UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

DEPARTAMENTO ACADÊMICO DE ELETROTÉCNICA

ENGENHARIA DE CONTROLE E AUTOMAÇÃO

GERHARD DIX JORDAN

LUIZ FELIPE MIRANDA LIMA

RAFAEL DE MOURA MARQUES DOS SANTOS

CLASSIFICADOR FUZZY DO GRAU DE SEVERIDADE DA SOBRECARGA EM MOTORES DE INDUÇÃO UTILIZANDO UM MICROCONTROLADOR

TRABALHO DE CONCLUSÃO DE CURSO

CURITIBA

2018

GERHARD DIX JORDAN

LUIZ FELIPE MIRANDA LIMA

RAFAEL DE MOURA MARQUES DOS SANTOS

CLASSIFICADOR FUZZY DO GRAU DE SEVERIDADE DA SOBRECARGA EM MOTORES DE INDUÇÃO UTILIZANDO UM MICROCONTROLADOR

Trabalho de conclusão de curso de graduação apresentado à disciplina de TCC 2, do Curso Superior em Engenharia de Controle e Automação do Departamento Acadêmico de Eletrotécnica – DAELT – da Universidade Tecnológica Federal do Paraná – UTFPR, como requisito parcial para obtenção do título de Engenheiro de Controle e Automação.

Orientador: Prof. Ednilson Soares Maciel, Msc

CURITIBA

2018

GERHARD DIX JORDAN

LUIZ FELIPE MIRANDA LIMA

RAFAEL DE MOURA MARQUES DOS SANTOS

CLASSIFICADOR FUZZY DO GRAU DE SEVERIDADE DA SOBRECARGA EM MOTORES DE INDUÇÃO UTILIZANDO UM MICROCONTROLADOR

Este trabalho de Conclusão de curso de graduação foi julgado e aprovado como requisito parcial para a obtenção do título de Engenheiro de Controle e Automação, do curso de Engenharia de controle e Automação do Departamento Acadêmico de Eletrotécnica (DAELT), da Universidade Tecnológica Federal do Paraná (UTFPR).

Curitiba, junho de 2018

_____
Prof. Paulo Sérgio Walênia
Coordenador de Curso
Engenharia de Controle e Automação

_____
Profa. Annemarlen Gehrke Castagna, Dr.
Responsável pelos Trabalhos de Conclusão de Curso do DAELT

ORIENTAÇÃO

_____
Ednilson Soares Maciel, Msc.
Universidade Tecnológica Federal do Paraná
Orientador

BANCA EXAMINADORA

_____
Joaquim Eloir Rocha, Dr.
Universidade Tecnológica Federal do Paraná

_____
Emerson Rigoni, Dr.
Universidade Tecnológica Federal do Paraná

A folha de aprovação assinada encontra-se na Coordenação do Curso de Engenharia Elétrica

Resumo

JORDAN, Gerhard D; LIMA, Luiz F M; SANTOS, Rafael D M M D. **CLASSIFICADOR FUZZY DO GRAU DE SEVERIDADE DA SOBRECARGA EM MOTORES DE INDUÇÃO UTILIZANDO UM MICROCONTROLADOR.** 2018. 106f. Trabalho de Conclusão de Curso (Graduação) – Engenharia de Controle e Automação – Universidade Tecnológica Federal do Paraná, 2018.

Nos ambientes industriais um dos grandes desafios é manter as máquinas motrizes de forma que produzam o máximo da sua capacidade com o mínimo de recursos. É indispensável que seja mantida a disponibilidade destas, porém muitas são as variáveis que envolvem o seu funcionamento, nas quais o motor de indução tem uma participação significativa. Na aplicação da manutenção preditiva, existe sempre a necessidade de um especialista para avaliar os dados coletados. Neste trabalho foi avaliada a técnica de se adquirir os valores absolutos das correntes do estator juntamente à temperatura de cabeça de bobina. Com isso, foi possível inseri-las em um *software* especialista *Fuzzy* confeccionado a partir de um banco de dados colhido através de entrevistas com um especialista. Com a utilização da coleta de dados *online* juntamente com o algoritmo *Fuzzy* permite-se diagnosticar se há alguma anomalia no funcionamento do motor, gerando assim uma melhor capacidade de tomada de decisão do nível gerencial, pois se sabe o estado de funcionamento do motor pelo grau de severidade da operação. Ao Final da implementação, a veracidade dos resultados obtidos foi verificada através da *toolbox* do Matlab, obtendo resultados similares que validaram os testes e confirmaram o sucesso da proposta.

Palavras chave: Motor de indução, manutenção preditiva, *Fuzzy, software* especialista, aquisição de dados *online*, Arduino.

Abstract

JORDAN, Gerhard D; LIMA, Luiz F M; SANTOS, Rafael D M M D. **Fuzzy classification of overload severity degree on induction motors using a microcontroller.** 2018. 106p. Trabalho de Conclusão de Curso (Graduação) – Engenharia de Controle e Automação – Universidade Tecnológica Federal do Paraná, 2018.

In the industrial environment one of the great challenges is to keep the driving machines in order to deliver the maximum capacity with minimal resources. It is essential to keep the high availability of them, but there are many variables involved in this operation, in which the induction motor has a high role. In the application of the predictive maintenance there is always a necessity of having a specialist to analyze the collected data. In this work was evaluated the technique of acquiring the absolute values of current of the stator along with the coil head temperature and inputting them in a Fuzzy specialist software programed from a data base collected through interviews with a specialist, using online data collection. With a Fuzzy algorithm, it allow us to diagnose if there is any anomaly in the working of the motor, thus generating a better decision making capability at the management level, because the motor condition is known through his grade of severity of operation. At the end of the implementation, the veracity of the obtained results was verified with the Matlab toolbox, getting similar results that validated the tests and confirmed the success of the proposed.

Key words: Induction motor, predictive maintenance, Fuzzy, specialist software, online data acquisition, Arduino.

LISTA DE SIGLAS

ABNT: Associação Brasileira de Normas Técnicas;

CC: Corrente Contínua;

EFLL: *Embedded Fuzzy Logic Library* (Biblioteca Embarcada de Lógica *Fuzzy*);

EPVA: *Extended Park Vector Approach* (Abordagem Estendida do Vetor de Park);

ESA: *Electrical Signature Analysis* (Análise da Assinatura Elétrica);

FCEM: Força Contra Eletromotriz;

IDE: Integrated Development Environment (Ambiente de Desenvolvimento Integrado);

IPSA: *Instantaneous Power Signature Analysis* (Análise Espectral da Potência Instantânea);

MIT: Motor de indução trifásico;

MPd: Manutenção preditiva;

MCSA: *Motor Current Signature Analysis* (Assinatura de Corrente do Motor);

NBR: Norma Brasileira Regulamentadora;

NEMA*: National Electrical Manufacturers Association* (Associação Nacional de Fabricantes Elétricos);

Ts: Temperatura de serviço;

VDI: *Voltage Distortion and Imbalance* (Desbalanceamento e Distorção de Tensão)*;*

VSA: *Voltage Signature Analisys* (Análise do Espectro de Tensão);

VUF: *Voltage Unbalance Factor* (Fator de Desbalanceamento de Tensão);

LISTA DE FIGURAS

Figura 2.1: Vista Explodida de um Motor de Indução. 16
Figura 2.2: Rotor do Motor de Indução do tipo Gaiola de Esquilo 17
Figura 2.3: Máquina de Indução Assíncrona 18
Figura 2.4: Máquina de Indução Assíncrona 19
Figura 2.5: Circuito Equivalente do Estator de um MIT. 20
Figura 2.6: Circuito Equivalente Monofásico de um MIT Polifásico 22
Figura 2.7: Normalização de dados de modos de falha em motores de indução 23
Figura 2.8: Faixas de temperatura de serviço 28
Figura 3.1: Núcleo, suporte e limite de um conjunto Fuzzy 35
Figura 3.2: Complemento, interseção e união de conjuntos Fuzzy 37
Figura 3.3: Partição de conjuntos Fuzzy 38
Figura 3.4: Modificadores linguísticos Fuzzy 39
Figura 3.5: Diagrama típico de um modelo de inferência de Mamdani 41
Figura 3.6: Exemplo de um processo de inferência Fuzzy 41
Figura 4.1: Dados de placa do MIT de estudo 47
Figura 5.1: Datasheet YDHC SCT-013-020 55
Figura 5.2: Divisor de tensão e capacitor referentes ao primeiro sensor 56
Figura 5.3: Esquema de ligação dos sensores ao Arduino 57
Figura 5.4: Multimedidor Trifásico Mult - K 57
Figura 5.5: Função de Pertinência para Temperatura 64
Figura 5.6: Função de pertinência para as três correntes 64
Figura 5.7: Função de pertinência para a severidade 64
Figura 6.1: MIT acoplado a um motor CC 70
Figura 6.2: Potência X Conjugado mecânico 71
Figura 6.3: Simulação para motor a vazio 72
Figura 6.4; Simulação de falta de fase com motor a vazio 73

LISTA DE TABELAS

Tabela 3.1: Propriedades dos conjuntos Fuzzy.....36
Tabela 6.1: Tabela de expectativa e resultados.....80

SUMÁRIO

1. INTRODUÇÃO 11
1.1. TEMA 11
1.2. DELIMITAÇÂO DO TEMA 12
1.3. PROBLEMAS E PREMISSAS 12
1.4. OBJETIVOS 13
1.4.1. Objetivo Geral 13
1.4.2. Objetivos Específicos 14
1.5. JUSTIFICATIVA 14
1.6. PROCEDIMENTOS METODOLÓGICOS 15
2. REVISÃO TEÓRICA 16
2.1. MOTORES DE INDUÇÃO TRIFÁSICOS (MITs) 16
2.1.1. Modelagem de um MIT 18
2.1.2. Defeito e Falha 22
2.1.3. Principais defeitos em MITs 23
2.1.4. Qualidade de Energia Elétrica 24
2.1.4.1. Desequilíbrio de Tensão 25
2.1.5. Problemas de vibração 27
2.1.6. Problemas de Temperatura 28
2.1.6.1. Efeitos da temperatura no escorregamento 29
2.2. TÉCNICAS DE MANUTENÇÃO PREDITIVA 31
2.2.2. Análise de assinatura de corrente do motor (MCSA) 32
***3.* LÓGICA FUZZY 34**
3.1. INTRODUÇÃO 34
3.2. DEFINIÇÕES DOS CONJUNTOS *FUZZY* 34
3.3. PROPRIEDADES DOS CONJUNTOS *FUZZY* 36
3.4. OPERAÇÕES *FUZZY* 36
3.5. VARIÁVEIS E MODIFICADORES LINGUÍSTICOS 37
3.6. REGRAS DE PRODUÇÃO *FUZZY* 39
3.7. PROCESSO DE INFERÊNCIA *FUZZY* 40
3.7.1. Avaliação dos Antecedentes 42
3.7.2. Implicação 42
3.7.3. Agregação dos consequentes 43
3.7.4. Condensação dos consequentes 43
3.8. SÍNTESE DO CAPÍTULO 44
4. MATERIAIS E MÉTODOS 45

4.1. OBJETO DE ESTUDO 45
4.2. DELINEAMENTO 45
4.3. DETALHAMENTO DE PROCEDIMENTOS E MATERIAIS 46
4.3.1. Procedimento Básico 46
4.3.2. Hardwares 47
4.3.2.1. Sensor de corrente 48
4.3.2.2. Sensor de temperatura 49
4.3.2.3. Placa microcontroladora Arduino 50
4.3.3. Diagrama de Ligação dos hardwares 51
4.3.4. Softwares 52
4.3.4.1. Ambiente de programação (Arduino IDE) 53
4.3.4.2. Matlab 53
4.4. ANÁLISE DOS DADOS 54
4.5. SÍNTESE DO CAPÍTULO 54
5. AQUISIÇÃO DE DADOS E IMPLEMENTAÇÃO DA LÓGICA *FUZZY* 55
5.1. AQUISIÇÃO DOS VALORES DA CORRENTE DO MOTOR 55
5.1.1. Implementação da Lógica para Aquisição das Correntes dos Motores no Arduino 58
5.1.2. Descrição do Código que Calcula a Corrente Eficaz 58
5.2. AQUISIÇÃO DA TEMPERATURA 59
5.2.1. Implementação do Código Referente ao Sensoriamento da Temperatura 62
5.3. IMPLEMENTAÇÃO DA LÓGICA FUZZY NO ARDUINO 63
5.3.1. Biblioteca Effl 63
5.3.2. Delimitação das Funções de Pertinência 63
5.3.3. Declaração das funções de pertinência no *software* 65
5.3.4. Confecção Das Regras 65
5.3.5. Declaração das Regras no *Software* 67
5.3.6. Montando Estrutura de Repetição 68
5.4. SÍNTESE DO CAPÍTULO 69
6. ENSAIOS E ANÁLISE DOS RESULTADOS 70
6.1. DETERMINAÇÃO DE INTERVALOS DE OPERAÇÃO 70
6.2. ENSAIOS, SIMULAÇÕES E RESULTADOS 71
6.2.1. Ensaio com Motor a Vazio 72
6.2.2. Ensaio de Falta de fase com motor a vazio 73
6.2.3. Ensaio de Desequilíbrio de tensão em plena carga 74
6.2.4. Ensaio com Fator de serviço a 115% 75
6.2.5. Ensaio de ventilação bloqueada em fator de serviço 115% 77

6.3. ANÁLISE DOS RESULTADOS 80
7. CONCLUSÃO 83
REFERÊNCIAS 85
APÊNDICES 89
Apêndice A 89

1. INTRODUÇÃO

1.1. TEMA

De acordo com levantamento do Ministério de Minas e Energia (MME) referente ao Plano Nacional de Eficiência Energética, as indústrias lideram o consumo de energia elétrica no país, representando 43,7% de participação entre todos os setores da economia, citando transportes, residencial, energético, etc. Um levantamento publicado, também pelo MME, destacou que a Força Motriz na indústria, incluindo bombas, ventiladores, compressores, entre outros, em geral motores elétricos, são os responsáveis por 68% do consumo, sendo assim, constata-se que aproximadamente 30% de toda a energia elétrica do Brasil é consumida por motores elétricos. Fica evidente que a Força Motriz tem uma participação importante no funcionamento das indústrias (MME, 2015).

Faz-se necessário a manutenção e o constante monitoramento dos motores elétricos para obterem-se os menores índices de defeitos e assim a maior disponibilidade das máquinas e equipamentos em que eles estão associados. Em função disso, ao longo dos anos, foram desenvolvidas várias técnicas para monitoramento do funcionamento, que são chamadas de Técnicas Preditivas, técnicas não invasivas que visam detectar tanto os defeitos elétricos quanto mecânicos antes das falhas de fato ocorrerem.

Para Carvalho (2006), um sistema especialista é um sistema computacional que emula a estratégia de resolução de problemas de um humano com grande conhecimento do assunto. Sistemas especialistas são programas destinados a solucionar problemas em campos específicos do conhecimento. Apesar de que este não pode chegar a possuir a capacidade cognitiva de um humano, na ausência deste, constitui-se uma ferramenta importante de resolução de problemas.

Segundo Earl Cox (1998), uma lógica matemática muito conhecida nos processos de auxílio à tomada de decisão é a lógica *fuzzy*. Também chamada de nebulosa ou difusa, esta lógica tem como base a análise de dados não digitalizados, ou seja, considera valores intermediários entre o máximo (“um” na lógica Booleana) e mínimo (“zero” na lógica Booleana). Analisar dados utilizando tal lógica significa

considerar os estágios compreendidos entre os extremos de determinadas entradas para que através da inferência de seus valores, chegue-se a uma ou mais saídas também com valores não digitais. A partir de regras, normalmente estabelecidas por especialistas no processo a ser analisado, pode-se mapear o comportamento do sistema.

Tendo em vista esta necessidade de monitoramento constante, propõe-se a implementação de um sistema especialista que monitore em tempo real as amplitudes das correntes estatóricas e a temperatura de uma máquina de indução para que este possa classificar o grau de severidade dos defeitos de operação a partir de uma análise com a utilização de uma lógica *Fuzzy*.

1.2. DELIMITAÇÂO DO TEMA

O tema deste trabalho se dá pela realização de um algoritmo *Fuzzy* implementado em Arduino, tendo como base os dados apresentados no artigo: "*A Simple Fuzzy Logic Approach for Induction Motors Stator Condition Monitoring*", feito pelos autores M. Zeraoulia, A. Mamoune, H. Mangel, M.E.H. Benbouzid em 2005, pelo qual foi feita uma análise de uma possível implementação de um algoritmo *Fuzzy* em Matlab para detecção de modos de falhas, analisando as correntes de cada fase e agora com o adicional da variável temperatura das bobinas de cabeça do motor.

1.3. PROBLEMAS E PREMISSAS

A manutenção preditiva (MPd) na indústria interpõe técnicas de monitoramento instrumentado e/ou sensitivo para tentar determinar o momento da falha funcional. Nos motores de indução trifásicos (MITs) existem diversos problemas mecânicos e elétricos que são detectados com o correto monitoramento preditivo, como desgastes e desbalanceamento nos rolamentos, folga mecânica, deslocamento do centro magnético (excentricidade), quebra de barras do rotor, problemas no

enrolamento do estator, degradação do isolamento, desbalanceamento de tensões e correntes trifásicas, etc. Esse monitoramento, de um modo mais sistêmico, é realizado periodicamente por empresas contratadas, e com a combinação de vários testes, como termografia, análise de vibrações, entre outros métodos, permitem prevenir a ocorrência dessas anomalias e realizar uma manutenção programada, que reduziria, e muito, o tempo perdido com a ocorrência um defeito não previsto. (Adaptado de SILVA, 2008).

Este trabalho tem como premissa verificar se a utilização de um sistema especialista em tempo real, que utiliza a logica *fuzzy*, pode melhorar a confiabilidade e consequentemente facilitar a tomada de decisões pelo gestor de uma equipe de manutenção.

Segundo Santos (2007), atualmente nas indústrias, a manutenção preditiva depende de rondas periódicas para determinação do estado em que as máquinas estão operando. Poderia um software especialista que faça a aquisição destes dados em tempo real e aplicando-os em um algoritmo *fuzzy,* oferecer uma base mais sólida para o auxílio à tomada de decisão por parte do responsável da manutenção de uma empresa?

1.4. OBJETIVOS

1.4.1. Objetivo Geral

Implementar um sistema especialista em Arduino, utilizando a lógica *Fuzzy* para detecção de defeitos em motores de indução, utilizando como base o artigo “*A Simple Fuzzy Logic Approach for Induction Motors Stator Condition Monitoring*” com a adição de mais uma variável de entrada, para determinação em tempo real da severidade de operação.

1.4.2. Objetivos Específicos

- Revisar a literatura pertinente;

- Descrição dos possíveis defeitos em um motor de indução e suas causas;

- Estruturar as regras *Fuzzy* que serão utilizadas a partir do conhecimento tácito de um especialista;

- Execução do algoritmo *Fuzzy* utilizando a *toolbox* do Matlab;

- Implementação das regras *Fuzzy* no Arduino;

- Testes on-line do sistema para algumas condições pré-estabelecidas, para validação do protótipo.

1.5. JUSTIFICATIVA

O trabalho visa o estudo e o desenvolvimento de um projeto de fácil implementação e utilização, sendo que toda a sua estrutura terá o mesmo formato para diferentes aplicações. O projeto poderá ser empregado para atividades de ensino e em indústrias, seja pelo suporte à equipe de manutenção, podendo reduzir significativamente o tempo de treinamentos, quanto pelo auxílio na tomada de decisões relacionadas aos ensaios que serão realizados neste trabalho.

Assim, busca-se desenvolver um software integrado simples com aquisição de dados, trazendo um método de fácil implementação em qualquer ambiente fabril que necessite de manutenção preditiva em MITs.

1.6. PROCEDIMENTOS METODOLÓGICOS

Para a execução do trabalho será realizada uma pesquisa sobre os tipos de defeitos em um motor elétrico que podem ser evidenciados por uma diferença no valor absoluto das correntes do estator e da temperatura de serviço.

A partir desta análise, será feito um levantamento de como estes modos de falha podem ser observados por uma lógica *Fuzzy* programada em um micro controlador arduino, para que este auxilie na tomada de decisão do quão grave e tolerável é o estado de operação do motor.

Após a coleta e análise destes dados, será implementado na interface programável deste microcontrolador o software especialista *Fuzzy* que recebe as informações de entrada, analisa estes dados via base científica e devolve uma resposta com o grau de severidade esperado, facilitando a tomada de decisão da manutenção preditiva através dos leds sinalizadores.

2. REVISÃO TEÓRICA

2.1. MOTORES DE INDUÇÃO TRIFÁSICOS (MITs)

Um motor de indução trifásico é uma máquina robusta, simples e composta de partes elétricas e mecânicas que permitem o seu correto funcionamento, como mostrado na Figura 2.1.

Figura 2.1: **Vista Explodida de um Motor de Indução.**

Fonte: Disponível em <http://zondatec.blogspot.com.br/2014/04/caracteristicas-do-motor-de-inducao.html>. Acesso em 30 Out 2017, 14:48

Dentre as partes mostradas na Figura 2.1, visualizam-se diversos componentes que podem apresentar falhas durante a vida útil do motor, das quais serão verificadas algumas dentro deste trabalho.

O motor de estudo deste trabalho é o motor de indução assíncrono trifásico com rotor do tipo gaiola de esquilo que consiste em um rotor formado por barras condutoras encaixadas nas ranhuras do rotor e curto circuitadas por anéis

condutores em suas extremidades, como pode ser visto na Figura 2.2 (Fitzgerald, 2006). Neste motor o rotor não é excitado eletricamente pelo estator, ou seja, não há conexão elétrica entre o rotor e estator, e a velocidade de funcionamento não é proporcional à frequência de alimentação (a velocidade de funcionamento é sempre menor que a do campo girante produzido pelas bobinas devido ao escorregamento). Esta a razão pelo motor ser chamado de assíncrono, por não ter a mesma velocidade síncrona da rede de alimentação.

Figura 2.2: Rotor do Motor de Indução do tipo Gaiola de Esquilo

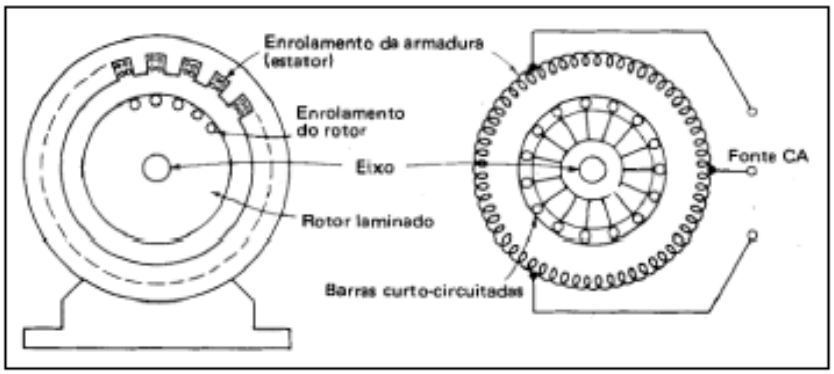
Fonte: Fitzgerald (2006, p 297).

Segundo Fitzgerald, quando o estator do motor é ligado a uma fonte trifásica equilibrada, este gera um campo magnético girante com velocidade síncrona que é determinada pelo número de polos do estator e a frequência da fonte. Este campo, por sua vez, induz no rotor uma força contra eletromotriz que o põe a girar, porém em uma velocidade menor que a velocidade síncrona uma vez que existem perdas inerentes às partes mecânicas móveis do motor. Essa diferença de velocidades é dada pelo escorregamento, geralmente expresso como uma fração da velocidade síncrona, dado pela equação 1.

$$S = \frac{n_s - n_r}{n_s} \quad (1)$$

Onde n_s é a velocidade síncrona do campo do estator e n_r é a velocidade rotórica. Na Figura 2.3 pode ser visto, à esquerda, uma vista em corte da máquina de indução e, à direita, os detalhes das conexões elétricas:

Figura 2.3: Máquina de Indução Assíncrona



Fonte: Barbi (2011).

Estas máquinas são utilizadas em grande parte das aplicações industriais, para movimentar cargas girantes e lineares, transformar energia elétrica em energia mecânica para realizar trabalho. É também uma boa opção para acionamentos controlados, pois possui algumas vantagens sobre o motor de corrente contínua, entre elas:

- Custo menor do que um motor CC de mesma potência;
- Manutenção simples;
- Menor consumo de energia nos processos de aceleração e frenagem;
- Pode-se obter velocidades maiores no MIT, consequentemente potências maiores.

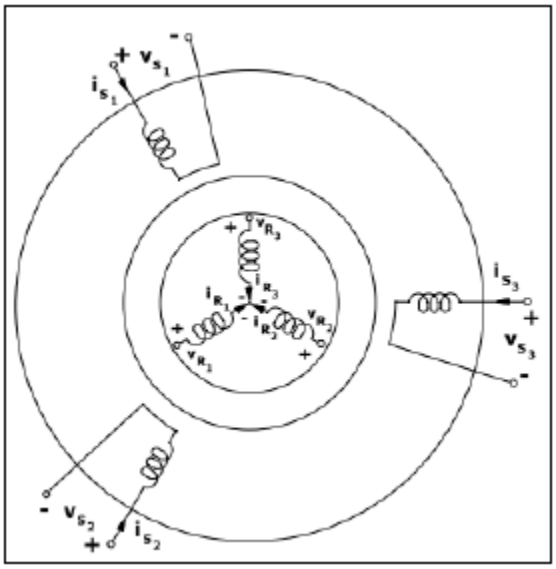
Faz-se necessário então, melhorar as condições de monitoramento das condições de operação destas máquinas para que estas continuem gerando rentabilidade para a empresa.

2.1.1. Modelagem de um MIT

Para fins de modelagem é necessário considerar uma máquina de indução trifásica simétrica e com rotor bobinado, ou seja, as bobinas tanto do rotor quanto do

estator são iguais entre si e possuem defasamentos iguais conforme na Figura 2.4 a seguir (BARBI, 2011).

