

**CENTRO UNIVERSITÁRIO EURÍPIDES DE MARÍLIA – UNIVEM
FUNDAÇÃO DE ENSINO “EURÍPIDES SOARES DA ROCHA”
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO**

DANILO COSTA BREDÁ

**AUTOMAÇÃO RESIDENCIAL: APLICAÇÃO DE
REDES NEURÁIS ARTIFICIAIS PARA CONTROLE
DE CLIMATIZAÇÃO.**

**MARÍLIA
2016**

DANILO COSTA BREDA

**AUTOMAÇÃO RESIDENCIAL: APLICAÇÃO DE
REDES NEURAS ARTIFICIAIS PARA CONTROLE
DE CLIMATIZAÇÃO.**

Trabalho de Curso apresentado ao Curso de Bacharelado em Ciência da Computação da Fundação de Ensino “Eurípides Soares da Rocha”, mantenedora do Centro Universitário Eurípides de Marília – UNIVEM, como requisito parcial para obtenção do grau de Bacharel em Ciência da Computação.

Orientador
Prof^ª: Ms. Rodolfo Barros Chiamonte

**MARÍLIA
2016**

BREDA, Danilo Costa

Automação Residencial: Aplicação de Redes Neurais Artificiais para Controle de Climatização. / Danilo Costa Breda; orientador: Prof. Ms. Rodolfo Barros Chiaramonte. Marília, SP: [s.n.], 2016.

Monografia (Trabalho de Conclusão de Curso em Bacharelado em Ciência da Computação) – Curso de Ciência da Computação, Fundação de Ensino “Eurípedes Soares da Rocha”, mantedora do Centro Universitário Eurípedes de Marília – UNIVEM, Marília, 2016.

76 Folhas

Monografia (Bacharelado em Ciência da Computação): Centro Universitário Eurípedes de Marília.



CENTRO UNIVERSITÁRIO EURÍPIDES DE MARÍLIA - UNIVEM
MANTIDO PELA FUNDAÇÃO DE ENSINO "EURÍPIDES SOARES DA ROCHA"


BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

Danilo Costa Breda

Automação residencial: Aplicação de redes neurais artificiais para controle de climatização.


Banca examinadora da monografia apresentada ao Curso de Bacharelado em
Ciência da Computação do UNIVEM/F.E.E.S.R., para obtenção do Título de
Bacharel em Ciência da Computação.

Nota: 10,0 (Dez)

Orientador: Rodolfo Barros Chiamonte 

1º. Examinador: Renata Aparecida de Carvalho

Paschoal 

2º. Examinador: Fabio Piola Navarro 

Marília, 06 de dezembro de 2016.

AGRADECIMENTOS

Agradeço ao professor orientador Rodolfo Chiaramonte, pela confiança passada para a conclusão do trabalho e ajuda no decorrer do ano.

Agradeço ainda o apoio recebido pela minha família, que acreditaram no meu potencial e deram forças no decorrer do ano, em especial meu pai, Milton Breda, pela grande ajuda e força, minha mãe, Claudia Breda, pela força e confiança transmitida e minha irmã Ana Beatriz Costa Breda, por sempre estar do meu lado.

Em especial agradeço a minha namorada Patrícia Pasinato, que esteve sempre presente comigo durante todo ano e me transformou em alguém mais autoconfiante.

Agradeço a todos os meus amigos, em especial Maurilio Matsuyama, por estar sempre ao meu lado me dando forças e sempre verdadeiro nas suas palavras, uma amizade criada no decorrer do curso e que vou levar sem sombra de dúvida, para minha vida toda.

Por fim, a todos que de uma maneira ou outra contribuíram para a realização deste trabalho.

RESUMO

Com o avanço da tecnologia, a automação residencial começou a ser mais familiarizada, pelo seu alto grau tecnológico, possibilitando o controle de dispositivos de forma simples e inteligente, gerando uma melhor resposta ao que a tecnologia pode oferecer. O principal problema abordado neste trabalho é a aplicação de redes neurais artificiais na área de automação residencial, mais especificamente em sistemas de climatização. O trabalho consiste em desenvolver um sistema aplicando redes neurais artificiais de múltiplas camadas, integrado a um simulador, visando resultados para uma melhora no conforto e economia de energia, de forma transparente ao usuário. Os resultados foram obtidos através de simulação e análise, comprovando a viabilidade do uso de redes neurais artificiais no controle de climatização na área de automação residencial.

Palavra-Chave: Inteligência Artificial, Redes Neurais Artificiais, Automação Residencial, Domótica, Climatização.

ABSTRACT

With the advancement of technology, home automation began to be more familiar, due to its high degree of technology, allowing the control of devices in a simple and intelligent way, generating a better response to what technology can offer to us. The main problem addressed in this work is the application of artificial neural networks in the area of home automation, more precisely in climatization systems. The work consists in developing a system applying artificial neural networks of multiple layers, integrated to a simulator, aiming at results for an improvement in comfort and energy saving, transparently for the user. The results were obtained through simulation and analysis, proving the feasibility of the use of artificial neural networks in the control of climatization systems in the area of home automation.

LISTA DE EQUAÇÕES

Equação 1 – Função de ativação de um neurônio	31
Equação 2 – Equação de aprendizado	35
Equação 3 – Fórmula de ajuste de pesos	38
Equação 4 – Fórmula da função de ativação <i>sigmoid</i>	39

LISTA DE FIGURAS

Figura 1 – Visualização do Simulador Home I/O	21
Figura 2 – Alteração de modo do dispositivo no simulador Home I/O.....	22
Figura 3 – Aquecedor do simulador Home I/O	23
Figura 4 – Termostato do simulador Home I/O	23
Figura 5 – Componentes do neurônio biológico	27
Figura 6 – Neurônio de McCulloch e Pitts	31
Figura 7 – Rede <i>feedforward</i> de uma única camada	32
Figura 8 – Rede <i>feedforward</i> de duas camadas	33
Figura 9 – Rede com recorrência entre saídas e camada intermediária	34
Figura 10 – Rede com recorrência auto-associativa.....	34
Figura 11 – Aprendizado supervisionado.....	36
Figura 12 – Aprendizado não-supervisionado.....	37
Figura 13 – Topologia de um perceptron simples com uma única saída	38
Figura 14 – Gráfico de função Sigmoidal	39
Figura 15 – Rede MLP típica com uma camada intermediária	40
Figura 16 – Fluxo de processamento do algoritmo Backpropagation.....	41
Figura 17 – Cômodos para o cenário escolhido em azul no simulador Home I/O.....	45
Figura 18 – Módulos do Sistema Proposto junto a integrações com SDK	47
Figura 19 – Padrão de arquivo de dados utilizado para treinamento da rede neural.....	56
Figura 20 – Demonstração da interação do usuário com o simulador HOME I/O	57
Figura 21 – Treinamento da rede neural através de pontos de dados de treinamento.....	59
Figura 22 – Salvando rede neural treinada no formato MLP.	60
Figura 23 – Carregando rede neural já treinada de arquivo no formato MLP.	60
Figura 24 – Execução da rede neural com o simulador Home I/O em tempo real.....	61
Figura 25 – Tabela de Referência Base de resultados obtidos	62
Figura 26 – Gráfico comparativo kWh de cômodos de mesma temperatura.	63
Figura 27 – Gráfico comparativo kWh de cômodos de temperaturas diferentes	64
Figura 28 – Gráfico comparativo de Desvio Total de temperatura de cômodos de mesma temperatura.....	65
Figura 29 – Gráfico comparativo de Desvio Total de temperatura de cômodos de temperaturas diferentes	65
Figura 30 – Gráfico da série de 5000 iterações	67

Figura 31 – Gráfico da série de 20.000 iterações	67
--	----

LISTA DE ABREVIATURAS E SIGLAS

IOT	Internet of Things – Internet das Coisas
TCP	Transmission Control Protocol - Protocolo de Controle de Transmissão
IP	Internet Protocol - Protocolo de Internet
IA	Inteligência Artificial
RNA	Rede neural artificial
RNAs	Redes neurais artificiais
MLP	Multilayer Perceptron – Perceptron de Múltiplas Camadas
SDK	Software Developer’s Kit
LED	Light-emitting Diode
IPC	Inter-process Communication
KWh	Kilowatt Hora
ABC+	Automação Baseada em Comportamento

SUMÁRIO

1	INTRODUÇÃO	14
1.1	Motivação e Justificativa.....	14
1.2	Objetivos	15
1.3	Estrutura do Trabalho	15
2	AUTOMAÇÃO RESIDENCIAL	17
2.1	Domótica	17
2.2	Evolução e Adoção.....	17
2.3	Ambiente Inteligente	18
2.4	Climatização.....	19
2.5	Internet das Coisas (IOT)	19
2.6	Home I/O.....	20
2.6.1	Conectando com Tecnologias Externas.....	21
2.6.2	Dispositivos Utilizados.....	22
2.6.3	Comportamento Térmico.....	24
2.7	Trabalhos Correlatos	24
3	REDES NEURAIS	26
3.1	Motivação.....	26
3.2	Inspiração Biológica.....	27
3.3	Redes Neurais Artificiais (RNAs)	28
3.4	Histórico	28
3.5	Neurônio Artificial: Modelo MCP	30
3.6	Arquitetura	32
3.7	Aprendizado	35
3.7.1	Aprendizado Supervisionado.....	35
3.7.2	Aprendizado Não-Supervisionado.....	36
3.8	Perceptron.....	37
3.9	Rede Perceptron de Múltiplas Camadas.....	38
3.10	Backpropagation.....	40
3.11	Encog.....	42
3.12	Trabalhos Correlatos	42
4	DESENVOLVIMENTO	44
4.1	Ferramentas Utilizadas	44

4.2	Definição de Cenário no Home I/O.....	45
4.3	Implementação do Sistema.....	46
4.3.1	Módulo de Integração.....	47
4.3.2	Módulo de Tomada de Decisão.....	50
4.3.2.1	Configuração.....	50
4.3.2.2	Treinamento.....	52
4.3.2.3	Validação.....	53
4.3.3	Módulo de Interface do Usuário.....	53
5	SIMULAÇÃO E RESULTADOS OBTIDOS.....	56
5.1	Geração de Dados para Treinamento.....	56
5.1.1	Captura via Simulador Home I/O.....	57
5.1.2	Geração de Dados Virtuais.....	58
5.2	Cenário de Treinamento.....	58
5.3	Simulação com o Home I/O.....	60
5.4	Resultados Obtidos.....	61
5.4.1	Dados de Referência.....	62
5.4.2	Avaliação da Rede Neural.....	63
5.4.3	Avaliação do Número de Neurônios na Camada Intermediária.....	66
6	CONCLUSÃO.....	69
6.1	Considerações Finais.....	69
6.2	Trabalhos Futuros.....	69
	REFERÊNCIAS.....	70
	APÊNDICES.....	73

1 INTRODUÇÃO

Com o avanço da tecnologia e a necessidade de auxílio de tarefas aos seres humanos, de modo a demandar menos tempo e proporcionar uma maior sensação de conforto, a automação residencial começou a ser mais familiarizada, possibilitando uma melhor resposta ao que a tecnologia pode nos oferecer perante a complexidade do mundo em que vivemos.

A climatização em ambientes residenciais é feita geralmente de sistemas de ar condicionado, sistemas de aquecimento ou sistemas de ventilação, existindo a necessidade de sua implantação em locais onde a temperatura não permanece na faixa de temperatura simpática dos seres humanos.

Na automação residencial, a área de climatização existe para facilitar a interação com esses equipamentos, permitindo um nível maior de conforto, diminuição de consumo de energia, desativação de equipamentos em casos de emergência e outras funções.

O trabalho consiste em desenvolver um sistema que consiga, junto com o uso de redes neurais artificiais, o controle de climatização de uma residência, através de sensores e atuadores, visando resultados que melhorem o conforto do usuário de forma transparente e também trazendo benefícios como economia de energia.

1.1 Motivação e Justificativa

Com a evolução da tecnologia, áreas da automação residencial tende a inovação, principalmente pela adoção dos usuários de novas tecnologias e as possibilidades que elas podem nos oferecer. De um modo geral, a sociedade tende à necessidade de aperfeiçoar suas tarefas cotidianas, de modo há demandar menos tempo e proporcionar uma sensação maior de conforto, segurança e bem-estar.

De acordo com estudo do PROCEL (Programa Nacional de Conservação de Energia Elétrica), o ar condicionado esta entre os grandes consumidores de energia elétrica das residências.

Em países onde sistemas de climatização são essenciais para a sobrevivência, o consumo de energia desses sistemas é bastante elevado, destacando a necessidade de economia de energia com base na preocupação que hoje temos com o nosso planeta e o meio ambiente.

A escolha deste tema se dá pela necessidade da criação de tecnologias mais eficazes

e inteligentes, transparentes ao usuário, embasado em estudos de fundamentação teórica, sendo esse trabalho de grande importância para desenvolvimentos futuros na área de automação residencial e climatização.

1.2 Objetivos

O objetivo geral desse trabalho é desenvolver um sistema de controle de climatização, aplicando redes neurais artificiais na área de automação residencial visando transparência ao usuário, conforto e economia de energia.

Para isso foram definidos alguns objetivos específicos, que são apresentados na sequência:

- Apresentação de um modelo de rede neural para solução de um sistema de controle de climatização de um determinado cenário.
- Escolha de ferramenta de simulação de climatização.
- Modelagem de um cenário na ferramenta de simulação.
- Desenvolvimento e integração de uma rede neural com a ferramenta de simulação.
- Simulação e geração de informações para análise de resultado.
- Conclusão através de análise dos resultados obtidos.

1.3 Estrutura do Trabalho

Capítulo 1 – Introdução: É apresentada a necessidade de inovação do mercado diante as tecnologias atuais na área de automação residencial e climatização, criando novas possibilidades e enfatizando o que a tecnologia nos pode oferecer no futuro.

Capítulo 2 – Automação Residencial: São apresentados neste capítulo, conceitos sobre automação residencial, sua evolução desde seu principio até os dias atuais e tendências. Também é apresentado a ferramenta de simulação Home I/O que será utilizada durante todo o trabalho.

Capítulo 3 – Rede Neural: Neste capítulo é conceituado as redes neurais, sua definição, origem, historia, funcionamento e principais características, com destaque em MCP e MLP.

Capítulo 4 – Desenvolvimento: Todo o desenvolvimento do sistema é realizado

nesse capítulo. Com ele, será possível a realização da simulação para aquisição de resultados.

Capítulo 5 – Simulação e resultados obtidos: É feito a simulação do sistema desenvolvido e resultados são obtidos e analisados.

Capítulo 6 – Conclusão: Através dos resultados obtidos é feita a conclusão.

2 AUTOMAÇÃO RESIDENCIAL

A automação residencial ou domótica é o uso da tecnologia através de serviços proporcionados por sistemas tecnológicos para tornar automática e facilitar tarefas habituais a modo de garantir segurança, comunicação, gestão energética e conforto de uma habitação. Ela é composta a partir de equipamentos com capacidade de comunicação e atuação, possibilitando interações com o usuário de uma forma tecnológica e transparente.

2.1 Domótica

O termo domótica vem da fusão da palavra latina *domus* (casa) e robótica. É o termo mais utilizado na Europa por conta de sua abrangência.

No Brasil, o termo mais utilizado é automação residencial, por conta de sua tradução literal americana de “home automation”. O termo domótica é mais abrangente, pois “automação” não englobaria sistemas de comunicação ou sonorização.

Existem algumas expressões para o termo domótica, por exemplo, “smart building”, “intelligent building”, “edifícios inteligentes” e “casa inteligente” (BOTELHO, 2005).

2.2 Evolução e Adoção

A década de 1970 segundo Sena (2005), é considerada um marco importante na história da domótica, quando são lançados nos EUA os primeiros módulos inteligentes de automação residencial, os chamados X-10®. O protocolo X-10® é uma linguagem de comunicação que admite que produtos compatíveis se comuniquem entre si através da linha elétrica existente. Ou seja, é uma tecnologia sem novos fios. No mercado, existe uma gama enorme de produtos X-10®, de diferentes fabricantes. Pela sua característica básica, o sistema X-10® é indicado para aplicações autônomas, não unificadas. Uma de suas restrições é de operar apenas funções simples como liga/desliga. O sistema posteriormente se viu instável.

Na década de 1980 conforme Sena (2005), com a evolução da informática e o surgimento de interfaces amigáveis e operações extremamente fáceis, novas possibilidades de automação passam a existir no mercado. Entretanto, é no fim da década de 90 que surge uma grande variedade de dispositivos para o mercado de automação residencial. Algumas inovações tecnológicas incorporadas ao nosso dia a dia, como o telefone celular e a Internet, despertaram no consumidor o gosto pelas versatilidades e facilidades que proporcionam.

