UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
DEPARTAMENTO DE ENGENHARIA MECÂNICA
ENGENHARIA MECÂNICA

LATIF DE FARIA GARCEZ

DESENVOLVIMENTO DE UM SISTEMA DE MANUTENÇÃO PREDITIVA PARA MOTORES ELÉTRICOS

TRABALHO DE CONCLUSÃO DE CURSO

LONDRINA
2021

LATIF DE FARIA GARCEZ

DESENVOLVIMENTO DE UM SISTEMA DE MANUTENÇÃO PREDITIVA PARA MOTORES ELÉTRICOS

Trabalho de Conclusão de Curso apresentada como requisito parcial à obtenção do título de Bacharel, em Engenharia Mecânica, do Departamento de Engenharia Mecânica, da Universidade Tecnológica Federal do Paraná.

Orientador: Prof. Dr. Roger Nabeyama Michels

Coorientador: Prof. Dr. Janksyn Bertozi

LONDRINA

2021

TERMO DE APROVAÇÃO

DESENVOLVIMENTO DE UM SISTEMA DE MANUTENÇÃO PREDITIVA PARA MOTORES ELÉTRICOS

por

LATIF DE FARIA GARCEZ

Este Trabalho de Conclusão de Curso foi apresentado em 30 de Novembro de 2021 como requisito parcial para a obtenção do título de Bacharel em Engenharia Mecânica. O candidato foi arguido pela Banca Examinadora composta pelos professores abaixo assinados. Após deliberação, a Banca Examinadora considerou o trabalho aprovado.

_____
Roger Nabeyama Michels
Profº. Orientador

_____
Amadeu Lombardi Neto
Membro titular

_____
Cláudia Santos Fiuza Lima
Membro titular

AGRADECIMENTOS

Agradeço primeiramente aos meus pais, por sempre me apoiarem, e acreditarem que chegar até este momento seria possível.

Agradeço aos meus amigos de classe, que juntos nos apoiamos e nos ajudamos para chegar até aqui. Tenho certeza que sem essa união, tudo seria mais difícil.

Gostaria de agradecer especialmente ao meu amigo Pedro Barion e seus pais Mário e Rosangela, jamais esquecerei toda ajuda, suporte e empatia. Com vocês eu aprendi como existem pessoas de bom coração, e nenhuma palavra será capaz de descrever a gratidão que tenho por vocês!

Por último, mas não menos importante, agradecer aos profissionais do NUAPE-LD que sempre estavam dispostos a ajudar, e aos docentes por toda bagagem de conhecimento transmitida, em especial, aos docentes Roger Michels, Janksyn Bertozzi, Alireza Ashtiani e Ricardo Ribeiro, vocês foram muito mais que professores, proferiram palavras de motivação e ajuda, que permitiram tornar esse sonho realidade, obrigado!

RESUMO

GARCEZ, Latif de Faria. **Desenvolvimento de um sistema de manutenção preditiva para motores elétricos**. 2021. 85. Trabalho de Conclusão de Curso de Bacharelado em Engenharia Mecânica - Universidade Tecnológica Federal do Paraná. Londrina, 2021.

A Indústria enfrenta diariamente novos desafios para manter-se competitiva diante de seus concorrentes. Para garantir que o processo produtivo de uma empresa seja confiável, é necessário conhecê-lo e monitorá-lo a ponto de afirmar que os parâmetros de funcionamento estejam dentro dos limites toleráveis. A manutenção preditiva permite prever falhas de equipamentos, e planejar uma manutenção programada, intervindo de maneira pouco significativa no processo de produção. Com isso, o presente trabalho possuí como objetivo, desenvolver um sistema de manutenção preditiva para motores elétricos, equipamentos estes, que representam em torno de 70% da demanda energética industrial. Por fim, o usuário terá total conhecimento dos parâmetros de funcionamento de seu equipamento, aumentando a confiabilidade e disponibilidade do processo.

Palavras-chave: Manutenção. VB.NET. Arduino. Motores Elétricos.

ABSTRACT

GARCEZ, Latif de Faria. **Development of a predictive maintenance system for electric motors**. 2021. 85. Trabalho de Conclusão de Curso de Bacharelado em Engenharia Mecânica - Federal Technology University – Parana. Londrina, 2021.

The industry faces new challenges every day to remain competitive with its competitors. To ensure that the production process of a company is reliable, it is necessary to know and monitor it to the point of affirming that the operating parameters are within tolerable limits. Predictive maintenance allows the prediction of equipment failure, and the planning of scheduled maintenance, intervening in a minor way in the production process. With this, the present work has the objective of developing a predictive maintenance system for electric motors, equipment that represent around 70% of the industrial energy demand. Finally, the user can have full knowledge of the operating parameters of their equipment, increasing the reliability and availability of the process.

Keywords: Maintenance. VB.NET. Arduino. Electric Motors.

LISTA DE ILUSTRAÇÕES

SUMÁRIO

1 INTRODUÇÃO

A competitividade da indústria é algo intrínseco no processo de ganho de espaço no mercado, tornando-se algo imperativo no cenário contemporâneo, em meio ao processo de constante e rápida evolução tecnológica e globalização financeira (MCT-MINISTÉRIO DA CIÊNCIA E TECNOLOGIA, 1993).

A globalização e a constante inovação dos meios de compra e venda de produtos, exigem das Indústrias atualmente, cada vez mais, maneiras de otimizar o processo de produção para atender a demanda exigida em um prazo inferior ao seu concorrente. Em sistemas de produção de bens, as falhas implicam em atraso de produção, desperdício de insumos, retrabalho, horas extras e estoques altos (CAVALCANTE; ALMEIDA, 2005).

