



UNIVERSIDADE FEDERAL DO AMAPÁ

DIEGO NASCIMENTO LIMA

JOE IGOR JANSEN SIQUEIRA

**PLATAFORMA DE AQUISIÇÃO DE DADOS E MONITORAMENTO DE
MOTORES ASSÍNCRONOS COM ROTOR BOBINADO - PADM_o.**

Macapá - AP

2017

DIEGO NASCIMENTO LIMA
JOE IGOR JANSEN SIQUEIRA

**PLATAFORMA DE AQUISIÇÃO DE DADOS E MONITORAMENTO DE
MOTORES ASSÍNCRONOS COM ROTOR BOBINADO – PADM_o.**

Trabalho de Conclusão de Curso apresentado à disciplina Trabalho de Conclusão de Curso II do curso de Engenharia Elétrica da Universidade Federal do Amapá, como requisito parcial para obtenção do grau de Bacharel em Engenharia Elétrica.

Área de Concentração:
Instrumentação e Controle.

Orientador:
Prof. Me. Raphael Diego Comesanha e Silva.

Macapá - AP
2017

Dados Internacionais de Catalogação na Publicação (CIP)
Biblioteca Central da Universidade Federal do Amapá

621.3

L732p Lima, Diego Nascimento.

Plataforma de aquisição de dados e monitoramento de motores assíncronos com rotor bobinado - PADMo / Diego Nascimento Lima, Joe Igor Jansen Siqueira; orientador, Raphael Diego Comesanha e Silva. -- Macapá, 2017.

109 p.

Trabalho de conclusão de curso (Graduação) – Fundação Universidade Federal do Amapá, Coordenação do Curso de Engenharia Elétrica.

1. Plataforma de aquisição de dados e monitoramento (PADMo). 2. Sistema supervisorio. 3. Arduino. 4. LabVIEW. 5. Motor de indução. I. Siqueira, Joe Igor Jansen; Silva, Raphael Diego Comesanha, orientador. II. Fundação Universidade Federal do Amapá.

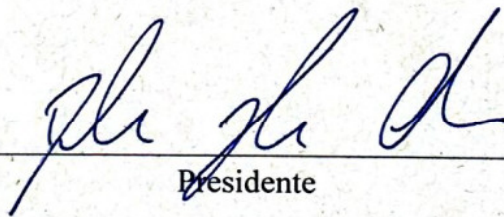


UNIVERSIDADE FEDERAL DO AMAPÁ
DEPARTAMENTO DE CIÊNCIAS EXATAS E TECNOLÓGICAS
COORDENAÇÃO DO CURSO DE ENGENHARIA ELÉTRICA

ATA DE DEFESA DE TCC

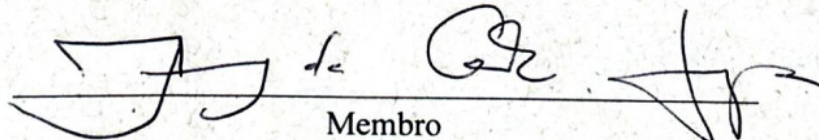
Às 17:40 horas do dia 07 de Julho de 2017 nas dependências da Universidade Federal do Amapá, reuniu-se a Banca Examinadora para defesa de TCC intitulado **PLATAFORMA DE AQUISIÇÃO DE DADOS E MONITORAMENTO DE MOTORES ASSÍNCRONOS COM ROTOR BOBINADO** de autoria do(a)s aluno(a)s **DIEGO NASCIMENTO LIMA** e **JOE IGOR JANSEN SIQUEIRA** regularmente matriculado(a)s no Curso de Engenharia Elétrica desta universidade. A banca Examinadora foi assim constituída: Prof(a). **Raphael Diego Comesanha e Silva**, Presidente da Banca e Orientador(a), Prof(a). **Dr. Geraldo Neves de Albuquerque Maranhão** e Prof(a). **Me. Andrey da Costa Lopes**, como examinadores. Concluída a defesa, foram realizadas as arguições e comentários. Em seguida procedeu-se o julgamento pelos membros da Banca Examinadora, tendo o projeto sido (APROVADO/REPROVADO) APROVADO, com NOTA (0 a 10pts) 9,72. E, para constar, eu, **Raphael Diego Comesanha e Silva**, presidente da Banca Examinadora, lavrei a presente ata que, após lida e achada conforme, foi assinada por mim e demais membros da Banca Examinadora.

Macapá(AP), 07 de JULHO de 2017



Presidente

Raphael Diego Comesanha e Silva
Professor do Magistério Superior
SIAPE: 2093645 - UNIFAP



Membro



Membro

Geraldo Neves de Albuquerque Maranhão
Professor do Magistério Superior
SIAPE: 1824695 - UNIFAP

Membro

Às nossas famílias, razões de nossas existências.

AGRADECIMENTOS

A Deus por ter nos dado saúde e força para superar as dificuldades.

À Universidade Federal do Amapá, seu corpo docente, direção e administração que oportunizaram a janela que hoje vislumbramos um horizonte superior.

Ao nosso orientador professor Me. Raphael Diego Comesanha e Silva, por nos acolher como seus orientandos, pelo suporte, paciência, correções e incentivos durante todo o desenvolvimento do trabalho.

Aos nossos familiares, pelo amor, incentivo e apoio incondicional.

E a todos que direta ou indiretamente fizeram parte da nossa formação, o nosso muito obrigado.

“Nenhuma grande obra é realizada de forma apressada. Realizar uma grande descoberta científica, pintar uma grande tela, escrever um poema imortal, tornar-se um ministro ou um general famoso – realizar qualquer coisa de grandioso requer tempo, paciência e perseverança. Essas coisas são realizadas gradualmente, ‘pouco a pouco’”.

(W. J. Wilmont Buxton)

RESUMO

O presente trabalho tem como intuito a apresentação do projeto e construção da Plataforma de Aquisição de Dados e Monitoramento, PADMo, ferramenta destinada a aquisição e processamento de variáveis de motor de indução trifásico com rotor bobinado em tempo real. A PADMo será aplicada no Laboratório de Conversão de Energia e Máquinas da Universidade Federal do Amapá para auxiliar professores e acadêmicos. As variáveis monitoradas são: velocidade de rotação do eixo, temperatura de operação interna da carcaça do motor, tensão elétrica que alimenta o circuito do estator e corrente elétrica nos circuitos do estator e rotor. A aquisição dos valores destas variáveis será realizada por meio de sensores específicos. A PADMo utiliza a plataforma de prototipagem Arduino para processar os dados oriundos dos sensores, enquanto que o ambiente gráfico de desenvolvimento de sistemas, LabVIEW, tem como finalidade expor os dados através de uma interface gráfica, além de oferecer condições para armazenamento destes dados. Os resultados obtidos com a ferramenta foram satisfatórios, com taxas de erros próximas às faixas de precisão de instrumentos de medição profissionais, credenciando-a para uso laboratorial.

Palavras-Chave: PADMo. Sistema Supervisório. Arduino. LabVIEW. Motor de Indução.

ABSTRACT

The purpose of this work is to present the design and construction of the Plataforma de Aquisição de Dados e Monitoramento (Data Acquisition and Monitoring Platform), PADMo, a tool for real-time acquisition and processing of variables of three-phase wound-rotor induction motor. The PADMo will be applied in the Laboratory of Conversion of Energy and Machines of the Federal University of Amapá to assist professors and academics. The variables monitored are: rotational speed of the shaft, internal operating temperature of the motor housing, electrical voltage supplying the stator circuit and electric current in the stator and rotor circuits. The acquisition of the values of these variables will be performed through specific sensors. PADMo uses the Arduino prototyping platform to process data from the sensors, while the system development graphical environment, LabVIEW, has the purpose of exposing the data through a graphical interface, as well as providing conditions for storing these data. The results obtained with the tool were satisfactory, with error rates close to the precision ranges of professional measuring instruments, accrediting it for laboratory use.

Keywords: PADMo. Supervisory System. Arduino. LabVIEW. Induction Motor.

LISTA DE FIGURAS

Figura 2.1 – Tipos de motores de indução trifásicos.....	18
Figura 2.2 – Esquema simplificado de funcionamento de motores de indução trifásicos.....	21
Figura 3.1 – Sistema de aquisição de dados típico.....	26
Figura 3.2 – Caracterização da PADMo.....	28
Figura 3.3 – Sensor de temperatura LM35.....	29
Figura 3.4 – Sensor LM35 com circuito RC série.....	29
Figura 3.5 – Pinagem do sensor LM35.....	30
Figura 3.6 – Sensor óptico reflexivo TCRT5000.....	31
Figura 3.7 – Esquema de ligação do sensor TCRT5000.....	31
Figura 3.8 – Sensor de corrente ACS712.....	32
Figura 3.9 – Corrente em um condutor sem um campo magnético presente.....	32
Figura 3.10 – Corrente em um condutor com a presença de um campo magnético perpendicular.....	33
Figura 3.11 – Sensor de tensão AC.....	35
Figura 3.12 – Placa Arduino Mega 2560.....	36
Figura 3.13 – Esquema simplificado de conversão e desconversão de dados no Arduino.....	37
Figura 3.14 – Painel Frontal e Diagrama de Blocos do LabVIEW.....	38
Figura 3.15 – Esquema de funcionamento do LINX.....	38
Figura 4.1 – Estrutura para testes de qualificação: temperatura.....	41
Figura 4.2 – Esquema simplificado da estrutura para testes de qualificação: temperatura.....	41
Figura 4.3 – Estrutura para os testes de qualificação: velocidade de rotação.....	45
Figura 4.4 – Esquema simplificado da estrutura para os testes de qualificação: velocidade de rotação.....	46
Figura 4.5 – Estrutura para os testes de qualificação: corrente elétrica.....	51
Figura 4.6 – Esquema simplificado da estrutura para os testes de qualificação: corrente elétrica.....	51
Figura 4.7 – Estrutura para os testes de qualificação: tensão elétrica.....	56
Figura 4.8 – Esquema simplificado da estrutura para os testes de qualificação: tensão elétrica.....	56
Figura 5.1 – Bancada de acionamento de máquinas elétricas.....	64
Figura 5.2 – Motor de indução trifásico com rotor bobinado.....	65
Figura 5.3 – Esquema de ligação triângulo do MIT.....	66

Figura 5.4 – Esquema de ligação estrela do MIT.....	66
Figura 5.5 – Estrutura física principal da PADMo.....	67
Figura 5.6 – Layout de disposição dos componentes da PADMo.....	68
Figura 5.7 – Localização dos sensores de temperatura no MIT.....	69
Figura 5.8 – Estrutura de fixação do sensor de velocidade no MIT.....	70
Figura 5.9 – Layout da interface gráfica da PADMo.....	71
Figura 5.10 – Aba com instruções de funcionamento da PADMo.....	72
Figura 5.11 – Aba com legenda, limites operacionais e resolução da PADMo.....	72
Figura 5.12 – PADMo em funcionamento.....	73
Figura 5.13 – Dados armazenados com cabeçalho em arquivo .xlsx.....	78
Figura 6.1 – Estrutura utilizada para os testes de validação da PADMo.....	79

LISTA DE GRÁFICOS

Gráfico 3.1 – Curva de operação característica do sensor ACS712.....	34
Gráfico 4.1 – Teste de qualificação com variação de temperatura: LM35 x Termopar EM6000.	42
Gráfico 4.2 – Erros no teste de qualificação com variação de temperatura: LM35 x Termopar EM6000.....	43
Gráfico 4.3 – Teste de qualificação sem variação de temperatura: LM35 x Termopar EM6000.	44
Gráfico 4.4 – Erros no teste de qualificação sem variação de temperatura: LM35 x Termopar EM6000.....	44
Gráfico 4.5 – Teste de qualificação com variação de velocidade: TCRT5000 x Tacômetro DT- 2236C.....	48
Gráfico 4.6 – Erros no teste de qualificação com variação de velocidade: TCRT5000 x Tacômetro DT-2236C.....	48
Gráfico 4.7 – Teste de qualificação sem variação de velocidade: TCRT5000 x Multímetro EM6000.....	49
Gráfico 4.8 – Erros no teste de qualificação sem variação de velocidade: TCRT5000 x Multímetro EM6000.....	50
Gráfico 4.9 – Teste de qualificação com variação de corrente (valores médios): ACS712 x Analisador de Qualidade de Energia Fluke 43B.....	53
Gráfico 4.10 – Erros no teste de qualificação com variação de corrente (valores médios): ACS712 x Analisador de Qualidade de Energia Fluke 43B.....	53
Gráfico 4.11 – Teste de qualificação sem variação de corrente (valores médios): ACS712 x Multímetro EM6000.....	54
Gráfico 4.12 – Erros no teste de qualificação sem variação de corrente (valores médios): ACS712 x Multímetro EM6000.....	55
Gráfico 4.13 – Curvas de operação característica dos STAC's.....	57
Gráfico 4.14 – Curvas de operação característica dos STAC's com o divisor de tensão.....	58
Gráfico 4.15 – Curva característica do divisor de tensão.....	59
Gráfico 4.16 – Teste de qualificação com variação de tensão.....	60
Gráfico 4.17 – Erros no teste de qualificação com variação de tensão.....	61
Gráfico 4.18 – Teste de qualificação sem variação de tensão.....	62
Gráfico 4.19 – Erros no teste de qualificação sem variação de tensão.....	62

Gráfico 6.1 – Validação das medições de temperatura da PADMo: STp1 x Termopar EM6000.	80
Gráfico 6.2 – Erros na validação das medições de temperatura da PADMo: STp1 x Termopar EM6000.	80
Gráfico 6.3 – Validação das medições de temperatura da PADMo: STp2 x Termopar EM6000.	81
Gráfico 6.4 – Erros na validação das medições de temperatura da PADMo: STp2 x Termopar EM6000.	81
Gráfico 6.5 – Validação das medições de velocidade da PADMo: SV x Tacômetro DT-2236C.	82
Gráfico 6.6 – Erros na validação das medições de velocidade da PADMo: SV x Tacômetro DT- 2236C.	83
Gráfico 6.7 – Validação da medição de corrente da PADMo: SC1 x EM6000.	84
Gráfico 6.8 – Erros na validação da medição de corrente da PADMo: SC1 x EM6000.	84
Gráfico 6.9 – Validação da medição de corrente da PADMo: SC2 x EM6000.	85
Gráfico 6.10 – Erros na validação da medição de corrente da PADMo: SC2 x EM6000.	85
Gráfico 6.11 – Validação da medição de corrente da PADMo: SC3 x EM6000.	86
Gráfico 6.12 – Erros na validação da medição de corrente da PADMo: SC3 x EM6000.	86
Gráfico 6.13 – Validação da medição de corrente da PADMo: SC4 x EM6000.	87
Gráfico 6.14 – Erros na validação da medição de corrente da PADMo: SC4 x EM6000.	87
Gráfico 6.15 – Validação da medição de tensão da PADMo: ST1 x EM6000.	88
Gráfico 6.16 – Erros na validação da medição de tensão da PADMo: ST1 x EM6000.	89
Gráfico 6.17 – Validação da medição de tensão da PADMo: ST2 x EM6000.	89
Gráfico 6.18 – Erros na validação da medição de tensão da PADMo: ST2 x EM6000.	90
Gráfico 6.19 – Validação da medição de tensão da PADMo: ST3 x EM6000.	90
Gráfico 6.20 – Erros na validação da medição de tensão da PADMo: ST3 x EM6000.	91

LISTA DE ABREVIATURAS E SIGLAS

DDP	Diferença de Potencial
IDE	Integrated Development Environment
LabVIEW	Laboratory Virtual Instrument Engineering Workbench
MIT	Motor de Indução Trifásico
NEMA	National Electrical Manufacturers Association
NI	National Instruments
OOP	Object-Oriented Programming
PADMo	Plataforma de Aquisição de Dados e Monitoramento
PWM	Pulse Width Modulation
RMS	Root Mean Square
RPM	Rotações por Minuto
RPS	Rotações por Segundo
RTC	Real Time Counter
SC1	Sensor de Corrente 1
SC2	Sensor de Corrente 2
SC3	Sensor de Corrente 3
SC4	Sensor de Corrente 4
ST1	Sensor de Tensão 1
ST2	Sensor de Tensão 2
ST3	Sensor de Tensão 3
STAC	Sensor de Tensão AC
STp1	Sensor de Temperatura 1
STp2	Sensor de Temperatura 2
SV	Sensor de Velocidade
UART	Universal Asynchronous Receiver/Transmitter
USB	Universal Serial Bus
VI	Virtual Instrument
VISA	National Instruments Virtual Instrument Software Architecture

SUMÁRIO

1 INTRODUÇÃO	15
1.1 Objetivos.....	16
1.1.1 Objetivo Geral	16
1.1.2 Objetivos Específicos	16
1.2 Estrutura do Trabalho Escrito.....	16
2 MOTORES DE INDUÇÃO TRIFÁSICOS – MIT.....	18
2.1 Rotor Gaiola de Esquilo	19
2.2 Rotor Bobinado.....	19
2.3 Princípio de Funcionamento.....	20
2.4 Variáveis Monitoradas	22
2.4.1 Tensão Elétrica	23
2.4.2 Corrente Elétrica.....	23
2.4.3 Temperatura.....	23
2.4.4 Velocidade	24
3 SISTEMA DE MONITORAMENTO.....	26
3.1 Sistema de Aquisição de Dados	26
3.1.1 Caracterização da PADMo	28
3.2 Sensores	28
3.2.1 LM35	28
3.2.2 TCRT5000	30
3.2.3 ACS712	32
3.2.4 STAC	35
3.3 Processamento de Dados	35
3.3.1 Arduino.....	35
3.3.2 Arduino Mega 2560.....	36
3.4 Interface Gráfica	37
3.4.1 LabVIEW	37
3.4.2 LINX.....	38
3.4.2.1 Comando Customizado.....	39

4	CONDICIONAMENTO DE SINAL E VALIDAÇÃO DE SENSORES.....	40
4.1	Testes para Validação do Sensor LM35	41
4.1.1	Características dos Testes.....	41
4.1.2	Código de Processamento.....	42
4.1.3	Procedimentos e Resultados Obtidos	42
4.1.3.1	<i>Simulação de regime transitório</i>	42
4.1.3.2	<i>Simulação de regime permanente</i>	43
4.2	Testes para Validação do Sensor TCRT5000.....	45
4.2.1	Características dos Testes.....	45
4.2.2	Código de Processamento.....	46
4.2.3	Procedimentos e Resultados Obtidos	47
4.2.3.1	<i>Simulação de fundo de escala</i>	47
4.2.3.2	<i>Simulação de regime permanente</i>	49
4.3	Testes para Validação do Sensor ACS712.....	50
4.3.1	Características dos Testes.....	50
4.3.2	Código de Processamento.....	51
4.3.3	Procedimentos e Resultados Obtidos	52
4.3.3.1	<i>Simulação de fundo de escala</i>	52
4.3.3.2	<i>Simulação de regime permanente</i>	54
4.4	Testes para Validação dos STAC	56
4.4.1	Características dos Testes.....	56
4.4.2	Código de Processamento.....	60
4.4.3	Procedimentos e Resultados Obtidos	60
4.4.3.1	<i>Simulação de fundo de escala</i>	60
4.4.3.2	<i>Simulação de regime permanente</i>	61
4.5	Conclusões dos Testes.....	63
5	PLATAFORMA DE AQUISIÇÃO DE DADOS E MONITORAMENTO – PADMo..	64
5.1	Bancada Didática de Acionamento de Máquinas Elétricas	64
5.2	Estrutura Física	66
5.2.1	Estrutura Principal.....	67
5.2.2	Estrutura dos Sensores de Temperatura.....	69
5.2.3	Estrutura do Sensor de Velocidade.....	69

5.3 Interface Gráfica	70
5.4 Código Final de Processamento.....	73
5.4.1 Programação Arduino.....	74
5.4.2 Programação LabVIEW	76
5.5 Condições para Funcionamento.....	78
6 VALIDAÇÃO DA PADMO.....	79
6.1 Medição da Temperatura Interna da Carcaça do Motor	79
6.1.1 Resultados do STp1	80
6.1.2 Resultados do STp2	81
6.2 Medição da Velocidade de Rotação do Eixo do Motor	82
6.3 Medição da Corrente Elétrica nos Enrolamentos do Motor	83
6.3.1 Resultados do SC1.....	84
6.3.2 Resultados do SC2.....	85
6.3.3 Resultados do SC3.....	86
6.3.4 Resultados do SC4.....	87
6.4 Medição da Tensão RMS de Alimentação do Motor.....	88
6.4.1 Resultados do ST1	88
6.4.2 Resultados do ST2	89
6.4.3 Resultados do ST3	90
6.5 Resumo dos Testes.....	91
7 CONSIDERAÇÕES FINAIS E SUGESTÕES	93
REFERÊNCIAS BIBLIOGRÁFICAS	95
APÊNDICE A – Características dos Instrumentos de Medição usados na PADMo.	99
APÊNDICE B – Códigos de Programação do Arduino para Testes de Validação dos Sensores.	100
APÊNDICE C – Código de Processamento do Arduino para a PADMo.	103
APÊNDICE D – Código de Processamento do LabVIEW para a PADMo.	108

1 INTRODUÇÃO

O motor de indução trifásico é o modelo mais utilizado no âmbito industrial, principalmente por ser simples, eficiente e robusto, oferecendo assim, um alto grau de confiabilidade, além de sua simplicidade quanto aos aspectos construtivos e operacionais (BHOWMIK; PRADHAN; PRAKASH, 2013; CARVALHO, 2011; CHAPMAN, 2013; KOSOW, 2005). Ciente de que essas máquinas são fundamentais, é imprescindível que elas operem em conformidade com o que foi designado como sendo seu comportamento padrão, minimizando assim, quedas no rendimento produtivo ou paradas completas dos processos em que elas estão inseridas, que culminariam em perdas financeiras.

Apesar das características positivas que tornam o seu uso atrativo, Bhowmik, Pradhan e Prakash (2013, p. 1, tradução nossa) afirmam que “[...] como qualquer outra máquina, eles [motores de indução] são vulneráveis a faltas, que se deixadas sem monitoramento, podem levar a falhas catastróficas da máquina a longo prazo”. Logo, no cenário industrial atual, não se pode esperar que equipamentos tão relevantes apresentem problemas imprevistos, por isso, parte dos recursos destinados aos programas de manutenção são direcionados à implantação de sistemas de monitoramento das condições dos equipamentos mais importantes, proporcionando assim, condições para que as empresas possam elaborar cronogramas de manutenção (TAVNER *et al.*, 2008).

Segundo Tavner *et al.* (2008, p. 27, tradução nossa) o monitoramento de condições de um equipamento significa “[...] a contínua avaliação da saúde da planta e equipamentos durante toda sua vida útil”. Os mesmos autores ainda dizem que “[...] a real função [do monitoramento de condições] deve sempre ser a de tentar reconhecer o desenvolvimento de faltas em um estado inicial”. Portanto, se o objetivo do monitoramento de condições é detectar sinais de que um possível problema está se formando em uma planta, processo ou máquina, e dada a relevância dos motores de indução, os possíveis benefícios provenientes da implementação de técnicas de monitoramento do maquinário de uma empresa tornam-se plausíveis.

Com o conhecimento das condições em que os motores presentes em uma planta se encontram, é possível tomar decisões que garantirão um grau de confiabilidade elevado das máquinas. Entretanto, para se implementar técnicas de monitoramento é necessário o desenvolvimento ou aquisição de uma ferramenta que faça a captação de variáveis físicas essenciais para análise do estado dessas máquinas.

Deste modo, neste trabalho, busca-se, justamente, o desenvolvimento de uma ferramenta capaz de realizar a captação e processamento destas variáveis, em tempo real, e tem

como denominação Plataforma de Aquisição de Dados e Monitoramento (PADMo) e estará alocada no Laboratório de Conversão de Energia e Máquinas da Universidade Federal do Amapá.

1.1 Objetivos

1.1.1 Objetivo Geral

Desenvolver uma ferramenta que realizará a aquisição, processamento e apresentação de variáveis de motor de indução trifásico de rotor bobinado em tempo real.

1.1.2 Objetivos Específicos

- Determinar as variáveis do motor a serem monitoradas.
- Definir os sensores que realizarão a medição das variáveis de interesse.
- Realizar comunicação entre os sensores e a plataforma Arduino.
- Efetuar comunicação entre a plataforma Arduino e software LabVIEW.
- Elaborar estrutura para abrigar o hardware necessário para o projeto.
- Desenvolver interface gráfica no software LabVIEW.
- Validar resultados obtidos pela ferramenta desenvolvida.

1.2 Estrutura do Trabalho Escrito

Este trabalho está estruturado em 7 capítulos e complementado por 4 apêndices. O Capítulo 1 refere-se à introdução do trabalho, ao objetivo e também à estrutura geral que apresenta.

O Capítulo 2 aborda o princípio de funcionamento dos motores de indução trifásico, sua estrutura física, as características das variáveis corrente e tensão elétricas, velocidade de rotação e temperatura.

O Capítulo 3 trata da conceituação de sistema de aquisição e processamento de dados, de interface gráfica, além dos sensores utilizados no projeto. Neste capítulo se discorre sobre a caracterização da PADMo, Arduino e LabVIEW.

No Capítulo 4, os testes para a qualificação dos sensores e o processamento de seus dados pelo Arduino é discutido. Aborda-se a estrutura, a programação utilizada e os resultados dos testes.

O Capítulo 5 explana sobre a PADMo. A bancada didática de acionamento de máquinas elétricas, onde as medições são realizadas; sua estrutura física, interface gráfica, código de processamento final, tanto do Arduino quanto do LabVIEW e as condições necessárias para que a PADMo opere em outros computadores.

No Capítulo 6, encontra-se os testes para validação da PADMo.

O Capítulo 7 apresenta as considerações finais e sugestões para trabalhos futuros.

O Apêndice A exhibe as especificações técnicas dos instrumentos utilizados nos testes de qualificação dos sensores e Arduino e validação da PADMo.

Os Apêndices B e C tratam dos códigos de programação utilizados nos testes de qualificação dos sensores e Arduino e validação da PADMo, respectivamente.

O Apêndice D expõe a programação desenvolvida para o LabVIEW utilizada nos testes de validação da PADMo.

2 MOTORES DE INDUÇÃO TRIFÁSICOS – MIT

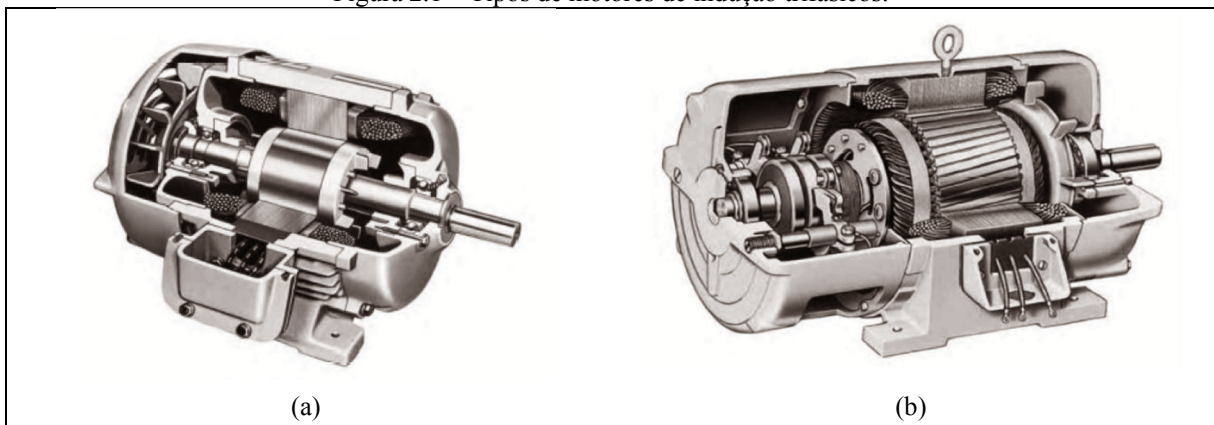
O motor de indução trifásico pode ser definido como um conversor eletromecânico de energia, baseado em princípios eletromagnéticos, capaz de realizar a conversão de energia elétrica em energia mecânica (BORTONI; SANTOS, 2006; KOSOW, 2005; REZEK, 2011). Para Chapman (2013, p. 307) “essas máquinas são denominadas de máquinas de indução porque a tensão [...] é induzida nos enrolamentos do rotor em vez de ser fornecida por meio de uma conexão física de fios”.

As máquinas de indução são comumente caracterizadas por duas partes principais (FRANCISCO, 2006):

- a) Estator: parte fixa da máquina, composto por um pacote de lâminas de aço que estão dispostas no formato de coroa circular ranhurada e são revestidas por meio de um verniz isolante. Pode ser seccionado em três partes: carcaça, núcleo de chapas e enrolamentos;
- b) Rotor: parte móvel da máquina, constituído por um núcleo ferromagnético laminado. É apoiado no eixo de rotação do motor que possui rolamentos nos extremos e que transmite à carga a energia mecânica produzida.

Os MIT's são classificados em duas categorias que se diferenciam pela forma como seus rotores são construídos: os de rotor gaiola de esquilo e os de rotor bobinado, apresentados na Figura 2.1(a) e 2.1(b), respectivamente (CHAPMAN, 2013; FITZGERALD; KINGSLEY; UMANS, 2006; KOSOW, 2005).

Figura 2.1 – Tipos de motores de indução trifásicos.



(a) Rotor gaiola de esquilo. (b) Rotor bobinado.

Fonte: Chapman (2013).

2.1 Rotor Gaiola de Esquilo

O rotor gaiola de esquilo “[...] consiste em barras condutoras encaixadas em ranhuras no ferro do rotor e curto-circuitadas em cada lado por anéis condutores” (FITZGERALD; KINGSLEY; UMANS, 2006, p. 295). Destaca-se como o mais utilizado em todas as áreas de aplicações pelo seu baixo custo, robustez, vida útil longa, ausência de escovas e baixa necessidade de manutenção (ELETROBRAS *et. al.*, 2009).

As principais vantagens e desvantagens do MIT gaiola de esquilo estão dispostas na Tabela 2.1.

Tabela 2.1 – Vantagens e desvantagens do rotor gaiola de esquilo

Vantagens	Desvantagens
Baixo custo de aquisição	Necessidade de inversor eletrônico ou outros dispositivos especiais para controle de velocidade
Bom conjugado na partida	Alta corrente de partida
Baixo custo de manutenção	Baixo Fator de potência, quando não for bem dimensionado para a carga.
Velocidade constante	

Fonte: Eletrobrás *et. al.* (2009).

2.2 Rotor Bobinado

O rotor bobinado “[...] é construído na forma de um enrolamento polifásico semelhante ao estator [...]. Os terminais do enrolamento do rotor são conectados a anéis deslizantes isolados montados sobre o eixo”. (FITZGERALD; KINGSLEY; UMANS, 2006, p. 295).

De acordo com Carvalho (2011), o rotor bobinado tem como finalidade permitir a inserção de resistências em série com o enrolamento do rotor, a fim de diminuir a corrente de partida da máquina, ocasionando uma partida mais suave e dando a possibilidade de efetuar um melhor controle sobre a sua velocidade.

Eletrobrás *et.al.* (2009) e WEG (2015) explicam que MIT's de rotor bobinado são aplicados em potências muito elevadas, geralmente superiores a 05 CV, pois possibilitam partir cargas com conjugado linear ou com elevado momento de inércia com uma pequena corrente de partida – cerca de uma vez e meia o valor da corrente nominal, sendo recomendados nos casos onde há a necessidade de partidas à plena carga. A Tabela 2.2 mostra algumas vantagens e desvantagens do MIT de rotor bobinado.

Tabela 2.2 – Vantagens e desvantagens do rotor bobinado

Vantagens	Desvantagens
Controle de Velocidade	Necessidade de reostato de grande potência
Conjugado alto na partida	Exige mais manutenção do que os MIT's com
Corrente de partida baixa	rotor gaiola de esquilo

Fonte: Eletrobrás *et. al.* (2009).

2.3 Princípio de Funcionamento

Quando há tensão elétrica trifásica aplicada ao circuito do estator ocorrerá neste a circulação de correntes elétricas, todas de mesma intensidade e defasadas de 120° entre si, que por sua vez, criarão um campo magnético girante de intensidade constante (CHAPMAN, 2013). O campo magnético girante, ao atravessar a superfície do rotor, provoca neste uma variação de fluxo em seus condutores, gerando, de acordo com a lei de Faraday, Equação 2.1, uma força eletromotriz induzida (f.e.m.) nos mesmos. Esta f.e.m. induz correntes elétricas nos condutores do rotor, que por sua vez, obedecendo aos princípios da lei de Lenz, produzem outro campo magnético com tendência a se opor ao campo magnético girante presente no estator (CARVALHO, 2011; CHAPMAN, 2013; FITZGERALD; KINGSLEY; UMANS, 2006; KOSOW, 2005).

$$e_{\text{ind}} = -N \frac{d\Phi}{dt} \quad (2.1)$$

Em que

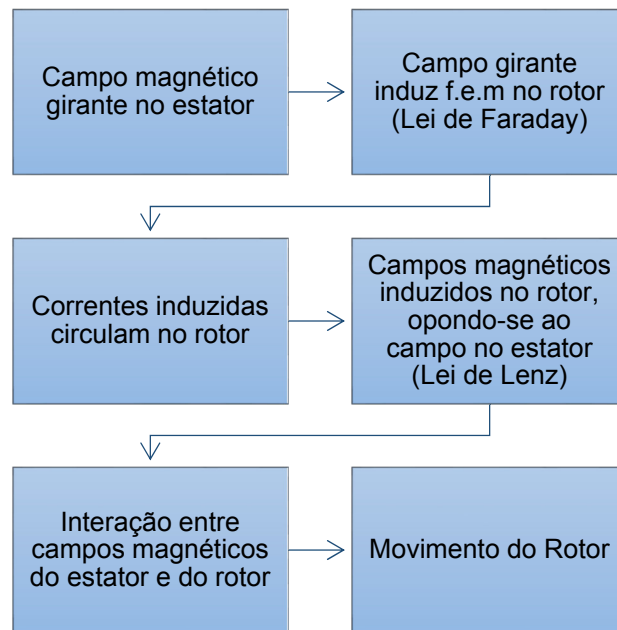
e_{ind} – É a tensão induzida na bobina, V;

N – É o número de espiras de fio da bobina;

Φ – É o fluxo que passa através da bobina.

A interação entre os campos magnéticos presentes no estator e no rotor criará um conjugado que fará com que o rotor gire buscando alinhar os campos magnéticos, assim ocorrerá a conversão de energia elétrica em energia mecânica (CARVALHO, 2011; CHAPMAN, 2013; FITZGERALD; KINGSLEY; UMANS, 2006; KOSOW, 2005). A Figura 2.2 ilustra, de forma simplificada, o princípio de funcionamento de um MIT.

Figura 2.2 – Esquema simplificado de funcionamento de motores de indução trifásicos.



Fonte: Autoria Própria.

Em Chapman (2013) é explicado que o rotor de um MIT não pode se movimentar na mesma velocidade do campo magnético girante, ou velocidade síncrona, caso contrário, as barras ou enrolamentos do rotor estariam estacionárias em relação ao campo magnético e não se teria tensão induzida, consequentemente, não haveria rotação. Em outras palavras, o movimento relativo entre o rotor e o campo magnético do estator é o que produz uma tensão induzida nos condutores do rotor, portanto, um MIT pode ganhar velocidade até próximo a velocidade síncrona, sem nunca a alcançar exatamente.