Segundo Godoi (2009), cronologicamente, o desenvolvimento dos sistemas de automação residencial surge depois de seus similares nas áreas industrial e comercial. Por óbvios motivos econômicos e de escala de produção, os fabricantes e os prestadores de serviços, num primeiro momento, se voltam a aqueles segmentos que lhes propiciam maior rapidez no retorno de seus investimentos. No mercado brasileiro isto ocorreu de maneira similar, os primeiros sistemas automatizados de controle foram concebidos para aplicações especificamente industriais, ainda na década de 70.

No Brasil, a absorção de novas tecnologias pelos usuários em sua vida diária é alta, porém a mesma tendência não se aplica na automação residencial, onde existe a necessidade de uma adoção no mercado de construção civil.

A adoção de usuários com necessidades específica que englobam a automação residencial, tem sido um fator fundamental para o impulsionamento de inovação do setor imobiliário no Brasil a níveis internacionais.

2.3 Ambiente Inteligente

Sistemas de automação residencial têm a capacidade de tornar ambientes mais inteligentes, através das suas fortes características de executar comandos e funções, e a integração com outros sistemas de maneira transparente e inteligente.

Segundo Sena (2005), um ambiente inteligente é aquele que aperfeiçoa certas funções inerentes à operações e administração de uma residência ou edifício. Estabelecendo uma analogia com um organismo vivo, a residência moderna parecerá ter vida própria, com cérebro e sentidos.

De acordo com Sena (2005), as principais características fundamentais que devemos encontrar em um sistema inteligente são:

- Capacidade para integrar todos os sistemas – os sistemas interligados por meio da rede doméstica devem possibilitar o monitoramento e o controle externo, bem como atualização remota de software e detecção de falhas;
- Atuação em condições variadas – o sistema deve ser capaz de operar em condições adversas (clima, vibrações, falta de energia) e prover múltiplas interfaces para os diferentes usuários, segundo o entendimento tecnológico, idade, etc., bem como auxiliar portadores de deficiência;
- Memória – o sistema deve ser capaz de memorizar suas funções principais mesmo em regime de falta de energia, deve possibilitar a criação de um

histórico das últimas funções e prover meio de checagem e auditoria destas funções;

- Noção temporal – o sistema deve ter a noção de tempo, bem como dia e noite e estações climáticas a fim de possibilitar a execução de processos e atividades baseadas nestes aspectos;
- Fácil relação com o usuário – o sistema deve prover interfaces de fácil acesso e usabilidade, pois os usuários detêm diferentes níveis de instrução e entendimento sobre novas tecnologias;
- Facilidade de reprogramação – o sistema deve permitir a fácil reprogramação dos equipamentos e prover ajustes pré-gravados em casos de falha ou mau funcionamento;
- Capacidade de autocorreção – o sistema deve ter a capacidade de identificar uma seleção de problemas e sugerir soluções.

2.4 Climatização

O controle de climatização em sistemas de automação residencial é geralmente composto por equipamentos de climatização, sensores e sistemas centrais de controle. Sistemas inteligentes oferecem vantagens exclusivas para esse tipo de sistema como:

- Conforto – diante da sua facilidade de operação, sendo a interação do usuário para acionamento do sistema ou configuração, quase nula;
- Economia de energia – Funcionamento do sistema de acordo com horários e presença do usuário e monitoramento de temperatura externa, permitindo o acionamento/desligamento do sistema de acordo com as características do meio;
- Segurança – desligamento dos sistemas em caso de emergência como incêndio ou em caso de esquecimento.

2.5 Internet das Coisas (IOT)

Segundo Zambarda(2014), a ideia de conectar objetos é discutida desde 1991, quando a conexão TCP/IP(Protocolo de Controle de Transmissão/Protocolo de Internet) e a Internet, como é conhecida hoje, começou a se popularizar. Em 1999, Kevin Ashton do MIT

(Instituto de Tecnologia de Massachusetts) propôs o termo “Internet das Coisas” e dez anos depois escreveu o artigo “A Coisa da Internet das Coisas” para o RFID Journal. Segundo o autor a “Internet das Coisas” se refere a uma revolução tecnológica que em breve conectará equipamentos como eletrodomésticos, meios de transporte, roupas e maçanetas conectadas à Internet e a outros dispositivos, como computadores e smartphones.

Segundo Ashton (2015), será possível acumular dados do movimento do corpo humano com uma precisão muito maior do que as informações obtidas hoje. Com esses registros, será possível reduzir, otimizar e economizar recursos naturais e energéticos, por exemplo. Para o especialista, essa revolução será maior do que o próprio desenvolvimento do mundo online que conhecemos hoje.

Para Atzori et al. (2010), atualmente, a Internet das Coisas (IOT) vem ganhando grande destaque no cenário das telecomunicações e está sendo considerada a revolução tecnológica que representa o futuro da computação e comunicação.

A Internet das Coisas esta totalmente ligada à automação residencial, por conta principalmente de sua interação próxima ao usuário, permitindo com que informações ricas do usuário sejam utilizadas para ajudar no processo de tomada de decisão em sistemas inteligentes.

2.6 Home I/O

O projeto utilizou como ferramenta de simulação o Home I/O, que é um sistema educacional que simula uma casa inteligente, criada para suprir um grande número de áreas como Ciência, Tecnologia, Engenharia e Matemática. O simulador demonstrado na Figura 1 permite aplicar conceitos de automação residencial, através da interação de sensores e dispositivos, como termostatos e aquecedores, para com o exemplo, controle de climatização.

Todo o sistema é integrado com um SDK (Software Development Kit), o que permite com que ferramentas de terceiro utilizem de dados fornecidos pelo simulador.

Figura 1: Visualização do Simulador Home I/O



Fonte: elaborado pelo autor.

2.6.1 Conectando com Tecnologias Externas

Todos os dispositivos da casa com exceção dos dispositivos de controle remoto podem ser utilizados em três diferentes modos:

- *Wired Mode*: A casa não está automatizada, é o padrão de instalação inicial do dispositivo, apenas ligado à tomada;
- *Wireless Mode*: Usado junto ao console de automação residencial. Esse modo permite com que sejam programados para automação da casa alguns dispositivos através de um *tablet* virtual;
- *External Mode*: Permite com que cada dispositivo e pontos de entrada e saída sejam acessados via SDK ou por programas externos.

Todos os dispositivos foram colocados no modo *External Mode*, como mostrado na Figura 2. Todos os dispositivos nesse modo são automaticamente detectados por sistemas integrados ao SDK.

Figura 2: Alteração de modo do dispositivo no simulador Home I/O



Fonte: elaborado pelo autor.

O SDK do Home I/O foi feito através da plataforma .NET 2.0, na qual permitem com que códigos customizáveis acessem os pontos de entrada e saída do simulador Home I/O. A comunicação entre processos (IPC) é feita através de arquivo mapeado em memória. (HOMEIO, 2016).

Segundo HOMEIO (2016), os mapeamentos de memória dos dispositivos estão divididos em três grupos:

- *Inputs*: Dispositivos do tipo de entrada;
- *Outputs*: Dispositivos do tipo de saída;
- *Memories*: Parâmetros em memória.

Também segundo HOMEIO (2016), cada dispositivo contido no mapeamento de memória tem as seguintes propriedades:

- *Memory Address*: Endereço de memória;
- *Name*: Nome do dispositivo;
- *Value*: Valor atual do dispositivo, sendo ele de vários tipos (*Bit, Byte, Short, Int, Long, Float, Double, String, DateTime, TimeSpan*).

2.6.2 Dispositivos Utilizados

O simulador disponibiliza 174 dispositivos comuns para interação, simulação em tempo real. Todos podem ser utilizados para automatizar a casa para através de pontos de entrada e saída (*input/output*).

No projeto proposto foram utilizados aquecedores e termostatos demonstrados nas Figuras 3 e 4 respectivamente.

Figura 3: Aquecedor do simulador Home I/O



Fonte: elaborado pelo autor.

Figura 4: Termostato do simulador Home I/O



Fonte: elaborado pelo autor.

A Figura 3 apresenta o aquecedor que permite com que ocorra a dispersão de calor para o cômodo em que ele está localizado. No modo *Wired Mode*, o sistema funciona em conjunto ao termostato, na qual inclui um LED que indica seu estado, ligada ou desligada.

A Figura 4 demonstra o termostato e suas possíveis opções de interação, que são utilizados para controlar a temperatura do cômodo, ele inclui dois modos, econômico e conforto, o que permite com que sejam guardados dois pontos de temperatura. As opções de interação mostradas na Figura 4 estão definidas a seguir:

- 1 - Liga e desliga o termostato;
- 2 - Modo Econômico;
- 3 - Modo conforto;
- 4 - Aumentar a temperatura;
- 5 - Diminui a temperatura.

O termostato possui um sensor de temperatura interno a ele, para acesso de temperatura do ambiente no momento atual.

Em cada dispositivo no simulador Home I/O, contém um endereço único, assim como o grupo que ele pertence. Nos capítulos seguintes, esses parâmetros serão discutidos, pois serão importantes para a identificação de cada dispositivo e o funcionamento do sistema a ser desenvolvido para o momento de simulação e teste.

2.6.3 Comportamento Térmico

O Home I/O inclui um modelo simplificado de transferência de calor que simula o comportamento térmico da casa. Ele aproxima o fenômeno de radiação, convecção e condução, considerando propriedades físicas dos materiais da casa.

O modelo é também afetado por condições do clima, localização geográfica e distúrbios gerados por correntes de ar entre os cômodos e pelo ambiente externo da casa.

As variáveis que são utilizados no modelo são as seguintes:

- Temperatura externa do ar: Afeta a temperatura de dentro da casa, janelas e portas são os maiores agentes sobre a alteração da temperatura interna pela externa;
- Velocidade do vento: Ventos aumentam o efeito causado pela temperatura do ar externo;
- Direção do vento: Cômodos que estão direcionados a favor do vento sofrem mais efeito sobre a temperatura externa;
- Nebulosidade: Diminui efeitos causados pela radiação do sol;
- Umidade: Altera o ponto de condensação da água, na qual muda como a temperatura do ar externo afeta a casa.

2.7 Trabalhos Correlatos

Após análise e estudo de trabalhos de pesquisa semelhantes ao projeto, na área de automação residencial, destaca-se um projeto correlato com o título “DOMÓTICA INTELIGENTE: AUTOMAÇÃO RESIDENCIAL BASEADA EM COMPORTAMENTO”, do autor Julio André Sgarbi, 2007. Segundo o autor (SGARBI, 2007), o principal problema abordado nesse projeto, foi desenvolver um sistema simples e amigável que abrangesse as

várias particularidades envolvidas na automação inteligente de uma residência, tais como as sequências causais de eventos, que geram regras indesejáveis e a inserção de novas regras para os habitantes, sem causar desconforto aos mesmos, além de abordar diferentes perfis de habitantes e ambientes.

O autor apresentou um sistema proposto ABC+ (Automação Baseada em Comportamento), que foi desenvolvido para observar e aprender regras em uma casa de acordo com o comportamento de seus habitantes, utilizando o conceito de aprendizado com regra de indução.

O projeto em questão serviu como base sustentável para o desenvolvimento desse trabalho.

3 REDE NEURAL

Dentre as ferramentas que constituem a IA (Inteligência Artificial) destacam-se os Algoritmos Genéticos, Lógica Difusa e Redes Neurais Artificiais (FILIPATTI F., et al., 2000), tendo este trabalho, enfoque em Redes Neurais Artificiais (RNAs).

A computação neural surgiu inspirada no funcionamento do cérebro. O termo neural está relacionado com as redes estudadas na neurociência, as quais serviram como motivação para os modelos atualmente utilizados (HERTZ et al., 1991).

Este capítulo irá apresentar conceito sobre Redes Neurais e Redes Neurais Artificiais, o foco da pesquisa nessa área está ligada a capacidade de aprender por meio de exemplos e de generalizar a informação aprendida.

3.1 Motivação

De acordo com Braga, Carvalho e Ludemir (2007), por não se basear em regras, a computação neural se constitui em uma alternativa à computação algorítmica convencional.

A solução de problemas por meio de RNAs é bastante atrativa, já que a forma como estes são representados internamente pela rede e o paralelismo natural inerente à arquitetura das RNAs criam a possibilidade de um desempenho superior ao dos modelos convencionais. Em RNAs, o procedimento usual na solução de problemas passa inicialmente por uma fase de aprendizagem, em que um conjunto de exemplos é apresentado para a rede, a qual extrai as características necessárias para representar a informação fornecida. Essas características são utilizadas posteriormente para gerar respostas para o problema (BRAGA, CARVALO, LUDEMIR, 2007)

A capacidade de aprender por meio de exemplos e de generalizar a informação aprendida é sem dúvida, o atrativo principal da solução de problemas por meio de RNAs. A generalização que está associada à capacidade da rede neural aprender por meio de um conjunto reduzido de exemplos e posteriormente dar respostas coerentes para dados não conhecidos, é uma demonstração de que as RNAs são capazes de extrair informações não apresentadas de forma explícita através dos exemplos. Não obstante, as RNAs são capazes de atuar como mapeadores universais de funções multivariáveis. Outra característica importante é a capacidade de auto-organização e de processamento temporal que, aliada àquelas citadas anteriormente, faz das RNAs uma ferramenta computacional atrativa para a solução de problemas complexos. (BRAGA, CARVALO, LUDEMIR, 2007).

3.2 Inspiração Biológica

Um dos principais fatores que leva ao estudo das RNAs é o fato de o cérebro humano realizar um grande número de tarefas, como reconhecimento de imagens e voz, com uma capacidade muito superior a de um computador. (VALIATI, 2000).

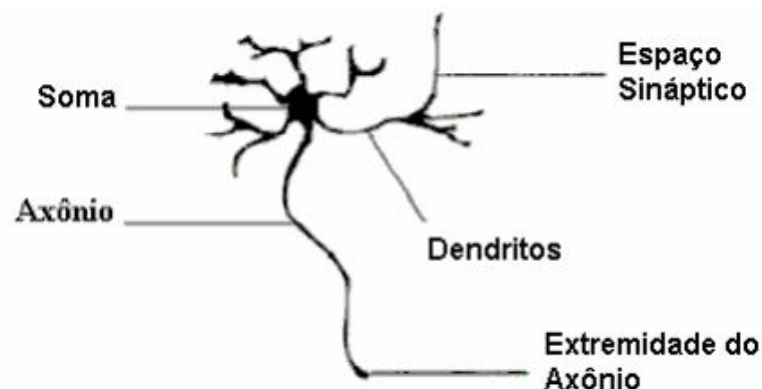
O cérebro humano contém em torno de 10^{11} neurônios, sua célula fundamental. Cada um desses neurônios processa e se comunica com milhares de outros continuamente e em paralelo. A estrutura individual desses neurônios, a topologia de suas conexões e o comportamento conjunto desses elementos de processamento naturais forma a base para o estudo de RNAs (BRAGA, CARVALO, LUDEMIR, 2007).

Outra característica relevante segundo Kovács (1996) é que o cérebro humano possui um número estimado de 10^{11} a 10^{14} neurônios, cada um com cerca de 10^3 à 10^4 conexões, representando uma unidade completa de computação. Lembrando que os computadores tradicionais possuem apenas uma única unidade de processamento central.

Os neurônios são divididos em corpo celular, dendritos e o axônio. Cada um é responsável por uma função específica, porém complementares. O corpo celular chamado de soma mede apenas alguns milímetros, e os dendritos, poucos milímetros de comprimento. O axônio é mais longo e tem calibre uniforme. Os dendritos têm por função receber as informações ou impulsos nervosos, provenientes de outros neurônios e conduzi-las até o corpo celular, onde a informação é processada, produzindo novos impulsos. Esses impulsos são transmitidos através do axônio até os dendritos de neurônios seguintes. É pelas sinapses que os neurônios se unem funcionalmente, formando as redes neurais biológicas.

A Figura 5 apresenta, de forma simplificada, um modelo do neurônio biológico.

Figura 5: Componentes do neurônio biológico;



Fonte: Braga, et al.(2007).

3.3 Redes Neurais Artificiais (RNAs)

RNAs (Redes Neurais Artificiais) são sistemas paralelos distribuídos, compostos por unidades de processamento simples (neurônio artificiais) que calculam determinadas funções matemáticas (normalmente não-lineares). Tais unidades são dispostas em uma ou mais camadas e interligadas por um número de conexões, geralmente unidirecionais. Na maioria dos modelos essas conexões estão associadas a pesos, os quais armazenam o conhecimento adquirido pelo modelo e servem para ponderar a entrada recebida por cada neurônio da rede (BRAGA, CARVALO, LUDEMIR, 2007).

Segundo Kohonen (1972, apud FERNANDES, 2003), as RNAs são definidas como redes massivamente paralelas e interconectadas, de elementos simples, com organização hierárquica. Estes elementos devem interagir com objetivos do mundo real, da mesma maneira que o sistema nervoso biológico.

3.4 Histórico

O trabalho em redes neurais artificiais, usualmente denominadas “redes neurais”, tem sido motivado desde o começo pelo reconhecimento de que o cérebro humano processa informações de uma forma inteiramente diferente do computador digital convencional (HAYKIN 2001).