Figura 2.4: Máquina de Indução Assíncrona



Fonte: Barbi (2011).

Para estudar esta máquina, é necessário partir de algumas premissas, a fim de obter as condições necessárias para a modelagem matemática.
Premissas de Estudo:

- Os três enrolamentos do rotor e os três enrolamentos do estator são iguais entre si;
- Os ângulos elétricos entre os enrolamentos são iguais tanto no rotor quanto no estator;
- Considerar o entreferro (distância entre o rotor e estator) constante;
- Desprezar a saturação do material ferromagnético;
- Não considerar as perdas magnéticas;
- Para simplificar o modelo, considerar uma máquina bipolar.

Para gerar o modelo do motor de indução polifásico, deve-se primeiramente chegar à dedução de seu circuito equivalente, e a partir daí equacionar suas variáveis.

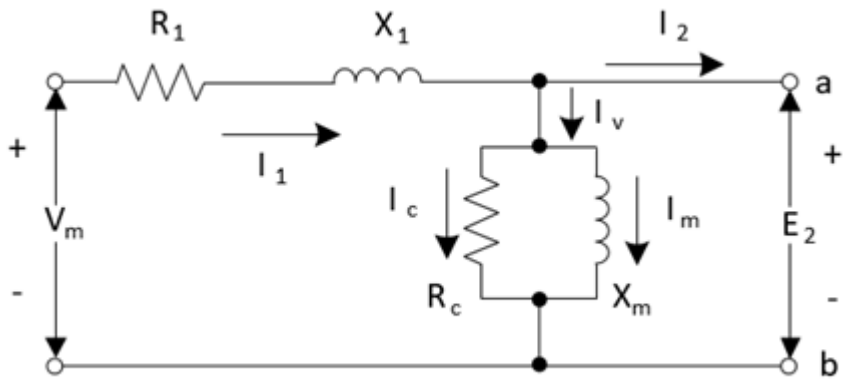
Considerando as condições do estator, a variação de fluxo eletromagnético no entreferro, variando sincronicamente gera as forças contra eletromotrizes (FCEMs) equilibradas nas fases do estator. A tensão terminal do motor, no entanto, deve ser igual à FCEM acrescida às perdas dada pela impedância de dispersão do estator, conforme equação 2.

$$V_1 = E_2 + I_1(R_1 + jX_1) \quad (2)$$

Onde:
$V1$: Tensão de fase de terminal de estator;
$E2$: FCEM (de fase) gerada pelo fluxo do entreferro resultante;
$I1$: Corrente do estator;
$R1$: Resistencia efetiva do estator;
$X1$: Reatância de dispersão do estator.

Semelhante a um transformador, os fluxos magnéticos do estator induzem uma corrente no rotor esta indução gera uma componente de excitação que ainda pode ser subdividida em perdas no núcleo (Rc) e uma componente de magnetização (Xm) estas componentes dependem da frequência nominal da máquina, porém podem ser consideradas constantes caso haja pequenos desvios em $E2$:

Figura 2.5: Circuito Equivalente do Estator de um MIT.



Fonte: Fitzgerald (2008).

Basicamente, o modelo do estator é igual ao circuito equivalente do primário de um transformador. Para completarmos o modelo da máquina assíncrona, os efeitos do rotor devem ser considerados, do ponto de vista do circuito equivalente do

estator, o rotor pode ser representado como uma impedância equivalente Z2 que corresponde à impedância de dispersão de um secundário de um transformador. Como os terminais no rotor de gaiola de esquilo são curto circuitados, a tensão induzida percebe somente uma impedância de curto circuito, porém para o estator existe uma relação que considera o escorregamento do rotor e a frequência, demonstrado na figura 3.

$$Z_{2s} = \frac{E_{2s}}{I_{2s}} = N_{ef}^2 \left(\frac{E_{rotor}}{I_{rotor}}\right) = N_{ef}^2 . Z_{rotor} \quad (3)$$

Onde Nef é a relação de espiras entre o rotor e estator real. Considerando o movimento relativo entre o rotor e estator surge uma reatância de dispersão do rotor. Para efeitos deste estudo é considerado apenas um circuito equivalente baseado nas grandezas do rotor equivalente no qual o modelo a ser analisado serve tanto para rotor bobinado quanto para rotor gaiola de esquilo.

Logo, é possível representar o rotor do motor pela fórmula demonstrada na equação 4.

$$Z_{2s} = \frac{E_{2s}}{I_{2s}} = R_2 + jsX_2 \quad (4)$$

Onde,

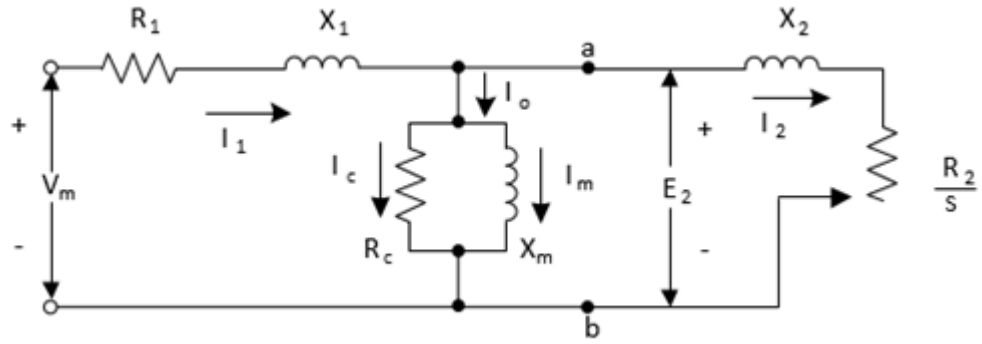
$R2$: Resistencia referida ao rotor;

$jsX2$: Reatância de dispersão referida ao rotor, na frequência de escorregamento.

A letra “s” representa o escorregamento do motor, ou seja, a diferença entre a frequência elétrica e a frequência mecânica da máquina. Então, ao se isolar Z2 para obter a expressão da impedância equivalente estacionária do rotor, obtém-se a equação 5.

$$Z_2 = \frac{E_{2s}}{I_{2s}} = \frac{R_2}{s} + jX_2 \quad (5)$$

Figura 2.6: Circuito Equivalente Monofásico de um MIT Polifásico



Fonte: Fitzgerald (2008).

A partir destas equações é representado um motor de indução trifásico, com isso podemos analisá-lo matematicamente de maneira objetiva e extrair as informações necessárias para modela-lo e com isso garantir o bom funcionamento do motor, dentro de parâmetros conhecidos.

2.1.2. Defeito e Falha

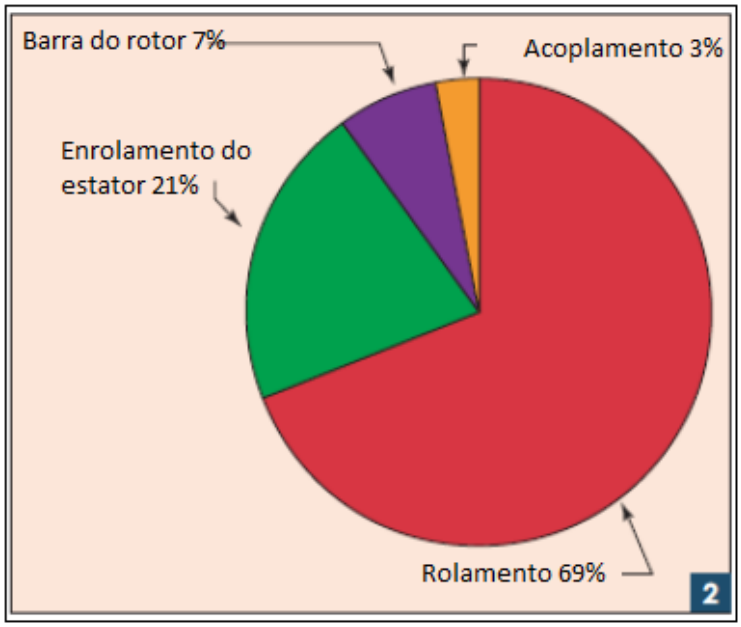
Um comum equívoco cometido é na distinção entre os conceitos de defeito e falha. Sobre isso, existem diversas literaturas a respeito, porém, de forma resumida, é possível claramente observar que defeito é qualquer desvio de uma característica de um item em relação aos seus requisitos, que podem ser uma especificação e pode afetar a capacidade de um item desempenhar uma função requerida. A falha é definida por um evento, caracterizado pelo término da capacidade de um item desempenhar a função requerida. Ela pode ser resultado de um defeito. Após a falha, um item entra em estado de pane, excluindo-se a incapacidade durante a manutenção preventiva, demais ações planejadas ou a falta de recursos externos, conforme a NBR 5462 (ASSOCIAÇÃO..., 1994).

2.1.3. Principais defeitos em MITs

A manutenção do bom funcionamento dos motores dentro dos mais diversos processos é fator fundamental e muito almejado, sejam em processos contínuos ou esporádicos, a disponibilidade dessas máquinas requer uma grande atenção e um correto plano para evitar falhas funcionais geradas por modos de falha e, por sua vez, que transtornos aconteçam. Existem três grupos de fatores principais que influenciam no funcionamento adequado dos motores e são focos dos mais diversos estudos: Os de aspecto elétrico, como a qualidade de energia; os de aspecto mecânico, como rolamentos e vibrações e também os de aspectos térmicos, como temperatura ambiente e de serviço.

É interessante mostrar a relevância dos defeitos que ocorrem nos componentes dos motores, mais especificamente os de indução trifásicos, foco desse trabalho. Em um estudo divulgado pela revista IEEE INDUSTRY APPLICATIONS MAGAZINE, de 2008, podem ser vistos dados estatísticos dos principais defeitos em motores nas indústrias petrolífera e química. Através de uma normalização de dados e exclusão de fatores desconhecidos externos foram verificados os defeitos nos componentes mais comuns como mostrado na Figura 2.7. (BONNET; YUNG; 2008).

Figura 2.7: Normalização de dados de modos de falha em motores de indução



Fonte: Adptado de Bonnet, Yung (2008).

Os defeitos nos enrolamentos do estator e seu respectivo isolamento representam a maioria dentro do aspecto elétrico. Alguns fatores podem contribuir para deterioração do isolamento, como envelhecimento térmico, surtos de tensão causados por raios, chaveamento ou pulsos periódicos, danos ao isolamento causados por ações mecânicas como vibrações ou forças cortantes e agentes externos como químicos ou hidrocarbonetos (ABREU e EMANUEL, 2002).

2.1.4. Qualidade de Energia Elétrica

Em um sistema elétrico, é imprescindível que o fornecimento da energia esteja dentro de parâmetros aceitáveis para sua utilização, desde níveis de tensão frequência, flutuação, continuidade, entre outros.

Desde 1978, no Brasil, já existem regulamentações que estabelecem limites para a tensão em regime permanente a serem respeitados pelas concessionárias de distribuição. Em 2008 foram introduzidos os Procedimentos de Distribuição de Energia Elétrica no Sistema Elétrico Nacional (PRODIST), que trouxeram muitas inovações quanto à qualidade do produto, a energia elétrica, estabelecendo valores de referência para fenômenos de curta duração (variações e flutuações de tensão e variações de frequência) e de regime permanente (harmônicos e desequilíbrio de tensão).

Os vários efeitos das variações e desequilíbrios de tensão, também conhecidas como VDI (*Voltage Distortion and Imbalance)* sobre motores de indução são estudados desde 1950. Elas implicam em perdas de potência adicionais, as quais aumentam a tensão média e causam aumento da temperatura do enrolamento do estator. Nessas situações, o torque eletromagnético desenvolvido é menor que o torque em condições ideais, tendo como consequência um maior tempo de partida e o processo de fadiga do motor é acelerado (ABREU; EMANUEL, 2002).

2.1.4.1. Desequilíbrio de Tensão

Em um sistema trifásico ideal, as tensões nas três fases possuem a mesma amplitude e estão defasadas 120 graus entre si. Assim, o desequilíbrio ou o desbalanceamento de tensão é uma condição na qual as fases apresentam entre si tensão com módulos diferentes, ou defasagem angular entre elas que não sejam os 120 graus ou, ainda, as duas condições simultaneamente.

O desequilíbrio de tensão provoca efeitos no sistema elétrico, tais como sobreaquecimentos e redução da vida útil em motores, transformadores e cabos, mau funcionamento e falhas dos dispositivos de proteção, entre outros (ANEEL, 2011).

O desequilíbrio de tensão é definido pela *National Electrical Manufacturers Association* (NEMA) (JOUANNE; BANERJEE, 2001) pela expressão da equação 6.

$$\%Desequilíbrio = \frac{Desvio\ Máximo\ da\ Média}{Média\ das\ três\ tensões\ fase{-}fase} . 100 \qquad (6)$$

Outro padrão para indicar o grau do fator de desequilíbrio é o europeu *Voltage Unbalance Factor* (VUF), que é a razão entre a tensão de sequência negativa $V2$ e a tensão de sequência positiva $V1$ (JOUANNE; BANERJEE, 2001). A regulamentação brasileira se baseia neste padrão e apresenta a mesma fórmula (equação 7) para o cálculo, chamando este de Fator de Desequilíbrio, com a notação FD% (ANEEL, 2011).

$$FD\% = \%\, VUF = \frac{V_2}{V_1} . 100 \qquad (7)$$

A regulamentação também possibilita a avaliação FD através das tensões de linha, usando a expressão mostrada na equação 8.

$$FD\% = 100. \sqrt{\frac{1-\sqrt{3-6\beta}}{1+\sqrt{3-6\beta}}} . 100 \qquad (8)$$

Possibilitando a partir da equação 8 a expressão mostrada na equação 9.

$$\beta = \frac{V_{ab}^4+V_{bc}^4+V_{ca}^4}{(V_{ab}^2+V_{bc}^2+V_{ca}^2)^2} \quad (9)$$

Em motores de indução, podem ser listados basicamente quatros tipos de perdas causadas pelas variações e desequilíbrios de tensão: Perdas no Ferro, Perdas no enrolamento do Estator e no Rotor Gaiola, Perdas de potência nas Barras do Rotor e, Perdas pulsantes e de Superfície (ABREU; EMANUEL, 2002).

No que se refere às perdas, tem-se:

a) Perdas no Ferro

As perdas no núcleo do estator ocorrem em função do pico no acoplamento indutivo. A presença de uma tensão fundamental de sequência negativa $V1-$ é que causa o desbalanceamento de tensão. A definição de desbalanceamento de tensão é dada resumidamente pela equação 10 (ABREU; EMANUEL, 2002).

$$u = \frac{V_1^-}{V_1^+} \approx \frac{V_1^-}{V_R} \quad (10)$$

Onde $V1+$ é a tensão fundamental de sequência positiva. Em situações práticas, onde $u<0{,}05$, o desequilíbrio de tensão tem um mínimo efeito para as perdas no núcleo do estator. O campo rotativo faz com que algumas regiões do núcleo do estator obtenham um ligeiro aumento na densidade de fluxo magnético, enquanto no restante do núcleo ocorre o contrário.

b) Perdas enrolamento do estator e no rotor gaiola de esquilo

Perdas adicionais de potência no enrolamento do estator são causadas por desequilíbrios de tensão e também por distorções harmônicas. Segundo o estudo de ABREU e EMANUEL (2002), considerando vários fatores, como corrente eficaz

causada pela tensão fundamental de sequência negativa, corrente harmônica, efeito periférico da corrente, reatâncias, entre outros.

c) Perdas de potência nas entre barras

Quando a tensão entre as barras adjacentes é grande o suficiente, correntes parasitas irão fluir através das laminas do rotor e pular para as barras adjacentes. Estas perdas adicionais pelas correntes entre barras são difíceis de serem previstas, são fortemente dependentes da resistência de contato entre as barras e as laminações, e pode assumir e ser distribuída entre os dentes do rotor.

d) Perdas na superfície e pulsantes

São perdas no ferro causadas pelas altas frequências da indução do entreferro, e estão localizadas na superfície e no interior dos dentes do rotor.

2.1.5. Problemas de vibração

Em se tratando de componentes mecânicos é visível, através da Figura 7, que a principal causa das falhas é o defeito em rolamentos, tornando pertinente um aprofundado estudo sobre os tipos de falhas causadas por eles.

Conforme Cardoso (1991), um rolamento é formado por quatro elementos, anel externo, anel interno, gaiola e corpos rolantes para permitir o movimento deve haver alguma folga entre essas partes e exatamente por haver estas folgas existe a vibração que possuem frequências características de cada componente que ao girar cria um movimento repetitivo com frequência determinada e influenciam diretamente na corrente do motor, o rolamento causa perdas por fricção entre suas partes móveis.

2.1.6. Problemas de Temperatura

O motor de indução é construído para trabalhar em diversos ambientes e regimes, isto é dado por suas características construtivas e seu grau de proteção. Para Fitzgerald (2006, p 630) a potência nominal especificada para componentes elétricos é limitada por características térmicas e mecânicas, sendo no primeiro caso a corrente máxima admissível limitada pela temperatura que o material isolante pode suportar sem sofrer danos ou comprometer o funcionamento da máquina.

Segundo a WEG (2014) a vida útil de um motor de indução depende quase exclusivamente da vida útil dos materiais responsáveis pela isolação de seus enrolamentos.

Como o limite de temperatura depende do tipo de material empregado, para fins de normalização, os materiais isolantes e os sistemas de isolamento são agrupados em CLASSES DE ISOLAMENTO, cada qual definida pelo respectivo limite de temperatura, ou seja, pela maior temperatura que o material ou o sistema de isolamento pode suportar continuamente sem que seja afetada sua vida útil. As classes de isolamento utilizadas em máquinas elétricas e os respectivos limites de temperatura conforme ABNT NBR 17094-1 e IEC 60034- 1 são mostrados na figura 2.8.

Figura 2.8: Faixas de temperatura de serviço

Classe (Class)	T (ºC)
A	105
E	120
B	130
F	155
H	180

Fonte: ABNT NBR 17094-1

Também a temperatura de serviço é um fator importante para determinar as perdas e está intimamente ligada à vida útil do motor. A temperatura de serviço

depende da temperatura ambiente em que a máquina está instalada e da capacidade de seu sistema de arrefecimento.

A determinação com precisão da temperatura de serviço do motor é difícil, se realizada com sensores ou termômetros, pois cada ponto do motor possui uma temperatura e é difícil saber qual o ponto exato que esta temperatura é máxima, sendo mais indicado o cálculo de variação do valor da resistência ôhmica do estator, que determina indiretamente o valor da temperatura de serviço aplicando a fórmula da equação 11.

$$T_S = \frac{R_S}{R_A}\,(M_C + T_A) - M_C \qquad (11)$$

Onde:

T_S: Temperatura de serviço calculada no enrolamento do estator;

T_A: Temperatura ambiente medida;

R_A: Resistência medida na temperatura ambiente;

R_S: Resistência medida na temperatura de serviço;

M_C: Constante de proporcionalidade adimensional, referente ao tipo de condutor (para o cobre M = 234,5).

Para um MIT, a temperatura de serviço não pode ultrapassar o valor especificado pelo fabricante, uma vez que a partir da temperatura especificada o motor pode apresentar defeitos que podem vir a gerar falhas. É importante ressaltar que para haver falha por temperatura basta que um ponto interno do motor ultrapasse o valor de referência. A temperatura ambiente influencia na determinação da temperatura máxima, devendo ser somada.

2.1.6.1. Efeitos da temperatura no escorregamento

Além de a temperatura influenciar na resistência ôhmica dos enrolamentos do motor, existe também a influencia da temperatura no escorregamento, pois o conjugado eletromagnético Cm depende da resistência rotórica, conforme Fitzgerald (2006, p. 307) a partir da fórmula mostrada na equação 12.

$$C_m = \frac{3.I_2'^2.\frac{R'_2}{s}}{w_s} \quad (12)$$

Onde:

$I2'$: Corrente do rotor referenciada ao estator;

$R'2$: Resistência do rotor referenciada ao estator;

ws: Frequência do campo magnético do estator.

Pelas equações 13 e 14, tem-se que:

$$I_2'^2 = \frac{V_1}{Z} \quad (13)$$

$$Z = \sqrt{(\frac{s.r_1+r_2}{s})^2 + (x_1 + x'_2)^2} \quad (14)$$

Onde:

$r1$: Valor da resistência enrolamento do estator.

Fazendo-se as substituições necessárias, obtém-se a equação 15.

$$C_m = \frac{3.V_1^2.R'_2}{w_s.[(s.r_1+r'_2)^2+s^2(x_{1+}x'_2)^2]} \quad (15)$$

Para encontrar o valor do conjugado máximo, é preciso derivar a expressão anterior e igualar seu resultado à zero, para então se obter seu ponto máximo (equação 16).

$$\frac{dC_m}{ds} = \frac{\frac{3.V_1^2}{ws}[R'_2((s.r_1+r'_2)^2+s^2(x_{1+}x'_2)^2)]-[2(s.r_1+r'_2) \quad .r1+2s \quad (x_{1+}x'_2)^2].s.r'_2}{w_s.[(s.r_1+r'_2)^2+s^2(x_{1+}x'_2)^2]} \quad (16)$$

Resolvendo a equação, a fim de obter o valor do escorregamento máximo $smax$, obtém-se a relação mostrada na equação 17.

$$s_{max} = \pm \frac{3.V_1^2}{ws} . \frac{1}{2(\sqrt{r_1^2+(x_1+x'_2)^2}-r_1} \quad (17)$$

Com isso, é possível verificar que o escorregamento está ligado indiretamente à variação da temperatura dos enrolamentos, uma vez que a resistência elétrica varia em função da temperatura.

2.2. TÉCNICAS DE MANUTENÇÃO PREDITIVA

A Manutenção Preditiva, Também definida como Manutenção Controlada, é uma evolução da manutenção preventiva e foi desenvolvida com base no monitoramento. É baseada na aplicação sistemática de técnicas de análise através de meios de supervisão ou amostragem, intervém o mínimo possível na planta, buscando reduzir a manutenção preventiva e também a corretiva, interfere pouco na planta e trás resultados positivos, apesar de ainda não ser o tipo de manutenção preferido pelas empresas brasileiras, pois se avaliado e gerenciado de forma incorreta, pode gerar custos elevados. Apesar disso, foram desenvolvidas várias técnicas de monitoramento para se detectar possíveis defeitos e antecipar as ações de manutenção.

2.2.1. ESA (*Electrical Signature Analysis*)

Segundo Intech (2012), a Análise da Assinatura Elétrica (ESA) é um termo genérico para as técnicas de monitoramento através da análise de sinais elétricos como tensão e corrente. A aplicação destas técnicas nas indústrias visa aumentar a confiabilidade dos equipamentos, reduzindo tempos de pane e por sua vez aumentando a disponibilidade.

Dentro desse grupo podem ser citadas, segundo Maciel (2013), as principais técnicas de manutenção preditiva para detecção de defeitos em motores de indução são a Análise de Assinatura de Corrente do Motor (MCSA, do inglês *Motor Current Signature Analysis*), a Abordagem Estendida do Vetor de Park (EPVA, *Extended park Vector Approach*) e a Análise Espectral da Potência Instantânea (IPSA, do inglês *Instantaneous Power Signature AnalysisI*).

2.2.2. Análise de assinatura de corrente do motor (MCSA)

Segundo Bonaldi (2005), o MCSA é uma técnica não invasiva para diagnóstico de problemas em motores de indução. Ela utiliza a amostragem de corrente estatórica de apenas uma das três fases do motor e o seu sinal de corrente é analisado gerando um espectro de frequência, conhecido como assinatura de corrente do motor. O padrão da assinatura é observado na situação de normalidade do funcionamento do motor e para que se possa comparar com possíveis padrões de defeitos.

Segundo Bonaldi (2005), é importante conhecer-se um histórico comportamental do conjunto formado pelo motor, o sistema de transmissão e a carga, para que, aliado ao conhecimento do especialista (ou de um sistema especialista), ajude nas decisões para resolver os problemas. Através do MCSA é possível detectar defeitos principalmente de aspecto mecânico, como excentricidade no rotor, defeitos em rolamentos e no acoplamento e na carga acoplada. Apesar disso, existem estudos relacionados aos problemas de aspecto elétrico, como enrolamentos do estator.

2.3. SÍNTESE DO CAPÍTULO

Neste capítulo foram elencados os principais modos de falha que ocorrem em um MIT e como a manutenção preditiva atua sobre estes para fazer a predição a fim de evitar falhas funcionais.

Como o trabalho tem como foco a aplicação uma técnica de análise de caráter preditivo, é importante ter como base a origem destas técnicas e porque elas devem ser aplicadas na indústria.

No próximo capítulo, será apresentada a lógica *Fuzzy*, que tem o objetivo de traduzir o pensamento humano para linguagem de máquina e após o processamento é feita a retradução, de forma a apresentar um resultado que seja compreensível pelo usuário.

3. LÓGICA FUZZY

3.1. INTRODUÇÃO

Em 1965, o engenheiro eletrônico Lofti Asker Zadeh, professor de teoria dos sistemas da universidade de Berkeley, mencionou pela primeira vez à comunidade científica a expressão Lógica *Fuzzy*. Na década de 70, ampliando seus conceitos desenvolvidos na década anterior, propôs sua extensão com o conceito de variável linguística.

A lógica *Fuzzy* trabalha com limites incertos e imprecisos. A comunicação humana contém diversas incertezas na forma de expressões verbais que são vagas e imprecisas, utilizando muitas vezes, palavras iguais que no contexto tem significados diferentes. Para os seres humanos, as palavras não representam uma ideia única, mas representam um conjunto de ideias.

Por outro lado, os computadores não conseguem entender os termos *Fuzzy* da comunicação humana, podendo raciocinar apenas de forma binária. A Lógica *Fuzzy* pode preencher este vazio e traduzir os graus de verdade das afirmações de uma maneira que os computadores possam processar tal informação, fazendo com que estes “raciocinem” conforme humanos.

