerebral, Trindade, 2007. Disponivel em: <http://www.fug.edu.br/revista/artigos/Organizados/desenvolvimentosn.pdf>.
3. ALENCAR, V. F. S. Atributos e domínos de interpolação eficientes em reconhecimento de voz distribuído, 2005.
4. RABINER, L. R.; SCHAFER, R. W. **Digital Processing Of Speech Signals**. [S.l.]: Prentice Hall, 1978.
5. ISHI, C. T. Análise de Um Sistema de Reconhecimento de Voz Baseado em Fonemas, 1998.
6. MARQUES, P. A. C. Introdução à Filtragem Adaptativa. Disponivel em: <http://www.deetc.isel.ipl.pt/sistemastele/STDS/Bibliografia/Texto-FiltragemAdaptativa.pdf>. Acesso em: 15 ago. 2010.
7. YNOGUTI, C. A. Reconhecimento de Fala Contínua Utilizando Modelos Ocultos de Markov, 1999.
8. VILELA, A. L. M. Sistema Nervoso. **Analtomia e Fisiologia Humana**. Disponivel em: <http://www.afh.bio.br/nervoso/nervoso1.asp>. Acesso em: 04 nov. 2010.
9. BRAGA, A. P.; LUDERMIR, T. B.; CARVALHO, A. P. L. F. **Redes Neurais Artificiais Teorias e Aplicações**. Rio de Janeiro: LTC, 2000.
10. INC, T. M. Neural Network Toolbox, User`s Guide for Use with Matlab, 2009.
11. NOORY, B.; GROZA, V. A Reconfigurables Approach To Hardware Implementation Of Neural Networks. **Electrical and Computer Engineering, 2003. IEEE CCECE 2003. Canadian Conference on**, Ottawa, 2003.
12. BECKER, J.; HARTENSTEIN, R. W. Configware and morphware going mainsteam, Journal of System Architecture, 2003.
13. TANENBAUM, A. S. **Organização Estruturada de Computadores**. [S.l.]: Prentice Hall, 1992.

14. BRAGA, A. L. S. VANNGen - Uma Ferramenta CAD Flexível para a Implementação de Redes Neurais Artificiais em Hardware, Brasília, 2005.

15. BRAURER, E. J. et al. Hardware Implementation of a Neural Network Pattern Shaper Algorithm, 1999.

16. REAY, D. S.; GREEN, T. C.; WILLIAMS, B. W. Fild Programmable Gate Array Implementationqof Neural Network Accelerator, 9 Mar 1994.

17. PEREZ-URIBE, A.; SANCHEZ, E. FPGA Implementation of an Adaptable-Size Neural Network, 1996.

18. KUROKAWA, T.; YAMASHITA, H. Bus Connnected Neural Network Hardware System, 1994.

Anexo 1

Algoritmo Least Mean Squares [6]

Este algoritmo utiliza o valor instantâneo de $e^2(i)$ como estimativa da função de custo calculada por $E[e^2(i)]$. O algoritmo é o seguinte:

1. Estima-se o gradiente da função:

$$\hat{\nabla}(i) = \frac{\partial e^2(i)}{\partial W} = -2e(i)X(i)$$

2. Como o gradiente é o vetor que aponta no sentido do máximo da função de custo, desloca-se o vetor de pesos na direção oposta com o objetivo de procurar o mínimo:

$$\widehat{W}(i+1) = \widehat{W}(i) - \mu\hat{\nabla}(i)$$

Ou seja,

$$\widehat{W}(i+1) = \widehat{W}(i) + 2e(i)X(i)$$

A constante μ designa-se habitualmente por passo de adaptação e controla a estabilidade e velocidade de convergência do algoritmo. É demonstrado que há existência de estabilidade do algoritmo desde que o valor do passo de adaptação obedeça à seguinte desigualdade:

$$0 < \mu < \frac{1}{(N+1) * (Potencia\ de\ X(i))}$$

O erro residual será proporcional a μ, sendo por isso normal escolher valores pequenos para esta constante.

Para aumentar a velocidade de convergência é também proposto na bibliografia utilizar μ elevado no início do processo de adaptação, e lentamente diminuí-lo, por forma a conduzir a um valor baixo de erro residual. Em situação de não-estacionaridade o passo muito reduzido levará a baixa capacidade de seguimento das mudanças no ambiente de trabalho.

O algoritmo LMS é atualmente o mais utilizado devido, sobretudo, à sua baixa complexidade.