A velocidade síncrona (N_{sinc}), em RPM, pode ser calculada conhecendo-se o número de polos da máquina (P) e a frequência de linha (f), conforme Equação 2.2 (CHAPMAN, 2013).

$$N_{sinc} = \frac{120f}{P} \quad (2.2)$$

Um MIT é conhecido também como motor assíncrono, justamente pelo fato do eixo de seu rotor não poder girar na velocidade síncrona. Essa diferença entre as velocidades síncrona e do eixo do rotor do MIT é conhecida como velocidade de escorregamento. A velocidade de escorregamento pode ser obtida a partir da Equação 2.3 (CHAPMAN, 2013).

$$N_{esc} = N_{sinc} - N_m \quad (2.3)$$

Em que:

N_{esc} – É a velocidade de escorregamento da máquina, RPM;

N_{sinc} – É a velocidade dos campos magnéticos, RPM;

N_m – É a velocidade mecânica do eixo do motor, RPM.

A velocidade relativa expressa em uma base por unidade ou porcentagem é definida como escorregamento e é apresentado na Equação 2.4 (CHAPMAN, 2013). Quanto menor seu valor, mais próximas serão as velocidades do rotor e do campo magnético girante.

$$S = \frac{N_{sinc} - N_m}{N_{sinc}} \times 100\% \quad (2.4)$$

Onde:

S – É o escorregamento do motor.

2.4 Variáveis Monitoradas

As variáveis do motor de indução que serão monitoradas são:

- A tensão e corrente elétricas no circuito do estator e no rotor;
- A temperatura interna da carcaça do motor;
- A velocidade de rotação do eixo do motor.

Cada motor apresenta valores típicos dessas variáveis quando está funcionando corretamente, portanto, a presença de oscilações ou distúrbios nesses valores pode servir como indicativo de que algum problema está ocorrendo no motor e levar ao diagnóstico antecipado de faltas ou falhas. Como exemplo desses problemas podemos citar: sobrecarga, rotor travado, subtensão e sobretensão, falhas no enrolamento do estator, elevação de temperatura, entre outros (HAMMO, 2014; KARMAKAR *et al.*, 2016; MEHALA, 2010; SILVA, 2006; WEG, 2012).

2.4.1 Tensão Elétrica

A tensão elétrica que alimenta um MIT geralmente é proveniente da rede de distribuição da concessionária de energia elétrica local, seguindo valores padronizados por normas, tais como: 127 V, 220 V, 380 V, 440 V. Comumente, os MIT's podem ser alimentados por pelo menos dois valores diferentes de tensão elétrica, bastando apenas a mudança no fechamento de seus bobinados de estrela para triângulo ou vice-versa. MAMEDE FILHO (2007, p. 230) explica que “os motores devem trabalhar dentro de limites de desempenho satisfatório para uma variação de tensão de $\pm 10\%$ da sua tensão nominal, desde que a frequência não varie”.

2.4.2 Corrente Elétrica

A corrente solicitada da rede de alimentação pelo motor, trabalhando à plena carga, com frequência e tensão nominais é dada pela Equação 2.5 (MAMEDE FILHO, 2007).

$$I_{nm} = \frac{736 \times P_{nm}}{\sqrt{3} \times V \times \eta \times \cos\psi} \quad (A) \quad (2.5)$$

Onde:

P_{nm} – Potência nominal do motor, CV;

V – Tensão nominal trifásica, V;

η – Rendimento do motor;

$\cos\psi$ – Fator de potência sob carga nominal.

Durante a partida dos motores de indução deve-se tomar cuidado com a corrente consumida, pois neste momento o motor solicita da rede de alimentação uma corrente de 06 a 10 vezes o valor nominal. Como consequência, caso nenhuma medida seja tomada para suavizar esta elevação, pode ocorrer queda de tensão do sistema em que se encontra o motor, causando distúrbios em outros processos e máquinas, além da ativação desnecessária de equipamentos de comando e proteção. (MAMEDE FILHO, 2007).

2.4.3 Temperatura

Para Kosow (2005) e Mamede Filho (2007) a temperatura é uma variável que afeta tanto a vida útil quanto a capacidade de uma máquina elétrica. Segundo Tavner *et. al.* (2008) os limites de capacidade de máquinas elétricas são geralmente definidos pela máxima temperatura

permitida que a isolação pode suportar. Mamede Filho (2007, p. 231) ainda afirma que “a vida útil de um motor está intimamente ligada ao aquecimento das bobinas dos enrolamentos, que não deve ultrapassar os limites previstos na fabricação”. O mesmo autor diz que o “aquecimento [...] provoca o envelhecimento gradual e generalizado do isolamento [...]”.

Conforme Kosow (2005, p. 487) “a margem de elevação de temperatura permissível, em máquinas elétricas de fabricação usual, é de 40 °C acima da temperatura ambiente”. Ele também explica que a vida útil do enrolamento de um motor é reduzida à metade para cada acréscimo de 10 °C na temperatura limite de funcionamento do motor. Inversamente, a vida útil dobra a cada decréscimo de 10 °C na temperatura.

De acordo com Tavner *et al.* (2008, p. 127, tradução nossa) existem três abordagens básicas para o monitoramento de temperatura:

- Medir temperaturas locais em pontos na máquina usando detectores de temperatura embarcados;
- Usar uma imagem térmica, alimentada com variáveis adequadas, para monitorar a temperatura do que é percebido como o ponto mais quente na máquina;
- Medir temperaturas distribuídas na máquina ou a maior parte das temperaturas dos fluidos de refrigeração.

Fica esclarecido em Tavner *et al.* (2008) que uma das dificuldades envolvendo o monitoramento de temperatura é justamente decidir qual dessas abordagens seguir.

2.4.4 Velocidade

Os motores são especificados para a velocidade na qual entregarão sua potência nominal de saída, quando alimentados à tensão nominal. Em geral, conforme a velocidade decresce, a potência também decresce proporcionalmente. Uma velocidade reduzida produzirá má ventilação e superaquecimento (KOSOW, 2005).

A NEMA, National Electrical Manufacturers Association, desenvolveu um sistema de classificação baseado nas características de velocidades dos motores:

- Motor de velocidade constante;
- Motor de velocidade variável;
- Motor de velocidade ajustável;
- Motor de velocidade variável e ajustável.

Os motores com rotor gaiola de esquilo se encaixam na classificação de velocidade constante, enquanto que os de rotor bobinado na de velocidade ajustável e variável (KOSOW, 2005).

Segundo Chapman (2013), os motores de indução passaram a ser opções viáveis em aplicações que requeriam o controle de velocidade após o advento dos acionamentos modernos de estado sólido. Ainda conforme Chapman (2013), há duas técnicas que podem ser usadas para controle de velocidade em motores de indução. Uma consiste em variar a velocidade síncrona e outra em variar o escorregamento do motor para uma carga.

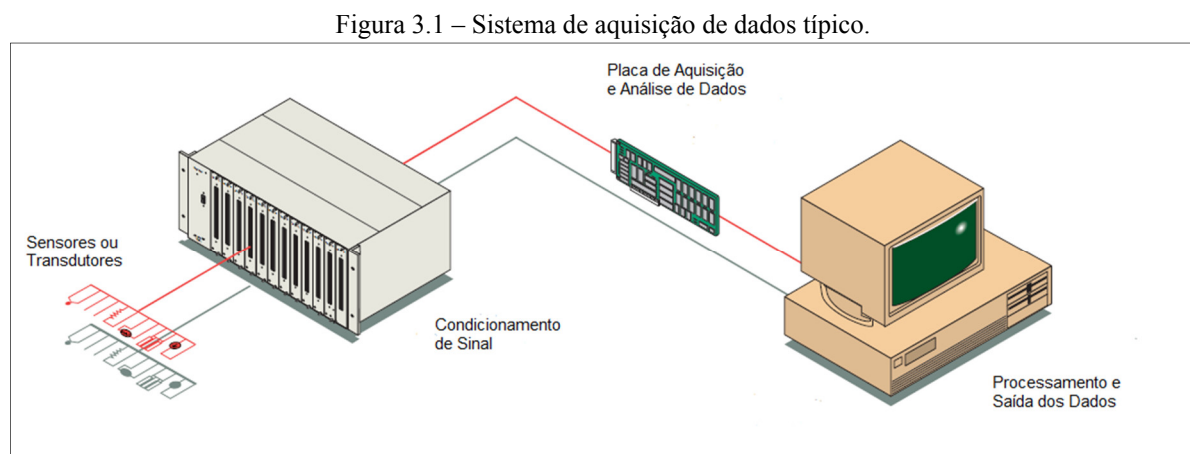
3 SISTEMA DE MONITORAMENTO

3.1 Sistema de Aquisição de Dados

A ferramenta proposta neste trabalho pode ser considerada como um sistema de aquisição de dados, uma vez que:

Aquisição de dados é o processo pelo qual fenômenos físicos do mundo real são transformados em sinais elétricos que são medidos e convertidos em formato digital para processamento, análise, e armazenamento por um computador. (PARK; MACKAY, 2003, p. 1, tradução nossa).

Um sistema de aquisição de dados típico pode ser composto conforme mostrado na Figura 3.1 (NATIONAL INSTRUMENTS CORPORATION, 1999; PARK; MACKAY, 2003).



Fonte: Adaptado de National Instruments Corporation (1999).

Os sensores e transdutores convertem os fenômenos físicos que se desejam medir em sinais elétricos aceitáveis pelo condicionamento de sinal ou mesmo pela placa de aquisição e análise de dados. Pode-se assumir que eles são os responsáveis por criar uma interface entre o mundo real e o sistema de aquisição de dados (NATIONAL INSTRUMENTS CORPORATION, 1999; PARK; MACKAY, 2003; WILSON, 2005).

Nem sempre os sinais entregues pelos sensores e atuadores apresentam características ideais para serem usados pelo sistema de aquisição de dados, assim sendo, por vezes é necessário que esses sinais passem por alguma forma de condicionamento. O condicionamento desses sinais pode ser, por exemplo, a filtração de ruídos que comprometem suas características; a amplificação da magnitude dos sinais; isolamento dos sensores e transdutores do computador, ou outras partes do sistema, devido a presença de tensões elétricas elevadas, além de outras

técnicas de condicionamento (NATIONAL INSTRUMENTS CORPORATION, 1999; PARK; MACKAY, 2003; WILSON, 2005).

Após o condicionamento, os sinais dos sensores são enviados para a placa de aquisição e análise de dados. Para Park e Mackay (2003) uma placa de aquisição pode realizar qualquer uma das seguintes funções:

- A entrada, processamento e conversão para formato digital, usando conversores A/D (analógico para digital), de sinais analógicos medidos de um sistema ou processo. Os dados são então transferidos para um computador para exibição, armazenamento e análise;
- A entrada de sinais digitais, que contêm informações de um sistema ou processo;
- O processamento e conversão para formato analógico, usando conversores D/A (digital para analógico), de sinais digitais de um computador. Os sinais de controle analógico são usados para controlar um sistema ou processo;
- A saída de sinais de controle digitais.

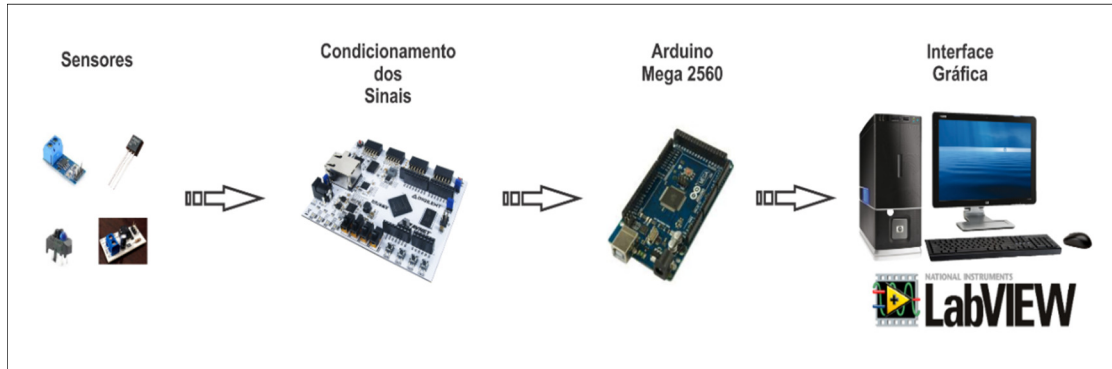
Um sistema de aquisição de dados se torna completo com a presença de um software de aplicação, que pode ser um painel de interação de tela cheia, um programa de controle de entrada/saída dedicado, um data logger, um manipulador de comunicação ou uma combinação de todos estes. É necessário também que o computador que esteja armazenando o software possua boas características, caso contrário o mesmo pode afetar a velocidade com que os dados podem ser continuamente e precisamente adquiridos, processados e armazenados (NATIONAL INSTRUMENTS CORPORATION, 1999; PARK; MACKAY, 2003).

Dependendo da aplicação ao qual o sistema de aquisição de dados esteja destinado é possível que haja a presença de elementos destinados a ações de controle (KEITHLEY INSTRUMENTS, 2001; PARK; MACKAY, 2003), porém este não é o caso da ferramenta em questão. Suas funções vão até ao ponto de apresentarem as variáveis do MIT ao usuário.

3.1.1 Caracterização da PADMo

O conceito da ferramenta desenvolvida neste trabalho consiste dos elementos apresentados na Figura 3.2.

Figura 3.2 – Caracterização da PADMo.



Fonte: Autoria Própria.

Os sensores utilizados para a aquisição das variáveis físicas do motor de indução com rotor bobinado, descritas no tópico 2.4, são abordados no tópico 3.2 a seguir e no Capítulo 4. O condicionamento de sinais é tratado no Capítulo 6. No entanto, não é necessário condicionar os sinais de todos os sensores, pois alguns destes já apresentam saídas adequadas ao Arduino.

O Arduino Mega 2560 fica a cargo de realizar as funções da placa de aquisição e análise de dados, ou seja, ele adquire, processa e converte para formato digital os sinais analógicos medidos pelos sensores. Os dados são, então, transferidos para um computador para exibição, armazenamento e análise.

O LabVIEW é o encarregado de receber os dados do Arduino Mega 2560 e apresentá-los ao usuário final por meio de uma interface gráfica, além de prover as condições para o armazenamento desses dados no computador.

3.2 Sensores

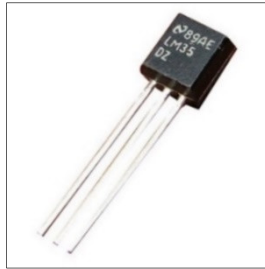
3.2.1 LM35

O LM35, apresentado na Figura 3.3, é um sensor semiconductor de temperatura transistorizado de circuito integrado de precisão, no qual sua saída é na forma de tensão e linearmente proporcional à temperatura em graus Celsius (TEXAS INSTRUMENTS INCORPORATED, 2016). Algumas de suas principais características são:

- Calibrado diretamente em Celsius (°C);
- Fator de escala linear +10 mV/°C;

- Precisão garantida de 0.5 °C (a 25 °C);
- Classificado para a faixa de – 55 °C a 150 °C;
- Opera com tensões entre 04 V a 30 V;
- Não-linearidade típica de $\pm 1/4$ °C.

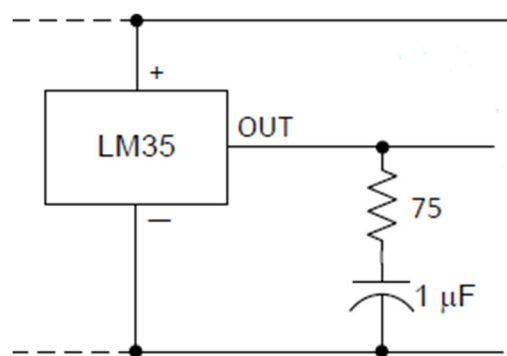
Figura 3.3 – Sensor de temperatura LM35.



Fonte: blog.vidadesilicio.com.br.

O LM35 pode ser colado ou cimentado à superfície que se deseja medir a temperatura. Para esta situação a temperatura medida idealmente terá uma diferença de 0.01 °C para a temperatura na superfície. Contudo, a performance do LM35 pode ser afetada quando este está conectado a cargas com alta capacitância. Para minimizar os efeitos oriundos dessas fontes recomenda-se a inserção de um resistor em série com um capacitor para atuar como filtro passa-baixa, com valores típicos de 75 Ω e 0,2 ou 01 μF , respectivamente, entre a saída (Vout) e o aterramento (GND), conforme Figura 3.4 (TEXAS INSTRUMENTS INCORPORATED, 2016).

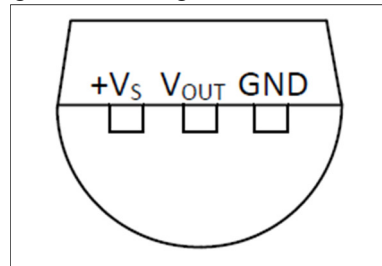
Figura 3.4 – Sensor LM35 com circuito RC série.



Fonte: Adaptado de Texas Instruments Incorporated (2016).

A configuração dos pinos do LM35, visto por baixo, é retratada na Figura 3.5.

Figura 3.5 – Pinagem do sensor LM35.



Sensor visto por baixo.

Fonte: Texas Instruments Incorporated (2016).

Onde:

$+V_S$ – Pino de alimentação positivo;

V_{out} – Saída analógica do sensor;

GND – Pino terra.

As especificações de precisão do LM35 são obtidas através da função de transferência linear apresentada na Equação 3.1 (TEXAS INSTRUMENTS INCORPORATED, 2016).

$$V_{out} = 10 \text{ mV} \times T \quad (3.1)$$

Onde:

V_{out} – é a saída do LM35 em volts;

T – é a temperatura em °C.

Manipulando a Equação 3.1 podemos obter a Equação 3.2, que nos permite calcular a temperatura em função da tensão do sensor.

$$T = V_{out} \times 100 \quad (3.2)$$

3.2.2 TCRT5000

O TCRT5000, Figura 3.6, é um sensor reflexivo, que inclui um emissor infravermelho e um fototransistor em um pacote com chumbo que bloqueia a luz visível. Pode ser usado como: sensor de posição para encoder de eixo; detector de material reflexível como papel, fitas magnéticas, etc. além de outras possibilidades (VISHAY INTERTECHNOLOGY, 2009). Sua saída pode assumir dois valores, 0 (low), quando o fototransistor detecta o feixe de luz

infravermelha emitido pelo emissor infravermelho, e 1 (high), quando o fototransistor não detecta o feixe de luz.

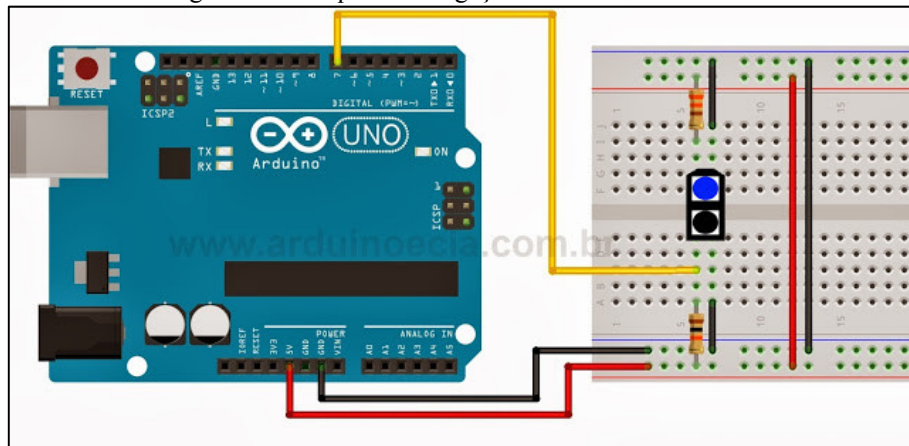
Figura 3.6 – Sensor óptico reflexivo TCRT5000.



Fonte: www.yourduino.com.

A ligação pode ser realizada conforme mostrado na Figura 3.7, onde, o resistor ligado ao receptor (elemento preto do sensor) é 10 k Ω e o ligado ao emissor (elemento azul) é de 330 Ω .

Figura 3.7 – Esquema de ligação do sensor TCRT5000.



Fonte: www.arduinoocia.com.br.

Entre suas características, destacam-se:

- Distância de operação máxima: 2,5 mm;
- Corrente de saída típica em teste $I_c = 01 \text{ mA}$;
- Filtro bloqueador de luz do dia;
- Comprimento de onda do emissor: 950 nm.

3.2.3 ACS712

O ACS712 é um sensor destinado à medição de corrente alternada (AC) ou contínua (DC) em aplicações industriais, comerciais e sistemas de comunicação. Aplicações típicas incluem o controle de motores, detecção de cargas, fontes de alimentação chaveadas e proteção contra sobrecorrentes (ALLEGRO MICROSYSTEMS, 2012). O sensor é ilustrado na Figura 3.8.

Figura 3.8 – Sensor de corrente ACS712.

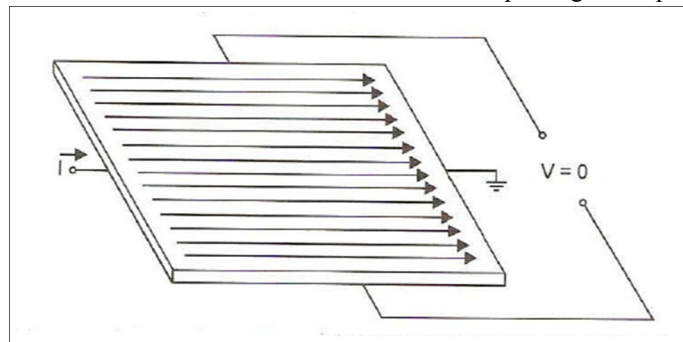


Fonte: www.filipeflop.com.

O sensor consiste de um circuito Hall linear preciso com um caminho de condução de cobre. A corrente que circula através desse caminho de cobre gera um campo magnético que o circuito Hall converte em uma tensão proporcional. Quanto maior a proximidade do sensor ao campo magnético, melhor será a precisão do dispositivo (ALLEGRO MICROSYSTEMS, 2012).

“O efeito Hall caracteriza-se basicamente pelo aparecimento de um campo elétrico transversal em um condutor percorrido por uma corrente elétrica, quando ele se encontra mergulhado em um campo magnético” (THOMAZINI; ALBUQUERQUE, 2010, p. 174). Na Figura 3.9 um filme de material semiconductor é percorrido por uma corrente elétrica constante. A distribuição de corrente nele é uniforme e não existe diferença de potencial na saída (THOMAZINI; ALBUQUERQUE, 2010).

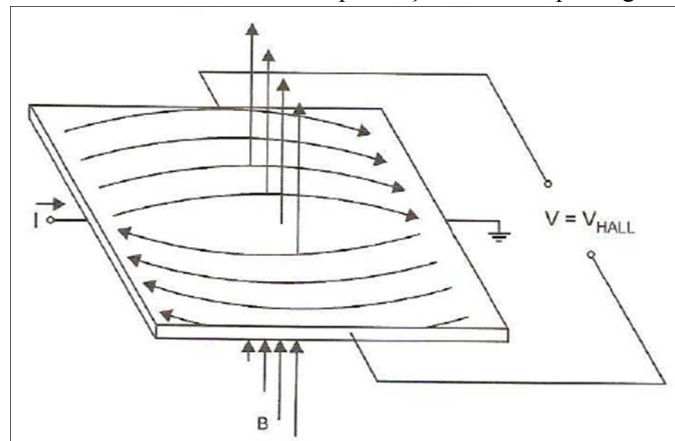
Figura 3.9 – Corrente em um condutor sem um campo magnético presente.



Fonte: Thomazini; Albuquerque (2010).

Na presença de um campo magnético perpendicular, Figura 3.10, o fluxo de corrente é distorcido. A distribuição resultante provoca o aparecimento de uma diferença de potencial (ddp) entre os terminais de saída. Essa ddp chama-se tensão de Hall (THOMAZINI; ALBUQUERQUE, 2010).

Figura 3.10 – Corrente em um condutor com a presença de um campo magnético perpendicular.



Fonte: Thomazini; Albuquerque (2010).

Pode-se destacar as seguintes características do ACS712:

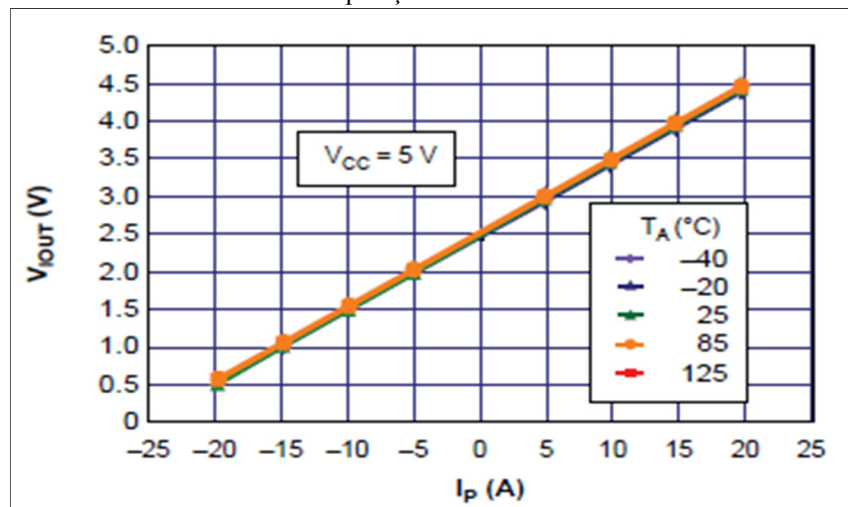
- Largura de banda: 80 kHz;
- Erro total de saída: 1,5% a 25 °C;
- Tempo de subida da saída para sinal de entrada em degrau: 05 μ s;
- Sensibilidade de saída: 100 mV/A;
- Tensão de saída proporcional a corrente DC e AC;
- Faixa de medição: - 20 a 20 A;
- Tensão de alimentação: 4,5 a 5,5 V;
- Corrente de alimentação (para 05 V): 10 a 13 mA.

No caso de medições de ondas senoidais o sensor irá medir os valores dos semiciclos positivo e negativo de forma independente, porém como se deseja a medição dos valores de corrente RMS (root mean square) é necessário realizar um tratamento dos dados provenientes dos sensores ACS712. Esse tratamento é efetuado através da aplicação de média quadrática, Equação 3.3, de um número de amostras das medições do sensor. A média quadrática será implementada através do código de programação no Arduino, conforme é discutido no tópico 4.3.

$$x_q = \sqrt{\frac{x_1^2 + x_2^2 + \dots + x_n^2}{n}} \quad (3.3)$$

O Gráfico 3.1 apresenta os valores de tensão de saída do sensor em função da corrente detectada.

Gráfico 3.1 – Curva de operação característica do sensor ACS712.



Fonte: Allegro Microsystems (2012).

Do Gráfico 3.1 pode-se obter a Equação 3.4 que descreve o comportamento do sensor.

$$V_{out} = 0,1 \times I_p + 2,5 \quad (V) \quad (3.4)$$

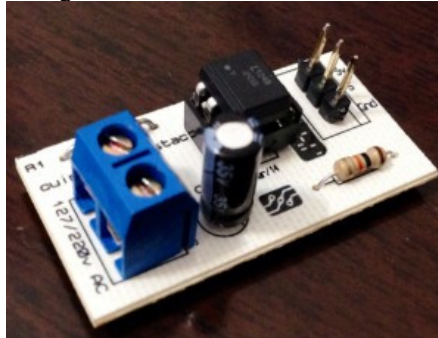
E da Equação 3.4 obtém-se a Equação 3.5, de onde pode-se obter o valor de corrente tendo a tensão de saída do sensor.

$$I_p = \frac{V_{out} - 2,5}{0,1} = (V_{out} - 2,5) \times 10 \quad (A) \quad (3.5)$$

3.2.4 STAC

A medição de tensão elétrica RMS é efetuada pelo sensor apresentado na Figura 3.11. O sensor trabalha com tensões nas faixas de 127 V e 220 V e utiliza um optoacoplador para isolamento da rede AC do sinal DC que é enviado para o microcontrolador. Sua saída varia entre 0 V e 05 V, proporcional ao valor de tensão de entrada (CUIN, 2015).

Figura 3.11 – Sensor de tensão AC.



Fonte: www.cuin.com.br.

O STAC é um sensor artesanal, não possui uma ficha técnica de dados para consulta, portanto, foi necessário verificar o seu comportamento para, então, aplicá-lo no projeto. Todos os dados obtidos por meio de testes estão descritos no tópico 4.4 deste trabalho.

3.3 Processamento de Dados

3.3.1 Arduino

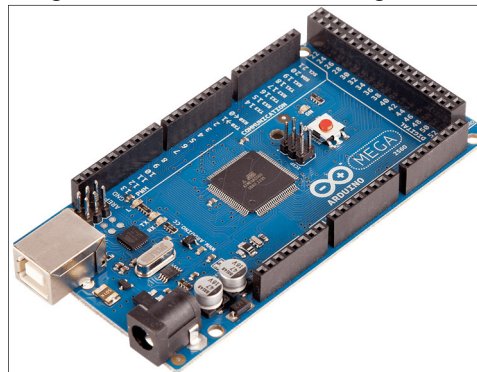
Segundo Arduino (2016a, tradução nossa), “O Arduino é uma plataforma de prototipagem open-source baseado em hardware e software de fácil aquisição e utilização”. Por ser open-source, os projetos e mesmo o hardware podem ser utilizados livremente por qualquer pessoa e com qualquer propósito. Dessa forma, há muitas placas com base no Arduino disponíveis para comercialização ou que podem ser criadas a partir de um diagrama (MCROBERTS, 2011). De acordo com McRoberts (2011, p. 22), “em termos práticos, um Arduino é um pequeno computador que você pode programar para processar entradas e saídas entre o dispositivo e os componentes externos conectados a ele”. Seu software (IDE – Integrated Development Environment ou Ambiente de Desenvolvimento Integrado) utiliza uma linguagem baseada em C/C++, linguagem bem difundida. É compatível com os sistemas operacionais Windows, Macintosh OSX e Linux. Suas placas são relativamente mais baratas comparadas a outras plataformas microcontroladas (VIDA DE SILÍCIO, entre 2014 e 2016).

O Arduino pode ser utilizado para desenvolver objetos interativos independentes, ou pode ser conectado a um computador, a uma rede, ou até mesmo à internet para recuperar e enviar dados do Arduino e atuar sobre eles. Em outras palavras, ele pode enviar um conjunto de dados recebidos de alguns sensores para um site, dados estes que poderão, assim, ser exibidos na forma de um gráfico (MCROBETS, 2011).

3.3.2 Arduino Mega 2560

A placa Arduino Mega 2560 – Figura 3.12, é um modelo da plataforma Arduino baseada no microcontrolador ATmega2560. O ATmega2560 é um microcontrolador CMOS de 8-bit com baixo consumo de energia baseado na arquitetura avançada fabricado pela ATMEL e possui entre suas principais características: 256 KB de memória Flash programável, 4 KB de EEPROM, 8 KB de SRAM, 54 pinos de entradas e saídas digitais onde 15 destes podem ser utilizados como saídas PWM, 16 entradas analógicas com conversores analógico-digital (ADC) de 10-bit, 04 portas de comunicação serial (UARTs), um oscilador de cristal de 16 MHz, Real Time Counter (RTC), 06 portas de para interrupção externa. O Arduino Mega 2560 ainda conta com uma conexão USB, uma entrada jack, conexão ICSP e botão reset (ARDUINO, 2016b; SOUZA, 2014; ATMEL CORPORATION, 2014).

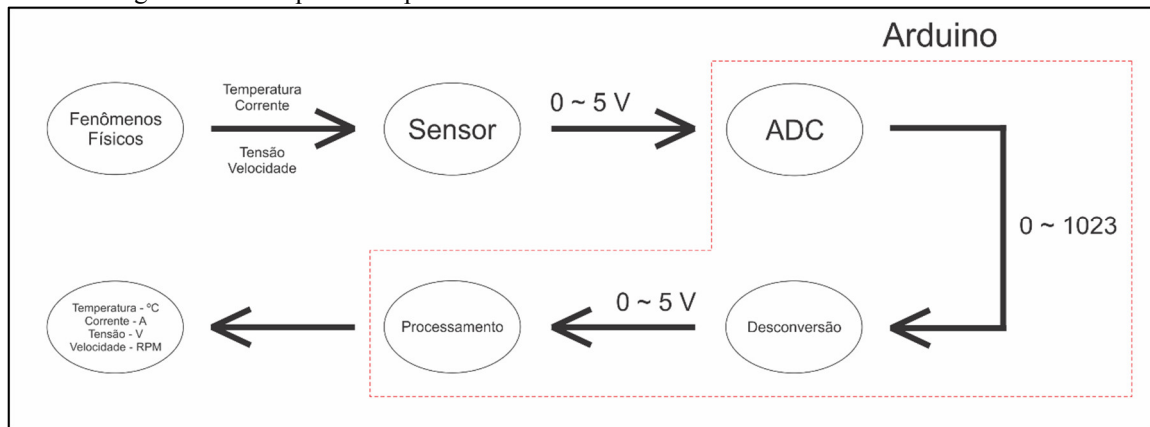
Figura 3.12 – Placa Arduino Mega 2560.



Fonte: www.tato.ind.br.

Por se tratar de um conversor de 10-bits, o ADC trabalha com resolução de 1024 valores para representar o intervalo de 0V a 5V que as portas analógicas do Arduino suportam. Assim, se 5V é representado por um valor de 1024, dividindo 05 por 1024 resultará no valor 0.0048828125, que serve para realizar a conversão de digital (1024) para analógico (5V) e vice-versa. A Figura 3.13 apresenta um esquema que evidencia o processo citado, onde a etapa de desconversão é realizado através da manipulação matemática no código de programação do Arduino.

Figura 3.13 – Esquema simplificado de conversão e desconversão de dados no Arduino.



Fonte: Autoria Própria.

3.4 Interface Gráfica

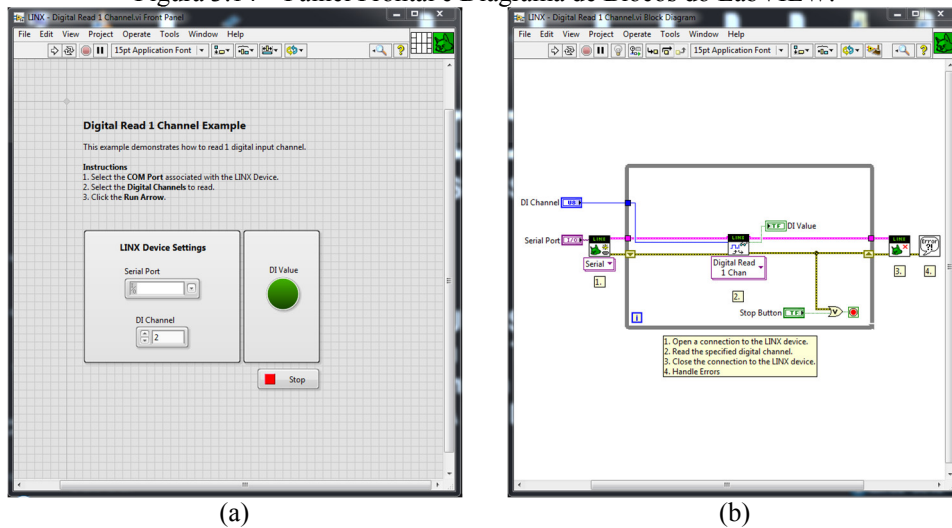
3.4.1 LabVIEW

O LabVIEW, abreviação para Laboratory Virtual Instrument Engineering Workbench, é um ambiente de programação baseado em linguagem gráfica, conhecida como G, desenvolvido pela National Instruments. Também pode ser considerado como um sistema interativo de desenvolvimento e execução de programas projetados para cientistas e engenheiros (BITTER; MOHIUDDIN; NAWROCKI, 2006; TRAVIS; KRING, 2006).

Conforme Bitter, Mohiuddin e Nawrocki (2006), devido à sua natureza gráfica o LabVIEW é ideal para aplicações de teste e medição, automação, instrumentos de controle, aquisição e análise de dados. Para Travis e Kring (2006) a linguagem gráfica do LabVIEW pode melhorar a produtividade no desenvolvimento de programas se comparada a outras linguagens, pois esta foi projetada especificamente para realizar medições, analisar os dados e apresentar os resultados ao usuário.