3.2. DEFINIÇÕES DOS CONJUNTOS *FUZZY*

Na teoria clássica dos conjuntos, se um elemento “X” do universo de discurso “U”, pertence a um dado conjunto “A”, então este elemento satisfaz um predicado associado a este conjunto. Pode-se então definir este conjunto por meio de uma função, chamada de função característica mapeada, porque associa cada elemento do universo de discurso U um binário (equações 18 e 19).

$$\gamma_A(x) = 1 \; se \; x \in A \tag{18}$$

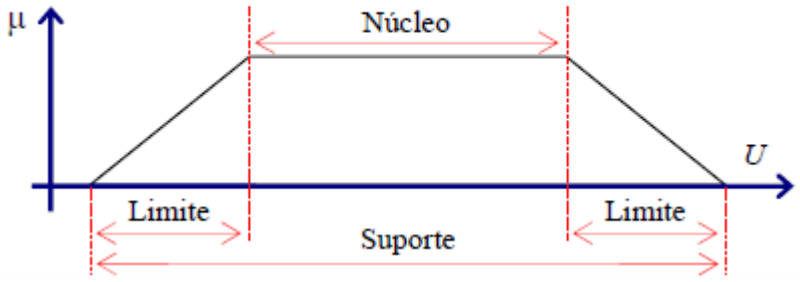
$$\gamma_A(x) = 0 \; se \; x \notin A \tag{19}$$

Um conjunto *Fuzzy* caracteriza-se através do seu vetor de pertinência, com graus de pertinência individuais dentro do intervalo numérico {0,1}. Esses graus de pertinência podem ser considerados como medidas que avaliam se um elemento é ou não um membro de um dado conjunto *Fuzzy*.

As terminologias que descrevem e definem um conjunto *Fuzzy* são citadas abaixo e mostradas graficamente na Figura 3.1.

- Núcleo: região do universo de discurso caracterizada por ter uma pertinência total ao conjunto *Fuzzy*, $\mu(x) = 1 \; \forall \; x \in$ núcleo.
- Suporte: região do universo de discurso caracterizada por ter uma pertinência total ao conjunto *Fuzzy* diferente de zero, $\mu(x) \neq 0, \; \forall \; x \in$ suporte.
- Limite: região do universo de discurso caracterizada por ter uma pertinência total ao conjunto *Fuzzy* entre 0 e 1, $0 < \mu(x) < 1, \; \forall \; x \in$ limite.

Figura 3.1: Núcleo, suporte e limite de um conjunto Fuzzy



Fonte: RIGONI, 2009.

3.3. PROPRIEDADES DOS CONJUNTOS *FUZZY*

Considerando A, B e C conjuntos *Fuzzy* de um dado universo de discurso U, as propriedades destes conjuntos são mostradas na Tabela 3.1.

Tabela 3.1: Propriedades dos conjuntos *Fuzzy*

Propriedade Comutativa	$A \cup B = B \cup A$ e $A \cap B = B \cap A$
Propriedade Associativa	$(A \cup B) \cup C = A \cup (B \cup C)$ e $(A \cap B) \cap C = A \cap (B \cap C)$
Idempotência	$A \cup A = A$ e $A \cap A = A$
Distributividade em relação à União	$A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$
Distributividade em relação à Interseção	$A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$
Conjunto *Fuzzy* e seu complemento	$A \cup \bar{A} \neq U$ e $A \cap \bar{A} \neq \varphi$
Conjunto *Fuzzy* e seu conjunto nulo	$A \cup \varphi = A$ e $A \cap \varphi = \varphi$
Conjunto *Fuzzy* e seu universo	$A \cup U = U$ e $A \cap U = A$
Involução	$\neg\bar{A} = A$
Teorema de Morgan	$\neg(A \cup B) = \neg A \cup \neg B$ e $\neg(A \cap B) = \neg A \cap \neg B$

Fonte: REZENDE 2003

3.4. OPERAÇÕES *FUZZY*

As operações entre conjuntos *Fuzzy* que serão relevantes para este trabalho são: complemento, inserção e união. Estas operações entre conjuntos, pertencentes a universos de discursos diferentes, possibilitam a construção da base de conhecimento de um sistema (SHAW e SIMÕES, 2002).

- Complemento

O complemento de um conjunto *Fuzzy* A normatizado, corresponde ao conectivo “não”, Normalmente denotado por $\bar{A}$. A função de pertinência deste conjunto em um universo U é dada pela equação 20.

$$\mu\bar{A}(x) = 1 - \mu A(x), \forall x \in U \qquad (20)$$

- Interseção:

A interseção entre dois conjuntos *Fuzzy* A e B de um universo de discurso U corresponde ao conectivo “e”, e pode ser representada por C = A∩B. Neste caso a função de pertinência proposta por Zadeh, é definida pela equação 21.

$$\mu_c(x) = \mu_{A\cap B} = \min\{\mu_A(x)\}, \forall\, x \in U \qquad (21)$$

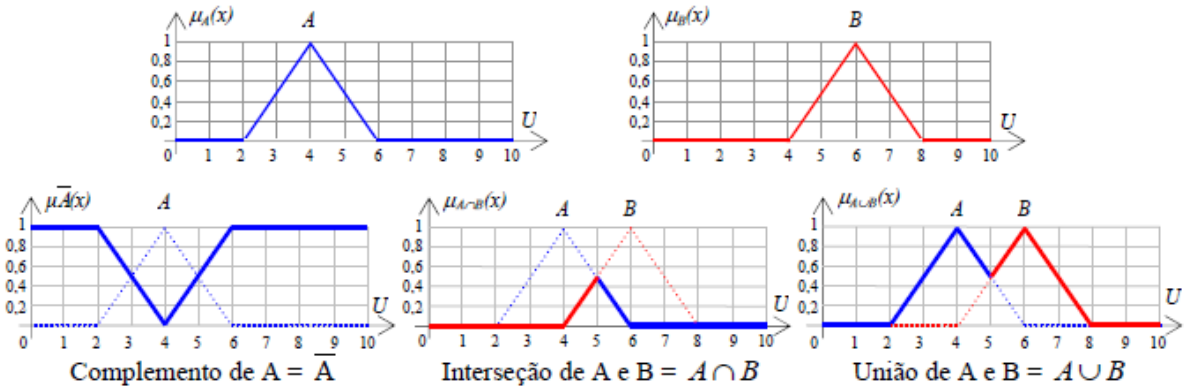
- União

A união de dois conjuntos *Fuzzy* A e B do universo de discurso U, corresponde ao conectivo “ou”, e pode representada por C = A∪B, com C pertencente ao mesmo universo de discurso U. A função de pertinência da operação de união, proposta por Zadeh, é definida pela equação 22.

$$\mu_c(x) = \mu_{A\cup B} = \max\{\mu_A(x), \mu_B(x)\}, \forall\, x \in U \qquad (22)$$

Considerando um universo de discurso U = [0,10] têm-se na figura 3.2 as operações complemento, interseção e união dos conjuntos *Fuzzy*.

Figura 3.2: Complemento, interseção e união de conjuntos Fuzzy



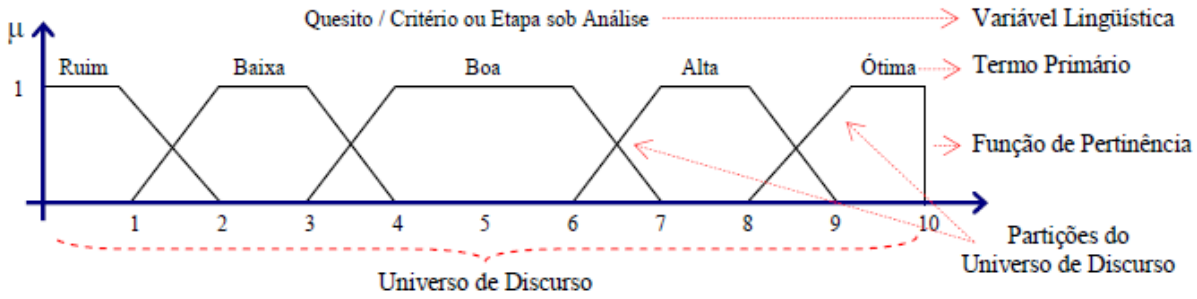
Fonte: RIGONI, 2009.

3.5. VARIÁVEIS E MODIFICADORES LINGUÍSTICOS

Segundo Cox (1994), uma variável linguística pode ser considerada como sendo o nome dado a um conjunto *Fuzzy*, sendo esta a essência da técnica de

modelagem *Fuzzy*. As variáveis linguísticas representam de um modo impreciso, conceitos de variáveis de um dado problema, admitindo como valores somente expressões linguísticas que são chamadas de termos primários. Uma variável linguística pode ter seu termo primário representado por um conjunto *Fuzzy* existente no universo discurso, é associado um conceito linguístico que classifica ou define um valor linguistico para a variável linguística *Fuzzy* em questão. A variável linguística pode ser entendida como o quanto de um dado elemento “x” do universo de discurso “U”, satisfaz o conceito representado por um conjunto *Fuzzy* “A”. Na Figura 2.3 tem-se uma partição de um conjunto *Fuzzy* e seus principais elementos.

Figura 3.3: Partição de conjuntos Fuzzy

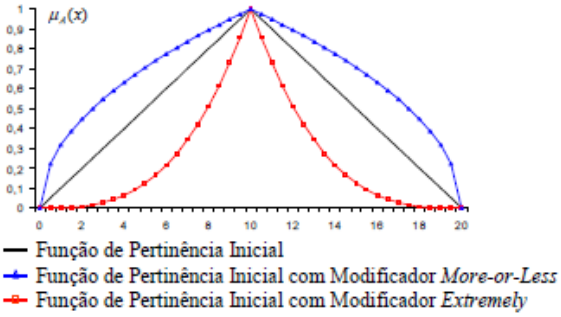


Fonte: RIGONI, 2009

Os modificadores linguísticos são operações, que associados aos termos primários das variáveis linguísticas, alteram a forma das funções de pertinência, introduzindo um novo significado ao conjunto original, criando um conjunto *Fuzzy*, composto. Na Figura 3.4 mostra-se exemplos de modificadores e sua ação na função de pertinência de um conjunto *Fuzzy*.

Figura 3.4: Modificadores linguísticos Fuzzy

Nome do Modificador		Ação
Not	Não	$1-\mu_A(x_i), \forall x_i \in U$
Somewhat	Um Pouco	$\mu_A^{1/3}(x_i), \forall x_i \in U$
More-or-Less	Mais ou Menos	$\mu_A^{0,5}(x_i), \forall x_i \in U$
Very	Muito	$\mu_A^{2}(x_i), \forall x_i \in U$
Extremely	Extremamente	$\mu_A^{3}(x_i), \forall x_i \in U$



Fonte: RIGONI, 2009.

3.6. REGRAS DE PRODUÇÃO *FUZZY*

Segundo Rezende (2003), as regras de produção *Fuzzy* são o modo mais comum de armazenar informações em uma base de conhecimento *Fuzzy.* Essas regras de produção são mostradas abaixo em formas equivalentes:

SE{situação} ENTÃO {ação} ←→ SE {x é A} ENTÃO {Y é B}

SE {“Variável linguística de entrada” é “Termo primário de entrada”}

ENTÃO{“Variável linguística de saída” é “Termo primário de saída”}

De acordo com Bárdossy e Duckstein (1995), é necessário definir regras para se construir uma modelagem baseada em regras difusas. A definição das regras é o procedimento em que o conhecimento e os dados disponíveis são transcritos em regras. Durante esse processo, deve-se observar as quatro situações abaixo:

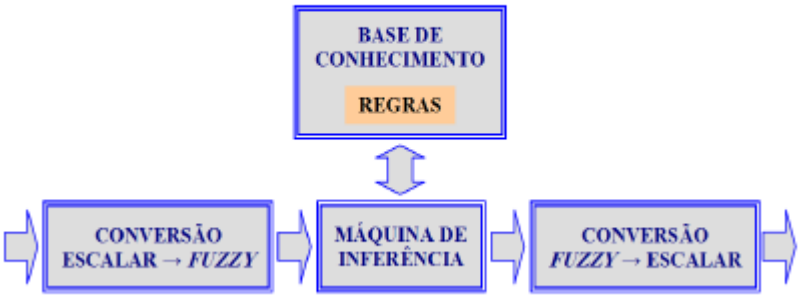
1. As Regras são bem conhecidas pelos especialistas e podem ser descritas diretamente;

2. As regras podem ser definidas por especialistas, mas os dados disponíveis devem ser utilizados para atualiza-las;
3. As regras não são bem conhecidas explicitamente, mas as variáveis requeridas para a descrição do sistema podem ser especificadas por especialistas;
4. Somente um conjunto de observações está disponível, e um sistema de regras tem de ser definido de forma a descrever as interconexões entre os elementos desse conjunto de dados.

3.7. PROCESSO DE INFERÊNCIA *FUZZY*

Em 1975, Mamdani propôs um método de inferência que foi por muitos anos um padrão para a utilização dos conceitos da lógica *Fuzzy* em processamento de conhecimento. As regras de produção em um modelo de Mamdani possuem relações *Fuzzy* tanto em seus antecedentes como em seus consequentes. O modelo Mamdani possui módulos de interface que transformam as variáveis de entrada baseadas em grandezas numéricas (*crisp*), em conjuntos *Fuzzy* equivalentes e, posteriormente, as variáveis *Fuzzy* geradas em variáveis numéricas (*crisp*) proporcionais. Os dados provenientes da interface com o usuário são *fuzzyficados* no módulo de conversão Escalar → *Fuzzy*, a máquina de inferência recebe estes dados e processa as regras existentes na base de conhecimento gerando, a partir da composição de todas as regras disparadas, um conjunto *Fuzzy* de saída para o módulo de conversão *Fuzzy* → Escalar que *desfuzzyfica* os resultados do processo de inferência, para posterior apresentação ao usuário (RIGONI, 2009). Na Figura 3.5 é detalhado este processo.

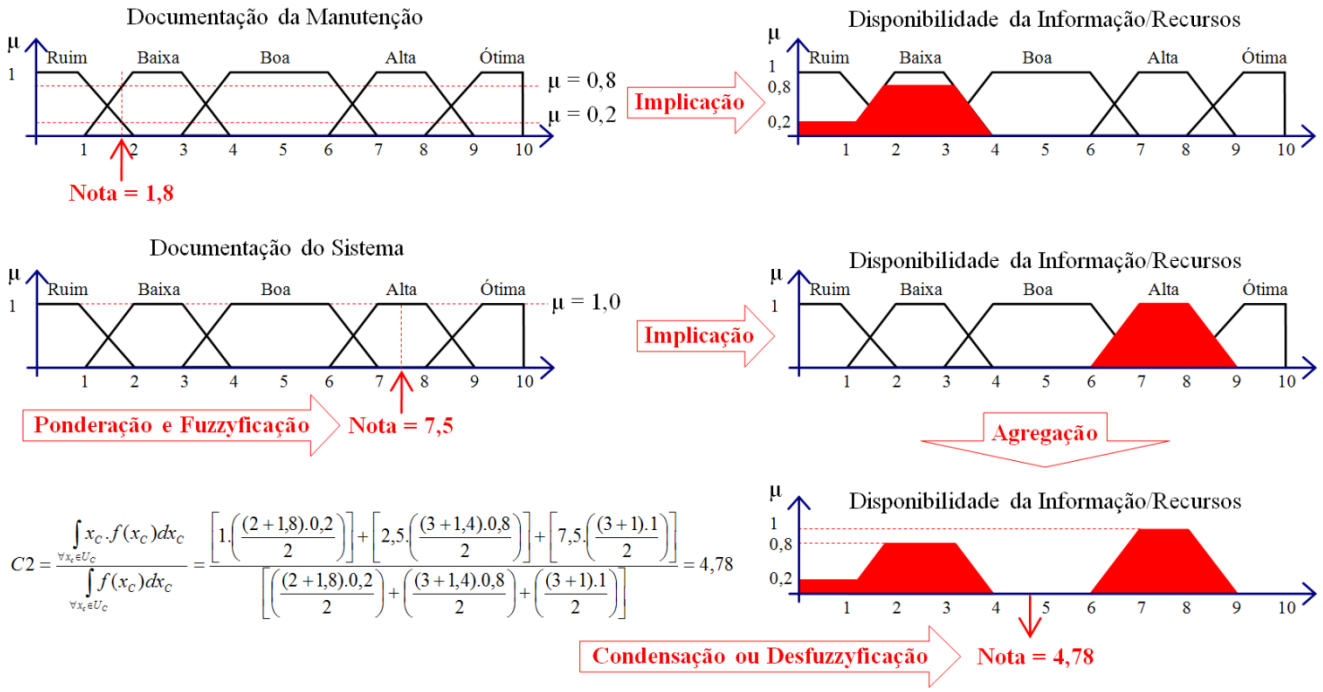
Figura 3.5: Diagrama típico de um modelo de inferência de Mamdani



Fonte: Adaptado de REZENDE, 2003.

O processo de definição e implementação das regras genéricas para uma modelagem baseada em regras difusas é aplicado seguindo as seguintes etapas: avaliação dos antecedentes, implicação, agregação dos consequentes a condensação dos consequentes. Na figura 3.6 explica-se esse processo.

Figura 3.6: Exemplo de um processo de inferência Fuzzy



Fonte: RIGONI, 2009.

O processo inicia-se com a *fuzzyficação* de um quesito referente a uma etapa do MCC. Para este quesito é associado uma nota que está dentro dos limites impostos pelos subconjuntos *Fuzzy*. Logo após é feita a implicação que determinará os subconjuntos *Fuzzy* que serão afetados. Feito isso para outro quesito, os dois itens analisados são agregados em um conjunto, o qual é *desfuzzyficado*. O resultado é o baricentro da agregação dos conjuntos.

3.7.1. Avaliação dos Antecedentes

A avaliação dos antecedentes (premissas ou situações) de uma regra se compõe, em geral, de duas etapas. Primeiramente, os valores numéricos dados para cada variável de entrada são avaliados de acordo com as funções de pertinência associadas à variável correspondente, resultando o grau de pertinência de cada valor nos termos linguísticos correspondentes. Entende-se esse processo, também, como a transformação dos valores numéricos das variáveis de entrada em números *Fuzzy* (*fuzzyficação*). Na segunda etapa, uma função é aplicada aos graus de pertinência obtidos para cada proposição antecedente, produzindo um valor numérico, entre 0 e 1, que representa o grau com que a expressão condicional da regra é satisfeita (grau de aplicabilidade da regra). As funções utilizadas nesse processo dependem do operador lógico usado na combinação das proposições, sendo as mais comumente adotadas as funções de mínimo para o operador “E” e máximo para o operador “OU”, tal que as operações de interseção e união de conjuntos *Fuzzy,* respectivamente (RIGONI, 2009).

3.7.2. Implicação

É o processo em que os consequentes das regras, referidas como regras aplicáveis, são calculadas com base nos respectivos graus de aplicabilidade. Este processo encerra a ideia de que: o antecedente da regra é verdadeiro com algum grau, então o consequente é, também, verdadeiro, com o mesmo grau. Nos casos

em que as regras possuem mais de um consequente, todos os consequentes são igualmente afetados pelo grau de aplicabilidade (RIGONI, 2009).

O processo de implicação consiste basicamente na modificação dos conjuntos *Fuzzy* associados com os consequentes da regra. No modelo de inferência *Fuzzy* de Mamdani, o conjunto é truncado em nível correspondente ao grau de aplicabilidade da regra (RIGONI, 2009).

3.7.3. Agregação dos consequentes

Quando um sistema de regras é avaliado para um conjunto de valores dados para as variáveis de entrada, encontram-se, em geral, mais uma regra aplicável. Neste caso, os consequentes obtidos pela inferência destas regras devem ser combinados ou agregados para produzir uma resposta única do sistema para cada variável de saída. No modelo de inferência *Fuzzy* de Mamdani, o método de agregação dos consequentes é a união dos conjuntos difusos (RIGONI, 2009).

3.7.4. Condensação dos consequentes

O conjunto *Fuzzy* gerado ao final do processo de agregação pode então ser utilizado diretamente em um diagnóstico quantitativo de tomada de decisão. Entretanto, em alguns casos, os conjuntos difusos obtidos pela agregação dos consequentes não são suficientes como respostas do sistema, sendo necessária a escolha de valores numéricos (*crisp*) representativos das respostas difusas. No modelo de inferência *Fuzzy* de Mamdani, este valor correspondente ao baricentro geométrico ou centroide da área definida pelo conjunto *Fuzzy* resultante da agregação dos consequentes. Este processo é comumente de *desfuzzyficação*. No método do baricentro geométrico, para um dado conjunto *Fuzzy* de saída, proveniente de uma base de conhecimento é utilizada como valor escalar de saída (RIGONI, 2009).

3.8. SÍNTESE DO CAPÍTULO

Neste capítulo foi abordada a lógica *Fuzzy* e seu funcionamento, esta lógica é útil para traduzir o modo do pensamento humano, pois não lida com absolutos, como a linguagem binária, lida com termos nebulosos e que são alterados de maneira gradual.

Está lógica será muito útil para este trabalho, pois como o sistema emulará um especialista, este não delineia seu pensamento como uma máquina, mas com linguagem nebulosa, que se altera com os diferentes valores de corrente e temperatura de serviço, levando em consideração os dois fatores para obter uma conclusão sobre a severidade da sobrecarga.

O ajuste dos valores do grau de severidade da sobrecarga será dado a partir de valores gerados empiricamente e também levando em consideração a experiência de vida do especialista.

No próximo capítulo será abordado especificamente como estes valores serão obtidos empiricamente e como estes serão incluídos em uma lógica nebulosa, para que quando o sistema receba os mesmos valores em sua entrada, a sua saída seja a mesma do que a conclusão de um especialista na área, confirmando assim a conformidade do sistema com a realidade.

4. MATERIAIS E MÉTODOS

4.1. OBJETO DE ESTUDO

O objeto de estudo se resume a um conjunto de dados proveniente das correntes de linha das bobinas e da temperatura de cabeça de bobina do MIT, as quais através da modelagem *Fuzzy* irão identificar as condições de funcionamento de um motor de indução trifásico (MIT). Estes dados de corrente e temperatura serão coletados via sensores provenientes do processo experimental com variados tipos de ensaios.

4.2. DELINEAMENTO

Como a lógica *Fuzzy* tenta se aproximar das decisões tomadas por um pensamento humano através de variáveis linguísticas, ou seja, um sistema nebuloso ou de possibilidades que vão além do sistema binário, os graus de severidade de defeitos dos MIT's destas variáveis serão classificados como "Baixo", "Médio" e "Alto", com base no artigo "*A Simple Fuzzy Logic Approach for Induction Motors Stator Condition Monitoring*" (Zeraoulia,*et al*, 2005).

As variáveis linguísticas e condições de defeito do MIT serão detectadas apenas pelos dados provenientes das correntes do estator, sendo estes coletados pela montagem experimental do conjunto motor, hardwares e softwares. Com relação à quantidade de regras da lógica fuzzy, estas serão determinadas conforme o artigo citado, mas também com o acréscimo de regras inerentes à variável "Temperatura do motor", visto que existe uma relação direta entre esta e o efeito joule causado pelas correntes.

A importância da aplicação da lógica *fuzzy* no trabalho se dá justamente pelo fato de esta trabalhar com informações vagas ou imprecisas, ou seja, esta consegue representar informação numérica em informação linguística, que é justamente a intenção do trabalho, transformar dados em avisos ou alertas visuais ao operador

sem que haja necessidade da parada do equipamento para verificação da possível causa de defeito.

4.3. DETALHAMENTO DE PROCEDIMENTOS E MATERIAIS

Para realização da análise de dados, faz-se necessário a descrição do processo experimental, procedimentos e demais detalhes relacionados aos materiais utilizados no trabalho, como hardwares, softwares e equipamentos auxiliares.

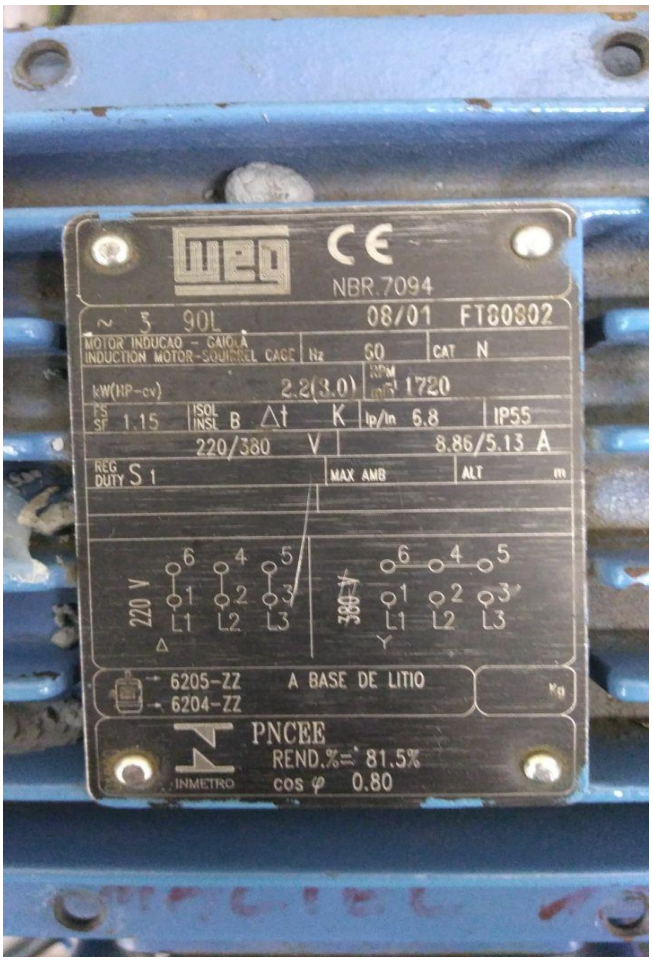
4.3.1. Procedimento Básico

O experimento será desenvolvido nos seguintes moldes e ensaios:

- Entrevista com especialista para a determinação da base do conhecimento;
- Aquisição dos dados de um motor em bom estado conectado em delta, para determinação dos valores de corrente do estator e da temperatura de operação, com plena carga, seguindo o que consta na NBR 17094-1
- Aquisição dos valores de corrente e temperatura do mesmo motor operando a vazio e operando sem uma das fases a vazio, simulando uma falha na alimentação ou um estator rompido;
- Aquisição dos valores de corrente e temperatura do mesmo motor com um desequilíbrio de tensão à plena carga.
- Aquisição dos valores de corrente e temperatura do mesmo motor operando em fator de serviço a 115%.
- Aquisição dos valores de corrente e temperatura do mesmo motor com sua ventilação bloqueada em fator de serviço a 115%.