O primeiro modelo artificial de um neurônio biológico foi fruto do trabalho pioneiro de Warren McCulloch e Walter Pitts, em 1943 (MCCULLOCH; PITTS, 1943).

Segundo HAYKIN (2001), no seu clássico artigo, McCulloch e Pitts descrevem um cálculo lógico das redes neurais que unifica os estudos de neurofisiologia e da lógica matemática. Eles assumiram que o seu modelo formal de um neurônio seguia uma lei “tudo ou nada”. Com um número suficiente dessas unidades simples (neurônios) e com conexões sinápticas ajustadas apropriadamente e operando de forma síncrona, McCulloch e Pitts mostraram que uma rede assim constituída realizaria, a princípio, a computação de qualquer função computável. Este era um resultado muito significativo e com ele é geralmente aceito o nascimento das disciplinas de Redes Neurais Artificiais e inteligência artificial.

O próximo desenvolvimento significativo de RNAs veio em 1949, segundo Braga, et al.(2000) o primeiro trabalho demonstrando ligações direta com o aprendizado de redes artificiais foi apresentado por Donald Hebb (1949). Hebb mostrou como a plasticidade da aprendizagem de redes neurais é conseguida através da variação dos pesos de entrada dos

neurônios. Ele propôs uma teoria para explicar o aprendizado em neurônios biológicos baseada no esforço das ligações sinápticas entre neurônios excitados. A regra de Hebb, como é conhecida a sua teoria na comunidade de RNAs, foi interpretada do ponto de vista matemático e é hoje utilizada em vários algoritmos de aprendizado. Mais tarde, Widrow e Hoff sugeriram uma regra de aprendizado, conhecida como regra de Widrow-Hoff, ou regra delta, que é ainda hoje bastante utilizada.

Em 1958, Frank Rosenblatt demonstrou com o seu novo modelo, o *perceptron*, que se fossem acrescentadas de sinapses ajustáveis, as RNAs com nodos MCP (McCulloch e Pitts) poderiam ser treinadas para classificar certos tipos de padrões. Rosenblatt descreveu uma topologia de RNAs, estrutura de ligação entre os nodos e, o mais importante, propôs um algoritmo para treinar a rede para executar determinados tipos de funções. O *perceptron* descrito por Rosenblatt possui três camadas: a primeira recebe as entradas do exterior e tem conexões fixas (retinas); a segunda recebe impulsos da primeira por meio de conexões cuja eficiência de transmissão (peso) é ajustável e, por sua vez, envia saídas para a terceira camada (resposta). Este tipo de *perceptron* comporta-se como um classificador de padrões, dividindo o espaço de entrada em regiões distintas para cada uma das classes existentes. O que Rosenblatt buscava era projetar RNAs que fossem capazes de fazer descobertas interessantes sem necessidade de regras (BRAGA, CARVALO, LUDEMIR, 2007).

Também nos anos 60, Windrow e Hoff (1996, apud VALLE FILHO, 2003) desenvolveram o ADALINE (ADaptative LINear Element) e o MADALINE (Manu ADALINE) para o reconhecimento de padrões. O algoritmo de aprendizagem de Windrow é conhecido como Regra Delta (FERNANDES, 2003).

Em 1969, segundo Braga, et al.(2007) Minsky e Papert chamaram a atenção para algumas tarefas que o *perceptron* de uma única camada descrito por Rosenblatt não era capaz de executar. O *perceptron*, por exemplo, não consegue detectar paridade, conectividade e simetria, que são problemas não - linearmente separáveis. Seus principais argumentos eram que o problema do crescimento explosivo, tanto de espaço ocupado como do tempo requerido para a solução de problemas complexos, grandes obstáculos que a escola clássica de inteligência artificial já enfrentava, afetaria, cedo ou tarde, as RNAs, inclusive o *perceptron*.

Nos anos de 1970, a abordagem conexionista teve um período de desuso, apesar de alguns poucos pesquisadores continuarem trabalhando na área.

Em 1982 John Hopfield publicou um artigo que chamou a atenção para propriedades associativas das RNAs. Este artigo foi responsável por parte da retomada da pesquisas na área. O grande feito de Hopfield foi, sem dúvida, mostrar a relação entre redes recorrentes

auto-associativas e sistemas número, o que também abriu espaço para as teorias correntes da Física para estudar tais modelos. Não obstante, a descrição do algoritmo de treinamento *Backpropagation* alguns anos mais tarde mostrou que a visão de Minsky e Papert sobre o *perceptron* era bastante pessimista. As RNAs de múltiplas camadas são, sem dúvidas, capazes de resolver “problemas difíceis de aprender” (BRAGA, CARVALO, LUDEMIR, 2007).

O final da década de 1980 marcou o ressurgimento da área de Redes Neurais Artificiais (RNAs), também conhecidas como conexionismo ou sistemas de processamento paralelo e distribuído. Essa forma de computação não-algorítmica é caracterizada por sistemas que, em algum nível, relembram a estrutura do cérebro humano (BRAGA, CARVALO, LUDEMIR, 2007).

Segundo Braga, et al.(2007), a partir de meados da década de 1980, houve nova explosão de interesse pelas RNAs na comunidade internacional. Dentre os fatores responsáveis estão: em primeiro lugar, o avanço da tecnologia, sobretudo da microeletrônica, que permitiu a realização física de modelos de neurônios e suas interconexões de um modo antes impensável; em segundo, o fato de a escola de IA simbólica, a despeito de seu sucesso na solução de determinados tipos de problemas, não ter conseguido avanços significativos na resolução de alguns problemas simples para um ser humano.

Para Braga, et al.(2007), o conhecimento atual que se tem até o momento são teorias, as quais são reavaliadas a cada nova descoberta. No entanto, o comportamento individual dos neurônios biológicos é bem entendido do ponto de vista funcional, e é exatamente nesse comportamento conhecido que se baseiam as RNAs.

3.5 Neurônio Artificial: Modelo MCP

O modelo de neurônio artificial proposto por McCulloch e Pitts é uma simplificação do que se sabia na época a respeito do neurônio biológico. Sua descrição matemática resultou em um modelo com n terminais de entrada (dendritos) que recebem os valores x_1, x_2, \dots, x_n (que representam as ativações dos neurônios anteriores) e apenas um terminal de saída y (representando o axônio). Para representar o comportamento das sinapses, os terminais de entrada do neurônio tem pesos acoplados w_1, w_2, \dots, w_n , cujos valores podem ser positivos ou negativos, dependendo de as sinapses correspondentes serem inibitórias ou excitatórias. O efeito de uma sinapse particular i no neurônio deve considerar sinais de disparo que ocorrem naquela conexão (BRAGA, CARVALO, LUDEMIR, 2007).

Um neurônio biológico dispara quando a soma dos impulsos que ele recebe

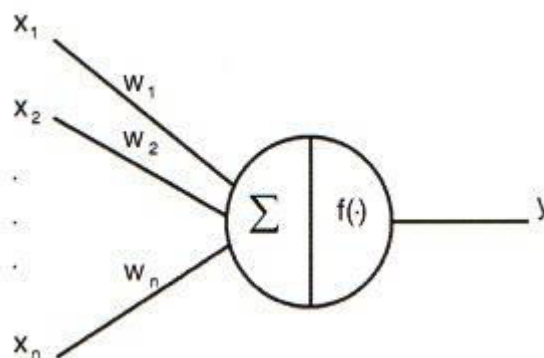
ultrapassa o seu limiar de excitação (*threshold*). Esse comportamento do neurônio biológico, por sua vez, é representado no modelo artificial por um mecanismo simples, que faz a soma dos valores $x_i w_i$ recebidos pelo neurônio (soma ponderada) e decide se o neurônio deve ou não disparar (saída igual a 1 ou 0), comparando a soma obtida ao limiar ou threshold do neurônio. No modelo MCP, a ativação do neurônio é obtido através da aplicação de uma “função de ativação”, que ativa ou não a saída, dependendo do valor da soma ponderada das suas entradas (BRAGA, CARVALO, LUDEMIR, 2007).

A Figura 6 representa o modelo MCP, nele, o disparo do neurônio ocorre quando a soma ponderada de suas entradas atinge um limiar θ , conforme a Equação 1.

Equação 1:

$$\sum_i^n x_i w_i \geq \theta$$

Figura 6: Neurônio de McCulloch e Pitts;



Fonte: Braga, et al.(2007)

Na Figura 6, Σ representa a soma ponderada das entradas e $f(\cdot)$ a função de ativação.

No modelo MCP, as seguintes simplificações são estipuladas (MCCULLOCH; PITTS, 1943):

- A atividade do neurônio é um processo binário;
- Um número fixo de sinapses deve ser excitado dentro do período de repouso a fim de excitar um neurônio em qualquer tempo, e esse número é independente da atividade anterior e posição do neurônio;
- O único atraso significativo no sistema nervoso é o atraso sináptico;

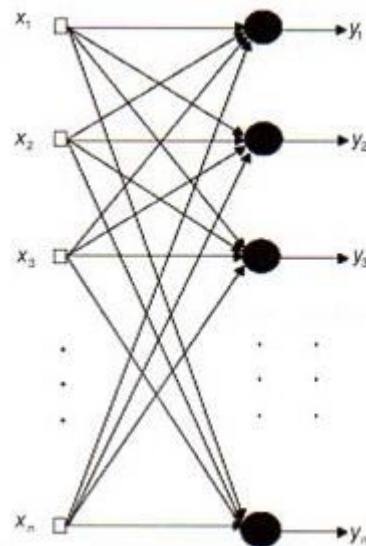
- A atividade de qualquer sinapse inibitória previne completamente a excitação do neurônio naquele período;
- A estrutura da rede não muda com o tempo.

3.6 Arquitetura

Segundo Braga, et al.(2007), todos os neurônios individuais possuem capacidade computacional limitada, independente da função de ativação escolhida. No entanto, um conjunto de neurônios artificiais conectados na forma de uma rede, é capaz de resolver problemas de complexidade elevada.

A Figura 7 apresenta a estrutura mais simples, que corresponde a uma rede neural de camada única alimentada para frente (*feedforward*). Estruturas como essa são capazes de resolver problemas multivariáveis de múltiplas funções acopladas, mas com algumas restrições de complexidade, por serem de uma única camada.

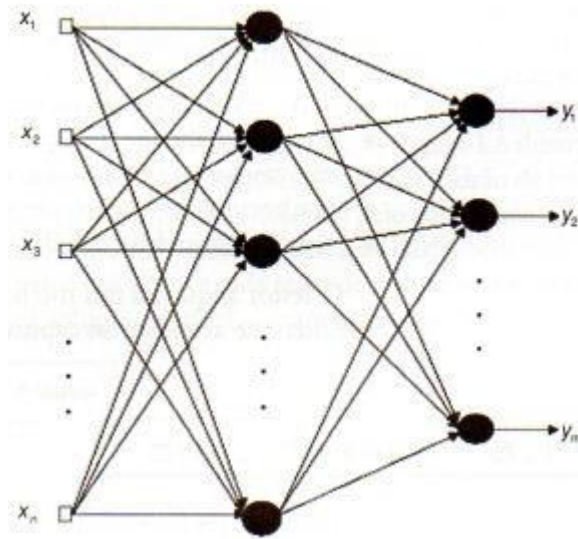
Figura 7: Rede feedforward de uma única camada;



Fonte: Braga, et al.(2007).

A Figura 8 é semelhante a Figura 7, com a diferença de que possui uma camada adicional. A camada intermediária confere à RNA uma maior capacidade computacional e universalidade na aproximação de funções contínuas.

Figura 8: Rede feedforward de duas camadas;

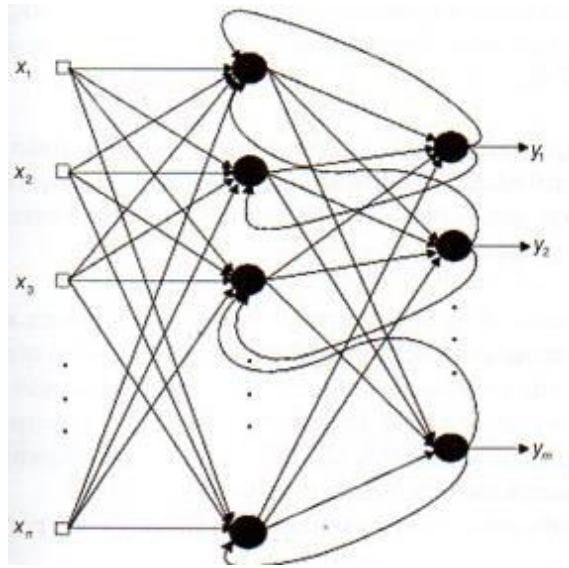


Fonte: Braga, et al.(2007).

As RNAs da Figura 7 e 8 são consideradas estáticas, já que não possuem recorrência em sua estrutura: as suas saídas em um determinado instante dependem apenas das entradas atuais. Já as estruturas das Figuras 9 e 10 possuem conexões recorrentes entre neurônios de um mesmo nível ou entre neurônios de saída e de camada anteriores.

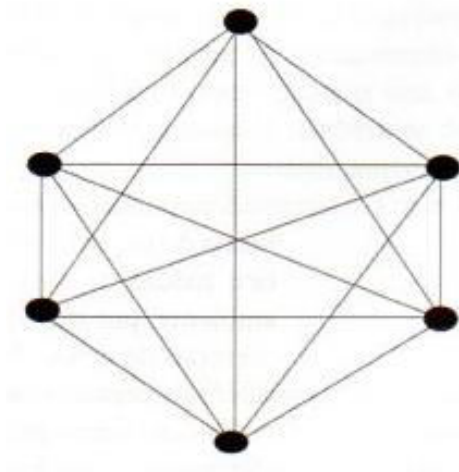
Na Figura 9, a saída não depende somente das entradas, mas também do seu valor atual. Essa estrutura de RNA é utilizada na resolução de problemas que envolvam processamento temporal, como em previsão de eventos futuros. Já a estrutura da Figura 10 possui um único nível de neurônios, em que a saída de cada um deles está conectada as entradas de todos os outros. A rede não possui entradas externas, e sua operação se forma auto-associativa. Esta é uma estrutura típica de uma rede de Hopfield.

Figura 9: Rede com recorrência entre saídas e camada intermediária;



Fonte: Braga, et al.(2007).

Figura 10: Rede com recorrência auto-associativa;



Fonte: Braga, et al.(2007).

Segundo Braga, et al.(2007), a definição de estrutura de uma RNA para a resolução de um determinado problema depende de vários fatores, entre eles:

- Complexidade do problema;
- Dimensionalidade do espaço de entrada;
- Características dinâmicas ou estáticas;
- Conhecimento a priori sobre o problema;
- Representatividade dos dados.

3.7 Aprendizado

A aprendizagem das redes neurais é definida como o processo pelo qual os parâmetros da rede são ajustados através de uma forma continuada de estímulo, pelo ambiente no qual a mesma está operando; o tipo de aprendizagem realizada é definido pela maneira particular como ocorrem os ajustes realizados nos parâmetros (Mendel, 1970).

Braga, et al.(2007) ressalta que o conceito de aprendizado está relacionado com a melhoria do desempenho da rede segundo algum critério preestabelecido. O erro quadrático médio da resposta da rede em relação ao conjunto de dados fornecidos pelo ambiente, por exemplo, é utilizado como critério de desempenho pelos algoritmos de correção de erros. Assim, quando estes algoritmos são utilizados no treinamento de RNAs, espera-se que o erro diminua à medida que o aprendizado prossiga.

O valor do vetor de pesos $\mathbf{w}(t + 1)$ no instante $t + 1$ pode ser escrito conforme a Equação 2,

Equação 2:

$$\mathbf{w}(t + 1) = \mathbf{w}(t) + \Delta\mathbf{w}(t)$$

Onde $\mathbf{w}(t)$ e $\mathbf{w}(t + 1)$ representam valores dos pesos nos instantes t e $t + 1$, respectivamente, e $\Delta\mathbf{w}(t)$ é o ajuste aplicado aos pesos.

Os algoritmos de aprendizado diferem, basicamente, na forma como $\Delta\mathbf{w}(t)$ é calculado. Há vários algoritmos diferentes para treinamento de redes neurais, podendo os mesmos serem agrupados em dois paradigmas principais: aprendizado supervisionado e aprendizado não-supervisionado (BRAGA, CARVALO, LUDEMIR, 2007).

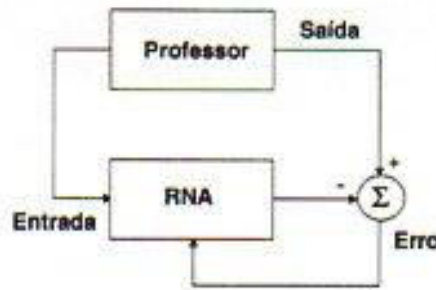
3.7.1 Aprendizado Supervisionado

Aprendizado Supervisionado implica, necessariamente, a existência de um supervisor, ou professor externo, o qual é responsável por estimular as entradas da rede por meio de padrões de entrada e observar a saída calculada pela mesma, comparando-a com a saída desejada. Como a resposta da rede é função dos valores atuais do seu conjunto de pesos, estes são ajustados de forma a aproximar a saída da rede da saída desejada (BRAGA, CARVALO, LUDEMIR, 2007).

