Lucas Goulart Grossi

Solução Móvel para Pré-análise de Motores de Indução

Belo Horizonte

2017

Lucas Goulart Grossi

Solução Móvel para Pré-análise de Motores de Indução

Monografia apresentada durante o Seminário dos Trabalhos de Conclusão do Curso de Graduação em Engenharia Elétrica da UFMG, como parte dos requisitos necessários à obtenção do título de Engenheiro Eletricista.

Universidade Federal de Minas Gerais – UFMG

Escola de Engenharia

Curso de Graduação em Engenharia Elétrica

Orientador: Prof. Sidelmo Magalhães Silva

Belo Horizonte

2017

Resumo

Nos últimos dois séculos foram notáveis os incessantes avanços tecnológicos responsáveis pelos dispositivos eletrônicos dos quais desfrutamos no nosso dia-a-dia. No âmbito de motores elétricos, os principais progressos se deram a partir da invenção dos motores de corrente contínua e alternada. São consideráveis as melhorias no último século, principalmente no que diz respeito à eficiência, controle e acionamento destes motores.

Embora primordiais para os motores funcionarem, como os conhecemos hoje, tais avanços, muitas vezes, não são visíveis ao usuário final. Externamente os motores mantiveram basicamente a mesma estrutura e aparência de cinquenta, sessenta anos atrás, fato que ocasiona dois problemas: a falsa aparência de não melhoria ou avanço dos motores e a não agregação de tecnologias atuais voltadas para interação com o usuário, no âmbito das máquinas elétricas.

O presente trabalho visa agregar conceitos de interação com o usuário, já difundidos em outras áreas de tecnologia, como computadores e celulares, no universo de motores elétricos. Seguindo esse propósito, será projetada e implementada uma aplicação móvel para dispositivos *Android* que visa facilitar o trabalho do engenheiro em campo, fornecendo, a partir dos dados de placa do motor de indução ou de ensaios simples, informações básicas sobre o mesmo, que permitam a realização de uma pré-análise de seu comportamento. Assim, fornece-se ao engenheiro uma visão geral da situação, antes mesmo de sair do local para realizar análises mais complexas.

Palavras-chaves: tecnologia. motores elétricos. motor de indução. aplicação móvel. pré-análise. *android*.

Abstract

In the last two centuries the incessant technological advances responsible for the devices and technologies we have enjoyed in our day-to-day life have been remarkable. In the scope of electric motors the main advances have been made from the invention of the motors of direct and alternating current. These improvements are notable in the last century, especially in terms of efficiency, control and actuation.

Although it is important for engines to work as we know them today, such advances are often not visible to the end user. Externally, the engines have basically retained the same structure and appearance as fifty, sixty years ago, a fact that causes two problems: the false appearance that electrical engines had not been improved over the years and the fact that electrical machines do not aggregate current technologies for iteration with the user.

The present work aims to aggregate concepts of iteration with the user already diffused in other areas of technology, such as computers and cell phones, in the universe of electric motors. Following this purpose, a mobile application for Android devices will be designed and implemented which aims to facilitate the work of the engineer in field by providing from the induction motor's catalog data or simple tests, basic information about the motor that allows a pre-analysis of its behavior. Thus, the engineer has an overview of the situation before even leaving the site to perform more complex analysis.

Key-words: technology. electric motor. induction motor. mobile aplication. android. pre-analysis.

Lista de ilustrações

Lista de abreviaturas e siglas

OS	Operation System
IDE	Integrated Development Environment
API	Application Programming Interface
POO	Programação Orientada a Objetos
UFMG	Universidade Federal de Minas Gerais

Lista de símbolos

η	Rendimento
η_{50}	Rendimento com 50% da potência nominal
η_{75}	Rendimento com 75% da potência nominal
η_{100}	Rendimento com 100% da potência nominal
$\cos(\phi)$	Fator de potência
$\cos(\phi)_{50}$	Fator de potência com 50% da potência nominal
$\cos(\phi)_{75}$	Fator de potência com 75% da potência nominal
$\cos(\phi)_{100}$	Fator de potência com 100% da potência nominal
τ	Torque
τ_n	Torque Nominal
τ_{bl}	Torque com Rotor Bloqueado
$\tau_{\max}$	Torque Máximo
I_n	Corrente Nominal
I_{bl}	Corrente com Rotor BloqueadoNominal
I_0	Corrente a vazio
I_1	Corrente no Estator
I_2	Corrente no Rotor
V_n	Tensão Nominal
V_{bl}	Tensão com Rotor Bloqueado
V_0	Tensão a vazio
V_1	Tensão no Estator
V_2	Tensão no Rotor
V_{th}	Tensão de Thevenin

R_1	Resistência no Estator
R_2	Resistência no Rotor
R_c	Resistência no núcleo
R_{th}	Resistência de Thevenin
X_1	Reatância no Estator
X_2	Reatância no Rotor
X_m	Reatância de magnetização
X_{th}	Reatância de Thevenin
S	Escorregamento
S_{n}	Escorregamento Nominal
ω	Velocidade angular
ω_s	Velocidade angular Síncrona
ω_n	Velocidade angular Nominal
n	Velocidade
n_s	Velocidade Síncrona
n_n	Velocidade Nominal
P	Potência mecânica
P_{in}	Potência de entrada
P_{bl}	Potência com rotor bloqueaedo
P_0	Potência a vazio
p	Número de Polos

Sumário

1 Introdução

1.1 Contexto

Os últimos séculos foram marcados por inúmeros avanços tecnológicos que vão desde a invenção da luz elétrica e das primeiras redes de distribuição, perpassam a revolução microeletrônica da década de 1960 e culminam nos dispositivos que temos hoje como computadores e *smartphones*, sistemas de automação, motores, geradores e outros.

No que diz respeito à engenharia e desenvolvimento de software, podemos destacar alguns pontos fundamentais dentre os inúmeros progressos tecnológicos ocorridos nos últimos anos.

Primeiramente, a revolução microeletrônica (REID, 2007), essenciais para que se desenvolvessem os fundamentos teóricos que nortearam as conquistas em torno dos dispositivos semicondutores do século seguinte. Entre elas estão o Efeito Semicondutor (Michael Faraday - 1833), o Efeito Ratificador (Ferdinand Braum - 1874) e o Efeito Fotovoltaico (Alexandre Becquerel - 1874) (GROUP, 2007) e (BURGESS, 2008). A importância desses acontecimentos e seus desdobramentos está ligada ao fato de a invenção dos transistores tipo CMOS e a sua utilização na produção de circuitos integrados serem a base dos computadores como os conhecemos hoje (MEHL, 2013).

Paralelamente ao surgimento dos computadores eletrônicos, o desenvolvimento das linguagens de programação e o advento da Internet tiveram grande importância. O primeiro possibilitou oferecer maior variedade, versatilidade e especificidade aos computadores, uma vez que as linguagens de programação surgiram para - juntamente com suas IDEs e APIs - agilizar o processo de desenvolvimento de software, permitindo que aplicações para diversas áreas fossem criadas de forma mais ampla e eficiente (GABBRIELLI; MARTINI, 2010). O surgimento da Internet, por sua vez, facilitou a interação entre vários computadores distintos, simultaneamente, ao redor do mundo, o que contribuiu com a difusão da tecnologia e da informação. (ABBATE, 2000)

Um terceiro ponto fundamental foi o grande investimento nos últimos trinta anos destinado à melhoria da interface gráfica. O que antes se resumia a uma tela preta com diversos símbolos e números, hoje, a interface com o usuário é uma área de grande expressão e atenção dentro da computação, parte chave de qualquer software que é desenvolvido nos dias atuais. Tal investimento foi responsável por desmistificar a tecnologia, isto é, trazê-la para o contexto do usuário final, de forma amigável e possibilitando seu uso de maneira fácil, em qualquer contexto.

É nesse contexto que este trabalho se desenvolve, encontrando no cenário atual

inúmeras lacunas na engenharia que se transformaram em oportunidades para o desenvolvimento de aplicativos que contribuem como ferramentas para o trabalho do engenheiro..

1.2 Motivação

No âmbito das máquinas elétricas, não obstante os significativos avanços nos campos de atuação e automação, há, uma carência significativa de avanços voltados para otimização, facilitação e melhoria na interação do usuário final com a máquina - como é o caso, por exemplo, de computadores e celulares que têm buscado ao longo dos anos ser cada vez mas amigáveis ao usuário e de fácil uso. Pouco se vê no campo de máquinas elétricas da tendência tecnológica atual que é produzir aplicações e dispositivos que tornem os processos mais ágeis, práticos e acessíveis.

Há uma grande necessidade de tecnologias atuais voltadas para a interação com o usuário final dentro do universo das máquinas elétricas. Existem soluções voltadas à análise de motores elétricos mas, em sua grande maioria, demandam o uso de computadores e softwares pesados para sua realização, não cobrindo, portanto, a carência de aplicações que sejam práticas, ágeis e de fácil manuseio.

Um exemplo pertinente é a demanda de uma solução que auxilie o engenheiro em campo para realizar a análise de uma máquina elétrica. Seja para identificar um problema de uma máquina já em funcionamento, propor uma melhoria ou a instalação de uma nova máquina, o engenheiro necessita ter informações sobre o funcionamento da mesma.

Neste sentido, motores elétricos já possuem informações de placa e catálogo fornecidos pelo fabricante. Entretanto, essas informações são, em muitos casos, insuficientes para fazer análises simples, como estimativas da variação do torque e eficiência do motor em função da velocidade. Soluções móveis, que, a partir dos dados fornecidos pelo fabricante, fossem capazes de gerar algumas destas análises para o engenheiro, facilitariam e acelerariam o processo de entendimento do funcionamento da máquina. Agilizaria também a detecção de problemas, a tomada de decisão para sua manutenção ou troca e, para o projetista, a determinação de qual modelo mais adequado para se utilizar durante a concepção de um projeto.

Na conjuntura exposta, é extremamente cabível a utilização de tecnologias que possibilitem ao engenheiro realizar, de forma prática e em mãos, essa pré-análise do motor, dando-lhe uma visão geral sobre o mesmo.

1.3 Objetivo

Tendo em vista essa notável lacuna, objetiva-se com esse trabalho trazer conceitos atuais, já bastante difundidos em outros contextos tecnológicos, para o âmbito da enge-

nharia, mais especificamente, dos motores elétricos. Agilidade, portabilidade, facilidade e melhorias na interação do usuário final com os motores são concepções notavelmente escassas ou ausentes nesse cenário e, entende-se como uma melhoria significativa, agregá-las a este contexto.

Com isso em mente, observou-se que o desenvolvimento de uma aplicação móvel com várias funcionalidades, que otimizem o trabalho do engenheiro em campo, seria uma forma interessante de aplicar esses conceitos. Com isso, o processo se tornaria mais ágil e prático, reduzindo custos com deslocamentos e uso de outros dispositivos, além de diminuir o tempo gasto para o entendimento da situação por parte do engenheiro. Uma vez minimizados os esforços de deslocamento, cálculo e quantidade de operações e dispositivos a serem utilizados, há, claramente, uma redução do tempo e custo e um aumento na facilidade de execução da tarefa. Todavia, dada a vastidão da conjuntura de motores elétricos, é incongruente tentar elaborar uma solução que consiga conciliar, de uma só vez, melhorias em todas as áreas envolvidas.

Desta forma, o presente trabalho se limitará a expor de forma clara e objetiva todo o processo de elaboração e implementação de uma aplicação móvel para dispositivos *Android* voltada para motores de indução. Como será visto ao longo do trabalho, essas escolhas se deram pela ampla utilização do motor de indução, pela a abrangência e alcance dos dispositivos *Android* e também por ser congruente com experiências pessoais que já englobavam desenvolvimento *Java* para *Android* e conhecimentos no campo de máquinas de indução. Assim sendo, daqui para frente, tudo que será mencionado sobre motores elétricos será referente exclusivamente ao motor de indução. Há um escopo pré-definido para o projeto, onde contemplaremos a determinação dos parâmetros de circuito equivalente do motor, a utilização desses parâmetros para extrair informações úteis e traçar curvas características do motor de interesse do usuário.

O trabalho contemplará uma revisão teórica acerca de determinação de parâmetros de máquinas de indução e sua utilização na obtenção de informações e gráficos úteis relativos ao motor, uma explicitação sobre a metodologia a ser utilizada - tecnologias a serem usadas e motivo de escolha. Em seguida serão explanados detalhes do projeto da aplicação, minúcias da implementação feita, breve descrição da aplicação final resultante do trabalho e objetivos futuros.

2 Revisão Bibliográfica

Neste capítulo vamos listar os principais conteúdos abordados na bibliografia atual relacionados ao conteúdo desenvolvido no presente trabalho. Primeiramente, entretanto, vale ressaltar que não foram encontradas publicações relevantes que conciliassem aplicações móveis com o contexto de máquinas elétricas, o que, mais uma vez, chama atenção para a carência da área.

As máquinas de indução tem ampla aplicação, sendo utilizadas para diferentes propósitos e em diferentes áreas no mundo inteiro. Dada sua relevância, existem diversos estudos em cima do tema que vão desde fundamentos teóricos consolidados até propostas inovações técnicas e novos conceitos.

(SEN, 1996) e (FITZGERALD, 2003) são dois livros de renome que tratam de máquinas elétricas. No âmbito de motores de indução, fazem uma excelente introdução teórica dos mesmo, seus princípios de funcionamento e principais características. Os avanços envolvendo motores de indução são notáveis e valem ser destacados aqui. Propostas para melhoria de controle de máquinas de indução foram feitas nos últimos anos, como as patentes expostas em (NOLA, 1977) e (HEATH, 1997).

Obras que abrangem melhoria de eficiência de máquinas de indução são outro exemplo de esforços feitos em prol do avanço da área. Em (MOHAN, 1980) temos uma proposta de melhoria de eficiência energética de motores de indução utilizando controle de tensão, enquanto em (NOVOTNY D.J. GRITTER, 1977) é discutida a eficiência de geração da máquina de indução utilizando inversor.

(STERN, 1978) e (JIAN N. L. SCHMITZ, 1983) que propõem o estudo e implementação de otimizações na área de motores de indução. Existem, também, obras que trazem discussões no âmbito de desenvolvimento de softwares que contemplem o universo das máquinas de indução. Em (WENGERKIEVICZ et al., 2017) é feita uma revisão de literatura a partir da implementação de diversos métodos de determinação de parâmetros a partir de dados de catálogo, apresentando e discutindo os resultados encontrados.

(ASSUNçãO J. T., 2006) propõe um algoritmo para simulação do motor de indução trifásico. Nessa obra, é desenvolvida uma metodologia para determinação de parâmetros de motores de indução trifásicos, a partir dos dados de catalogo. O autor faz a implementação da metodologia discutida em *Matlab*, desenvolvendo um *software* de computador para análise de motores de indução que estima os parâmetros da máquina e fornece uma análise do motor a partir deles.

Foi feita pesquisa na loja de aplicativos da *Google*, a *Google Play* [1], a procura de aplicativos que contemplem assuntos similares ao tratado neste trabalho. Não foram encontradas aplicações que realizassem algum tipo de análise ou pré-análise de motores de indução ou máquinas elétricas em geral.

Entretanto, foram encontradas aplicações interessantes que contemplam a área de motores elétricos. (SCIENCEAPPS, 2015) e (PRODUCTION, 2014) propõem o desenvolvimento de um taquímetro digital utilizando a luz de flash da câmera do *smartphone*. O taquímetro é um instrumento muito utilizado para contar o número de giros de uma máquina e, a partir disso, determinar sua velocidade de rotação

Na parte de desenvolvimento de *software*, (ARNOLD JAMES GOSLING, 2005) faz uma excelente introdução e traz uma base muito sólida relacionada á linguagem de programação *Java*. O paradigma de POO é introduzido e detalhado em (WEGNER, 1990), enquanto a forma que o *Java* lida e incorpora os conceitos desse modelo de programação é detalhada em (ORACLE, 2010). A base para o desenvolvimento voltado para *Android* é fornecido pela própria *Google*, responsável por seu desenvolvimento e manutenção. (GOOGLE, 2014) é um guia introdutório à programação *Android* que fornece os principais conceitos, exemplos e exercícios para fixação de aprendizado.

Pode-se concluir que a literatura atual, ao mesmo tempo que é rica e ampla, trazendo avanços em diversas áreas e campos envolvendo motores de indução, é, também, carente de obras que primem por soluções tecnológicas modernas, envolvendo conceitos atuais para auxiliar e facilitar a interação com o usuário final.