Para atender a exigência de mercado e estar à par da concorrência, é necessário que todo o processo de fabricação, manutenção e logística, estejam alinhados, de maneira que um não venha interferir no outro. Porém, no Brasil, em um estudo realizado em 2012, cerca de 41,17% das empresas não desenvolviam nenhuma atividade de engenharia de manutenção (REIS; COSTA; ALMEIDA, 2012). Garantir a produção continuada, com interrupções de manutenção antecipadamente planejadas, é garantir também a confiabilidade do processo e a competitividade de mercado.

Segundo a CPFL ENERGIA (2021), as Indústrias brasileiras consomem cerca de 40% da energia elétrica utilizada no país, e a parcela de eletricidade consumida por motores elétricos dentro da Indústria, corresponde a 70% de toda a demanda de consumo energético industrial.

Com base em uma demanda tão elevada de utilização de motores elétricos nos processos industriais, e uma considerável parcela de empresas que não desenvolvem atividade de manutenção, este trabalho propõe desenvolver um sistema de manutenção preditiva para motores elétricos, visando garantir a máxima disponibilidade do equipamento sob condições ideais de uso.

1.1 OBJETIVO

1.1.1 Objetivo Geral

O objetivo deste presente trabalho, é criar um sistema de manutenção preditiva para motores elétricos, integrando a plataforma Arduino, *software* em VB.NET e um banco de dados em MySQL.

1.1.2 Objetivos Específicos

- Obter temperatura do estator do motor;
- Obter temperatura da carcaça do motor;
- Medir valores de corrente e tensão do motor;
- Avaliar rotação de funcionamento do motor;
- Desenvolver um software em VB.NET para análise dos dados coletados;
- Desenvolver um banco de dados em MySQL para armazenamento dos dados.

1.2 JUSTIFICATIVA

Em meio a um cenário industrial cada vez mais competitivo, a eficiência de produção é um parâmetro fundamental para uma produção contínua e confiável. Para garantir a confiabilidade de um processo produtivo, é necessário conhecer o estado de funcionamento de cada equipamento pertencente ao processo, além das variáveis globais do ambiente. Existe uma vasta gama de aplicações de motores elétricos dentro da indústria, que pode ir desde o acionamento de um portão automático, como a movimentação de uma esteira transportadora. Para cada aplicação, existirá parâmetros ideais de funcionamento, estabelecidos pelo fabricante ou pela equipe de engenharia, que garantirão a longevidade do equipamento. Um sistema de manutenção preditiva,

permite que a planta da indústria opere de maneira controlada e confiável, garantindo produtividade sem quebras inesperadas de equipamentos.

2 REFERENCIAL TEÓRICO

2.1 MANUTENÇÃO

O processo de manutenção conhecido atualmente, foi um caminho de evolução conjunta com a indústria. O termo manutenção, vem do cerne de manter, preservar, garantir seu pleno funcionamento ao qual o foi destinado. Esse princípio de preservar os equipamentos, existe desde épocas mais remotas, porém começou a ser conhecida com o nome de manutenção por volta do século XVI na Europa central, com surgimento do relógio mecânico, quando surgiram os primeiros técnicos em montagem e assistência (MORO; AURAS, 2007).

Desde então, o processo de manutenção sofreu constantes mudanças para adequar-se ao cenário em que a indústria se encontrava em um determinado momento da história. Com a evolução tecnológica, impulsionada principalmente por eventos como a Revolução Industrial e a Segunda Guerra Mundial, novas técnicas de manutenção foram sendo desenvolvidas, e aprimoradas até encontrarem-se no cenário atual.

Segundo Pinto E Xavier (2003), os critérios de desempenho na área de manutenção se deram em três gerações:

- Primeira Geração, anterior a 1940, caracterizada pelo tipo de manutenção apenas após a falha;
- Segunda Geração, de 1940 até 1970, caracterizada pela maior disponibilidade e vida útil dos equipamentos, com planejamentos de manutenção manuais e monitoramento por tempo;
- Terceira Geração, posterior a 1970, caracterizando tudo aquilo conhecido atualmente como manutenção, destacando-se aspectos como maior

disponibilidade, confiabilidade, otimização de custo benefício, maior segurança, melhor qualidade, entre outros.

2.1.1 Tipos De Manutenção

Kardec e Nascif (2009) apresentam a manutenção conhecida atualmente, classificada em dois grupos: a planejada e a não planejada.

- Manutenção não planejada: É realizada sem programação antecipada, portanto, de caráter corretivo, que visa corrigir algum problema. E dentro da manutenção corretiva, existem dois tipos de manutenção corretiva:
    - Inesperada: Consiste em reparar defeitos que ocorrem repentinamente durante o regime operacional do equipamento.
    - Ocasional: Tem como objetivo realizar um reparo de alguma falha que não para o funcionamento do equipamento.
- Manutenção planejada: É realizada com planejamento prévio, e não necessariamente corrige algum problema, tem caráter antecipativo. A manutenção planejada divide-se em três categorias:
    - Preventiva: Caracteriza-se o conjunto de ações que garantirão o funcionamento do equipamento, independentemente do estado operacional de seus componentes.
    - Preditiva: É caracterizada pela ação preventiva com base em conhecimento prévio dos parâmetros de funcionamento de cada componente do equipamento.
    - Detectiva: É o tipo de manutenção que realiza verificação de possíveis falhas em sistemas de proteção dos equipamentos.

Dentre todos os tipos de manutenção descritos anteriormente, o nível mais elevado da manutenção, é a Engenharia de Manutenção, que engloba elementos de todos os tipos de manutenção. Ela é o suporte técnico da manutenção, e está dedicada a consolidar a rotina e implantar a melhoria (KARDEC; NASCIF, 2009).