A Figura 11 mostra uma representação esquemática do aprendizado supervisionado. Para cada entrada, existe uma saída corrente comparada com a saída desejada pelo supervisor,

que fornece informações sobre a direção de ajuste dos pesos.

Figura 11: Aprendizado supervisionado;



Fonte: Braga, et al.(2007).

Segundo Braga, et al.(2007), o aprendizado pode ser implementado basicamente de duas formas:

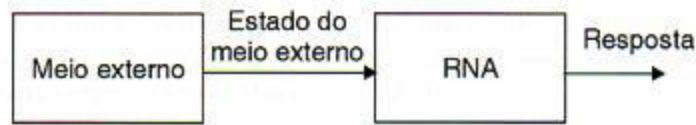
- *Off-line* – os dados do conjunto de treinamento não mudam, e, uma vez obtida uma solução para a rede, esta deve permanecer fixa. Caso novos dados sejam adicionados, um novo treinamento, envolvendo também os dados anteriores, deve ser realizado para se evitar interferência no treinamento anterior;
- *On-line* – o conjunto de dados muda continuamente, e a rede deve estar em um contínuo processo de adaptação.

3.7.2 Aprendizado Não-Supervisionado

No aprendizado não-supervisionado, não há um professor ou supervisor externo para acompanhar o processo de aprendizado. Neste esquema de treinamento somente os padrões de entrada estão disponíveis para a rede, ao contrario do aprendizado supervisionado, cujo conjunto de treinamento possui pares de entrada e saída desejada. Durante o processo de aprendizado os padrões de entrada são apresentados continuamente à rede, e a existência de regularidade nesses dados faz com que o aprendizado seja possível. Regularidade e redundância nas entradas são características essenciais para haver aprendizado não-supervisionado (BRAGA, CARVALO, LUDEMIR, 2007).

Na figura 12 está apresentado um esquema genérico do aprendizado não-supervisionado.

Figura 12: Aprendizado não-supervisionado;



Fonte: Braga, et al.(2007).

Braga, et al.(2007) comenta que o aprendizado não-supervisionado se aplica a problemas que visam à descoberta de características estatisticamente relevantes nos dados de entrada, como, por exemplo, a descoberta de agrupamentos, ou classes.

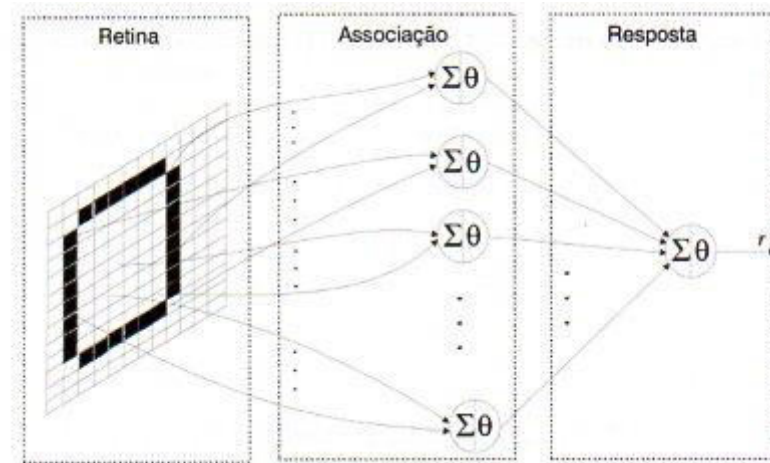
3.8 Perceptron

Segundo Braga, et al.(2007), o modelo proposto por Rosenblatt, conhecido como *perceptron*, era composto por uma estrutura de rede, tendo como unidades básicas neurônios MCP, e por uma regra de aprendizado. Alguns anos mais tarde, Rosenblatt demonstrou o teorema de convergência do *perceptron*, que mostra que o neurônio MCP treinado com o algoritmo de aprendizado do *perceptron* sempre converge caso o problema quem questão seja linearmente separável.

A topologia original descrita por Rosenblatt era composta por unidades de entrada (retina), por um nível intermediário formado pelas unidades de associação e por um nível de saída formado pelas unidades de resposta. Embora essa topologia original possua três níveis, ela é conhecida como perceptron de uma única camada, já que somente o nível de saída (unidades de resposta) apresenta propriedades adaptativas. A retina consiste basicamente em unidades sensores, e as unidades intermediárias de associação, embora formadas por neurônios MCP, possuem pesos fixos, definidos antes do período de treinamento (BRAGA, CARVALO, LUDEMIR, 2007, p.28).

O esboço da topologia do *perceptron* está ilustrada na Figura 13 abaixo.

Figura 13: Topologia de um *perceptron* simples com uma única saída;



Fonte: Braga, et al.(2007).

O *perceptron* funciona da seguinte forma: na ativação de um neurônio artificial, valores de entrada são obtidos, a soma ponderada é calculada e seus resultados são enviados para a função de transferência, gerando o valor de saída. O valor de saída é comparado com o de saída desejada. O erro é calculado diminuindo o valor da saída desejada pelo valor de saída obtido. Este erro é utilizado para cálculo do ajuste dos pesos, dado pela Equação 3, onde *Entrada* é o valor recebido na entrada do neurônio, *Erro* é o erro calculado na saída e β é o passo, que é um valor entre 0 e 1 que determina a velocidade do processo de aprendizado.

Equação 3:

$$Peso - Novo_i = Peso - Antigo_i + \frac{\beta * Erro_i * Entrada_i}{|Entrada_i|}$$

3.9 Rede Perceptron de Múltiplas Camadas

Redes neurais de única camada têm limitações de capacidade de resolução de problemas com características apenas lineares. Tendo em vista que problemas não lineares é a maioria das situações de problemas reais, tendo assim a necessidade de uma estrutura capaz de resolver esses problemas de maior complexidade.

As não-linearidades são incorporadas a modelos neurais através das funções de ativação (não lineares) de cada neurônio da rede e da composição da sua estrutura em camada sucessivas. Assim, a respostas da camada mais externa da rede corresponde à composição das

respostas dos neurônios das camadas anteriores. À rede neural de múltiplas camadas compostas por neurônios com funções de ativação sigmoidais das camadas intermediárias dá-se o nome de *Perceptron* de Múltiplas Camadas (MLP – Multilayer Perceptron) (BRAGA, CARVALO, LUDEMIR, 2007).

Uma rede MLP apresenta segundo Barreto (1999) três características principais:

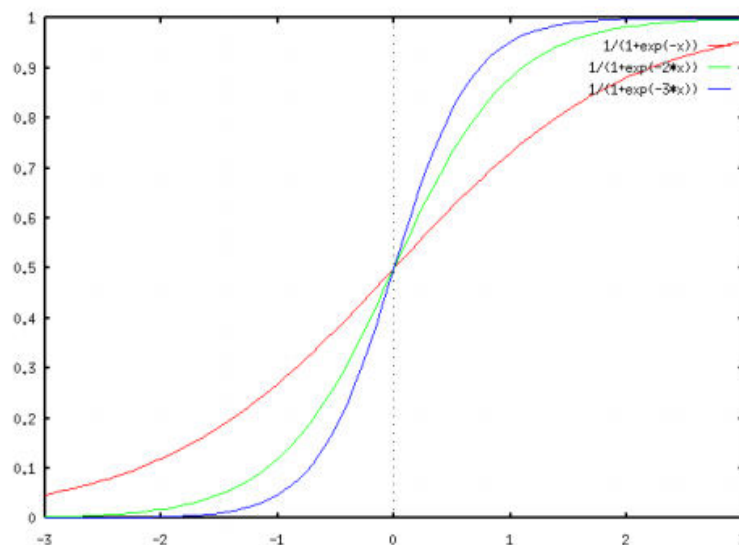
- A rede possui pelo menos uma camada oculta de processamento com neurônios que não fazem parte da entrada ou da saída;
- A rede possui alto grau de conectividade entre seus elementos processadores. Esta conectividade é definida pelos pesos sinápticos;
- O modelo de cada neurônio ou elemento processador da rede possui uma função de ativação não-linear. A função sigmoidal atende esta exigência e é muito utilizada em redes MLP.

Função Logística Sigmoidal é a função de ativação mais utilizada em redes neurais artificiais. Definida como uma função mono tônica crescente que apresenta propriedades assintóticas e de suavidade. Um exemplo de função sigmoidal é a função logística definida pela Equação 4 com sua representação na Figura 14.

Equação 4:

$$f(x) = \frac{1}{1 + e^{-ax}}$$

Figura 14: Gráfico de função Sigmoidal;

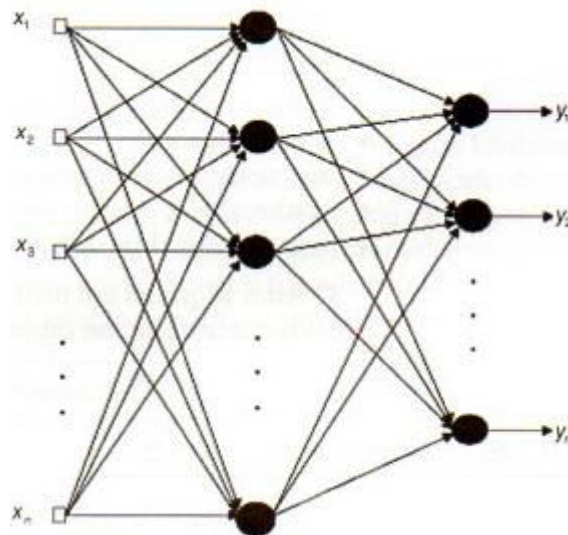


Fonte: elaborado pelo autor.

Redes MLP apresentam um poder computacional maiores do que aquele apresentado pelas redes de uma única camada. Em problemas de classificação, por exemplo, MLPs podem lidar com conjuntos de dados que não sejam linearmente separáveis. Teoricamente, redes com duas camadas intermediárias podem programar qualquer função seja ela linearmente separável ou não (CYBENKO, 1989).

A complexidade de uma rede MLP dependerá do número de neurônios utilizados nas camadas intermediárias, conseqüentemente, a qualidade da aproximação obtida, dependerá da complexidade da rede. A Figura 15 demonstra uma rede MLP típica com uma camada intermediária.

Figura 15: Rede MLP típica com uma camada intermediária;



Fonte: Braga, et al.(2007).

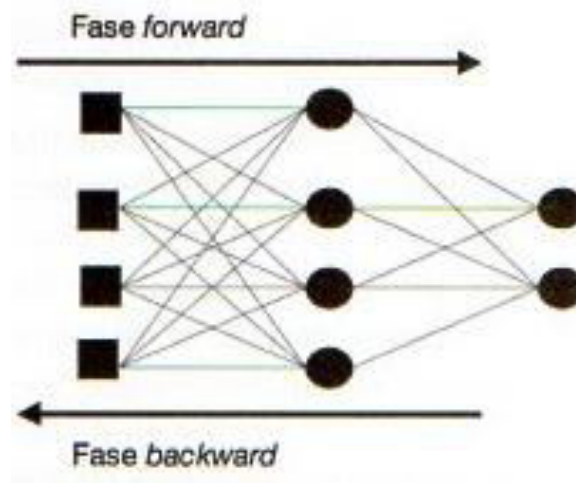
3.10 Backpropagation

O *Backpropagation* é o algoritmo mais popular para treinamento de redes MLP, é supervisionada e utiliza pares de entrada e saída (x, y_d) possibilitando, a correção de erros e ajustes de pesos da rede.

O treinamento ocorre em duas fases também demonstradas na Figura 16:

- Fase *forward* – é utilizada para definir a saída da rede para um dado padrão de entrada.
- Fase *backward* – é utilizada a saída desejada e a saída fornecida pela rede para atualizar os pesos de suas conexões.

Figura 16: Fluxo de processamento do algoritmo *Backpropagation*;



Fonte: Braga, et al.(2007).

A Figura 16 acima ilustra os dados que seguem da entrada para a saída no sentido *forward*, e os erros, da saída para a entrada no sentido *backward*.

Após o treinamento, a variável erro deve estar em um nível satisfatório, e assim a rede poderá ser utilizada como uma ferramenta para classificação de novos dados.

Segundo Fahlman(1988 apud HEINEN, 2002), o *Backpropagation* possui inúmeros problemas e limitações:

- Definição de arquitetura: no *Backpropagation*, é preciso definir manualmente o número de neurônios do RNA, o número de camadas e as interconexões entre os neurônios destas camadas, o que exige que sejam feitas várias simulações até que se consiga chegar a uma arquitetura ideal para a solução de determinado problema;
- Dependência de inicialização dos pesos: como os pesos são iniciados de forma aleatória, nunca se sabe exatamente em que ponto da curva de erro o aprendizado ira iniciar, o que faz com que cada simulação realizada traga resultados diferentes, algumas vezes não sendo possível se chegar a valores satisfatórios;
- Minimização do erro lenta e incerta: são necessárias muitas épocas de aprendizado para que o aprendizado ocorra, e nunca se sabe como antecedência se o aprendizado ira ocorrer de forma satisfatória com a configuração e a topologia utilizada;
- Plasticidade e elasticidade: depois que a uma RNA convergiu para uma

solução, é muito difícil adicionar novos exemplos e continuar o aprendizado do ponto onde parou, e ao se tentar fazer isto, corre-se o risco de fazer com que a RNA esqueça todos os conhecimentos adquiridos anteriormente (esquecimento catastrófico).

Com a finalidade de tentar resolver a maioria destes problemas, Fahlman propôs um novo modelo de Redes Neurais, chamado de *Cascade Correlation*, que corrigem muitas das deficiências presentes no modelo *Backpropagation* (Fahlman, 1988 apud HEINEN, 2002).

3.11 Encog

O Encog é um avançado framework de aprendizado de máquina, no qual suporta uma grande variação de algoritmos, assim como normalização e processamento de dados. Uma variedade de algoritmos de aprendizagem de máquina é suportada, entre eles, algoritmos de redes neurais artificiais. Encog fornece diferentes arquiteturas de redes neurais, diferentes algoritmos de treinamento e suporta a configuração dinâmica de camadas de uma rede neural MLP, como diferentes funções de ativação, definições customizadas de número de camadas e número de neurônios por camadas. (HEATON, 2010).

O Encog Framework é um projeto de código livre, que foi criado em 2008, na qual conta com uma comunidade fortemente ativa até hoje, sendo atualmente disponível para JAVA e plataforma .NET.

No desenvolvimento do sistema, o Encog em sua versão 3.3.0, será utilizado para a criação, treinamento e execução de uma rede neural no módulo de tomada de decisão, que será descrito nos próximos capítulos.

3.12 Trabalhos Correlatos

Com análise e estudo de trabalhos de pesquisa semelhantes ao projeto, na área de RNAs e eficiência energética, destaca-se um artigo com o título “REDES NEURAI APLICADA NA AUTOMAÇÃO DE UMA RESIDÊNCIA ENERGETICAMENTE EFICIENTE”, dos autores F. Franco. Muriel, P. Ferrugem. Anderso e B. Silveira. Antônio César (MURIEL; ANDERSO; CÉSAR, 2014), o trabalho aborda a utilização de redes neurais artificiais para a automação de uma residência energeticamente eficiente, criando um controle eficaz para a realização de abertura e fechamento de janelas da residência a fim de estabelecer um estado de conforto térmico para os ocupantes.

Foi concluído que a utilização de RNAs foi fundamental para aproximar-se o conforto térmico da residência e também criando um controle eficiente para economia de energia.

O projeto em questão serviu como base sustentável para o desenvolvimento desse trabalho na área de redes neurais.

4 DESENVOLVIMENTO

Nas próximas seções do capítulo são apresentados, em detalhes, como foi realizado o desenvolvimento do sistema proposto, assim como todo o processo de teste e estudo da rede neural.

O sistema desenvolvido está escrito na linguagem C# junto a plataforma .Net Framework na versão 4.5.2.

A plataforma .NET é uma iniciativa da empresa Microsoft, comumente conhecida como Microsoft .Net Framework, semelhante à plataforma JAVA, na qual visa uma plataforma única, formada por vários componentes e características como segurança, tratamento de exceções e serviços. Com a plataforma, o programador passa a utilizar o framework fornecido para uma maior abstração de elementos do sistema ou dispositivo específico, ou seja, ele passa a escrever para a plataforma .NET. (MICROSOFT, 2016)

A escolha do C# se deve por ser uma linguagem atual e orientado a objetos, facilitando o desenvolvimento do sistema, lembrando que ele é totalmente compatível com o .NET Framework. Essa escolha foi favorecida pela grande tendência atual da plataforma e o que ela nos pode oferecer, principalmente em questões de compatibilidade entre o SDK (Software Development Kit) de ambiente de simulação do Home I/O e o framework de rede neural Encog.

4.1 Ferramentas Utilizadas

Todas as ferramentas utilizadas no projeto foram escolhidas por principalmente serem compatíveis com a plataforma .NET e sanarem a necessidade requerida.