Além disso, é possível perceber que o outro lado da literatura, isto é, no âmbito da computação e do desenvolvimento móvel, também há muita riqueza, com amplo embasamento teórico e guias práticos que visam o aprendizado das tecnologias atuais de desenvolvimento. Com isso, pode-se concluir que não há uma carência de literatura ou de informação de uma das áreas. Trata-se, sim, de uma carência de iniciativas que juntem os dois conhecimentos, no intuito de propor soluções modernas e relevantes.

[1] https://play.google.com/

3 Fundamentação Teórica

O presente trabalho apresenta-se como uma obra interdisciplinar, aplicando conceitos da computação no campo de motores elétricos visando melhorias tecnológicas no âmbito do usuário final. Para tal, há a utilização de concepções e ferramentas das duas esferas de conhecimento, afim de apresentar uma solução prática para aperfeiçoar a interação usuário, máquina e simplificar o trabalho do engenheiro em campo.

Tendo isso em vista, é necessário esclarecer parte da teoria de máquina de indução, para que se tenha completo entendimento do que é tratado, proposto e executado neste trabalho. Este capítulo se dedica a fazer uma introdução teórica da máquina de indução baseando-se em conceitos bastante conhecidos no universo de motores de indução. É necessário que estes princípios sejam compreendidos, e, por já serem bastante consolidados, existem dois livros que são referência no assunto e dos quais essas concepções foram transcritas afim de introduzir o assunto teórico.

Cientes disso, será recordado aqui conceitos fundamentais da teoria de (SEN, 1996) e (FITZGERALD, 2003), que norteiam o entendimento de motores elétricos, como princípios de funcionamento, determinação de equivalência por circuitos e extração de características importantes dos motores a partir disso. Serão adicionados, também, a teoria dos livros aqui citados, conhecimentos prévios provindos de disciplinas do curso de engenharia elétrica, que abordam em detalhes o funcionamento de motores de indução, como é o caso da disciplina Laboratório de Conversão.

Além disso, serão repassados princípios da área de computação cujo entendimento é imprescindível para compreender detalhes do desenvolvimento da aplicação.

3.1 A máquina de indução

Motores de indução são construídos de forma que uma de suas principais características é a produção de conjugado a partir da interação entre dois campos magnéticos girantes, característica essa que será detalhada adiante. No motor de indução, corrente elétrica alternada trifásica é fornecida diretamente ao estator.

As correntes estabelecidas no estator são responsáveis por, através de indução eletromagnética - como o nome sugere - estabelecer a corrente elétrica no rotor. Na máquina de indução não há linearidade entre o valor dos campos magnéticos (de rotor e estator) e o conjugado gerado. Como consequência disto, o controle de conjugado e velocidade de um motor de indução não é imediato e linear.

Um importante parâmetro para a operação de um motor de indução é o escor-

regamento, que indica a diferença entre a velocidade síncrona e a velocidade com que o rotor efetivamente gira. A maneira de se obter o valor do escorregamento para um valor momentâneo de velocidade é expresso na equação 3.1.

$$S = \frac{n_s - n}{n_s} \tag{3.1}$$

Como visto na relação 3.1, para calcular o escorregamento é necessário obter a velocidade síncrona do motor, que pode ser facilmente deduzida a partir da equação 3.2.

$$n_s = \frac{120 * f}{p} \tag{3.2}$$

Uma vez deduzida a velocidade síncrona do motor pode-se obter a velocidade angular síncrona do motor pela relação 3.3.

$$\omega = \frac{2 * \pi * n}{60} \tag{3.3}$$

O rotor de um motor de indução constitui-se de uma superposição de discos finos, feitos de material ferromagnético, que são "empilhados" horizontalmente. No cilindro ferromagnético assim formado, ranhuras longitudinais em sua estrutura são preenchidas com alumínio ou cobre liquefeito, que ao se enrijecer forma um conjunto de barras condutoras que dão origem à estrutura referida como gaiola de esquilo, como ilustrado na figura 1.

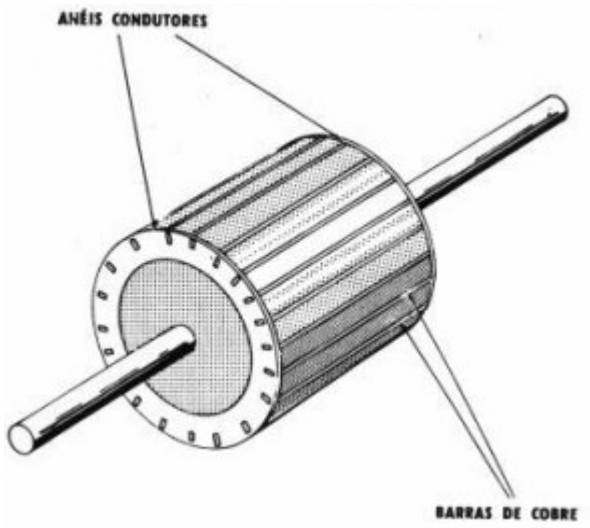

Figura 1 – Rotor do tipo gaiola de esquilo.

Nas duas extremidades do cilindro são adicionados anéis condutores, que unem física e eletricamente as barras condutoras formadas. Essa construção da máquina de indução resulta em uma estrutura construtivamente bastante simples e robusta, dispensando a manutenção frequente e de custo de fabricação consideravelmente baixo, se comparado

a outras máquinas. Esta série de vantagens associadas à máquina de indução faz com que o maior número de motores empregados atualmente seja deste tipo, um dos motivos da escolha do motor de indução como objeto de estudo deste trabalho.

3.1.1 Princípio de Funcionamento

A operação da máquina de indução para a geração de conjugado envolve a interação de dois campos girantes no espaço e no tempo. Um destes campos é gerado pelas correntes estabelecidas nos enrolamentos de estator, enquanto o outro é gerado pelas correntes induzidas no rotor. A disposição espacial do enrolamento trifásico de estator, como será descrito, faz com que seu campo magnético tenha intensidade constante, sendo apenas sua orientação variante no tempo, de tal forma que seu eixo de direção descreve um movimento circular.

Iniciando a descrição do processo para o estabelecimento deste campo girante, a primeira imagem abaixo mostra a orientação das linhas do fluxo magnético gerado pelas correntes no enrolamento de estator no caso em que este é alimentado apenas por um circuito monofásico. Uma mesma corrente elétrica segue longitudinalmente pela estrutura cilíndrica do estator, retornando pelo "lado" oposto. Ou seja, uma fase consiste em duas correntes defasadas de 180 graus no tempo e no espaço descrito pela circunferência do estator, como ilustrado na figura 2.

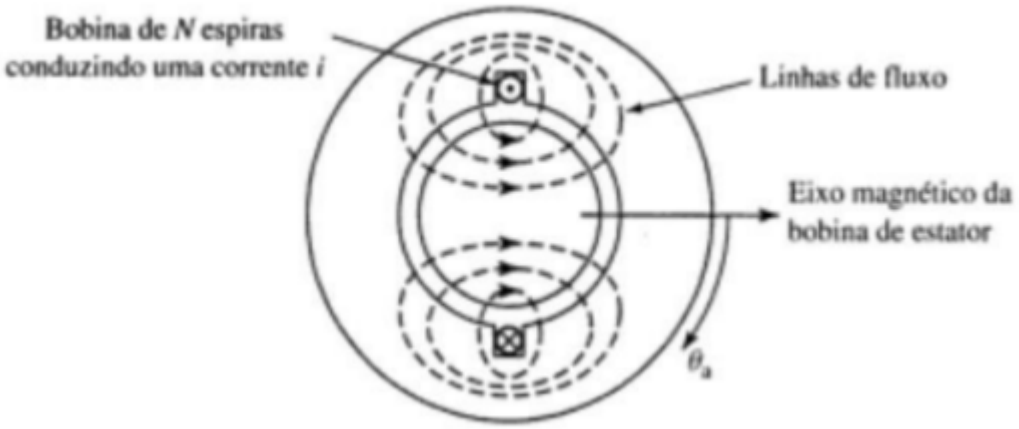

Figura 2 – Linhas de fluxo magnético diante da alimentação de estator por circuito monofásico.

O campo magnético gerado nesta situação ilustrativa tem componentes apenas em uma direção, neste caso na direção horizontal. Sendo a corrente que o origina alternada e senoidal, o sentido deste campo também o é. O valor do campo pode ser determinado a partir da equação 3.4

$$B_x(t) = B_{max} * sen(\omega_{ele} * t) \tag{3.4}$$

Este tipo de campo magnético é referido como “campo pulsante”, pois oscila no tempo apenas em uma direção. Um campo girante pode ser obtido da superposição de campos pulsantes – daí a necessidade de se utilizar um enrolamento trifásico. Estabelecendo-se três enrolamentos semelhantes àquele exemplificado na imagem acima, que conduzem correntes elétricas de mesma intensidade e de mesma frequência, defasadas de 120 graus no tempo, e ainda separados também por 120 graus no espaço da circunferência, o campo resultante da superposição é um campo girante. A distribuição espacial do enrolamento de cada fase do circuito trifásico para esta situação é mostrada na figura 3.

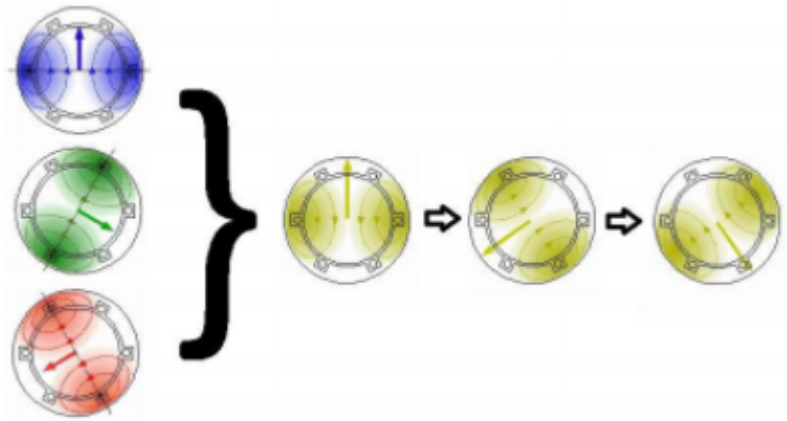

Figura 3 – Distribuição espacial de cada fase do circuito de alimentação trifásico para geração de campo girante.

O campo resultante de estator, por sua vez, induz correntes no rotor, que produzem seu próprio campo girante. A interação entre o campo resultante de estator e o campo do rotor resulta na geração de conjugado. A frequência angular com que o campo resultante de estator gira depende do número de polos da máquina de indução, como explicitado na equação 3.5

$$\omega_{campo} = \frac{2 * \omega_{ele}}{p} \tag{3.5}$$

O efeito do aumento do número de polos é a redução da velocidade do campo girante e, consequentemente, da velocidade síncrona do motor. Para que seja aplicado um número de polos maior, o enrolamento de cada fase da alimentação trifásica deve sofrer aumento no número de voltas que descreve longitudinalmente na estrutura cilíndrica do motor.

3.1.2 Circuito Equivalente

A corrente elétrica no rotor do motor de indução, como sugere o nome, é estabelecida por meio de indução eletromagnética por meio das corretes estabelecidas no estator. Por tal motivo, há uma analogia entre o motor de indução e o transformador. O modelo de

circuito do motor, semelhante ao de um transformador, está ilustrado na figura 4, sendo equivalente a uma das três fases da alimentação trifásica do motor. O modelo envolve as reatâncias e resistências dos enrolamentos, o escorregamento e a magnetização do núcleo ferromagnético.

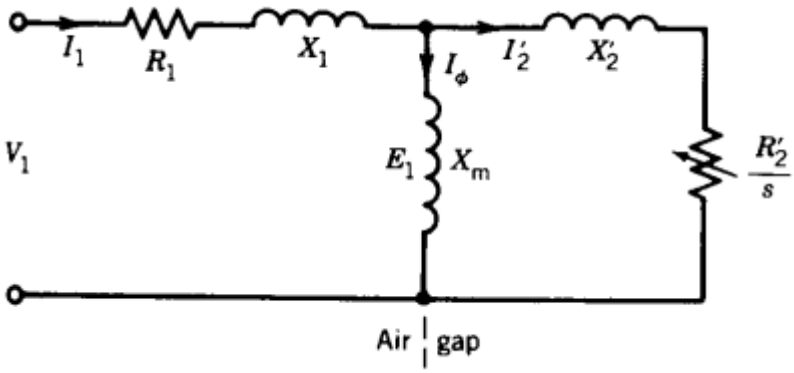

Figura 4 – Modelo de circuito para o motor de indução.

Existem modelos que apresentam ainda uma resistência em paralelo com a indutância de magnetização, representando as perdas joulicas que ocorrem no núcleo ferromagnético em função de correntes induzidas de Focault e em função da histerese. No modelo que será utilizado neste estudo, esta resistência será desprezada por seu valor ser, normalmente, muito pequeno.

A resistência dinâmica do rotor dada por R_2/s pode ser compreendida como a soma de duas resistências especiais, sendo uma a representação da conversão de energia elétrica em conjugado mecânico e a outra a representação de perdas ohmicas no rotor, como demonstrado na equação 3.6.

$$\frac{R_2}{s} = \frac{R_2}{s} * (1 - s) + R_2 \tag{3.6}$$

Dessa forma, podemos calcular a potência mecânica obtida da conversão eletromecânica pela expressão 3.7.

$$P = 3 * |I_2|^2 * \frac{R_2}{s} * (1 - s) \tag{3.7}$$

A multiplicação por 3 no lado direito da equação significa a união das três fases do motor trifásico na geração de conjugado, lembrando que o modelo de circuito equivale a apenas uma das três fases. Substituindo a equação 3.1 e a expressão para o torque mecânico dada em 3.8, chegamos à relação 3.9 para o torque mecânico em função da corrente e resistência de estator e velocidade angular nominal.

$$\tau = \frac{P}{\omega} \tag{3.8}$$

$$\tau = 3 * |I_2|^2 * \frac{R_2}{s * \omega_s} \tag{3.9}$$

Entretanto, dada a complexidade da obtenção do valor de I_2 por esse modelo é muito usual a utilização do equivalente de Thevenin do circuito de estator. O modelo com o equivalente de Thevenin pode ser visto na figura 5. As grandezas do equivalente de Thevenin V_{th}, R_{th} e X_{th} podem ser obtidas pelas equações 3.10, 3.11 e 3.12, respectivamente, sendo K_{th} representada pela equação 3.13.

$$V_{th} = K_{th} * V1 \tag{3.10}$$

$$R_{th} = K_{th}^2 * R_1 \tag{3.11}$$

$$X_{th} \approx X_1 \tag{3.12}$$

$$K_{th} = \frac{X_m}{X_1 + X_m} \tag{3.13}$$

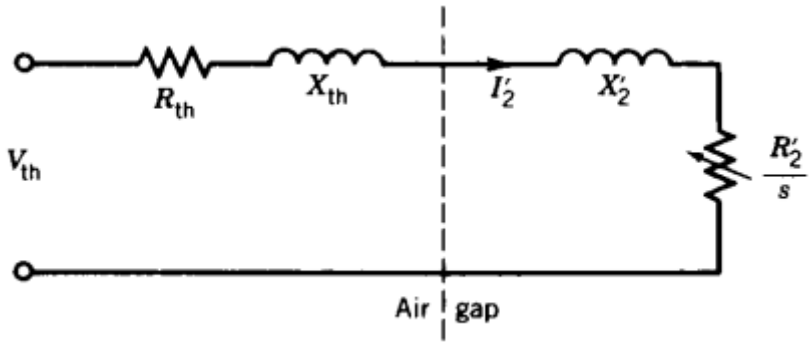

Figura 5 – Modelo de circuito para o motor de indução, com equivalente de Thevenin do circuito de estator.