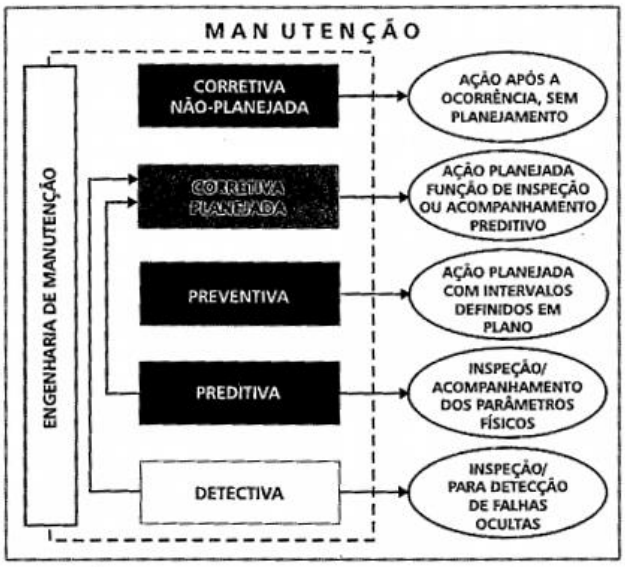

Figura 1- Tipos de manutenção.
Fonte: (KARDEC; NASCIF, 2009).

2.1.2 Manutenção Preditiva

Como já descrito anteriormente, a manutenção preditiva utiliza-se como fundamento, os parâmetros prévios de funcionamento de um equipamento. Utilizar-se destes parâmetros permite que o operador e/ou a central de operações de equipamentos, monitore o seu estado de funcionamento para assegurar-se de que tudo está dentro das condições normais de operação.

Esta técnica de manutenção é a primeira grande quebra de paradigma na manutenção, em que quanto mais intensificam-se o desenvolvimento de conhecimentos tecnológicos, surgem equipamentos mais precisos que permitem avaliações mais confiáveis de instalações em funcionamento (KARDEC; NASCIF, 2009).

Através da manutenção preditiva, reduz-se a necessidade de serviços de manutenção do equipamento, elimina-se a chance de desmontagem

desnecessária, eleva-se significantemente o tempo de disponibilidade dos equipamentos, reduz-se paradas de emergência, aumenta-se o aproveitamento da vida útil dos equipamentos e a confiabilidade do desempenho e determina-se previamente interrupções de fabricação (ALMEIDA, 2008).

Porém, como este tipo de técnica de manutenção exige monitoramento contínuo dos equipamentos para comparação com parâmetros normais de funcionamento, é necessário atender algumas condições básicas para a sua aplicação correta. Segundo Kardec e Nascif (2009), as condições básicas para adotar-se um sistema de Manutenção Preditiva são as seguintes:

- O equipamento, o sistema ou a instalação, devem permitir algum tipo de monitoramento/medição.
- O equipamento, o sistema ou a instalação devem merecer esse tipo de ação, custos envolvidos.
- As falhas devem ser oriundas de causas que possam ser monitoradas e ter sua progressão acompanhada
- Seja estabelecido um programa de acompanhamento, análise e diagnóstico, sistematizado.

A necessidade de um sistema de monitoramento faz com que este tipo de manutenção tenha um custo inicial de implantação considerável. Kardec e Nascif (2009) apresentam duas óticas diferentes para análise dos custos envolvidos:

- O acompanhamento periódico através de instrumentos de medição não é algo que represente um custo muito elevado, uma vez que a mão de obra envolvida não apresenta custo significativo haja vista a possibilidade de acompanhamento pelos próprios operadores dos equipamentos.
- Para a instalação de sistemas de monitoramento contínuo *online*, apresenta-se um custo inicial relativamente elevado. Em perspectiva a todos os custos envolvidos estima-se inicial que o

> investimento seja de cerca de 1% do capital total do equipamento a ser monitorado em que um programa de acompanhamento de equipamentos bem gerenciado apresenta uma relação custo-benefício de 1/5.

Coletar os dados de funcionamento de um equipamento é fundamental para a implantação de um sistema preditivo, porém, mais importante que coletar os dados, é a interpretação dada à eles. Portanto, há a necessidade de operários devidamente treinados e qualificados para gerir e garantir a eficácia de um sistema preditivo.

2.1.2.1Tipos de análises preditivas

Algumas das técnicas mais comuns de monitoramento em uma análise preditiva são.

- Temperatura
- Umidade
- Ultrassom
- Vibração
- Corrente
- Tensão
- Contagem de partículas
- Ferrografia

2.2 MOTOR ELÉTRICO

O motor elétrico pode ser definido como um equipamento capaz de converter energia elétrica em energia mecânica. Seu funcionamento parte do princípio da interação entre campos eletromagnéticos, existindo também, alguns tipos de motores que possuem como base fenômenos eletromecânicos (FRANCHI, 2008).

Atualmente existem uma infinidade de motores elétricos presentes no mercado, cada qual com características que o destinam para uma determinada

aplicação. Apesar da grande variedade, pode-se caracterizá-los em três grandes grupos:

- Motores Síncronos;
- Motores Assíncronos;
- Motores de corrente contínua.

2.2.1 Motores Síncronos

Os motores síncronos são caracterizados por possuírem velocidade de rotação sincronizada com à frequência da rede de alimentação, menor corrente de partida, operam a uma velocidade constante além de possuir maior rendimento (WEG, 2021).

A arquitetura de um motor síncrono é composta por um estator alimentado com corrente alternada, um rotor bobinado alimentado com corrente contínua, e escovas que são responsáveis por fazerem a alimentação das bobinas do rotor. A Figura 2 ilustra os componentes básicos de um motor síncrono.