O Home I/O é utilizado para simular uma casa e seus dispositivos, tornando fácil a simulação e a verificação de resultados por conta de seu dinamismo, sendo ele totalmente integrável via SDK via plataforma .NET Framework.

O Encog é um framework utilizado para simplificar a implementação de uma rede neural no decorrer da implementação do projeto. Ele conta também com varias extensões, como normalização, geração de topologia de rede neural, treinamento, o que ajuda no decorrer do desenvolvimento.

4.2 Definição de Cenário no Home I/O

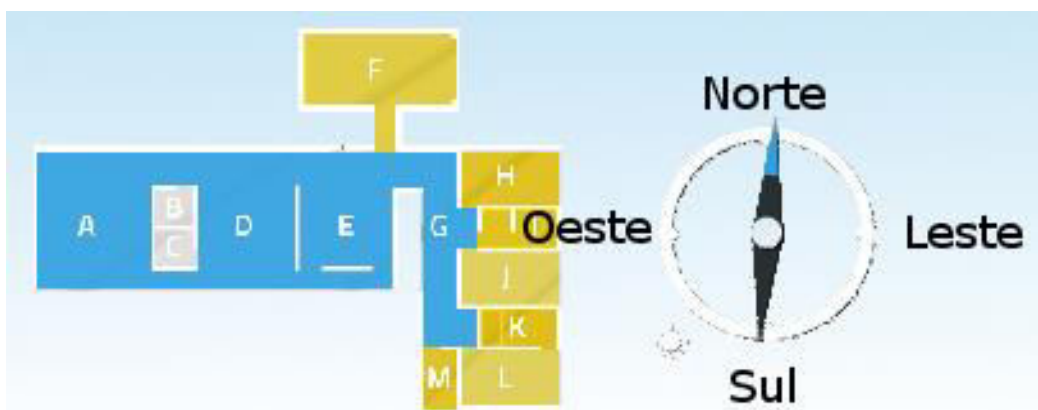
No projeto proposto, serão utilizados quatro cômodos compartilhados do simulador Home I/O como mostrado em azul na Figura 17. Isso irá garantir com que junto ao comportamento térmico do simulador, um cômodo interaja com outro, garantindo um maior dinamismo na simulação. A configuração dos cômodos perante os dispositivos, que estão representados na Figura 17, fica definida como:

- Cada cômodo A, D e E, contém um termostato e um aquecedor;
- O cômodo G contém um termostato e dois aquecedores;
- Todos os outros cômodos não terão termostatos ou aquecedores funcionando.

Os parâmetros globais do simulador ficam definidos como:

- Temperatura Mínima: 10°C;
- Temperatura Máxima: 18°C;
- Umidade: 65%;
- Velocidade do Vento: 14 km/h;
- Nebulosidade: 10%;
- Direção do vento: Norte;
- Nascer do sol: Sul, Fixo;
- Data: 26/10/2016;
- Longitude: 41.14997;
- Latitude: -7.38976.

Figura 17: Cômodos para o cenário escolhido em azul no simulador Home I/O.



Fonte: elaborado pelo autor.

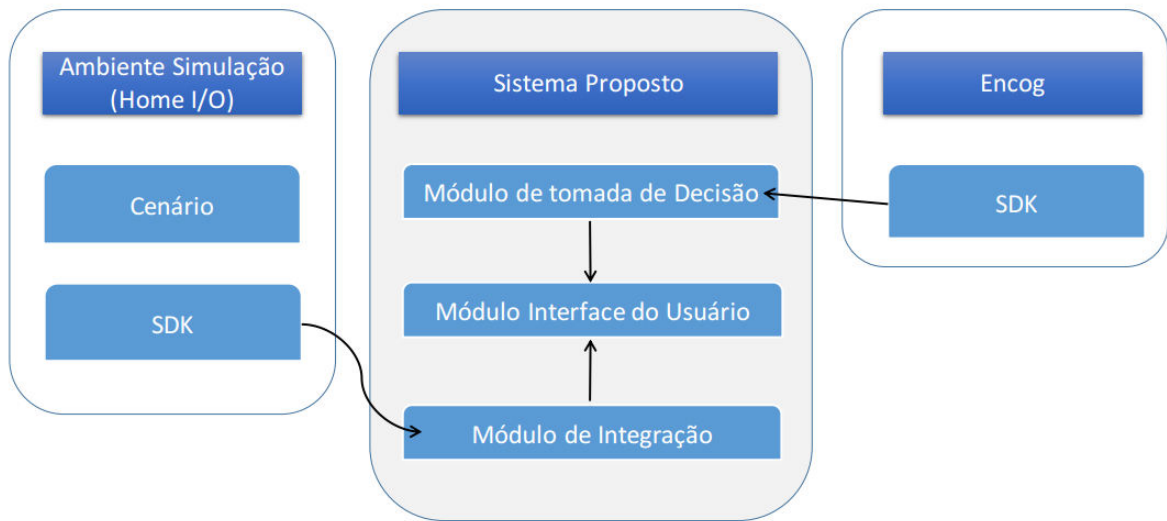
Todos os dispositivos (Termostatos e Aquecedores) dos cômodos a serem utilizados no projeto (A, D, E, G) serão colocados no modo *External Mode*, para que possa ser possível a integração via SDK pelo Módulo de Integração desenvolvido nas próximas seções.

4.3 Implementação do Sistema

O sistema proposto foi implementado na linguagem C# junto a plataforma .NET, utilizando o framework Encog, que foi explicado no capítulo 3.11, para auxílio no desenvolvimento das redes neurais. O sistema proposto foi dividido em três módulos por conterem responsabilidades únicas e proporcionarem um melhor entendimento, conforme a Figura 18 e especificados a seguir:

- Módulo de Integração: Integra com o SDK do Home I/O, fornecendo abstração necessária para utilização do mesmo em forma de classes estáticas. Também possui modelos e bibliotecas importantes de normalização, conversores (temperatura e métricas);
- Módulo de Tomada de Decisão: Junto com o framework Encog, essa biblioteca abstrai o uso da rede neural aplicada. Também é onde todo o processamento do sistema é feito;
- Módulo de Interface do Usuário: Interação com o usuário final, permitindo gerenciar a rede neural (carregar, salvar, treinar e executar). Nesse módulo, utiliza-se o Módulo de Integração em conjunto ao Módulo de Tomada de Decisão, integrando assim, o controle de dispositivos simulados com resultados da tomada de decisão da rede neural em tempo real.

Figura 18: Módulos do sistema proposto junto a integrações com SDK



Fonte: elaborado pelo autor.

O sistema utilizado no projeto foi criado com base nas especificações dos próximos capítulos que serão apresentados, explicando minuciosamente o desenvolvimento dos três módulos, objetivos e funcionalidades.

Todo o sistema desenvolvido está em código aberto, disponível em um repositório no endereço: “<https://github.com/danilobreda/TCC>”.

4.3.1 Módulo de Integração

O objetivo do módulo de integração é abstrair a conexão com o SDK do Home I/O, simplificando a interação do módulo de interface do usuário com o simulador.

Dados são disponibilizados de forma normalizada para uma melhor integração posteriormente com o módulo de tomada de decisão como descrito nos capítulos posteriores.

O SDK do Home I/O fornece a classe *MemoryMap* que é *Singleton*, o que significa que apenas uma instância dela pode existir em qualquer período de tempo na execução. Nela está representado uma cópia em *cache* do arquivo mapeado em memória, que contém todos os dados do ambiente de simulação, como sensores e parâmetros globais. (HOMEIO, 2016).

O método *MemoryMap.Instance.Update* é responsável por sincronizar os dados do simulador, contidos no arquivo mapeado em memória com o *cache* que está situado na propriedade *MemoryMap.Instance*. O mesmo precisa ser chamado todas as vezes que forem necessários os últimos dados do simulador. (HOMEIO, 2016).

No código 1.1, o método *Update* é chamado para atualização do *cache* da instância. Posteriormente é realizado a coleta de um objeto do tipo *MemoryFloat* do cachê através do método *GetFloat*, passando como parâmetros o endereço de memória e o tipo de grupo do mesmo.

```
MemoryMap.Instance.Update();
MemoryFloat temperatura = MemoryMap.Instance.GetFloat(1, MemoryType.Input);
```

Código 1.1: Atualizando dados do *singleton MemoryMap* e recebendo temperatura de um dispositivo.

Dados como temperatura, não podem ser diretamente utilizados em uma rede neural. RNAs podem ser inteligentes, porém não consegue receber um dado qualquer e produzir um resultado que tenha um sentido. Sempre o dado de entrada precisa ser normalizado. (HEATON, 2010).

No sistema, a normalização está como padrão de saída de valores sendo zero a um. Essa escolha está baseada no uso da função *Sigmoid* e está descrita nas próximas seções. Essa escolha é a mais comumente utilizada em sistemas de rede neural MLP. Por conta dos dados se originarem do módulo de integração, se viu viável a normalização já ser realizada nesse módulo, para que os dados sejam utilizados sem problemas posteriormente por outros módulos.

O módulo de integração conta com uma biblioteca de normalização. No código 1.2 abaixo, um valor não normalizado é recebido através da variável *temperatura* criada no código 1.1. O valor então é normalizado através da função *Norm_Temp*, da classe *Normalizacao*, passando o valor a ser normalizado como parâmetro e retornando um valor normalizado para a variável *valorNormalizado*.

```
var valor = temperatura.Value;
var valorNormalizado = Normalizacao.Norm_Temp(temperatura.Value);
```

Código 1.2: Variável *valor* recebe propriedade de temperatura e normalizado para variável *valorNormalizado*.

Para ligar ou desligar um aquecedor, o módulo de integração realiza a seguinte operação descrita no código 1.3 e 1.4. No código 1.3, o método *AtualizarBit* é chamado, passando o parâmetro do endereço de memória e o estado requerido. No código 1.4, o método *AtualizarBit* é implementado, nele, realiza-se então a atualização de cachê da instância do

MemoryMap, recebe o objeto *MemoryBit* de um determinado endereço de memória e realiza a atualização da propriedade *Value* pelo novo valor atribuído. O *MemoryMap* depois é atualizado através do método *Update*, passando os novos dados para o simulador. Ligando ou desligando o aquecedor em tempo real.

```
public void LigarAquecedor_A()
{
    AtualizarBit(9, true);
}
public void DesligarAquecedor_A()
{
    AtualizarBit(9, false);
}
```

Código 1.3: Métodos de ligar e desligar aquecedores.

```
private void AtualizarBit(int endereco, bool estado)
{
    MemoryMap.Instance.Update();
    MemoryBit elemento = MemoryMap.Instance.GetBit(endereco, MemoryType.Output);
    elemento.Value = estado;
    MemoryMap.Instance.Update();
}
```

Código 1.4: Método de abstração do processo de alteração de propriedade da classe *MemoryMap*.

Para uma melhor organização e facilitar a utilização, os dados do simulador, contidos na instância de *MemoryMap*, foram agrupados para varias classes que seguem:

- *DadosMemoryClima*: Dados do clima;
- *DadosMemoryDateTime*: Dados do horário do simulador;
- *DadosMemoryEnergia*: Dados de consumo de energia;
- *DadosMemoryTemperatura*: Temperaturas de parâmetros globais;
- *DadosTermostato*: Dados do termostato utilizados no Home I/O, como temperatura atual, temperatura desejada e estado(ligado/desligado).

Existe ainda a classe *DadosMemory*, que agrupa *DadosMemoryClima*, *DadosMemoryDateTime*, *DadosMemoryEnergia* e *DadosMemoryTemperatura* em uma única classe, para simplificação de retorno dos dados.

Junto aos grupos de divisão de memória do Home I/O (*Inputs*, *Outputs* e *Memories*), todas as classes de abstração estão definidas através desses mesmos grupos, que são:

- *SimulationMemory*: Retorno de objeto do tipo *DadosMemory*, contendo todos

os dados de parâmetros globais do simulador, sendo suas propriedades normalizadas e não normalizadas;

- *SimulationInput*: Retorno de dados de dispositivos determinados, os métodos *DadosTermostato_A* por exemplo, retorna um objeto do tipo *DadosTermostato*, contendo todos os parâmetros do dispositivo, sendo suas propriedades normalizadas e não normalizadas;
- *SimulationOutput*: Contém métodos que realizam uma ação, como por exemplo, ligar ou desligar os aquecedores, através dos métodos *LigarAquecedor_A* e *DesligarAquecedor_A*.

Nesse ponto, o módulo de integração contém toda a abstração e normalização de dados necessários para ser utilizada posteriormente.

4.3.2 Módulo de Tomada de Decisão

O objetivo do módulo de tomada de decisão é aplicar uma rede neural MLP que represente o ambiente de simulação e tenha resultados satisfatórios.

Outro ponto importante a ser destacado, é a abstração da rede neural para uso em outros módulos de forma simplificada.

4.3.2.1 Configuração

As definições da configuração da rede podem ser feitas de forma empírica, ela ainda é considerada uma arte, contudo sua organização pode ser dividida em três etapas:

- Seleção de paradigma neural apropriado à aplicação (tipo de rede neural).
- Determinação da topologia da rede a ser utilizada (número de camadas e neurônios por camadas).
- Determinação dos parâmetros do algoritmo de treinamento e função de ativação.

Por existirem entradas e principalmente saídas desejadas, foi utilizado uma rede supervisionada, ao contrario da rede não-supervisionada, que não leva em conta as saídas desejadas em seu treinamento.

Uma rede neural *feedforward* como a MLP tem o objetivo de resolver problemas não-linearmente separáveis quem contenha inúmeras entradas e saídas. Ele é o mais comumente utilizado e totalmente compatível com o Encog.

O número de camadas intermediárias e neurônios por camadas influenciam diretamente o desempenho e resultados, sendo a quantidade ligada diretamente à complexidade do problema a ser resolvido.

As camadas de entrada, assim como a camada intermediária, contêm uma unidade estratégica especial chamada *Bias*, que é utilizada para aumento do grau de liberdade de uma rede neural, gerando um melhor desempenho para adaptação, pela rede neural, através de amostras fornecidas no momento do treinamento. O *Bias* é totalmente compatível pelo Encog.

No código 2.1 abaixo é definido uma rede neural do tipo *BasicNetwork* no Encog. São adicionados 3 camadas, sendo eles:

- Camada de Entrada: 8 neurônios (entradas), com a função de ativação do tipo *Sigmoid*, possui *Bias*. Cada cômodo, do cenário do capítulo 4.2, contém duas entradas, sendo temperatura atual e temperatura desejada;
- Camada Intermediária: 5 neurônios, com a função de ativação do tipo *Sigmoid*, possui *Bias*. O número de neurônios para essa camada pode ser definida de modo empírico, sua escolha foi mais bem definida e explicada no capítulo 5.1;
- Camada de Saída: 4 neurônios (saídas), com a função de ativação do tipo *Sigmoid*, não possui *Bias* por não serem usuais na camada de saída. Cada cômodo, do cenário do capítulo 4.2, contém uma saída, que é a ação de ligar ou desligar o aquecedor.

No código 2.1, está exemplificado o processo de criação de uma rede neural no módulo de tomada de decisão. Nele é realizado o método *FinalizeStructure*, para com que a estrutura da rede seja finalizada.

O método *Reset* insere valores randômicos em todos os limiares da rede neural, sendo isso necessário para que ocorra o treinamento do mesmo.

A rede neural está criada e pronta para o processo de treinamento inicial.

```

public const int N_input = 8;
public const int N_hidden = 5;
public const int N_output = 4;

BasicNetwork network = new BasicNetwork();
network.AddLayer(new BasicLayer(new ActivationSigmoid(), true, N_input));
network.AddLayer(new BasicLayer(new ActivationSigmoid(), true, N_hidden));
network.AddLayer(new BasicLayer(new ActivationSigmoid(), false, N_output));
network.Structure.FinalizeStructure();
network.Reset();

```

Código 2.1: Configuração de uma rede neural no Encog.

4.3.2.2 Treinamento

Nos códigos 2.2 e 2.3 abaixo, estão exemplificado o processo treinamento de uma rede neural no módulo de tomada de decisão.

O treinamento inicial da rede neural consiste em apresentar padrões para a rede, e assim executar seu treinamento fazendo com que os limiares sejam atualizados de acordo com as entradas e saídas desejadas.

O algoritmo utilizado para a rede neural MLP é o *Backpropagation*, na qual é totalmente compatível com o framework Encog. No código 2.2, uma representação dos dados de treinamento foi instanciado pela classe *BasicNeuralDataSet*, posteriormente um objeto *Backpropagation* é instanciado, passando a rede neural já configurada e o *trainingSet* como parâmetro no construtor.

É importante lembrar que a taxa de aprendizagem e o parâmetro *momentum*, não foram definidos no treinamento, passando para o framework Encog, a responsabilidade de definição, que por padrão é atribuído internamente através de estratégias de treinamento através das classes *SmartLearningRate* e *SmartMomentum*.

```

var trainingSet = new BasicNeuralDataSet(pointsConverted.entrada, pointsConverted.saida);

var train = new Backpropagation(network, trainingSet);

```

Código 2.2: Instanciando *BasicNeuralDataSet* e treinamento *Backpropagation*.