A seguir, na figura 4.0, apresenta-se os dados de placa do MIT em questão:

Figura 4.1: Dados de placa do MIT de estudo



Fonte: Autoria própria.

4.3.2. Hardwares

A parte de hardwares necessários para execução do processo experimental se dá pelos seguintes itens:

- Motor de Indução trifásico (MIT);
- Sensores de corrente;
- Sensor de temperatura (módulo leitor de temperatura);
- Placa Microcontroladora Arduino;
- Computador.

Nos subtópicos a seguir são detalhados todos estes componentes, exceto o MIT (que já foi detalhado no referencial teórico) e computador. Aqui é válido ressaltar que, apesar de não ser o foco ressaltar a marca, modelo ou fabricante dos materiais utilizados, alguns componentes a serem descritos estão tão inseridos no mercado com sua marca (em questões de funcionamento e praticidade) que não há sentido não mencioná-las. Um exemplo disso é o próprio Arduino.

4.3.2.1. Sensor de corrente

Para a medição de corrente, irá se utilizar o sensor SCT-013-020 (20A), como mostrado na Figura 4.1 por ser um sensor não invasivo e apresentar saída linear em tensão que pode se ser analisado com o Arduino.

Figura 4.1: Sensor não invasivo de corrente

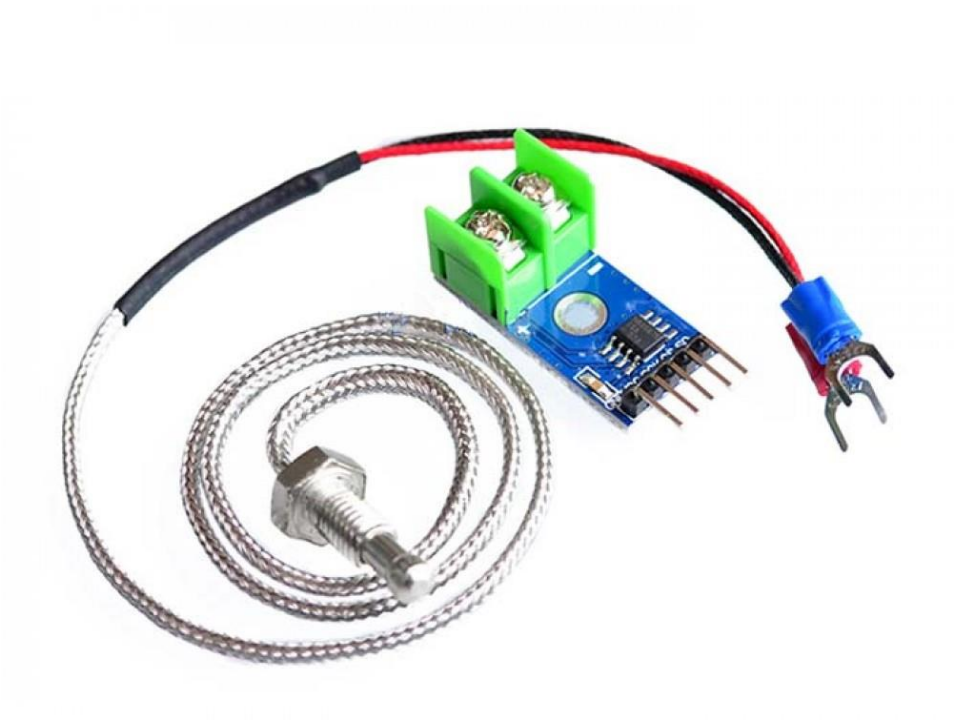
Fonte: Disponível em < https://www.filipeflop.com/produto/sensor-de-corrente-nao-invasivo-20a-sct-013/ > Acesso em: 24/11/2017 16:25

Este sensor funciona como um transformador de corrente, mas como existe um resistor de carga interno, este entrega uma resposta em tensão.

4.3.2.2. Sensor de temperatura

Para medirmos a temperatura diretamente na bobina do motor e identificar algum defeito térmico usou-se o sensor módulo leitor de temperatura MAX6675, representado pela figura 4.2.

Figura 4.2: Módulo leitor de temperatura MAX6675



Fonte: Disponível em <https://www.makerfabs.com/image/cache/makerfabs/K-Type%20Thermocouple%20with%20MAX6675%20AD%20Module/K-Type%20Thermocouple%20with%20MAX6675%20AD%20Module-1000x750.jpg> Acesso em: 05 maio 2018 11:25

Este é constituído por um termopar do tipo K aliado a um circuito interno conversor A/D (analógico/digital) e também amplificador de sinal. Como informações mais importantes de *Datasheet*, sabe-se que o range de leitura do termopar é de 0 a 800 ºC com resolução de 0,25 V/ºC. A sonda do termopar é revestida por uma blindagem que permite o grande alcance de temperatura e também evita interferência de ruídos externos.

De forma prática, a medição de temperatura por termopares se dá pela junção de dois condutores metálicos os quais ao se unirem nas pontas formam um circuito.

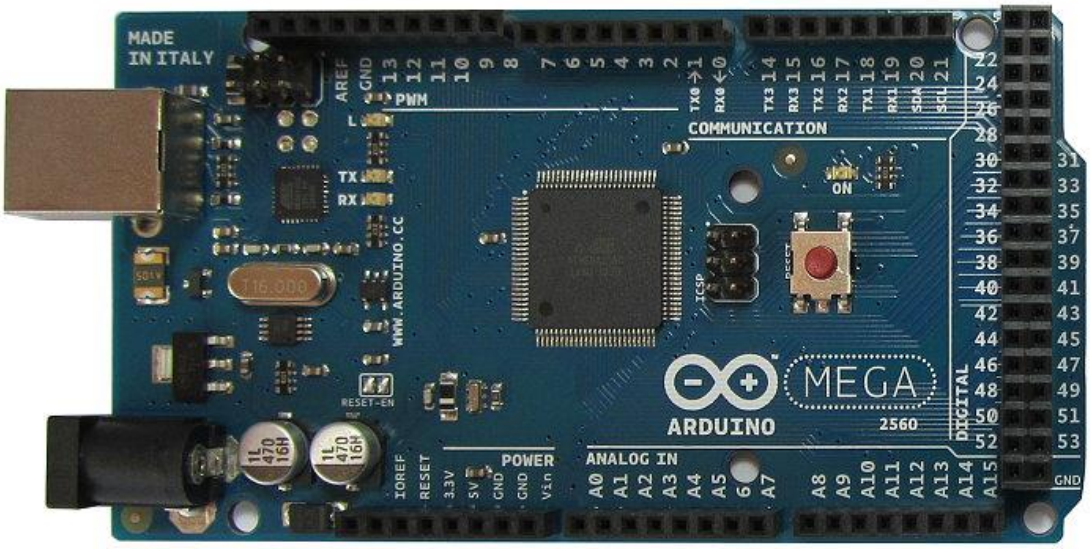
Estas pontas, quando submetidas a temperaturas geram uma diferença de potencial, sendo este o efeito *Seebeck*. Devido a isto, os elétrons tem sua energia cinética variando conforme o gradiente de temperatura das pontas do termopar. Quanto maior esta energia cinética, mais os elétrons se acumulam na ponta mais fria do termopar, o que acaba gerando uma diferença de potencial elétrico entre as extremidades que podem ser lidas em forma de miliVolts (mV).

4.3.2.3. Placa microcontroladora Arduino

O Arduino é uma plataforma de prototipagem de código aberto baseada em hardware e software fáceis de usar. As placas Arduino são capazes de ler entradas - luz, proximidade ou qualidade do ar em um sensor e transformá-lo em uma saída, como por exemplo, ativar um motor, ligar uma luz ou desencadear eventos externos.

Na Figura 4.3 a seguir é demonstrada a placa Arduino MEGA, que será utilizada devido à sua maior capacidade de processamento que o modelo básico UNO:

Figura 4.3: Placa microcontroladora Arduino MEGA



Fonte: https://startingelectronics.org/articles/arduino/which-arduino-for-beginners/arduino-mega-2560.jpg (Acesso em 05 mai 2018, 11:55).

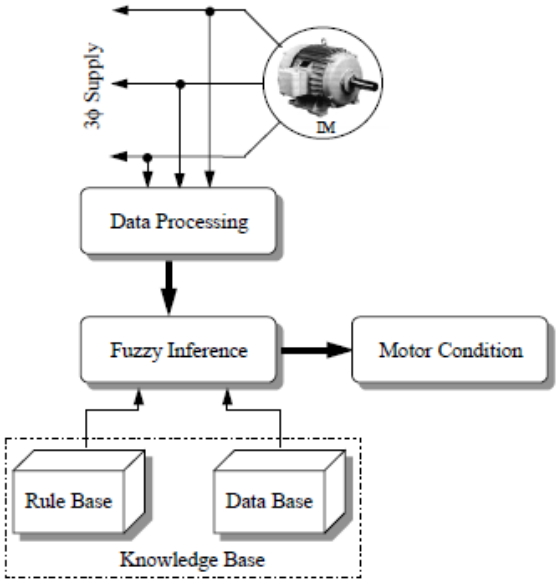
Esta placa microcontroladora possui pinos de entradas analógicas, pinos de entrada e saída digitais (dos quais 6 podem ser usados como PWM), um ressonador cerâmico (cristal oscilador de Clock) de 16 MHz, conexão USB, uma tomada de força e botão Reset, além de demais periféricos. Por fim, é comandada pelo microcontrolador ATmega328,

Devido à sua simplicidade de entendimento, manuseio e fácil comunicação com o ambiente de programação do computador via USB, o Arduino será usado como desenvolvedor de algoritmo para as inferências *Fuzzy*, principalmente pela utilização de bibliotecas prontas para estas aplicações. Os valores RMS das correntes e valores de temperatura em graus Celsius (ºC) também serão obtidos por um algoritmo realizado na interface de programação do Arduino.

4.3.3. Diagrama de Ligação dos hardwares

Na Figura 4.4 mostra-se o diagrama do processo de aquisição dos hardwares para aplicação do algoritmo *Fuzzy*.

Figura 4.4: Diagrama do processo de aquisição dos hardwares

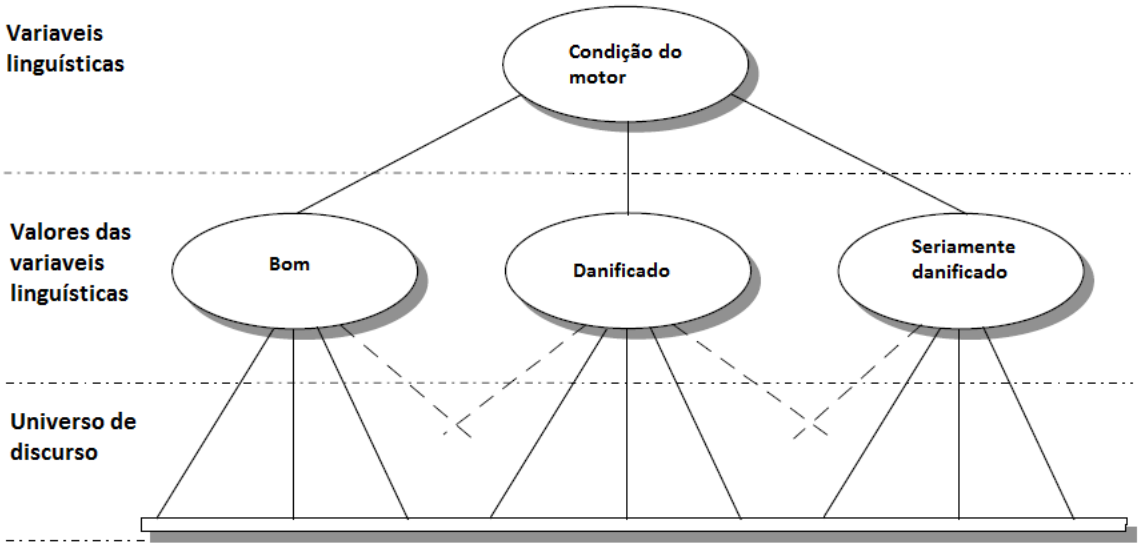


Fonte: Zeraoulia, *et all* ,2005

Com os dados provenientes deste esquema de ligação pode-se montar a estrutura e as regras para a lógica *Fuzzy,* detalhando em linguagem de máquina quais são os valores (zero, baixo, médio e alto) para as correntes e para a temperatura do estator.

Na figura 4.5 ilustra-se como se dá a construção das regras *Fuzzy* e seus valores de variáveis linguísticas.

Fura 4.5: Variáveis linguísticas



Fonte: Adaptado de Zeraoulia et al 2005.

Após a construção do algoritmo *Fuzzy*, será testado o sistema *on-line*, variando estes valores enquanto o algoritmo identifica os resultados dando uma resposta do estado do estator, através do grau de severidade. Isto serve para fornecer ao trabalho maior confiança e conferência na análise final dos resultados.

4.3.4. Softwares

Além dos hardwares e demais componentes auxiliares da parte física, faz-se de extrema importância detalhar os softwares de utilização nesta experiência, visto

que é através destes que será aplicado diretamente o algoritmo *Fuzzy,* além dos gráficos trapezoidais de simulação.

4.3.4.1. Ambiente de programação (Arduino IDE)

O Arduino fornece uma ferramenta de software de programação de código aberto e fácil de usar, o qual compila o código e o carrega para a placa micro controladora. Esta interface é referida como Arduino *IDE* (*Integrated Development Environment*) que significa Ambiente de Desenvolvimento Integrado.

É através desta interface de linhas de código que será implementado o algoritmo de inteligência artificial da lógica "*Fuzzy*", algoritmo este que também fará a obtenção do valor RMS das correntes de estudo e da temperatura do motor. A linguagem de programação a ser usada se enquadra como uma mescla de C/C++ com uma Linguagem de Domínio Específico (DSL).

Para realização do trabalho, além de maior facilidade e eficiência com a lógica "*Fuzzy*" aplicada à Arduinos, será usada a biblioteca *Fuzzy* especialista em sistemas embarcados e Arduinos eFLL (*Embedded Fuzzy Logic Library*) desenvolvida pelo Robotic Research Group (*RRG*) na Universidade Estadual do Piauí (UESPI-Teresina), que será integrada à IDE.

4.3.4.2. Matlab

Para fins de melhor análise, além de ser um complemento ao resultado que virá do Arduino IDE, será utilizado o software simulador especialista em tratamento de matrizes e sinais Matlab. Nele, será utilizado uma toolbox própria para a inteligência *Fuzzy*, que irá gerar diagramas da inferência *Fuzzy* para a situação desejada. Ou seja, os resultados simulados no Matlab deverão ser coerentes com os resultados provenientes do Arduino.

4.4. ANÁLISE DOS DADOS

Por fim, a análise dos dados será realizada em cima das simulações do Matlab e dos resultados provenientes do Arduino, de maneira que seja possível executá-los on-line, isto é, simulando-os em tempo real. As condições impostas pela lógica *Fuzzy* deverão responder adequadamente a valores numéricos diferentes. Portanto, será uma análise comparativa sobre a expectativa do tipo de ensaio versus os resultados obtidos do experimento.

4.5. SÍNTESE DO CAPÍTULO

Neste capítulo foi abordado o aspecto prático do experimento de aquisição dos dados de corrente e temperatura para um motor em bom estado e com sobrecarga.

A partir destes dados e do conhecimento do especialista, serão formuladas as regras *Fuzzy*, as quais após sua implementação deverão gerar o mesmo resultado que o especialista, garantindo a confiabilidade do sistema.

No próximo capítulo serão abordados os aspectos de programação para que sejam feitas as aquisições dos dados de temperatura e de corrente em tempo real, utilizando os sensores apresentados neste capítulo e em seguida como estes dados foram inseridos na lógica *Fuzzy* e quais regras foram utilizadas.

5. AQUISIÇÃO DE DADOS E IMPLEMENTAÇÃO DA LÓGICA *FUZZY*

5.1. AQUISIÇÃO DOS VALORES DA CORRENTE DO MOTOR

Para aquisição dos valores das correntes do motor foram utilizados três sensores do tipo SCT-013-020 indicados na figura 4.1. Este sensor pode medir valores de 0 até 20 A de corrente alternada. Em sua saída é fornecido valores entre 0 a 1 V conforme indicado na Figura 5.1. Esses valores de saída são proporcionais ao valor de corrente percorrido no condutor principal.

Figura 5.1: Datasheet YDHC SCT-013-020

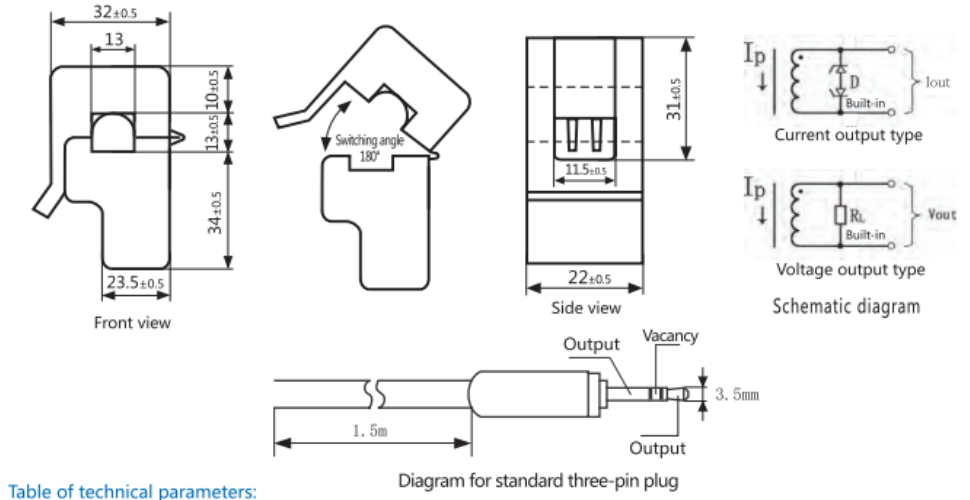


Table of technical parameters:

Model	SCT-013-000	SCT-013-005	SCT-013-010	SCT-013-015	SCT-013-020
Input current	0-100A	0-5A	0-10A	0-15A	0-20A
Output mode	Current/33mA	Voltage/1V	Voltage/1V	Voltage/1V	Voltage/1V
Model	SCT-013-025	SCT-013-030	SCT-013-050	SCT-013-060	SCT-013-070
Input current	0-25A	0-30A	0-50A	0-60A	
Output mode	Voltage/1V	Voltage/1V	Voltage/1V	Voltage/1V	

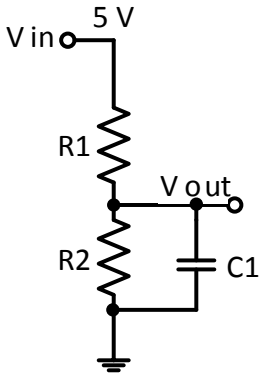
Fonte: https://nicegear.nz/obj/pdf/SCT-013-datasheet.pdf

Inicialmente tinha-se um sinal de tensão alternada variando entre 2,5 V positivo e 2,5 V negativo. O microcontrolador Arduino não pode medir tensão negativa, então se fez necessário somar 2,5 V ao sinal para que ele varie entre 0 V a 5 V. Sendo assim, foi inserido um circuito divisor de tensão para cada sensor usando a alimentação de 5 V que a placa Arduino fornece. Considerou-se para o primeiro

sensor os resistores R1 e R2 iguais a 10 kΩ, e com isso, a tensão sobre eles será igual, pois os 5 V provenientes do Arduino se dividirá igualmente entre eles.

Foi adicionado em paralelo com o resistor R2 um capacitor de 100 µF para filtrar o sinal de saída entregue pelo sensor de corrente, sendo assim, obteve-se uma menor variação nos valores de correntes indicados no software. Foi realizado o mesmo procedimento para os outros dois sensores. O esquema de ligação está representado na figura 5.2.

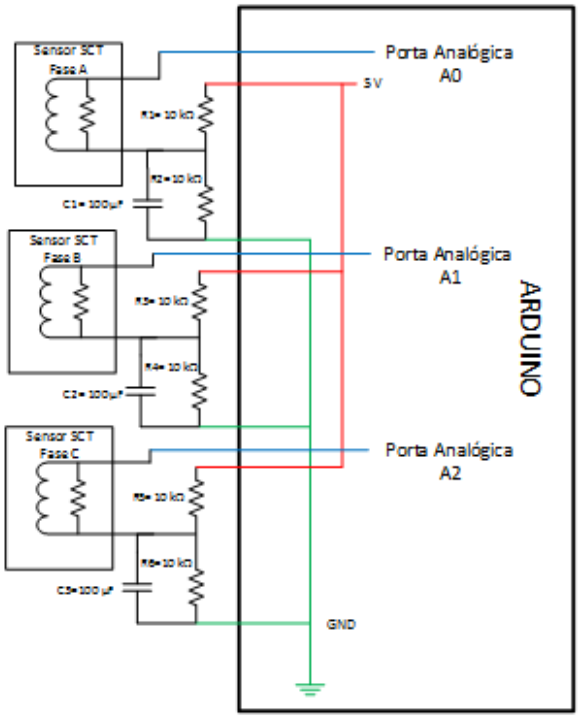
Figura 5.2: Divisor de tensão e capacitor referentes ao primeiro sensor



Fonte: Autoria própria

O esquema elétrico para aquisição dos valores das correntes referente às três fases está indicado na Figura 5.3.

Figura 5.3: Esquema de ligação dos sensores ao Arduino



Fonte: Autoria própria

Para averiguar se as correntes realmente estavam sendo aferidas corretamente foi utilizado um multimedidor trifásico, indicado na Figura 5.4.

Figura 5.4: Multimedidor Trifásico Mult - K

Fonte: Autoria própria

Este medidor pode aferir as correntes, tensões e consequentemente a potência.

5.1.1. Implementação da Lógica para Aquisição das Correntes dos Motores no Arduino

Os valores de corrente obtidos tinham que ser processados no software, para isso foi necessário incluir no código uma biblioteca chamada “EmonLib.h”, que é a responsável por realizar os cálculos para encontrar o valor eficaz da corrente a ser medida, baseando-se nas amostras colhidas pelas entradas analógicas do Arduino.

5.1.2. Descrição do Código que Calcula a Corrente Eficaz

1. #include "EmonLib.h"

A linha 1 refere-se à Inclusão da biblioteca *EmonLib* no código, que é responsável pelo cálculo da corrente eficaz.

2. EnergyMonitor SensorSCT_A;
3. EnergyMonitor SensorSCT_B;
4. EnergyMonitor SensorSCT_C;

As linhas 2, 3 e 4 referem-se ao objeto que será utilizado para representar o sensor SCT referente as fases A, B e C.

5. int pinSCT_A = A0;
6. int pinSCT_B = A1;
7. int pinSCT_C = A2;

As linhas 5, 6 e 7 referem-se aos pinos de entrada analógicos utilizados para aquisição dos dados fornecidos pelos sensores de corrente.

8. SensorSCT_A.current(pinSCT_A, 25);

```
9.  SensorSCT_B.current(pinSCT_B, 25);
10. SensorSCT_C.current(pinSCT_C, 25);
```

Nas linhas 8, 9 e 10 é onde a função *current* da biblioteca *EmonLib* é chamada, e nela foram atribuídos os valores obtidos nas portas de entrada analógica do arduino referentes às correntes I_a, I_b e I_c.

```
11. float IA_rms = SensorSCT_A.calcIrms(1480);
12. float IB_rms = SensorSCT_B.calcIrms(1480);
13. float IC_rms = SensorSCT_C.calcIrms(1480);
```

As linhas 11, 12 e 13 referem-se à chamada da função *calcIrms()* da biblioteca, onde é atribuído os valores de corrente às variáveis *IA_rms, IB_rms e IC_rms*. O valor entre parênteses representa o número de amostras que a função irá ler dos sensores SCT conectados ao Arduino, e assim realizará o cálculo dos valores eficaz de cada corrente.