A partir do equivalente de Thevenin pode-se agora calcular de forma mais fácil o valor de I_2. Pelo circuito equivalente resultante, podemos calcular I_2 tal qual mostrado na equação 3.14.

$$I_2 = \frac{|V_{th}|}{|\frac{R_2}{s} + R_{th} + j(X_{th} + X_2)|} = \frac{|V_{th}|}{\sqrt{(\frac{R_2}{s} + R_{th})^2 + (X_{th} + X_2)^2}} \tag{3.14}$$

Voltando à equação 3.9, substituindo o valor encontrado para I_2 na equação 3.14 a partir do circuito equivalente de Thevenin, podemos calcular o torque mecânico em função dos elementos do circuito equivalente, como mostrado na equação 3.15.

$$\tau = \frac{3}{\omega_s} * \frac{|V_{th}|^2}{(\frac{R_2}{s} + R_{th})^2 + (X_{th} + X_2)^2} * \frac{R_2}{s} \tag{3.15}$$

3.1.2.1 Determinação dos Parâmetros de Circuito Equivalente

Na seção 3.1.2 nos atemos à teoria que envolve o circuito equivalente que representa um motor de indução, sua variante equivalente de Thevenin, bem como informações que podemos aferir a partir destes circuitos. Nesta seção, vamos discutir formas de determinação dos parâmetros do circuito equivalente, as aproximações necessárias a tal e suas implicações. A importância e utilização do circuito equivalente para analisar certas características do motor serão melhores detalhados na seção 3.1.3.

3.1.2.1.1 Determinação a partir dos Ensaios

Os parâmetros R_1, X_1, R_2, X_2 e X_m, referentes ao circuito equivalente detalhado na seção anterior, podem ser determinados a partir da combinação dos resultados de três ensaios: a Vazio, a Rotor Bloqueado e medição da Resistência de Estator. Os três casos e a utilização dos mesmos para determinação dos parâmetros serão detalhados neste tópico.

3.1.2.1.2 Ensaio a Vazio

O ensaio a vazio na máquina de indução, assim como o ensaio de circuito aberto em um transformador, fornece informações sobre corrente de excitação e perdas rotacionais. Consiste em aplicar tensão nominal no circuito de estator estando a máquina desacoplada de toda e qualquer carga mecânica. A perda de potência constatada pode ser atribuída à perdas no núcleo e por fricção.

Medindo-se I_0, P_0 e V_0, pode-se calcular os valores de R_0 e X_0, como demonstrado nas equações 3.16 e 3.17.

$$R_0 = \frac{P_0}{3 * I_0^2} \tag{3.16}$$

$$X_0 = \sqrt{\frac{V_0^2}{3 * I_0^2} - (R_0)^2} \tag{3.17}$$

3.1.2.1.3 Ensaio a Rotor Bloqueado

O ensaio de rotor bloqueado na máquina de indução, assim como o ensaio de curto-circuito em um transformador, fornece informações sobre impedâncias de fuga. Bloqueia-se o rotor de forma a não permitir que o motor gire e é aplicada tensão ao circuito de estator. O teste de rotor bloqueado ocorre em tensão reduzida e corrente nominal para garantir

que o teste aconteça sob as mesmas condições de frequência e corrente no rotor que vão prevalecer durante sua operação em condições.

É aconselhável, também, aplicar frequência reduzida, uma vez que a resistência efetiva do rotor e a indutância de fuga em frequência reduzida - correspondente a baixos valores de escorregamento - terão diferenças significativas daqueles em frequência nominal. Sugere-se uma frequência de 25% da frequência nominal. Para motores com potência abaixo de 20-hp, entretanto, os efeitos são desprezíveis e o teste pode ocorrer em frequência nominal.

Medindo-se I_{bl}, P_{bl} e V_{bl}, pode-se calcular os valores de R_{bl} e X_{bl}, como demonstrado nas equações 3.18 e 3.19.

$$R_{bl} = \frac{P_{bl}}{3 * I_{bl}^2} \tag{3.18}$$

$$X_{bl} = \sqrt{\frac{V_{bl}^2}{3 * I_{bl}^2} - (R_{bl})^2} \tag{3.19}$$

3.1.2.1.4 Ensaio para medição da Resistência de Estator

A medição da resistência de estator consiste em aplicar uma corrente I entre dois terminais do motor e medir a tensão V necessária para estabelecer esse nível de corrente. Dividindo-se tensão por corrente encontramos uma resistência duas vezes maior do que a resistência de estator, considerando conexão em estrela entre os terminais. Assim, pode-se calcular R_1 a partir da equação 3.20.

$$R_1 = \frac{V}{2 * I} \tag{3.20}$$

Com os dados dos três ensaios em mãos, podemos determinar os parâmetros do circuito equivalente remanescentes. R_1 já foi determinado a partir do terceiro ensaio, como mostrado em 3.20. A utilização das medições dos ensaios para determinação R_2, X_1 e X_m é ilustrada nas equações 3.22, 3.26 e 3.28, respectivamente. É válido ressaltar que a igualdade 3.24 foi utilizada para facilitar a determinação dos parâmetros, uma vez que essa é uma relação verdadeira para boa parte dos motores de indução existentes.

$$R_{bl} = R_1 + R_2 \tag{3.21}$$

$$R_2 = R_{bl} - R_1 \tag{3.22}$$

$$X_{bl} = X_1 + X_2 \tag{3.23}$$

$$X_1 = X_2 \tag{3.24}$$
$$X_{bl} = 2X_1 \tag{3.25}$$
$$X_1 = X_2 = \frac{X_{bl}}{2} \tag{3.26}$$

$$X_0 = X_1 + X_m \tag{3.27}$$
$$X_m = X_0 - X_1 \tag{3.28}$$

3.1.2.2 Determinação a partir dos Dados do Fabricante

Em campo, muitas vezes o engenheiro não tem a possibilidade de realizar ensaios na máquina com a qual está trabalhando. Isso exige formas alternativas de determinar o circuito equivalente que utilizem dados que ele já possui, como os dados fornecidos pelo fabricante.

Tendo o catálogo do fabricante é possível estimar de forma aceitável os parâmetros R_1, R_2, X_1, X_2 e X_m do circuito equivalente do motor de indução. Nossa metodologia começa desconsiderando o valor de R_1, isto é, $R_1 = 0$. Esta é, na verdade, uma aproximação relativamente grosseira. Porém, se justifica pelo fato de, R_1, ter valor consideravelmente baixo se comparado aos outros parâmetros e estar mais diretamente relacionado ao torque de partida do motor. Com isso, podemos considerar praticamente nula a influência direta de R_1 na faixa de torque próximo ao nominal, aquela de maior interesse nas análises que desejamos fazer (mais detalhes na sessão 3.1.3).

Uma vez desprezado o valor de R_1, se voltarmos à equação 3.13, temos que $K_{th} = 1$ para valores de X_m muito maiores que X_1. Uma vez que $K_{th} = 1$, podemos aproximar V_{th} de V_1. Todas essas considerações nos levam às expressões 3.29, 3.30 e 3.31.

$$K_{th} = \frac{X_m}{(X_m + X_1)} \approx 1, se X_m >> X_1 \tag{3.29}$$
$$V_{th} = K_{th} * V_1 = V_1 \tag{3.30}$$
$$R_{th} = K_{th}^2 * R_1 = R_1 = 0 \tag{3.31}$$

Sabe-se que o torque máximo do motor de indução pode ser calculado a partir da expressão 3.32. Uma vez que $R_1 = 0$ e considerando as expressões 3.30 e 3.31, a expressão para o torque máximo pode ser simplificada como em 3.33

$$\tau_{max} = \frac{3}{2 * \omega_s} * \frac{V_{th}^2}{R_{th} + \sqrt{R_{th}^2 + (X_{th} + X_2)^2}} \tag{3.32}$$

$$\tau_{max} = \frac{3}{2 * \omega_s} * \frac{V_1^2}{\sqrt{(X_1 + X_2)^2}} \tag{3.33}$$

Mais uma aproximação comum, que será aplicada nesse trabalho, é descrita em 3.34. Isso posto que esta relação entre X_1 e X_2 ocorre em boa parte dos motores de indução existente e, portanto, assume-se aqui que ela é verdadeira.

$$X_1 = X_2 \tag{3.34}$$

Por conseguinte, temos a equação 3.35 para o torque máximo que depende somente do parâmetro X_1. Uma vez que as demais variáveis presentes na equação 3.35 são conhecidos, podemos isolar X_1 para determiná-lo, conforme a equação 3.36 demonstra.

$$\tau_{max} = \frac{3}{2 * \omega_s} * \frac{V_1^2}{2 * X_1} \tag{3.35}$$

$$X_1 = X_2 = \frac{3}{4 * \omega_s} * \frac{V_1^2}{\tau max} \tag{3.36}$$

A resistência de rotor é o parâmetro de rotor e estator que nos resta determinar. Pode-se determinar R_2 a partir do torque nominal, cuja expressão pode ser deduzida a partir da equação 3.15. Para isso, é necessário assumir-se que, em velocidade nominal, o escorregamento é quase zero, ou ainda, considerar as condições expostas na equação 3.37. A partir disso, chega-se à expressão 3.38 para o torque nominal da qual podemos isolar R_2 para determinar seu valor, como feito na equação 3.39.

$$\frac{R_2}{s} >> R_{th}, \frac{R_2}{s} >> X_{th} + X_2 \tag{3.37}$$

$$\tau_N = \frac{3}{\omega_s} * \frac{V_{th}^2}{R_2} * S_n \tag{3.38}$$

$$R_2 = \frac{3}{\omega_s} * \frac{V_{th}^2}{\tau_N} * S_n \tag{3.39}$$

Com isso, conseguimos inferir os parâmetros do circuito equivalente correspondentes ao estator e ao rotor. Precisamos agora determinar a reatância de magnetização, X_m. Aplicando-se a equação 3.9 para o torque nominal, dado fornecido pelo fabricante, podemos isolar I_2 tal qual mostrado na equação 3.40.

$$I_2 = \sqrt{\frac{\tau_n * \omega_s}{3} * \frac{S_n}{R_2}} \tag{3.40}$$

Como estamos analisando o torque nominal, podemos determinar módulo e fase de I_1 a partir da corrente nominal e do fator de potência nominal, respectivamente, como demonstrado em 3.41.

$$I_1 = |I_n| \underline{/\arccos(\cos(\phi)_{100}))} \tag{3.41}$$

Podemos considerar que I_1 é a soma da corrente no rotor I_2 com a corrente de magnetização I_m e isolar I_m para achar seu módulo, como na equação 3.42.

$$|I_m| = |I_2 - I_1| = |R + jI_m| = \sqrt{R^2 + I_m^2} \tag{3.42}$$

Por fim, podemos aplicar uma equação básica de circuitos para deduzir o valor de X_M, como mostrado em 3.43.

$$X_m = \frac{V_1}{|I_m|} \tag{3.43}$$

Uma prática recorrente que pode melhorar a aproximação dos valores de circuito equivalente é recalcular os parâmetros a partir dos dados aferidos. Isto é, calcular o novo K_{th} para os valores de X_m e X_1 encontrados. Esse processo recursivo pode ser realizado de duas a três vezes com intuito de se obter aproximações ainda melhores para os parâmetros.

Tem-se então duas formas de determinar os cinco parâmetros que compõe o circuito equivalente do motor de indução abordado nesse trabalho: a partir da realização de ensaios e a partir dos dados fornecidos pelo fabricante. A seção seguinte tratará da utilização desses parâmetros para apontar características importantes do funcionamento do motor.

3.1.3 Determinação de Características do Motor de Indução a partir de seu Circuito Equivalente

O circuito equivalente, tratado nas seções 3.1.2 e 3.1.2.1, pode ser utilizado para prever algumas características de performance do motor de indução. As principais características que podemos prever a partir do circuito equivalente estimado anteriormente e que serão detalhadas nessa seção são: Torque, Corrente de Estator, Eficiência e Fator de Potência, todas elas em função da variação da velocidade do motor.

3.1.3.1 Torque

Uma vez calculados os parâmetros de circuito equivalente e de Thevenin, podemos assumir que estes são constantes. Dessa forma, pela equação 3.15 temos o torque variando exclusivamente em função do escorregamento e, portanto, da velocidade do motor, uma

vez que a velocidade síncrona também é constante. Com isso, podemos traçar o perfil característico do torque do motor de indução em função de sua velocidade.

A figura 6 ilustra a curva característica do motor de indução em função da sua velocidade (n) para diferentes valores de V_1. É importante ressaltar que esse é um esboço da curva de torque e que ela varia, de amplitude e deslocando-se para esquerda ou direita, dependendo do seu circuito equivalente.

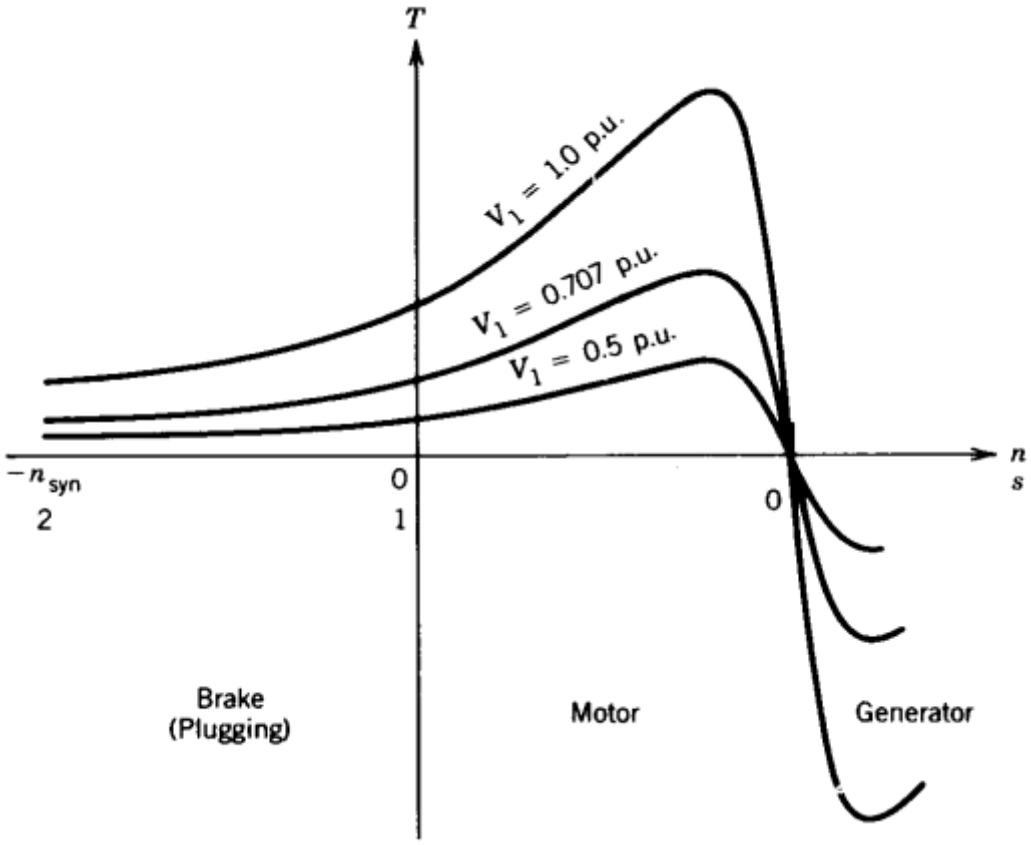

Figura 6 – Curva característica do torque em função da velocidade do motor de indução para diferentes valores de V_1.

3.1.3.2 Corrente no Estator

Com base na figura 4, citada no item 3.3 deste trabalho, podemos estimar a impedância vista pelo estator, isto é, a impedância de entrada do circuito, como mostrado na equação 3.46. A partir da impedância de entrada podemos calcular a corrente no estator, como mostrado em 3.47.

$$Z_1 = R_1 + jX_1 + X_m//(\frac{R_2}{s} + jX_2) \tag{3.44}$$

$$Z_1 = R_1 + jX_1 + \frac{jX_m * (\frac{R_2}{s} + jX_2)}{\frac{R_2}{s} + j(X_m + X_2)} \tag{3.45}$$

$$Z_1 = |Z_1|\underline{/\theta_1} \tag{3.46}$$

$$I_1 = \frac{V_1}{Z_1} \tag{3.47}$$

Novamente, temos uma característica do motor que depende somente de constantes relativas a seu circuito equivalente e seu escorregamento. Assim, é possível prever a curva característica do módulo da corrente de estator em função da velocidade do motor, como é ilustrado na figura 7.