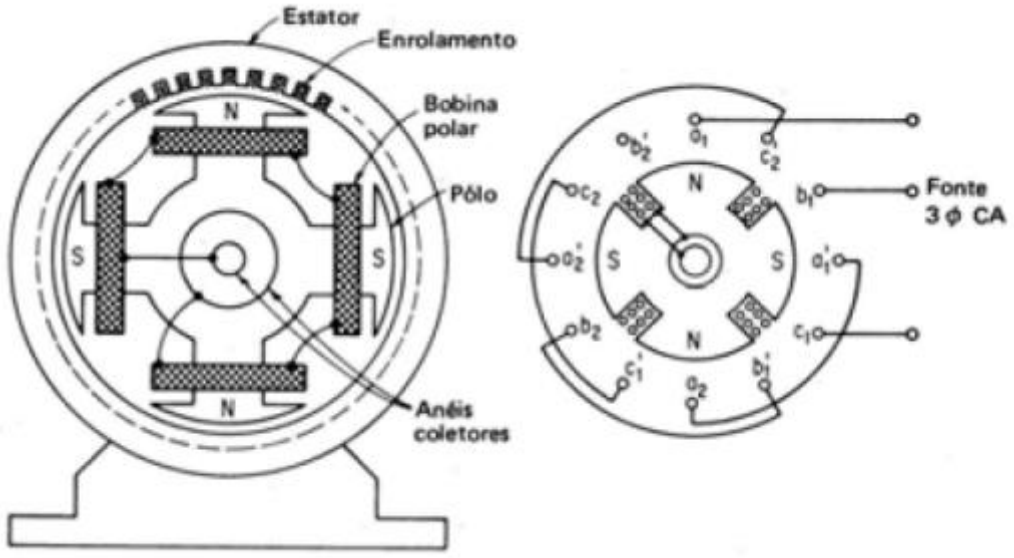

Figura 2- Representação interna de um motor elétrico.
Fonte: (PROCEL, 2009).

Como o rotor e o estator são alimentados, diz-se que este tipo de motor possuí excitação dupla, como resultado disso o torque é desenvolvido a apenas

uma velocidade, chamada de velocidade síncrona. A velocidade síncrona é caracterizada como a velocidade em que o rotor e o campo de fluxo do rotor estão parados entre si, gerando um movimento síncrono (PROCEL, 2009).

2.2.2 Motores Assíncronos

Os motores assíncronos, também conhecidos como motores de indução, são motores que possuem velocidade de rotação próxima a velocidade de sincronismo. Geralmente são comercializados em duas categorias, os monofásicos e os trifásicos.

Um motor assíncrono monofásico, possuí em sua arquitetura um estator, ao qual estão acopladas as bobinas de campo, um rotor do tipo gaiola, sistema de ventilação acoplado ao eixo, rolamentos, bobinas de campo e um interruptor centrífugo (Figura 3).

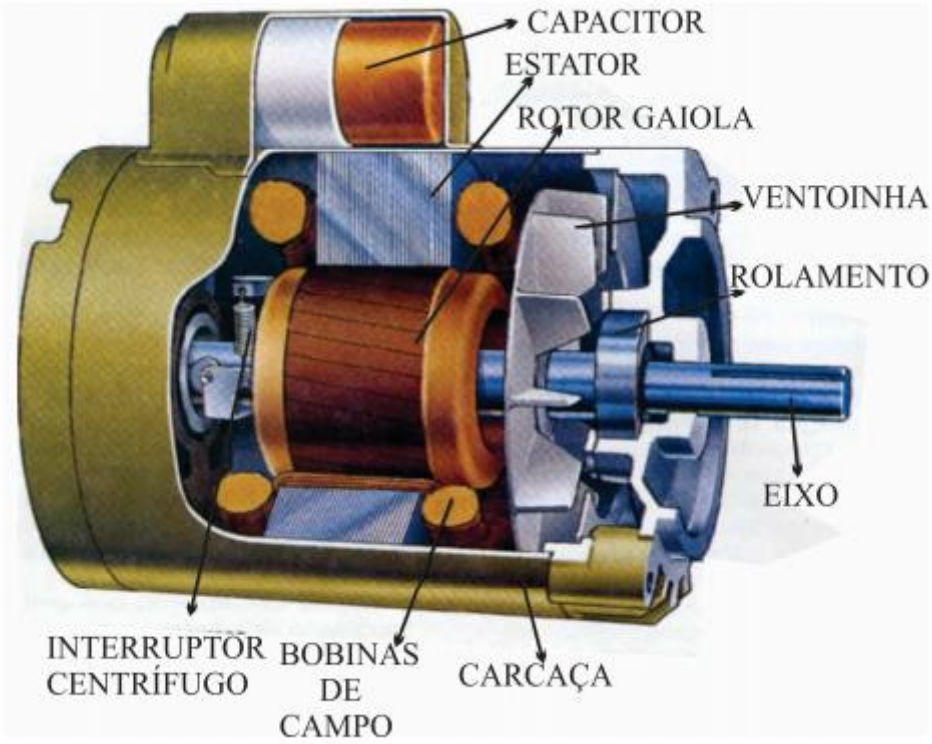

Figura 3- Arquitetura de um motor assíncrono.
Fonte: (GUIA CNC, 2003).

Como nesse tipo de motor, o rotor não possuí enrolamento, a alimentação é feita apenas no estator, o qual é responsável por gerar um campo magnético alternante com a corrente. O campo gerado pelo estator, induz

corrente no rotor, um campo magnético de polaridade oposta é gerado, fazendo com que surja um conjugado de momento de mesma intensidade em direções opostas, fazendo com que o rotor continue parado (WEG, 2021). A Figura 4 ilustra o conjugado de forças causando momento em direções opostas.

Portanto, para que este tipo de motor inicie seu giro, é necessário que seja realizado uma partida. Os dois tipos mais comuns de partida, são o de campo distorcido e o de fase auxiliar, sendo este último o de mais larga aplicação.