5.2. AQUISIÇÃO DA TEMPERATURA

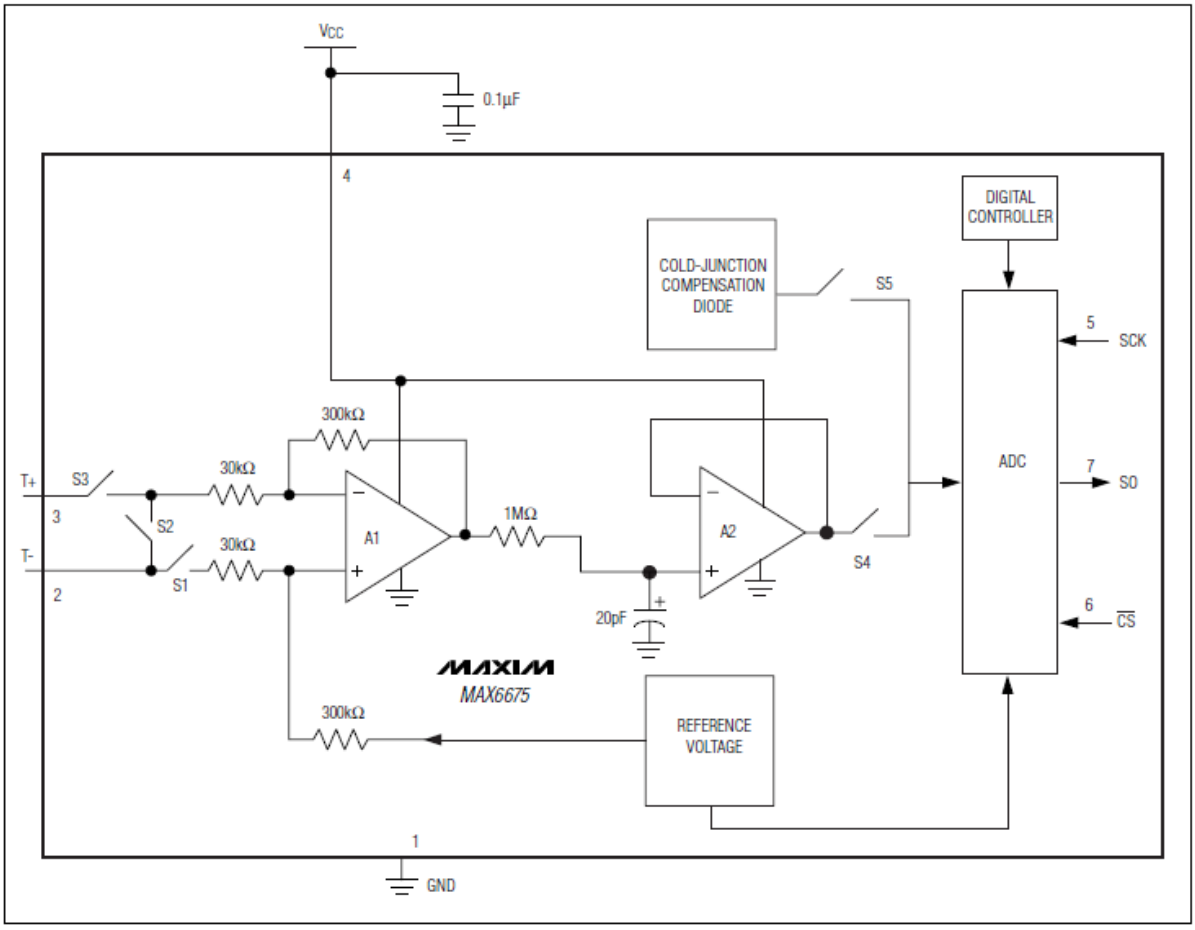
Como já mencionado, a variável temperatura passa a ser o diferencial ou incremento deste trabalho em relação a outros similares. A temperatura é um fator de extrema relevância neste experimento devido a esta apresentar um peso forte na tomada de decisão da lógica *Fuzzy*, já que tanto um aumento quanto diminuição excessivos desta são preocupantes para o operador considerando uma atuação normal do MIT.

É considerado normal para os dados de placa do MIT de estudo valores de temperatura próximos a 40 ºC, isto em operações normais. Aqui é válido ressaltar o fato de que o motor não atinge esta temperatura de maneira rápida, ou seja, fazem-se necessários alguns minutos para que o motor gradativamente estabeleça-se próximo a esta temperatura.

A composição interna deste módulo leitor de temperatura se dá pela figura 5.5 através de um diagrama de blocos, na qual é possível observar-se os amplificadores de sinal com filtros capacitivos e também o conversor analógico-digital para ligação

com o microcontrolador. Isto porque o sinal de saída é muito baixo, e nem mesmo o multímetro conseguiria ler este sinal por muito tempo caso não houvesse o circuito amplificador, ainda mais se a temperatura se elevar intensamente.

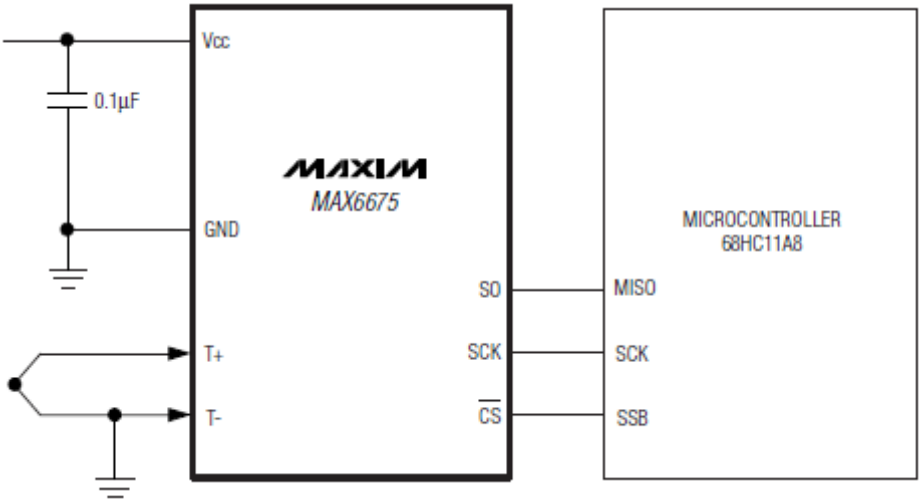
Figura 5.5: Diagrama de Blocos da composição Interna do módulo leitor de temperatura



Fonte: Disponível em <https://cdn-shop.adafruit.com/datasheets/MAX6675.pdf> Acesso em 04 jun às 10:40

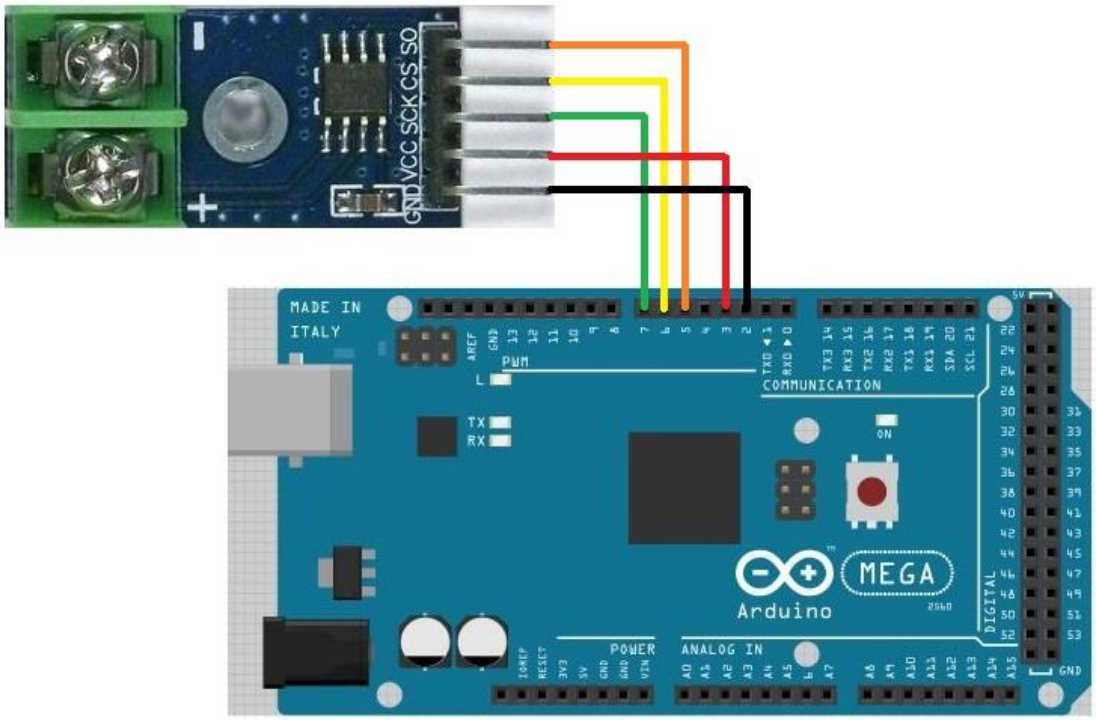
O esquema de ligação do módulo de sensoriamento com o microcontrolador Arduino MEGA se dá pelas figuras 5.6 e 5.7 a seguir.

Figura 5.6: Esquema genérico de ligação do sensor com um microcontrolador



Fonte: Disponível em <https://cdn-shop.adafruit.com/datasheets/MAX6675.pdf> Acesso em 04 jun 2018 às 10:45

Figura 5.7: Esquema de ligação do sensor com as portas de entrada do Arduino MEGA



Fonte: Autoria própria

As portas digitais 2 e 3 foram configuradas como alimentação do módulo.

5.2.1. Implementação do Código Referente ao Sensoriamento da Temperatura

Nesta seção serão explicados de maneira breve e simples os principais comandos usados na lógica de leitura da temperatura.

Primeiramente, foi usada uma biblioteca própria do módulo de leitura de temperatura (max6675.h) e também uma biblioteca referente à leitura por cabo/sonda (wire.h):

- #include <Wire.h>
- #include <max6675.h>

Após a inclusão das bibliotecas fez-se como de costume a declaração das variáveis necessárias para o programa e também a associação dos pinos digitais de leitura do Arduino com os pinos do modulo max6675. Nesta etapa também foi possível setar 2 pinos digitais como alimentação do sensor, de maneira a deixar livres para conexão mais pinos analógicos para os circuitos referentes ao sensoriamento das correntes:

- int thermoDO = 5;
- int thermoCS = 6;
- int thermoCLK = 7;
- MAX6675 thermocouple(thermoCLK, thermoCS, thermoDO);
- int vccPin = 3;
- int gndPin = 2;
- int amostragem = 10;

Depois das declarações de variáveis primárias, entra-se nos ajustes (*setup*) e depois na área de execução dos loops do programa para que gerem os resultados desejados. A parte da temperatura é dada através da criação de uma nova variável de temperatura iniciada em zero, desta vez numérica e não linguística. Ou seja, é a variável que receberá os dados de temperatura. Por fim, criou-se um *loop* necessário para a leitura das amostras (até 10) e após colocou-se a média das temperaturas pela amostragem, para melhor confiança dos resultados.

- void loop() {
- float temper = 0;
- for(int index =0; index < amostragem; index++){
- temper = thermocouple.readCelsius() + temper;
- delay(200);
- }
- temper = temper / amostragem;

Com estas linhas de código, o sensoriamento e leitura de temperatura puderam ser realizados e ao fim do programa ser lançado na tela junto aos valores das correntes e do grau de severidade da sobrecarga para efeito de certificação dos resultados.

5.3. IMPLEMENTAÇÃO DA LÓGICA FUZZY NO ARDUINO

5.3.1. Biblioteca Effl

Para a implementação da lógica *Fuzzy* no Arduino utilizamos a biblioteca eFFL, desenvolvida pelo Robotic Research Group (Grupo de Pesquisas em Robótica) na Universidade Estadual do Piauí (UESPI), escrita em C++/C, usa apenas a biblioteca padrão da linguagem C “stdlib.h”, por isso a eFFL é uma biblioteca destinada não somente a Arduino, mas qualquer sistema que tenha seus comandos escritos em C.

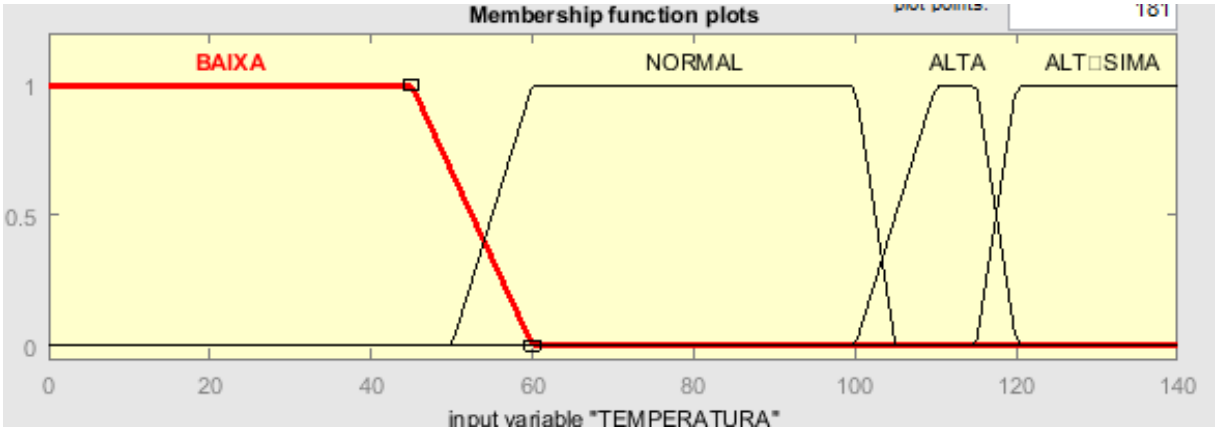
Não possui limitações explícitas de quantidade de conjuntos *Fuzzy*, regras *Fuzzy,* entradas ou saídas, estas se limitam a capacidade de processamento e armazenamento de cada microcontrolador. A biblioteca utiliza o processo mínimo de Mamdani para a inferência e composição e centro de área para a defuzzyficação, no universo contínuo.

5.3.2. Delimitação das Funções de Pertinência

Para a delimitação dos valores das funções de pertinência de entrada e saída, contou-se com o auxilio do professor orientador que atuou como especialista e

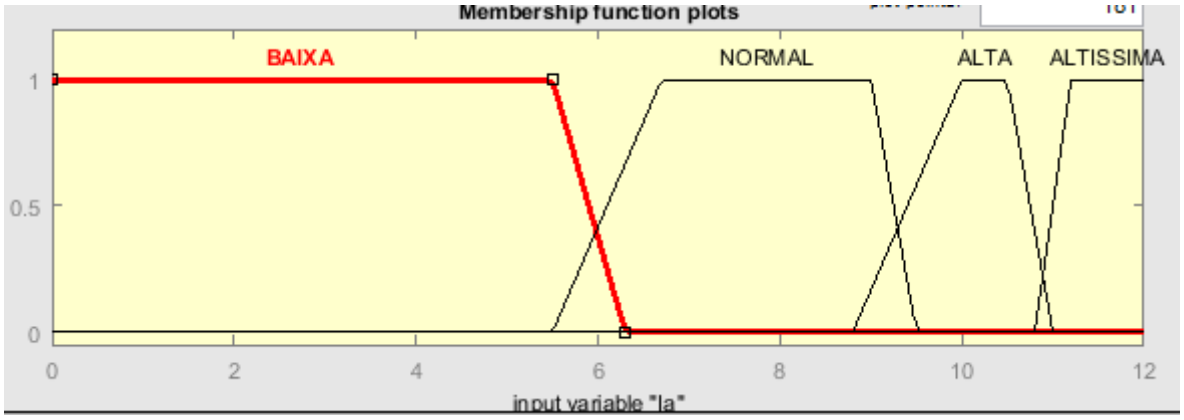
através de ensaios no motor, determinou-se os intervalos a serem seguidos para a temperatura de cabeça de bobina e as três correntes. Os gráficos das funções de pertinência são mostrados pelas figuras 6.1 à 6.3.

Figura 5.5: Função de Pertinência para Temperatura



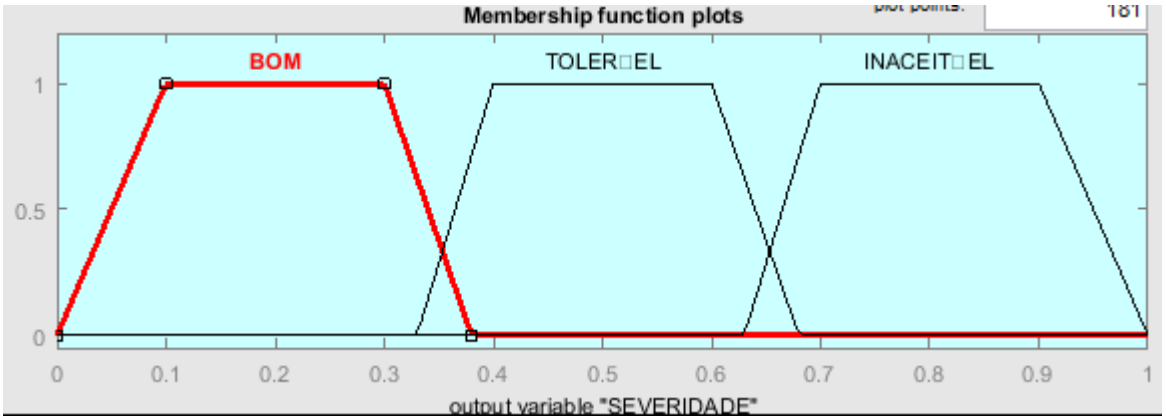
Fonte: Autoria própria

Figura 5.6: Função de pertinência para as três correntes



Fonte: Autoria própria

Figura 5.7: Função de pertinência para a severidade



Fonte: Autoria própria

Estes valores foram inseridos primeiramente no MatLab, pois o programa possibilita a representação gráfica das funções.

5.3.3. Declaração das funções de pertinência no *software*

Para inserir as funções de pertinência no *software* do Arduino foi necessário seguir os seguintes passos.

- Declaração dos intervalos e nomes das variáveis linguísticas;
  EX: FuzzySet* T_baixa = new FuzzySet(-50, -5, 45, 60);

  Neste caso, foi montado um trapézio que representa os valores de temperatura que são considerados como baixos.

- Inserção das funções como entradas e saída do processo *Fuzzy.*

  EX: FuzzyInput* Temp = new FuzzyInput(1);

  Temp->addFuzzySet(T_baixa);

  Neste caso, foi adicionada uma função de pertinência de entrada de entrada cujo nome é Temp e foi incluído o trapézio montado anteriormente.

5.3.4. Confecção Das Regras

Novamente, sob a instrução do especialista, foram montadas 40 regras que abrangem o escopo do projeto:

1. Se temperatura é altíssima ou Ia é altíssima ou Ib é altíssima ou Ic é altíssima então severidade é inaceitável;
2. Se Ia é baixa e Ib é baixa e Ic é baixa então severidade é bom;

3. Se Ia é baixa e Ib é baixa e Ic é média então severidade é tolerável;
4. Se Ia é baixa e Ib é baixa e Ic é alta então severidade é inaceitável;
5. Se Ia é baixa e Ib é média e Ic é baixa então severidade é tolerável;
6. Se Ia é baixa e Ib é média e Ic é média então severidade é tolerável;
7. Se Ia é baixa e Ib é média e Ic é alta então severidade é inaceitável;
8. Se Ia é baixa e Ib é alta e Ic é baixa então severidade é inaceitável;
9. Se Ia é baixa e Ib é alta e Ic média então severidade é inaceitável;
10. Se Ia é baixa e Ib é alta e Ic é alta então severidade é inaceitável;
11. Se Ia é média e Ib é baixa e Ic é baixa então severidade é tolerável;
12. Se Ia é média e Ib é baixa e Ic é média então severidade é tolerável;
13. Se Ia é média e Ib é baixa e Ic é alta então severidade é inaceitável;
14. Se Ia é média e Ib é média e Ic é baixa então severidade é tolerável;
15. Se Ia é média e Ib é média e Ic é média então severidade é bom;
16. Se Ia é média e Ib é média e Ic é alta então severidade é tolerável;
17. Se Ia é média e Ib é alta e Ic é baixa então severidade é inaceitável;
18. Se Ia é média e Ib é alta e Ic é média então severidade é tolerável;
19. Se Ia é média e Ib é alta e Ic é alta então severidade é tolerável;
20. Se Ia é alta e Ib é baixa e Ic é baixa então severidade é inaceitável;
21. Se Ia é alta e Ib é baixa e Ic é média então severidade é inaceitável;
22. Se Ia é alta e Ib é baixa e Ic é alta então severidade é inaceitável;
23. Se Ia é alta e Ib é média e Ic é baixa então severidade é inaceitável;
24. Se Ia é alta e Ib é média e Ic é média então severidade é tolerável;
25. Se Ia é alta e Ib é média e Ic é alta então severidade é tolerável;
26. Se Ia é alta e Ib é alta e Ic é baixa então severidade é inaceitável;
27. Se Ia é alta e Ib é alta e Ic é média então severidade é tolerável;
28. Se Ia é alta e Ib é alta e Ic é alta então severidade tolerável;
29. Se temperatura é alta e Ia é baixa e Ib é baixa e Ic é média então severidade é inaceitável;
30. Se temperatura é alta e Ia é baixa e Ib é média e Ic é baixa então severidade é inaceitável;
31. Se temperatura é alta e Ia é baixa e Ib é média e Ic é média então severidade é inaceitável;
32. Se temperatura é alta e Ia é média e Ib é baixa e Ic é baixa então severidade é inaceitável;

33. Se temperatura é alta e Ia é média e Ib é baixa e Ic é média então severidade é inaceitável;
34. Se temperatura é alta e Ia é média e Ib é média e Ic é baixa então severidade é inaceitável;
35. Se temperatura é alta e Ia é média e Ib é média e Ic é alta então severidade é inaceitável;
36. Se temperatura é alta e Ia é média e Ib é alta e Ic é média então severidade é inaceitável;
37. Se temperatura é alta e Ia é média e Ib é alta e Ic é alta então severidade é inaceitável
38. Se temperatura é alta e Ia é alta e Ib é média e Ic média então severidade é inaceitável;
39. Se temperatura é alta e Ia é alta e Ib é média e Ic é alta então severidade é inaceitável;
40. Se temperatura é alta e Ia é alta e Ib é alta e Ic é média então severidade é inaceitável;

5.3.5. Declaração das Regras no *Software*

Para inserir as regras no *software* do Arduino foi necessário seguir os seguintes passos:

- Declaração dos antecedentes;
  Ex: FuzzyRuleAntecedent* TempT_altissimaOrI1Ia_altissima = new FuzzyRuleAntecedent();
  TempT_altissimaOrI1Ia_altissima->joinWithOR(T_altissima, Ia_altissima);

  Neste caso, foi declarada a junção de duas funções de pertinência com a ligação OU. A biblioteca só faz a junção dois a dois, por isto para associar quatro funções precisa-se declarar dois antecedentes e associa-los entre sí.

- Associação de antecedentes;
  Ex: FuzzyRuleAntecedent*ifTempT_altissimaOrI1Ia_altissimaOr

```
I2Ib_altissimaOrI3Ic_altissima  = new FuzzyRuleAntecedent();
 ifTempT_altissimaOrI1Ia_altissimaOrI2Ib_altissimaOrI3Ic_altissima-
>joinWithOR(TempT_altissimaOrI1Ia_altissima,
I2Ib_altissimaOrI3Ic_altissima);
```

Neste caso foram associados dois antecedentes OU com outro OU, compondo assim o antecedente necessário.

- Declaração do consequente;
  Ex: FuzzyRuleConsequent* thenSisInaceitavel = new FuzzyRuleConsequent();
  thenSisInaceitavel->addOutput(Inaceitavel);

Neste caso, foi adicionado o consequente “Inaceitável” à regra e ligado este à saída *Fuzzy* que foi declarada anteriormente.

- Montagem da regra.

  Ex: FuzzyRule* fuzzyRule1 = new FuzzyRule(1, ifTempT_altissimaOrI1Ia_altissimaOrI2Ib_altissimaOrI3Ic_altissima, thenSisInaceitavel);

  fuzzy->addFuzzyRule(fuzzyRule1);

Neste caso, o antecedente foi ligado com o consequente e adicionou-se a regra ao banco de dados.

5.3.6. Montando Estrutura de Repetição

Após montadas as 40 regras, terminou-se a declaração de variáveis e começou-se a montagem da estrutura de repetição.

- Variáveis de entrada;

  O microcontrolador fará a aquisição dos dados analógicos e os enviará como dados de entrada no programa *Fuzzy.*

- Variáveis de saída.

  float Sev = fuzzy->defuzzify(1);

O programa, por sua vez, fará todo o processo de inferência e retornará uma variável de saída defuzzyficada.

Para este valor de saída, foi montada uma estrutura condicional que fará o acionamento dos LED's, que nos indicarão a resposta do processo.

Ex: if (Sev <= 0,4) {

digitalWrite(led_verde, HIGH); //acende led verde e apaga vermelho e amarelo

digitalWrite(led_vermelho, LOW);

digitalWrite(led_amarelo, LOW); }

5.4. SÍNTESE DO CAPÍTULO

Neste capítulo abordou-se como foi feita a aquisição dos dados de temperatura e correntes e a implementação destes dados na lógica *Fuzzy* no Arduino utilizando a biblioteca eFFL, além de como foram feitas as declarações das funções de pertinência , declaração e montagem das regras.

Com o auxílio do especialista, as regras foram montadas e inseridas no Arduino.

No próximo capítulo as variáveis serão inseridas na lógica *Fuzzy* em tempo real e após o processo de fuzzyficação e defuzzyficação, o Arduino retornará uma resposta do estado do motor de indução segundo a análise de suas correntes e da temperatura.

6. ENSAIOS E ANÁLISE DOS RESULTADOS

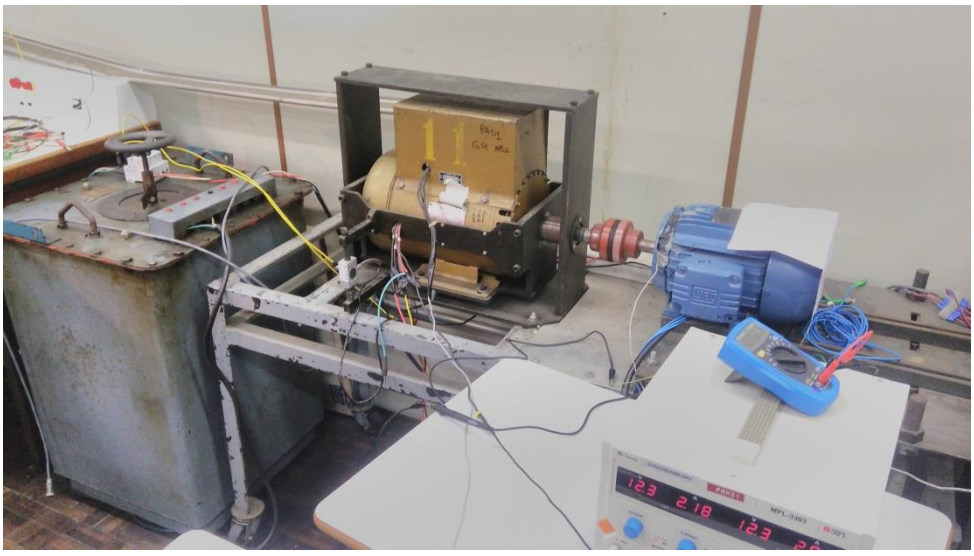
Como dito anteriormente, para a validação do programa os experimentos foram realizados e os dados coletados a partir de testes conduzidos em laboratório, sob supervisão do professor orientador.