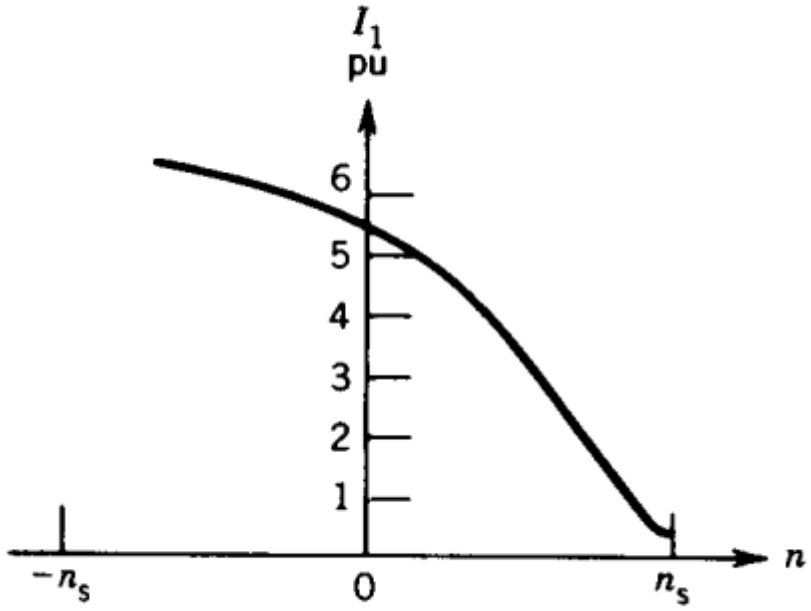

Figura 7 – Curva característica da corrente de estator em função da velocidade do motor de indução.

3.1.3.3 Fator de Potência

Uma vez determinadas a impedância de entrada equivalente e a corrente de estator, o fator de potência pode ser facilmente calculado utilizando o ângulo de fase da corrente de estator que, para o circuito equivalente estudado, é o mesmo ângulo da impedância de entrada. O cálculo do fator de potência a partir do ângulo de fase é mostrado em 3.48 e sua curva característica em função da velocidade do motor é ilustrada na figura 8.

$$FP = \cos\theta_1 \tag{3.48}$$

3.1.3.4 Eficiência

A última característica associada ao circuito equivalente do motor de indução que será avaliada nesse trabalho é a eficiência do motor. A eficiência pode ser definida como

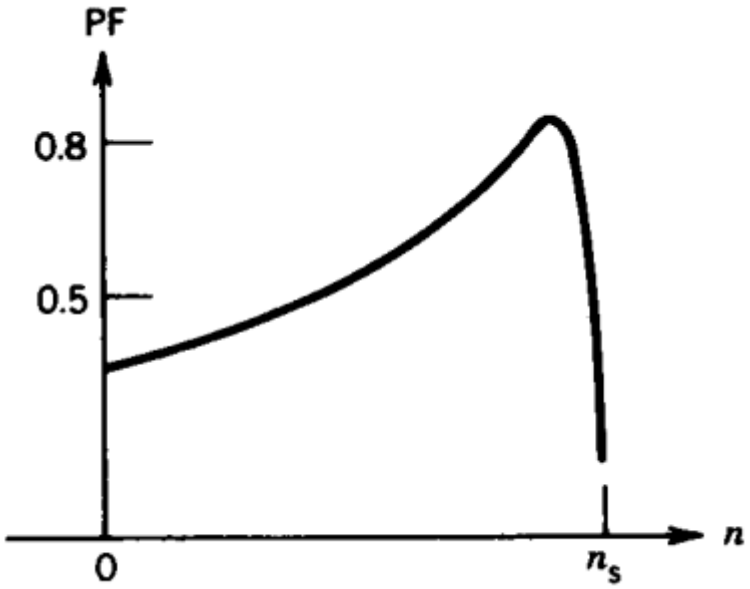

Figura 8 – Curva característica do fator de potência em função da velocidade do motor de indução.

a porcentagem da potência de entrada entregue ao rotor após as perdas ôhimcas e de histerese.

Para o cálculo da eficiência precisamos, primeiramente, calcular as potências de entrada e saída do motor. Na fundamentação deste trabalho foi explicado que a resistência R_2/s é dividida em duas partes, uma associada à perdas ôhmicas e outra associada à potência mecânica do motor. A potência mecânica corresponde a potência de saída e é definida em 3.7.

A potência de entrada, por sua vez, pode ser calculada a partir de V_1, I_1 e o ângulo de fase entre eles, como é mostrado na equação 3.49.

$$P_{in} = 3 * |V_1| * |I_1| * cos(\theta_1) \tag{3.49}$$

Por fim, a eficiência pode ser calculada pela razão entre a potência de saída e a potência de entrada, como mostrado na equação 3.50.

$$\eta = \frac{P_{out}}{P_{in}} = \frac{P}{P_{in}} \tag{3.50}$$

Novamente, podemos esboçar sua curva característica em função da velocidade, como ilustrado na figura 9.

Concluímos a fundamentação teórica do presente trabalho, após esmiuçar os principais conceitos do motor de indução, seus princípios de funcionamento, seu circuito equivalente e maneiras apropriadas de aproximar seus parâmetros, bem como as características

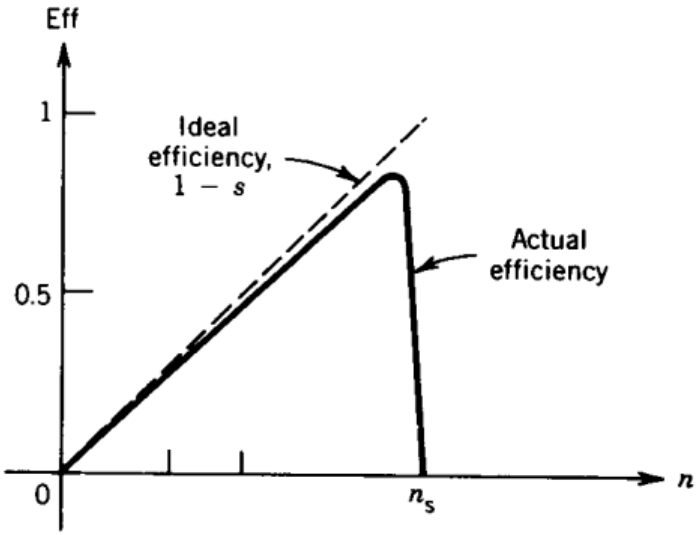

Figura 9 – Curva característica da eficiência em função da velocidade do motor de indução.

importantes do motor que podem ser deduzidas a partir do seu circuito equivalente. Ter em mãos essas características e como o motor se comporta em diferentes momentos da sua atuação é de suma importância para o engenheiro estimar a causa de possíveis defeitos e/ou, ao realizar um projeto e determinar se a máquina em questão será capaz de cumprir a tarefa para a qual ela está sendo designada.

Os capítulos que se seguem tratam da proposta e elaboração de uma solução móvel onde o engenheiro conseguirá ter em mãos essas informações de forma fácil, apenas com o uso de um celular *Android*.

3.2 Princípios da Computação

Nesta seção será feito uma rápida revisão teórica de conceitos da área de informática que são amplamente utilizados nesta monografia e precisam ser compreendidos para total entendimento do texto. Para fundamentação destes conceitos foram utilizadas algumas literaturas de renome. No âmbito da programação, linguagem *Java* e Orientação a Objetos, utiliza-se como base as teorias abordadas no artigo (WEGNER, 1990), no livro (ARNOLD JAMES GOSLING, 2005) e no guia orientação a objetos em *Java* da *Oracle*[1] (ORACLE, 2010). A parte *Android* é habilmente detalhada pela *Google*[2], em seu guia de desenvolvimento *Android* (GOOGLE, 2014).

[1] Empresa responsável pela criação e desenvolvimento da linguagem *Java*
[2] Empresa responsável pela criação e desenvolvimento do sistema operacional *Android*

3.2.1 Orientação a Objeto

No contexto de desenvolvimento de *software* existem diversos modelos de projeto, análise e programação de sistemas. A linguagem de programação *Java*, cuja escolha para o projeto em questão será devidamente explicada na seção de metologia, mais à frente, é completamente ligada ao paradigma de Programação Orientação a Objetos (POO). Toda e qualquer estrutura em *Java* é ou pertence a uma classe sendo impossível falar de desenvolvimento em *Java* sem abordar POO.

3.2.1.1 Objetos

A POO se baseia na composição e interação de diversos blocos de software chamados de objetos, sendo o entendimento do conceito de objeto fundamental para compreender este paradigma. Se olharmos ao nosso redor, no mundo real, temos vários exemplos de objetos: livro, computador, caneta, entre outros. Tais objetos do mundo real podem ser resumidos a duas características principais que os definem: atributos e comportamentos. Um livro, por exemplo, tem atributos (autor, título, ano, número de páginas, página atual) e comportamentos (mudar de página, abrir/fechar). Uma boa prática para começar a entender orientação a objetos é analisar objetos do mundo real detalhando seus estados e comportamentos fundamentais.

Objetos no contexto de *software* são conceitualmente similares aos objetos do mundo real e também são compostos por atributos e comportamentos. Os atributos são armazenados em variáveis, enquanto os comportamentos são manifestados através de funções internas ao objeto, chamadas de métodos. Os estados de um objeto normalmente são internos a este, sendo operados a partir de métodos. Os métodos são, portanto, a principal forma de comunicação entre os objetos. O princípio de manter os estados internos a um objeto e forçar que toda interação seja feita através dos métodos é chamado de encapsulamento e é fundamental na teoria de POO, por prover modularidade ao programa.

Dentro da POO existe um conceito muito importante, por trás do conceito de objetos: o conceito de classes. Uma classe é o protótipo de um objeto, ou seja, é a classe que define os estados e comportamentos que os objetos daquela classe possuem. Em termos de código, é na definição da classe que são criados os atributos e métodos que regem seu funcionamento. A classe, no entanto, tem um papel abstrato, de definição somente. Os objetos é que são utilizados pelo programa durante sua execução e representam instâncias da classe que os definem.

Em uma analogia com o mundo real, uma classe seria a descrição genérica de um tipo de objeto e o objeto seria cada objeto específico que atende aquela descrição. Pode-se descrever um livro como um objeto composto por uma capa e várias páginas, com um autor, editora, ano de publicação e que trata de um assunto qualquer. Essa seria a classe

de um livro. Os livros físicos que usamos no dia-a-dia, como um livro de histórias infantis ou o livro texto de máquinas elétricas, seriam objetos da classe "livro".

Um outro conceito importante da POO é o conceito de herança. Em um projeto de *software* classes, chamadas de filhas, podem herdar outra classe, mãe, afim de dar maior especificidade a ela. Por exemplo, uma classe do tipo "calçados"possui atributos e métodos referente a "calçado". Pode-se criar uma classe do tipo "chinelo"que herda a classe "calçado", ou seja, também é do tipo "calçado", porém tem atributos e métodos específicos ao tipo "chinelo", que nem todo "calçado"tem. Basicamente, todo "chinelo"é um "calçado", mas nem todo "calçado"é um "chinelo".

A utilização do paradigma de POO traz uma série de benefícios para o programa e seu código, como:

a) **Modularidade:** Divisão do conteúdo em módulos, objetos, possibilitando a manutenção de um objeto sem interferir ou depender dos demais.

b) **Privacidade dos Detalhes de Implementação:** Como os objetos são encapsulados e interagem entre si a partir dos métodos, suas implementações acabam ficando privadas o que preserva detalhes de cópias e dificulta a ação de usuários mal intencionados em busca de falhas ou brechas na solução.

c) **Boa Redigibilidade:** A redigibilidade está ligada ao processo de manutenção do código, a facilidade de ler, escrever, compreender e modificar o código feito.

d) **Reutilização de Código:** Como é feito em módulos, o código de um programa pode ser reaproveitado para outros programas ou funcionalidades que necessitem de objetos iguais, parecidos ou com mesmo objetivo dos desenvolvidos neste.

Essas vantagens estão totalmente alinhadas com o nosso propósito de permitir melhorias futuras ao *software*, o que justifica a nossa escolha por uma abordagem orientada ao objeto.

3.2.2 Desenvolvimento *Android*

A escolha da plataforma *Android* será melhor detalhada na metodologia, porém, vale adiantar que trata-se do sistema operacional mais utilizado no universo dos *smartphones*, o que concede grande alcance e difusão para a aplicação. Em segundo lugar, seu desenvolvimento nativo é feito em *Java*, linguagem que, além de ser totalmente voltada ao paradigma de POO, já foi utilizada pelo autor em outros projetos.

Java é uma linguagem de programação muito ampla e uma das mais utilizadas no mundo hoje. No desenvolvimento voltado para dispositivos *Android* existe uma biblioteca, isto é, um conjunto de classes, desenvolvidos pela *Google* para garantir que o

desenvolvimento seja feito de uma maneira compatível com o sistema operacional.

Tendo isso em vista, existem dois conceitos do desenvolvimento *Android* que são necessários explicitar para que se tenha entendimento total de como foi concebido o projeto: atividade e fragmento. A atividade é uma classe genérica que corresponde a uma tela do aplicativo. Durante o desenvolvimento cria-se diversas classes que herdam a classe atividade e a especificam para o uso daquela tela em questão como, por exemplo, uma tela de listagem de máquinas, uma tela de detalhe, entre outras. Assim, cada tela do programa é uma atividade e tem sua classe específica, com seus métodos e atributos.

Os fragmentos, por sua vez, também são uma classe genérica básica do *Android* e correspondem a uma espécide de sub-atividade. Uma atividade pode ter vários fragmentos diferentes que funcionam em paralelo entre si. Um fragmento pode definir uma tela inteira ou pedaços de uma tela e a alternância dos fragmentos é controlada pela atividade que os contém.

A diferença principal de funcionamento entre atividades e fragmentos é que, quando uma atividade está em segundo plano o programa ignora seu funcionamento, isto é, ela não é processada ou gerenciada e pouco, ou nenhum, recurso (memória e processamento) são destinados a ela. Os fragmentos, por sua vez, continuam no processamento se a atividade que os contém estiver ativa, mesmo que esteja em segundo plano.

4 Metodologia

Utilizando conceitos de engenharia de software elaborou-se o projeto do aplicativo a ser desenvolvido. A primeira definição considerada foi o escopo da aplicação, isto é, qual a abrangência do software a ser desenvolvido dentro do contexto de máquinas elétricas. Para tal, foi necessário primeiramente avaliar todas as possibilidades dentro no nosso contexto e alinhar o que foi considerado prioritário ao tempo disponível para execução do projeto, que envolve implementação, testes, correções e análise do resultado final. Com base nisso, optamos por restringir nosso escopo à somente motor de indução, uma vez que é um grupo reduzido, todavia abrangente, isto é, é um motor de ampla abrangência sendo um dos tipos de motores mais utilizados atualmente.

Definido o escopo, a etapa seguinte engloba a definição de requisitos do projeto, ou seja, funcionalidades e restrições inerentes ao software a ser desenvolvido que permitirão ao engenheiro manusear informações sobre o motor de de indução de modo a executar cálculos que facilitem sua tomada de decisão. Neste trabalho, os requisitos estabelecidos foram:

a) Possibilitar ao usuário inserção dos dados referentes ao motor;
   - Informações básicas que descrevem o motor;
   - Dados fornecidos pelo fabricante (placa e catálogo);
   - Dados referentes aos ensaios a vazio e bloqueado (opcional);
   - Dados referentes aos circuitos equivalentes de estator e rotor (opcional).

b) Determinação dos parâmetros do circuito equivalente baseado nos dados fornecidos:
   - A partir dos dados de placa;
   - A partir dos dados dos ensaios;
   - A partir dos parâmetros dos circuitos diretamente fornecidos pelo usuário.

c) Permitir que o usuário obtenha informações e características do motor:
   - Circuito equivalente;
   - Circuito de Thevenin;
   - Curva de Torque;
   - Curva de Corrente de Estator;
   - Curva de Fator de Potência;
   - Curva de Eficiência;

d) Permitir que o usuário armazene, edite e carregue motores de indução do banco de dados.

e) Permitir que o usuário compartilhe os dados obtidos via e-mail.

Estabelecidos os requisitos desejáveis para o projeto, o último passo a ser tomado na definição do software a ser desenvolvido é a definição do dispositivo e do sistema operacional (OS) onde o software será executado, bem como a linguagem de programação na qual ele será escrito e programas a serem utilizados no seu desenvolvimento.

Se tratando de uma solução móvel, uma vez que alguns dos objetivos são otimizar tarefas, reduzir tempo e locomoção, foi escolhido o universo dos smartphones para a implementação do nosso software. Por uma questão de facilidade, abrangência e robustez foi escolhido o sistema operacional *Android*[1], da Google. Inerente a escolha do OS há também a escolha da linguagem de programação: Java[2], linguagem na qual os aplicativos para *Android* são escritos, o que foi um facilitador visto que já havia experiência prévia dos envolvidos com a mesma.