Os programas desenvolvidos no LabVIEW são chamados de Virtual Instruments (VI). Um VI consiste de um painel frontal, um diagrama de blocos e um ícone que o representa. O painel frontal – Figura 3.14(a), é usado para indicar controles e indicadores para o usuário e o diagrama de blocos – Figura 3.14(b), contém o código de programação da VI. O ícone, além de ser a representação do VI, apresenta conexões para entradas e saídas disponíveis caso se deseje utilizá-lo com outros VI's. Um VI pode conter diversos outros programas que executam partes do programa principal e estes programas secundários denominam-se subVI's.

Figura 3.14 – Painel Frontal e Diagrama de Blocos do LabVIEW.



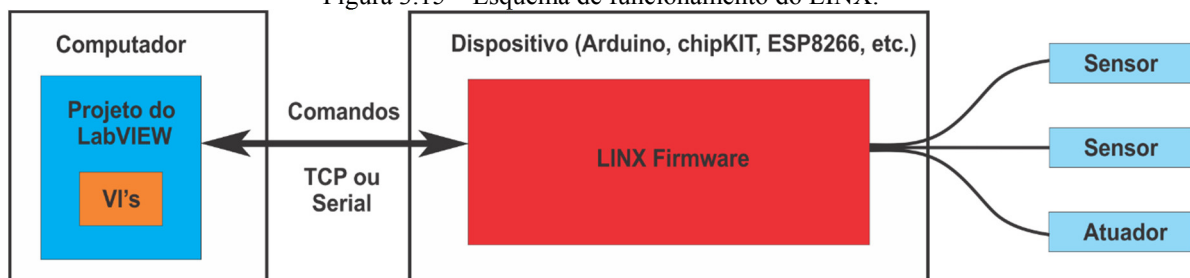
(a) Painel Frontal. (b) Diagrama de Blocos.

Fonte: Acervo Próprio.

3.4.2 LINX

O LINX pode ser considerado como uma camada abstrata de hardware que permite o usuário ter uma única interface do LabVIEW conectada a diferentes dispositivos, como Arduino, chipKIT e myRIO. Ele também pode ser considerado como um grupo de VI's direcionados a interação com plataformas comuns embutidas como as citadas anteriormente (LABVIEW MAKERHUB, 2016). Em outras palavras, o LINX facilita a comunicação entre o Arduino e o LabVIEW, pois este já vem com as configurações necessárias para tanto. O modo como o LINX opera é apresentado na Figura 3.15.

Figura 3.15 – Esquema de funcionamento do LINX.



Fonte: adaptado de www.labviewmakerhub.com.

Utilizando as VI's do LINX e um firmware instalado no dispositivo, neste caso o Arduino, o usuário pode enviar comandos do LabVIEW através de conexão USB, WiFi ou Ethernet. Estes comandos informam ao firmware para realizar ações pré-determinadas como a leitura de uma entrada analógica ou enviar um valor digital ao dispositivo (LABVIEW MAKERHUB, 2016).

3.4.2.1 Comando Customizado

O Comando Customizado, ou Custom Command em inglês, é um recurso que permite a criação e execução de funções customizadas no dispositivo em que o firmware do LINX esteja instalado. Esse recurso permite enviar uma matriz U8 (U8 array), que de forma simplificada representa números inteiros que vão de 0 a 255, do LabVIEW para o dispositivo, e receber outra matriz U8 como resposta. Assim, neste trabalho o Comando Customizado é utilizado como recurso na transferência de dados entre Arduino e LabVIEW.

4 CONDICIONAMENTO DE SINAL E VALIDAÇÃO DE SENSORES

Tanto os sensores quanto o Arduino Mega 2560 possuem características que aparentemente são suficientes para a aplicação proposta neste trabalho. Contudo, para ter certeza da eficácia do funcionamento dos sensores e seus respectivos códigos de processamento, alguns testes foram realizados. Os testes consistem de duas partes. Na primeira, os sensores e os códigos são testados quando há variação da grandeza física sendo medida, e na segunda, quando a grandeza física se mantém constante. Esses testes não foram executados no motor de indução, mas em objetos ou circunstâncias que remetem às características do motor.

Para efeito de validação, as medições dos sensores e Arduino são comparadas com as medições de instrumentos profissionais. Algumas das características técnicas destes instrumentos estão no Apêndice A.

A Equação 4.1 foi utilizada para determinar o erro percentual entre as medições dos sensores e instrumentos de comparação (TADANO, 2014).

$$e = \frac{|x - \bar{x}|}{\bar{x}} 100 \quad (\%) \quad (4.1)$$

Onde,

- e – erro relativo percentual;
- x – valor aproximado (valor dos sensores);
- \bar{x} – valor verdadeiro (valor do instrumento).

As versões completas dos códigos de processamento abordados neste capítulo estão no Apêndice B.

A Tabela 4.1 indica a quantidade de sensores que foram testados.

Tabela 4.1 – Quantidade de sensores testados

Sensor	Quantidade
ACS712	5
TCRT5000	1
STAC	4
LM35	1

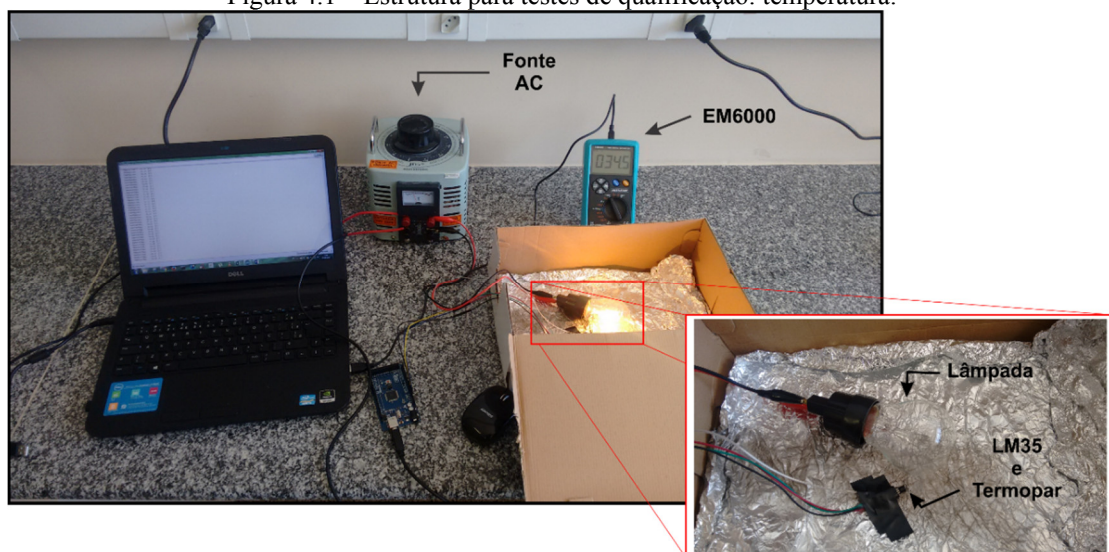
Fonte: Autoria Própria.

4.1 Testes para Validação do Sensor LM35

4.1.1 Características dos Testes

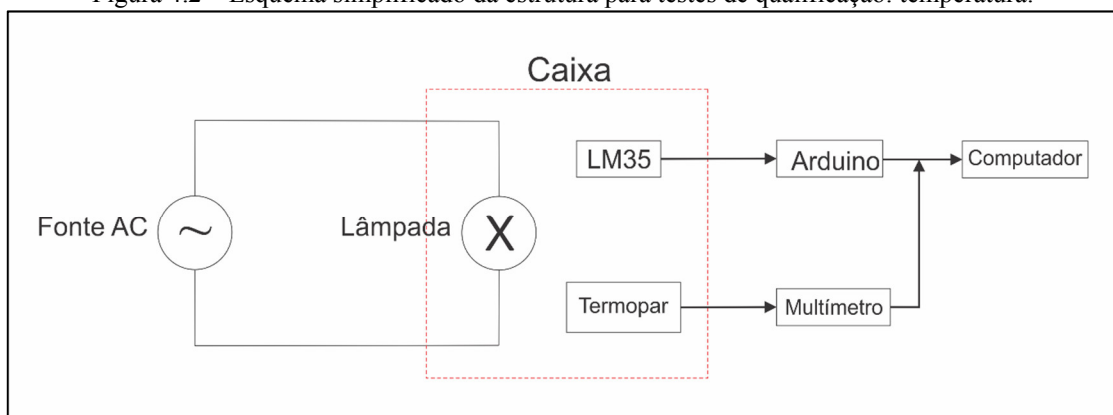
Esses testes tratam da medição da temperatura interna em uma caixa de papelão, no qual fez-se uso de uma lâmpada incandescente – 60 W / 127 V, para elevação da temperatura. A lâmpada foi alimentada por uma fonte AC ajustável. Para comparação de resultados, utilizou-se o multímetro Instrutemp EM6000. A Figura 4.1 apresenta a estrutura real montada para a realização dos testes, enquanto que a Figura 4.2, o esquema simplificado dessa estrutura.

Figura 4.1 – Estrutura para testes de qualificação: temperatura.



Fonte: Autoria Própria.

Figura 4.2 – Esquema simplificado da estrutura para testes de qualificação: temperatura.



Fonte: Autoria Própria.

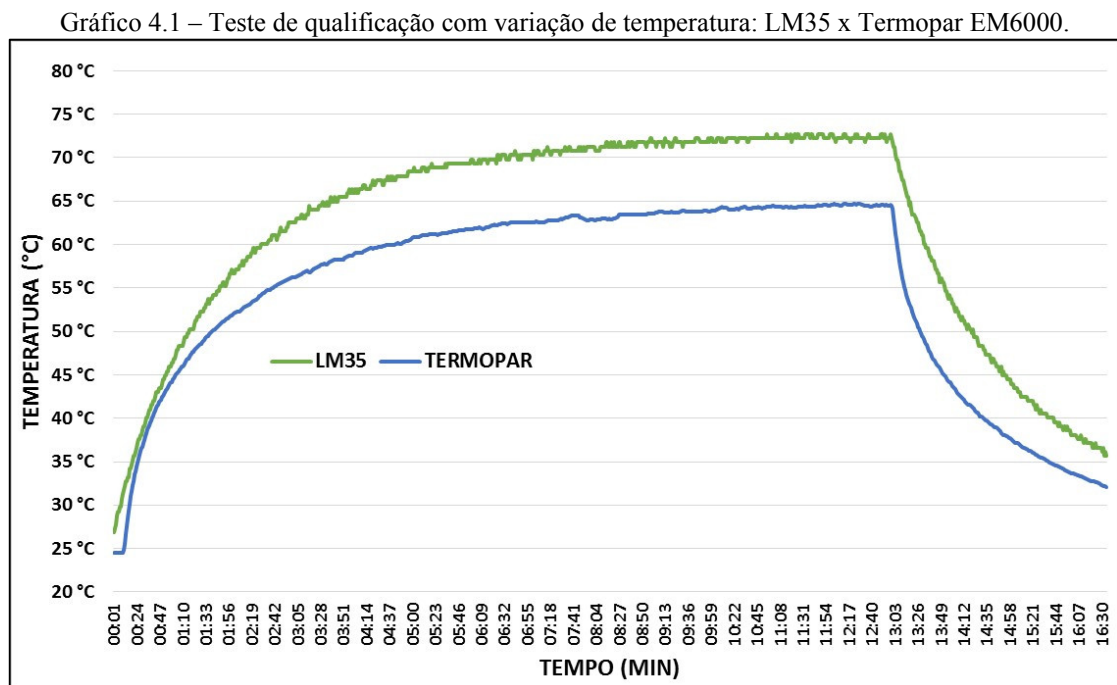
4.1.2 Código de Processamento

O código de programação utilizado nestes testes se encontra no Apêndice B.1. Nas linhas de código 02 a 05 estão as variáveis criadas para auxiliar nos testes. Destas variáveis, `pinSensor` é a variável para definir a entrada analógica a qual será conectada a saída do sensor LM35 – neste caso o pino A0; `voltsporUnidade` é a variável que converte para volts os valores gerados pelo conversor analógico-digital (ADC) do Arduino. Já na linha 14, há a conversão dos valores de tensão para a escala Celsius (°C) conforme Equação 3.2, presente na seção 3.2.1.

4.1.3 Procedimentos e Resultados Obtidos

4.1.3.1 Simulação de regime transitório

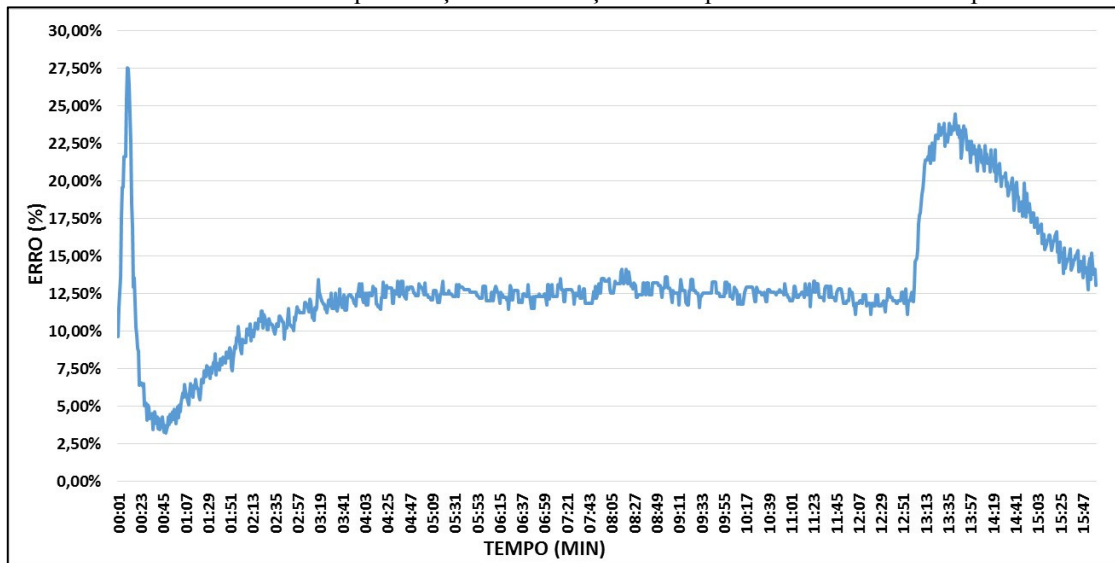
Nesta simulação foram coletados valores da temperatura interna da caixa durante 16 minutos e 35 segundos, onde, durante os 13 primeiros minutos, manteve-se a lâmpada acesa, ocasionando a elevação de temperatura. Após esse período, desligou-se a alimentação da lâmpada de modo que fosse possível verificar as medições quando ocorre queda de temperatura. Os valores coletados estão exibidos no Gráfico 4.1.



Fonte: Autoria Própria.

No Gráfico 4.2 os erros percentuais entre as medições do LM35 e termopar são expostos. Observa-se que, num primeiro momento, o erro tem uma variação brusca, atingindo valores máximo e mínimo, devido à natureza transitória do início das medições. A partir de, aproximadamente, 55 °C os erros tendem a permanecer na faixa de 10 a 15% até o momento em que a lâmpada é desligada, ocasionando um novo distúrbio nos erros, que tendem a decair de forma quase constante. Ressalta-se que, para o valor de temperatura operacional da máquina, 44,5 °C, o erro é de 4,25% aproximadamente.

Gráfico 4.2 – Erros no teste de qualificação com variação de temperatura: LM35 x Termopar EM6000.



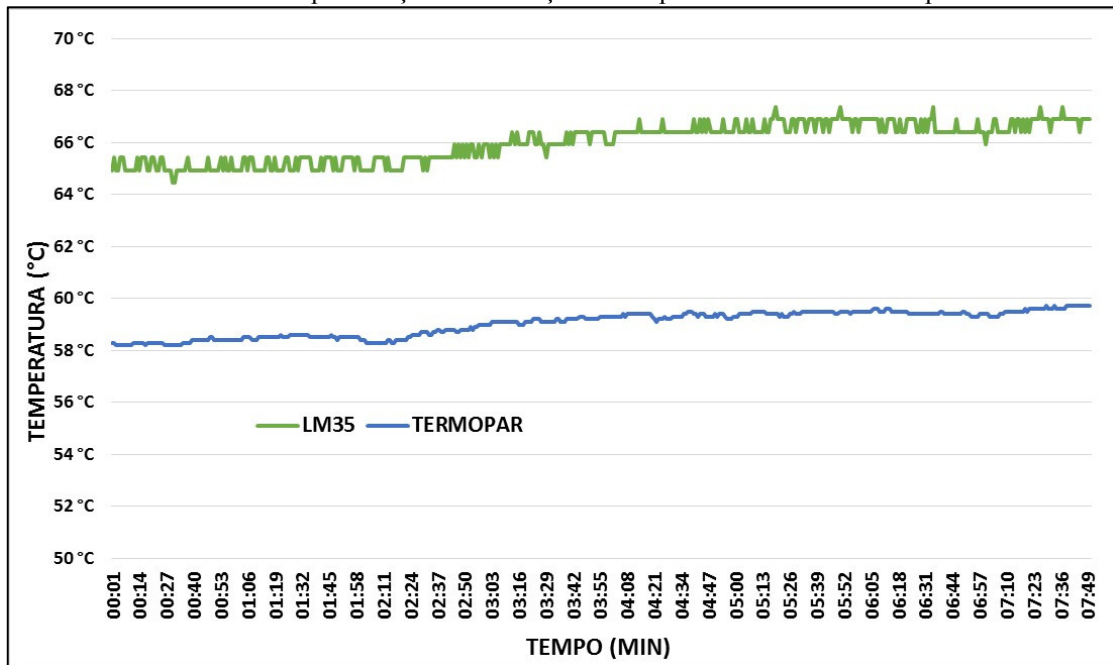
Fonte: Autoria Própria.

4.1.3.2 Simulação de regime permanente

Na análise sem variação da temperatura a lâmpada permaneceu acesa por 15 minutos ininterruptos antes do início da coleta dos dados. Em seguida, durante 7 minutos e 50 segundos, os valores de temperatura foram colhidos.

O resultado do teste está exposto no Gráfico 4.3, de onde se observa que a diferença entre as medições ficou em torno de 6,89 °C.

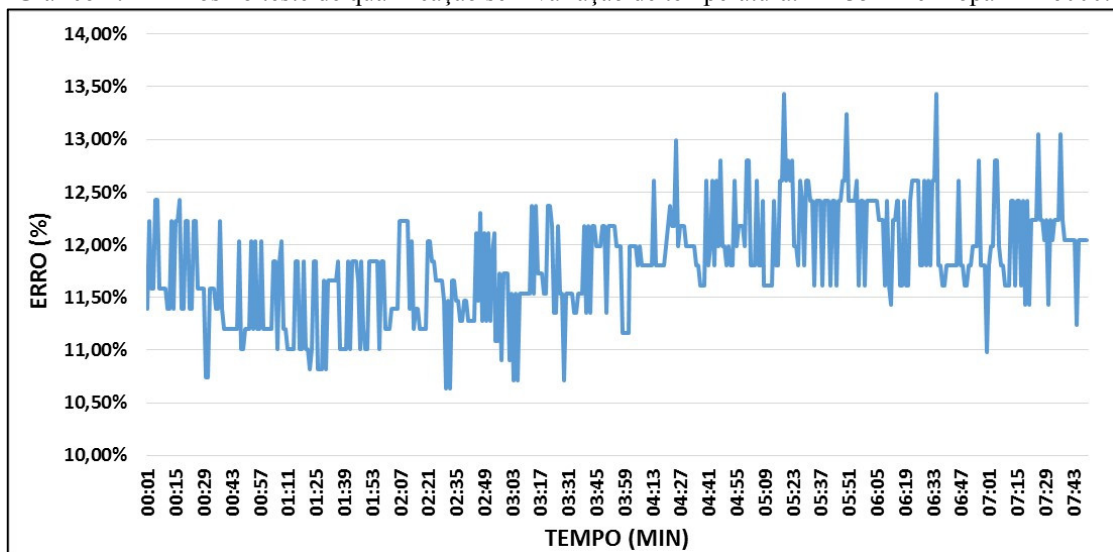
Gráfico 4.3 – Teste de qualificação sem variação de temperatura: LM35 x Termopar EM6000.



Fonte: Autoria Própria.

Analisando o Gráfico 4.4, observa-se que os erros nas medições se mantiveram aproximadamente constantes – com média de 11,82%, o que permite realizar a mitigação destes erros através de manipulação matemática na programação.

Gráfico 4.4 – Erros no teste de qualificação sem variação de temperatura: LM35 x Termopar EM6000.



Fonte: Autoria Própria.

4.2 Testes para Validação do Sensor TCRT5000

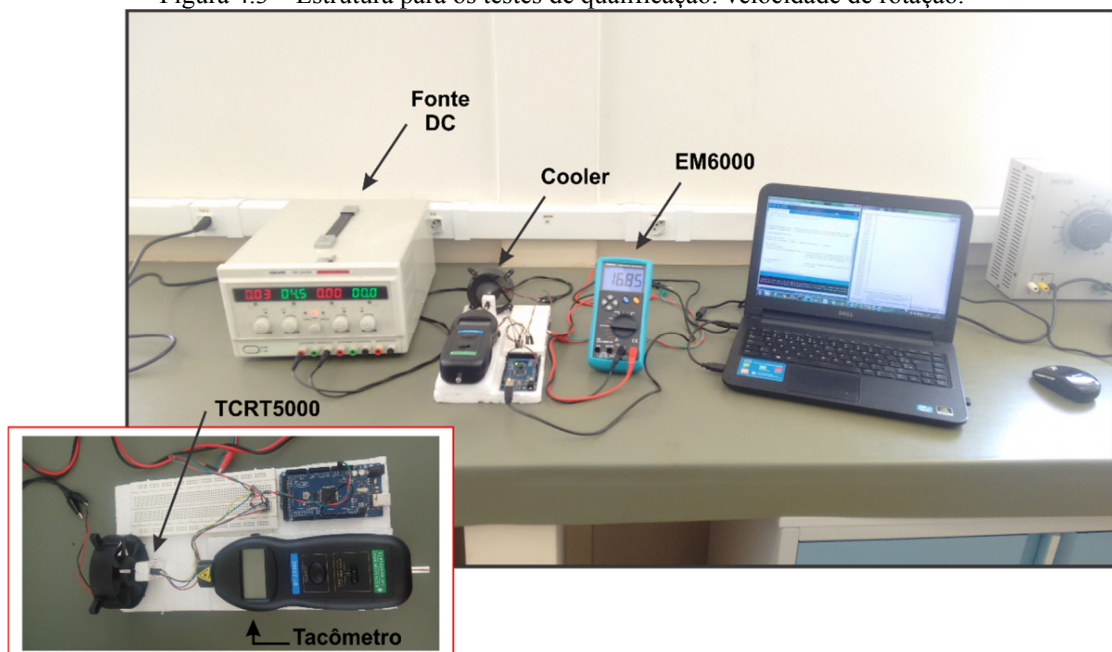
4.2.1 Características dos Testes

Esses testes consistem na aquisição da velocidade de rotação das palhetas de um cooler, ou exaustor de pequeno porte, comumente encontrado em microcomputadores, com o uso do sensor TCRT5000. Compõem, também, a estrutura deste teste, uma faixa reflexiva de dimensões 24x03 mm conectada a uma das palhetas do cooler e uma fonte DC que controla a velocidade de rotação através da variação de tensão que chega ao cooler.

Para a determinação da velocidade o TCRT5000 detecta a passagem da faixa reflexiva, que está 02 mm distante dele, sempre que a palheta em que a faixa se encontra completa uma rotação. Somando-se a quantidade de detecções em determinado intervalo de tempo obtém-se a velocidade de rotação das palhetas do cooler. Todavia, devido ao uso de apenas uma faixa reflexiva a resolução das medições com o Arduino é de 60 RPM, significando que o Arduino medirá apenas velocidades que sejam múltiplas de 60.

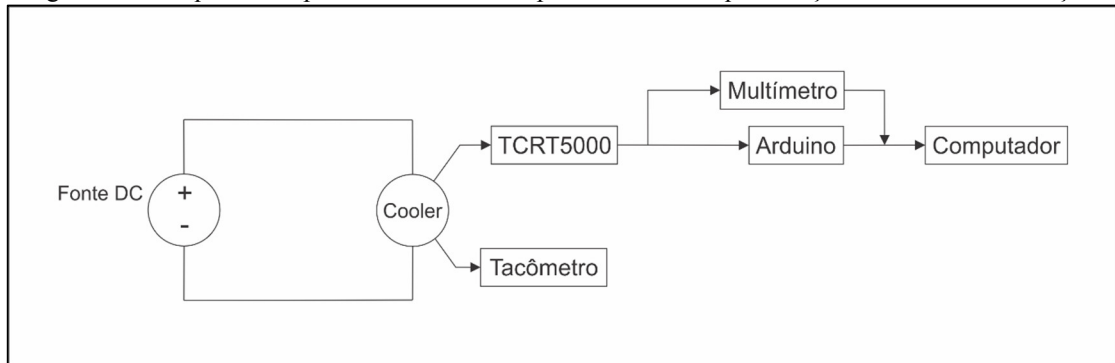
As Figura 4.3 e 4.4 apresentam, respectivamente, a estrutura utilizada para a realização dos testes e o esquema que a elucida.

Figura 4.3 – Estrutura para os testes de qualificação: velocidade de rotação.



Fonte: Autoria Própria.

Figura 4.4 – Esquema simplificado da estrutura para os testes de qualificação: velocidade de rotação.



Fonte: Autoria Própria.

4.2.2 Código de Processamento

O código de programação utilizado nestes testes se encontra no Apêndice B.2. Na linha de código 01 há a inserção de uma biblioteca destinada a contagem de tempo em milissegundos. A linha de código 02 expõe a variável `timeElapsed` que inicializa a biblioteca de tempo. Essa variável começa a contar o tempo a partir do momento em que o código é executado.

A variável `pulsos`, linha 05, é usada para a contagem das passagens da faixa reflexiva pelo sensor e é necessário que esta variável seja do tipo `volatile`, pois, desta forma, ela pode transitar entre diferentes funções dentro do código sem que possa ocorrer erros em seus valores, diferentemente das variáveis comuns. A variável `intervalo`, linha 06, é o intervalo no qual a contagem das passagens da faixa reflexiva é realizada.

As linhas 08 a 10 tratam da função que será executada caso a interrupção externa seja ativada por um evento externo. No caso presente, a função conta as vezes em que o sensor TCRT5000 detecta a passagem da faixa reflexiva.

Na linha 14 estão estabelecidas as configurações para o uso de interrupção externa no Arduino. As interrupções têm como finalidade priorizar eventos tanto internos quanto externos ao Arduino, ou seja, outras funções dentro de um código serão pausadas para que os eventos detectados pela interrupção sejam processados e então o código voltará a sua rotina original. A detecção pelo sensor TCRT5000 da faixa reflexiva pode ser citada como um exemplo de um evento externo. Os parâmetros `0`, `interrupcao`, `FALLING`, são, respectivamente, o número de identificação da interrupção sendo utilizada, o nome da função que processa os dados do evento externo e o modo como a interrupção externa é ativada.

O Arduino Mega 2560 possui 06 portas digitais que podem ser utilizadas na detecção de interrupções externas. A Tabela 4.2 mostra as interrupções externas e as portas digitais associadas a elas (ARDUINO, 2016b).

Tabela 4.2 – Interrupções externas do Arduino Mega 2560.

	Interrupção 0	Interrupção 1	Interrupção 2	Interrupção 3	Interrupção 4	Interrupção 5
Porta Digital	02	03	21	20	19	18

Fonte: Arduino (2016b).

Os modos de ativação da interrupção externa são quatro:

- **LOW:** Ativa a interrupção quando a tensão na porta digital está em 0 V;
- **CHANGE:** Ativa sempre que o sinal que chegando a porta digital muda de estado, de 0 (0 V) para 1 (5 V) ou vice-versa;
- **RISING:** Ativa somente quando há mudança de estado de 0 (0 V) para 1 (5 V);
- **FALLING:** Ativa somente quando há mudança de estado de 1 (5 V) para 0 (0 V).

Já o código presente nas linhas 21 a 29 tem como finalidade verificar se o tempo para contagem das detecções da faixa chegou ao fim e, em caso positivo, apresentar as medições já convertidas para RPM ao utilizador, além de reiniciar as variáveis `pulsos` e `timeElapsed` para que novas medições possam ocorrer.

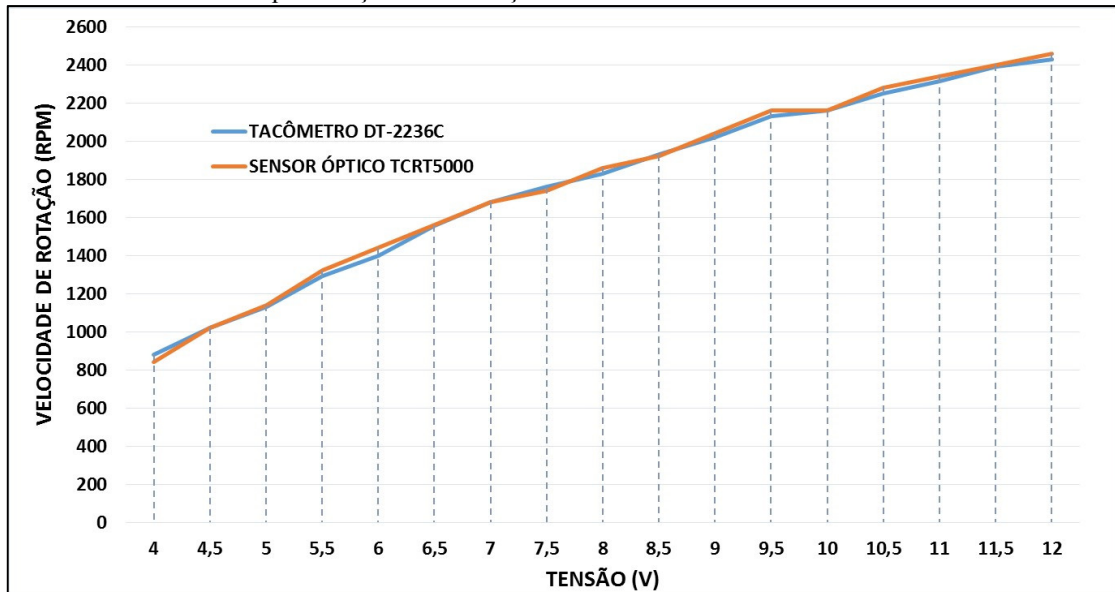
4.2.3 Procedimentos e Resultados Obtidos

4.2.3.1 Simulação de fundo de escala

Para esta análise, os valores de tensão que alimentam o cooler foram variados de 0 a 12 V com aumento gradativo de 0,5 V a cada medição. Para efeito de validação, utilizou-se como referência os valores de velocidade obtidos pelo tacômetro óptico modelo DT-2236C. Todavia, enquanto a alimentação do cooler não atinge 04 V ele não sai de seu estado de inércia. Assim, os dados para valores de tensão menores que 04 V foram desconsiderados.

As medições obtidas são apresentadas no Gráfico 4.5.

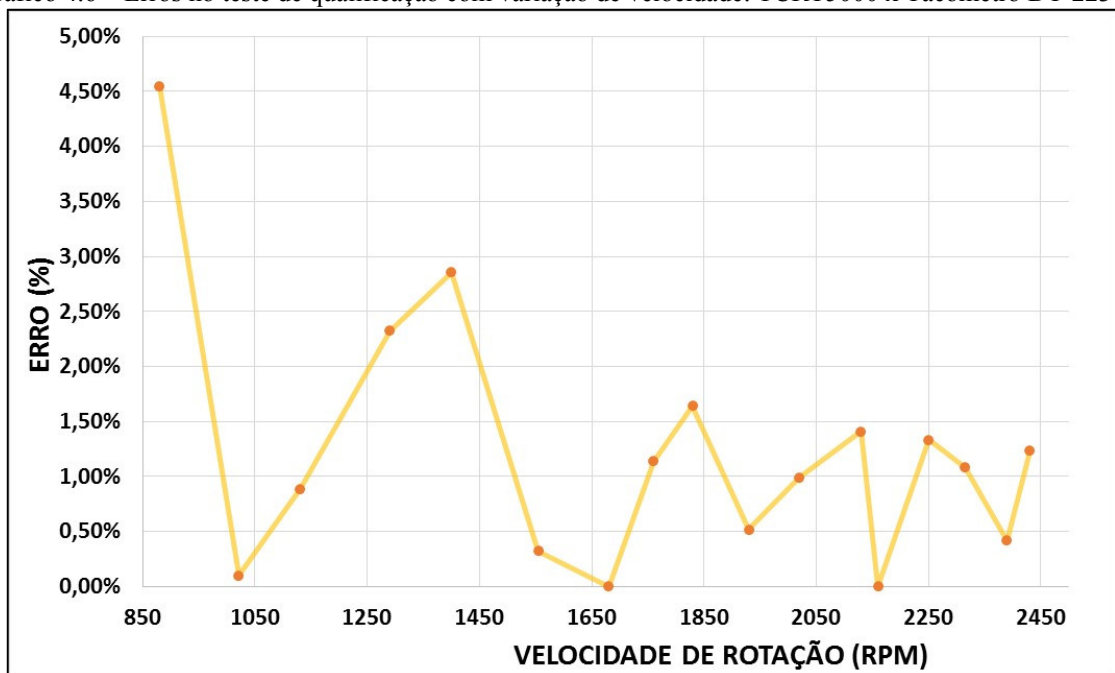
Gráfico 4.5 – Teste de qualificação com variação de velocidade: TCRT5000 x Tacômetro DT-2236C.



Fonte: Autoria Própria.

O erro médio percentual entre as medições do tacômetro e do sensor foi de 1,22%, enquanto que para a faixa de rotação do MIT, 1800 RPM, o erro situou-se entre 1,14% e 1,64%, em concordância com o que é exposto no Gráfico 4.6.

Gráfico 4.6 – Erros no teste de qualificação com variação de velocidade: TCRT5000 x Tacômetro DT-2236C.

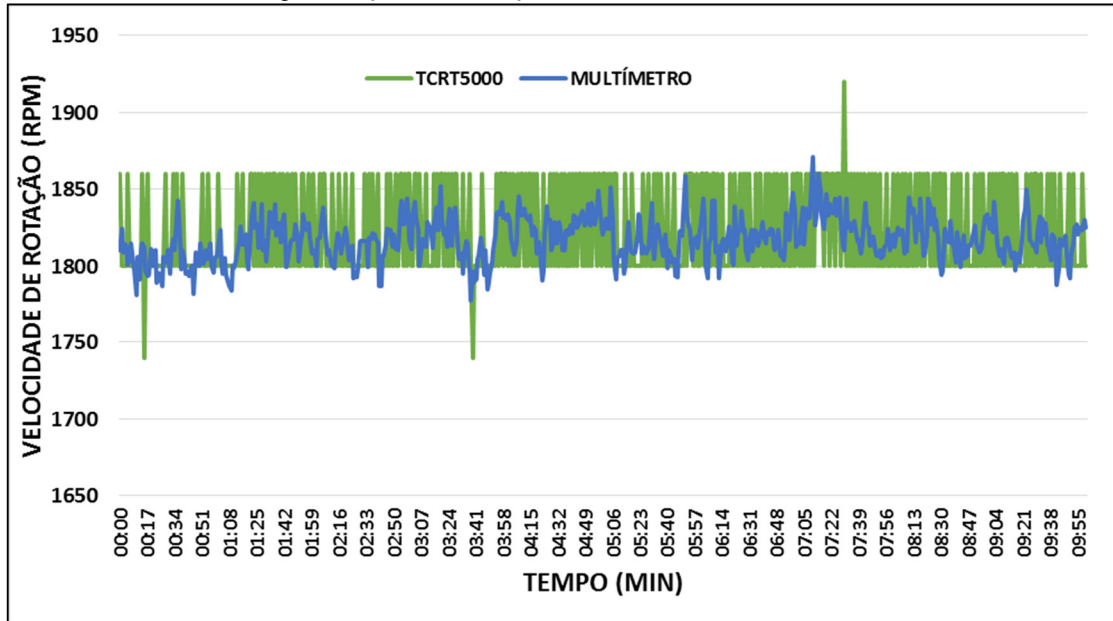


Fonte: Autoria Própria.