Em seguida, estes mesmos valores foram inseridos no MatLab para verificar a congruência dos resultados.

6.1. DETERMINAÇÃO DE INTERVALOS DE OPERAÇÃO

Para iniciar os testes, precisou-se primeiramente determinar em quais intervalos de operação o motor iria ser sujeitado. Para isso, o MIT foi acoplado a um alternador, como mostrado na figura 6.1, e com isso conseguiu-se controlar a carga aplicada ao MIT.

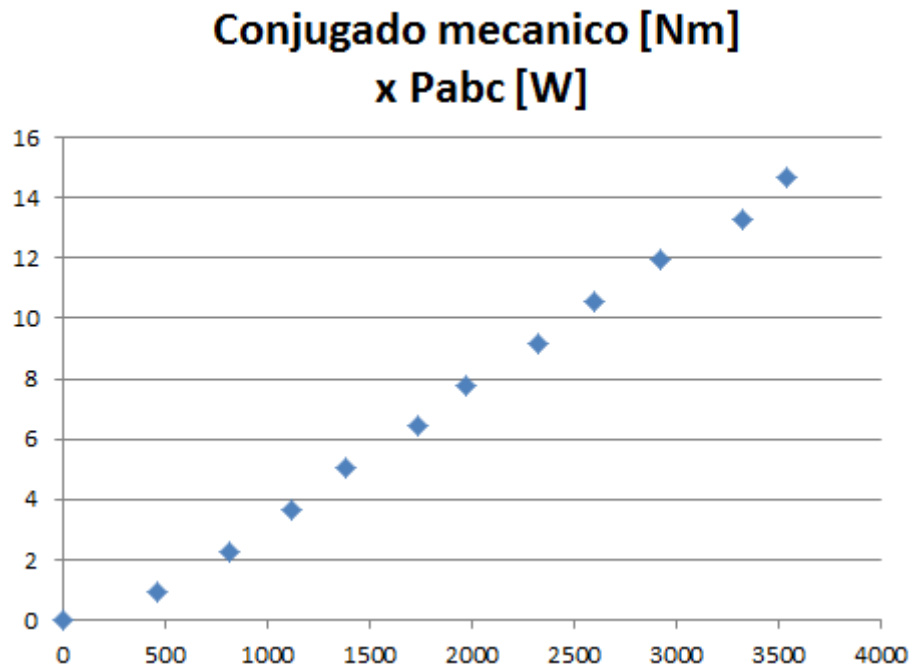
Figura 6.1: MIT acoplado a um motor CC



Fonte: Autoria própria

Utilizando uma célula de carga acoplada ao alternador, foi possível observar a carga sendo fornecida pelo MIT. Conseguiram-se então os dados de potência e do conjugado mecânico, para com isso montar-se o gráfico mostrado na Figura 6.2.

Figura 6.2: Potência X Conjugado mecânico



Fonte: Autoria própria

Como este motor fornece em plena carga um conjugado mecânico de 12 Nm (Newton x metro), conseguiu-se obter o valor do conjugado para o fator de serviço (115%): cerca de 13,3 Nm.

6.2. ENSAIOS, SIMULAÇÕES E RESULTADOS

A seguir serão apresentados os ensaios realizados para validar a programação. As relações entre os indicadores de estados (LEDs) e o grau de severidade da sobrecarga são dadas da seguinte maneira:

- LED Branco aceso → Severidade **BOA**
- LED Amarelo aceso → Severidade **TOLERÁVEL**
- LED Vermelho aceso → Severidade **INACEITÁVEL**

Aqui é importante informar que o valor de referência para o grau de severidade se dá numa escala de 1,00 a 10,00, sendo 5,00 e seus entornos (de 4,0 a 6,0) valores médios para esta classificação.

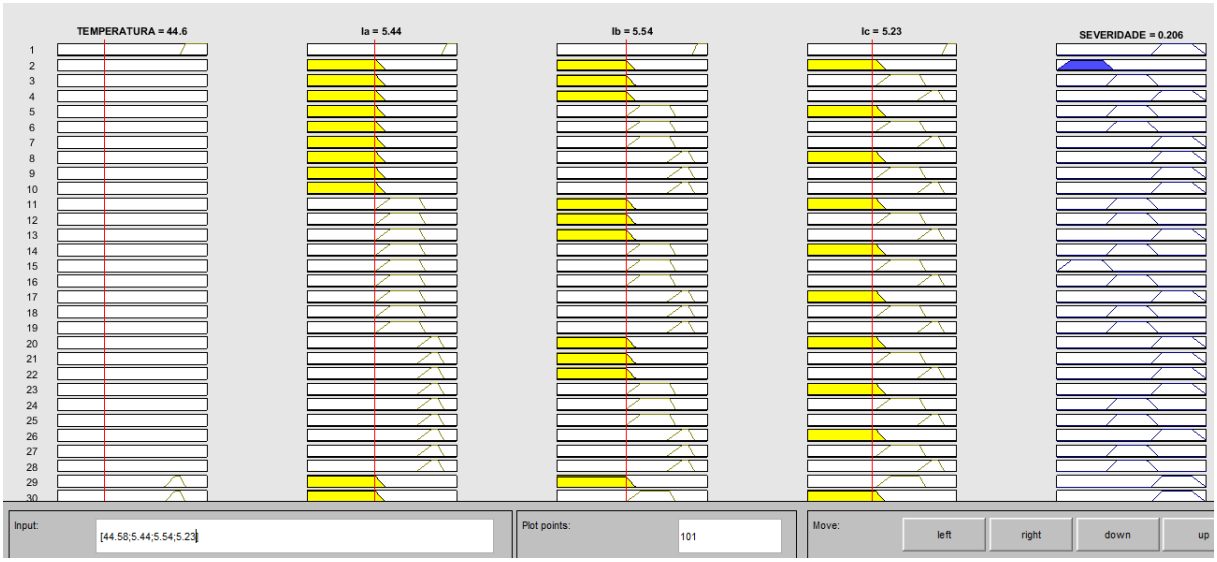
6.2.1. Ensaio com Motor a Vazio

Para a condição de motor a vazio, os dados recolhidos pelo programa foram:

- Corrente da fase 1 = 5.44 A;
- Corrente da fase 2 = 5.54 A;
- Corrente da fase 3 = 5.23 A;
- Temperatura em ºC = 44.58;
- Grau de severidade = 3.44;
- Led branco aceso.

A simulação no MatLab, é mostrada na Figura 6.3.

Figura 6.3: Simulação para motor a vazio



Fonte: Autoria própria

Na Toolbox do Matlab é possível simular os resultados inserindo os mesmos valores, esta então mostra todas as regras e onde as variáveis estão situadas nas funções de pertinência, mostrando também quais regras estão sendo ativadas.

6.2.2. Ensaio de Falta de fase com motor a vazio

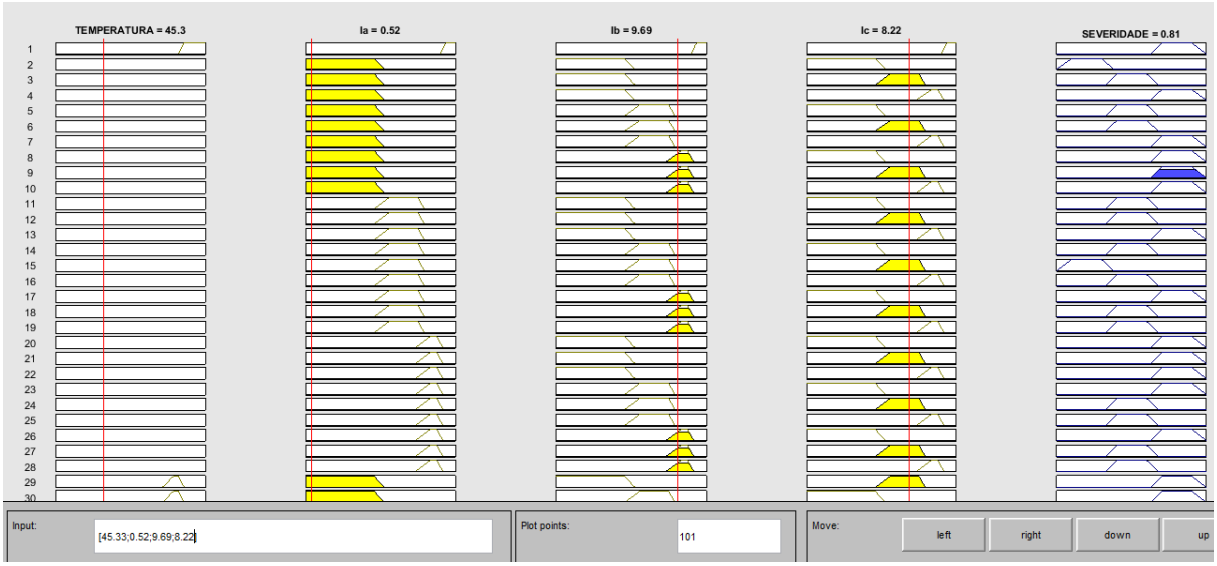
Para simular a falta de fase com motor a vazio, retirou-se o cabo de alimentação de seu borne da bancada, pois isto simula um rompimento repentino do cabo.

Os dados recolhidos pelo programa foram:

- Corrente da fase 1 = 0.52 A;
- Corrente da fase 2 = 9.69 A;
- Corrente da fase 3 = 8.22 A;
- Temperatura em ºC = 45.33;
- Grau de severidade = 8.08;
- Led vermelho aceso.

A simulação no MatLab, é mostrada na Figura 6.4.

Figura 6.4; Simulação de falta de fase com motor a vazio



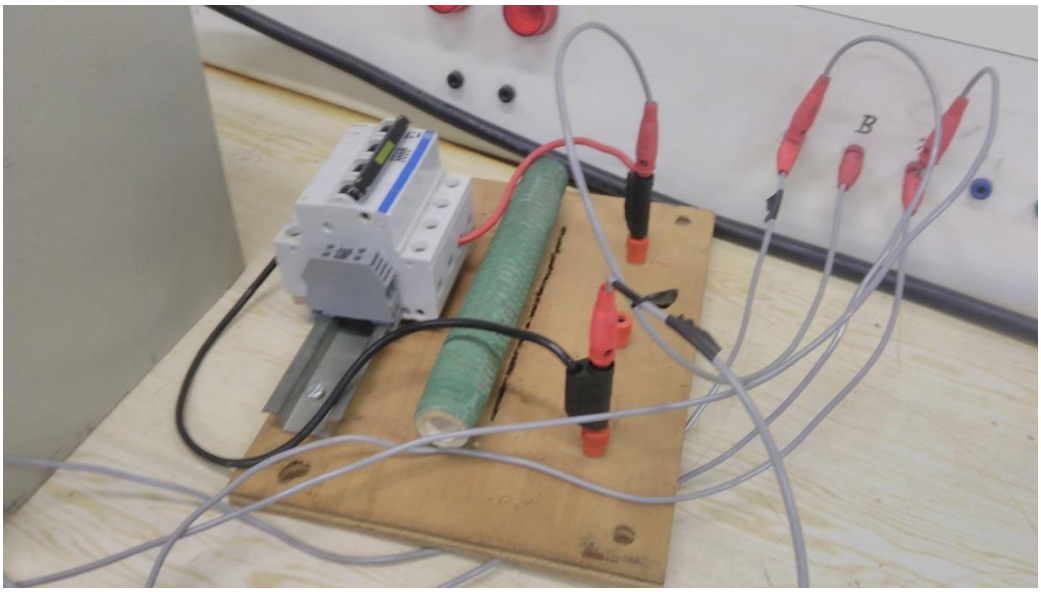
Fonte: Autoria própria

Deve-se ressaltar que este ensaio foi realizado durante um curto período de tempo para não causar danos no motor, por isso que a temperatura permaneceu baixa.

6.2.3. Ensaio de Desequilíbrio de tensão em plena carga

Para simular um desequilíbrio de tensão de 4% em plena carga, adicionou-se um resistor de 1Ω x 1kW em série com uma das bobinas, como mostrado na figura 6.5.

Figura 6.5: Resistor em série para causar desequilíbrio de tensão



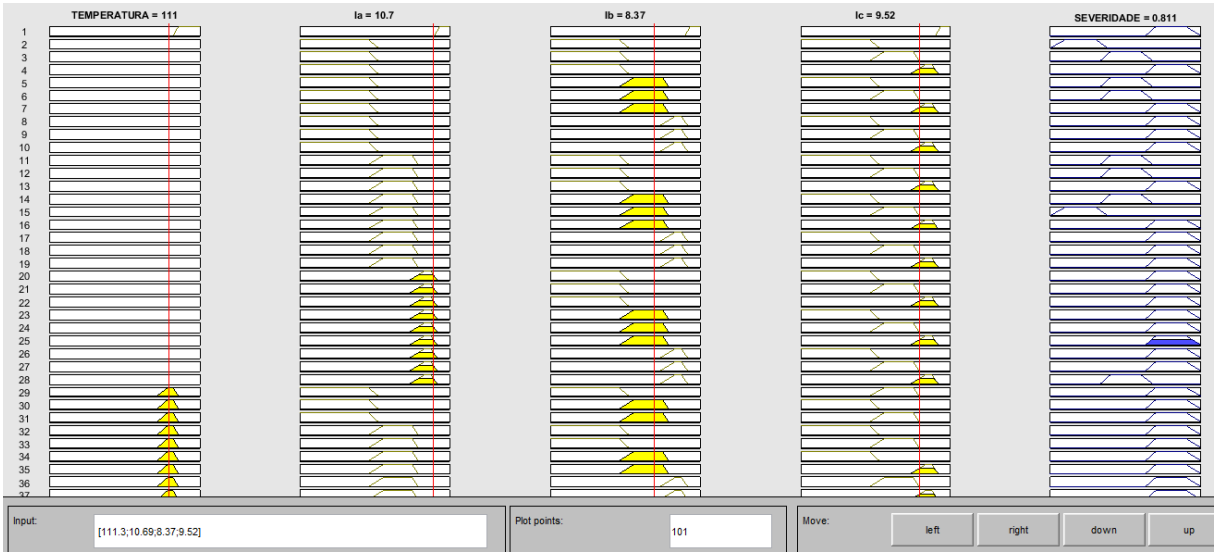
Fonte: Autoria própria

Os dados recolhidos pelo programa foram:

- Corrente da fase 1 = 10.69 A;
- Corrente da fase 2 = 8.37 A;
- Corrente da fase 3 = 9.52 A;
- Temperatura em ºC = 111.33;
- Grau de severidade = 8.11;
- Led vermelho aceso.

A simulação no MatLab, é mostrada na Figura 6.6

Figura 6.6: Simulação de desequilíbrio de tensão em plena carga



Fonte: Autoria própria

Como a temperatura estava elevada, a regra que resultou no grau de severidade mostrado foi a regra 39 (Se temperatura é alta e Ia é alta e Ib é média e Ic é alta então severidade é inaceitável).

6.2.4. Ensaio com Fator de serviço a 115%

Para simular uma carga excedendo a plenitude, aumentou-se a corrente de enrolamento do alternador, com isso as correntes do motor entraram na faixa do fator de serviço, que é o máximo que o motor pode operar segundo o fabricante. Este aumento de corrente foi executado via conexão de uma fonte de tensão/corrente CC ao alternador de maneira que esta injeção de corrente de campo pudesse ser lida e controlada via multímetro (em valores de tensão) devido à célula de carga instalada neste motor. O valor de referência para fator de serviço a 115% na saída da célula de carga resultou em 5,4 mV. A figura 6.7 demonstra esta situação.

Figura 6.7: Injeção e controle de corrente de enrolamento no alternador

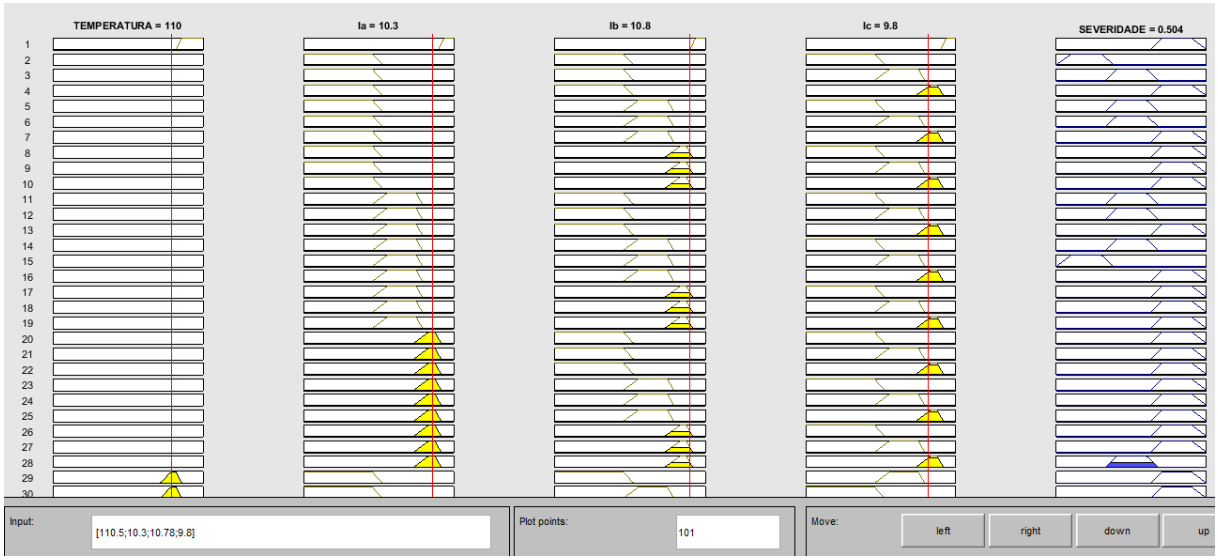
Fonte: Autoria própria

Os dados recolhidos pelo programa foram:

- Corrente da fase 1 = 10.34 A;
- Corrente da fase 2 = 10.78 A;
- Corrente da fase 3 = 9.83 A;
- Temperatura em ºC = 110.47;
- Grau de severidade = 5.34;
- Led amarelo aceso.

A simulação no MatLab, é mostrada na Figura 6.8

Figura 6.8: Simulação fator de serviço



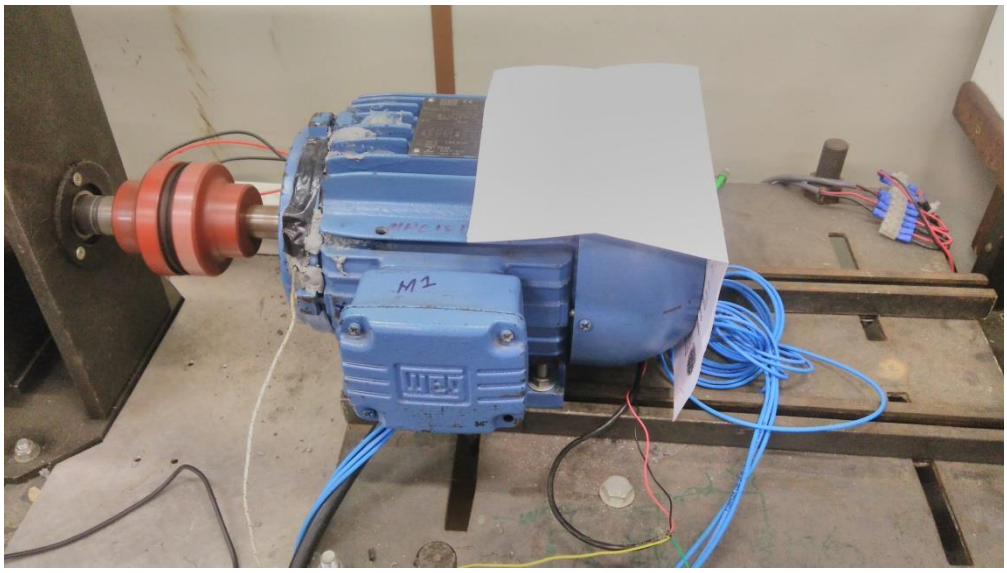
Fonte: Autoria própria

A regra que resultou no grau de severidade mostrado foi a regra 28 (Se Ia é alta e Ib é alta e Ic é alta então severidade tolerável)

6.2.5. Ensaio de ventilação bloqueada em fator de serviço 115%

Para bloquear a ventilação operando em fator de serviço a 115%, simplesmente colocou-se uma folha de papel na entrada de ar do motor. Isto simula o acúmulo de partículas de sujeira ou qualquer outra condição que impeça à entrada de ar no MIT. A figura 6.9 representa esta situação.

Figura 6.9: Bloqueio de ventilação do MIT em fator de serviço a 115%



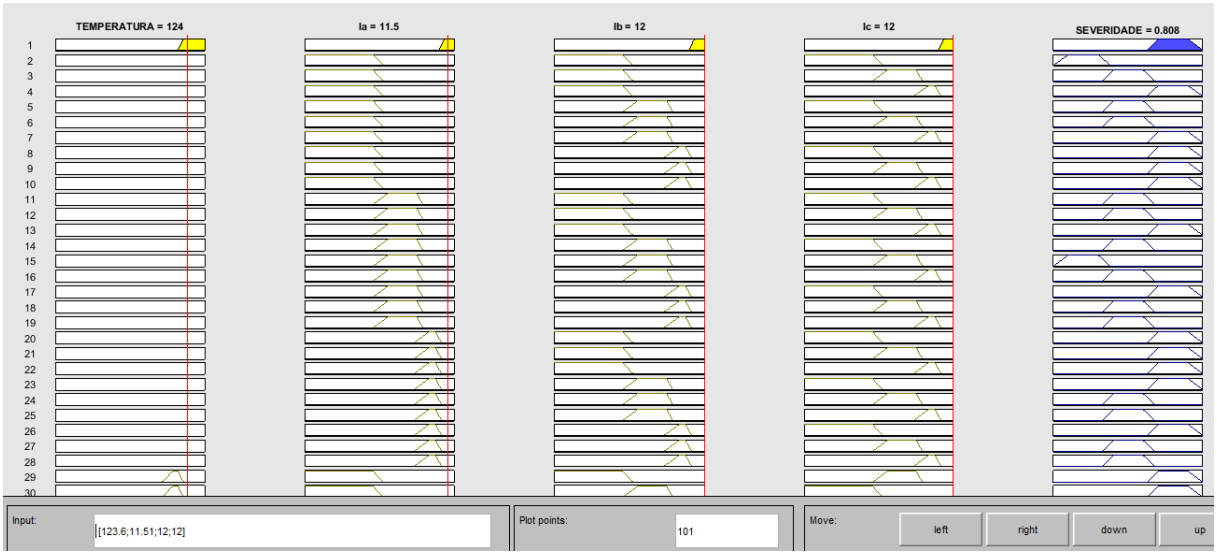
Fonte: Autoria própria

Os dados recolhidos pelo programa foram:

- Corrente da fase 1 = 11.51 A;
- Corrente da fase 2 = 12.40 A;
- Corrente da fase 3 = 12.31 A;
- Temperatura em ºC = 123.58;
- Grau de severidade = 8.08;
- Led vermelho aceso.

A simulação no MatLab, é mostrada na Figura 6.10.

Figura 6.10: Simulação de ventilação fechada em fator de serviço 115%



Fonte: Autoria própria

Como a temperatura e as correntes estavam na gama “Altíssimo” a primeira regra foi acionada (Se temperatura é altíssima ou Ia é altíssima ou Ib é altíssima ou Ic é altíssima então severidade é inaceitável).

6.3. ANÁLISE DOS RESULTADOS

Para efeitos de análise dos resultados, têm-se na tabela abaixo os tipos de ensaios realizados, a expectativa dos resultados provenientes das simulações e os valores obtidos na prática.

Tabela 6.1: Tabela de expectativa e resultados

ENSAIO	EXPECTATIVA		RESULTADOS						
	Led aceso esperado	Grau de severidade esperado	Corrente Fase 1 (em A)	Corrente Fase 2 (em A)	Corrente Fase 3 (em A)	Temp. em °C	Led aceso	Grau de severidade obtido	
A vazio	Branco	Baixo	5,44	5,54	5,23	44,58	Branco	3,44	Baixo
Falta de fase a vazio	Vermelho	Alto	0,52	9,69	8,22	45,33	Vermelho	8,08	Alto
Desequilibrio de tensão à plena carga	Vermelho	Alto	10,69	8,37	9,52	111,33	Vermelho	8,11	Alto
Fator de serviço a 115%	Amarelo	Médio	10,34	10,78	9,83	110,47	Amarelo	5,34	Médio
Ventilação bloqueada com fator de serviço a 115%	Vermelho	Alto	11,51	12,40	12,31	123,58	Vermelho	8,08	Alto

Fonte: Autoria própria

Com base nesta tabela, observa-se que os resultados obtidos foram correspondentes à expectativa, considerando o embasamento teórico que fundamenta os ensaios e a base científica e empírica fornecida pelo especialista.

No ensaio a vazio, percebe-se um menor nível das correntes atuantes se comparado aos demais ensaios, o que é coerente já que não há carga presente. Enquanto isto, a temperatura ao longo do tempo permanece no nível baixo, isto é, em torno dos 40 ºC como já mencionado. Logo, espera-se um comportamento bom para o motor sem valores extremos para as variáveis de corrente e temperatura, ou seja, o grau de severidade esperado para este ensaio é baixo. Logo, o LED a ser acesso deve ser o branco, o que de fato ocorreu e o grau obtido foi de 3,44, o que também corresponde ao esperado.