Para o desenvolvimento da aplicação, isto é, implementação e testes, é utilizado o Ambiente de Desenvolvimento Integrado (IDE) da própria Google, o *Android Studio*[3] e o smartphone da *Asus*[4] *Zenfone2* e o simulador virtual do *Android Studio* para o modelo *Nexus 5s*, ilustrado na figura 10.

Para a elaboração dos gráficos a partir dos dados característicos calculados foi utilizada uma biblioteca gráfica *open-source* desenvolvida para android chamada GraphView[5]. Sua escolha foi definida por já conter uma série de funcionalidades úteis para a versão atual do programa bem como versões posteriores que serão abordadas no capítulo de direções futuras.

Além disso, foi utilizado também um sistema de controle de versão de arquivos, um repositório *open source*[6] para o código desenvolvido, ilustrado na figura 11. O objetivo do repositório é, além de garantir o versionamento do software e evitar perdas durante o desenvolvimento, disponibilizar para a comunidade a aplicação com objetivo de difundir conhecimento e reunir possíveis colaboradores no futuro.

Optou-se por realizar o desenvolvimento do software em duas partes distintas:

a) ***Backend*** Consiste na *engine* de processamento[7] do projeto, onde foram definidas as classes pertinentes à solução de software proposta bem como as classes

[1] https://developer.android.com/develop/
[2] http://www.oracle.com/technetwork/java/javase/overview/index.html
[3] https://developer.android.com/studio/index.html
[4] https://www.asus.com/br/
[5] http://www.android-graphview.org/
[6] https://github.com/lgrossi/TCC
[7] *Software*, ou parte de, responsável por realizar as funções centrais do programa.

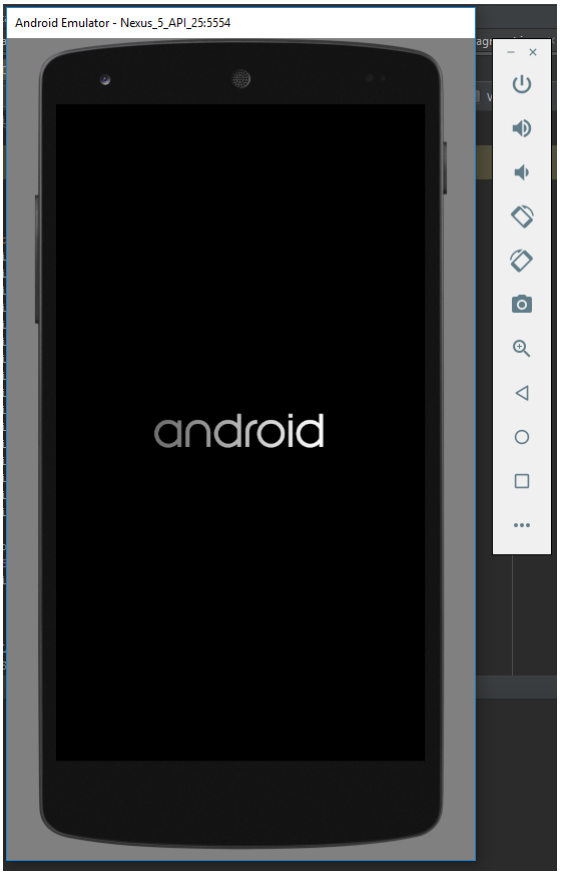

Figura 10 – Simulador nativo do Android modelo Nexus 5S.

responsáveis pelos cálculos e processamento de dados. É o responsável por gerenciar as informações fornecidas pelo frontend e devolver as análises de interesse do usuário.

b) ***Frontend*** Composto por toda a parte visual e de interação com o usuário final. É a casca do programa que gerencia a interação do usuário final com o *backend*. Isto é, fica responsável por receber os comandos do usuário e processá-los de forma a enviar as informações necessárias para a engine de processamento, recebendo de volta as informações que o usuário espera receber. É ele também o responsável por gerênciar a interação com o banco de dados, isso é, fazer as *queries*[8] de consulta e armazenamento de dados fornecidos pelo usuário.

Essa divisão foi escolhida pensando-se em uma solução abrangente e de desenvolvimento a longo prazo, isto é, que virá a ser melhorada e incrementada no futuro, podendo dar suporte a outras plataformas não *Android*. Desta forma, o backend, ou seja, toda a parte de cálculo e processamento de dados, é somente uma biblioteca importada no aplicativo final, sendo independente, portanto, do mesmo. Assim, pode-se usar essa biblioteca para outros fins, como aplicativos em outras plataformas ou mesmo um servidor online que recebe requisições *http* e devolve os cálculos necessários para o cliente.

[8] Requerimentos no banco de dados.

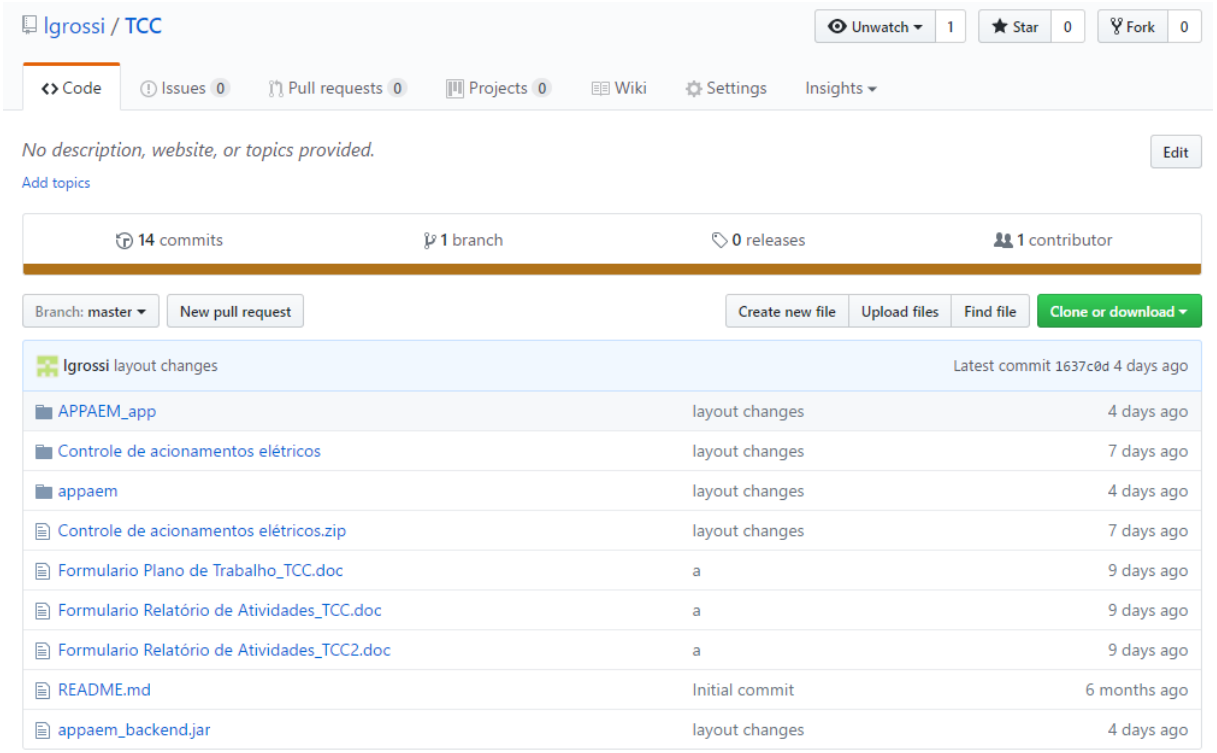

Figura 11 – Repositório do projeto no github.

É um padrão de boas práticas de desenvolvimento a elaboração do banco de dados no *backend* para não sobrecarregar a aplicação cliente. No nosso caso, o banco de dados foi construído no *frontend* pelo fato do *Android* já possuir uma boa biblioteca de banco de dados MySQL, facilitando o desenvolvimento. Além disso, trata-se de tabelas simples e, no caso, o *backend* é somente uma biblioteca que foi desenvolvida e não um servidor propriamente dito. Por conseguinte, *backend* e *frontend* rodam no cliente e, portanto, não faz diferença para o nosso caso em qual parte da aplicação vamos colocar o banco de dados.

Para fins didáticos de comprovação da eficiência do programa e da qualidade da aproximação de seus cálculos, foi utilizado o motor de indução trifásico da WEG modelo IP55, quadripolo, de frequência 60Hz e potência de 5.0 Hp, ilustrado na figura 12 retirada de (WEG, 2005). A escolha desse motor se deu por seu catálogo ser de fácil acesso e por haver um estudo prévio sobre o mesmo, ensaios e circuito equivalente já determinado, possibilitando comparar os resultados com os obtidos pela aplicação.

Potência		Carcaça	RPM	Corrente nominal em 220V (A)	Corrente com rotor bloqueado I_p / I_n	Conjugado nominal C_n (kgfm)	Conjugado com rotor bloqueado C_p / C_n	Conjugado máximo $C_{máx.}/C_n$	Rendimento η %			Fator de potência Cos φ		
cv	kW								% da potência nominal					
									50	75	100	50	75	100
4 Pólos - 60 Hz														
0,16	0,12	63	1720	0,89	4,5	0,07	3,2	3,4	45,0	52,0	57,0	0,46	0,55	0,62
0,25	0,18	63	1710	1,14	4,5	0,10	2,8	3,0	53,0	60,0	64,0	0,47	0,57	0,65
0,33	0,25	63	1710	1,44	4,5	0,14	2,9	2,9	59,0	64,0	67,0	0,48	0,59	0,68
0,50	0,37	71	1720	2,07	5,0	0,21	2,7	3,0	56,0	64,0	68,0	0,48	0,59	0,69
0,75	0,55	71	1705	2,90	5,5	0,31	3,0	3,2	62,0	69,0	71,0	0,49	0,60	0,70
1,0	0,75	80	1720	3,02	7,2	0,42	2,5	2,9	72,0	77,5	79,5	0,62	0,74	0,82
1,5	1,1	80	1720	4,43	7,8	0,62	2,9	3,2	72,0	77,0	79,5	0,60	0,73	0,82
2,0	1,5	90S	1740	6,12	6,4	0,82	2,5	3,0	77,0	81,0	82,5	0,60	0,72	0,78
3,0	2,2	90L	1725	8,70	6,8	1,25	2,6	2,8	79,0	82,0	83,0	0,64	0,75	0,80
4,0	3,0	100L	1725	11,8	7,5	1,66	2,6	2,8	82,0	83,0	83,5	0,61	0,73	0,80
5,0	3,7	100L	1715	14,0	7,6	2,09	2,9	3,1	82,5	84,3	85,5	0,63	0,75	0,81

Figura 12 – Catálogo do motor escolhido para testes.

5 Desenvolvimento e Discussões

Como mencionado na metodologia do presente trabalho, o desenvolvimento foi dividido em *backend* e *frontend*, afim de tornar o código mais reaproveitável, abrangente e configurável para receber melhorias e manutenções no futuro. É importante ressaltar que, em ambas as etapas de desenvolvimento, o projeto do *software* foi pensado afim de possibilitar melhorias futuras, sendo, portanto, o mais flexível, robusto e encapsulado possível. O projeto das duas etapas de desenvolvimento juntamente com seus detalhes de implementação serão detalhados ao longo deste capítulo.

5.1 *Backend*

Primeiramente, é ilustrado na figura 13 o diagrama de classes com as classes mais relevantes que compõem o *backend* do *software*.

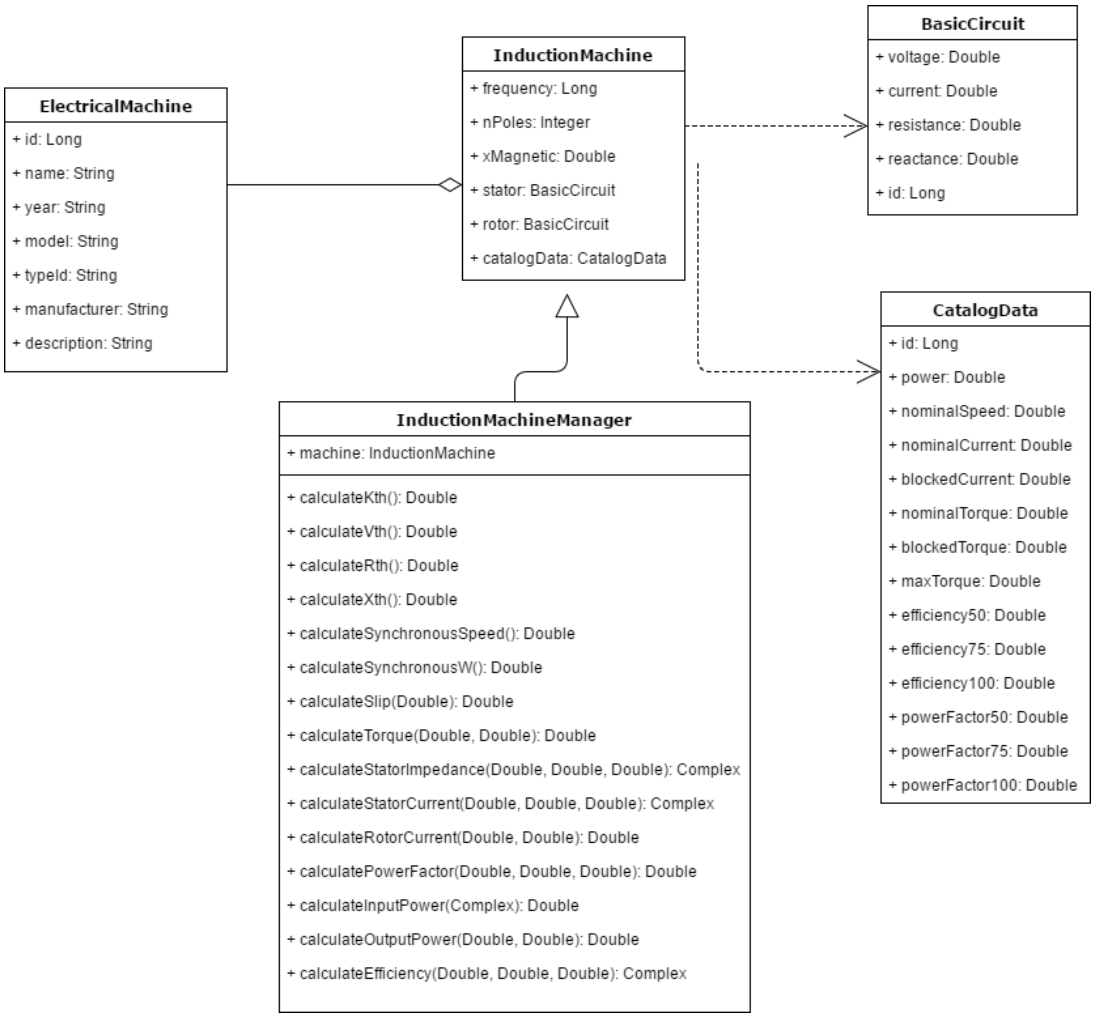

Figura 13 – Diagrama de classes com as classes principais da engine de processamento.

Visando detalhar o projeto e implementação do *backend*, justificando as escolhas feitas, será explicado a seguir cada uma das classes individualmente, assim como os atributos e métodos mais relevantes das mesmas.

5.1.1 *ElectricalMachine*

Afim de realizar o encapsulamento do *software* e deixá-lo preparado para receber melhorias no futuro como, por exemplo, a adição de outros tipos de máquinas elétricas além dos motores de indução, foi criada uma interface genérica para máquinas elétricas.

O intuito dessa interface é ser o mais genérica possível, garantindo assim a abrangência a qualquer tipo de máquina elétrica que venha a ser adicionada ao *software* posteriormente. Para tal, buscou-se reunir somente atributos bastante genéricos nessa interface, isto é, atributos que qualquer máquina elétrica pode ter.

Poderia ter sido criada uma única classe *ElectricalMachine* que comportasse todas as máquinas elétricas e fosse tendo atributos novos adicionados a ela a medida que novas máquinas fossem suportadas pelo programa. Porém, essa abordagem não foi considerada ideal por dificultar manutenção, exigir diversas mudanças no código já funcional para adição de novas máquinas e acabar poluindo o código com muita informação, diminuindo a redigibilidade, dificultando a edição e o entendimento do mesmo.