4.2.3.2 Simulação de regime permanente

Na análise do desempenho do sensor para os valores nominais de rotação do MIT, 1800 RPM, o cooler ficou em funcionamento por 10 minutos ininterruptos, porém, desta vez, o teste foi realizado com o auxílio do multímetro EM6000, que mediu a frequência de saída das detecções do TCRT5000. Os resultados estão expostos no Gráfico 4.7.

Gráfico 4.7 – Teste de qualificação sem variação de velocidade: TCRT5000 x Multímetro EM6000.

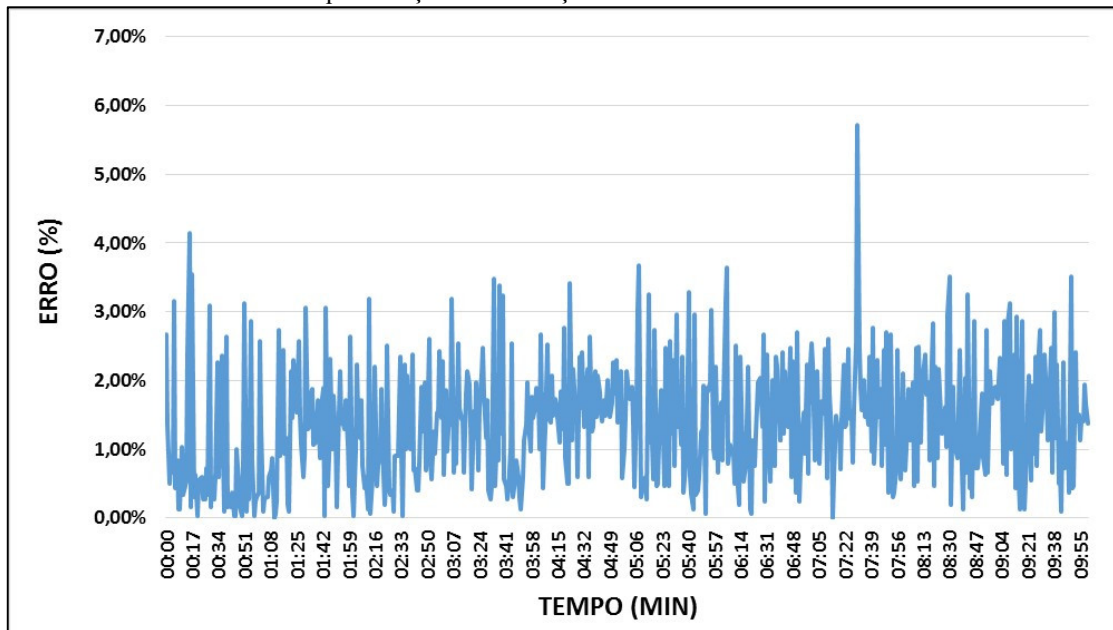


Fonte: Autoria Própria.

As medições com o EM6000 são as que possuem menores erros devido a sua resolução ser melhor que a do sensor. Já o TCRT5000 apresenta dois valores em suas medições, 1800 e 1860, com alguns picos, em virtude da limitação decorrente de sua resolução de 60 RPM, exposta anteriormente.

O erro percentual entre as medições atinge um pico máximo de 5,72%, contudo, em geral, se mantém na faixa de 0 a 3%, com valor médio de 1,36% conforme Gráfico 4.8.

Gráfico 4.8 – Erros no teste de qualificação sem variação de velocidade: TCRT5000 x Multímetro EM6000.



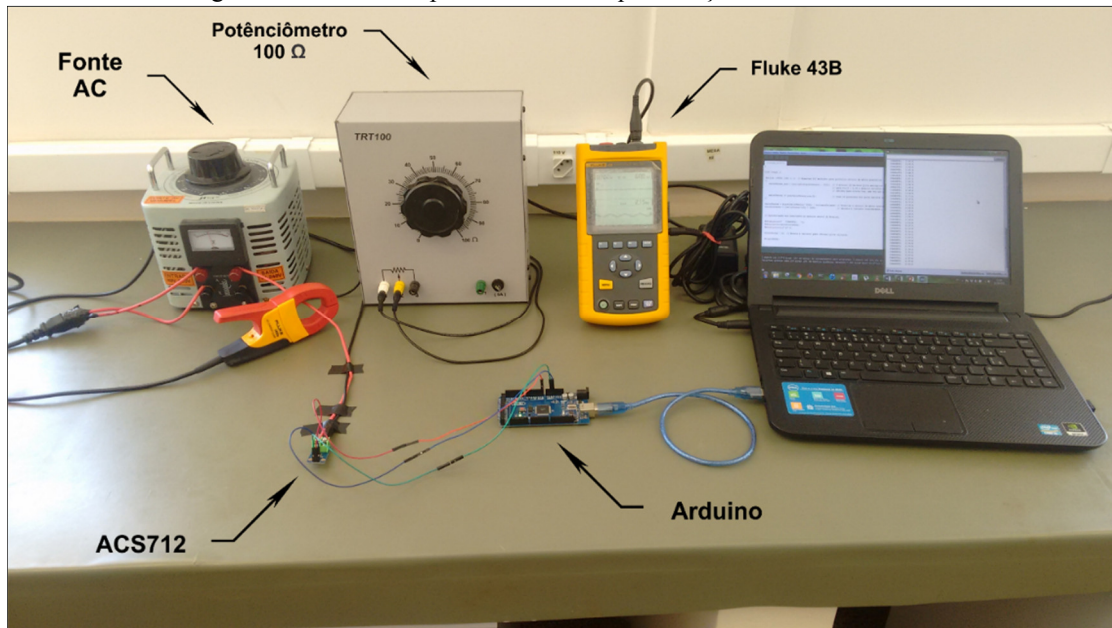
Fonte: Autoria Própria.

4.3 Testes para Validação do Sensor ACS712

4.3.1 Características dos Testes

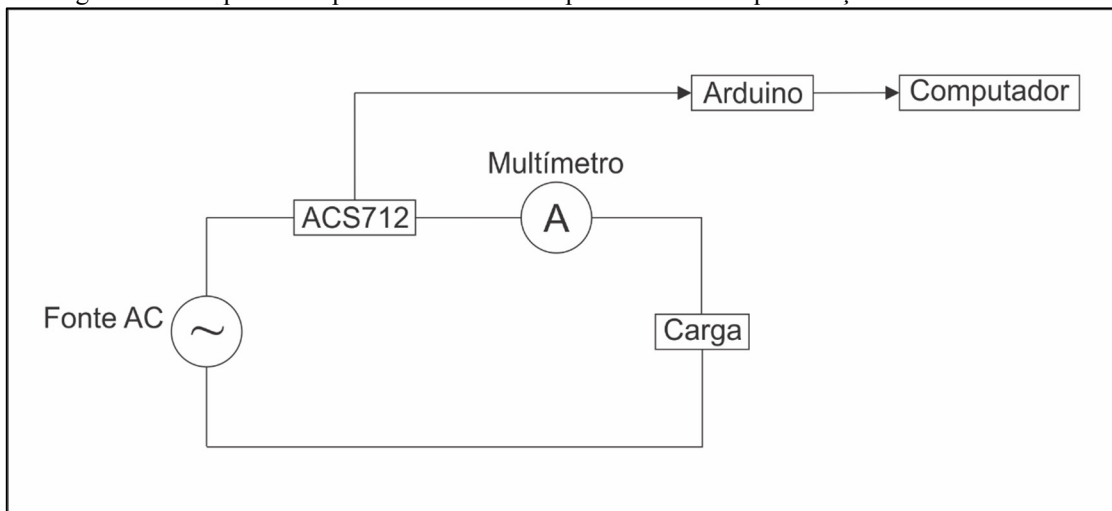
Nestes testes foi medida a corrente elétrica que circula em um circuito, que tem como carga um potenciômetro de 100 Ω , utilizando os sensores ACS712, conforme mostram as Figura 4.5 e 4.6. Também se utilizou uma fonte de tensão AC ajustável para manter a tensão de alimentação constante em 127 V.

Figura 4.5 – Estrutura para os testes de qualificação: corrente elétrica.



Fonte: Autoria Própria.

Figura 4.6 – Esquema simplificado da estrutura para os testes de qualificação: corrente elétrica.



Fonte: Autoria Própria.

4.3.2 Código de Processamento

O código de programação utilizado nestes testes se encontra no Apêndice B.3. Este código foi retirado de Dutra (2013), com a única diferença o valor 100, presente na linha 21, que no código original era 66.

Nas linhas 02 a 06 estão declaradas as variáveis que são usadas para realizar a medição de corrente.

Das linhas 14 a 28 é efetuada a média quadrática de 500 amostras das medições do sensor, em conformidade com o que fora descrito na seção 3.2.3.

Na linha 16, o trecho do código (`analogRead(pinoSensor) - 511`) é equivalente a $(V_{\text{out}} - 2,5)$ na Equação 3.5, porém como se trata de dados entendíveis pelo Arduino, seção 3.3.2, 511 é o equivalente a 2,5 V.

$$I_p = (V_{\text{out}} - 2,5) \times 10 \quad (\text{A}) \quad (3.5)$$

Já na linha 21 ocorre a divisão pelo valor de sensibilidade do sensor, 100 mV/A no caso do modelo utilizado, e então a conversão de miliampères (mA) para ampères (A) através da multiplicação por 1000. É importante enfatizar que tanto a divisão por 100, quanto a multiplicação por 1000 podem ser substituídos por uma multiplicação pelo valor 10, de acordo com a Equação 3.5, porém como esse é um código genérico desenvolvido para ser utilizado com todos os modelos de sensores da linha ACS712, o desenvolvedor optou por manter o código com os trechos de divisão e multiplicação, assim na hipótese do uso de um modelo com sensibilidade diferente de 100 mV/A basta a substituição do valor 100 pelo valor adequado do sensor sendo utilizado. Por fim, na linha 28 o valor obtido através da média quadrática é reiniciado e uma nova medição pode começar.

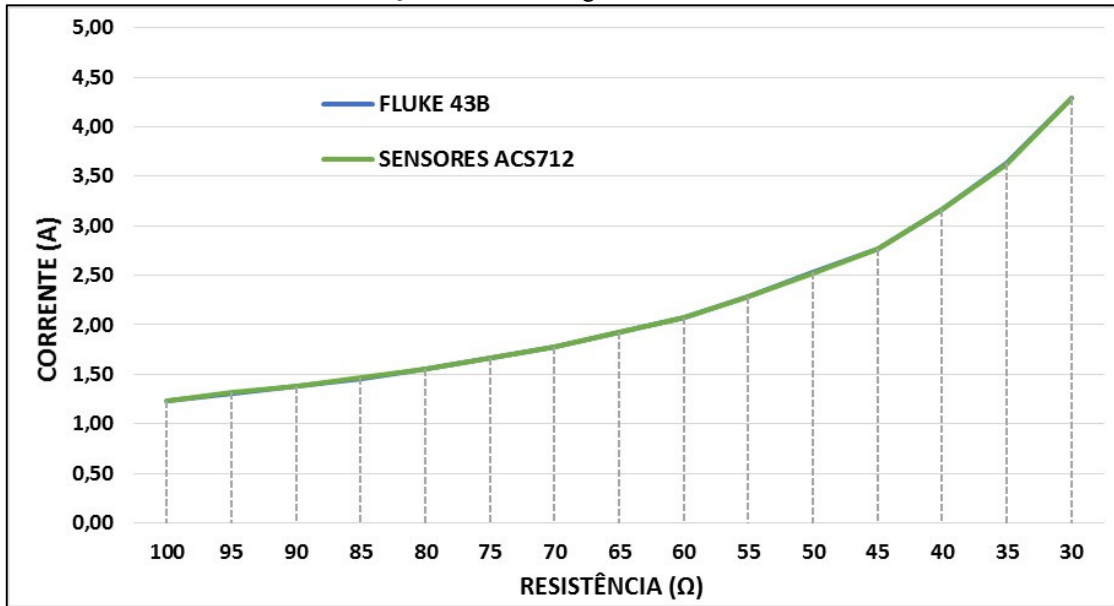
4.3.3 Procedimentos e Resultados Obtidos

4.3.3.1 Simulação de fundo de escala

As medições de corrente foram efetuadas variando-se o valor de resistência do potenciômetro de 100 a 30 Ω , em intervalos de 05 Ω . A validação ocorreu com o uso do analisador de qualidade de energia Fluke 43B.

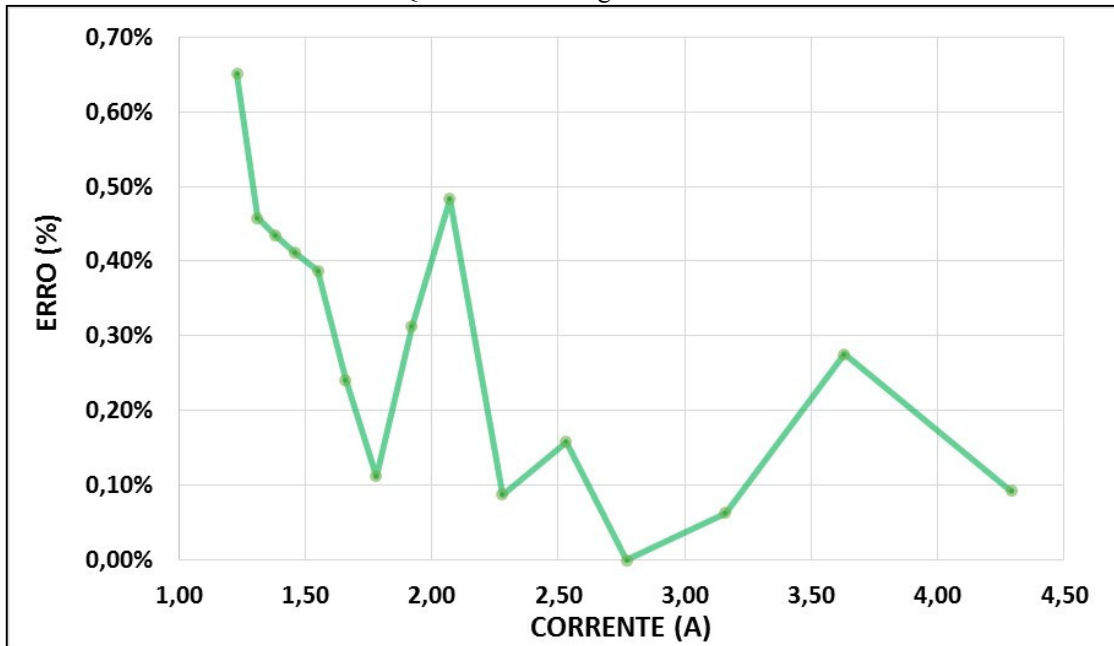
Devido ao número de curvas, 5 dos ACS712 e 5 do Fluke 43B, e pela proximidade dos valores obtidos, optou-se por apresentar um único gráfico com as médias dessas curvas para melhor visualização. Os resultados estão no Gráfico 4.9.

Gráfico 4.9 – Teste de qualificação com variação de corrente (valores médios): ACS712 x Analisador de Qualidade de Energia Fluke 43B.



Fonte: Autoria Própria.

Gráfico 4.10 – Erros no teste de qualificação com variação de corrente (valores médios): ACS712 x Analisador de Qualidade de Energia Fluke 43B.



Fonte: Autoria Própria.

O erro médio percentual geral é de 0,28%, enquanto que, para a faixa de operação nominal do MIT – 3,8 A, o erro é de 0,2%, de acordo com o Gráfico 4.10.

A Tabela 4.3 detalha os valores de corrente e os erros percentuais obtidos para uma resistência de 35 Ω , cujos valores são os mais próximos à corrente nominal do MIT.

Tabela 4.3 – Comparativo entre as medições, do teste com variação de corrente, para resistência de 35 Ω

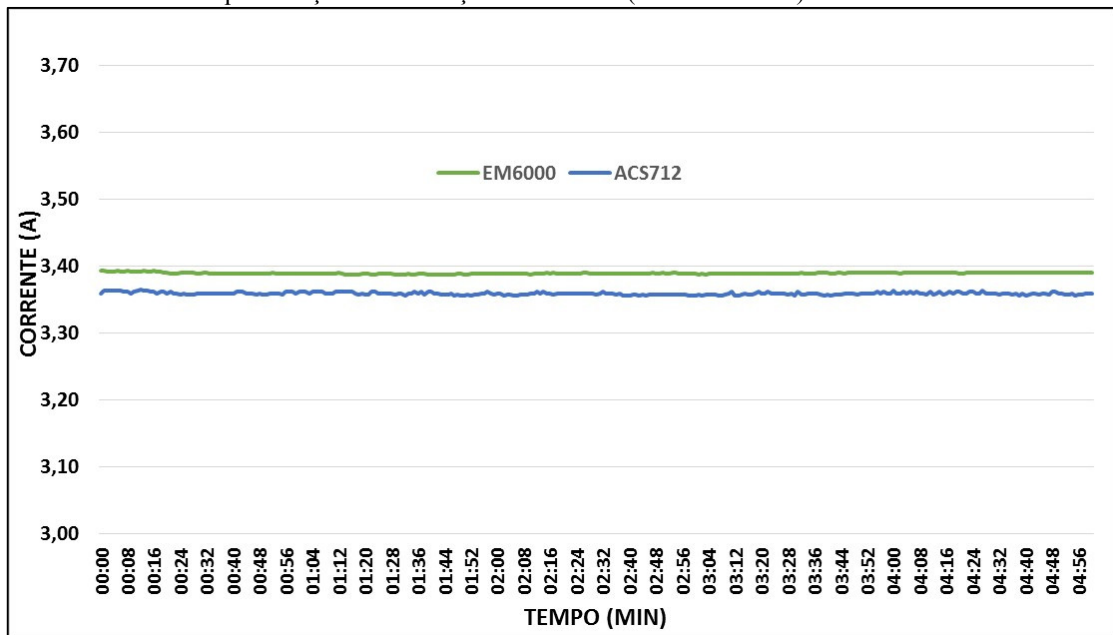
	Sensor I	Sensor II	Sensor III	Sensor IV	Sensor V	Média
Corrente Fluke 43B (A)	3,51	3,67	3,78	3,65	3,53	3,63
Corrente Sensor (A)	3,52	3,66	3,76	3,60	3,56	3,62
Erro (%)	0,28	0,27	0,53	1,37	0,85	0,28

Fonte: Autoria Própria.

4.3.3.2 Simulação de regime permanente

Nesta simulação mediu-se a corrente elétrica de um circuito com uma carga de 425 W e alimentado por uma tensão de 127 V durante 05 minutos. Para a comparação, utilizou-se o multímetro EM6000. O Gráfico 4.11 expõe a média dessas medições.

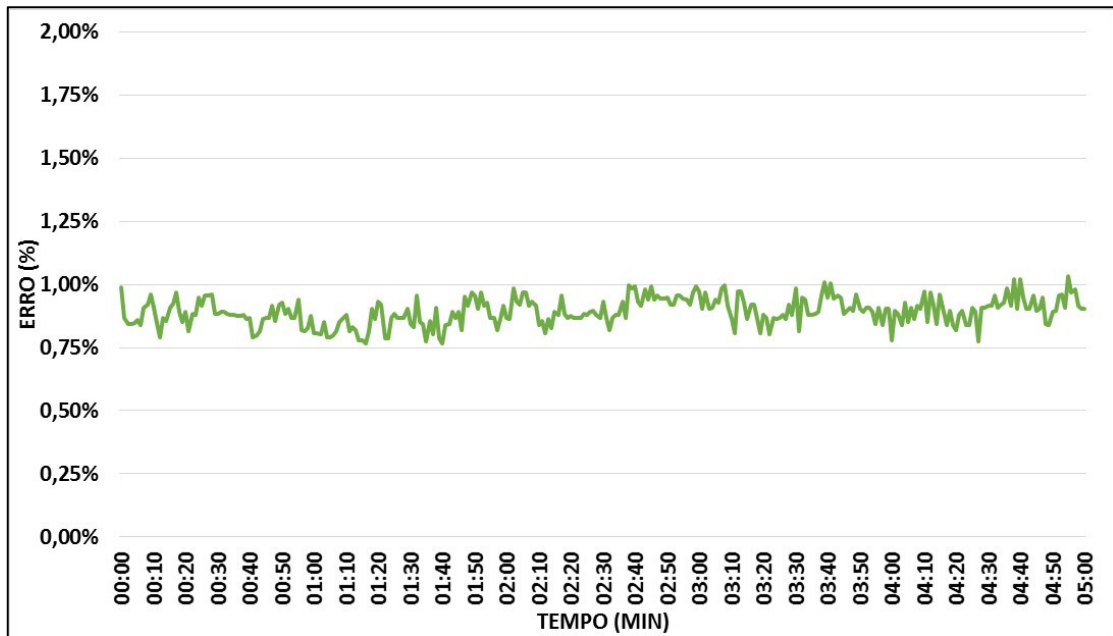
Gráfico 4.11 – Teste de qualificação sem variação de corrente (valores médios): ACS712 x Multímetro EM6000.



Fonte: Autoria Própria.

Conforme é mostrado no Gráfico 4.12, constata-se que o erro das medições médias entre os instrumentos atinge um pico de pouco mais de 1%, contudo seu valor médio fica em torno de 0,88%.

Gráfico 4.12 – Erros no teste de qualificação sem variação de corrente (valores médios): ACS712 x Multímetro EM6000.



Fonte: Autoria Própria.

A Tabela 4.4 exibe um comparativo entre os valores de corrente e erros percentuais encontrados para este teste.

Tabela 4.4 – Comparativo entre as medições do teste sem variação de corrente

	Sensor I	Sensor II	Sensor III	Sensor IV	Sensor V	Média
Corrente EM6000 (A)	3,38	3,39	3,39	3,40	3,39	3,39
Corrente Sensor (A)	3,36	3,34	3,37	3,35	3,38	3,36
Erro (%)	0,59	1,47	0,59	1,47	0,30	0,88

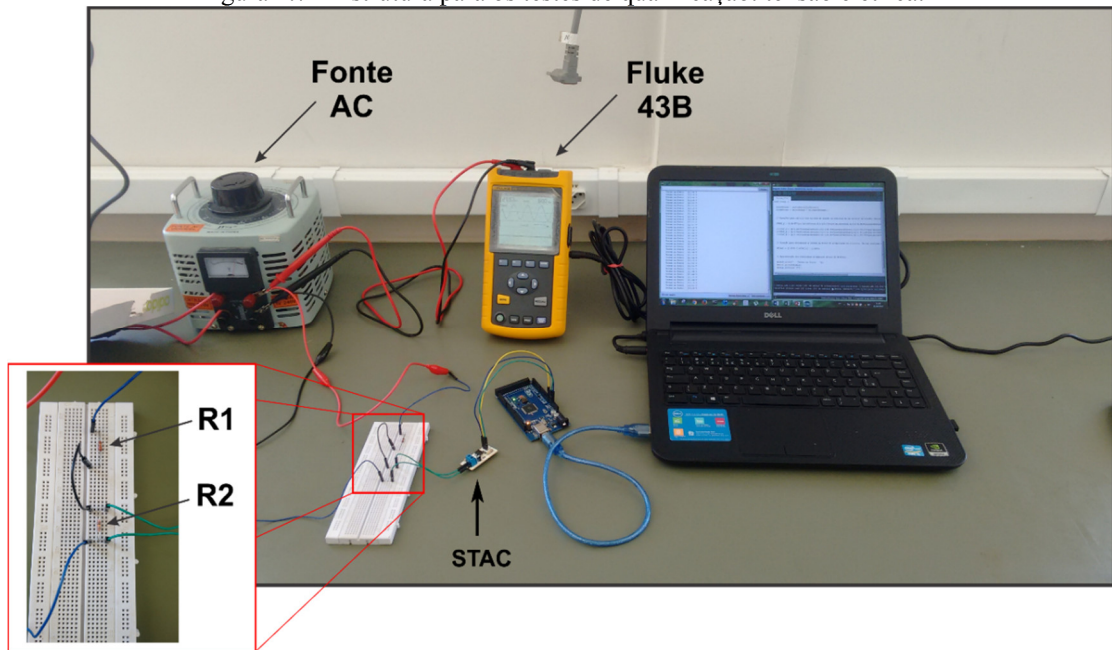
Fonte: Autoria Própria.

4.4 Testes para Validação dos STAC

4.4.1 Características dos Testes

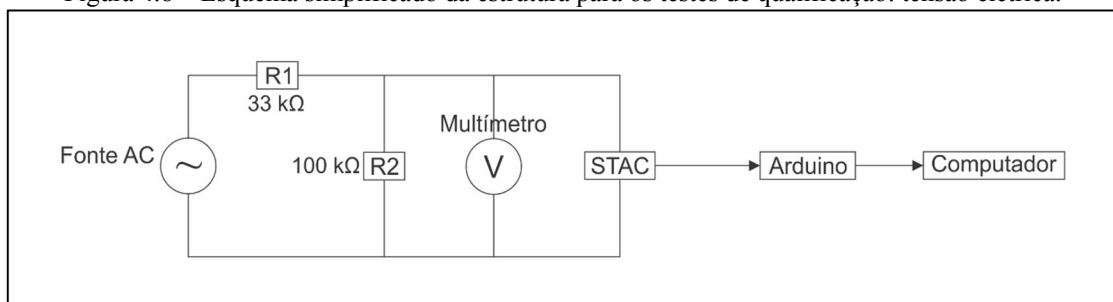
Os testes consistem na utilização dos STAC's para medir a tensão de uma fonte AC ajustável. Paralela a medição dos STAC's, usou-se o analisador de qualidade de energia Fluke 43B e o multímetro Instrutemp EM6000 para comparação de valores. Na Figura 4.7, tem-se a estrutura utilizada para a realização dos testes e na Figura 4.8, o esquema dessa estrutura.

Figura 4.7 – Estrutura para os testes de qualificação: tensão elétrica.



Fonte: Autoria Própria.

Figura 4.8 – Esquema simplificado da estrutura para os testes de qualificação: tensão elétrica.

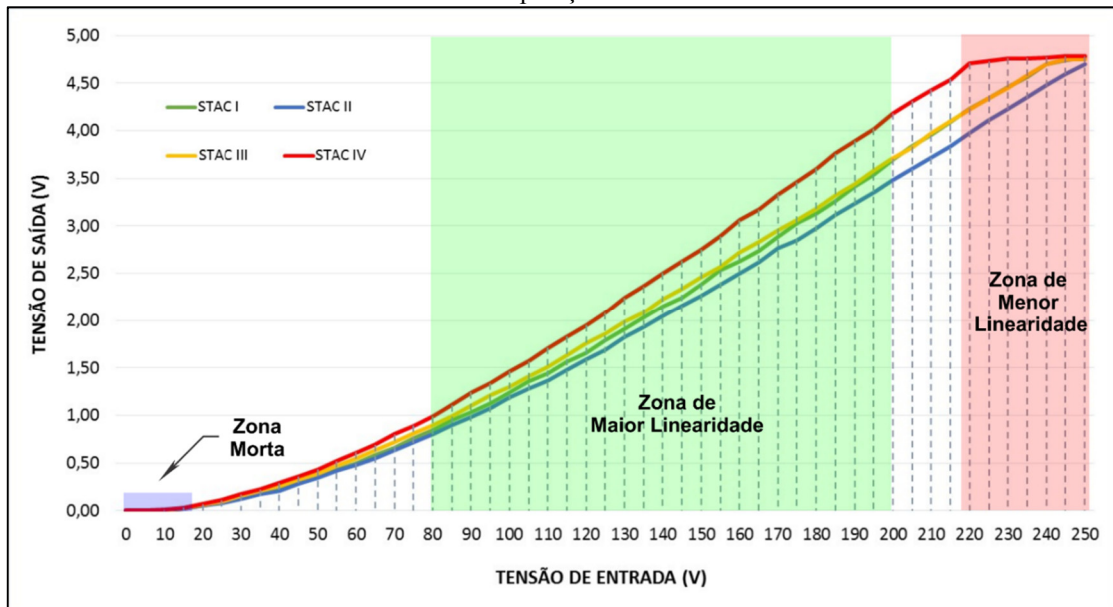


Fonte: Autoria Própria.

Por se tratar de sensores sem ficha de dados (datasheet) foi necessário efetuar testes a fim de coletar informações de funcionamento e desempenho dos STAC's. Nesta coleta de dados foram medidos os dados de saída do sensor, sem qualquer forma de tratamento e sem o uso do

divisor de tensão, proporcionais à variação de tensão – de 0 V a 250 V, em intervalos de 05 V, da fonte AC. Os resultados dos testes para os quatro STAC's estão exibidos no Gráfico 4.13.

Gráfico 4.13 – Curvas de operação característica dos STAC's.



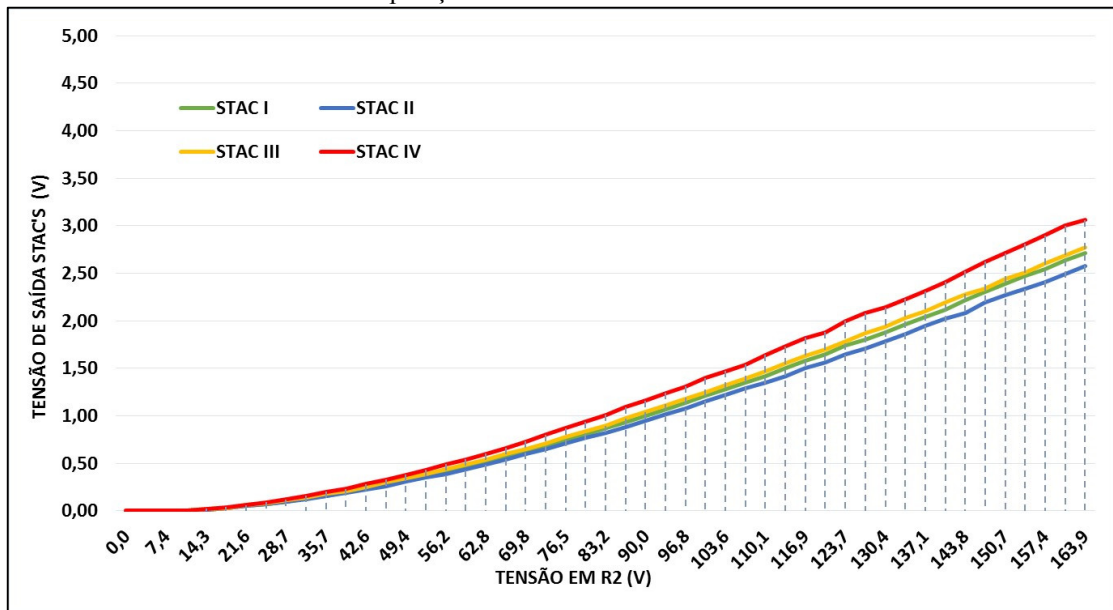
Fonte: Autoria Própria.

Conforme os dados apresentados no Gráfico 4.13, observa-se que a partir da faixa de tensão em que o MIT opera – 220 V, o STAC IV encontra-se na zona de menor linearidade, acarretando assim, na impossibilidade de utilizá-lo para a medição de tensão neste projeto. Como solução para o problema apresentado pelo STAC IV, decidiu-se empregar um divisor de tensão, Figura 4.8, para fazer com que os sensores atuem entre os valores de 80 V a 200 V, onde se tem a maior linearidade das curvas. Também pode-se observar que os sensores começam a operar somente quando a tensão de entrada – tensão da fonte AC, chega a valores próximos a 20 V.

Os resistores presentes no divisor de tensão, $R1 = 33 \text{ k}\Omega$ e $R2 = 100 \text{ k}\Omega$, foram determinados por meio de simulações com o software Excel e o software de projeto de circuitos Multisim, da National Instruments.

Como a tensão em R2 passa a ser então a tensão medida pelos sensores, novos testes para avaliar o comportamento dos STAC's, com o uso do divisor de tensão, foram realizados e, conforme é mostrado no Gráfico 4.14, o problema de operação na zona de menor linearidade foi sanado.

Gráfico 4.14 – Curvas de operação característica dos STAC's com o divisor de tensão.



Fonte: Autoria Própria.

Invertendo os valores do Gráfico 4.14 e através de modelagem matemática obtém-se as Equações 4.2 a 4.5 que são utilizadas para determinar a tensão no resistor R2 em função das respostas dos sensores. A modelagem matemática foi realizada com auxílio do software Excel que utiliza o método dos mínimos quadrados para gerar essas equações. Também, verifica-se com estas medições que os STAC's operam a partir de aproximadamente 14 V no resistor R2.

- STAC I:

$$Y_1 = -4,677x^4 + 30,876x^3 - 74,856x^2 + 123,37x + 16,212 \quad (4.2)$$

- STAC II:

$$Y_2 = -5,4367x^4 + 34,244x^3 - 79,475x^2 + 127,19x + 16,125 \quad (4.3)$$

- STAC III:

$$Y_3 = -3,8669x^4 + 26,593x^3 - 67,298x^2 + 117,81x + 15,45 \quad (4.4)$$

- STAC IV:

$$Y_4 = -2,7451x^4 + 20,447x^3 - 55,939x^2 + 106,22x + 15,443 \quad (4.5)$$

Em que

Y_n – Tensão no resistor R2, V; $n = 1,2,3,4$;

x – Saída do STAC, V.

De posse da tensão no resistor R2 pode-se determinar a tensão da fonte AC através da Equação 4.6, que foi obtida por meio da modelagem da curva presente no Gráfico 4.15.

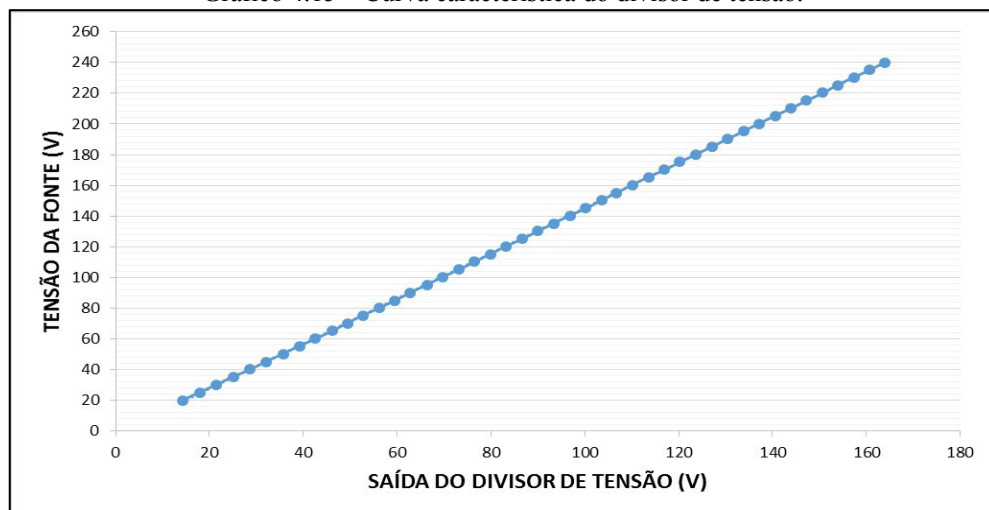
$$Y = 1,4765x - 2,3659 \quad (4.6)$$

Onde,

Y – Tensão de alimentação, V;

x – Tensão no resistor R2, V.