O treinamento então deve ser iniciado, um loop é criado no código 2.3, na qual cada interação o método *Iteration* é invocado, realizando a atualização dos limiares da rede neural com base nos dados de treinamento passados anteriormente. Esse processo continua até que o número de épocas (variável *epoch*) seja menor que 20.000, ou que o erro seja menor que 0.1%. Sendo essas variáveis definidas de forma empírica, porém explicadas no capítulo 5.1.

```

var epoch = 0;
do
{
    train.Iteration();
    epoch++;
} while ((epoch <= 20000) && (train.Error > 0.01));

```

Código 2.3: Processo de treinamento da rede neural no Encog.

4.3.2.3 Validação

Para validação da rede neural treinada, o módulo de tomada de decisão contém o método `Compute`, como no código 2.4, que recebe dados do tipo *BasicMLData* para reconhecimento, gerando a variável *outData*. Esta variável do tipo *IMLData*, que é um tipo do Encog, deve ser transformada em um *array*, que é uma estrutura de representação de dados, do tipo *double*, que é um tipo de representação de dados decimal, para que possa ser retornado os valores para outros módulos que não tenham referência ao Encog, como por exemplo, o módulo de interação do usuário. Assim sendo, o *array* da variável *arrayOutput* é gerado e retornado.

```

public static double[] Compute(BasicMLData inputData)
{
    var outData = network.Compute(inputData);
    var arrayOutput = new double[N_output];
    for (int i = 0; i < N_output; i++)
    {
        arrayOutput [i] = outData [i];
    }
    return arrayOutput;
}

```

Código 2.4: Processamento de resultado da rede neural no Encog.

O código 2.4 acima está exemplificado o processo validação e retorno de resultado em rede neural anteriormente criada e treinada no módulo de tomada de decisão.

4.3.3 Módulo de Interface do Usuário

O objetivo do módulo de interface é utilizar o módulo de integração junto ao módulo de tomada de decisão, permitindo com que o ambiente de simulação do Home I/O seja simulado integralmente com a rede neural em tempo real. O módulo de interface de usuário também permite para o usuário final opções como:

- Treinamento da rede neural através de arquivo de texto;

- Salvar rede neural anteriormente treinada;
- Carregar rede neural salva anteriormente pelo sistema;
- Executar simulação em tempo real junto ao simulador Home I/O.

No processo treinamento da rede neural, o código 3.1 realiza a leitura do arquivo de treinamento e o transforma em um *array* do tipo *string*. O *array* é transformado para um objeto do tipo *PointsConvertor* através da classe *PointsConvertor* no método *Converter*, passando o *array* de *string* como parâmetro. Utilizando o módulo de tomada de decisão, a rede neural é treinada através da classe MLP no método *Train*, passando o parâmetro *PointsConvertor*, já convertido. Com isso, a rede neural será treinada, um processo que pode demorar alguns minutos.

```
var neuralFile = File.ReadAllLines(ofd.FileName);
var pc = PointsConvertor.Converter(neuralFile);
MLP.Train(pc);
```

Código 3.1: Processo de Treinamento do módulo de interface de usuário.

Após esse processo, a rede neural está treinada para reconhecer padrões do simulador Home I/O. A opção de executar a simulação como mostrado no código 3.2, realiza a execução do método *Executa* que está implementado no código 3.3.

```
Executa();
```

Código 3.2: Código para execução de simulação.

No método *Executa* descrito no código 3.3, é executado um loop, na qual são recebidos dados de sensores do simulador Home I/O, através do módulo de integração, e dados normalizados são colocados em variáveis para posteriormente a serem utilizados.

```
do
{
    var Dados_A = Simulation.Input.Termostato_A();
    var Dados_D = Simulation.Input.Termostato_D();
    ...
    var TempA = Dados_A.TemperaturaNormalizado;
    var SetA = Dados_A.SetPointNormalizado;
    var TempD = Dados_D.TemperaturaNormalizado;
    var SetD = Dados_D.SetPointNormalizado;
    ...
}
while (true);
```

Código 3.3: Método *Executa*. Preparação de dados para computação de resultado.

Dados de entrada deverão ser disponibilizados para a rede neural poder calcular a saída. O Encog utiliza a classe *BasicMLData* como classe para dados de entrada de uma rede neural MLP. Conforme o código 3.4, os dados de entrada do tipo *BasicMLData* é criado, carregando dados de sensores que foram guardados no código 3.3.

```
var dataEntrada = new BasicMLData(new double[]
    {
        TempA, SetA, TempD, SetD, TempE, SetE, TempG, SetG
    }
);
```

Código 3.4: Carregamento dos dados de sensores

Para o reconhecimento dos dados de entrada, é realizado o processamento através do método *Compute* no código 3.5, o resultado do módulo de tomada de decisão, é recebido na variável *dataSaida*.

```
var dataSaida = MLP.Compute(dataEntrada);
```

Código 3.5: Reconhecimento do padrão

Por conta de a rede neural ter quatro saídas, a variável de saída será um *array* do tipo *double* com quatro elementos, sendo um para cada aquecedor. Como a saída da rede neural é um valor de zero a um, é necessário a verificação para a execução da ação desejada. No código 3.6, é verificado se a saída de cada elemento do *array* é maior ou menor que 0.5, executando assim a opção desejada do módulo de integração.

```
if (dataSaida[0] >= 0.5)
    Simulation.Output.LigarAquecedor_A();
else
    Simulation.Output.DesligarAquecedor_A();

if (dataSaida[1] >= 0.5)
    Simulation.Output.LigarAquecedor_D();
else
    Simulation.Output.DesligarAquecedor_D();
...

```

Código 3.6: Reconhecimento do padrão

Assim, o módulo de interface do usuário está integrado com o módulo de integração e o módulo de tomada de decisão, possibilitando sua simulação como demonstrado acima.

5 SIMULAÇÃO E RESULTADOS OBTIDOS

Nas próximas seções do capítulo é realizada a simulação junto ao sistema proposto para análise detalhada do comportamento de variáveis.

Foi necessária a captura de dados para o treinamento da rede neural desenvolvida no capítulo anterior e execução do sistema junto ao simulador HOME I/O para análise de resultados.

5.1 Geração de Dados para Treinamento

O sistema desenvolvido deverá ter sua rede neural treinada, através de dados fornecidos, e posteriormente validado, realizando suas tomadas de decisão. Nas próximas seções, serão apresentados os dois métodos realizados para a obtenção de dados.

É importante lembrar que toda a aquisição de dados de treinamento foi baseada no cenário proposto no capítulo 4.2.

O padrão de dados no arquivo gerado demonstrado na Figura 19 é totalmente compatível ao cenário proposto e a rede neural desenvolvida.

Figura 19: Padrão de arquivo de dados utilizado para treinamento da rede neural.

```

{0}::{1}::{2}::{3}::{4}::{5}::{6}::{7}::{8}::{9}::{10}::{11}

{0} = TemperaturaAtual_A
{1} = TemperaturaDesejada_A
{2} = TemperaturaAtual_D
{3} = TemperaturaDesejada_D
{4} = TemperaturaAtual_E
{5} = TemperaturaDesejada_E
{6} = TemperaturaAtual_G
{7} = TemperaturaDesejada_G

{8} = Status_Aquecedor_A
{9} = Status_Aquecedor_D
{10} = Status_Aquecedor_E
{11} = Status_Aquecedor_G

Saída Final:
0,25::0,35::0,26::0,75::0,26::0,63::0,56::0,89::1::1::0::1

```

Fonte: elaborado pelo autor.

5.1.1 Captura via Simulador Home I/O

A captura dos dados de sensores do simulador Home I/O é feita através de um módulo desenvolvido exclusivamente para seu propósito, chamado *ModuloCapturaSimulacao*. Sua finalidade é gerar dados de treinamento baseados no ambiente simulado. Este módulo utiliza o módulo de integração apenas para leitura dos dados de sensores e atuadores do simulador e transformando-o em um arquivo padronizado que será utilizado posteriormente para treinamento da rede neural do sistema.

A captura é feita até um total de dez mil dados gerados pelo módulo. No decorrer da captura, é realizada a interação com o simulador HOME I/O, trocando valores dos termostatos, e parâmetros como mostrado na Figura 20. É necessário que todos os dispositivos da casa estejam no modo *External Mode* durante a captura, ou seja, a casa está com sua lógica de atuação baseada no módulo desenvolvido descrito anteriormente.

Figura 20: Demonstração da interação do usuário com o simulador HOME I/O



Fonte: elaborado pelo autor.

O módulo *ModuloCapturaSimulacao*, realiza a leitura dos sensores e termostatos, verifica a temperatura atual e temperatura desejada, e após uma lógica simples, liga ou desliga o aquecedor. Todos os valores são salvos para que posteriormente possa ser gerado o arquivo de treinamento. É importante saber que o processo lógico de ligar ou desligar o aquecedor é

baseado em sistemas de tomada de decisão simples de aquecedores e termostatos comuns, utilizados usualmente.

Após a captura, o módulo cria um arquivo que será utilizado nos próximos capítulos para treinamento da rede neural do sistema proposto.

5.1.2 Geração de Dados Virtuais

A geração de dados de sensores e atuadores é realizada por um módulo desenvolvido exclusivamente para seu propósito. O módulo é chamado *ModuloCapturaVirtual* e vem suprir a necessidade de geração de dados de treinamento precisos, sem a interação do usuário com o simulador e do simulador em si.

O módulo realiza a geração de dez mil dados de sensores e atuadores, que são feitas virtualmente, baseadas nos dispositivos informados no cenário proposto no capítulo 4.2.

Seu funcionamento consiste em gerar valores decimais aleatórios em intervalos padronizados do simulador Home I/O descritos a seguir:

- Temperatura Ambiente: -20° Celsius até 50° Celsius
- Temperatura Desejada: 5° Celsius até 30° Celsius

Depois de gerado os valores de cada termostato dos cômodos, o módulo realiza uma lógica simples de tomada de decisão, ligando ou desligando o aquecedor, semelhante ao módulo *ModuloCapturaSimulacao*, demonstrado no capítulo anterior.

Após a geração dos dados, o módulo gera um arquivo de treinamento de dados padronizados que será utilizado nos próximos capítulos.

5.2 Cenário de Treinamento

Junto ao cenário do simulador Home I/O definida no capítulo 4.2, será necessária a definição de cenários de treinamento para uma melhor diversidade de resultados nas simulações efetuadas nas próximas seções, comparando, e concluindo como a rede neural pode ser treinada e melhorada.

Serão definidos três cenários diferentes de treinamento da rede neural:

- Treinamento via dados de Captura: São utilizados apenas dados capturados pelo simulador Home I/O e interações realizadas. Referente ao capítulo 5.2.1 via módulo *ModuloCapturaSimulacao*.

- Treinamento via dados Gerados: São utilizados apenas dados gerados virtualmente. Referente ao capítulo 5.2.2 via módulo *ModuloCapturaVirtual*.
- Treinamento via dados Gerados e de Captura: São utilizados tanto dados gerados virtualmente quanto dados capturados do simulador Home I/O. Referente ao capítulo 5.2.1 e 5.2.2.

Cada cenário contém 20.000 iterações de treinamento da rede neural ou erro menor que 0.1%. O número de pontos de dados de treinamentos é de 10.000.

Todos os cenários serão treinados pelo sistema desenvolvido como mostra a Figura 21, via opção *Treinar*. Após clicar na opção, é necessário selecionar o arquivo de pontos de dados de treinamento e aguardar alguns minutos. Você pode selecionar mais que um arquivo de pontos de dados de treinamento.

Figura 21: Treinamento da rede neural através de pontos de dados de treinamento.

```
MENU:
1 - Treinar
2 - Carregar Rede Neural
3 - Salvar Rede Neural
4 - Executar
0 - Sair

Aguarde...
Epoch #0 Error:0,339287475168597
Epoch #1 Error:0,308830170184603
Epoch #2 Error:0,282882499227764
Epoch #3 Error:0,26550693114764
Epoch #4 Error:0,256338556026959
Epoch #5 Error:0,252308538189601
Epoch #6 Error:0,250710297725486
Epoch #7 Error:0,250097907341767
Epoch #8 Error:0,249857358378219
Epoch #9 Error:0,249752773592663
Epoch #10 Error:0,249697203393916
Epoch #11 Error:0,249659135672128
Epoch #12 Error:0,249627235161039
Epoch #13 Error:0,249597440662084
Epoch #14 Error:0,249568301031732
Epoch #15 Error:0,249539298462071
Epoch #16 Error:0,249510247586276
Epoch #17 Error:0,249481080741805
Epoch #18 Error:0,249451771524908
Epoch #19 Error:0,249422307744301
Epoch #20 Error:0,249392681952999
Treino realizado com sucesso
Inicio: 04:17:45
Fim: 04:17:46
```

Fonte: elaborado pelo autor.

Ao finalizar o processo, a rede neural estará treinada, com isso será necessário salvar como arquivo do MLP. Esse arquivo é gerado pelo Encog, tendo sua implementação abstraída no sistema desenvolvido. Todo esse processo é demonstrado na Figura 22.

Figura 22: Salvando rede neural treinada no formato MLP.

```
MENU:
1 - Treinar
2 - Carregar Rede Neural
3 - Salvar Rede Neural
4 - Executar
0 - Sair

Aguarde...
Arquivo salvo em: C:\Users\bredi\Desktop\TCC\TCC\TrainingData\redeneural_arquivo.mlp
```

Fonte: elaborado pelo autor.

5.3 Simulação com o Home I/O

Toda a simulação realizada nesse capítulo foi realizada através do simulador Home I/O, no cenário proposto do capítulo 4.2. Os resultados obtidos com base nas simulações serão apresentados na próxima seção.

Para carregar o arquivo do tipo MLP é necessário utilizar a opção e selecionar o arquivo. Com isso, todos os dados da rede neural serão recarregados prontos para serem executados, como mostrado na Figura 23 abaixo.

Figura 23: Carregando rede neural já treinada de arquivo no formato MLP.

```
MENU:
1 - Treinar
2 - Carregar Rede Neural
3 - Salvar Rede Neural
4 - Executar
0 - Sair

Aguarde...
Rede neural recarregada com sucesso.
```

Fonte: elaborado pelo autor.

É necessária a configuração do cenário do simulador Home I/O igual ao cenário proposto no capítulo 4.2.

Com isso, a rede neural está pronta para ser executada. Selecionando a opção de executar mostrada na Figura 24 junto com o simulador Home I/O aberto, é mostrado informações em tela da simulação ocorrendo em tempo real com a rede neural em execução. Na tela algumas informações essenciais de sensores e atuadores, bem como saídas da rede neural, tomadas de decisão realizada e diferenças entre temperatura desejada e temperatura atual.

Figura 24: Execução da rede neural com o simulador Home I/O em tempo real.

```

MENU:
1 - Treinar
2 - Carregar Rede Neural
3 - Salvar Rede Neural
4 - Executar
0 - Sair

Aguarde...
A: 0,430913638319407 | D: 0,265467493906094 | E: 0,0180618509940111 | G: 0,942447681145688
A: OFF T: 0,0 | D: OFF T: 1,0 | E: OFF T: 3,9 | G: ON T: -3,1 |
A: 0,880867509775056 | D: 0,231316618442442 | E: 0,0121692355345526 | G: 0,903882542114217
A: ON T: -3,0 | D: OFF T: 1,0 | E: OFF T: 3,9 | G: ON T: -3,1 |
A: 0,88087246894031 | D: 0,231321507893811 | E: 0,0121683027195761 | G: 0,903886418070403
A: ON T: -3,0 | D: OFF T: 1,0 | E: OFF T: 3,9 | G: ON T: -3,1 |
A: 0,880878144305707 | D: 0,231327312739393 | E: 0,0121672426925131 | G: 0,90389092123648
A: ON T: -3,0 | D: OFF T: 1,0 | E: OFF T: 3,9 | G: ON T: -3,1 |
A: 0,880883112746115 | D: 0,23133283756509 | E: 0,0121662490988532 | G: 0,903895043192002
A: ON T: -3,0 | D: OFF T: 1,0 | E: OFF T: 3,9 | G: ON T: -3,1 |
A: 0,880888253147759 | D: 0,231338373009703 | E: 0,0121652133587974 | G: 0,903899292883773
A: ON T: -3,0 | D: OFF T: 1,0 | E: OFF T: 3,9 | G: ON T: -3,1 |
A: 0,880893232805152 | D: 0,231344270776574 | E: 0,012164158425849 | G: 0,903903812887233
A: ON T: -3,0 | D: OFF T: 1,0 | E: OFF T: 3,9 | G: ON T: -3,1 |

```

Fonte: elaborado pelo autor.

Até nesse momento, podemos visualizar no simulador Home I/O, em um determinado cenário, o sistema de controle de temperatura de uma casa sendo controlado por uma rede neural em tempo real, que foi previamente treinada, podendo ainda interagir através da mudança de parâmetros de temperatura e visualizando seus resultados.