Os atributos considerados genéricos o suficiente para compor a classe básica de máquinas elétricas foram:

a) ***id*** - Atributo do tipo *Long* correspondente ao ID da máquina no nosso sistema, isto é, um indexador único utilizado na tabela de máquinas do banco de dados para identificá-la. É gerado automaticamente ao registrar a máquina no banco de dados.

b) ***name*** - Atributo do tipo *String* que corresponde ao nome dado pelo o usuário à máquina para facilitar sua identificação.

c) ***year*** - Atributo do tipo *String* que corresponde ao ano de fabricação da máquina.

d) ***model*** - Atributo do tipo *String* correspondente ao modelo da máquina, normalmente fornecido pelo fabricante.

e) ***typeId*** - Atributo do tipo *String* correspondente ao tipo da máquina. No escopo atual, todas as máquinas são do tipo INDUCTION_MACHINE.

f) ***manufacturer*** - Atributo do tipo *String* que corresponde ao nome do fabricante da máquina.

g) ***description*** - Atributo do tipo *String* que corresponde a uma descrição dada a máquina pelo usuário.

5.1.2 *InductionMachine*

Essa classe estende a interface *ElectricalMachine*. A herança, em orientação a objeto, é utilizada para especializar um objeto de forma a manter as características da classe mãe ao mesmo tempo que se incorpora novas características específicas da classe filha.

Com isso, *InductionMachine* ainda é uma máquina elétrica e, como tal, possui todos os atributos da interface *ElectricalMachine*. Além dos atributos da interface que ela estende, a classe *InductionMachine* possui atributos que julgamos ser específicos ou que abrangem quase que exclusivamente as máquinas de indução. São eles:

a) ***frequency*** - Atributo do tipo *Long* correspondente à frequência de funcionamento da máquina. Vale ressaltar aqui que este atributo, especificamente, não é específico de motores de indução. Porém, preferiu-se deixar específico por, nas máquinas de indução, ter grande usabilidade para determinação de alguns parâmetros, como velocidade nominal.

b) ***nPoles*** - Atributo do tipo *Integer* que corresponde ao número de polos da máquina de indução.

c) ***xMagnetic*** - Atributo do tipo *Double* correspondente à reatância de magnetização do circuito equivalente do motor de indução.

d) ***stator*** - Atributo do tipo *BasicCircuit*, classe que será explicada mais a frente, que referencia o circuito de estator.

e) ***rotor*** - Atributo do tipo *BasicCircuit*, classe que será explicada mais a frente, que referencia o circuito de rotor.

f) ***catalogData*** - Atributo do tipo *catalogData*, classe que será explicada mais a frente, que referencia aos dados de catálogo da máquina de indução, isto é, os dados fornecidos pelo fabricante.

Cabe aqui uma observação de que, no futuro, seria uma boa prática a implementação de classes mais genéricas que *InductionMachine* que estendam *ElectricalMachine* para a divisão entre máquinas de corrente alternada e máquinas de corrente contínua. Não foi feito por não haver necessidade uma vez que, no nosso escopo, seria uma classe ociosa.

5.1.3 *BasicCircuit*

Seguindo a filosofia de encapsulamento de código, facilitar redigibilidade, manutenção e futuras alterações e melhorias, essa classe foi criada para representar um modelo básico de circuito. Uma vez que os elementos tensão, corrente, resistência e reatância compõe um circuito básico e são utilizados com frequência, entendeu-se que era uma boa prática isolar esse grupo em uma classe específica que defina um circuito básico.

A função dessa classe, portanto, é encapsular os elementos de um circuito básico em uma única classe, evitando-se criar vários atributos referentes a diferentes circuitos que pode possuir uma máquina. Visto que, várias abordagens, em vários tipos de máquinas diferentes, envolvem a aproximação a partir de circuitos elétricos, é prudente encapsular este grupo de informações para facilitar sua reprodução.

A própria classe *InductionMachine* é um bom exemplo de caso onde o *BasicCircuit* teve um papel importante na organização de código. Ao invés de criar dentro da máquina de indução variáveis que representem cada elemento individual dos circuitos de estator e rotor, criou-se somente duas variáveis que referenciam estes circuitos e, internamente, definem os atributos que compõem um circuito básico.

Os atributos dessa classe são:

a) ***voltage*** - Atributo do tipo *Double* correspondente à tensão de entrada do circuito.

b) ***current*** - Atributo do tipo *Double* que corresponde à corrente do circuito.

c) ***resistance*** - Atributo do tipo *Double* correspondente à resistência total do circuito.

d) ***reactance*** - Atributo do tipo *Double* correspondente à reatância total do circuito

e) ***id*** - Atributo do tipo *Long* correspondente ao ID do circuito no nosso sistema, isto é, um indexador único utilizado na tabela de circuitos do banco de dados para identificá-lo. É gerado automaticamente ao registrar um circuito no banco de dados.

Pode-se questionar a composição dessa classe por não flexibilizar a adição de componentes que talvez estejam presentes em um circuito básico e não são contemplados por ela, bem como a não possibilidade de elaboração de circuitos mais complexos. Entretanto, o intuito dessa classe é, justamente, mediante a uma necessidade futura, permitir tais implementações, simplesmente, estendendo-a ou herdando-a em uma classe mais específica. Além disso, acréscimo de parâmetros à essa classe no futuro, não levaria à quebra da funcionalidade do código já existente.

5.1.4 *CatalogData*

A função dessa classe é reunir toda a informação dada pelo fabricante, isto é, as informações de catálogo da máquina de indução em um único objeto, facilitando obtenção, manuseio e edição de dados. A classe é auto-explicativa, a seguir são listados os atributos nela presentes:

a) ***id*** - Atributo do tipo *Long* correspondente ao ID do dado de catálogo no nosso sistema, isto é, um indexador único utilizado na tabela de dados de catálogo do banco de dados para identificá-lo. É gerado automaticamente ao registrar um dado de catálogo no banco de dados.

b) ***power*** - Atributo do tipo *Double* que corresponde à potência do motor, em watts.

c) ***nominalSpeed*** - Atributo do tipo *Double* correspondente à velocidade nominal do motor.

d) ***nominalCurrent*** - Atributo do tipo *Double* correspondente à corrente nominal do motor.

e) ***blockedCurrent*** - Atributo do tipo *Double* correspondente à corrente com rotor bloqueado do motor.

f) ***nominalTorque*** - Atributo do tipo *Double* que corresponde ao torque nominal do motor.

g) ***blockedTorque*** - Atributo do tipo *Double* correspondente ao torque com rotor bloqueado do motor.

h) ***maxTorque*** - Atributo do tipo *Double* correspondente ao torque máximo do motor.

i) ***efficiency50*** - Atributo do tipo *Double* correspondente à eficiência do motor com 50% da potência.

j) ***efficiency75*** - Atributo do tipo *Double* correspondente à eficiência do motor com 75% da potência.

k) ***efficiency100*** - Atributo do tipo *Double* correspondente à eficiência do motor com 100% da potência.

l) ***powerFactor50*** - Atributo do tipo *Double* correspondente ao fator de potência do motor com 50% da potência.

m) ***powerFactor75*** - Atributo do tipo *Double* correspondente ao fator de potência do motor com 75% da potência.

n) ***powerFactor100*** - Atributo do tipo *Double* correspondente ao fator de potência do motor com 100% da potência.

.

Pensando a nível de projeto, seria prudente a alteração do nome dessa classe para *IMCatalogData*, uma vez que outros tipos de máquinas também podem vir a possuir dados de catalogo, fornecidos pelo fabricante e os dados aqui contidos são específicos para motores de indução. Uma boa prática seria, inclusive, estender uma classe genérica *CatalogData*

que fizesse parte dos dados genéricos de máquinas elétricas, em *ElectricalMachine*. Essas alterações não foram feitas de imediato visando a entrega do projeto dentro do prazo, pois não interferem na versão atual e sua relevância é pequena se comparada a funcionalidades mais relevantes que estão em foco do desenvolvimento no momento.

5.1.5 *InductionMachineManager*

Corresponde à classe principal do *backend* desenvolvido. Responsável por reunir toda a parte de cálculo do projeto, essa classe é, como o nome diz, uma gerente de máquina de indução. Isto é, ela recebe como parâmetro um motor de indução e possibilita a requisição de diversas operações relacionadas ao mesmo que venham a ser necessárias.

Optou-se por esse modelo centralizado em uma classe destinada especificamente aos cálculos envolvendo motores de indução para, em primeiro lugar, deixar a classe de motor de indução simples e legível. Em segundo lugar, foi escolhido concentrar todo o código da parte de cálculos em um único lugar facilitando, assim, pontos considerados importantes e sobre os quais o projeto deste *software* foi desenvolvido como: redigibilidade, manutenção e flexibilidade para melhorias no futuro.

Como já mencionado, o único atributo que essa classe possui é uma máquina de indução, do tipo *InductionMachine*, correspondente à máquina sobre as quais os cálculos serão realizados.

Diferentemente das demais classes, essa classe possui métodos que vão além de *getters* e *setters*[1] e que são de suma importância para o entendimento do funcionamento do *software*. Portanto, nesta seção vamos nos ater à descrição detalhada de cada um dos métodos contextualizando, quando necessário, seu funcionamento com expressões relativas ao capítulo de fundamentação teórica deste trabalho.

Métodos contemplados nesta classe:

a) ***Double calculateKth ()***

- **Descrição:** Esse método é responsável pelo cálculo de K_{th} à partir do circuito equivalente da máquina de indução. Utiliza-se da fórmula expressa na equação 3.13.
- **Parâmetros:** Não recebe nenhum parâmetro de entrada.
- **Retorno:** Retorna o valor calculado de K_{th}.

b) ***Double calculateVth ()***

- **Descrição:** Esse método é responsável pelo cálculo de V_{th}, isto é, a tensão de Thevenin, à partir do circuito equivalente da máquina de indução. Utiliza-se da fórmula expressa na equação 3.10.

[1] Métodos utilizados para gerenciar atributos internos a uma classe.

- **Parâmetros:** Não recebe nenhum parâmetro de entrada.
- **Retorno:** Retorna o valor calculado de V_{th}.

c) ***Double calculateRth ()***

- **Descrição:** Esse método é responsável pelo cálculo de R_{th}, isto é, a resistência de Thevenin, à partir do circuito equivalente da máquina de indução. Utiliza-se da fórmula expressa na equação 3.11.
- **Parâmetros:** Não recebe nenhum parâmetro de entrada.
- **Retorno:** Retorna o valor calculado de R_{th}.

.

d) ***Double calculateXth ()***

- **Descrição:** Esse método é responsável pelo cálculo de X_{th}, isto é, a reatância de Thevenin, à partir do circuito equivalente da máquina de indução. Utiliza-se da fórmula expressa na equação 3.12.
- **Parâmetros:** Não recebe nenhum parâmetro de entrada.
- **Retorno:** Retorna o valor calculado de X_{th}.

.

e) ***Double calculateSynchronousSpeed ()***

- **Descrição:** Esse método é responsável pelo cálculo de n_s, isto é, a velocidade síncrona da máquina de indução. Utiliza-se da fórmula expressa na equação 3.2.
- **Parâmetros:** Não recebe nenhum parâmetro de entrada.
- **Retorno:** Retorna o valor calculado da velocidade síncrona.

.

f) ***Double calculateSynchronousW ()***

- **Descrição:** Esse método é responsável pelo cálculo de ω_s, isto é, a velocidade angular síncrona da máquina de indução. Utiliza-se da fórmula expressa na equação 3.3.
- **Parâmetros:** Não recebe nenhum parâmetro de entrada.
- **Retorno:** Retorna o valor calculado de ω_s.

.

g) ***Double calculateSlip (Double n)***

- **Descrição:** Esse método é responsável pelo cálculo do escorregamento da máquina dada uma velocidade n de motor. Utiliza-se da equação 3.1 para realizar o cálculo do escorregamento.

- **Parâmetros:** Variável n do tipo *Double* que corresponde à velocidade instântanea da máquina.
- **Retorno:** Retorna o valor calculado para o escorregamento a partir da velocidade instantânea n recebida.

.

h) ***Double calculateTorque (Double n, Double r2)***

- **Descrição:** Esse método é responsável pelo cálculo do torque momentâneo da máquina de indução dada uma velocidade n e um valor de resistência r2. Por padrão, utiliza-se a resistência do próprio circuito equivalente de rotor. Entretanto, foi feito desta forma afim de suportar, no futuro, o acréscimo de resistências agregadas ao rotor para melhorar o torque de partida. Refere-se à equação 3.15.
- **Parâmetros:** Variável n do tipo *Double* que corresponde à velocidade instântanea da máquina e varíavel r2 do tipo *Double* que corresponde à resistência de rotor da máquina.
- **Retorno:** Retorna o valor calculado para o Torque momentâneo a partir da velocidade instantânea n e da resistência de rotor recebidas.

.

i) ***Complex calculateStatorImpedance (Double n)***

- **Descrição:** Esse método é responsável pelo cálculo da impedância de entrada momentânea do estator da máquina de indução dada uma velocidade n. Refere-se à equação 3.45.
- **Parâmetros:** Variável n do tipo *Double* que corresponde à velocidade instântanea da máquina.
- **Retorno:** Retorna um valor do tipo *Complex* que corresponde a um número complexo com módulo e fase, calculados para a impedãncia de entrada momentânea do estator a partir da velocidade instantânea n.

.

j) ***Complex calculateStatorCurrent (Double n)***

- **Descrição:** Esse método é responsável pelo cálculo da corrente momentânea de estator da máquina de indução dada uma velocidade n. Refere-se à equação 3.47.
- **Parâmetros:** Variável n do tipo *Double* que corresponde à velocidade instântanea da máquina.
- **Retorno:** Retorna um valor do tipo *Complex* que corresponde a um número complexo com módulo e fase, calculados para a corrente de a partir da velocidade instantânea n.

.

k) ***Double calculateRotorCurrent (Double n, Double r2)***

- **Descrição:** Esse método é responsável pelo cálculo da corrente momentânea no rotor da máquina de indução dada uma velocidade n e um valor de resistência r2. Por padrão, utiliza-se a resistência do próprio circuito equivalente de rotor. Entretanto, foi feito desta forma afim de suportar, no futuro, o acréscimo de resistências agregadas ao rotor para melhorar o torque de partida. Refere-se à equação 3.14.
- **Parâmetros:** Variável n do tipo *Double* que corresponde à velocidade instântanea da máquina e varíavel r2 do tipo Double que corresponde à resistência de rotor da máquina.
- **Retorno:** Retorna o valor calculado para o módulo da corrente no rotor a partir da velocidade instantânea n e da resistência de rotor recebidas.

.

l) ***Double calculatePowerFactor (Double n)***

- **Descrição:** Esse método é responsável pelo cálculo do fator de potência momentâneo da máquina de indução dada uma velocidade n. Refere-se à equação 3.48.
- **Parâmetros:** Variável n do tipo *Double* que corresponde à velocidade instântanea da máquina.
- **Retorno:** Retorna o valor calculado para o fator de potência momentâneo da máquina de indução a partir da velocidade instantânea n.

.

m) ***Double calculateInputPower (Complex statorCurrent)***

- **Descrição:** Esse método é responsável pelo cálculo da potência de entrada momentânea da máquina de indução dados módulo e fase da corrente de estator. Refere-se à equação 3.49.
- **Parâmetros:** Varíavel do tipo *Complex* correspondente a corrente de estator, com seu módulo e fase.
- **Retorno:** Retorna o valor calculado para a potência de entrada momentânea da máquina de indução a partir da corrente de estator naquele momento.

.

n) ***Double calculateOutputPower (Double rotorCurrent, Double slip)***

- **Descrição:** Esse método é responsável pelo cálculo da potência mecânica momentânea da máquina de indução a partir do módulo da corrente do rotor e do escorregamento momentâneos. Refere-se à equação 3.7.

- **Parâmetros:** Varíaveis do tipo *Double* correspondentes ao módulo da corrente no rotor e ao escorregamento momentâneos.
- **Retorno:** Retorna o valor calculado para a potência mecânica momentânea da máquina de indução a partir do módulo da corrente de rotor e do escorregamento naquele momento.

.

o) ***Double calculateEfficiency (Double n)***

- **Descrição:** Esse método é responsável pelo cálculo da eficiência momentânea da máquina de indução dado uma velocidade instantânea n. Refere-se à equação 3.50.
- **Parâmetros:** Varíavel do tipo *Double* correspondente a velocidade momentânea da máquina.
- **Retorno:** Retorna o valor calculado para a eficiência da máquina de indução a partir da velocidade da mesma naquele momento.