Gráfico 4.15 – Curva característica do divisor de tensão.



Fonte: Autoria Própria.

4.4.2 Código de Processamento

O código de programação utilizado nestes testes se encontra no Apêndice B.4. As linhas 02 a 07 tratam da declaração das variáveis necessárias para realizar as medições, onde v_{STAC_1} corresponde a tensão que é lida pelos sensores STAC's, ou seja, a tensão em R2, enquanto que, v_{Final} é a tensão da fonte AC.

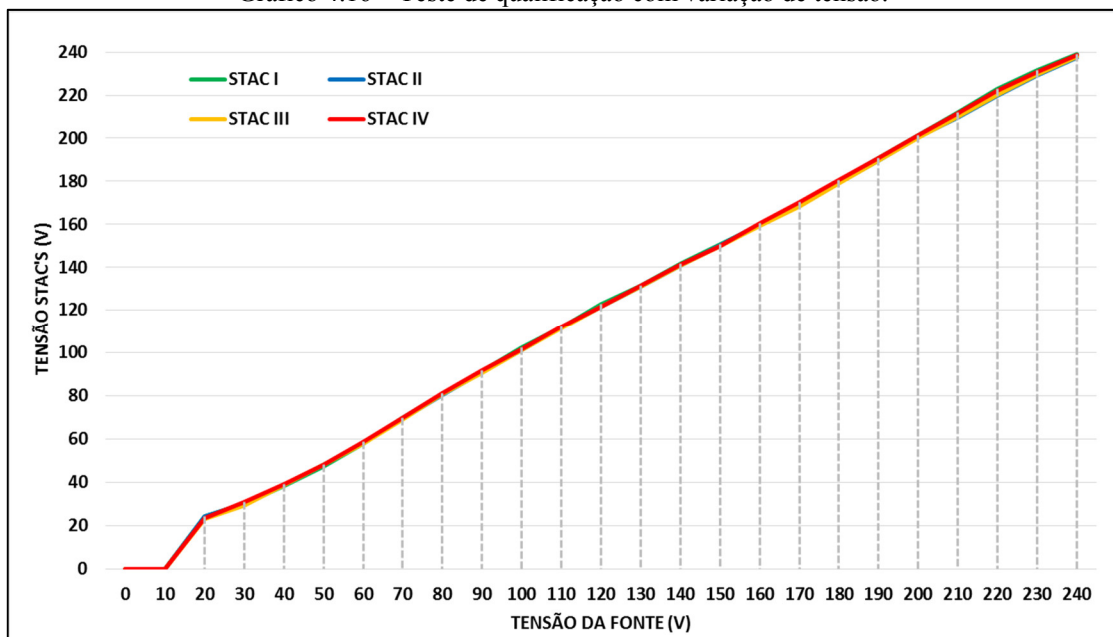
Já nas linhas 18 a 22 estão as Equações 4.2 a 4.5 e na linha 25 a Equação 4.6 que foram descritas na seção 4.4.1.

4.4.3 Procedimentos e Resultados Obtidos

4.4.3.1 Simulação de fundo de escala

Neste teste variou-se a tensão na fonte AC de 0 V a 250 V, em intervalos de 05 V. O resultado geral pode ser visto no Gráfico 4.16.

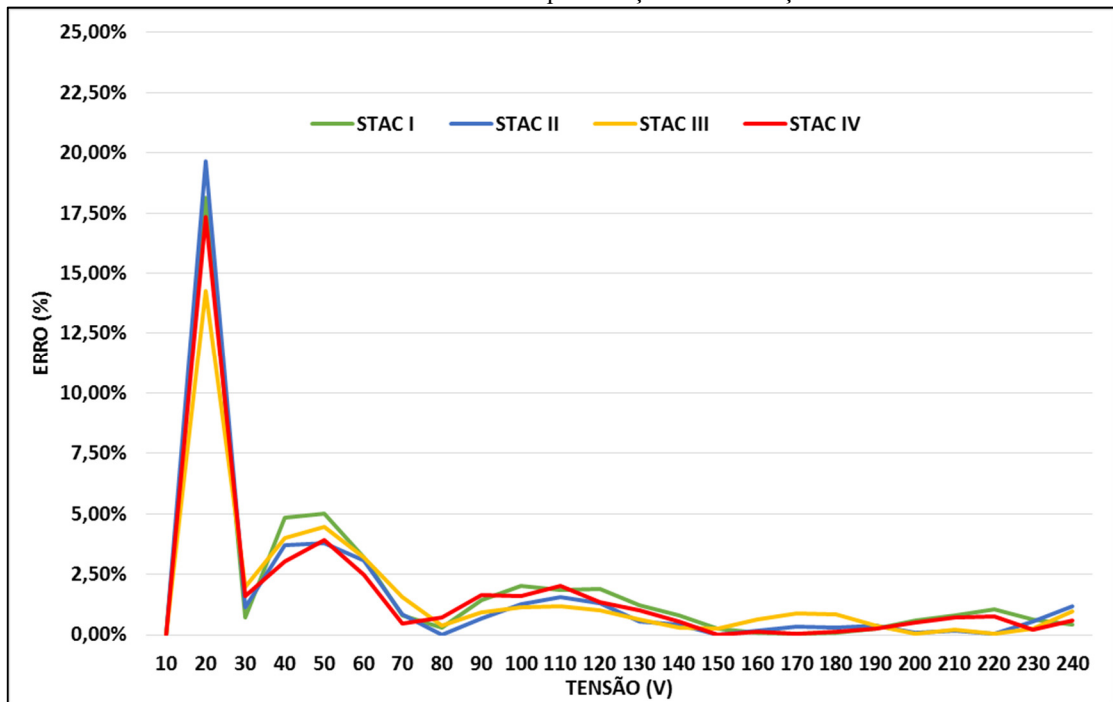
Gráfico 4.16 – Teste de qualificação com variação de tensão.



Fonte: Autoria Própria.

Nas medições exibidas no Gráfico 4.16, o STAC II apresentou maior erro percentual, 19,65% para 20 V, no entanto, na faixa de operação do MIT – 220 V, os STAC's apresentaram erros, mostrados no Gráfico 4.17, menores que 2%. Os valores de tensão e erros dos STAC's para a faixa de 220 V estão discriminados na Tabela 4.5.

Gráfico 4.17 – Erros no teste de qualificação com variação de tensão.



Fonte: Autoria Própria.

Tabela 4.5 – Comparativo entre as medições, do teste com variação de tensão, para 220 V

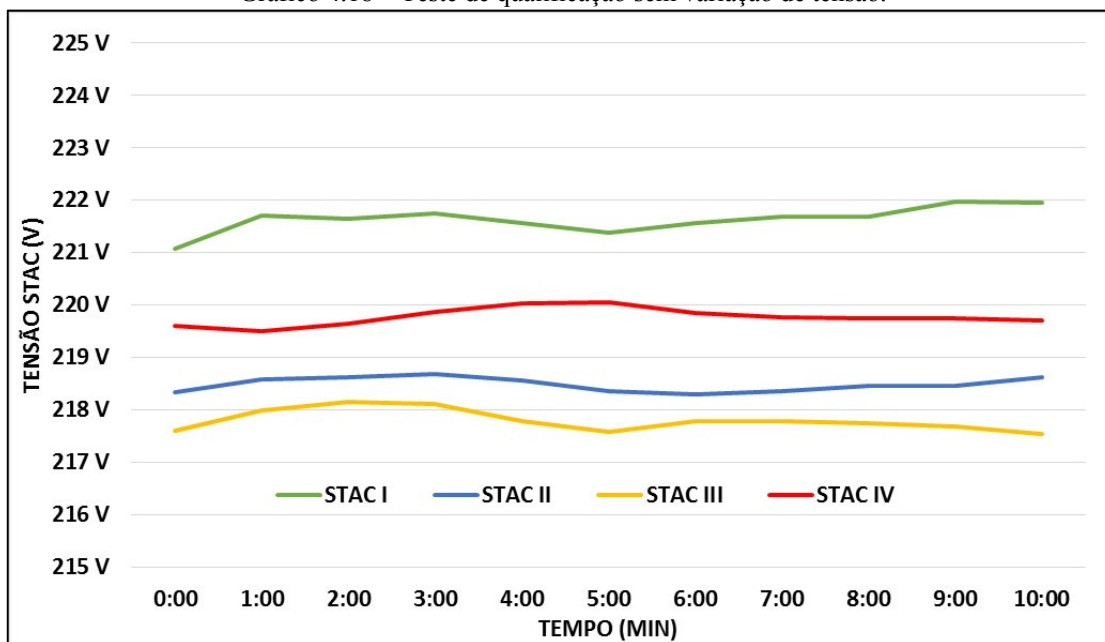
	STAC I	STAC II	STAC III	STAC IV
Tensão Sensor (V)	222,8	219,88	220,24	222,13
Erro (%)	1,04	0,05	0,06	0,74

Fonte: Autoria Própria.

4.4.3.2 Simulação de regime permanente

Nesta simulação os STAC's mediram os valores de tensão da fonte AC por um período de 10 minutos sem interrupções. No Gráfico 4.18 são apresentados os resultados das medições.

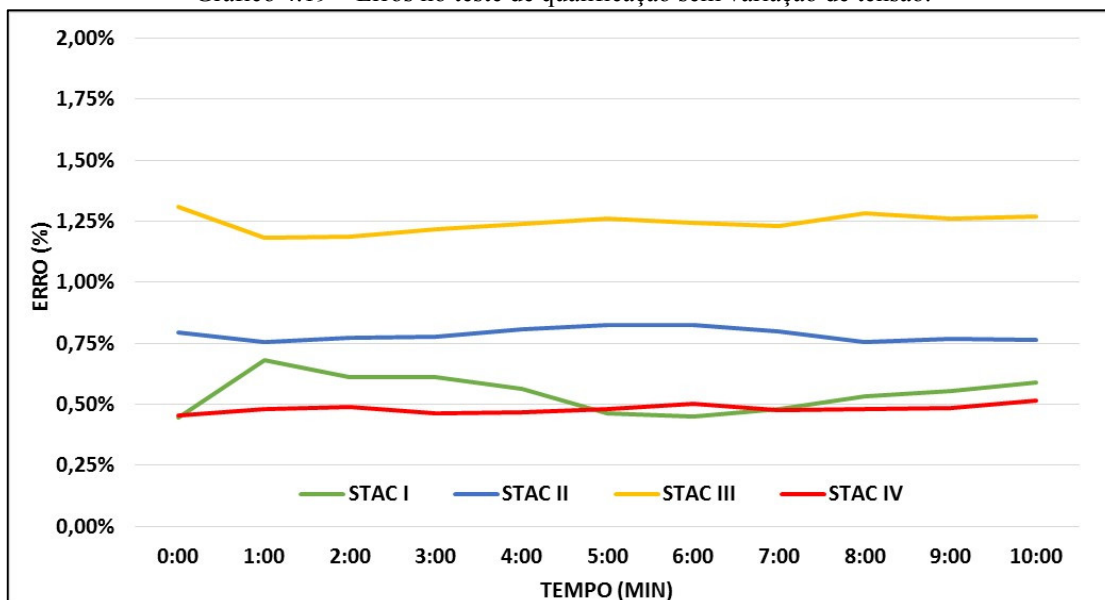
Gráfico 4.18 – Teste de qualificação sem variação de tensão.



Fonte: Autoria Própria.

Os erros percentuais para as medições sem variação de tensão são exibidos no Gráfico 4.19. Destaca-se o STAC III que apresenta o maior erro, 1,24%, e o STAC IV, que apresenta o menor, 0,48%. A Tabela 4.6 apresenta um comparativo dos desempenhos dos STAC's.

Gráfico 4.19 – Erros no teste de qualificação sem variação de tensão.



Fonte: Autoria Própria.

Tabela 4.6 – Comparativo entre as medições do teste sem variação de tensão.

	STAC I	STAC II	STAC III	STAC IV
Tensão Sensor (V)	221,64	218,49	217,8	219,78
Erro (%)	0,54	0,79	1,24	0,48

Fonte: Autoria Própria.

4.5 Conclusões dos Testes

De forma geral, as medições apresentaram resultados satisfatórios, o que indica que os sensores, assim como o Arduino, estão adequados para serem utilizados no projeto da PADMo. Contudo, alterações são necessárias. Estas alterações consistem, basicamente em:

- Melhorar a resolução das medições de velocidade;
- Corrigir os erros nas medições de temperatura;
- Realizar média matemática das medições para diminuir a variação dos resultados de tensão e temperatura.

A resolução das medições de velocidade pode ser corrigida aumentando-se o número de faixas reflexivas detectáveis pelo sensor, já o erro nas medições de temperatura, bem como a realização de média matemática, pode ser tratado com manipulação matemática no código do Arduino.

5 PLATAFORMA DE AQUISIÇÃO DE DADOS E MONITORAMENTO – PADMO

5.1 Bancada Didática de Acionamento de Máquinas Elétricas

A PADMo será aplicada em uma bancada didática, Figura 5.1, presente no Laboratório de Conversão de Energia e Máquinas da Universidade Federal do Amapá. Esta bancada é constituída por uma plataforma principal, que possui base para ensaios, circuito de proteção geral, medidor digital de força, conexões da máquina DC, conversor AC DC apropriado para acionamento da máquina DC, conexões da máquina síncrona, conexões da máquina assíncrona de rotor bobinado; máquina de corrente contínua de 1 CV; máquina síncrona de 1 CV de quatro tensões; máquina assíncrona de 1 CV com rotor bobinado de quatro tensões.

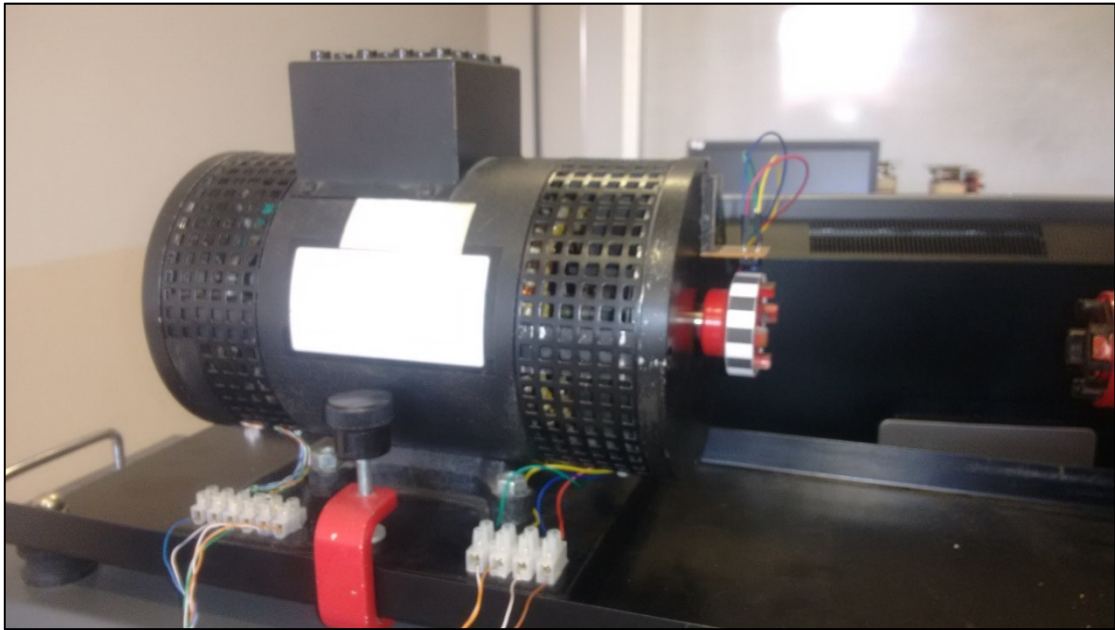
Figura 5.1 – Bancada de acionamento de máquinas elétricas.



Fonte: Acervo Próprio.

Das máquinas citadas acima, a que será utilizada para realização das medições por meio da PADMo será a máquina assíncrona de rotor bobinado mostrado na Figura 5.2. As características desta máquina são exibidas na Tabela 5.1, onde, o valor de temperatura (a vazio) foi obtido com o multímetro EM6000 em um teste que durou 20 minutos.

Figura 5.2 – Motor de indução trifásico com rotor bobinado.



Fonte: Acervo Próprio.

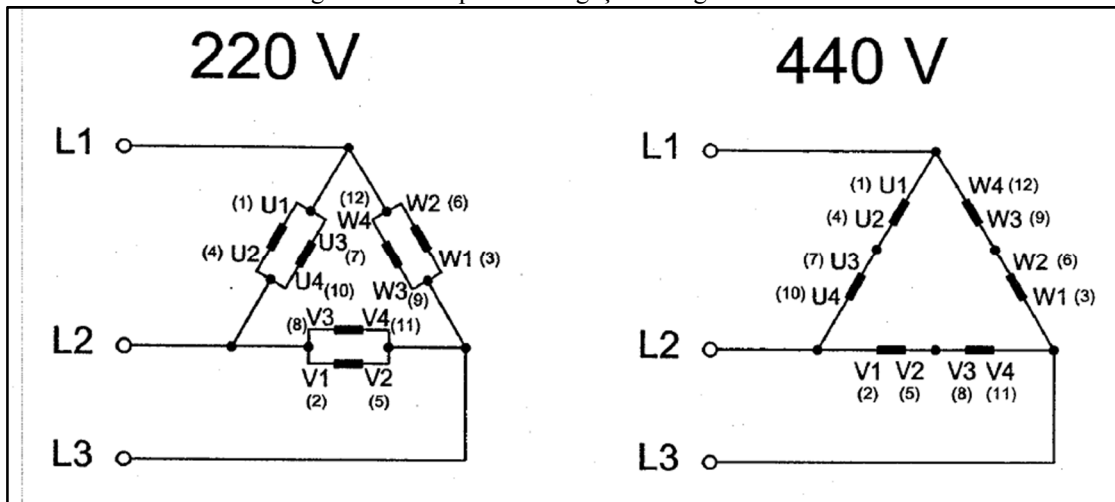
Tabela 5.1 – Valores nominais do MIT com rotor bobinado

Variáveis	Valores Nominais
Potência	1 CV
Corrente de Armadura	3,8 A
Tensão Elétrica	220 V
Velocidade de Rotação	1800 RPM
Temperatura Interna	44,5 °C

Fonte: Aatoria Própria.

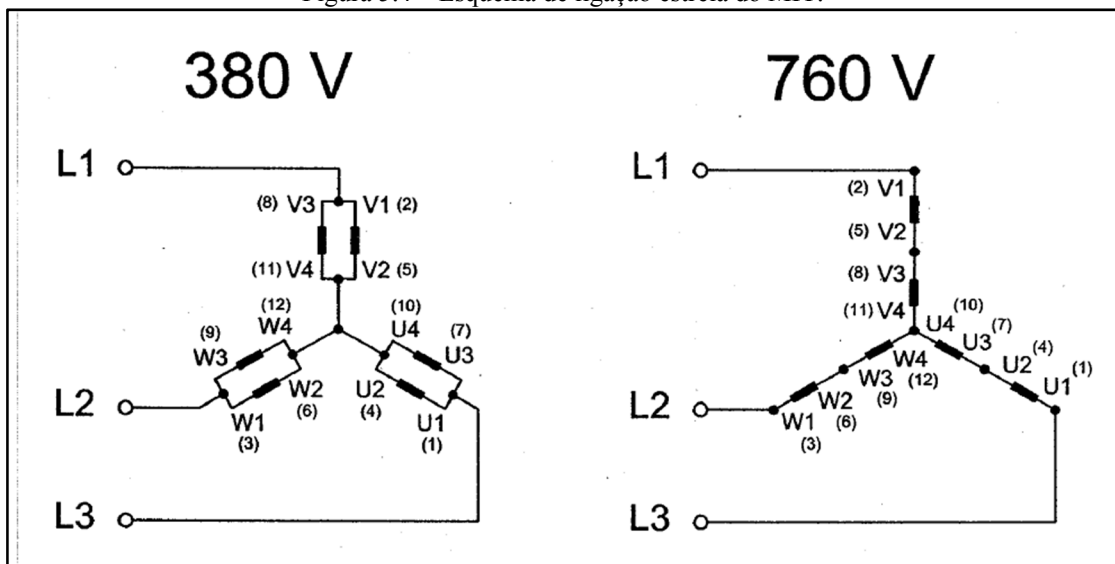
Conforme mostrado na Figura 5.1 a bancada possui terminais para realizar o fechamento das bobinas da máquina. O fechamento das bobinas pode ser realizado de acordo com os esquemas de ligação apresentados nas Figuras 5.3 e 5.4.

Figura 5.3 – Esquema de ligação triângulo do MIT.



(a) Esquema de ligação duplo-triângulo ou duplo-delta. (b) Esquema de ligação triângulo ou delta.
 Fonte: Motron Indústria de Motores Redutores (s.d).

Figura 5.4 – Esquema de ligação estrela do MIT.



(a) Esquema de ligação duplo-estrela ou duplo-Y. (b) Esquema de ligação estrela ou Y.
 Fonte: Motron Indústria de Motores Redutores (s.d).

5.2 Estrutura Física

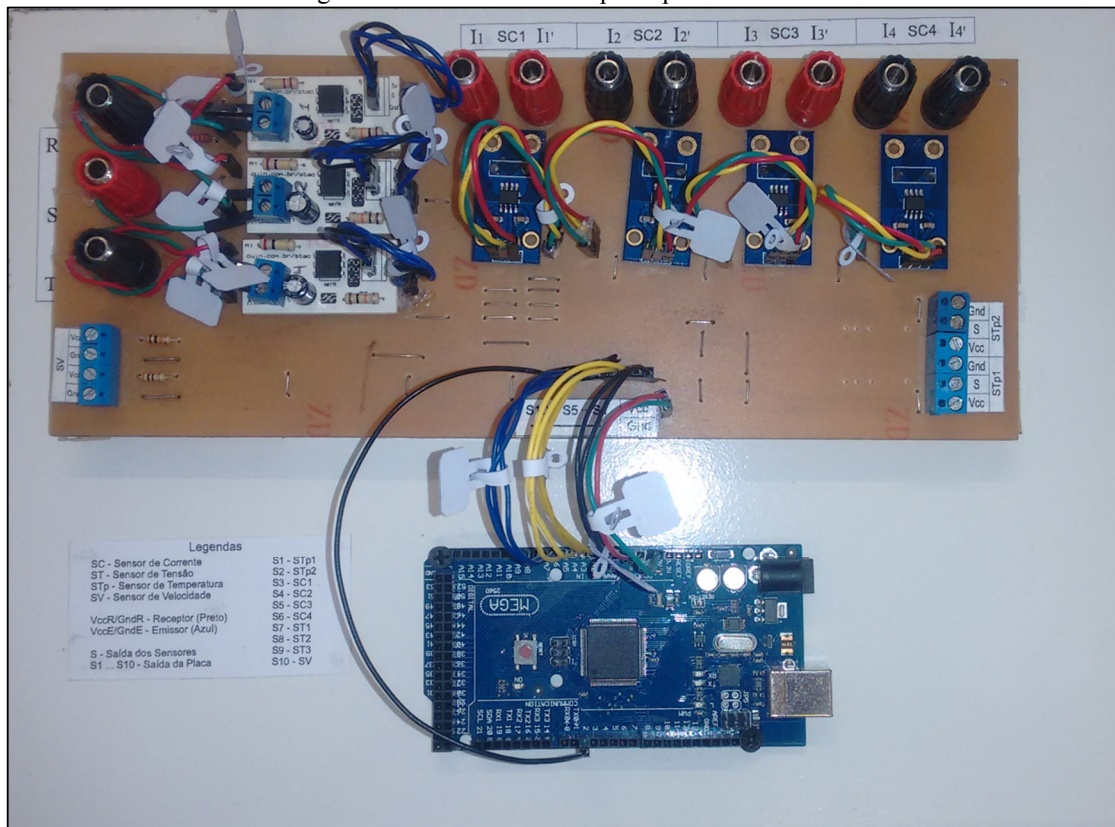
A estrutura física da PADMo consiste de três partes: uma placa de circuito impresso (PCI), fixa em um suporte móvel, que agrega os sensores de corrente (ACS712) e tensão (STAC), bem como os bornes para conexão do circuito do motor; uma que prende os sensores de temperatura (LM35) à parte interna da carcaça do motor; e uma outra que aloja o sensor de velocidade (TCRT5000) ao motor. Também, ao eixo do motor está fixada uma tira com 14 faixas reflexivas. As faixas reflexivas possuem dimensões 5,5x5,5 mm, e são separadas por uma distância de 5,5 mm. As características da tira de faixas reflexivas foram obtidas por meio

de testes nos quais a quantidade, dimensões das faixas e distância entre elas eram alteradas. Destes testes conclui-se que para faixas com dimensões menores que as citadas acima, o sensor TCRT5000 não conseguiria determinar a quantidade exata de faixas presentes na fita.

5.2.1 Estrutura Principal

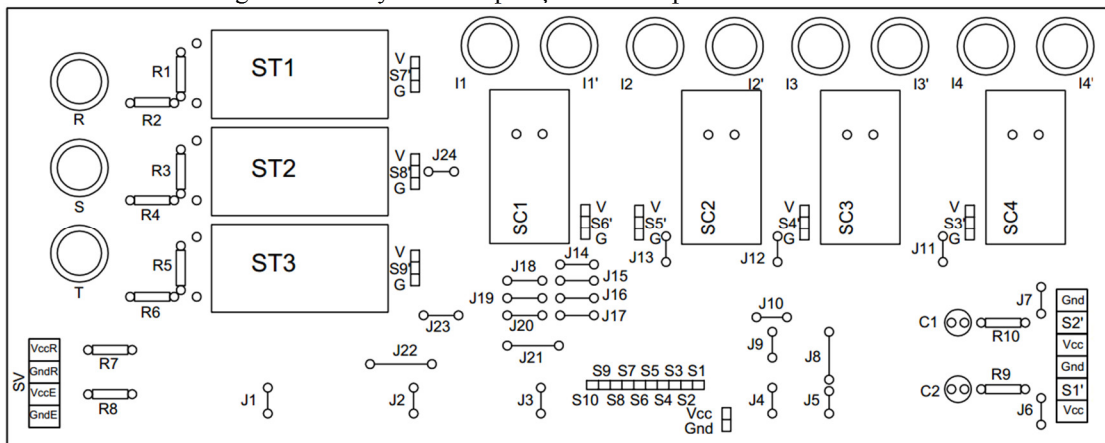
A estrutura geral da PADMo é mostrada na Figura 5.5. Já na Figura 5.6 o layout superior, ou de disposição dos componentes, é exposto, o mesmo fora desenvolvido no software AutoCAD.

Figura 5.5 – Estrutura física principal da PADMo.



Fonte: Acervo Próprio.

Figura 5.6 – Layout de disposição dos componentes da PADMo.



Fonte: Autoria Própria.

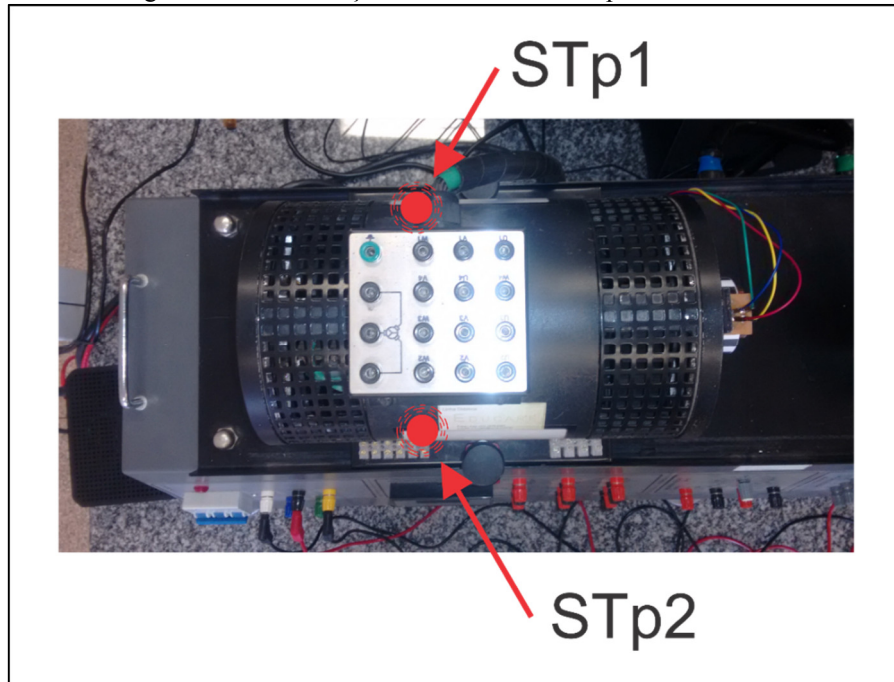
Da Figura 5.6:

- SC – Sensor de Corrente;
- ST – Sensor de Tensão;
- STp – Sensor de Temperatura;
- SV – Sensor de Velocidade;
- VccR / GndR – Alimentação e terra do receptor do sensor TCRT5000;
- VccE / GndE – Alimentação e terra do emissor do sensor TCRT5000;
- S1 a S10 – Saída da placa;
- S1' a S9' – Saída dos sensores;
- V / G – Alimentação e terra dos sensores;
- R / S / T – Fases da tensão de alimentação;
- Ix / Ix' - Bornes de entrada e saída de corrente, respectivamente;
- R1 / R3 / R5 – Resistores de 100 k Ω ;
- R2 / R4 / R6 – Resistores de 33 k Ω ;
- R7 – Resistor de 10 k Ω ;
- R8 – Resistor de 300 Ω ;
- R9 / R10 – 75 Ω ;
- C1 / C2 – Capacitores de 1 μ F;
- J1 a J24 – Jumpers.

5.2.2 Estrutura dos Sensores de Temperatura

Os sensores LM35 foram alocados no motor de indução como é mostrado na Figura 5.7. Os cabos de ligação dos sensores são conectados a uma régua sindal fixa a estrutura de suporte do motor de indução. Dessa forma os sensores podem ser conectados e desconectados à estrutura principal da PADMo sem que se perca a mobilidade da bancada didática.

Figura 5.7 – Localização dos sensores de temperatura no MIT.

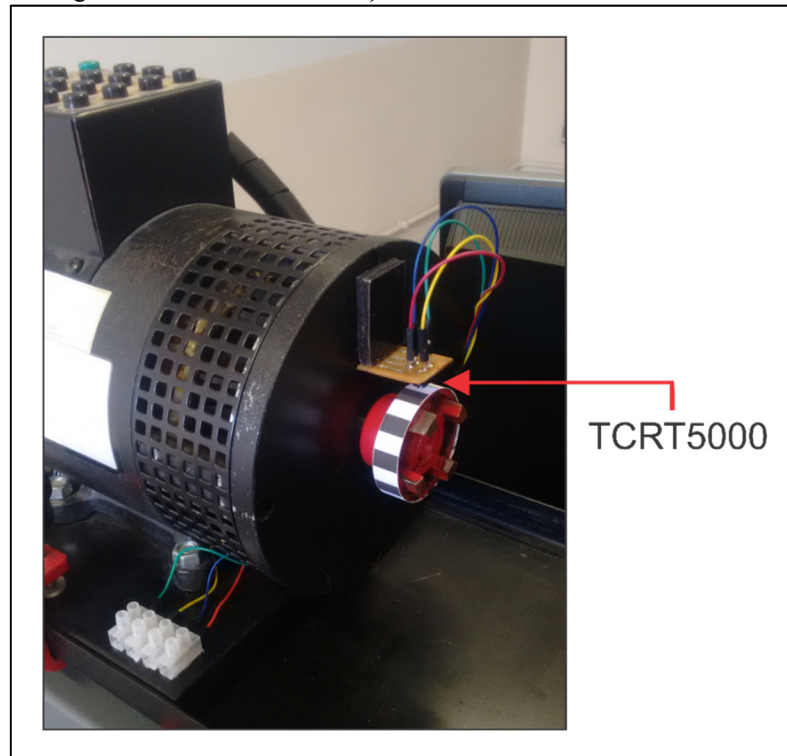


Fonte: Autoria Própria.

5.2.3 Estrutura do Sensor de Velocidade

A estrutura que mantém o sensor TCRT5000 fixado próximo ao eixo do motor de indução, bem como a tira com as faixas reflexivas, está exposta na Figura 5.8. Assim como ocorre com os sensores LM35, os cabos de ligação do TCRT5000 são conectados a uma régua sindal.

Figura 5.8 – Estrutura de fixação do sensor de velocidade no MIT.

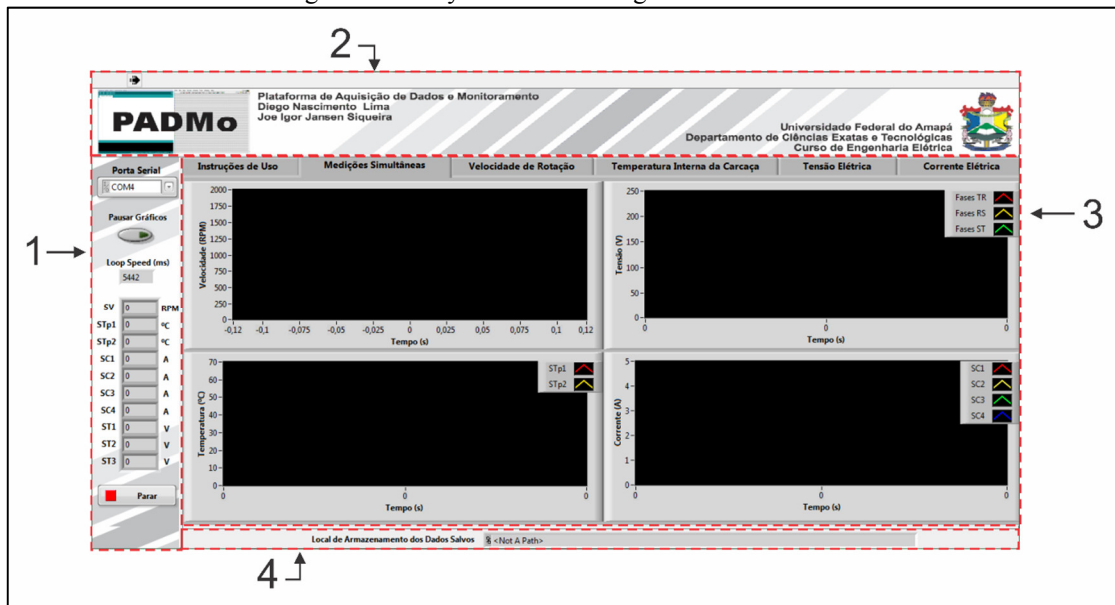


Fonte: Autoria Própria.

5.3 Interface Gráfica

A interface gráfica da PADMo, apresentada na Figura 5.9, é dividida em 4 painéis. O painel 1, ou painel esquerdo, possui alguns dos botões de operação da PADMo e alguns indicadores; o painel 2, ou painel superior, possui o botão de execução do programa e o cabeçalho do trabalho; o painel 3, ou painel central, trata das instruções de uso da PADMo e dos gráficos com as medições; e o painel 4, ou painel inferior, mostra o caminho no computador onde as medições estão salvas.

Figura 5.9 – Layout da interface gráfica da PADMo.