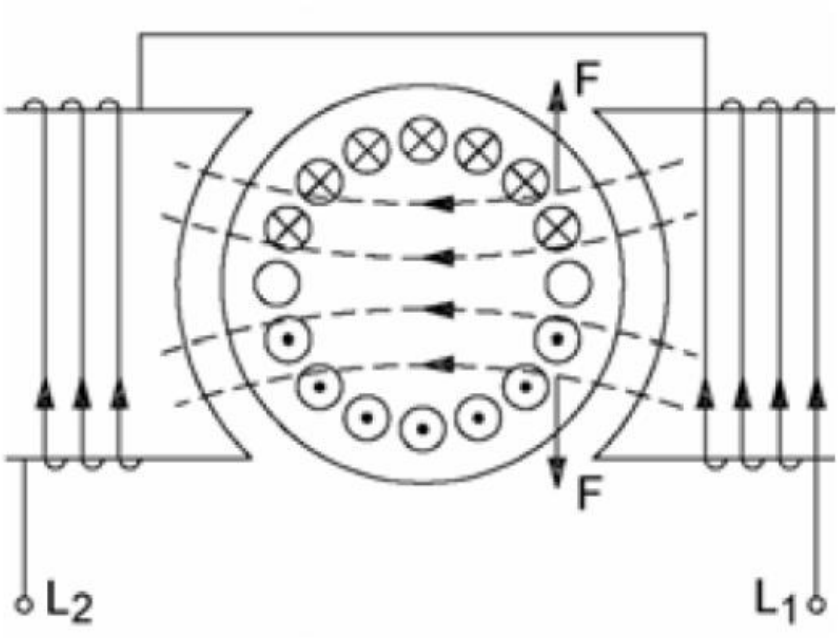

Figura 4- Representação do campo eletromagnético em um motor monofásico.
Fonte: (PINHEIRO, 2015).

Os motores de fase auxiliar, possuem em sua construção dois enrolamentos no estator. Um enrolamento é composto por um fio menos espesso e com poucas espiras, responsável por realizar a partida do motor. O outro enrolamento é o principal, que possuí um fio mais espesso, e com grande quantidade de espiras. Esse último, fica ligado durante todo o funcionamento do equipamento, enquanto o enrolamento auxiliar atua somente durante a partida.

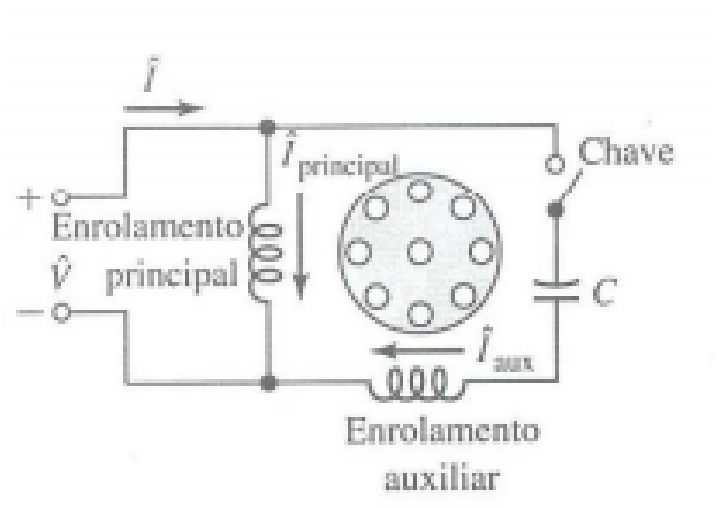

Figura 5- Esquema elétrico motor capacitor-start.
Fonte: (PINHEIRO, 2015).

A Figura 5 ilustra o esquema de funcionamento de um motor do tipo *capacitor-start.* O capacitor ligado em série com o enrolamento auxiliar permite uma maior defasagem em relação às correntes do enrolamento principal, fazendo com que o conjugado de partida seja elevado (PROCEL, 2009).

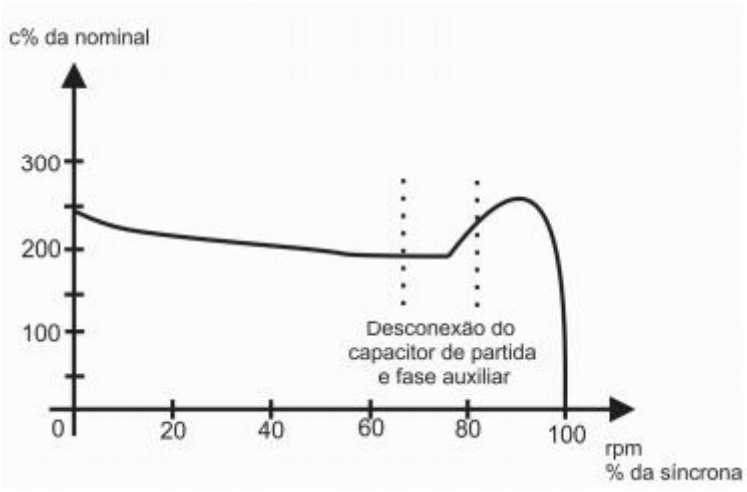

Figura 6- Rotação do motor com capacitor de partida.
Fonte: (PINHEIRO, 2015).

É notório no gráfico da Figura 6 que o conjugado de partida atinge valores elevados, cerca de 250 a 300% do conjugado nominal, fazendo com que sua aplicação seja muito ampla, com fabricações variando de ¼ cv a 15 cv (PINHEIRO, 2015). A Figura 7 ilustra o esquema elétrico das bobinas de um motor monofásico de fase auxiliar.

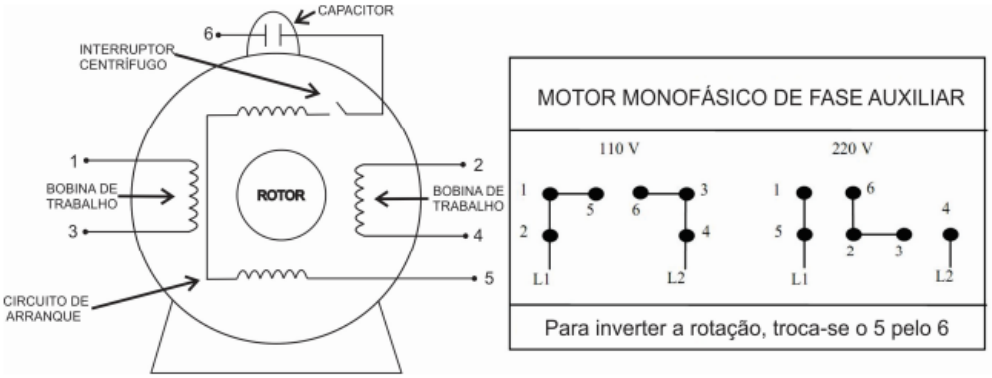

Figura 7- Esquema de bobinas e terminais de um motor monofásico com capacitor de partida.
Fonte: (PINHEIRO, 2015).