.

p) ***Pair<Double, BasicCircuit> calculateEquivalentCircuitFromTests(Double p0, Double i0, Double v0, Double pBl, Double iBl, Double vBl, Double r1)***

- **Descrição:** Esse método calcula o circuito equivalente a partir dos dados dos ensaios descritos na seção 3.4.1, utilizando-se das expressões 3.16, 3.16, 3.18, 3.19, 3.22, 3.26 e 3.28.
- **Parâmetros:** Os parâmetros de entrada correspondem aos sete parâmetros determinados a partir dos testes a vazio (I_0, V_0 e P_0), com rotor bloqueado (I_{bl}, V_{bl}, P_{bl}) e medição da resistência de estator (R_1).
- **Retorno:** O retorno desse é composto de um par de elementos contendo um *Double*, referente à indutância de magnetização, e um *BasicCircuit*, referente ao circuito de rotor. A princípio esse retorno pode parecer estranho uma vez que o circuito de estator também faz parte do circuito equivalente. Optou-se por esse retorno por uma questão de otimização uma vez que, para o circuito de estator, R_1 já é conhecido e X_1 é equivalente a X_2, então, determiná-lo não é necessário.

.

q) ***Pair<Double, BasicCircuit> calculateEquivalentCircuitFromCatalog(Double pf100, Double iN, Double wS, Double tN, Double sN, Double v1, Double tMax, Double kTh)***

- **Descrição:** Esse método calcula o circuito equivalente a partir dos dados de fabricante por meio do método descrito na seção 3.4.2, utilizando-se das

equações 3.29, 3.32, 3.33, 3.34, 3.36, 3.39, 3.40, 3.41, 3.42 e 3.43. Esse método utiliza os parâmetros fornecidos pelo fabricante e k_{th} (inicialmente igual a 1) e realiza de duas a três iterações recursivas para melhorar a aproximação dos valores.

- **Parâmetros:** Os parâmetros de entrada correspondem aos parâmetros fornecidos pelo fabricante necessários para determinação do circuito equivalente: fator de potência nominal, corrente nominal, velocidade angular síncrona, torque nominal, escorregamento nominal, tensão de estator nominal e torque máximo, além de kTh.
- **Retorno:** O retorno desse é composto de um par de elementos contendo um *Double*, referente à indutância de magnetização, e um *BasicCircuit*, referente ao circuito de rotor. A princípio esse retorno pode parecer estranho uma vez que o circuito de estator também faz parte do circuito equivalente. Optou-se por esse retorno por uma questão de otimização uma vez que, para o circuito de estator, R_1 já é conhecido e X_1 é equivalente a X_2, então, determiná-lo não é necessário.

.

.

5.2 *Frontend*

O projeto do *frontend* se estrutura a partir de uma cadeia navegável de atividades, isto é, uma atividade principal é exibida quando o programa inicializa e o usuário pode interagir com a aplicação navegando por atividades ligadas entre si ou utilizando o menu de navegação. As objetos utilizados no *frontend* são resumidos, basicamente, em atividades e fragmentos, classes nativas da biblioteca *Android*, responsáveis pelo controle da parte gráfica, cada qual ligada a um arquivo *xml* correspondente ao seu *layout*.

Cada atividade corresponde a um escopo, isto é, agrupa fragmentos e funcionalidades responsáveis pelo mesmo tipo de tarefa. Exemplo: a atividade detalhar motor de indução reúne os fragmentos 'detalhe do motor' e 'estimar características do motor', bem como os métodos relativos às chamadas ao *backend* que geram os dados das curvas características. Cada escopo é individual e se torna inativo quando a atividade correspondente a ele não esta em primeiro plano.

Os fragmentos, por sua vez, são *tasks* internas a uma atividade que funcionam em paralelo entre si, dentro daquela atividade. Estando essa atividade em primeiro plano, seu escopo está ativado e todos os fragmentos internos a ela estão em funcionamento, estando eles em primeiro plano ou não.

Essa noção de diferença de funcionamento e hierarquia entre fragmentos e atividades *Android* é importante para se projetar a distribuição das telas entre atividades e fragmentos afim de tornar o mais eficiente possível, isto é, deixar os escopos bem definidos evitando o mínimo de processamento desnecessário rodando em paralelo.

Uma vez entendido o funcionamento das atividades e fragmentos e dado os fatos de que a parte relevante de cálculo se encontra implementada no *backend* e as classes de atividades e fragmentos padronizadas e muito parecidas entre si, esta seção não vai se ater a detalhes de implementação com exposição de código. Será dado enfoque nas telas e funcionalidades do aplicativo, ou seja, na disposição da sua parte gráfica.

Assim sendo, ao invés de detalhar de maneira minuciosa cada classe como foi feito com o *backend*, detalhar-se-a as partes relevantes do aplicativo: telas e menus.

5.2.1 *Menu*

O menu é importante para navegação do usuário dentro do *software*, possibilitando o entendimento dos blocos de funcionalidades principais que ele contem. Por esse motivo, optou-se por modelar um menu do tipo *Navigation-Drawer*, isto é, um menu lateral de navegação como mostrado na figura 14, muito comum em dispositivos *Android*.

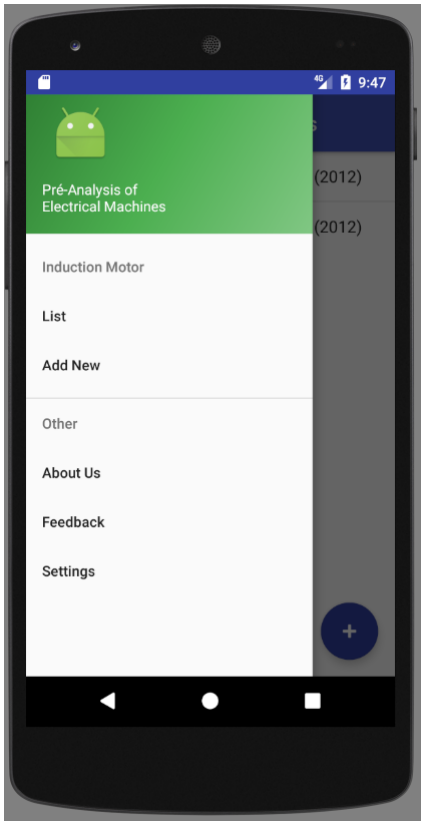

Figura 14 – Menu de navegação do aplicativo.

Como ilustrado na figura 14, o menu projetado é dividido em blocos correspondentes a grupos de escopos correlacionados, cada bloco contendo apenas funcionalidades

principais. Modelou-se dessa forma com o propósito de minimizar a quantidade de informação exposta no menu, por motivos de clareza, sem deixar de dar uma visão geral da aplicação ao usuário.

Além disso, a forma como o menu foi projetado permite adição de novos grupos e atividades sem prejudicar o layout e funcionamento atual. Os dois grandes grupos no menu atual são Motores de Indução e Outros, uma vez que o escopo do aplicativo aqui proposto abrange somente motores de indução. Os itens do menu serão detalhados individualmente a seguir.

InductionMotor

Agrupa atividades genéricas referentes a motores de indução, isto é, que não estão diretamente ligados a um objeto motor específico. Demais funções como detalhar motor de indução, estimar circuito equivalente e obter curvas características não foram adicionadas ao menu por serem específicas e são acessadas navegando-se diretamente pelas telas.

a) ***List:*** Direciona o usuário para a tela de listagem de motores de indução.

b) ***Add:*** Direciona o usuário para a tela onde é possível adicionar um novo motor de indução.

.

Others

Agrupa atividades genéricas diversas que não estão diretamente ligadas ao escopo principal do programa. Aqui optou-se por incluir atividades que, normalmente, toda aplicação tem para *feedback:* do usuário, informações sobre o *software* e configurações.

a) ***About Us:*** Direciona o usuário para a tela de informações sobre o *software*.

b) ***Feedback:*** Direciona o usuário para a tela de feedback, canal pelo qual o usuário poderá se comunicar com os desenvolvedores.

c) ***Settings:*** Direciona o usuário para a tela de configurações, onde o usuário poderá customizar e ajustar alguns detalhes conforme preferir.

.

5.2.2 Tela Listagem de Motores de Indução

Tela correspondente à atividade principal do aplicativo, ou seja, é a tela exibida quando o *software* é inicializado. É responsável por reunir os motores de indução cadastrados pelo usuário e listá-los, em ordem alfabética. A tela de listagem é ilustrada pela figura 15.

Conforme mostrado na figura 15, a tela de listagem de motores de indução lista as máquinas em ordem alfabética exibindo o index, o nome dado a ela pelo usuário, modelo e

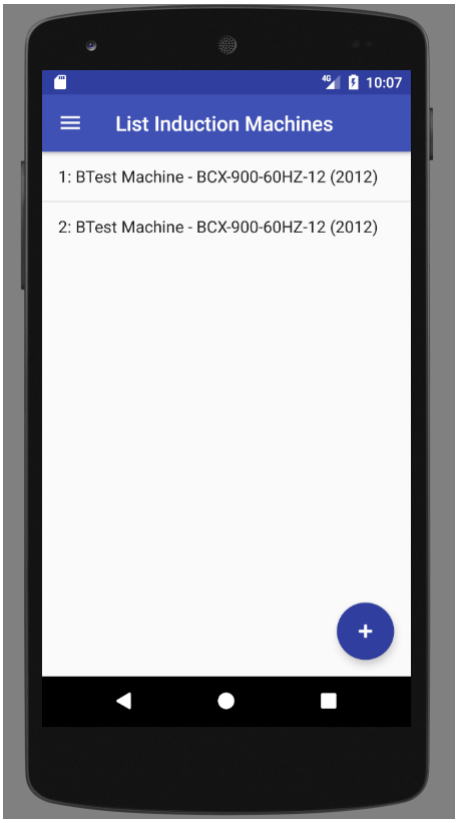

Figura 15 – Tela de listagem de máquinas de indução do aplicativo.

ano da máquina. A atividade dessa tela é responsável pela busca das máquinas registradas no banco de dados, ordenação delas alfabeticamente e montagem dos *labels* de cada linha a partir das informações da máquina.

Essa tela permite que o usuário execute algumas ações de navegação independentes do menu principal. Clicar no botão flutuante de adicionar nova máquina irá chamar a tela de adição de motor de indução, equivalente aquela referenciada pelo item do menu principal. O usuário pode, também, clicar em uma das linhas da listagem, detalhando a máquina escolhida na tela de detalhes de motor de indução.

5.2.3 Tela Detalhe de Motor de Indução

Responsável por detalhar o motor de indução escolhido, essa tela traz as informações pertinentes a uma dada máquina e está ilustrada na figura 16. Conforme mostrado, a tela de detalhes traz as informações do motor registradas pelo usuário, bem como o circuito equivalente associado a ele.

Mais simples que a tela de listagem, a tela em questão não possui interação com o banco de dados. As informações listadas são do objeto Máquina de Indução que foi criado na tela de listagem a partir da busca no banco e, aqui, são somente exibidas de forma clara e visível para o usuário.

Figura 16 – Tela de detalhe de máquina de indução do aplicativo.

Essa tela permite que o usuário execute algumas ações de navegação independentes do menu principal. Clicar no botão flutuante trará opções ao usuário: associar um circuito equivalente à máquina, que redireciona à tela de determinação de circuito equivalente ou traçar suas curvas características, que redireciona à tela de plotagem de gráficos referentes ao motor de indução. O usuário pode também retornar à atividade anterior,a de listagem, apertando voltar, ou seja, subindo um nível na cadeia de atividades.

5.2.4 Tela Curvas Características

Essa tela possibilita ao usuário traçar perfis característicos do motor com base no circuito equivalente para ele definido. A figura 17 ilustra a tela em questão. Em termos de navegação ela é mais simples que as demais mostradas até agora, permitindo somente voltar à atividade anterior, a de detalhes do motor, apertando voltar, ou seja, subindo um nível na cadeia de atividades.

O usuário pode alternar a característica da qual deseja exibir a curva clicando nos botões logo abaixo do gráfico. Além disso, o gráfico permite interação do usuário, isto é, clicar no mesmo realiza ações como detalhar o ponto ou deslocar a curva. Tais ações são mais complexas e, portanto, não foi possível cobrí-las totalmente no escopo deste projeto. Assim, serão tratadas com mais detalhes na seção de direções futuras no capítulo de conclusão do presente artigo.

Vale ressaltar que a utilização da biblioteca de gráficos *open-source* explicitada na metodologia possibilita que o usuário interaja com os gráficos dando *zoom* e navegando pela escala, da forma que preferir.

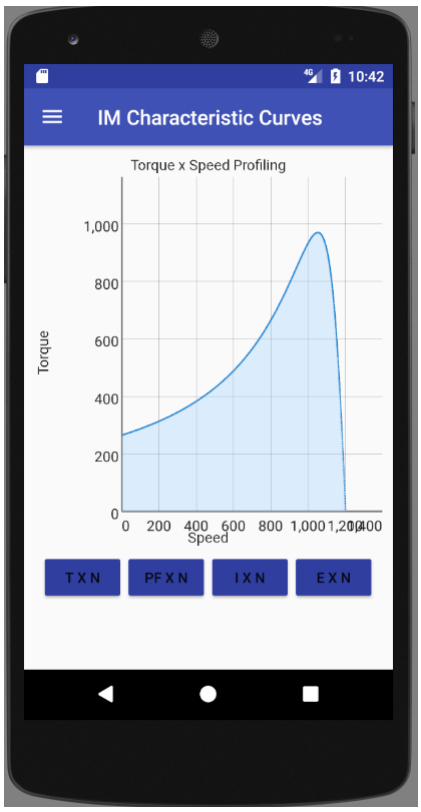

Figura 17 – Tela de curvas características de máquina de indução do aplicativo.

5.2.5 Tela Determinação de Circuito Equivalente

Nessa tela o usuário poderá vincular um circuito equivalente à máquina de indução. Por padrão, se a máquina já tiver um circuito equivalente associado a ela, ao definir um novo o usuário irá sobrescrever o antigo. Há três formas de definir um circuito equivalente para o motor de indução:

a) **Inserção dos parâmetros diretamente:** O usuário pode inserir diretamente os parâmetros de circuito equivalente, caso ele já os tenha calculado empiricamente, por ensaio ou de alguma outra forma que prefira.

b) **Determinação a partir de ensaios:** O usuário pode inserir os dados dos três ensaios descritos na seção 3.4.1 para que a aplicação realize a determinação do circuito equivalente a partir deles.

c) **Determinação a partir dos dados de fabricante:** O usuário pode inserir os dados de catálogo e placa, os quais a aplicação utilizará para realizar a determinação dos parâmetros a partir do método descrito na seção 3.4.2.

5.2.6 Outras telas

Mais três telas foram desenhadas para esse projeto inicial: sobre nós, fale conosco e configurações. Fora do escopo principal do projeto, essas telas servem para agregar valor ao desenvolvimento do software, possibilitando coletar informaçoes do usuário (sugestões, bugs e outros) por meio do "Fale Conosco", permitir ao usuário personalizar algumas funcionalidades (*layout*, escala padrão do gráfico) por meio do "Configurações"e saber detalhes do projeto, versões e equipe de desenvolvimento a partir do "Sobre Nós".

5.3 Discussões

O software desenvolvido, como detalhado neste capítulo, implementa uma solução para o problema abordado na introdução do corrente trabalho. A solução foi divida em duas partes, uma voltada para a interface com o usuário outra projetada para realizar os cálculos relativos às máquinas de indução.

O método proposto na seção 4.3.2 para determinação de parâmetros de dados de placa é, como mencionado no capítulo de revisão bibliográfica, deduzido com base em métodos observados na literatura e fórmulas da fundamentação teórica deste trabalho. Apesar de realizar algumas aproximações, esse método se mostrou suficiente para uma primeira análise em campo, uma vez que trouxe resultados satisfatórios para os valores de circuito equivalente, com erros de três a seis por cento.

Tais valores foram obtidos de testes realizados com um motor quadripolo da *WEG*, frequência de 60Hz e potência de 5.0 Hp, como detalhado no capítulo de Metodologia do presente artigo, e são apresentados nos gráficos das figuras 19 e 18. Separou-se X_m dos demais, por motivo de escala, melhorando a visualização.