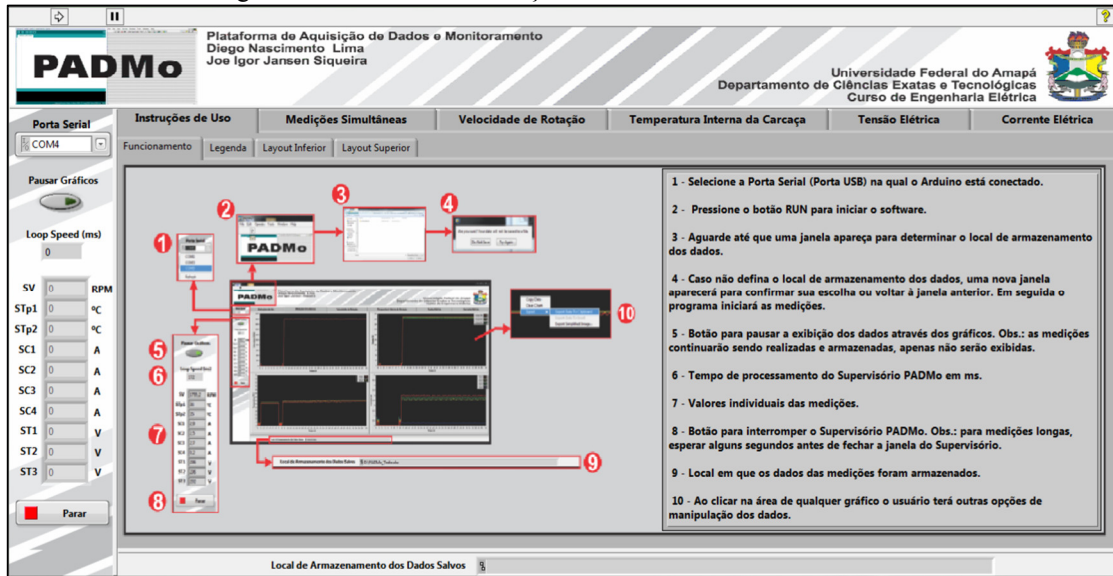


Fonte: Autoria Própria.

As instruções de uso da PADMo são mostradas na Figura 5.10 e consistem de:

1. Selecione a Porta Serial (Porta USB) na qual o Arduino está conectado.
2. Pressione o botão RUN para iniciar o software.
3. Aguarde até que uma janela apareça para determinar o local de armazenamento dos dados.
4. Caso não defina o local de armazenamento dos dados, uma nova janela aparecerá para confirmar sua escolha ou voltar à janela anterior. Em seguida o programa iniciará as medições.
5. Botão para pausar a exibição dos dados através dos gráficos. Obs.: as medições continuarão sendo realizadas e armazenadas, apenas não serão exibidas.
6. Tempo de processamento do Supervisorio PADMo em ms.
7. Valores individuais das medições.
8. Botão para interromper o Supervisorio PADMo. Obs.: para medições longas, esperar alguns segundos antes de fechar a janela do Supervisorio.
9. Local em que os dados das medições foram armazenados.
10. Ao clicar na área de qualquer gráfico o usuário terá outras opções de manipulação dos dados.

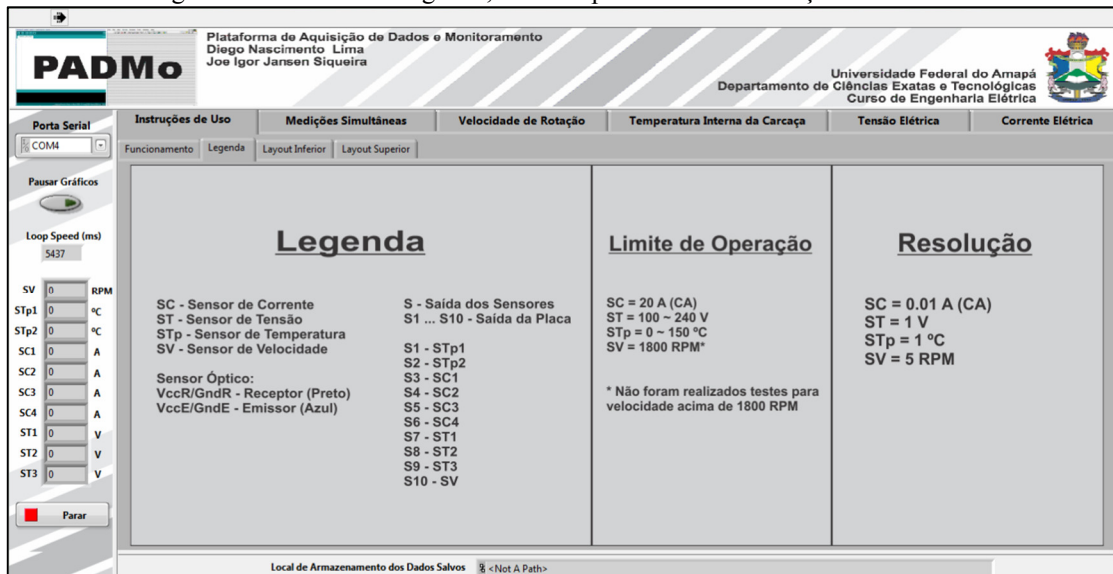
Figura 5.10 – Aba com instruções de funcionamento da PADMo.



Fonte: Acervo Próprio.

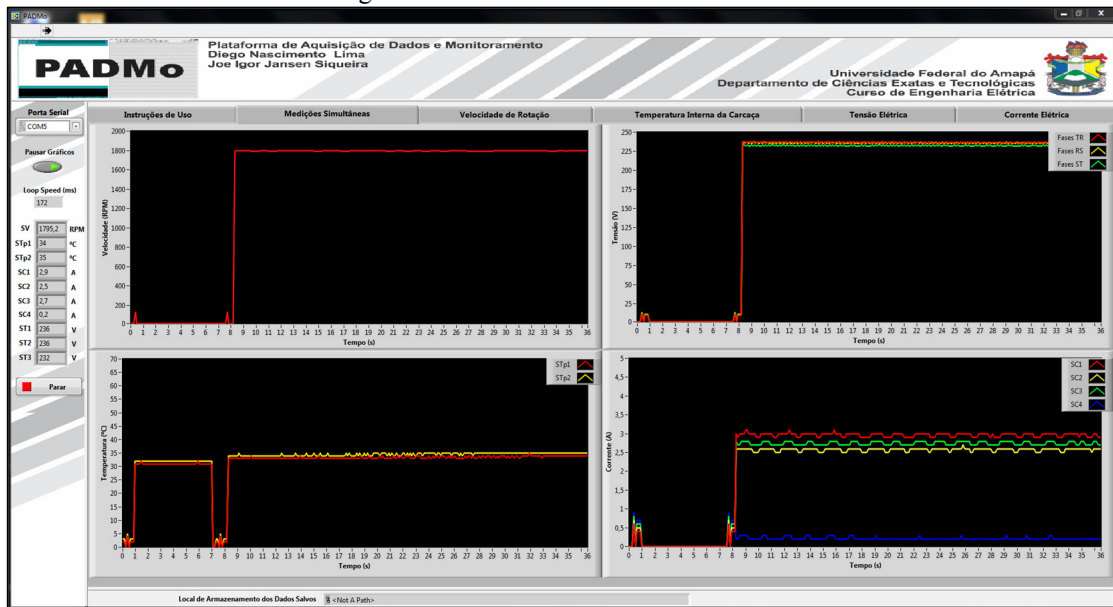
A interface gráfica também possui uma aba contendo legendas e características da PADMo, Figura 5.11, como o limite de operação e resolução da mesma. E na Figura 5.12 a PADMo é mostrada em operação.

Figura 5.11 – Aba com legenda, limites operacionais e resolução da PADMo.



Fonte: Acervo Próprio.

Figura 5.12 – PADMo em funcionamento.



Fonte: Acervo Próprio.

5.4 Código Final de Processamento

Os principais desafios na elaboração do código final de processamento da PADMo foram a junção dos códigos-fonte individuais explanados no Capítulo 4, e mostrados no Apêndice B, em um único código-fonte, e a troca de dados entre Arduino e LabVIEW devido as limitações do Comando Customizado do LINX, abordado na seção 3.4.2.1.

A solução para o problema de junção dos códigos foi a implantação de Programação Orientada a Objetos, OOP em inglês. Programação Orientada a Objetos é um modelo de linguagem de programação organizado em torno de objetos em vez de ações e dados em vez de lógica. Nesse modelo de linguagem de programação o importante são os objetos que se deseja manipular em vez da lógica requerida para os manipular. Exemplos de objetos variam desde seres humanos, descritos por nomes, endereços, etc., até pequenos dispositivos de computadores, como botões e teclas (ROUSE, 2008).

Na programação do código-fonte da PADMo foram criadas três classes para a manipulação dos sensores de temperatura, corrente e tensão. Como utilizou-se apenas um sensor de velocidade, optou-se por não desenvolver uma classe para este único objeto. Já as dificuldades na troca de dados entre Arduino e LabVIEW foram sanadas através de manipulação matemática de variáveis, conforme é explicado na seção 5.3.1 a seguir.

O intuito inicial era que todo o tratamento de dados dos sensores fosse realizado no Arduino, ao passo que o LabVIEW ficaria encarregado apenas de apresentar e armazenar os

valores medidos das variáveis do motor. Entretanto, devido a limitações do Comando Customizado do LINX, teve-se que dividir o tratamento dos dados em duas etapas: a primeira é realizada pelo Arduino e a segunda pelo LabVIEW. As programações completas desenvolvidas para o Arduino e LabVIEW estão presentes no Apêndice C e D, respectivamente.

5.4.1 Programação Arduino

Nas linhas 13 e 14 estão a biblioteca utilizada para estimar tempo e a variável que armazena o valor estimado. A linha 20 trata do vetor, ou array, que armazena os dados processados dos sensores para que estes sejam, posteriormente, enviados ao LabVIEW. Já da linha 23 a 27 são apresentadas as variáveis utilizadas na medição de velocidade. A variável `Detecoes` é a encarregada de armazenar o número de vezes que as faixas reflexivas são detectadas pelo sensor. A variável `timeOn` é o intervalo de tempo, em milissegundos, a ser comparado com a variável `timeElapsed`. Já a função das variáveis `Detec1` a `Detec3` será abordada mais à frente.

A parte inicial, linhas 32 a 50, da classe `Temperatura`, assim como ocorre com as demais classes, trata dos elementos necessários para a implementação de uma classe. Esses elementos consistem do cabeçalho da classe, onde as variáveis utilizadas são declaradas, e o construtor, no qual as variáveis declaradas são inicializadas.

A função `Update()` é a parte da classe responsável pelo processamento dos dados dos sensores. No caso dos sensores de temperatura, o processamento dos dados é idêntico ao exposto na seção 4.1.2, com exceção da média quadrática realizada e o armazenamento dos dados processados no vetor `Dados`.

O processamento realizado pela classe `Corrente` fora descrito na seção 4.3.2, com o diferencial apresentado na linha 98, onde os valores processados são multiplicados por 10 para contornar a limitação de uso de valores inteiros pelo Comando Customizado do LINX. Essa manipulação só é válida para valores de corrente de até 25,5 A, já que quando multiplicado por 10 o valor se tornará 255 que é o valor máximo que o Comando Customizado pode enviar para o LabVIEW.

A classe `Tensão` funciona de forma semelhante ao código apresentado na seção 4.4.2, porém as equações das curvas características dos sensores, assim como a equação de conversão da tensão no divisor de tensão para a tensão da fonte AC, Equações 5.1 a 5.6, são diferentes. A forma como as novas equações foram obtidas é idêntica a descrita na seção 4.4.1, contudo, os testes foram realizados apenas para os STAC 1, 2 e 4, pois estes apresentaram melhores

resultados nos testes de qualificação. Outra mudança é que agora há uma equação de conversão da tensão do divisor para a tensão na fonte AC para cada sensor, diferente do que ocorria no código de qualificação, onde existia uma equação geral para todos os sensores. Essa mudança ocorreu em virtude da diferença comportamental dos sensores. Por fim, optou-se por limitar a tensão que seria medida para a partir de 100 V, já que os sensores estão operando em suas zonas de maior linearidade nessa faixa de tensão.

- ST1 / STAC1:

$$VDiv_1 = -0,4803x^4 + 3,9407x^3 - 14,631x^2 + 68,535x + 30,12 \quad (5.1)$$

$$Vfinal_1 = 1,3072VDiv_1^{1,0215} \quad (5.2)$$

- ST2 / STAC2:

$$VDiv_2 = -1,1342x^4 + 9,1582x^3 - 29,449x^2 + 87,867x + 24,123 \quad (5.3)$$

$$Vfinal_2 = 1,309VDiv_2^{1,0238} \quad (5.4)$$

- ST3 / STAC3:

$$VDiv_3 = 0,2958x^4 - 1,1038x^3 - 3,416x^2 + 54,087x + 31,274 \quad (5.5)$$

$$Vfinal_3 = 1,2997VDiv_3^{1,025} \quad (5.6)$$

Em que,

x – tensão de saída dos sensores, V;

$VDiv_n$ – Tensão no resistor de referência do divisor de tensão, V; $n = 1,2,3$;

$Vfinal_n$ – Tensão de alimentação, V; $n = 1,2,3$.

As linhas 166 a 176 criam instâncias das classes descritas acima. Cada instância corresponde a um sensor. De forma simplificada, pode-se dizer que as instâncias servem para iniciar o uso das classes.

A função `Interrupcao()` é a função que é executada quando o sensor detecta as faixas reflexivas. A contagem das faixas será realizada somente dentro de um intervalo de

tempo definido por `timeOn`, dessa forma não há como ocorrer medições errôneas. Já a inicialização do Comando Customizado e da interrupção externa é mostrada nas linhas 189 e 199, respectivamente. Para a inicialização do Comando Customizado é necessário o número do comando, que varia de 0 a 15, e um nome de indentificação.

A parte inicial do código na função `void loop()`, linhas 208 a 220, finaliza o processamento dos dados do sensor de velocidade. Um problema dos testes de qualificação do sensor de velocidade era a resolução de 60 RPM. Esse problema era decorrente do fato de ser usada apenas uma faixa reflexiva, logo, para solucioná-lo bastava aumentar o número de faixas reflexivas, neste caso, 14. Com as 14 faixas a resolução da medição de velocidade passa a ser de 4,3 RPM. O que significa que em 60 segundos, ou 1 minuto, as 14 faixas são detectadas 4,3 vezes. Assim, se as detecções das faixas forem somadas durante um segundo e depois divididas por 14, obtém-se a velocidade de rotação do eixo motor em RPS.

Como o Comando Customizado só pode enviar números inteiros de 0 a 255 para o LabVIEW, não seria possível enviar diretamente as medições de velocidade do motor de indução, já que sua velocidade nominal é 1800 RPM. A solução deste problema é enviar para o LabVIEW os valores em RPS em vez de RPM, assim, no caso deste motor o valor enviado seria idealmente 30 RPS. Contudo, devido a resolução de 4,3 RPM os valores de velocidade medidos não seriam números inteiros, logo, também não poderiam ser enviados pelo Comando Customizado. Para solucionar este problema resolveu-se dividir as medições de velocidade em RPS em duas partes: a primeira seria a parte inteira do valor medido, variável `Detec2`, e a segunda, a parte fracionária do valor medido, variável `Detec3`. Nas linhas 218 e 219 a contagem das faixas e de tempo são, respectivamente, reinicializadas.

Na parte final do código na função `void loop()`, linhas 222 a 232, as instâncias chamam a função `Update()` de suas respectivas classes para realizar o processamento dos dados dos sensores.

Por fim, os dados armazenados no vetor `Dados` são então transferidos para o vetor `response` do Comando Customizado para a transferência para o LabVIEW.

5.4.2 Programação LabVIEW

A marcação número 1, conforme Apêndice D, indica a inicialização do dispositivo conectado ao LabVIEW, neste caso o Arduino. Para que a comunicação seja iniciada é necessário selecionar a porta serial na qual o Arduino está conectado. Em seguida, o bloco de VI's do LINX tentará estabelecer comunicação com o Arduino por meio do firmware instalado

no mesmo. Caso o LINX consiga a comunicação com o Arduino o programa continuará sua execução normalmente, do contrário, o programa será interrompido e uma mensagem contendo o possível erro que levou a falha na comunicação será apresentada.

A marcação número 2 trata do Comando Customizado do LINX. Esse bloco é responsável por verificar os dados do vetor `response` a cada determinado intervalo de tempo, 100 milissegundos neste trabalho, e então, envia os dados para o bloco na marcação 4. Também, é necessário definir qual número está sendo utilizado para identificar o Comando Customizado no código do Arduino, neste caso 0.

A marcação 3 mostra os blocos que contêm as abas usadas na interface gráfica da PADMo. Na marcação 4 os dados do vetor `response` são separados. Já nas marcações 5 e 6 os dados das medições de velocidade e corrente passam pela última etapa de processamento, respectivamente. Os dados de velocidade que chegam separados ao LabVIEW são unidos e convertidos para RPM, enquanto que os dados de corrente são divididos por 10 para voltarem a sua forma original.

Na marcação 7, os valores das medições, já processados, são apresentados no painel esquerdo da interface gráfica da PADMo. A marcação 8 traz a junção dos dados com características semelhantes, tensão com tensão, corrente com corrente, etc., em novos vetores para que estes possam ser apresentados nos mesmos gráficos. Além disto, os dados são enviados, individualmente, para o processamento necessário para o armazenamento dos mesmos.

Após a etapa exposta na marcação 8, os dados são apresentados em dois conjuntos de gráficos, conforme é mostrado na marcação 10. O primeiro conjunto trata dos gráficos separados por diferentes abas, e o segundo, dos gráficos da aba de medições simultâneas. A exibição dos dados pode ser pausada através do botão “Pausar Gráficos” que é indicado na marcação 9. É importante salientar que este botão não interrompe o funcionamento do programa de forma definitiva, apenas cessa o fluxo de dados até os gráficos, o que gera a impressão de que o programa está parado.

Na marcação 11, o tempo de amostragem das medições é contado e então enviado para ser armazenado. Já na marcação 12, outro relógio é inserido, porém a finalidade deste é determinar o intervalo de tempo para salvamento das medições. Neste caso, os dados são salvos a cada 01 segundo.

A marcação 13 exhibe os blocos usados para determinar o tempo de processamento da PADMo. O valor determinado é exposto no painel esquerdo da interface gráfica da PADMo.

A inserção de um cabeçalho às informações salvas é realizada conforme mostrado na marcação 14, e a junção das informações individuais em um único conjunto, na marcação 15.

E na marcação 16, os valores das medições são salvos em formato .xlsx e o caminho para o local do arquivo salvo é exposto no painel inferior da interface gráfica da PADMo. Na Figura 5.13 os dados armazenados com os cabeçalhos são mostrados.

Figura 5.13 – Dados armazenados com cabeçalho em arquivo .xlsx.

	A	B	C	D	E	F	G	H	I	J	K
1	Tempo(s)	Velocidade(RPM)	Temperatura 1(°C)	Temperatura 2(°C)	Tensão 1(V)	Tensão 2(V)	Tensão 3(V)	Corrente 1(A)	Corrente 2(A)	Corrente 3(A)	Corrente 4(A)
2	5,385	0,6	2	3	0	0	0	0	0	0	0
3	6,55	0	40	40	0	0	0	0	0	0	0
4	7,558	0	39	41	0	0	0	0	0	0	0
5	8,569	0	40	40	0	0	0	0	0	0	0
6	9,589	390	40	41	0	0	0	3	2,6	2,9	0,2
7	10,63	1791	40	41	186	186	187	3	2,5	2,8	0,2
8	11,67	1791	40	41	215	218	216	3	2,6	2,8	0,3
9	12,714	1791	40	41	229	228	227	3	2,6	2,8	0,2
10	13,759	1791	40	41	232	233	231	3	2,5	2,8	0,2
11	14,799	1791	40	41	236	233	233	3	2,6	2,9	0,3
12	15,844	1791	40	41	235	236	233	3	2,5	2,8	0,2
13	16,888	1791	40	40	237	234	234	3	2,5	2,8	0,2
14	17,933	1791	40	41	236	236	234	3,1	2,6	2,9	0,2
15	18,973	1791	40	41	237	235	234	3	2,5	2,8	0,2
16	20,018	1795,2	40	41	236	236	234	3	2,6	2,9	0,3
17	21,062	1791	40	41	237	235	234	3	2,5	2,8	0,2
18	22,102	1791	40	41	236	236	234	3	2,5	2,8	0,2
19	23,147	1795,2	40	41	237	235	234	3	2,6	2,9	0,3
20	24,191	1791	40	41	236	236	234	3	2,5	2,8	0,2
21	25,232	1791	40	41	237	234	234	3	2,5	2,8	0,2
22	26,276	1795,2	40	41	236	236	234	3,1	2,6	2,9	0,3
23	27,321	1791	40	41	237	234	234	3	2,5	2,8	0,2

Fonte: Acervo Próprio.

A marcação 17 mostra o “Botão Parar” que interrompe o funcionamento do programa. O programa também é interrompido caso ocorra algum erro. Quando o programa é finalizado, os históricos de dados dos gráficos são apagados, conforme marcação 18.

Por fim, na ocorrência de algum erro ou caso o programa seja interrompido a conexão com o Arduino é cessada, de acordo com o que é exposto na marcação 19.

5.5 Condições para Funcionamento

A interface gráfica da PADMo está disponível em forma de arquivo com extensão .exe e para que este funcione em outros computadores, dois softwares da National Instruments são necessários: o NI VISA e LabVIEW Run-Time Engine. Ambos softwares podem ser adquiridos de forma gratuita no site da empresa.

O Arduino presente na PADMo já está com seu código de programação salvo em sua memória, porém, caso o usuário deseje alterar alguma parte do código será necessário ter acesso a IDE do Arduino, a biblioteca ElapsedMillis, que podem facilmente ser encontradas na internet, as bibliotecas fornecidas e firmware do LINX, que requerem a instalação do LabVIEW e LINX no computador.

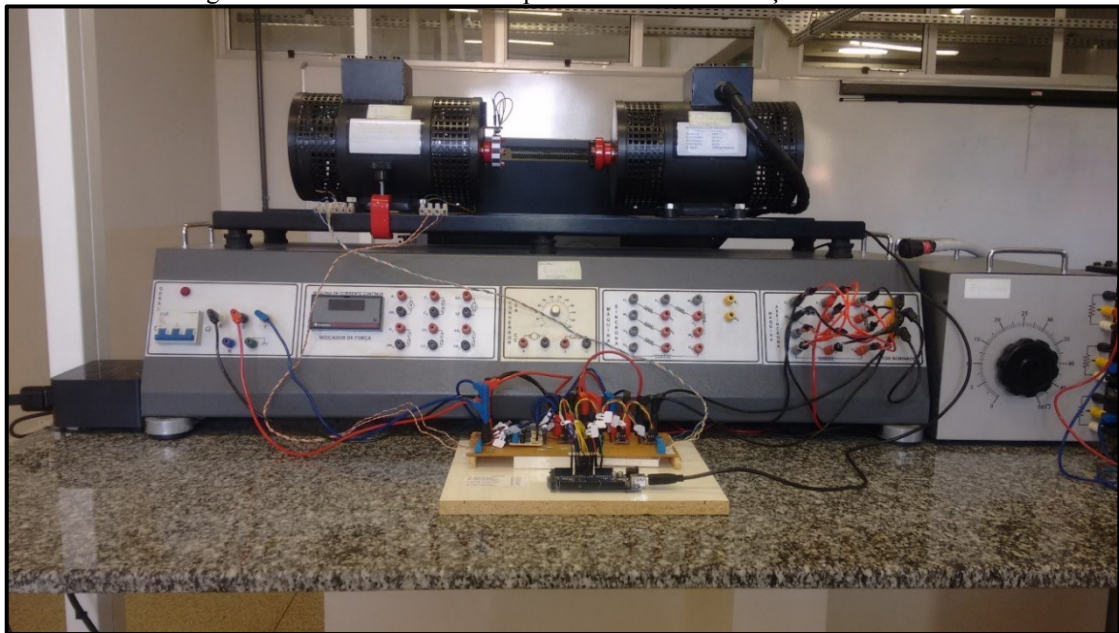
6 VALIDAÇÃO DA PADMO

Após a etapas de programação e montagem da estrutura física que abriga a ferramenta, realizaram-se os testes para validação das medições da PADMo. Para isto, além da PADMo, foram utilizados multímetros EM6000 para a medição dos valores de tensão, corrente e temperatura do MIT de rotor bobinado presente no Laboratório de Conversão de Energia e Máquinas, e para a medição da velocidade de rotação, utilizou-se o tacômetro DT-2236C.

Os testes de medição de tensão, corrente e temperatura tiveram uma duração de 30 minutos para cada sensor, enquanto que para a velocidade de rotação a duração foi de 10 minutos.

A Figura 6.1 apresenta a conexão da PADMo junto a bancada de acionamento, que foram utilizadas nos testes de validação.

Figura 6.1 – Estrutura utilizada para os testes de validação da PADMo.



Fonte: Acervo Próprio.

6.1 Medição da Temperatura Interna da Carcaça do Motor

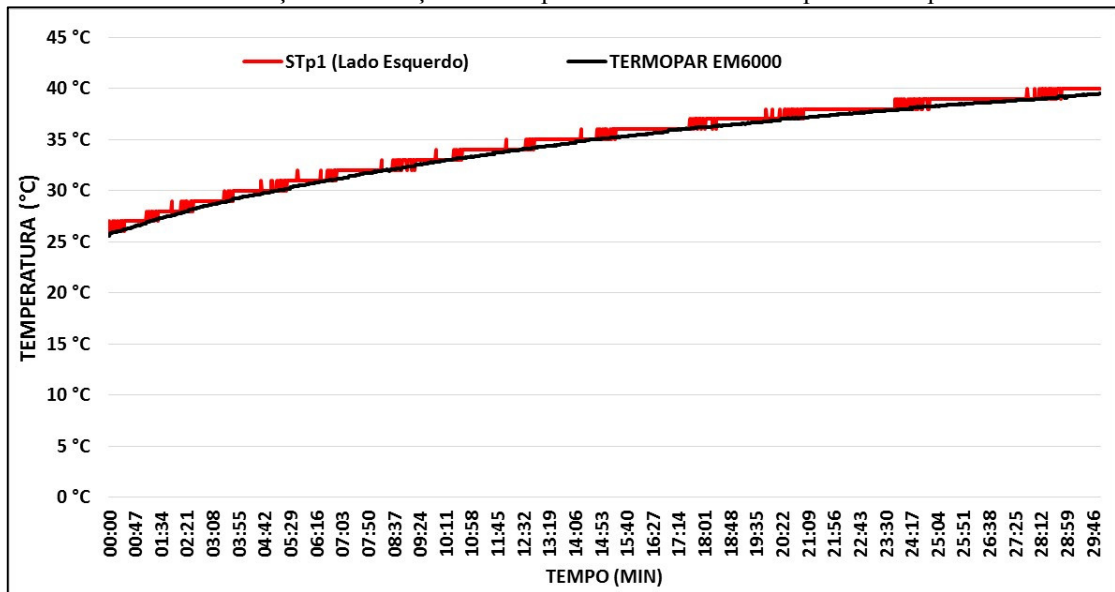
Neste teste mediu-se a temperatura interna da carcaça do motor, onde utilizou-se dois sensores LM35 (STp1 e STp2), sendo o STp1 alocado no lado esquerdo da máquina e o STp2 no lado direito.

Para a validação dos dados, fez-se uso de dois multímetros EM6000 e seus sensores de temperatura (termopares), um em cada lateral do motor, assim como feito com os sensores LM35.

6.1.1 Resultados do STp1

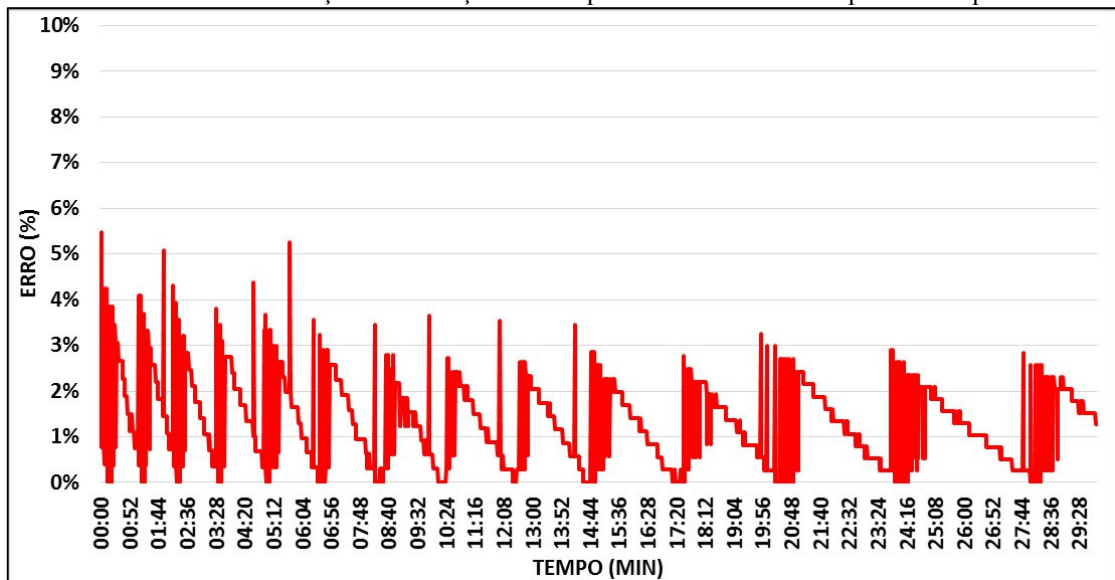
O Gráfico 6.1 exibe os dados das medições do STp1 e o Gráfico 6.2 apresenta os erros percentuais entre as medições do LM35 e do termopar.

Gráfico 6.1 – Validação das medições de temperatura da PADMo: STp1 x Termopar EM6000.



Fonte: Autoria Própria.

Gráfico 6.2 – Erros na validação das medições de temperatura da PADMo: STp1 x Termopar EM6000.

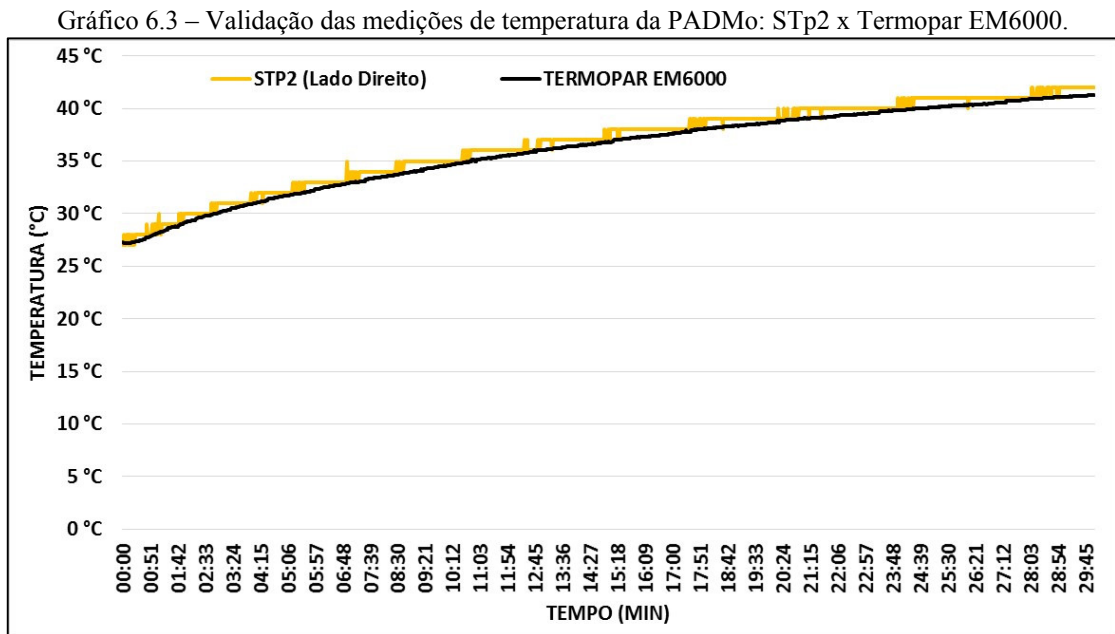


Fonte: Autoria Própria.

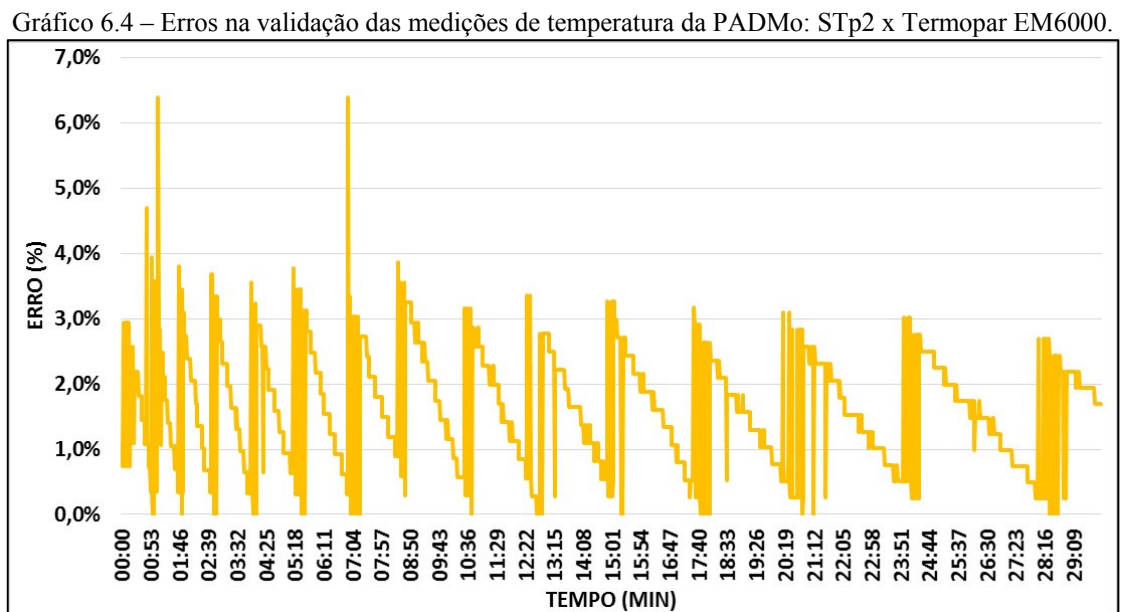
Verifica-se que, o erro percentual máximo atingiu um valor de 5,5% aproximadamente, correspondendo a 1,6 °C, contudo o erro médio percentual situou-se na faixa de 1,3%, o equivalente a 0,45 °C.

6.1.2 Resultados do STp2

Para o STp2, obteve-se os dados expostos no Gráfico 6.3 e o Gráfico 6.4 exhibe os erros percentuais do teste.



Fonte: Autoria Própria.



Fonte: Autoria Própria.

Nesta medição, o erro percentual máximo alcançou a marca de 6,4%, equivalente a 2,1 °C, enquanto que o erro médio percentual manteve-se na faixa de 1,6%, correspondendo a 0,57 °C.

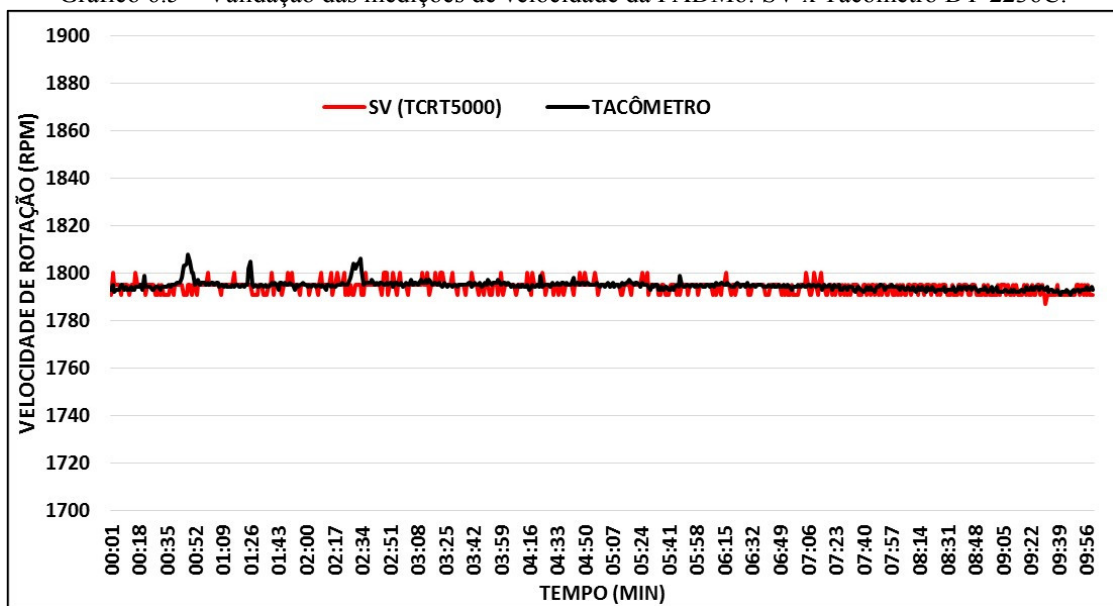
Nos testes de validação de temperatura, não foi necessário realizar nenhuma alteração no código de programação, visto que, a temperatura máxima atingida, dentre todas as aferições, fora de 45 °C, onde os erros se estabeleciam na faixa de 1,5%, cerca de 0,5 °C, aproximadamente.