5.4 Resultados Obtidos

Nesta seção, os resultados obtidos serão analisados, através de gráficos e tabelas, dados esses que foram gerados através de testes e simulação, utilizando o sistema desenvolvido junto ao simulador Home I/O no cenário proposto no capítulo 4.2.

5.4.1 Dados de Referência

Esse capítulo irá definir valores de referência base para que possam ser comparados com valores obtidos através de simulação nos próximos capítulos, visando uma análise concreta para conclusão.

Os valores de referência base foram adquiridos através do simulador Home I/O nos mesmos cenários propostos. A lógica de aquisição de dados do simulador é baseada em sistemas de tomada de decisão simples de aquecedores e termostatos comuns, utilizados usualmente.

Como referências para os resultados desse capítulo foram utilizadas tabelas comparativas divididas em duas categorias, demonstrados na Figura 25 e também disponível no Apêndice B, sendo elas:

- Mesma Temperatura: Todos os cômodos do cenário proposto da simulação utilizaram o parâmetro temperatura desejada equivalentes a 20°C;
- Temperatura Diferente: Todos os cômodos do cenário proposto da simulação utilizaram parâmetros de temperatura desejada variadas, sendo elas: Cômodo A - 23°C; Cômodo D – 21°C; Cômodo E – 22°C; Cômodo G – 20°C.

Figura 25: Tabela de Referência Base de resultados obtidos.

Referencia Base		
Categoria	Gasto Médio (Kwh)	Desvio Temperatura Total (°C)
Mesma Temperatura	31,48	0,37
Temperatura Diferente	43,6	1,28

Fonte: elaborado pelo autor.

A figura 25, define Gasto Médio em KWh e o Desvio de Temperatura Total em grau Celsius.

Segue algumas definições importantes que serão utilizadas nos próximos capítulos:

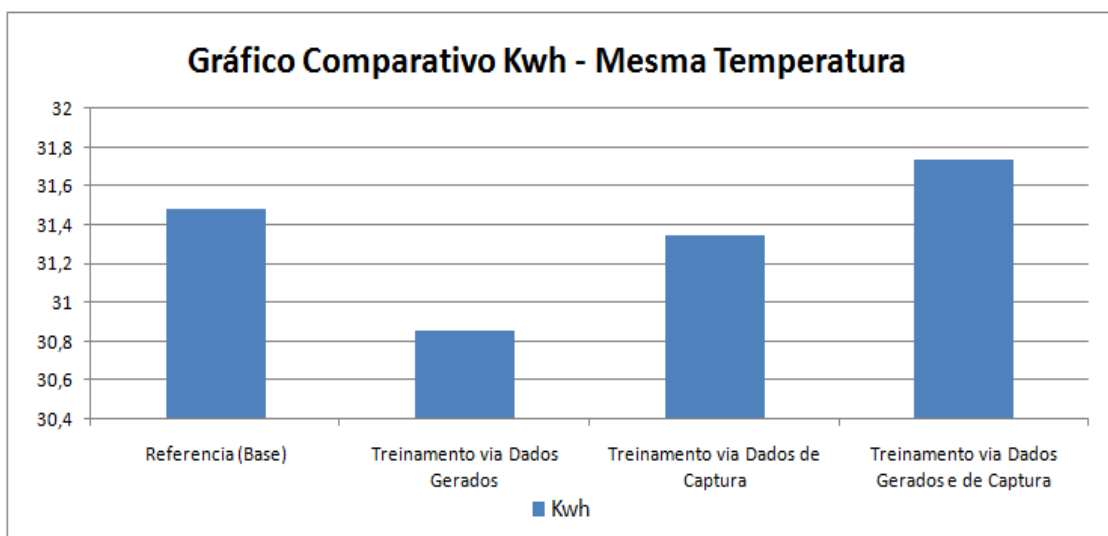
- O gasto total de energia é definido pelo parâmetro *Gasto Médio*, conforme a Figura 25.
- O conforto é definido como o menor valor do parâmetro *Desvio de Temperatura Total*, que é o total da diferença entre *Temperatura Desejada* e *Temperatura Média* de todos os cômodos do cenário, conforme dados contidos no Apêndice E.

5.4.2 Avaliação da Rede Neural

Essa seção vem com o objetivo de analisar resultados do uso de RNAs, contidos no Apêndice E, em relação aos dados de referencia base, a fim de se obter justificativa.

No ponto de economia de energia representado pelos dados em Apêndice C e D, as Figuras 26 e 27 foram apresentadas abaixo, com dados de cômodos de mesma temperatura e de temperaturas diferentes respectivamente.

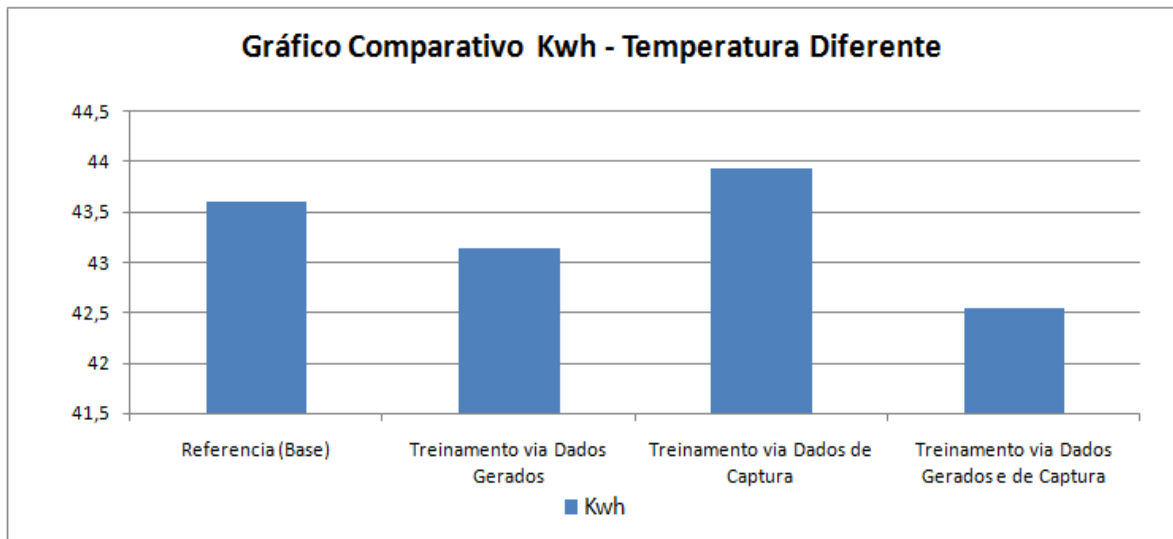
Figura 26: Gráfico comparativo kWh de cômodos de mesma temperatura.



Fonte: elaborado pelo autor.

Após análise da Figura 26, observou-se que com cômodos de mesma temperatura, uma rede neural com treinamento via dados gerados gastou um total de 0,62 kWh ou 2,01% a menos que a referência base, dados estes contidos no Apêndice C. A rede neural com treinamento via dados de captura não teve diferença significativa com a referência base. Já a rede neural com treinamento via dados gerados e de captura teve um gasto maior que a referência base.

Figura 27: Gráfico comparativo kWh de cômodos de temperaturas diferentes.

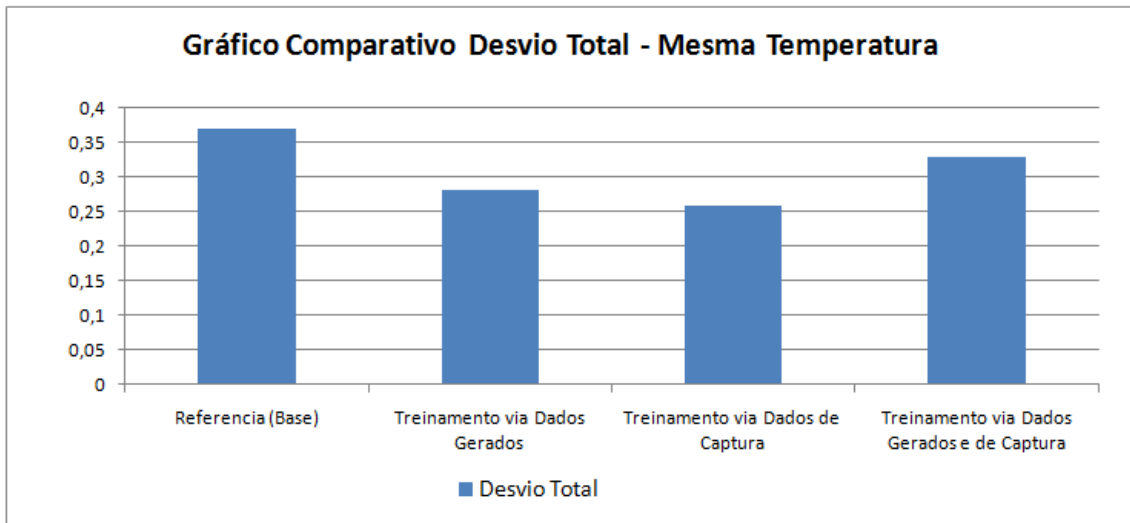


Fonte: elaborado pelo autor.

Com análise da Figura 27, observou-se que com cômodos de temperaturas diferentes, uma rede neural com treinamento via dados gerados e de captura gastou um total de 1,05 kWh ou 2,47% a menos que a referência base, dados estes contidos no Apêndice D. A rede neural com treinamento via dados gerados teve também uma pequena diminuição. Já a rede neural com treinamento via dados de captura teve um gasto de 0,77% mais de energia que a referência base.

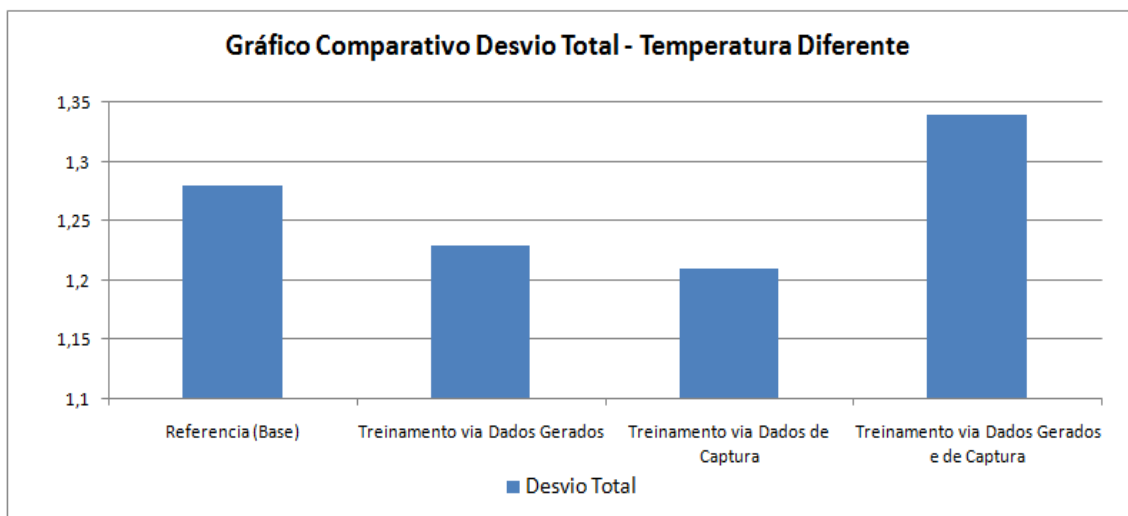
No ponto de conforto representado pelos dados em Apêndice C e D, as Figuras 28 e 29 foram apresentadas a seguir, com dados de cômodos de mesma temperatura e de temperaturas diferentes respectivamente.

Figura 28: Gráfico comparativo de Desvio Total de temperatura de cômodos de mesma temperatura.



Fonte: elaborado pelo autor.

Figura 29: Gráfico comparativo de Desvio Total de temperatura de cômodos de temperaturas diferentes.



Fonte: elaborado pelo autor.

Na Figura 28, o gráfico mostra uma melhora no conforto no uso de rede neural com treinamento via dados de captura. O mesmo ocorre na Figura 29.

O uso de rede neural com treinamento via dados gerados também teve uma melhora no conforto, porém menor, tanto na Figura 28 quanto na Figura 29.

O uso de rede neural com treinamento via dados gerados e de captura na Figura 28 não tiveram diferenças significativas, já na Figura 29, o gráfico mostrou que ocorreu um

desvio maior que a referência base, gerando assim, um desconforto maior.

5.4.3 Avaliação do Número de Neurônios na Camada Intermediária

O objetivo da avaliação é definir qual o melhor número de neurônios na camada intermediária atrelada a sua arquitetura.

Como o próprio erro no treinamento da rede neural é a definição de exatidão no momento de processamento do reconhecimento de padrão para a tomada de decisão, sendo essa propriedade fundamental para o problema proposto do trabalho.

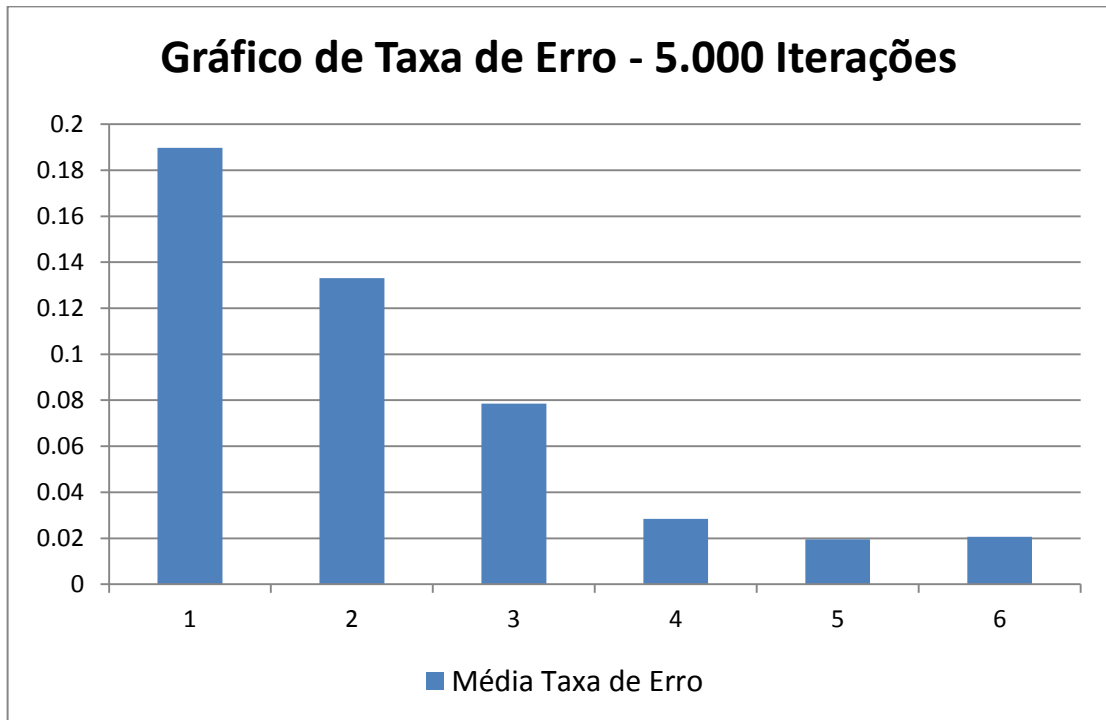
Em relação ao número de neurônios na camada intermediária, geralmente é definido empiricamente. O treinamento da rede neural, como definido no capítulo 4.3.2.2, foi executado em 20.000 iterações. O número de neurônios na camada intermediária foi de 5. Os parâmetros de arquitetura e treinamento não tiveram que ser alterados, pois os resultados obtidos foram satisfatórios, baseados na porcentagem de erro do treinamento.

Um dos problemas enfrentados é exceder o número de neurônios na camada intermediária, podendo causar um fenômeno chamado *overfitting*, que é a memorização de dados de treinamento na rede neural, causando resultados incorretos em sua execução. Em virtude disso, foi utilizado um número de neurônio baixo e necessário como apresentado nos próximos parágrafos.