Já no ensaio da falta de fase a vazio, espera-se uma situação totalmente inaceitável, considerando que este é um dos piores defeitos a ocorrerem em MIT's (e também em outras máquinas). Com a falta de uma das fases, percebe-se a queda brusca (não foi nula devido à presença do capacitor de filtro na aquisição) de uma das correntes enquanto as outras são aumentadas de maneira proporcional para manter a potência entregue à carga. A temperatura por sua vez acaba não alterando muito, já que por algum tempo o motor consegue operar assim, mas se o tempo se prolongar, pode ocorrer a queima de algum enrolamento, o que elevaria muito mais a temperatura. O fato é que, para este tipo de ensaio, apenas a alteração em uma das correntes já é o suficiente para trazer à tona o estado de severidade alta, o que de fato ocorreu com seu valor de 8,08 e também trouxe consigo o LED vermelho aceso.

No ensaio de desequilíbrio de tensão à plena carga, encontra-se uma situação indesejada, já que se espera que a concessionária forneça de maneira igualitária as tensões trifásicas. Entretanto, às vezes o problema pode derivar do próprio local de trabalho, o qual pode acabar desequilibrando de forma leve uma das tensões em relação às outras ou mais de uma em relação ao valor monofásico de 127 V. Nas correntes adquiridas, não houve uma alteração significativa entre seus valores, o que é possível se considerar que uma máquina destas pode vir a operar com duas fases por certo tempo, apesar de ser algo indesejado também. Porém, no que se trata da temperatura, percebe-se um grande aumento no valor desta em comparação aos outros ensaios. Isto é de fato inaceitável, pois a temperatura estipulada para o *Fuzzy* encontra-se como tolerável (porém já no limite) no intervalo de 105 a 110ºC. Como ela está levemente acima deste valor, torna-se suficiente para acender o LED vermelho e aumentar o grau de severidade para alto (8,11), acarretando em outra congruência entre expectativa e resultado.

Para o ensaio de fator de serviço a 115%, obtido previamente pela relação do conjugado, tem-se uma situação limiar e delicada, já que a temperatura encontra-se no limite do tolerável e as correntes permanecem de maneira quase que idênticas, com leves discrepâncias. Para outras situações, este valor de 110,47 ºC deve sim ser encarado como intolerável. Entretanto, a base científica empírica do especialista aliada aos dados de placa do MIT e demais dados do fabricante explica que este valor de fator de serviço em 115% é o máximo que o motor pode operar como

situação de estado tolerável. Portanto, espera-se no fim que a situação de limiaridade seja concebida como tolerável e que o grau de severidade seja médio, o que de fato ocorreu, já que ambos os valores resultaram em LED amarelo aceso e 5,34, respectivamente. Aqui já é possível apontar uma das grandes vantagens de se utilizar um sistema *Fuzzy*, pois outras lógicas não dariam conta de estipular este valor nebuloso de severidade.

Por fim, tem-se o ensaio com ventilação bloqueada com fator de serviço em 115%. Este ensaio é, junto ao de falta de fase, um terrível problema de extremo cuidado a se ter em tratamentos de motores trifásicos, já que o bloqueio de sua ventilação pode ocasionar um superaquecimento do motor e danificar seria e permanentemente seus enrolamentos de estator, rotor, e também periféricos. Logo, é de se esperar um grande e mais rápido aumento da temperatura se comparado aos outros tipos de ensaios. O resultado foi a elevação da temperatura para imensos 123.58 ºC num intervalo mais rápido de tempo (o que já está no extremo de sua tolerância para o *Fuzzy*) e uma média de 12,07 A para as correntes, números estes que acenderam o LED vermelho e trouxeram consigo o grau de severidade de 8,08, ou seja, ambos corresponderam ao esperado. Aqui é válido ressaltar que o MIT em questão pertence à classe de isolamento B, tendo seu valor máximo de temperatura estipulado em 130ºC.

Com relação às simulações do Matlab, confirma-se a congruência dos resultados, visto que todos os valores de severidade ficaram muito próximos e alguns até se igualaram às simulações.

7. CONCLUSÃO

Conclui-se de maneira geral que o trabalho foi bem sucedido, considerando as premissas e condições pré-estabelecidas e também que os objetivos específicos e geral foram atingidos. O artigo de referência mencionado previamente foi reproduzido, porém com a adição de mais uma variável de entrada além das correntes das estatóricas: a temperatura de cabeça de bobina. Este foi o grande diferencial do trabalho, envolver a temperatura na análise da tomada de decisão do estado do motor através da sua inclusão na lógica *Fuzzy* do sistema microcontrolado. Além disso, o grau de severidade da sobrecarga foi correspondente para cada ensaio realizado, tendo sua sinalização dada pelos LED's, o que pode ser visto como uma facilitação visual ao tomador de decisão para a manutenção preditiva de motores, por exemplo.

As discrepâncias dos valores de correntes foram pequenas, e podem ser atribuídas tanto por descalibrações dos componentes de medição quanto a pequenas porcentagens de erros já estabelecidos nos manuais destes.

A lógica *Fuzzy* por sua vez, um dos grandes pilares neste trabalho, também se demonstrou correta ao detectar um valor intermediário do grau de severidade de 5,34, visto que outros tipos de lógicas poderiam não responder da mesma forma ou retornar somente valores extremos. Logo, aqui se tem um acréscimo na qualidade da tomada de decisão e também fortalece o termo "manutenção preditiva", já que este meio termo nebuloso que o *Fuzzy* é capaz de levar em conta pode ser crucial nas operações industriais. A maioria dos processos industriais não pode parar devido a problemas de máquinas ou devido à manutenções invasivas que poderiam ser evitadas, pois estes gargalos de tempo parado dos ativos de uma indústria implicam em dinheiro desperdiçado. Logo, este trabalho apresenta uma melhoria econômica no setor de manutenção preditiva através da automação de sensores, mesmo que em pequena escala. O fato dos componentes principais do experimento não serem caros também corrobora para a questão econômica.

Por fim, a partir do que foi feito neste trabalho pode-se ter um modelo tanto didático para melhorias futuras quanto um modelo industrial, se adaptado de maneira adequada e em maior escala de equipamentos e processadores de dados.

Outros estudos relativos às variáveis em função do tempo, como por exemplo o gradiente de elevação da temperatura, podem ser derivados e aprofundados a partir deste trabalho, visto que a gama de ensaios e situações de manutenção que um MIT apresenta é muito maior do que o abordado.

REFERÊNCIAS

ABNT NBR 5462: **Confiabilidade e mantenabilidade**: Rio de Janeiro 1994.

ABREU, Jose P. G.; EMANUEL, Alexander E. **Induction Motor Thermal Aging Caused by Voltage Distortion and Imbalance: Loss of Useful Life and Its Estimated Cost.** IEEE Industry Applications, vol.18, no. 1, 2002.

ANEEL. **Revisão da Regulamentação sobre a Qualidade do Produto no Sistema de Distribuição de Energia Elétrica**. Superintendência de Regulação dos Serviços de Distribuição – SRD. Braslília. Data 30/06/2011. Disponível em: <http://www.aneel.gov.br/aplicacoes/consulta_publica/documentos/NT_29_2011-SRD_ANEEL_ANEXO.pdf>. Acesso em: 30 out. 2017, 18:05.

BARBI, Ivo. **Teoria Fundamental do Motor de Indução**. Edição do Autor. Data 09/12/2011. Disponível em: <http://ivobarbi.com/teoria-fundamental-do-motor-de-inducao>. Acesso em: 30 Out. 2017 15:42.

BÁRDOSSY, A. DUCKSTEIN, L. **Fuzzy Rule-Based Modeling with Applications to Geophysical,** Biological and Engineering Systems CRC Press, 1995.

BONALDI, Erik Leandro. **Diagnóstico Preditivo de Avarias em Motores de Indução Trifásicos com MCSA e Teoria de Conjuntos Aproximados**. 2005. Dissertação (Doutorado em Ciências em Engenharia Elétrica).- Programa de Pós-Graduação em Engenharia Elétrica, Escola Federal de Engenharia de Itajubá, Itajubá, 2005.

BONNET, Austin H. e YUNG, Chuck. **Increased efficiency versus increased reliability.** IEEE Industry Applications Magazine, v. 14, n. 1. jan/fev.2008. Disponível em: <http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4432942>. Acesso em 30 Out 2017, 15:46.

BRANCO FILHO, **Gil. Dicionário de Termos de Manutenção, Confiabilidade e Qualidade**. 2 ed. Rio de Janeiro: Ciência Moderna, 2000.

CARDOSO, A. J. M.; Diagnóstico de avarias em motores de indução trifásicos. Coimbra Editora, 1991.

Carvalho, C. L. "**Sistemas Especialistas**". Instituto de Informática, UFG, 2006.

COX, E. **The Fuzzy Systems Handbook: A Practitioner's Guide to Building, Using and Maintaining Fuzzy systems.** A. P professional 1994.

DAS, S.; PURKAIT, P.; CHAKRAVORTI, S. **Space-Vector Characterization of Induction Motor Operating Conditions.** Fifteenth National Power Systems Conference, 2008.

Earl Cox, **The fuzzy systems handbook**, Morgan Kaufmann, USA, 1998.

https://blog.zerokol.com/2012/09/arduinofuzzy-uma-biblioteca-fuzzy-para.html
< Acesso em 11 de Maio de 2018; 15:50>

http://projetosdoborba.blogspot.com.br/2013/06/arduino-logica-fuzzy-sensacional.html <Acesso em 11 de Maio de 2018; 15:50>

http://www.abraman.org.br/arquivos/403/403.pdf <Acesso em 22 de setembro de 2017; 15:30>

FITZGERALD, A. E.; JR., C. Kingsley; UMANS, Stephen D. **Máquinas Elétricas**. 6. ed. Porto Alegre: Bookman, 2006.

http://www.brasil.gov.br/infraestrutura/2015/11/aneel-estimula-troca-de-motores-eletricos-para-promover-eficiencia-energetica <Acesso em: 25 abr. 2017 20:16>

http://www.feis.unesp.br/Home/departamentos/engenhariaeletrica/catalogo_weg_motores-eletricos_4-44.pdf <Acesso em 24 Nov. 2017 16:30>

INTECH. Predictive Maintenance by Electrical Signature Analysis to Induction Motors. Chapter. 20. Data 14/11/2012. Disponível em: http://www.intechopen.com/books/induction-motors-modelling-and-control/predictive-maintenance-by-electrical-signature-analysis-to-induction-motors <Acesso em: 27 Out. 2017, 11:15>

KARDEC, Alan; NASCIF, Júlio. **Manutenção: Função Estratégica**. 2. ed. Rio de Janeiro: Qualitymark, 2002.

JOUANNE, Annete von; BANERJEE, Basudeb. **Assessment of Voltage Unbalance.** IEEE Transactions on Power Delivery, vol.16, no. 4, 2001.
KARDEC, Alan; NASCIF, Júlio. **Manutenção: Função Estratégica**. 2. ed. Rio de Janeiro: Qualitymark, 2002.

MACIEL, Ednilson Soares. **Detecção de defeitos em motores de indução pela análise da potência elétrica utilizando a lógica fuzzy**. Dissertação (Mestrado em Engenharia Elétrica). Programa de Pós-Graduação em Engenharia Elétrica, Universidade Federal do Paraná, 2013.

MAIER, R. **Protection of Squirrel-Cage Induction Motor Utilizing Instantaneous Power and Phase Information.** Industry Applications, IEEE Transactions on. 1992, Vol.28, nº 2, pp. 376-380.

MAMDANI, E. H; ASSILIAN, S. **An Experiment in Linguistic Synthesis with a Fuzzy Logic controller.** IEE Transaction. Man-Machine studies, v.7, n.1. p 1-13, 1975.

NOVACKI, A; ALBERTON, C; ORTEGA, L,F. **Proposta de um diagrama de tomada de decisão para a manutenção centrada em confiabilidade, fundamentado na lógica Fuzzy.** UTFPR, 2009.

PARRA, Alejandro PAZ. **Stator fault diagnosis on squirrel cage induction motors by ESA and EPVA.** IEEE Power Electronics and Power Quality Applications (PEPQA), 2013.

REZENDE, Solange Oliveira. **Sistemas Inteligentes – Fundamentos e Aplicações.** São Paulo, Editora Manole, 2003.

RIGONI, Emerson. **Metodologia para implantação da manutenção centrada em confiabilidade: uma abordagem fundamentada em sistemas baseados em conhecimento e logica *Fuzzy.*** Tese apresentada ao programa de pós-graduação em engenharia mecânica da Universidade Federal de Santa Catarina, como requisito parcial para a obtenção do título de Doutor em Engenharia, Florianópolis 2009.

SANTOS, VALDIR APARECIDO. **Manual prático da Manutenção Industrial**. São Paulo, 2007. Ícone. 301 f.

SERENATO, A. L. NETO, P. M. "**DETECÇÃO DE DEFEITOS EM MOTORES DE INDUÇÃO UTILIZANDO A ANÁLISE DAS CORRENTES DE LINHA**". UTFPR, 2014.

SHAW, I. S.; SIMÕES M.G. **Controle e Modelagem Fuzzy**. FAPESP, Editora Edgard Blücher LTDA, São Paulo, 2002.

SILVA, Jonas Guedes Borges da. **Aplicação da análise de componentes principais (PCA) no diagnóstico de defeitos em rolamentos através da assinatura elétrica de motores de indução**. Dissertação (Mestrado em Ciências em Engenharia Elétrica). Programa de Pós-Graduação em Engenharia Elétrica, Universidade Federal de Itajubá, 2008.

WEG. **Manual Geral de Instalação, Operação e Manutenção de Motores Elétricos.** Disponível em: <http://ecatalog.weg.net/files/wegnet/WEG-iom-general-manual-of-electric-motors-manual-general-de-iom-de-motores-electricos-manual-geral-de-iom-de-motores-electricos-50033244-manual-english.pdf>. Acesso em: 30 out. 2017, 18:13.

Zeraoulia,*et all*, **A simple *Fuzzy* logic approach for induction motors stator condition monitoring.** J. Electrical Systems (2005).

ZIDANI, Fatiha; BENBOUZID, M. E. H.; DIALLO, Demba; NAÏT-SAÏD, M. S. **Induction Motor Stator Faults Diagnosis by a Current Concordia Pattern-Based Fuzzy Decision System.** IEEE Transactions on Energy Conversion, vol.18, no. 4, 2003.