6.2 Medição da Velocidade de Rotação do Eixo do Motor

A coleta dos dados de velocidade de rotação do eixo da máquina, medidos pelo tacômetro DT-2236C, foi realizada de forma manual através da gravação de um vídeo com duração de 10 minutos onde, a cada segundo, um valor de velocidade de rotação era colhido, totalizando 600 medições. A coleta manual foi necessária pois o tacômetro DT-2236C não possui um meio de comunicação com computadores, diferente do que acontece com o multímetro EM6000.

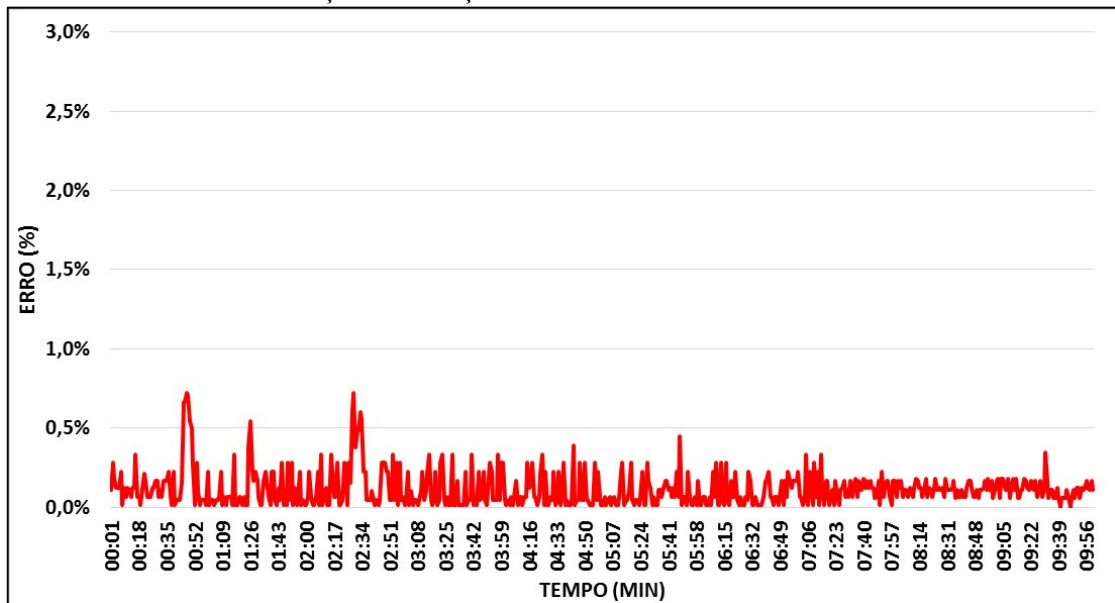
O resultado das medições da PADMo e tacômetro está apresentado no Gráfico 6.5, os valores de erros percentuais estão denotados no Gráfico 6.6.

Gráfico 6.5 – Validação das medições de velocidade da PADMo: SV x Tacômetro DT-2236C.



Fonte: Autoria Própria.

Gráfico 6.6 – Erros na validação das medições de velocidade da PADMo: SV x Tacômetro DT-2236C.



Fonte: Autoria Própria.

A utilização da tira com 14 faixas reflexivas reduziu a resolução inicial de 60 RPM, descrita nos testes de qualificação, para cerca de 4,3 RPM, diminuindo, consideravelmente, os erros para esta variável, tendo como consequência as menores taxas de erros dentre todos os sensores da PADMo. O erro máximo entre as medições foi de 13 RPM, equivalente a 0,7%, enquanto que o erro médio atingiu a faixa de 2 RPM ou 0,1%.

6.3 Medição da Corrente Elétrica nos Enrolamentos do Motor

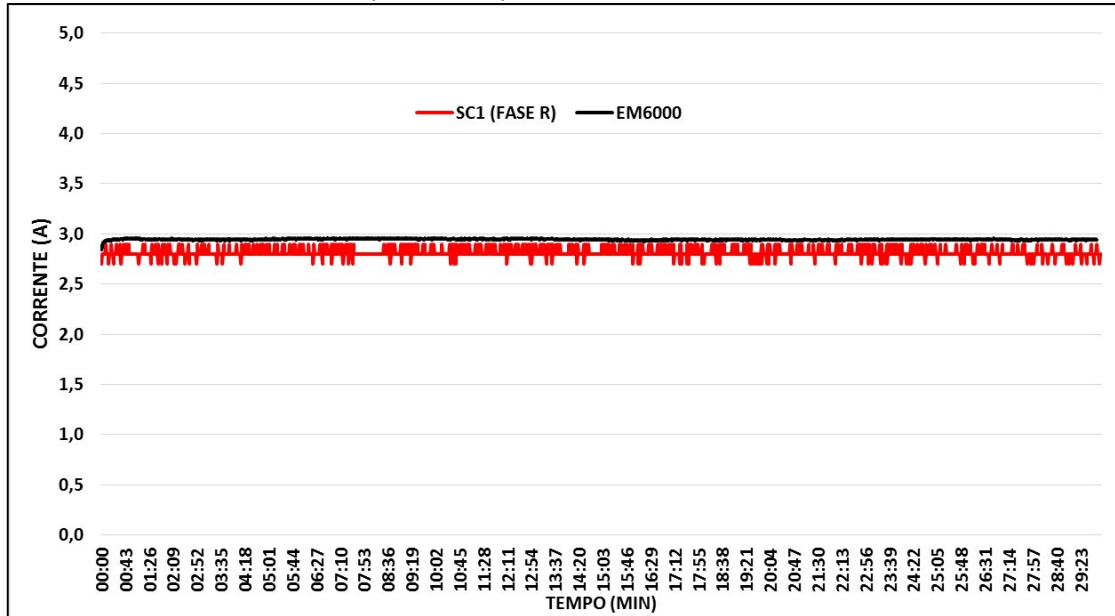
Neste teste mediu-se as correntes circulantes em cada fase da máquina, assim como a corrente no circuito do estator, individualmente, sendo que os sensores de corrente 1, 2, 3 e 4 correspondem, respectivamente, as correntes nas fases R, S, T e corrente no rotor. Conectado em série aos sensores estavam multímetros EM6000.

Vale observar que, devido a grandeza dos valores de corrente medidos, qualquer diferença entre esses valores pode ocasionar um erro percentual considerável, o que não caracteriza as medições como inadequadas.

6.3.1 Resultados do SC1

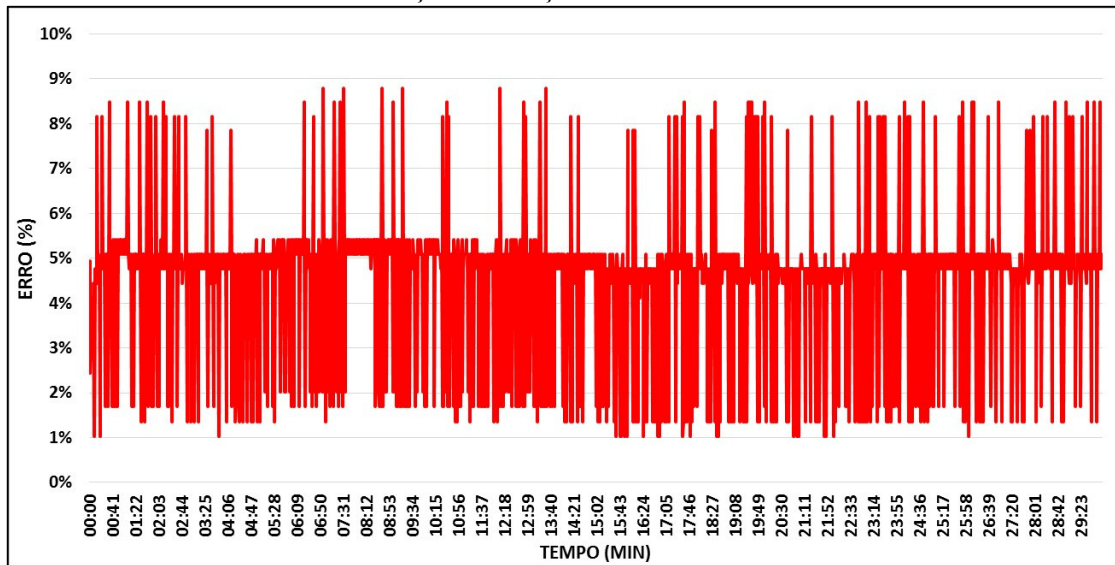
O Gráfico 6.7 exibe os dados das medições com o SC1 e o EM6000 e o Gráfico 6.8 apresenta os erros percentuais das medições.

Gráfico 6.7 – Validação da medição de corrente da PADMo: SC1 x EM6000.



Fonte: Autoria Própria.

Gráfico 6.8 – Erros na validação da medição de corrente da PADMo: SC1 x EM6000.



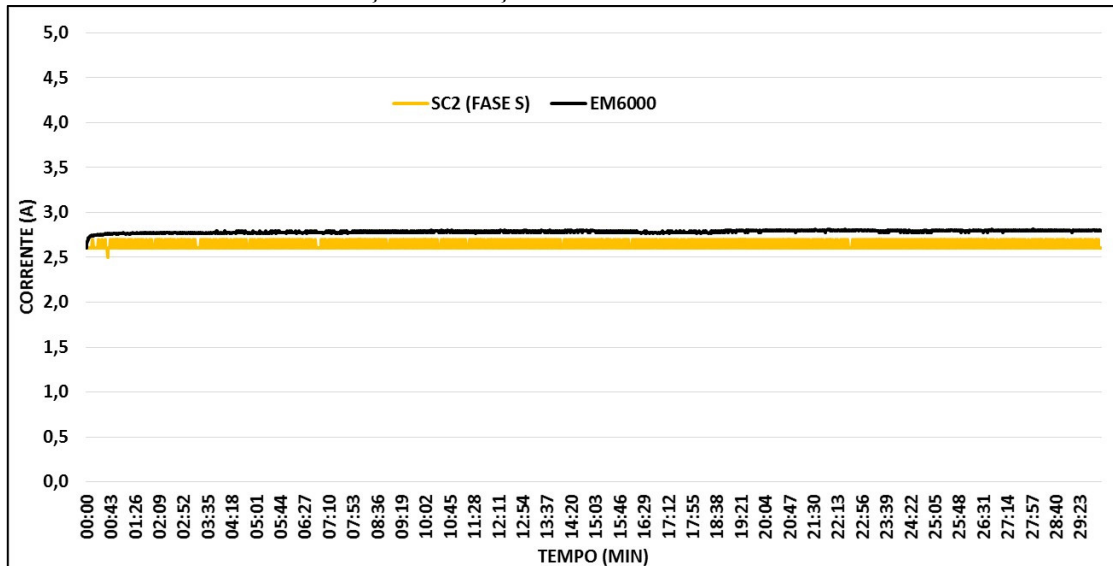
Fonte: Autoria Própria.

O erro máximo foi de 0,26 A, o equivalente a 8,8%, enquanto que o erro médio foi 0,14 A, ficando na faixa de 4,6 % de erro percentual.

6.3.2 Resultados do SC2

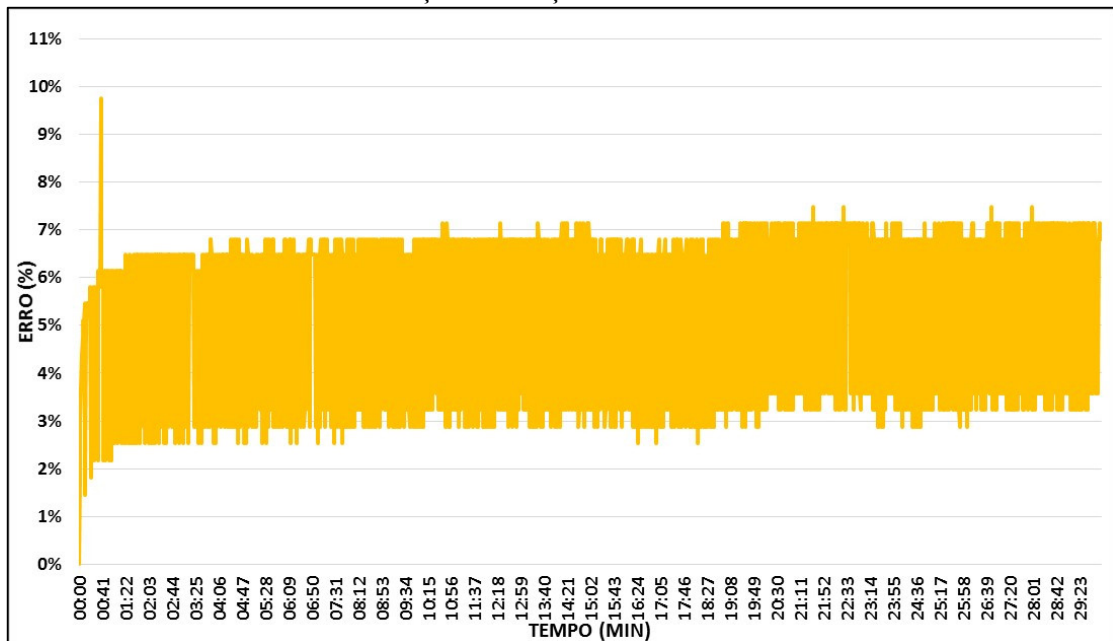
Para o SC2, os dados das medições e dos erros são mostrados, respectivamente, nos Gráficos 6.9 e 6.10.

Gráfico 6.9 – Validação da medição de corrente da PADMo: SC2 x EM6000.



Fonte: Autoria Própria.

Gráfico 6.10 – Erros na validação da medição de corrente da PADMo: SC2 x EM6000.



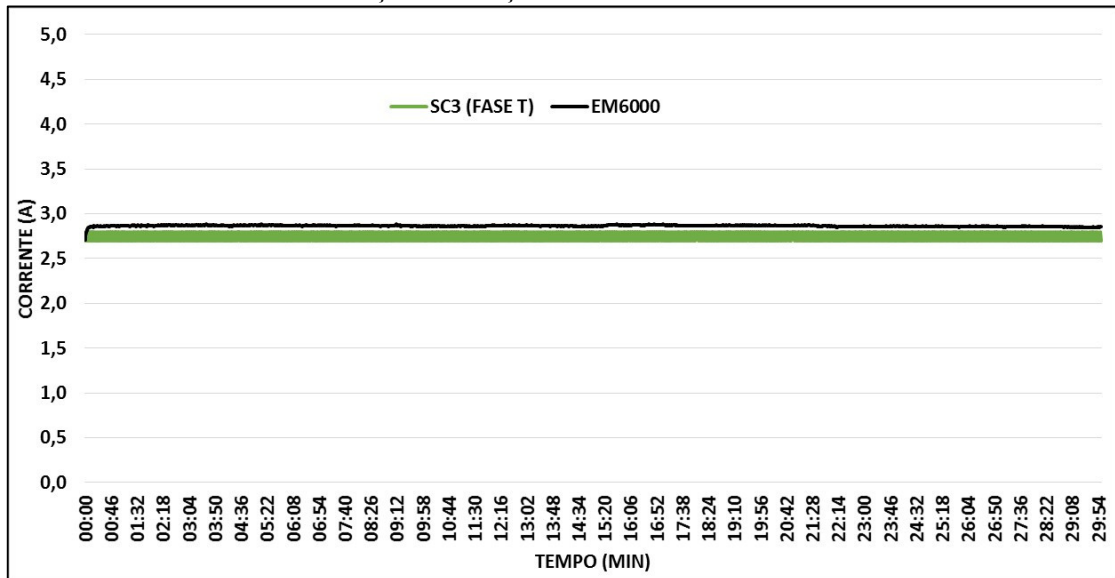
Fonte: Autoria Pópria.

Esta medição apresentou um erro máximo de 9,7%, correspondente a 0,27 A, contudo o erro médio situou-se na faixa de 5,4%, o equivalente a 0,15 A.

6.3.3 Resultados do SC3

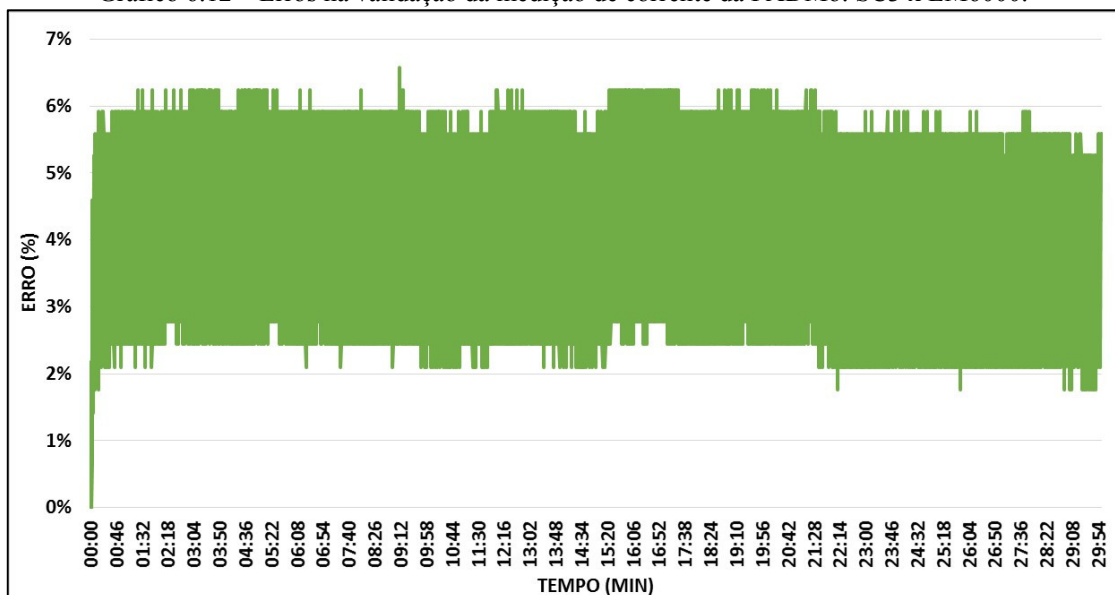
Os dados das medições do SC3 e seus erros percentuais estão dispostos nos Gráficos 6.11 e 6.12, na devida ordem.

Gráfico 6.11 – Validação da medição de corrente da PADMo: SC3 x EM6000.



Fonte: Autoria Própria.

Gráfico 6.12 – Erros na validação da medição de corrente da PADMo: SC3 x EM6000.



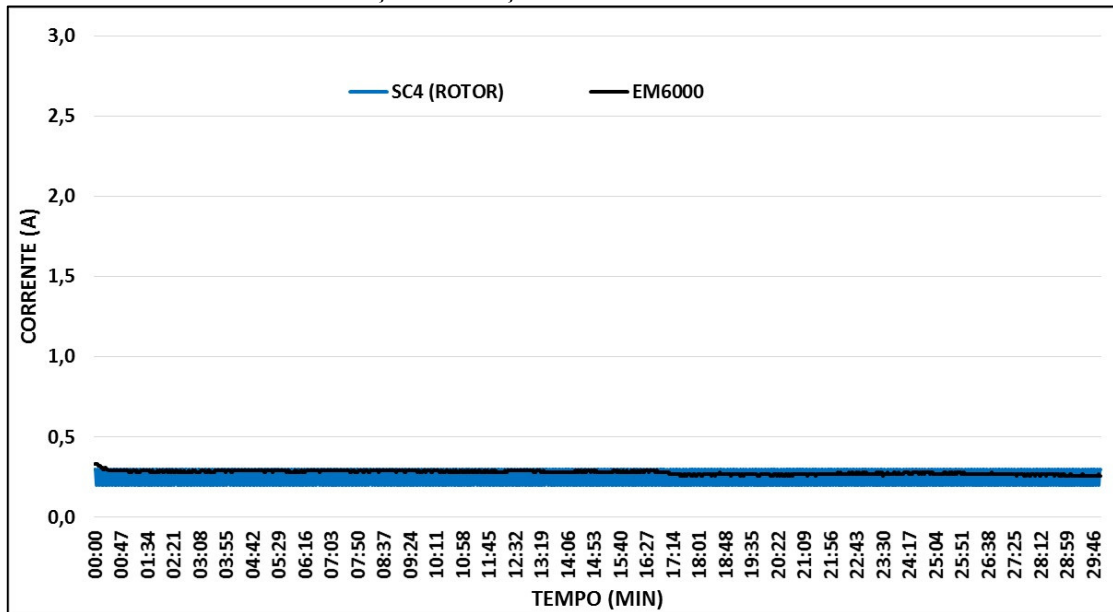
Fonte: Autoria Própria.

Este sensor apresentou as menores taxas de erros máximo e médio, tendo um pico de 6,6%, equivalente a 0,19 A, e uma média percentual de 3,9%, correspondente a 0,11 A.

6.3.4 Resultados do SC4

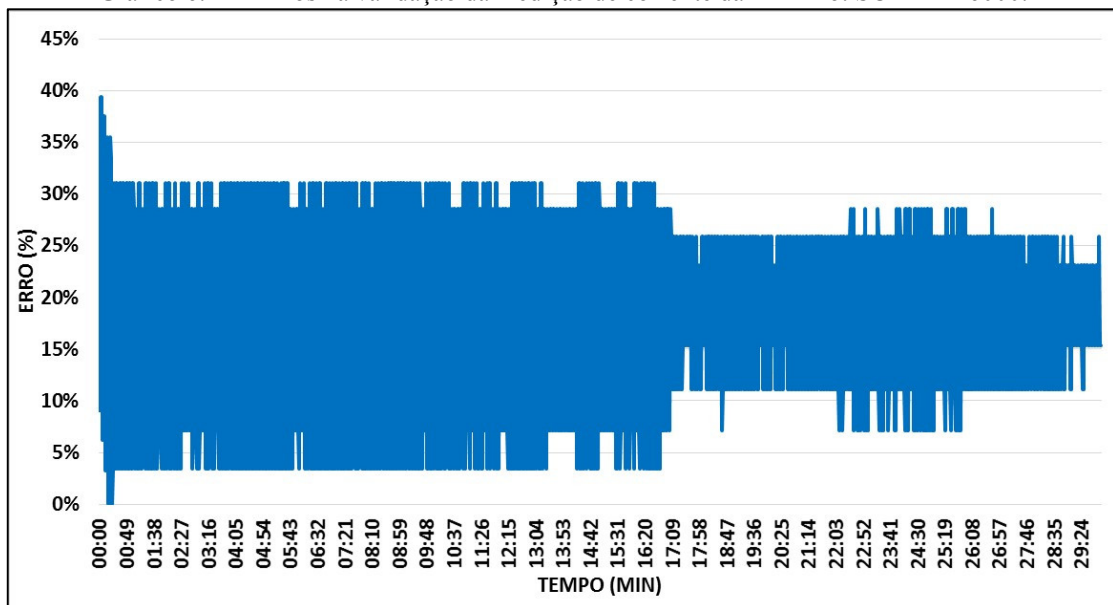
Os dados das medições e erros percentuais do SC4 são denotados nos Gráficos 6.13 e 6.14, nesta ordem.

Gráfico 6.13 – Validação da medição de corrente da PADMo: SC4 x EM6000.



Fonte: Autoria Própria.

Gráfico 6.14 – Erros na validação da medição de corrente da PADMo: SC4 x EM6000.



Fonte: Autoria Própria.

Em contrapartida ao SC3, o SC4 foi o sensor que exibiu os maiores erros percentuais máximos e médios. Em termos numéricos, o erro máximo foi de 39,4% ou 0,13 A, já o erro médio ficou situado na faixa de 15,2% ou 0,04 A.

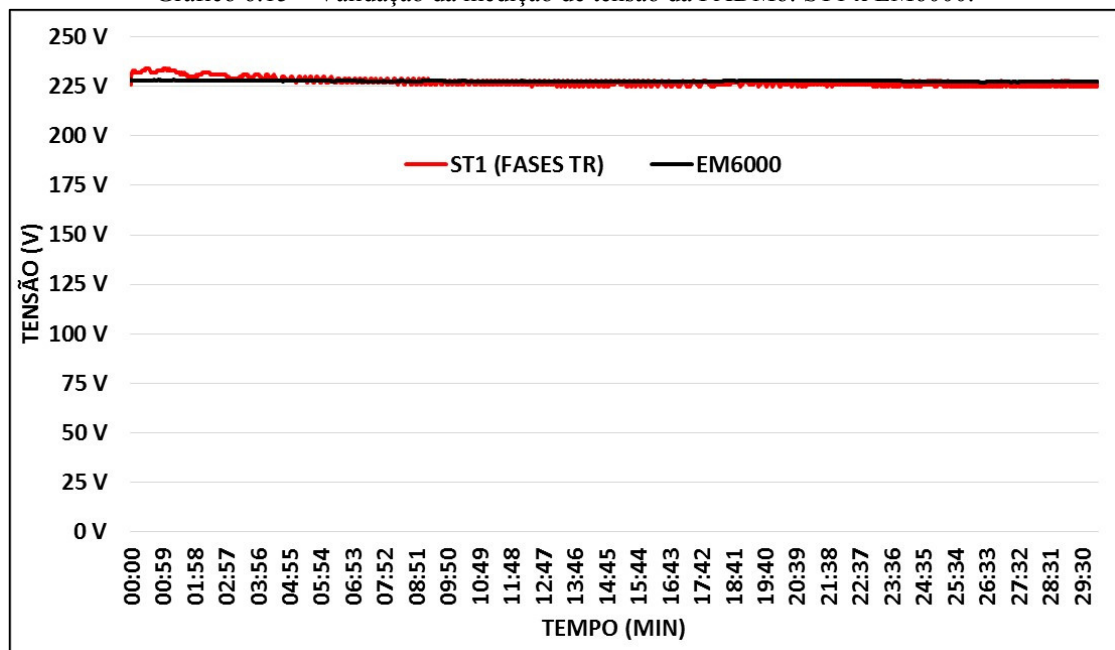
6.4 Medição da Tensão RMS de Alimentação do Motor

Foram medidas as tensões RMS entre as fases de alimentação da máquina, de modo que, o ST1 ficou responsável pela medição das fases T e R, o ST2 com as fases R e S e, por fim, o ST3 com as fases S e T. Para a validação, usou-se o multímetro EM6000.

6.4.1 Resultados do ST1

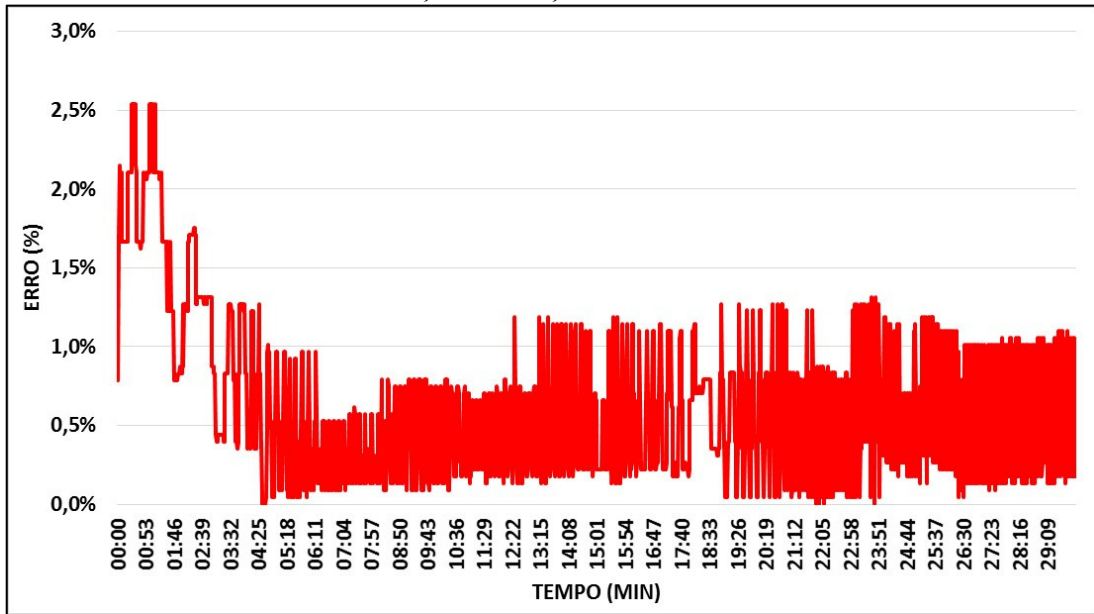
Os dados coletados referentes as medições do ST1 com o EM6000 estão expostos no Gráfico 6.15 e os erros percentuais no Gráfico 6.16.

Gráfico 6.15 – Validação da medição de tensão da PADMo: ST1 x EM6000.



Fonte: Autoria Própria.

Gráfico 6.16 – Erros na validação da medição de tensão da PADMo: ST1 x EM6000.



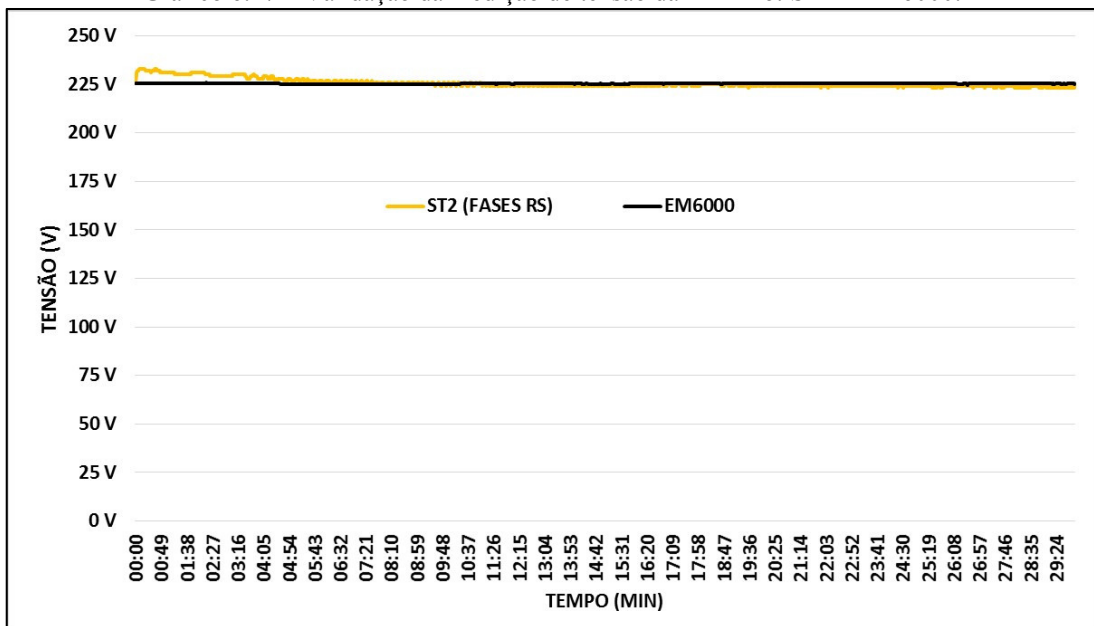
Fonte: Autorial Própria.

A taxa de erros atingiu um pico de 5,8 V, correspondendo a 2,54 %, já o erro médio se estabilizou na faixa de 1,43 V ou 0,63%.

6.4.2 Resultados do ST2

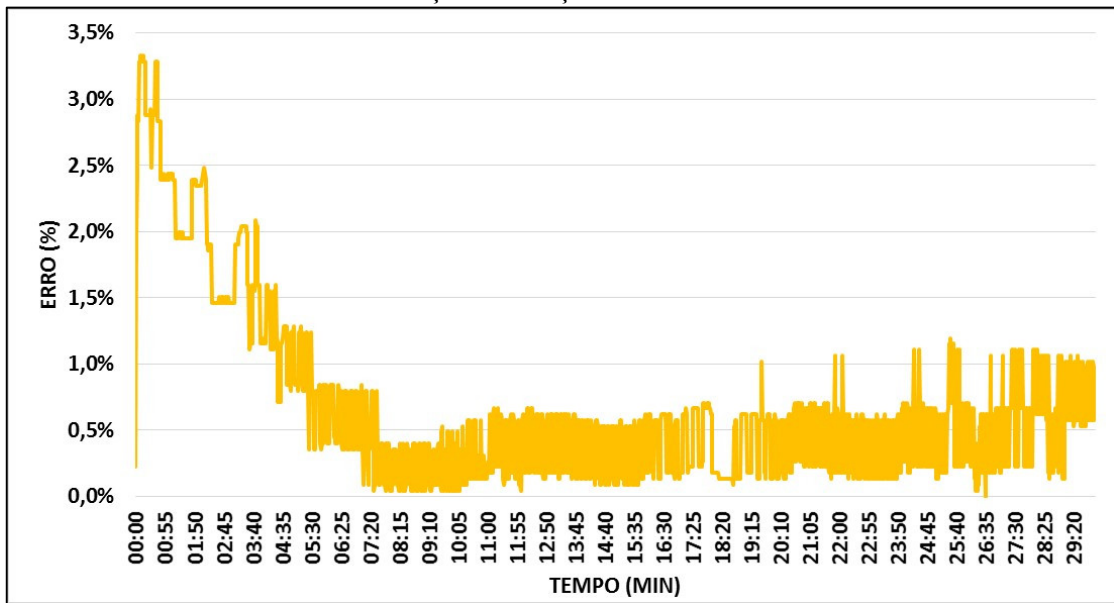
As medições com o ST2 e seus erros percentuais estão apresentados nos Gráficos 6.17 e 6.18, respectivamente.

Gráfico 6.17 – Validação da medição de tensão da PADMo: ST2 x EM6000.



Fonte: Autorial Própria.

Gráfico 6.18 – Erros na validação da medição de tensão da PADMo: ST2 x EM6000.