Com o intuito de avaliar o número mínimo de neurônios na camada intermediária, foram utilizadas duas séries que também estão representadas no Apêndice A:

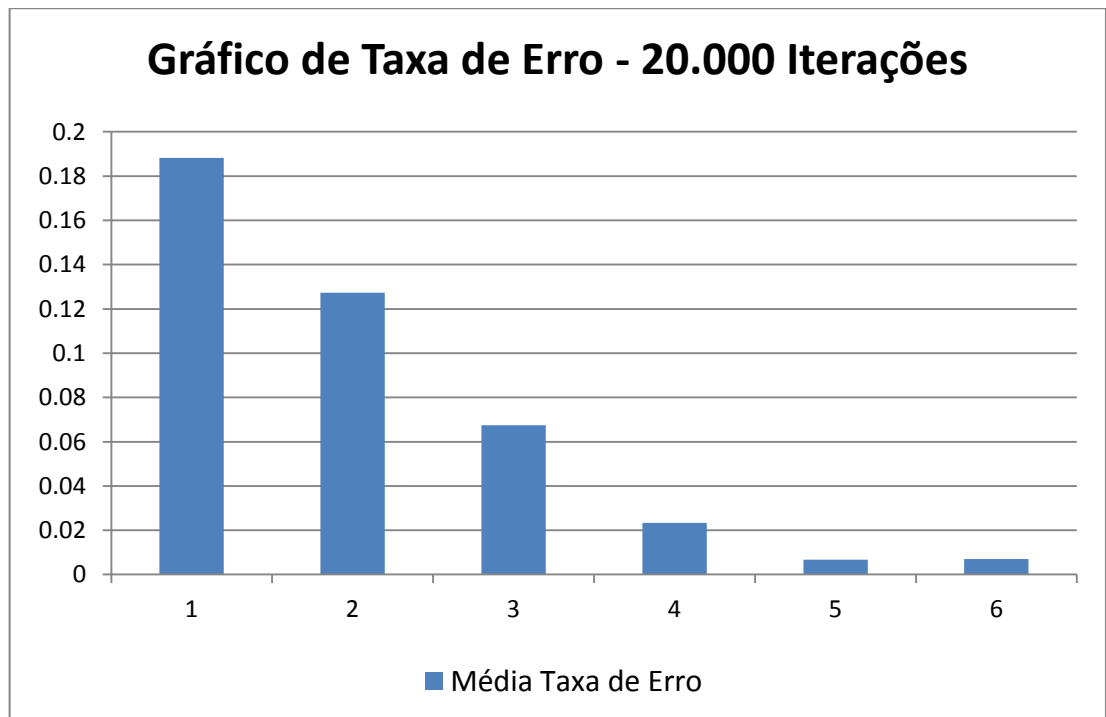
- Série 1: Figura 30, médias de erros de 3 tipos de amostras, sendo eles, dados de captura, dados de geração, dados de captura e geração, realizados no treinamento no total de 5.000 iterações;
- Série 2: Figura 31, médias de erros de 3 tipos de amostras, sendo eles, dados de captura, dados de geração, dados de captura e geração, realizados no treinamento no total de 20.000 iterações.

Figura 30: Gráfico da série de 5000 iterações.



Fonte: elaborado pelo autor.

Figura 31: Gráfico da série de 20.000 iterações.



Fonte: elaborado pelo autor.

Após a análise das Figuras 30 e 31, observou-se a necessidade da utilização de 5 neurônios na camada intermediária, a utilização de menos que 5 neurônios ocorre *underfitting* (a rede não converge durante seu treinamento), sendo que em 5 e 6 neurônios, ocorreu um estacionamento da taxa de erro mínima atingida, sendo assim, não ocorrendo mais o *underfitting* nesse intervalo. Também é visível uma menor taxa de erros na utilização de 5 neurônios com 20.000 iterações ao contrario de 5.000 iterações que apresentou uma taxa de erro maior.

Assim a utilização de 5 neurônios na camada intermediária e um treinamento com 20.000 iterações é a recomendada.

6 CONCLUSÃO

6.1 Considerações Finais

Após análise teórica, observações correlatas, desenvolvimento do sistema, simulação e testes, os resultados obtidos foram considerados satisfatórios.

No capítulo de desenvolvimento, a integração entre o simulador Home I/O e o sistema desenvolvido foi executada com sucesso, possibilitando a simulação de redes neurais no ambiente de simulação.

Visando economia de energia, tanto a rede neural treinada via dados gerados quanto à rede neural treinada via dados gerados e de captura, apresentaram um melhor resultado médio em comparação a referência base.

Visando conforto, a rede neural treinada via dados de captura também apresentou um melhor resultado em comparação a referência base.

As amostras demonstraram que ocorreu uma melhora tanto na economia de energia quanto no conforto com o uso de redes neurais para tomada de decisão.

Conclui-se que é viável a utilização de redes neurais artificiais para o controle de climatização, possibilitando melhoras em pontos como conforto e economia de energia.

6.2 Trabalhos futuros

Com o avanço da tecnologia sistemas complexos de tomada de decisão estarão mais próximos, tornando mais inteligente nosso dia-a-dia no futuro.

É importante destacar a utilização deste trabalho como base em futuros projetos na área de automação residencial, indicando que é possível ter um controle eficiente de uma residência com o uso de redes neurais artificiais.

A implantação de controle de resfriamento do ambiente junto ao sistema de climatização é um ponto a ser continuado, assim como a validação do uso da rede neural em ambientes reais.

Visando a forte tendência atual e futura da área de automação residencial, a utilização de rede neural no controle de outros dispositivos é considerável, assim sendo, viável para trabalhos futuros.

REFERÊNCIAS

ASHTON, Kevin. **Internet das Coisas, nova revolução da conectividade**. Porto Alegre:2014. Inovação em Pauta, Porto Alegre, n. 18, p. 6-9, 14 dez. 2014. Entrevista concedida a Inovação em Pauta. Disponível em: < <http://www.flip3d.com.br/web/pub/finep/>>. Acessado em; 28 de abr. 2016.

ATZORI, Luigi; IERA, Antonio; MORABITO, Giacomo. **The Internet of Things: A survey, 2010**. Computer Networks 54 (2010), p. 2787–2805.

BARRETO, Jorge Munis. **Inteligência Artificial no Limiar do Século XXI**, 2ª. Ed., Florianópolis: Duplic Edições, 1999.

BOTELHO, Wagner. T. **Um sistema de identificação e adaptação pervasivo para a casa inteligente utilizando sistemas multi-agentes**. Dissertação de Mestrado, Instituto Militar de Engenharia.

BRAGA, A. P.; CARVALHO; A. P. L. F.; LUDEMIR, T. B. **Redes neurais artificiais: teoria e aplicações**. 2.ed. Rio de Janeiro: LTC, 2007. 226p.

CYBENKO, G. **Approximation by superpositions of a sigmoid function**. *Mathematics of Control, Signals and Systems*, 2:303-314, 1989.

FERNANDES, Anita M. da Rocha. **Inteligência Artificial Noções Gerais**, Florianópolis, Visual Books, 2003.

FILIPATTI F. FRANCESCHINI G.; TASSONI C.; VAS P. (2000). **Recent developments of induction motor drives fault diagnosis using AI techniques**. *IEEE Transactions on Industrial Electronics*, v. 47, n°. 5, 994-1004.

GODOI, Ricardo. S. **AUTOMAÇÃO DE RESIDÊNCIAS. Uma análise da viabilidade da aplicação da domótica**. Guaratinguetá, 2009. 73p. Monografia, Faculdade de Tecnologia de Guaratinguetá.

HAYKIN, Simon. **Redes Neurais. Principio e prática.** Tradutor: Paulo Martins Engel. 2 edição. Porto Alegre: Bookman. 2001.

HEATON, J. **Programming Neural Networks with Encog 2 in Java.** Heaton Research, Inc., 2010. ISBN 1604390077, 9781604390070.

HEBB, D. O. **The Organization of Behavior: A Neuropsychological Theory.** New York: Wiley and Sons., 1949. ISBN 9780471367277.

HEINEN, Milton Roberto; OSÓRIO, Fernando Santos. **Aplicação de Redes Neurais Artificiais no Reconhecimento On-Line de Assinaturas.** WORKCAP – WORKSHOP DE COMPUTAÇÃO APLICADA DO PIPCA, 2002, São Leopoldo – RS. Anais... Unisinos, 2002. V.1. P.3-4.

HERTZ, J.; KROGH, A.; PALMER, R.G. **Introduction to Theory of Neural Computation.** Redwood City, CA; Addison-Wesley, 1991.

HOMEIO, 2016 . **Home I/O Documentation (English)** . Disponível em: <<https://realgames.co/documentation/pages/viewpage.action?pageId=1147689>> Acesso em: 09/11/2016.

KOVÁCS, Z.L. **Redes Neurais: Fundamentos e Aplicações.** 2. Ed. São Paulo: Ed. Acadêmica, 1996.

MCCULLOCH, W. S.; PITTS, W. H. **A logical calculus of the ideas immanent in nervous activity.** Bulletin of Mathematical Biology, v. 5, n. 4, p. 115-133, 1943.

MENDEL, J.M.; McLAREN, R.W. **Adaptive, learning, and pattern recognition systems; theory and applications, chapter Reinforcement-learning control and pattern recognition systems,** PP. 287-318. New York: Academic Press, 1970.

MICROSOFT, 2016. **About .NET.** Disponível em: <<https://docs.microsoft.com/en-us/dotnet/articles/about/>>. Acesso em: 08/11/2016.

MURIEL, F. Franco.; ANDERSON, P. Ferrugem.; CÉSAR, B. Silveira. Antônio; "**REDES NEURAI APLICADA NA AUTOMAÇÃO DE UMA RESIDÊNCIA ENERGETICAMENTE EFICIENTE**", p. 1395-1398 . In: In Proceedings of the 10th World Congress on Computational Mechanics [Blucher Mechanical Engineering Proceedings, v. 1, n. 1]. São Paulo: Blucher, 2014.

SENA, Diana. C. S. **Automação Residencial**. 2005. 109f. Monográfica (Graduação em Engenharia Elétrica) – Universidade Federal do Espírito Santo, Vitória, 2005.

SGARBI, Julio André. **Domótica Inteligente: Automação residencial baseada em comportamento**, 2007.

VALLE FILHO, Adhmar Maria. **Modelo para Implementação de Consciência em Robôs Móveis**, Tese (Doutorado) – Universidade Federal de Santa Catarina – UFSC Florianópolis, 2003.

VALIATI, João F. **Reconhecimento de Voz para Comandos de Direcionamento por Meio de Redes Neurais**. Porto Alegre: PPGC-UFRGS, 2000 (Ms. Thesis).

ZAMBARDA, Pedro. **‘Internet das Coisas’: entenda o conceito e o que muda com a tecnologia**, 2014.

APÊNDICES

APÊNDICE A – SÉRIES UTILIZADAS EM TESTE DE NEURÔNIO DA CAMADA INTERMEDIÁRIA.

Série 5.000 Iterações.

		CAPTURA	GERAÇÃO	CAPTURA + GERAÇÃO		Média
Neuronios	Epocas	Erro	Erro	Erro		
1	5000	0,1898	0,1901	0,1894		0,189767
2	5000	0,13	0,1329	0,1363		0,133067
3	5000	0,0741	0,084	0,0774		0,0785
4	5000	0,0347	0,0296	0,0209		0,0284
5	5000	0,0184	0,0177	0,0225		0,019533
6	5000	0,0218	0,0202	0,0199		0,020633
7	5000	0,0218	0,018	0,019		0,0196
8	5000	0,0176	0,0173	0,0217		0,018867
9	5000	0,0189	0,0221	0,0171		0,019367
10	5000	0,0186	0,0171	0,0176		0,017767
11	5000	0,0199	0,0174	0,0186		0,018633
12	5000	0,0165	0,0188	0,0167		0,017333
13	5000	0,018	0,0177	0,0164		0,017367
14	5000	0,0174	0,0174	0,0175		0,017433
15	5000	0,0165	0,0163	0,0175		0,016767
16	5000	0,0171	0,0177	0,0176		0,017467

Série de 20.000 Iterações.

		CAPTURA	GERAÇÃO	CAPTURA + GERAÇÃO		Média
Neuronios	Epocas	Erro	Erro	Erro		
1	20000	0,1882	0,1883	0,1882		0,188233
2	20000	0,1272	0,1274	0,1275		0,127367
3	20000	0,0674	0,0668	0,068		0,0674
4	20000	0,0362	0,0161	0,0178		0,023367
5	20000	0,0065	0,0068	0,007		0,006767
6	20000	0,0073	0,0066	0,0073		0,007067
7	20000	0,007	0,0064	0,0065		0,006633
8	20000	0,0073	0,0071	0,0071		0,007167
9	20000	0,0074	0,0067	0,0067		0,006933
10	20000	0,0079	0,0078	0,0067		0,007467
11	20000	0,0081	0,0084	0,0067		0,007733
12	20000	0,0067	0,0069	0,0085		0,007367
13	20000	0,0067	0,0067	0,0069		0,006767
14	20000	0,0067	0,0073	0,0069		0,006967
15	20000	0,0073	0,0083	0,0069		0,0075
16	20000	0,0077	0,0084	0,083		0,033033

APÊNDICE B – DADOS DE REFERÊNCIA BASE OBTIDOS NA SIMULAÇÃO

DADOS DE REFERENCIA (BASE) - MESMAS TEMPERATURAS			
Cômodo	Temperatura Desejada (°C)	Temperatura Média (°C)	Desvio Temperatura Total (°C)
A	20	20,09	0,09
D	20	20,11	0,11
E	20	20,06	0,06
G	20	20,11	0,11
Total:		20,0925	0,37
Gasto(kWh):	31,48		

DADOS DE REFERENCIA (BASE) - TEMPERATURAS DIFERENTES			
Cômodo	Temperatura Desejada (°C)	Temperatura Média (°C)	Desvio Temperatura Total (°C)
A	23	23,06	0,06
D	21	21,86	0,86
E	22	21,82	-0,18
G	20	20,18	0,18
Total:		21,73	1,28
Gasto(kWh):	43,6		

APÊNDICE C – DADOS COMPARATIVOS DE MESMA TEMPERATURA

DADOS COMPARATIVOS DE MESMA TEMPERATURA				
Tipo de treinamento	Gasto Médio Kwh	% kWh Referencia Base	Desvio Total Temperatura	% Desvio Total Temperatura Base
Referencia (Base)	31,48	0%	0,37	0,00%
Treinamento via Dados Gerados	30,86	2,01%	0,281	31,67%
Treinamento via Dados de Captura	31,35	0,41%	0,26	42,31%
Treinamento via Dados Gerados e de Captura	31,74	-0,82%	0,33	12,12%

APÊNDICE D – DADOS COMPARATIVOS DE TEMPERATURAS DIFERENTES

DADOS COMPARATIVOS DE TEMPERATURAS DIFERENTES				
Tipo de treinamento	Gasto Médio Kwh	% kWh Referencia Base	Desvio Total Temperatura	% Desvio Total Temperatura Base
Referencia (Base)	43,6	0,00%	1,28	0,00%
Treinamento via Dados Gerados	43,14	1,07%	1,23	4,07%
Treinamento via Dados de Captura	43,94	-0,77%	1,21	5,79%
Treinamento via Dados Gerados e de Captura	42,55	2,47%	1,34	-4,48%

APÊNDICE E – DADOS OBTIDOS NA SIMULAÇÃO

TREINAMENTO VIA DADOS GERADOS - MESMAS TEMPERATURAS			
Cômodo	Temperatura Desejada (°C)	Temperatura Média (°C)	Desvio Temperatura Total (°C)
A	20	19,898	-0,102
D	20	19,893	-0,107
E	20	19,953	-0,047
G	20	19,975	-0,025
Total:		19,92975	0,281
Gasto(kWh):	30,86		

TREINAMENTO VIA DADOS GERADOS- TEMPERATURAS DIFERENTES			
Cômodo	Temperatura Desejada (°C)	Temperatura Média (°C)	Desvio Temperatura Total (°C)
A	23	22,82	-0,18
D	21	21,79	0,79
E	22	21,79	-0,21
G	20	19,95	-0,05
Total:		21,5875	1,23
Gasto(kWh):	43,14		

TREINAMENTO VIA DADOS DE CAPTURA - MESMAS TEMPERATURAS			
Cômodo	Temperatura Desejada (°C)	Temperatura Média (°C)	Desvio Temperatura Total (°C)
A	20	20,17	0,17
D	20	20,01	0,01
E	20	20,06	0,06
G	20	19,98	-0,02
Total:		20,055	0,26
Gasto(kWh):	31,35		

TREINAMENTO VIA DADOS DE CAPTURA- TEMPERATURAS DIFERENTES			
Cômodo	Temperatura Desejada (°C)	Temperatura Média (°C)	Desvio Temperatura Total (°C)
A	23	23,16	0,16
D	21	21,9	0,9
E	22	21,87	-0,13
G	20	20,02	0,02
Total:		21,7375	1,21
Gasto(kWh):	43,94		

TREINAMENTO VIA DADOS GERADOS E DE CAPTURA - MESMAS TEMPERATURAS			
Cômodo	Temperatura Desejada (°C)	Temperatura Média (°C)	Desvio Temperatura Total (°C)
A	20	19,9	-0,1
D	20	19,94	-0,06
E	20	20,01	0,01
G	20	19,84	-0,16
Total:		19,9225	0,33
Gasto(kWh):	31,74		

TREINAMENTO VIA DADOS GERADOS E DE CAPTURA- TEMPERATURAS DIFERENTES			
Cômodo	Temperatura Desejada (°C)	Temperatura Média (°C)	Desvio Temperatura Total (°C)
A	23	22,81	-0,19
D	21	21,85	0,85
E	22	21,86	-0,14
G	20	19,84	-0,16
Total:		21,59	1,34
Gasto(kWh):	42,55		