2.3 ARDUINO

Arduino é uma plataforma eletrônica de prototipagem *open source*, que reúne características de *hardware* e *software* que garantem que seu processo de aprendizado e utilização, seja simplificado. Ele é capaz de executar tarefas de entrada e saída de dados e comandos, através de um conjunto de instruções programadas em seu microcontrolador (ARDUINO, 2018).

2.3.1 Hardware

O *hardware* varia muito de cada versão da plataforma, mas no geral, as placas possuem:

Figura 8- Componentes principais do Arduino UNO.
Fonte: (MULTILÓGICA-SHOP, 2021).

1. Conector USB para o cabo tipo AB
2. Botão de reset
3. Pinos de entrada e saída digital e PWM
4. LED verde de placa ligada
5. LED laranja conectado ao pin13
6. ATmega encarregado da comunicação com o computador
7. LED TX (transmissor) e RX (receptor) da comunicação serial
8. Porta ICSP para programação serial
9. Microcontrolador ATmega 328, cérebro do Arduino
10. Cristal de quartzo 16Mhz
11. Regulador de voltagem
12. Conector fêmea 2,1mm com centro positivo
13. Pinos de voltagem e terra
14. Entradas Analógicas

2.3.2 Software

A *IDE (Integrated Development Environment)* do Arduino é disponibilizada gratuitamente no site da plataforma, tem sua origem baseada em Processing e é compatível com sistemas Windows Linux e Mac OS (Figura 9).

Figura 9- Tela inicial da *IDE* do Arduino.
Fonte: (ARDUINO, 2021).

2.3.3 Programação

A programação destinada a compilação e execução no microcontrolador, são chamadas de Sketches e possuem extensão “.ino”. A linguagem de programação utilizada é de origem própria, baseada em Wiring permitindo que o usuário programe em C/C++ (ARDUINO, 2018).

Por ser uma linguagem de alto nível, é possível utilizar todos os recursos conhecidos de outras linguagens de alto nível, como:

- Variáveis
- Estruturas
- Operadores booleanos e aritméticos
- Estruturas de controle
- Funções digitais e analógicas

2.4 LINGUAGEM VB.NET

O VB.NET, é uma linguagem de programação orientada a objetos, que surgiu da união da linguagem Basic com os aspectos visuais do pacote, integrando o .NET devido fazer parte da plataforma Microsoft.NET. Ela foi desenvolvida em 2001 pela Microsoft Corporation, com foco principal na orientação a objetos, ampliando horizontes que antes não eram atingidos com o Visual Basic 6, que é a linguagem por muitos associada como antecessora do VB.NET, devido a semelhança entre elas.

2.4.1 Histórico da Linguagem

Segundo a Microsoft (2018), o VB.NET depende do .NET Framework, portanto costuma ter suas versões disponíveis de acordo com as atualizações do .NET Framework, e até a presente data somam-se dez versões lançadas:

- 2002 Ficou conhecida como VB 7.0 devido sua similaridade ao Visual Basic 6.

- 2003 Lançada em conjunto com o .NET Framework 1.1, e passou a ser disponibilizada para desenvolvimento no Visual Studio .NET 2003.
- 2005 Marcou a retirada do ".NET" de seu nome, passando a ser conhecida como Visual Basic 2005, focando na característica *RAD (Rapid application development).*
- 2008 Ficou conhecida como Visual Basic 9.0, e foi lançada em conjunto com o .NET Framework 3.5. Passou a englobar novos recursos como a inferência de tipo, métodos de extensão, tipos anônimos, operador condicional e várias outras implementações.
- 2010 Conhecida como Visual Basic 2010, surgiu junto com a inovação de trazer paridade com a linguagem C#.
- 2012 Foi lançada em conjunto com o .NET Framework 4.5, incluindo novos recursos de programação assíncrona, iteradores e hierarquia de chamadas.
- 2013 Foi lançada em conjunto com o .NET Framework 4.5.1, e ficou caracterizada por também ter a oportunidade de construir aplicativos .NET Framework.
- 2015 Intitulado como VB 14.0, foi lançada junto com a plataforma Visual Studio 2015, incluindo a opção de verificações nulas com o operador "?", além de permitir interpolação de *strings*.
- 2017 Denominado por VB 15.0, foi lançado junto com a plataforma Visual Studio 2017.
- 2019 Versão mais recente até o momento, é conhecida como VB 16.0 e foi lançada em conjunto com a plataforma Visual Studio 2019. Esta é a primeira versão do Visual Basic voltada para o.NET Core, apesar de dependerem de recursos que serão adicionados em versões posteriores. Permite conversões otimizadas de pontos flutuantes para inteiro, além de ser possível realizar comentários dentro de uma instrução na linha de comando.

2.4.2 Linguagem Estruturada

A programação estruturada é uma característica de linguagens de programação em ter seu desenvolvimento focado em práticas de blocos estruturados, executando chamada de dados e funções a partir de blocos (GUEDES, 2021).

A principal característica que define uma linguagem estruturada é a capacidade do programa seccionar e esconder o restante do programa as informações necessárias para realizar uma dada tarefa específica (SCHILDT, 1997).

Em uma linguagem estruturada é possível a realização de diferentes tipos de laços (*loops*) e condicionais, facilitando a programação, não exigindo um conceito rigoroso de campo, que é o caso de linguagens de alto nível não estruturadas como COBOL, FORTRAN e BASIC (SCHILDT, 1997).