APÊNDICES

Apêndice A

Código completo utilizado na IDE do microcontrolador:

```
#include <Fuzzy.h>
#include <FuzzyComposition.h>
#include <FuzzyInput.h>
#include <FuzzyIO.h>
#include <FuzzyOutput.h>
#include <FuzzyRule.h>
#include <FuzzyRuleAntecedent.h>
#include <FuzzyRuleConsequent.h>
#include <FuzzySet.h>

#include <Wire.h>
#include <max6675.h>

#include <EmonLib.h> // Inclusão da biblioteca EmonLib no código

//Pinos usados no Termopar
int thermoDO = 5;
int thermoCS = 6;
int thermoCLK = 7;

//Cria o objeto do Termopar e seta os Pinos Digitais
MAX6675 thermocouple(thermoCLK, thermoCS, thermoDO);

//Seta 5 Votls os pinos Digitais
int vccPin = 3;
```

```
int gndPin = 2;

//Numero de amostras
int amostragem = 3;

EnergyMonitor SensorSCT_A; // Objeto que será utilizado para representar o sensor SCT da fase A
EnergyMonitor SensorSCT_B; // Objeto que será utilizado para representar o sensor SCT da fase B
EnergyMonitor SensorSCT_C; // Objeto que será utilizado para representar o sensor SCT da fase C

//float temper;

int led_verde = 8;
int led_amarelo = 9;
int led_vermelho = 10;

int pinSCT_A = A0; //Pino analógico conectado ao SCT-013 referente a fase A
int pinSCT_B = A1; //Pino analógico conectado ao SCT-013 referente a fase B
int pinSCT_C = A2; //Pino analógico conectado ao SCT-013 referente a fase C

Fuzzy* fuzzy = new Fuzzy();

  FuzzySet* T_baixa = new FuzzySet(-50, -5, 45, 60);          //definindo parâmetros de temperatura
  FuzzySet* T_media = new FuzzySet(50, 60, 100, 105);
  FuzzySet* T_alta = new FuzzySet(100, 110, 115, 120);
  FuzzySet* T_altissima = new FuzzySet(115, 120, 150, 200);

  FuzzySet* Ia_baixa = new FuzzySet(-1, 0, 5.5, 6.3);         //definindo parâmetros de corrente 1
  FuzzySet* Ia_media = new FuzzySet(5.5, 6.7, 9, 9.5);
  FuzzySet* Ia_alta = new FuzzySet(8.8, 10, 10.5, 11);
  FuzzySet* Ia_altissima = new FuzzySet(10.8, 11.2, 50, 100);

  FuzzySet* Ib_baixa = new FuzzySet(-1, 0, 5.5, 6.3);         //definindo parâmetros de corrente 2
  FuzzySet* Ib_media = new FuzzySet(5.5, 6.7, 9, 9.5);
```

```
  FuzzySet* Ib_alta = new FuzzySet(8.8, 10, 10.5, 11);
  FuzzySet* Ib_altissima = new FuzzySet(10.8, 11.2, 50, 100);

  FuzzySet* Ic_baixa = new FuzzySet(-1, 0, 5.5, 6.3);          //definindo parâmetros de corrente 3
  FuzzySet* Ic_media = new FuzzySet(5.5, 6.7, 9, 9.5);
  FuzzySet* Ic_alta = new FuzzySet(8.8, 10, 10.5, 11);
  FuzzySet* Ic_altissima = new FuzzySet(10.8, 11.2, 50, 100);

void setup() {
  // put your setup code here, to run once:
   Serial.begin(9600);     // start serial port

  pinMode(led_verde, OUTPUT);
  pinMode(led_amarelo, OUTPUT);
  pinMode(led_vermelho, OUTPUT);

  SensorSCT_A.current(pinSCT_A, 29);//38); //Chamada da função .current da biblioteca EmonLib, para que seja realizado o cálculo do valor eficaz da corrente IA
  SensorSCT_B.current(pinSCT_B, 31); //Chamada da função .current da biblioteca EmonLib, para que seja realizado o cálculo do valor eficaz da corrente IB
  SensorSCT_C.current(pinSCT_C, 36); //Chamada da função .current da biblioteca EmonLib, para que seja realizado o cálculo do valor eficaz da corrente IC

  pinMode(vccPin, OUTPUT);
  digitalWrite(vccPin, HIGH);
  pinMode(gndPin, OUTPUT);
  digitalWrite(gndPin, LOW);

   // FuzzyInput temperatura
  FuzzyInput* Temp = new FuzzyInput(1);
  Temp->addFuzzySet(T_baixa);
  Temp->addFuzzySet(T_media);
  Temp->addFuzzySet(T_alta);
  Temp->addFuzzySet(T_altissima);
```

```
fuzzy->addFuzzyInput(Temp);

// FuzzyInput corrente 1
FuzzyInput* I1 = new FuzzyInput(2);
I1->addFuzzySet(Ia_baixa);
I1->addFuzzySet(Ia_media);
I1->addFuzzySet(Ia_alta);
I1->addFuzzySet(Ia_altissima);
fuzzy->addFuzzyInput(I1);

// FuzzyInput corrente 2
FuzzyInput* I2 = new FuzzyInput(3);
I2->addFuzzySet(Ib_baixa);
I2->addFuzzySet(Ib_media);
I2->addFuzzySet(Ib_alta);
I2->addFuzzySet(Ib_altissima);
fuzzy->addFuzzyInput(I2);

// FuzzyInput corrente 3
FuzzyInput* I3 = new FuzzyInput(4);
I3->addFuzzySet(Ic_baixa);
I3->addFuzzySet(Ic_media);
I3->addFuzzySet(Ic_alta);
I3->addFuzzySet(Ic_altissima);
fuzzy->addFuzzyInput(I3);


// Fuzzyoutput severidade
FuzzyOutput* S = new FuzzyOutput(1);

FuzzySet* Bom = new FuzzySet(0, 1, 3, 3.8);          //definindo parâmetros de severidade
S->addFuzzySet(Bom);
FuzzySet* Toleravel = new FuzzySet(3.3, 4, 6, 6.8);
```

```
S->addFuzzySet(Toleravel);

FuzzySet* Inaceitavel = new FuzzySet(6.3, 7, 9, 10);

S->addFuzzySet(Inaceitavel);


fuzzy->addFuzzyOutput(S);


// Building FuzzyRule 1

FuzzyRuleAntecedent* TempT_altissimaOrI1Ia_altissima = new FuzzyRuleAntecedent();

TempT_altissimaOrI1Ia_altissima->joinWithOR(T_altissima, Ia_altissima);


FuzzyRuleAntecedent* I2Ib_altissimaOrI3Ic_altissima = new FuzzyRuleAntecedent();

I2Ib_altissimaOrI3Ic_altissima->joinWithOR(Ib_altissima, Ic_altissima);


FuzzyRuleAntecedent* ifTempT_altissimaOrI1Ia_altissimaOrI2Ib_altissimaOrI3Ic_altissima = new FuzzyRuleAntecedent();

ifTempT_altissimaOrI1Ia_altissimaOrI2Ib_altissimaOrI3Ic_altissima->joinWithOR(TempT_altissimaOrI1Ia_altissima, I2Ib_altissimaOrI3Ic_altissima);


FuzzyRuleConsequent* thenSisInaceitavel = new FuzzyRuleConsequent();

thenSisInaceitavel->addOutput(Inaceitavel);


FuzzyRule* fuzzyRule1 = new FuzzyRule(1, ifTempT_altissimaOrI1Ia_altissimaOrI2Ib_altissimaOrI3Ic_altissima, thenSisInaceitavel);

fuzzy->addFuzzyRule(fuzzyRule1);



// Building FuzzyRule 2

FuzzyRuleAntecedent* I1Ia_baixaAndI2Ib_baixa = new FuzzyRuleAntecedent();

I1Ia_baixaAndI2Ib_baixa->joinWithAND(Ia_baixa, Ib_baixa);


FuzzyRuleAntecedent* I3Ic_baixa = new FuzzyRuleAntecedent();

I3Ic_baixa->joinSingle(Ic_baixa);


FuzzyRuleAntecedent* ifI1Ia_baixaAndI2Ib_baixaAndI3Ic_baixa  = new FuzzyRuleAntecedent();
```

```
  ifI1Ia_baixaAndI2Ib_baixaAndI3Ic_baixa->joinWithAND(I1Ia_baixaAndI2Ib_baixa, I3Ic_baixa);


  FuzzyRuleConsequent* thenSisBom = new FuzzyRuleConsequent();
  thenSisBom->addOutput(Bom);


  FuzzyRule* fuzzyRule2 = new FuzzyRule(2, ifI1Ia_baixaAndI2Ib_baixaAndI3Ic_baixa, thenSisBom);
  fuzzy->addFuzzyRule(fuzzyRule2);


  // Building FuzzyRule 3
  //FuzzyRuleAntecedent* I1Ia_baixaAndI2Ib_baixa = new FuzzyRuleAntecedent();   //o antecedente ja
existe
  //I1Ia_baixaAndI2Ib_baixa->joinWithAND(Ia_baixa, Ib_baixa);


  FuzzyRuleAntecedent* I3Ic_media = new FuzzyRuleAntecedent();
  I3Ic_media->joinSingle(Ic_media);


  FuzzyRuleAntecedent* ifI1Ia_baixaAndI2Ib_baixaAndI3Ic_media  = new FuzzyRuleAntecedent();
  ifI1Ia_baixaAndI2Ib_baixaAndI3Ic_media->joinWithAND(I1Ia_baixaAndI2Ib_baixa, I3Ic_media);


  FuzzyRuleConsequent* thenSisToleravel = new FuzzyRuleConsequent();
  thenSisBom->addOutput(Toleravel);


  FuzzyRule*  fuzzyRule3  =  new  FuzzyRule(3,  ifI1Ia_baixaAndI2Ib_baixaAndI3Ic_media,
thenSisToleravel);
  fuzzy->addFuzzyRule(fuzzyRule3);


  // Building FuzzyRule 4
  FuzzyRuleAntecedent* I3Ic_alta = new FuzzyRuleAntecedent();
  I3Ic_alta->joinSingle(Ic_alta);


  FuzzyRuleAntecedent* ifI1Ia_baixaAndI2Ib_baixaAndI3Ic_alta  = new FuzzyRuleAntecedent();
  ifI1Ia_baixaAndI2Ib_baixaAndI3Ic_alta->joinWithAND(I1Ia_baixaAndI2Ib_baixa, I3Ic_alta);
```

```
  //FuzzyRuleConsequent* thenSisInaceitavel = new FuzzyRuleConsequent();     //consequente ja existe

  //thenSisInaceitavel->addOutput(Inaceitavel);


  FuzzyRule* fuzzyRule4 = new FuzzyRule(4, ifI1Ia_baixaAndI2Ib_baixaAndI3Ic_alta, thenSisInaceitavel);

  fuzzy->addFuzzyRule(fuzzyRule4);


  // Building FuzzyRule 5

  FuzzyRuleAntecedent* I1Ia_baixaAndI2Ib_media = new FuzzyRuleAntecedent();

  I1Ia_baixaAndI2Ib_media->joinWithAND(Ia_baixa, Ib_media);


  FuzzyRuleAntecedent* ifI1Ia_baixaAndI2Ib_mediaAndI3Ic_baixa  = new FuzzyRuleAntecedent();

  ifI1Ia_baixaAndI2Ib_mediaAndI3Ic_baixa->joinWithAND(I1Ia_baixaAndI2Ib_media, I3Ic_baixa);


  FuzzyRule* fuzzyRule5 = new FuzzyRule(5, ifI1Ia_baixaAndI2Ib_mediaAndI3Ic_baixa, thenSisToleravel);

  fuzzy->addFuzzyRule(fuzzyRule5);


   // Building FuzzyRule 6

  FuzzyRuleAntecedent* ifI1Ia_baixaAndI2Ib_mediaAndI3Ic_media  = new FuzzyRuleAntecedent();

  ifI1Ia_baixaAndI2Ib_mediaAndI3Ic_media->joinWithAND(I1Ia_baixaAndI2Ib_media, I3Ic_media);


  FuzzyRule* fuzzyRule6 = new FuzzyRule(6, ifI1Ia_baixaAndI2Ib_mediaAndI3Ic_media, thenSisToleravel);

  fuzzy->addFuzzyRule(fuzzyRule6);


  // Building FuzzyRule 7

  FuzzyRuleAntecedent* ifI1Ia_baixaAndI2Ib_mediaAndI3Ic_alta  = new FuzzyRuleAntecedent();

  ifI1Ia_baixaAndI2Ib_mediaAndI3Ic_alta->joinWithAND(I1Ia_baixaAndI2Ib_media, I3Ic_alta);


  FuzzyRule* fuzzyRule7 = new FuzzyRule(7, ifI1Ia_baixaAndI2Ib_mediaAndI3Ic_alta, thenSisInaceitavel);

  fuzzy->addFuzzyRule(fuzzyRule7);
```

```
// Building FuzzyRule 8
  FuzzyRuleAntecedent* I1Ia_baixaAndI2Ib_alta = new FuzzyRuleAntecedent();
  I1Ia_baixaAndI2Ib_alta->joinWithAND(Ia_baixa, Ib_alta);

  FuzzyRuleAntecedent* ifI1Ia_baixaAndI2Ib_altaAndI3Ic_baixa  = new FuzzyRuleAntecedent();
  ifI1Ia_baixaAndI2Ib_altaAndI3Ic_baixa->joinWithAND(I1Ia_baixaAndI2Ib_alta, I3Ic_baixa);

  FuzzyRule* fuzzyRule8 = new FuzzyRule(8, ifI1Ia_baixaAndI2Ib_altaAndI3Ic_baixa,
thenSisInaceitavel);
  fuzzy->addFuzzyRule(fuzzyRule8);

// Building FuzzyRule 9
  FuzzyRuleAntecedent* ifI1Ia_baixaAndI2Ib_altaAndI3Ic_media  = new FuzzyRuleAntecedent();
  ifI1Ia_baixaAndI2Ib_altaAndI3Ic_media->joinWithAND(I1Ia_baixaAndI2Ib_alta, I3Ic_media);

  FuzzyRule* fuzzyRule9 = new FuzzyRule(9, ifI1Ia_baixaAndI2Ib_altaAndI3Ic_media,
thenSisInaceitavel);
  fuzzy->addFuzzyRule(fuzzyRule9);

  // Building FuzzyRule 10
  FuzzyRuleAntecedent* ifI1Ia_baixaAndI2Ib_altaAndI3Ic_alta  = new FuzzyRuleAntecedent();
  ifI1Ia_baixaAndI2Ib_altaAndI3Ic_alta->joinWithAND(I1Ia_baixaAndI2Ib_alta, I3Ic_alta);

  FuzzyRule* fuzzyRule10 = new FuzzyRule(10, ifI1Ia_baixaAndI2Ib_altaAndI3Ic_alta,
thenSisInaceitavel);
  fuzzy->addFuzzyRule(fuzzyRule10);

  // Building FuzzyRule 11
  FuzzyRuleAntecedent* I1Ia_mediaAndI2Ib_baixa = new FuzzyRuleAntecedent();
  I1Ia_mediaAndI2Ib_baixa->joinWithAND(Ia_media, Ib_baixa);

  FuzzyRuleAntecedent* ifI1Ia_mediaAndI2Ib_baixaAndI3Ic_baixa  = new FuzzyRuleAntecedent();
  ifI1Ia_mediaAndI2Ib_baixaAndI3Ic_baixa->joinWithAND(I1Ia_mediaAndI2Ib_baixa, I3Ic_baixa);
```

```
  FuzzyRule* fuzzyRule11 = new FuzzyRule(11, ifI1Ia_mediaAndI2Ib_baixaAndI3Ic_baixa, thenSisToleravel);

  fuzzy->addFuzzyRule(fuzzyRule11);


  // Building FuzzyRule 12

  FuzzyRuleAntecedent* ifI1Ia_mediaAndI2Ib_baixaAndI3Ic_media  = new FuzzyRuleAntecedent();

  ifI1Ia_mediaAndI2Ib_baixaAndI3Ic_media->joinWithAND(I1Ia_mediaAndI2Ib_baixa, I3Ic_media);


  FuzzyRule* fuzzyRule12 = new FuzzyRule(12, ifI1Ia_mediaAndI2Ib_baixaAndI3Ic_media, thenSisToleravel);

  fuzzy->addFuzzyRule(fuzzyRule12);


  // Building FuzzyRule 13

  FuzzyRuleAntecedent* ifI1Ia_mediaAndI2Ib_baixaAndI3Ic_alta  = new FuzzyRuleAntecedent();

  ifI1Ia_mediaAndI2Ib_baixaAndI3Ic_alta->joinWithAND(I1Ia_mediaAndI2Ib_baixa, I3Ic_alta);


  FuzzyRule* fuzzyRule13 = new FuzzyRule(13, ifI1Ia_mediaAndI2Ib_baixaAndI3Ic_alta, thenSisInaceitavel);

  fuzzy->addFuzzyRule(fuzzyRule13);


  // Building FuzzyRule 14

  FuzzyRuleAntecedent* I1Ia_mediaAndI2Ib_media = new FuzzyRuleAntecedent();

  I1Ia_mediaAndI2Ib_media->joinWithAND(Ia_media, Ib_media);


  FuzzyRuleAntecedent* ifI1Ia_mediaAndI2Ib_mediaAndI3Ic_baixa  = new FuzzyRuleAntecedent();

  ifI1Ia_mediaAndI2Ib_mediaAndI3Ic_baixa->joinWithAND(I1Ia_mediaAndI2Ib_media, I3Ic_baixa);


  FuzzyRule* fuzzyRule14 = new FuzzyRule(14, ifI1Ia_mediaAndI2Ib_mediaAndI3Ic_baixa, thenSisToleravel);

  fuzzy->addFuzzyRule(fuzzyRule14);


  // Building FuzzyRule 15

  FuzzyRuleAntecedent* ifI1Ia_mediaAndI2Ib_mediaAndI3Ic_media  = new FuzzyRuleAntecedent();
```

```
ifI1Ia_mediaAndI2Ib_mediaAndI3Ic_media->joinWithAND(I1Ia_mediaAndI2Ib_media, I3Ic_media);


FuzzyRule* fuzzyRule15 = new FuzzyRule(15, ifI1Ia_mediaAndI2Ib_mediaAndI3Ic_media, thenSisBom);

fuzzy->addFuzzyRule(fuzzyRule15);


// Building FuzzyRule 16

FuzzyRuleAntecedent* ifI1Ia_mediaAndI2Ib_mediaAndI3Ic_alta  = new FuzzyRuleAntecedent();

ifI1Ia_mediaAndI2Ib_mediaAndI3Ic_alta->joinWithAND(I1Ia_mediaAndI2Ib_media, I3Ic_alta);


FuzzyRule* fuzzyRule16 = new FuzzyRule(16, ifI1Ia_mediaAndI2Ib_mediaAndI3Ic_alta, thenSisToleravel);

fuzzy->addFuzzyRule(fuzzyRule16);


// Building FuzzyRule 17

FuzzyRuleAntecedent* I1Ia_mediaAndI2Ib_alta = new FuzzyRuleAntecedent();

I1Ia_mediaAndI2Ib_alta->joinWithAND(Ia_media, Ib_alta);


FuzzyRuleAntecedent* ifI1Ia_mediaAndI2Ib_altaAndI3Ic_baixa  = new FuzzyRuleAntecedent();

ifI1Ia_mediaAndI2Ib_altaAndI3Ic_baixa->joinWithAND(I1Ia_mediaAndI2Ib_alta, I3Ic_baixa);


FuzzyRule* fuzzyRule17 = new FuzzyRule(17, ifI1Ia_mediaAndI2Ib_altaAndI3Ic_baixa, thenSisInaceitavel);

fuzzy->addFuzzyRule(fuzzyRule17);


// Building FuzzyRule 18

FuzzyRuleAntecedent* ifI1Ia_mediaAndI2Ib_altaAndI3Ic_media  = new FuzzyRuleAntecedent();

ifI1Ia_mediaAndI2Ib_altaAndI3Ic_media->joinWithAND(I1Ia_mediaAndI2Ib_alta, I3Ic_media);


FuzzyRule* fuzzyRule18 = new FuzzyRule(18, ifI1Ia_mediaAndI2Ib_altaAndI3Ic_media, thenSisToleravel);

fuzzy->addFuzzyRule(fuzzyRule18);


// Building FuzzyRule 19
```

```
FuzzyRuleAntecedent* ifI1Ia_mediaAndI2Ib_altaAndI3Ic_alta  = new FuzzyRuleAntecedent();

ifI1Ia_mediaAndI2Ib_altaAndI3Ic_alta->joinWithAND(I1Ia_mediaAndI2Ib_alta, I3Ic_alta);


FuzzyRule* fuzzyRule19 = new FuzzyRule(19, ifI1Ia_mediaAndI2Ib_altaAndI3Ic_alta,
thenSisToleravel);

fuzzy->addFuzzyRule(fuzzyRule19);


// Building FuzzyRule 20

FuzzyRuleAntecedent* I1Ia_altaAndI2Ib_baixa = new FuzzyRuleAntecedent();

I1Ia_altaAndI2Ib_baixa->joinWithAND(Ia_alta, Ib_baixa);


FuzzyRuleAntecedent* ifI1Ia_altaAndI2Ib_baixaAndI3Ic_baixa  = new FuzzyRuleAntecedent();

ifI1Ia_altaAndI2Ib_baixaAndI3Ic_baixa->joinWithAND(I1Ia_altaAndI2Ib_baixa, I3Ic_baixa);


FuzzyRule* fuzzyRule20 = new FuzzyRule(20, ifI1Ia_altaAndI2Ib_baixaAndI3Ic_baixa,
thenSisInaceitavel);

fuzzy->addFuzzyRule(fuzzyRule20);


// Building FuzzyRule 21

FuzzyRuleAntecedent* ifI1Ia_altaAndI2Ib_baixaAndI3Ic_media  = new FuzzyRuleAntecedent();

ifI1Ia_altaAndI2Ib_baixaAndI3Ic_media->joinWithAND(I1Ia_altaAndI2Ib_baixa, I3Ic_media);


FuzzyRule* fuzzyRule21 = new FuzzyRule(21, ifI1Ia_altaAndI2Ib_baixaAndI3Ic_media,
thenSisInaceitavel);

fuzzy->addFuzzyRule(fuzzyRule21);


// Building FuzzyRule 22

FuzzyRuleAntecedent* ifI1Ia_altaAndI2Ib_baixaAndI3Ic_alta  = new FuzzyRuleAntecedent();

ifI1Ia_altaAndI2Ib_baixaAndI3Ic_alta->joinWithAND(I1Ia_altaAndI2Ib_baixa, I3Ic_alta);


FuzzyRule* fuzzyRule22 = new FuzzyRule(22, ifI1Ia_altaAndI2Ib_baixaAndI3Ic_alta,
thenSisInaceitavel);

fuzzy->addFuzzyRule(fuzzyRule22);
```

```
// Building FuzzyRule 23

FuzzyRuleAntecedent* I1Ia_altaAndI2Ib_media = new FuzzyRuleAntecedent();

I1Ia_altaAndI2Ib_media->joinWithAND(Ia_alta, Ib_media);


FuzzyRuleAntecedent* ifI1Ia_altaAndI2Ib_mediaAndI3Ic_baixa  = new FuzzyRuleAntecedent();

ifI1Ia_altaAndI2Ib_mediaAndI3Ic_baixa->joinWithAND(I1Ia_altaAndI2Ib_media, I3Ic_baixa);


FuzzyRule* fuzzyRule23 = new FuzzyRule(23, ifI1Ia_altaAndI2Ib_mediaAndI3Ic_baixa,
thenSisInaceitavel);

fuzzy->addFuzzyRule(fuzzyRule23);


// Building FuzzyRule 24

FuzzyRuleAntecedent* ifI1Ia_altaAndI2Ib_mediaAndI3Ic_media  = new FuzzyRuleAntecedent();

ifI1Ia_altaAndI2Ib_mediaAndI3Ic_media->joinWithAND(I1Ia_altaAndI2Ib_media, I3Ic_media);


FuzzyRule* fuzzyRule24 = new FuzzyRule(24, ifI1Ia_altaAndI2Ib_mediaAndI3Ic_media,
thenSisToleravel);

fuzzy->addFuzzyRule(fuzzyRule24);


// Building FuzzyRule 25

FuzzyRuleAntecedent* ifI1Ia_altaAndI2Ib_mediaAndI3Ic_alta  = new FuzzyRuleAntecedent();

ifI1Ia_altaAndI2Ib_mediaAndI3Ic_alta->joinWithAND(I1Ia_altaAndI2Ib_media, I3Ic_alta);


FuzzyRule* fuzzyRule25 = new FuzzyRule(25, ifI1Ia_altaAndI2Ib_mediaAndI3Ic_alta,
thenSisToleravel);

fuzzy->addFuzzyRule(fuzzyRule25);


// Building FuzzyRule 26

FuzzyRuleAntecedent* I1Ia_altaAndI2Ib_alta = new FuzzyRuleAntecedent();

I1Ia_altaAndI2Ib_alta->joinWithAND(Ia_alta, Ib_alta);


FuzzyRuleAntecedent* ifI1Ia_altaAndI2Ib_altaAndI3Ic_baixa  = new FuzzyRuleAntecedent();

ifI1Ia_altaAndI2Ib_altaAndI3Ic_baixa->joinWithAND(I1Ia_altaAndI2Ib_alta, I3Ic_baixa);
```

```
  FuzzyRule* fuzzyRule26 = new FuzzyRule(26, ifI1Ia_altaAndI2Ib_altaAndI3Ic_baixa, thenSisInaceitavel);

  fuzzy->addFuzzyRule(fuzzyRule26);


  // Building FuzzyRule 27

  FuzzyRuleAntecedent* ifI1Ia_altaAndI2Ib_altaAndI3Ic_media  = new FuzzyRuleAntecedent();

  ifI1Ia_altaAndI2Ib_altaAndI3Ic_media->joinWithAND(I1Ia_altaAndI2Ib_alta, I3Ic_media);


  FuzzyRule* fuzzyRule27 = new FuzzyRule(27, ifI1Ia_altaAndI2Ib_altaAndI3Ic_media, thenSisToleravel);

  fuzzy->addFuzzyRule(fuzzyRule27);


  // Building FuzzyRule 28

  FuzzyRuleAntecedent* ifI1Ia_altaAndI2Ib_altaAndI3Ic_alta  = new FuzzyRuleAntecedent();

  ifI1Ia_altaAndI2Ib_altaAndI3Ic_alta->joinWithAND(I1Ia_altaAndI2Ib_alta, I3Ic_alta);


  FuzzyRule* fuzzyRule28 = new FuzzyRule(28, ifI1Ia_altaAndI2Ib_altaAndI3Ic_alta, thenSisBom);

  fuzzy->addFuzzyRule(fuzzyRule28);


  // Building FuzzyRule 29

  FuzzyRuleAntecedent* I3Ic_mediaAndTempT_alta = new FuzzyRuleAntecedent();

  I3Ic_mediaAndTempT_alta->joinWithAND(Ic_media, T_alta);


  FuzzyRuleAntecedent* ifI1Ia_baixaAndI2Ib_baixaAndI3Ic_mediaAndTempT_alta = new FuzzyRuleAntecedent();

  ifI1Ia_baixaAndI2Ib_baixaAndI3Ic_mediaAndTempT_alta->joinWithAND(I1Ia_baixaAndI2Ib_baixa, I3Ic_mediaAndTempT_alta);


  FuzzyRule* fuzzyRule29 = new FuzzyRule(29, ifI1Ia_baixaAndI2Ib_baixaAndI3Ic_mediaAndTempT_alta, thenSisInaceitavel);

  fuzzy->addFuzzyRule(fuzzyRule29);


  // Building FuzzyRule 30

  FuzzyRuleAntecedent* I3Ic_baixaAndTempT_alta = new FuzzyRuleAntecedent();

  I3Ic_baixaAndTempT_alta->joinWithAND(Ic_baixa, T_alta);
```

```
  FuzzyRuleAntecedent* ifI1Ia_baixaAndI2Ib_mediaAndI3Ic_baixaAndTempT_alta = new FuzzyRuleAntecedent();

  ifI1Ia_baixaAndI2Ib_mediaAndI3Ic_baixaAndTempT_alta->joinWithAND(I1Ia_baixaAndI2Ib_media, I3Ic_baixaAndTempT_alta);


  FuzzyRule* fuzzyRule30 = new FuzzyRule(30, ifI1Ia_baixaAndI2Ib_mediaAndI3Ic_baixaAndTempT_alta, thenSisInaceitavel);

  fuzzy->addFuzzyRule(fuzzyRule30);


  // Building FuzzyRule 31

  FuzzyRuleAntecedent* ifI1Ia_baixaAndI2Ib_mediaAndI3Ic_mediaAndTempT_alta = new FuzzyRuleAntecedent();

  ifI1Ia_baixaAndI2Ib_mediaAndI3Ic_mediaAndTempT_alta->joinWithAND(I1Ia_baixaAndI2Ib_media, I3Ic_mediaAndTempT_alta);


  FuzzyRule* fuzzyRule31 = new FuzzyRule(31, ifI1Ia_baixaAndI2Ib_mediaAndI3Ic_mediaAndTempT_alta, thenSisInaceitavel);

  fuzzy->addFuzzyRule(fuzzyRule31);


  // Building FuzzyRule 32

  FuzzyRuleAntecedent* ifI1Ia_mediaAndI2Ib_baixaAndI3Ic_baixaAndTempT_alta = new FuzzyRuleAntecedent();

  ifI1Ia_mediaAndI2Ib_baixaAndI3Ic_baixaAndTempT_alta->joinWithAND(I1Ia_mediaAndI2Ib_baixa, I3Ic_baixaAndTempT_alta);


  FuzzyRule* fuzzyRule32 = new FuzzyRule(32, ifI1Ia_mediaAndI2Ib_baixaAndI3Ic_baixaAndTempT_alta, thenSisInaceitavel);

  fuzzy->addFuzzyRule(fuzzyRule32);


  // Building FuzzyRule 33

  FuzzyRuleAntecedent* ifI1Ia_mediaAndI2Ib_baixaAndI3Ic_mediaAndTempT_alta = new FuzzyRuleAntecedent();

  ifI1Ia_mediaAndI2Ib_baixaAndI3Ic_mediaAndTempT_alta->joinWithAND(I1Ia_mediaAndI2Ib_baixa, I3Ic_mediaAndTempT_alta);


  FuzzyRule* fuzzyRule33 = new FuzzyRule(33, ifI1Ia_mediaAndI2Ib_baixaAndI3Ic_mediaAndTempT_alta, thenSisInaceitavel);
```

```
  fuzzy->addFuzzyRule(fuzzyRule33);


// Building FuzzyRule 34

  FuzzyRuleAntecedent* ifI1Ia_mediaAndI2Ib_mediaAndI3Ic_baixaAndTempT_alta = new
FuzzyRuleAntecedent();

  ifI1Ia_mediaAndI2Ib_mediaAndI3Ic_baixaAndTempT_alta->joinWithAND(I1Ia_mediaAndI2Ib_media,
I3Ic_baixaAndTempT_alta);


  FuzzyRule* fuzzyRule34 = new FuzzyRule(34,
ifI1Ia_mediaAndI2Ib_mediaAndI3Ic_baixaAndTempT_alta, thenSisInaceitavel);

  fuzzy->addFuzzyRule(fuzzyRule34);


  // Building FuzzyRule 35

  FuzzyRuleAntecedent* I3Ic_altaAndTempT_alta = new FuzzyRuleAntecedent();

  I3Ic_altaAndTempT_alta->joinWithAND(Ic_alta, T_alta);


  FuzzyRuleAntecedent* ifI1Ia_mediaAndI2Ib_mediaAndI3Ic_altaAndTempT_alta = new
FuzzyRuleAntecedent();

  ifI1Ia_mediaAndI2Ib_mediaAndI3Ic_altaAndTempT_alta->joinWithAND(I1Ia_mediaAndI2Ib_media,
I3Ic_altaAndTempT_alta);


  FuzzyRule* fuzzyRule35 = new FuzzyRule(35,
ifI1Ia_mediaAndI2Ib_mediaAndI3Ic_altaAndTempT_alta, thenSisInaceitavel);

  fuzzy->addFuzzyRule(fuzzyRule35);


  // Building FuzzyRule 36

  FuzzyRuleAntecedent* ifI1Ia_mediaAndI2Ib_altaAndI3Ic_mediaAndTempT_alta = new
FuzzyRuleAntecedent();

  ifI1Ia_mediaAndI2Ib_altaAndI3Ic_mediaAndTempT_alta->joinWithAND(I1Ia_mediaAndI2Ib_alta,
I3Ic_mediaAndTempT_alta);


  FuzzyRule* fuzzyRule36 = new FuzzyRule(36,
ifI1Ia_mediaAndI2Ib_altaAndI3Ic_mediaAndTempT_alta, thenSisInaceitavel);

  fuzzy->addFuzzyRule(fuzzyRule36);


  // Building FuzzyRule 37
```

```
  FuzzyRuleAntecedent* ifI1Ia_mediaAndI2Ib_altaAndI3Ic_altaAndTempT_alta = new FuzzyRuleAntecedent();

  ifI1Ia_mediaAndI2Ib_altaAndI3Ic_altaAndTempT_alta->joinWithAND(I1Ia_mediaAndI2Ib_alta, I3Ic_altaAndTempT_alta);


  FuzzyRule* fuzzyRule37 = new FuzzyRule(37, ifI1Ia_mediaAndI2Ib_altaAndI3Ic_altaAndTempT_alta, thenSisInaceitavel);

  fuzzy->addFuzzyRule(fuzzyRule37);


  // Building FuzzyRule 38

  FuzzyRuleAntecedent* ifI1Ia_altaAndI2Ib_mediaAndI3Ic_mediaAndTempT_alta = new FuzzyRuleAntecedent();

  ifI1Ia_altaAndI2Ib_mediaAndI3Ic_mediaAndTempT_alta->joinWithAND(I1Ia_altaAndI2Ib_media, I3Ic_mediaAndTempT_alta);


  FuzzyRule* fuzzyRule38 = new FuzzyRule(38, ifI1Ia_altaAndI2Ib_mediaAndI3Ic_mediaAndTempT_alta, thenSisInaceitavel);

  fuzzy->addFuzzyRule(fuzzyRule38);


  // Building FuzzyRule 39

  FuzzyRuleAntecedent* ifI1Ia_altaAndI2Ib_mediaAndI3Ic_altaAndTempT_alta = new FuzzyRuleAntecedent();

  ifI1Ia_altaAndI2Ib_mediaAndI3Ic_altaAndTempT_alta->joinWithAND(I1Ia_altaAndI2Ib_media, I3Ic_altaAndTempT_alta);


  FuzzyRule* fuzzyRule39 = new FuzzyRule(39, ifI1Ia_altaAndI2Ib_mediaAndI3Ic_altaAndTempT_alta, thenSisInaceitavel);

  fuzzy->addFuzzyRule(fuzzyRule39);


  // Building FuzzyRule 40

  FuzzyRuleAntecedent* ifI1Ia_altaAndI2Ib_altaAndI3Ic_mediaAndTempT_alta = new FuzzyRuleAntecedent();

  ifI1Ia_altaAndI2Ib_altaAndI3Ic_mediaAndTempT_alta->joinWithAND(I1Ia_altaAndI2Ib_alta, I3Ic_mediaAndTempT_alta);


  FuzzyRule* fuzzyRule40 = new FuzzyRule(40, ifI1Ia_altaAndI2Ib_altaAndI3Ic_mediaAndTempT_alta, thenSisInaceitavel);

  fuzzy->addFuzzyRule(fuzzyRule40);
```

```
}

void loop() {

  //Variavel que recebera os dados de leitura
   float temper = 0;

   //Inicia a leitura das amostras
   for(int index =0; index < amostragem; index++){
     temper = thermocouple.readCelsius() + temper;
     delay(200);
   }

   //Tira a media das leituras
   temper = temper / amostragem;

    float IA_rms = SensorSCT_A.calcIrms(1480);  // Calcula o valor da Corrente Eficaz IA
    float IB_rms = SensorSCT_B.calcIrms(1480);  // Calcula o valor da Corrente Eficaz IB
    float IC_rms = SensorSCT_C.calcIrms(1480);  // Calcula o valor da Corrente Eficaz IC


  fuzzy->setInput(1,temper );                 //Adcionando variaveis de entrada
  fuzzy->setInput(2,IA_rms );
  fuzzy->setInput(3,IB_rms );
  fuzzy->setInput(4,IC_rms );

  fuzzy->fuzzify();                     //mandando para o processo de fuzzyficação

 float output = fuzzy->defuzzify(1);    // puxando a variavel de saída defuzzyficada

    Serial.print("Corrente fase 1 = ");
    Serial.print(IA_rms); // Mostra no monitor serial o valor da corrente IA medida na fase A
```

```
    Serial.println(" A");
    Serial.print("Corrente fase 2 = ");
    Serial.print(IB_rms); // Mostra no monitor serial o valor da corrente IB medida na fase B
    Serial.println(" A");
    Serial.print("Corrente fase 3 = ");
    Serial.print(IC_rms); // Mostra no monitor serial o valor da corrente IC medida na fase C
    Serial.println(" A");

    Serial.print("Temperatura em C = ");
    Serial.println(temper);

    Serial.print("severidade = ");
    Serial.println(output);


if (output <= 4){
  digitalWrite(led_verde, HIGH);    //acende led verde e apaga vermelho e amarelo
  digitalWrite(led_vermelho, LOW);
  digitalWrite(led_amarelo, LOW);
}
 else if  (output >= 6){
  digitalWrite(led_vermelho, HIGH);
  digitalWrite(led_verde, LOW);
  digitalWrite(led_amarelo, LOW);
}
  else{
  digitalWrite(led_amarelo, HIGH);
  digitalWrite(led_verde, LOW);
  digitalWrite(led_vermelho, LOW);
}

  delay(3000);                        //delay de 3s
  }
```