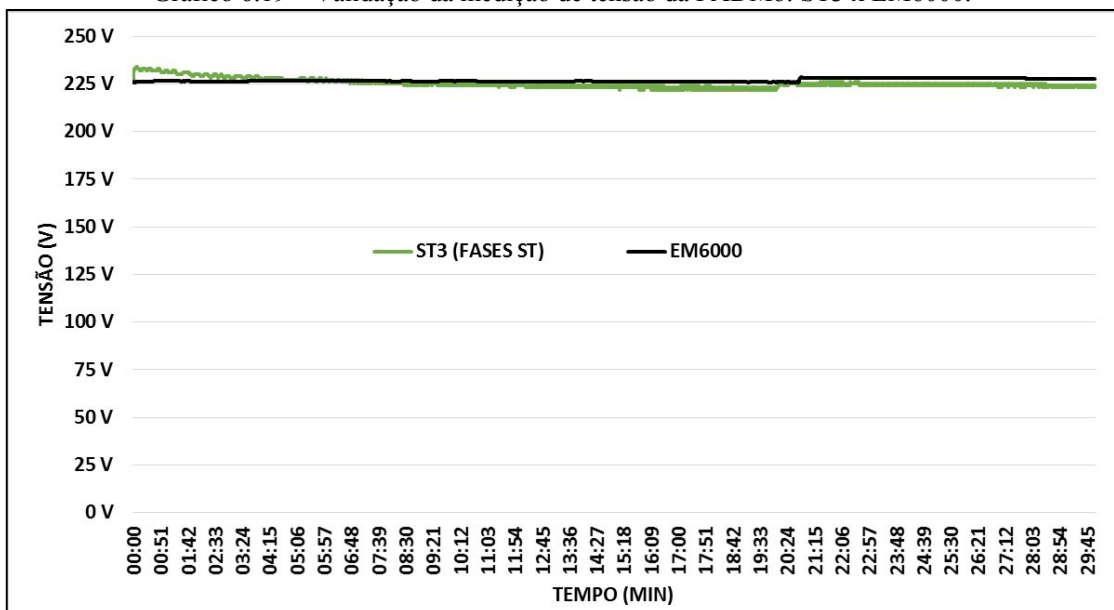
Fonte: Autoria Própria.

Este sensor apresentou um erro máximo de 7,5 V, equivalente a 3,33%, mas o erro médio ficou na faixa de 1,56 V ou 0,69%.

6.4.3 Resultados do ST3

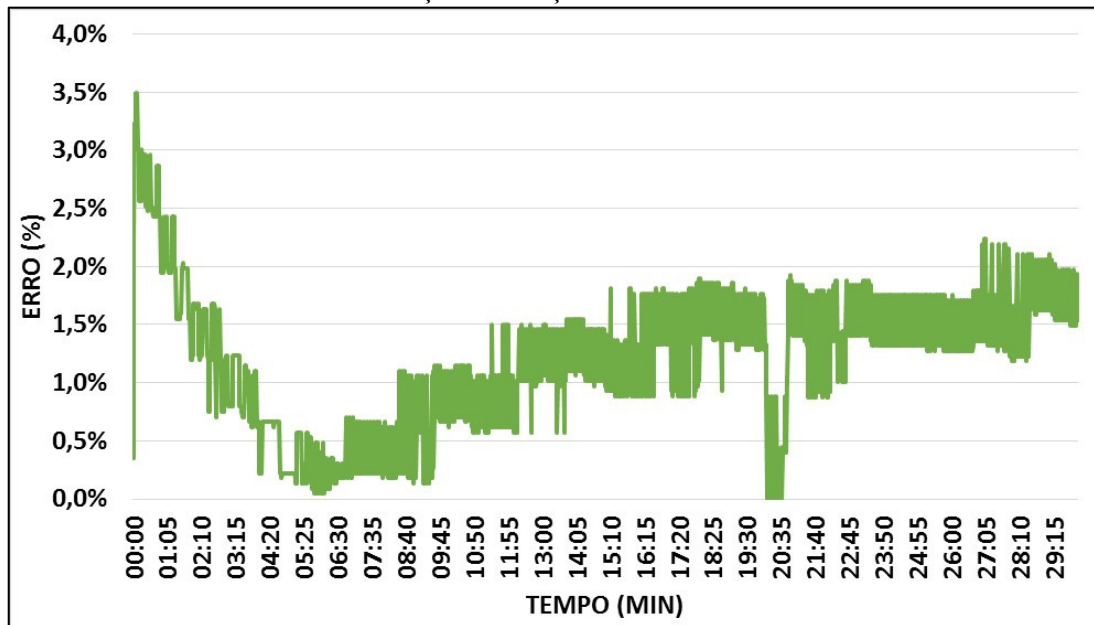
Os dados coletados com o ST3 estão expostos no Gráfico 6.19 e seus erros percentuais no Gráfico 6.20.

Gráfico 6.19 – Validação da medição de tensão da PADMo: ST3 x EM6000.



Fonte: Autoria Própria.

Gráfico 6.20 – Erros na validação da medição de tensão da PADMo: ST3 x EM6000.



Fonte: Autoria Própria.

O erro máximo para este sensor foi de 7,9 V ou 3,49 %, enquanto o seu erro médio atingiu 2,76 V, correspondendo a 1,22%.

6.5 Resumo dos Testes

Concluída a validação da PADMo, a Tabela 6.1 exibe o resumo da performance da mesma.

Tabela 6.1 – Características técnicas da PADMo.

Parâmetro	Sensor	Erro Máximo		Erro Médio		Instrumento para Validação
		Absoluto	%	Absoluto	%	
Temperatura	STp1	1,6 °C	5,5%	0,45 °C	1,3%	EM6000
	STp2	2,1 °C	6,4%	0,57 °C	1,6%	EM6000
Velocidade	SV	13 RPM	0,7%	2 RPM	0,1%	DT-2236C
Corrente	SC1	0,26 A	8,8%	0,14 A	4,6%	EM6000
	SC2	0,27 A	9,7%	0,15 A	5,4%	EM6000
	SC3	0,19 A	6,6%	0,11 A	3,9%	EM6000
	SC4	0,13 A	39,4%	0,04 A	15,2%	EM6000
Tensão	ST1	5,8 V	2,54%	1,43 V	0,63%	EM6000
	ST2	7,5 V	3,33%	1,56 V	0,69%	EM6000
	ST3	7,9 V	3,49%	2,76 V	1,22%	EM6000

Fonte: Autoria Própria.

De posse destas informações, verificou-se que a ferramenta atende à demanda para a qual foi desenvolvida, com erros médios percentuais abaixo dos 2%, com exceção das medições de corrente que são justificáveis devido as grandezas envolvidas. Também é importante salientar que os erros médios estão próximos das faixas de precisão dos instrumentos profissionais utilizados na validação.

7 CONSIDERAÇÕES FINAIS E SUGESTÕES

A proposta deste trabalho foi desenvolver uma ferramenta que realize a aquisição, processamento e apresentação de variáveis de motor de indução trifásico de rotor bobinado, em tempo real, que pudesse ser implementada no Laboratório de Conversão de Energia e Máquinas da Universidade Federal do Amapá.

Para o desenvolvimento da ferramenta foi necessário conhecer o funcionamento do motor de indução trifásico, de onde definiu-se as variáveis que seriam monitoradas. A próxima etapa consistiu na determinação dos sensores a serem utilizados e a interface para realizar o processamento e apresentação dos dados obtidos.

A plataforma de prototipagem Arduino foi definida como responsável pela maior parte do processamento dos dados, enquanto que o ambiente gráfico de desenvolvimento de sistemas LabVIEW realiza o tratamento final dos dados de alguns sensores e apresenta os dados ao usuário, além de armazená-los.

Em seguida, os sensores e o processamento de seus dados junto ao Arduino foram testados para averiguar se estavam qualificados para o uso no projeto. Esta etapa se mostrou essencial, uma vez que, foi preciso condicionar os sinais do sensor óptico TCRT5000 e, principalmente, do STAC, um sensor sem ficha técnica de dados para consulta, sendo necessário a análise do seu comportamento operacional, a aplicação de um divisor de tensão para que o mesmo atue na zona de maior linearidade, modelá-lo matematicamente através de ajustes de curvas por meio do método dos quadrados mínimos para, então, adequá-lo ao projeto. Após a adequação de todos os sensores, partiu-se para o desenvolvimento do código de programação final, tanto para Arduino quanto para LabVIEW, bem como a interface gráfica e a estrutura física.

Por fim, validou-se o projeto através de testes, onde a ferramenta apresentou resultados satisfatórios, principalmente para o sensor de velocidade que teve uma diminuição da sua resolução de 60 RPM para 4,3 RPM, aproximadamente, devido ao aumento no número de faixas reflexivas presentes no eixo do rotor da máquina. Os sensores de temperatura também denotaram resultados expressivos, com erros de 1,5% em média, para uma temperatura de até 45 °C, não havendo necessidade de mitigação dos erros através de manipulação matemática no código de programação. Os STAC's mantiveram as taxas de erros abaixo de 1,5%, o que constata a validação das equações das curvas obtidas por meio da modelagem matemática realizada no comissionamento dos seus sinais. Uma ressalva é feita aos sensores de corrente, pois apresentaram erros acima de 3,5%, entretanto, os resultados são aceitáveis, visto que, os

erros absolutos se estabeleceram em uma faixa de 0,15 A, aproximadamente e, cada disparidade entre as medições ocasiona um erro percentual significativo, mas que não inviabiliza a sua utilização.

Assim, de um modo geral, a ferramenta, mesmo sendo um protótipo, se mostrou confiável, apresentando taxas de erros bem próximas as faixas de precisão de instrumentos de medições profissionais, credenciando, portanto, a PADMo para uso laboratorial.

Entretanto, destaca-se que melhorias podem ser aplicadas. Por exemplo:

- Acréscimo das variáveis monitoradas, como, vibração, potências, fator de potência, torque, etc.;
- Correlacionar as grandezas, por exemplo, torque x velocidade, corrente x temperatura, etc.;
- Elaborar uma estrutura menos invasiva ao circuito da máquina a ser monitorada;
- Aplicar a ferramenta em outras máquinas;
- Expandir sua aplicabilidade para ambientes profissionais;
- Ampliar para o uso em técnicas de análise e diagnóstico de falhas e faltas.

REFERÊNCIAS BIBLIOGRÁFICAS

ALLEGRO MICROSYSTEMS, LLC. **ACS712**: Fully Integrated, Hall Effect-Based Linear Current Sensor IC with 2.1 kVRMS Isolation and a Low-Resistance Current Conductor. [datasheet], rev. 15, nov. 2012. Disponível em: <http://img.filipeflop.com/files/download/Datasheet_ACS712.pdf>. Acesso em: 28 de mar. 2016.

ARDUINO. **Introduction**. Disponível em: <<https://www.arduino.cc/en/Guide/Introduction>>. Acesso em: 04 abr. 2016.

_____. **Arduino MEGA 2560 & Genuino MEGA 2560**. Disponível em: <<https://www.arduino.cc/en/Main/ArduinoBoardMega2560>>. Acesso em: 04 abr. 2016.

ATMEL CORPORATION. **Atmel ATmega640/V-1280/V-1281/V-2560/V-2561/V**: 8-bit Atmel Microcontroller with 16/32/64KB In-System Programmable Flash. [datasheet], rev. fev. 2014. Disponível em: <http://www.atmel.com/Images/Atmel-2549-8-bit-AVR-Microcontroller-ATmega640-1280-1281-2560-2561_datasheet.pdf>. Acesso em: 17 de fev. 2017.

BHOWMIK, P. S.; PRADHAN, S.; PRAKASH, M. **Fault Diagnostic and Monitoring Methods of Induction Motor**: A Review. International Journal of Applied Control, Electrical and Electronics Engineering, vol.1, n. 1, may 2013. Disponível em: <<http://www.airccse.com/ijaceee/papers/1113ijaceee01.pdf>>. Acesso em: 10 de maio 2016.

BITTER, R.; MOHIUDDIN, T.; NAWROCKI, M. **LabVIEW**: Advanced Programming Techniques. 2. ed. [S.l.]: CRC Press, 2006.

BORTONI, E. C.; SANTOS, A. H. M. Capítulo 11- Acionamento com controles de indução trifásicos. In: **Conservação de Energia**: Eficiência Energética de Equipamentos e Instalações. 3. ed. Itajubá: [s.n.], 2006. p. 397-438. Disponível em: <http://arquivos.portaldaindustria.com.br/app/conteudo_18/2014/04/22/6281/Livro_Conservacao_de_Energiaed3.pdf>. Acesso em: 06 de abr. 2016.

CARVALHO, Geraldo. **Máquinas Elétricas**: Teoria e Ensaio. 4. ed. revisada. São Paulo: Érica, 2011.

CHAPMAN, S. J. **Fundamentos de Máquinas Elétricas**. Tradução: Anatólio Laschuk. 5. ed. Porto Alegre: AMGH, 2013.

CUIN, Marcelo. **Sensor de Tensão AC**. Disponível em: <<http://www.cuin.com.br/stac/>>. Acesso em: 18 de mai. 2016.

DUTRA, L. **Sensor de corrente ACS712 30A**. 2013. Disponível em: <<https://dutrleo.wordpress.com/2013/01/29/sensor-de-corrente-acs712-30a/>>. Acesso em: 03 de jun. 2016.

ELETROBRÁS *et. al.* **Motor Elétrico: Guia Básico**. Brasília, 2009. 190p. Disponível em: <http://arquivos.portaldaindustria.com.br/app/conteudo_18/2014/04/22/6281/Motor_eletrico.pdf>. Acesso em: 06 de abr. 2016.

FITZGERALD, A. E.; KINGSLEY, C., Jr.; UMANS, S. D. **Máquinas Elétricas: Com introdução à eletrônica de potência**. 6. ed. São Paulo: Bookman, 2006.

FRANCISCO, A. M. S. **Motores de Indução Trifásico**. 2 ed. [S.l]: ETEP - Edições Técnicas e Profissionais, 2006.

HAMMO, Rama. **Faults Identification in Three-Phase Induction Motors Using Support Vector Machines**. Dissertation (Master of Technology Management). Graduate College of Bowling Green State University, may 2014. Disponível em: <http://scholarworks.bgsu.edu/cgi/viewcontent.cgi?article=1000&context=ms_tech_mngmt>. Acesso em: 23 de jun. de 2016.

KARMAKAR, S. et al. Induction Motor and Faults. In: _____. **Induction Motor Fault Diagnosis: Approach through Current Signature Analysis**. 1. ed. [S.l]: Springer, 2016. p. 7-28.

KEITHLEY INSTRUMENTS, INC. **Data Acquisition and Control Handbook: A Guide to Hardware and Software for Computer-Based Measurement and Control**. 1. ed. [S.l], 2001.

KOSOW, I. L. **Maquinas elétricas e transformadores**. 15 ed. São Paulo: Globo, 2005.

LABVIEW MAKERHUB. **How LINX Works**. Disponível em: <<https://www.labviewmakerhub.com/doku.php?id=learn:libraries:linx:misc:linx-internals>>. Acesso em: 11 maio. 2016.

MAMEDE FILHO, João. **Instalações Elétricas Industriais**. 6. ed. Rio de Janeiro: LTC, 2007.

MCROBERTS, Michael. **Arduino Básico**. Tradução Rafael Zanolli. São Paulo: Novatec, 2011.
Tradução de: Beginning Arduino.

MEHALA, Neelam. **Condition Monitoring and Fault Diagnosis of Induction Motor Using Motor Current Signature Analysis**. Thesis (Doctor of Philosophy) - National Institute of Technology, Kurukshetra, India, oct. 2010.

MOTRON INDÚSTRIA DE MOTORES REDUTORES, LTDA. **Diagrama de Fechamento dos Motores**. [Apostila]. s.d.

NATIONAL INSTRUMENTS CORPORATION. **Data Acquisition (DAQ) Fundamentals**. Application Note 007, 1999. Disponível em:
<http://physweb.bgu.ac.il/COURSES/SignalNoise/data_aquisition_fundamental.pdf>. Acesso em: 13 de jun. 2016.

PARK, J.; MACKAY, S. **Practical Data Acquisition for Instrumentation and Control Systems**. Burlington: Elsevier, 2003.

REZEK, Â. J. J. **Fundamentos Básicos de Máquinas Elétricas: Teoria e Ensaio**s. Rio de Janeiro: Synergia; Itajubá, MG: Acta, 2011.

ROUSE, M. **Object-Oriented Programming (OOP)**. Aug. 2008. Disponível em:
<<http://searchmicroservices.techtarget.com/definition/object-oriented-programming-OOP>>. Acesso em: 07 de jun. 2017.

SILVA, A. M. da. **Induction Motor: Fault Diagnostic and Monitoring Methods**. Dissertation (Master of Electrical and Computer Engineering). Marquette University, Wisconsin, may 2006. Disponível em: <<http://kidlab.eece.mu.edu/publications/papers/dasilva.pdf>>. Acesso em: 23 de jun. 2016.

SOUZA, Fábio. **Arduino MEGA 2560**. 28 mar. 2014. Disponível em:
<<http://www.embarcados.com.br/arduino-mega-2560/>>. Acesso em: 04 de abr. 2016.

TADANO, Y. de S. **Cálculo Numérico**. [Apostila], Universidade Tecnológica Federal do Paraná, 2014. Disponível em: <http://paginapessoal.utfpr.edu.br/yaratadano/2014-01/calculo-numerico/qm34a/aulas/20-MetNum-EDO.pdf/at_download/file>. Acesso em: 21 de jun. 2017.

TAVNER, P. et al. **Condition Monitoring of Rotating Electrical Machines**. London: IET, 2008.

TEXAS INSTRUMENTS INCORPORATED. **LM35 Precision Centigrade Temperature Sensors**. [datasheet], 1999, rev. jan. 2016. Disponível em: <<http://www.ti.com/lit/ds/symlink/lm35.pdf>>. Acesso em: 24 de maio 2016.

THOMAZINI, D.; ALBUQUERQUE, P. U. B. **Sensores Industriais: Fundamentos e Aplicações**. 4. ed. rev. São Paulo: Érica, 2010.

TRAVIS, J.; KRING, J. **LabVIEW for Everyone: Graphical Programming Made Easy and Fun**. 3. ed. [S.l.]: Prentice Hall, 2006.

VIDA DE SILÍCIO. **Apostila Arduino Básico**. v. 1, [entre 2014 e 2016].

VISHAY INTERTECHNOLOGY, INC. **TCRT5000, TCRT5000L: Reflective Optical Sensor with Transistor Output**. [datasheet], 2009, rev. oct. 2012. Disponível em: <<http://www.vishay.com/docs/83760/tcrt5000.pdf>>. Acesso em: 28 de maio 2016.

WEG. **Danos em Enrolamentos: Motores Trifásicos**. [Folheto], rev. 03, Fevereiro, 2012. Disponível em: <<http://ecatalog.weg.net/files/wegnet/WEG-danos-em-enrolamentos-motores-trifasicos-50009255-guia-de-instalacao-portugues-br.pdf>>. Acesso em: 22 de abr. 2016.

WEG. **Motores elétricos assíncronos e síncronos de média tensão - especificação, características e manutenção**. [Apostila], 2015. Disponível em: <<http://ecatalog.weg.net/files/wegnet/WEG-curso-dt-6-motores-eletricos-assincrono-de-alta-tensao-artigo-tecnico-portugues-br.pdf>>. Acesso em: 22 de abr. de 2016.

WILSON, J. S. (Ed.) **Sensor Technology Handbook**. [S.l.]: Elsevier, 2005.

APÊNDICE A – Características dos Instrumentos de Medição usados na PADMo.

MULTÍMETRO INSTRUTEMP EM6000			
VARIÁVEL	LIMITE	RESOLUÇÃO	PRECISÃO
TENSÃO	Até 600 V	0,1 V	±1,5%
CORRENTE	6 A	1 mA	±2,5%
TEMPERATURA	0 a 400 °C	0,1 °C	±1,5%
FREQUÊNCIA	Até 99,99 Hz	0,01 Hz	±1%

ANALISADOR DE QUALIDADE DE ENERGIA FLUKE 43B		
VARIÁVEL	LIMITE	PRECISÃO
TENSÃO	5 a 500 V	±1,5%
CORRENTE	6 A	±2,5%

TACÔMETRO ÓPTICO DIGITAL DT-2236C			
VARIÁVEL	LIMITE	RESOLUÇÃO	PRECISÃO
VELOCIDADE DE ROTAÇÃO	Até 999,9 RPM	0,1 RPM	±0,05%
	Acima de 1000 RPM	1 RPM	

APÊNDICE B – Códigos de Programação do Arduino para Testes de Validação dos Sensores.

B.1 Código para o teste de qualificação de temperatura.

```

01 //Declaração das Variáveis.
02 const int   pinoSensor      = A1;
03 int         valorSensor     = 0;
04 float       valorTemperatura = 0;
05 float       voltsporUnidade = 0.0048828125; //Conversão A/D do
Arduino. 0.0048828125 = 5/1024.
06
07 void setup() {
08   Serial.begin(9600);
09 }
10
11 void loop() {
12   valorSensor = analogRead(pinoSensor);
13   valorTemperatura = valorSensor * voltsporUnidade; //Converte os
valores obtidos para volts.
14   valorTemperatura = valorTemperatura * 100;          //Calcula a
temperatura em °C.
15
16
17   //Apresentação dos resultados no monitor serial do Arduino.
18   Serial.print ("  TEMPERATURA:  ");
19   Serial.print(valorTemperatura);
20   Serial.println(" C");
21
22   delay(1000); //Limita as medições para 1 segundo.
23 }

```

B.2 Código para o teste de qualificação de velocidade.

```

01 #include <elapsedMillis.h> //Biblioteca para controlar o tempo de
amostragem.
02 elapsedMillis timeElapsed; //Criação da variável para controle do tempo
de amostragem.
03
04 // Declaração das Variáveis.
05 volatile float pulsos      = 0;
06 unsigned int  intervalo    = 1000; //Tempo de amostragem -> 1 segundo.
07
08 void interrupcao() { //Sempre que o TCRT5000 detectar a passagem da
faixa reflexiva a variável "pulsos" será acrescida de 1.
09   pulsos++; //.
10 }
11
12 void setup() {
13   Serial.begin(9600);
14   attachInterrupt(0,interrupcao,FALLING); //Inicialização da
Interrupção do Arduino. "0" é a indentificação da interrupção e fica no
pino digital 2. "interrupcao" é a função que será

```

```

15                                     //executada quando houver uma
mudança de estado na resposta do TCRT5000. Para esse caso, a mudança de
estado ocorrerá quando a resposta
16                                     //do TCRT5000 passar de 1
para 0 (FALLING).
17 }
18
19 void loop() {
20
21   if (timeElapsed > intervalo) { //Quando o tempo estipulado para que a
soma dos pulsos é atingido, o programa apresentará o resultado desta soma
por meio do monitor serial.
22     Serial.print("    RPM:        ");
23     Serial.println(pulsos*60); //Converte para RPM
24     Serial.println();
25
26     //Reinicia as variáveis para a realização de novos cálculos.
27     pulsos = 0;
28     timeElapsed = 0;
29   }
30 }

```

B.3 Código para o teste de qualificação de corrente.

```

01 // Declaração das Variáveis.
02 const int   pinoSensor      = A0;
03 float       valorSensor_aux = 0;
04 float       valorSensor     = 0;
05 float       valorCorrente   = 0;
06 float       voltsporUnidade = 0.0048828125; // Conversão A/D do
Arduino. 0.0048828125 = 5/1024.
07
08 void setup() {
09   Serial.begin(9600);
10 }
11
12 void loop() {
13
14   for(int i=500; i>0; i--) //Armazena 500 medições para posterior
cálculo da média quadrática.
15   {
16     valorSensor_aux = (analogRead(pinoSensor) - 511);
17     valorSensor += pow(valorSensor_aux,2);
18   }
19
20   valorSensor = (sqrt(valorSensor/ 500)) * voltsporUnidade; //Finaliza o
cálculo da média quadrática e ajusta o valor obtido para volts.
21   valorCorrente = (valorSensor/100) * 1000; //Calcula a
corrente considerando a sensibilidade do sensor (100 mV por ampere).
22
23   // Apresentação dos resultados no monitor serial do Arduino.
24   Serial.print("  CORRENTE:  ");
25   Serial.print(valorCorrente);
26   Serial.println(" A" );
27
28   valorSensor = 0; //Reinicia a variável para efetuar novos cálculos.
29
30   delay(1000);
31

```


32 }

B.4 Código para o teste de qualificação de tensão.

```

01 //Declaração das Variáveis.
02 const int   pinoSensor      = A9;
03 float       valorSensor     = 0;
04 float       saidaSensor     = 0;
05 float       vSTAC_2        = 0;
06 float       vFinal         = 0;
07 float       voltsporUnidade = 0.0048828125; //Conversão A/D do
Arduino. 0.0048828125 = 5/1024.
08
09 void setup() {
10   Serial.begin(9600);
11 }
12
13 void loop() {
14   valorSensor = analogRead(pinoSensor);
15   saidaSensor = valorSensor * voltsporUnidade ;
16
17   //Equações para cálculo dos valores de tensão no resistor R2 do
divisor de tensão.
18   vSTAC_1=((-4.677*pow(saidaSensor,4)) + (30.876*pow(saidaSensor,3)) + (-
74.856*pow(saidaSensor,2)) + (123.37*saidaSensor) + 16.212);
19
20   //vSTAC_2=((-5.4367*pow(saidaSensor,4)) + (34.244*pow(saidaSensor,3))
+ (-79.475*pow(saidaSensor,2)) + (127.19*saidaSensor) + 16.125);
21   //vSTAC_3=((-3.8669*pow(saidaSensor,4)) + (26.593*pow(saidaSensor,3))
+ (-67.298*pow(saidaSensor,2)) + (117.81*saidaSensor) + 15.45);
22   //vSTAC_4=((-2.7451*pow(saidaSensor,4)) + (20.447*pow(saidaSensor,3))
+ (-55.939*pow(saidaSensor,2)) + (106.22*saidaSensor) + 15.443);
23
24   //Equação para determinar a tensão na fonte de alimentação do
circuito.
25   vFinal = (1.4765 * vSTAC_1) - 2.3659;
26
27
28
29   //Apresentação dos resultados no monitor serial do Arduino.
30   Serial.print("   Tensao na Fonte:   ");
31   Serial.print(vFinal);
32   Serial.println( );
33
34   delay(1000); //Limita as medições para 1 segundo.
35 }

```

APÊNDICE C – Código de Processamento do Arduino para a PADMo.

```

01 //Bibliotecas periféricas utilizadas pelo LINX
02 #include <SPI.h>
03 #include <Wire.h>
04 #include <EEPROM.h>
05 #include <Servo.h>
06
07 //Inclui Header específico do dispositivo do sketch (código) >> Importa
Biblioteca (Nesse caso LinxArduinoMega2560.h)
08 //Também inclui Listener LINX desejado do sketch >> Importa Biblioteca
(Nesse caso LinxSerialListener.h)
09 #include <LinxArduinoMega2560.h>
10 #include <LinxSerialListener.h>
11
12 //Biblioteca para averiguação do tempo e variável que a utilizará
(timeElapsed)
13 #include <elapsedMillis.h>
14 elapsedMillis timeElapsed;
15
16 //Cria um Ponteiro (Pointer) para o objeto do dispositivo LINX
instanciado no Setup()
17 LinxArduinoMega2560* LinxDevice;
18
19 //Variável auxiliar para armazenamento dos dados dos sensores
20 volatile float Dados[11];
21
22 //Variáveis utilizadas para medição de Velocidade
23 volatile float Detecoes = 0; //variável que armazenará as detecções das
faixas reflexivas
24 float Detec1 = 0;
25 int Detec2 = 0;
26 float Detec3 = 0;
27 int timeOn = 1000; //intervalo de tempo
28
29 //Início da classe para medição de Temperatura
30 class Temperatura{
31 //Declaração das variáveis
32 int pinoSensor;
33 int posArray; //posição no vetor(array) Dados[]
34 float valSensorAux;
35 float valSensor;
36 float Temperatural;
37 float Vout
38 float vUnid;
39
40 //Constructor - inicializa as variáveis declaradas acima
41 public:
42 Temperatura(int pinoSensor_, int posArray_ ){
43 pinoSensor = pinoSensor_;
44 posArray = posArray_ ;
45 vUnid = 0.0048828125; //volts por unidade - 5/1024
46 valSensorAux = 0;
47 valSensor = 0;
48 Temperatural = 0;
49 Vout = 0;
50 }
51
52 //Função principal que executará o processamento dos dados do sensor -
semelhante ao void loop()

```

```

53 void Update(){
54     //Executa a soma de x leituras do sensor
55     for(int i=50; i>0; i--){
56         valSensorAux = analogRead(pinoSensor);
57         valSensor += pow(valSensorAux,2);
58     }
59     Vout = (sqrt(valSensor/50)) * vUnid; //calcula a média das x
leituras do sensor
60     Temperatur1 = Vout * 100; //calcula a temperatura em °C
61     valSensor = 0; //reinicia a variável para novas medições
62
63     Dados[posArray] = Temperatur1; //Armazena o valor de temperatura no
vetor(array)
64 }
65 }; //fim da classe de temperatura
66
67
68 //Início da classe para medição de Corrente
69 class Corrente{
70     //Declaração das variáveis
71     int pinoSensor;
72     int posArray;
73     float valSensorAux;
74     float valSensor;
75     float Correntel;
76     float vUnid;
77
78     //Constructor - inicializa as variáveis declaradas acima
79     public:
80     Corrente(int pinoSensor_, int posArray_){
81         pinoSensor = pinoSensor_;
82         posArray = posArray_;
83         vUnid = 0.0048828125;
84         valSensorAux = 0;
85         valSensor = 0;
86         Correntel = 0;
87     }
88
89     //Função principal que executará o processamento dos dados do sensor -
semelhante ao void loop()
90     void Update(){
91         //Realiza a média quadrática das x leituras do sensor
92         for(int i=250; i>0; i--){
93             valSensorAux = (analogRead(pinoSensor) - 511);
94             valSensor += pow(valSensorAux,2);
95         }
96         valSensor = (sqrt(valSensor/250)) * vUnid;
97         Correntel = (valSensor/100) * 1000; //Calcula a corrente em (A)
98         Correntel = Correntel * 10; //Divide por 10 para que os dados possam
ser enviados ao LabVIEW
99         valSensor = 0; //Reinicia a variável para novas medições
100
101         Dados[posArray] = Correntel; //Armazena o valor de corrente no
vetor
102     }
103 }; //Fim da classe de corrente
104
105
106 //Início da classe para medição de Tensão
107 class Tensao{
108     //Declaração das variáveis

```

```

109  int pinoSensor;
110  int nSTAC; //número de identificação do STAC
111  int posArray;
112  float valSensorAux;
113  float valSensor;
114  float vDiv; //Tensão no resistor de 100k presente no divisor de
tensão
115  float vFinal; //Tensão de alimentação - Vs
116  float vUnid;
117
118  //Constructor - inicializa as variáveis declaradas acima
119  public:
120  Tensao(int pinoSensor_, int nSTAC_, int posArray_){
121      pinoSensor = pinoSensor_;
122      nSTAC = nSTAC_;
123      posArray = posArray_;
124      vUnid = 0.0048828125;
125      valSensorAux = 0;
126      valSensor = 0;
127      vDiv = 0;
128      vFinal = 0;
129  }
130
131  //Função principal que executará o processamento dos dados do sensor
- semelhante ao void loop()
132  void Update(){
133      //Executa a soma de x leituras do sensor
134      for(int i=50; i>0; i--){
135          valSensorAux = analogRead(pinoSensor);
136          valSensor += pow(valSensorAux,2);
137      }
138      valSensor = (sqrt(valSensor/50)) * vUnid; //calcula a média das x
leituras do sensor
139
140      //Verifica qual o número de identificação do STAC para determinar
quais equações são as adequadas
141      if (nSTAC == 1){
142          vDiv = ((-0.4803*pow(valSensor,4))+(3.9407*pow(valSensor,3))+(-
14.631*pow(valSensor,2))+(68.535*valSensor)+30.12);
143          vFinal = 1.3072 * pow(vDiv,1.0215);
144      }
145      else if (nSTAC == 2){
146          vDiv = ((-1.1342*pow(valSensor,4))+(9.1582*pow(valSensor,3))+(-
29.449*pow(valSensor,2))+(87.867*valSensor)+24.123);
147          vFinal = 1.309 * pow(vDiv,1.0238);
148      }
149      else if (nSTAC == 3){
150          vDiv = ((0.2958*pow(valSensor,4))+(-1.1038*pow(valSensor,3))+(-
3.416*pow(valSensor,2))+(54.087*valSensor)+31.274);
151          vFinal = 1.2997 * pow(vDiv,1.025);
152      }
153
154      //O programa só mostra valores de tensão a partir de 100 V
155      if (vFinal < 100){
156          vFinal = 0;
157      }
158
159      valSensor = 0; //Reinicia a variável para novas medições
160
161      Dados[posArray] = vFinal; //Armazena o valor de tensão no vetor
162  }

```

```

163 }; //Fim da classe de tensão
164
165 //Inicialização das variáveis que utilizarão as classes
166 Temperatura temp1(A0,2);
167 Temperatura temp2(A1,3);
168
169 Corrente cor1(A5,4);
170 Corrente cor2(A4,5);
171 Corrente cor3(A3,6);
172 Corrente cor4(A2,7);
173
174 Tensao tens1(A6,1,8);
175 Tensao tens2(A7,2,9);
176 Tensao tens3(A8,3,10);
177
178 //Função que será executada quando a interrupção externa for ativada
179 //Esta função auxiliará na medição de velocidade
180 void Interrupcao(){
181   if (timeElapsed < timeOn){ //Limita o tempo em que as detecções das
faixas reflexivas são somadas
182     Detecoes++; //soma as detecções das faixas reflexivas
183   }
184 }
185
186 void setup()
187 {
188   //Inicializa o Comando Customizado do LINX
189   LinxSerialConnection.AttachCustomCommand(0, PADMo);
190
191   //Instacia o dispositivo LINX
192   LinxDevice = new LinxArduinoMega2560();
193
194   //O LINX Listener é pre instanciado, chama o início e passa um
ponteiro para
195   //o dispositivo LINX e canal UART para ficarem de prontidão
196   LinxSerialConnection.Start(LinxDevice, 0);
197
198   //Inicializa a interrupção externa do Arduino
199   attachInterrupt(0, Interrupcao, FALLING);
200 }
201
202 void loop()
203 {
204   //Espera por novos pacotes do LabVIEW
205   LinxSerialConnection.CheckForCommands();
206
207   //Verifica o número de detecções das faixas reflexivas a cada x
milisegundos
208   if (timeElapsed > timeOn){
209     Detec1 = Detecoes/14; //Calcula o valor de rotações por segundo do
eixo do motor - valor com casas decimais
210     Detec2 = Detecoes/14; //Semelhante ao descrito acima, porém apenas
com valores inteiros
211     Detec3 = (Detec1 - Detec2) * 100; //Determina a diferença entre as
duas variáveis acima e converte para um valor inteiro
212
213     //Armazena os dados para um novo processamento no LabVIEW
214     Dados[0] = Detec2;
215     Dados[1] = Detec3;
216
217     //As variáveis são reiniciadas para novas medições

```

```
218  Detecoos = 0;
219  timeElapsed = 0;
220 } //Fim do processamento dos dados do sensor de velocidade
221
222  temp1.Update(); //Processa os dados do sensor de temperatura 1
223  temp2.Update(); //Processa os dados do sensor de temperatura 1
224
225  tens1.Update(); //Processa os dados do sensor de tensão 1
226  tens2.Update(); //Processa os dados do sensor de tensão 2
227  tens3.Update(); //Processa os dados do sensor de tensão 3
228
229  cor1.Update(); //Processa os dados do sensor de corrente 1
230  cor2.Update(); //Processa os dados do sensor de corrente 2
231  cor3.Update(); //Processa os dados do sensor de corrente 3
232  cor4.Update(); //Processa os dados do sensor de corrente 4
233 }
234
235 //Comando Customizado do LINX
236 int PADMo(unsigned char numInputBytes, unsigned char* input, unsigned
char* numResponseBytes, unsigned char* response)
{
237
238 //Envia os dados armazenados no vetor Dados[] para o LabVIEW
239 response[0] = Dados[0]; //velocidade
240 response[1] = Dados[1];
241
242 response[2] = Dados[2]; //temperatura
243 response[3] = Dados[3];
244
245 response[4] = Dados[4]; //corrente
246 response[5] = Dados[5];
247 response[6] = Dados[6];
248 response[7] = Dados[7];
249
250 response[8] = Dados[8]; //tensao
251 response[9] = Dados[9];
252 response[10] = Dados[10];
253
254 *numResponseBytes = 11;
255
256 return 0;
257 }
```

APÊNDICE D – Código de Processamento do LabVIEW para a PADMO.