A linguagem VB.NET apresenta todos esses aspectos estruturados, permitindo o que o programador crie seções de códigos que só sejam executadas quando determinada função, rotina ou método, seja invocado, garantindo que o seu funcionamento seja fluido e otimizado.

3 METODOLOGIA

Para um bom rendimento de um motor, é necessário mantê-lo em condições ideais de funcionamento, garantindo que seus parâmetros cruciais de funcionamento não excedam um limite máximo (geralmente estabelecido pelo fabricante).

Para o desenvolvimento deste presente trabalho, optou-se pela organização metodológica seccionada em três partes fundamentais, apresentadas a seguir.

3.1 PROTÓTIPO

Para o desenvolvimento deste trabalho, optou-se pela criação de um protótipo de testes, de forma a facilitar a instrumentação e coleta dos dados. O protótipo foi montado a partir de um motor elétrico extraído de uma lavadora, e toda a instrumentação foi realizada utilizando módulos e sensores com comunicação com a plataforma Arduino, incluindo sensores de temperatura, tensão, corrente, rotação e umidade relativa e temperatura ambiente.

Além da instrumentação, foi acoplado uma alavanca de freio que atuou diretamente na polia do eixo do motor. O freio teve atuação na variação de torque exigido do motor, devido ao torque de atrito ao qual o motor foi condicionado, fazendo com que suas condições operacionais não fossem constantes, de maneira que foi possível extrair dados de funcionamento que extrapolaram as condições ideais de uso colocando em prova o sistema desenvolvido. A Figura 10 mostra a alavanca de frenagem na polia do motor em sua posição aberta e fechada.

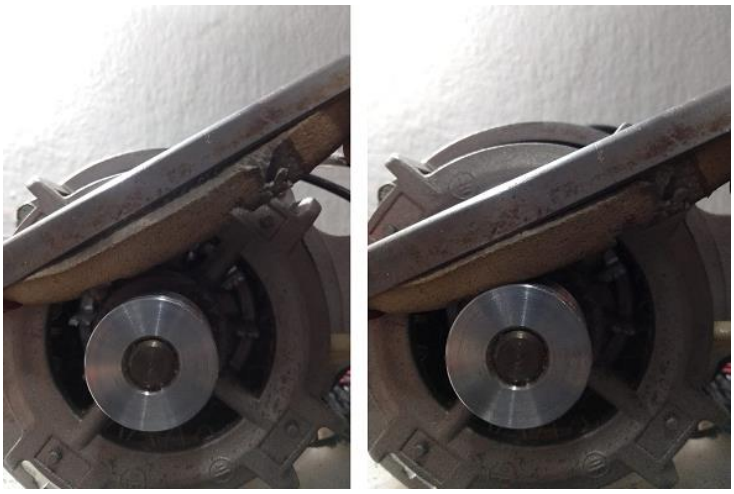

Figura 10- Alavanca de frenagem acoplada no motor de teste.
Fonte: Autoria própria.

3.1.1 Motor

O motor utilizado, foi extraído de uma lavadora (Figura 11), e é do tipo assíncrono com partida à capacitor.

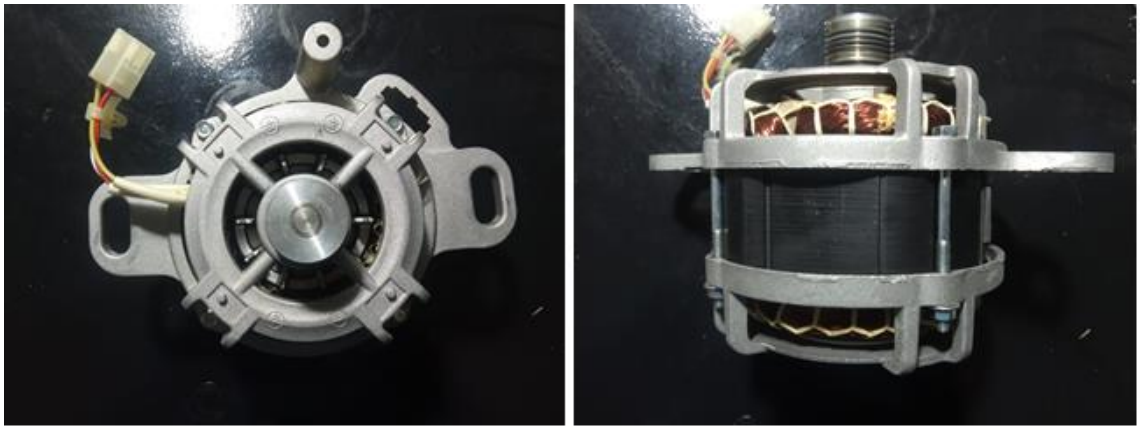

Figura 11- Motor de lavadora.
Fonte: Autoria própria.

As informações técnicas nominais de funcionamento, são apresentadas em um adesivo colado na carcaça do estator (Figura 12). Os principais dados são:

- Potência: 1/3 cv (aproximadamente 245 watts)
- Revoluções por minuto: 1620
- Corrente: 3 Ampères
- Tensão: 127 Volts

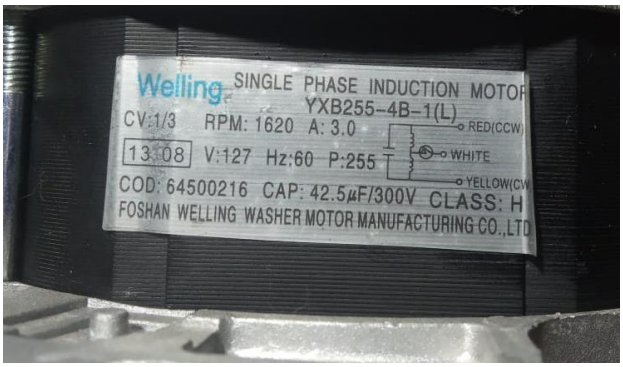

Figura 12- Informações técnicas do motor.
Fonte: Autoria própria.