Os resultados também estão descritos na tabela 1. Os erros ficaram numa faixa considerada boa, de três a seis por cento, em média, o que é satisfatório para o propósito deste trabalho. É pertinente ressaltar que R_1 não foi considerado nas comparações uma vez que seu valor é desprezado no método de determinação por dados de placa. Além disso, o ensaio que envolve sua determinação é bastante simples, sendo plausível sua utilização nos dois casos.

	Determinação por Dados do Fabricante	Determinação por Ensaios	Erro
R_2	0.528Ω	0.534Ω	1.12%
X_1	0.914Ω	0.97Ω	5.77%
X_2	0.914Ω	0.97Ω	5.77%
X_m	15.43Ω	15.03Ω	2.66%

Tabela 1 – Comparação dos resultados da estimação dos parâmetros de circuito equivalente por ensaio e por dados de fabricação.

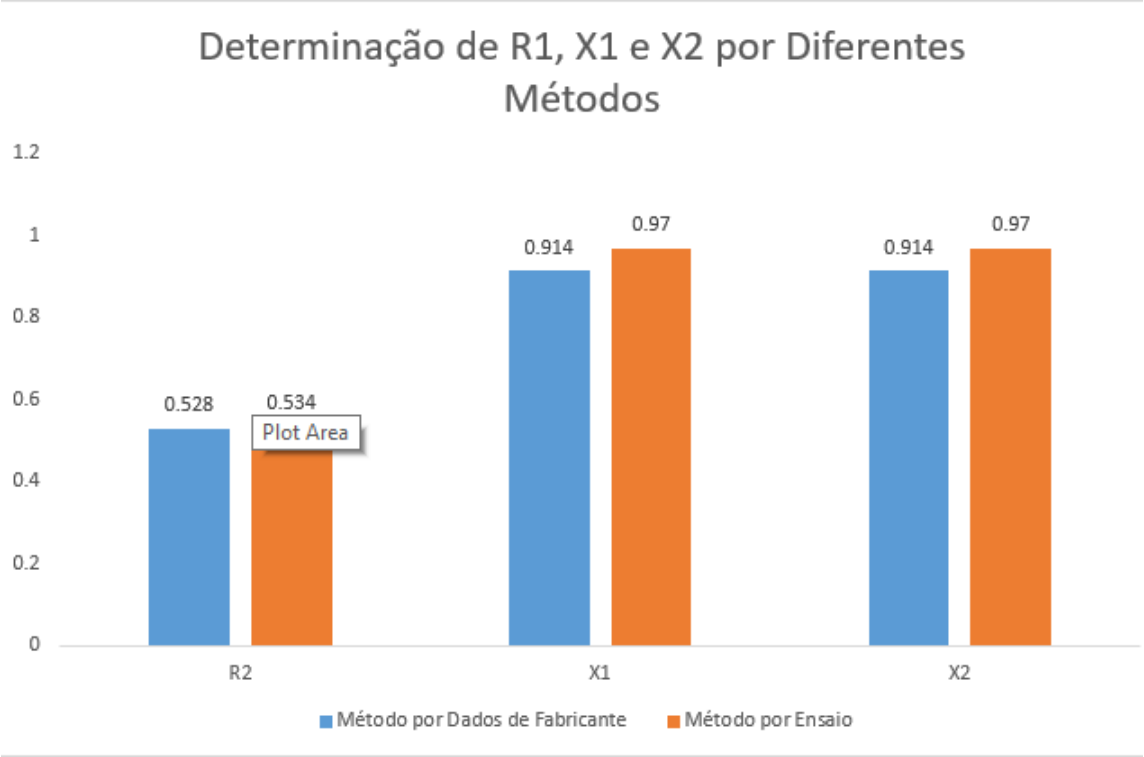

Figura 18 – Gráfico dos valores estimados de R_2, X_1 e X_2 a partir de ensaios e a partir de dados de fabricação.

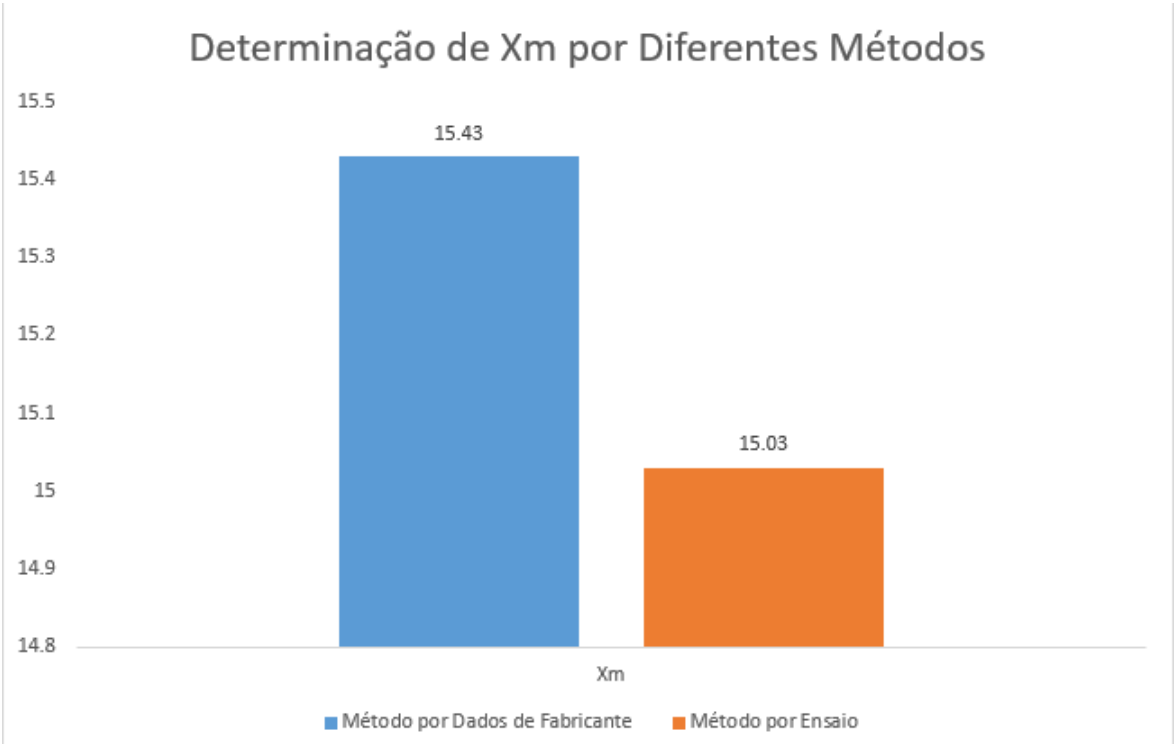

Figura 19 – Gráfico dos valores estimados de X_M a partir de ensaios e a partir de dados de fabricação.

6 Conclusão

6.1 Considerações Finais

Apresentou-se o projeto e desenvolvimento de um aplicativo móvel voltado para pré-análise de motores de indução. Motivado pela falta de aplicações modernas dentro do contexto de máquinas elétricas, esse trabalho se propôs a elaborar uma solução *Android* que o fizesse, aproximando as tecnologias modernas ao universo de máquinas elétricas. Para tal, baseou-se na necessidade do engenheiro em campo ter uma visão geral da máquina com a qual está lidando para determinar defeitos ou garantir que seu funcionamento atenda os projetos para os quais venha a ser destinada.

O *software* desenvolvido é uma aplicação móvel, que realiza de forma prática, pelo celular, cálculos e inferências relativas às máquinas de indução que, normalmente, demandam softwares e ferramentais externos ao campo de trabalho do engenheiro. A aplicação se mostrou eficiente ao realizar, em tempo real, os cálculos necessários e chegar nos mesmos resultados de outros *softwares* computacionais, com mesma função, como, por exemplo, *Scripts Matlab*.

Além disso, o aplicativo permite a determinação, caso o engenheiro não tenha, do circuito equivalente da máquina, seja a partir de ensaios realizados ou, simplesmente, pela inserção de alguns dados fornecidos pelo fabricante. O método por ensaios com a máquina é amplamente conhecido e é considerado um método confiável de aproximação de parâmetros de circuito equivalente. Seus resultados, para a máquina testada, foram equivalentes aos de outros *softwares*, de computador, que realizam os mesmos cálculos. Na comparação, considerou-se *Scripts Matlab*, utilizados na disciplina Laboratório de Conversão, da grade curricular do curso de Engenharia Elétrica da UFMG.

Já o método de inferência, a partir de dados fornecidos pelo fabricante, foi obtido desenvolvendo-se equações baseadas na teoria de máquinas de indução e, também, outros métodos de inferência. O valores fornecidos por este método apresentaram erros na faixa de três a seis por cento, quando comparados com a determinação por ensaios. Os resultados foram considerados satisfatórios, para uma primeira análise, uma vez que possuiu porcentagens baixas de erros, implicando que as curvas geradas a partir deles provém, com considerável qualidade, informações a respeito do funcionamento do motor.

A análise dessas curvas permite ao engenheiro tirar conclusões sobre a máquina em questão como: validar o funcionamento atual da máquina com o funcionamento ideal fornecido pelas curvas, verificar possíveis falhas ou comparar o funcionamento ideal obtido com o que deseja-se realizar com projeto específicos.

Portanto, considera-se satisfatório e de grande relevância para o engenheiro em campo contar, de forma tão imediata, com estas informações a respeito do motor e seu funcionamento.

6.2 Direções Futuras

Na concepção deste *software* foi idealizada uma aplicação robusta, multiplataformas, que atenda uma gama de necessidades do universo de motores elétricos. Por uma necessidade temporal para realização deste projeto, entretanto, optou-se por uma primeira versão que abrange menor escopo. Primeiramente, seu universo é restrito aos motores de indução e seu funcionamento se estende somente a dispositivos *Android*.

Assim, existe um grande leque de possibilidades para o futuro, envolvendo a solução aqui proposta. Um passo importante seria aumentar seu alcance. Primeiramente por meio de divulgação e ampliação dos meios de adquirí-lo: adicionar à *Google Play*[1] e criar site explicando o projeto, o aplicativo e disponibilizando-o para *download*. Outra maneira de aumentar seu alcance, seria estender o número de plataformas que o suportam. O *iOs*, sistema operacional da *Apple*[2], grande concorrente do *Android* e responsável por uma boa fatia do mercado de *smartphones*, é um exemplo de plataforma ampla e que é desejável que o *software* seja suportado na mesma.

Há ainda a possibilidade de criação de aplicações *web*, onde haveria um servidor com o *backend* e várias aplicações, inclusive em plataformas diferentes, que efetuariam chamadas no servidor para realizar os cálculos necessários. Essa abordagem seria limitada à necessidade de *internet*, mas aumentaria a flexibilidade do programa e facilitaria a manutenção e gerenciamento multi-plataformas. O aumento da abrangência de tipos de máquinas suportadas pelo aplicativo também é uma melhoria desejável e muito relevante para o crescimento e alcance do projeto.

Ademais, esse projeto volta atenção para um campo importante e não muito explorado hoje, que é a união dos conhecimentos de engenharia com a computação móvel. A variedade de possibilidades é tanta que, a partir das ideias que são destacadas aqui, pode-se pensar, para o futuro, inúmeros outros projetos que englobem engenharia e computação móvel, indo além do campo de máquinas elétricas, como soluções técnicas e gerenciais para comercialização de energia, eletrônica de potência e microeletrônica, por exemplo.

[1] Loja de aplicativos da *Google*. https://play.google.com/store/apps?hl=en
[2] https://www.apple.com/iphone/

Referências

ABBATE, J. *Inventing the Internet (Inside Technology)*. 1th. ed. [S.l.]: MIT Press, 2000. 264 p. Citado na página 15.

ARNOLD JAMES GOSLING, D. H. K. *THE Java^{TM} Programming Language*. 4th. ed. [S.l.]: Addison Wesley Professional, 2005. 928 p. Citado 2 vezes nas páginas 20 e 35.

ASSUNçãO J. T., G. H. D. Desenvolvimento de um algoritmo para simulação do motor de indução trifásico. In: *Anais do 12o Encontro de Iniciação Cientíca e PósGraduação do ITA - XII ENCITA*. [S.l.: s.n.], 2006. Citado na página 19.

BURGESS, M. P. D. *Faraday to Shockley*. 2008. Disponível em: <https://sites.google.com/site/transistorhistory/faraday-to-shockley>. Citado na página 15.

FITZGERALD, A. E. *Electric Machinery*. 6th. ed. [S.l.]: Mc Graw Hill, 2003. 306-340 p. Citado 2 vezes nas páginas 19 e 21.

GABBRIELLI, M.; MARTINI, S. *Programming Languages: Principles and Paradigms*. 1st. ed. [S.l.]: Springer Publishing Company, Incorporated, 2010. ISBN 1848829132, 9781848829138. Citado na página 15.

GOOGLE. *Guia de Desenvolvimento Android*. 2014. Disponível em: <https://developer.android.com/training/index.html>. Citado 2 vezes nas páginas 20 e 35.

GROUP, S. S. I. *Computer History Timeline*. 2007. Disponível em: <http://www.computerhistory.org/siliconengine/timeline/>. Citado na página 15.

HEATH, G. *Single-phase induction motor safety controller*. Google Patents, 1997. US Patent 5,670,858. Disponível em: <https://www.google.com/patents/US5670858>. Citado na página 19.

JIAN N. L. SCHMITZ, D. W. N. T. W. Characteristic induction motor slip values for variable voltage part load performances optimization. In: *IEEE Transactions on Power Apparatus and Systems*. [S.l.: s.n.], 1983. p. 38–46. Citado na página 19.

MEHL, E. L. *Do transistor ao microprocessador*. 2013. Disponível em: <http://www.eletr.ufpr.br/mehl/historia_4.pdf>. Citado na página 15.

MOHAN, N. Improvement in energy efficiency of induction motors by means of voltage control. In: *IEEE Transactions on Power Apparatus and Systems*. [S.l.: s.n.], 1980. p. 1466–1471. Citado na página 19.

NOLA, F. *Power factor control system for ac induction motors*. Google Patents, 1977. US Patent 4,052,648. Disponível em: <https://www.google.com/patents/US4052648>. Citado na página 19.

NOVOTNY D.J. GRITTER, G. S. D. Self-excitation in inverter driven induction machines. In: *IEEE Transactions on Power Apparatus and Systems*. [S.l.: s.n.], 1977. p. 1117–1125. Citado na página 19.

ORACLE. *Object-Oriented Programming Concepts*. 2010. Disponível em: <http://docs.oracle.com/javase/tutorial/java/concepts/index.html>. Citado 2 vezes nas páginas 20 e 35.

PRODUCTION, I. *Strobe-Tachometer*. 2014. Disponível em: <https://play.google.com/store/apps/details?id=com.strobo.stroboscopetachometer>. Citado na página 20.

REID, T. *The Chip: How Two Americans Invented the Microchip and Launched a Revolution*. 1th. ed. [S.l.]: Random House Publishing Group, 2007. 320 p. Citado na página 15.

SCIENCEAPPS. *Stroboscope Tachometer*. 2015. Disponível em: <https://play.google.com/store/apps/details?id=hamidlemon.scientific.apps.stroboskop>. Citado na página 20.

SEN, P. C. *Principles of electric machines and power electronics*. 2th. ed. [S.l.]: John Wiley & Sons, 1996. 207-239 p. Citado 2 vezes nas páginas 19 e 21.

STERN, D. W. N. R. A simplified approach to the determination of induction machine dynamic response. In: *IEEE Transactions on Power Apparatus and Systems*. [S.l.: s.n.], 1978. p. 1430–1439. Citado na página 19.

WEG. *Catalogo Motores Elétricos*. 2005. Disponível em: <http://www.coe.ufrj.br/~richard/Acionamentos/Catalogo%20de%20Motores.pdf>. Citado na página 42.

WEGNER, P. Concepts and paradigms of object-oriented programming. *SIGPLAN OOPS Mess.*, ACM, New York, NY, USA, v. 1, n. 1, p. 7–87, ago. 1990. ISSN 1055-6400. Disponível em: <http://doi.acm.org/10.1145/382192.383004>. Citado 2 vezes nas páginas 20 e 35.

WENGERKIEVICZ, C. A. C. et al. Estimation of Three-Phase Induction Motor Equivalent Circuit Parameters from Manufacturer Catalog Data. *Journal of Microwaves, Optoelectronics and Electromagnetic Applications*, scielo, v. 16, p. 90 – 107, 03 2017. ISSN 2179-1074. Disponível em: <http://www.scielo.br/scielo.php?script=sci_arttext&pid=S2179-10742017000100090&nrm=iso>. Citado na página 19.