3.1.2 Suporte

Para realizar a instrumentação do motor, e montar o protótipo, foi utilizado um suporte plástico extraído da mesma lavadora que foi extraído o motor. Esse suporte serviu de base de apoio para fixar o motor, e também os módulos em conjunto com o Arduino NANO. Os componentes eletrônicos não foram acoplados junto ao motor devido as elevadas temperaturas que poderiam danificar os componentes.

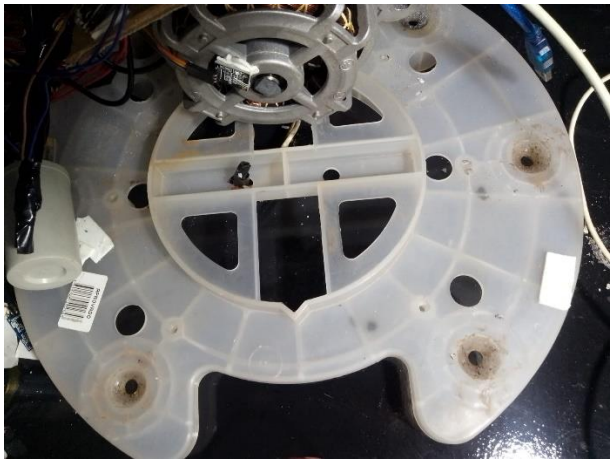

Figura 13- Base de fixação do motor.
Fonte: Autoria própria.

3.2 COLETA DE DADOS

3.2.1 Arduino

Para realizar a leitura dos dados de instrumentação do motor, foi utilizado um Arduino do modelo NANO V3.0. Como a instrumentação demandou apenas 6 sensores e um módulo de comunicação, a quantidade de portas solicitadas foi compatível com a quantidade de portas disponíveis neste modelo do Arduino (que somam 22 portas, entre 14 digitais e 8 analógicas).

Ele foi responsável por coletar os dados dos sensores instalados no motor, processá-los em um vetor *char* e enviá-los via *Wireless* para outro módulo configurado como receptor, acoplado à uma placa Arduino do tipo MEGA conectada via USB no servidor.

Não foi realizado nenhuma verificação condicional dos dados recebidos em comparação aos parâmetros ideais de funcionamento do motor em nenhuma das placas, esta tarefa foi direcionada ao *software* em VB.NET.

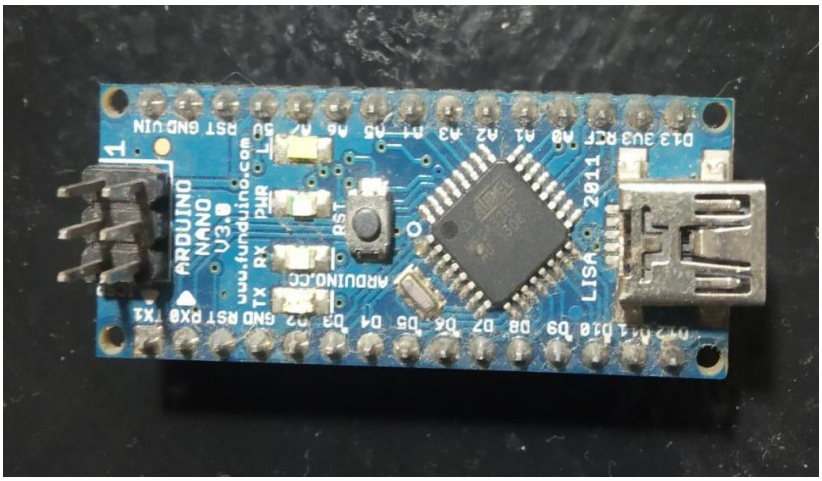

Figura 14- Arduino NANO.
Fonte: Autoria própria.

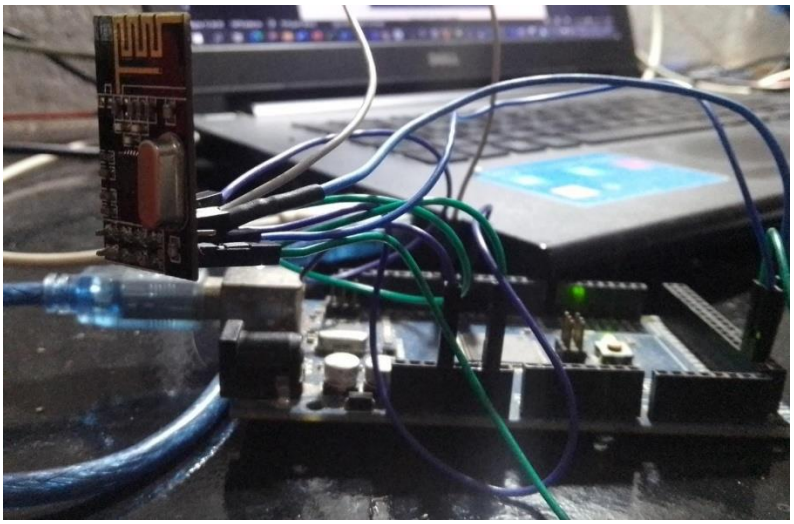

Figura 15- Arduino MEGA.
Fonte: Autoria própria.

3.2.2 Sensoriamento

Para realizar a instrumentação do motor, foram utilizados sensores compatíveis com a plataforma Arduino, de maneira que a comunicação pôde ser feita diretamente com as portas da placa. O sensoriamento foi realizado da seguinte maneira:

- 2 sensores de temperatura (um na carcaça e outro nas bobinas do estator)
- 1 sensor de corrente
- 1 sensor de tensão
- 1 sensor de rotação
- 1 sensor de temperatura ambiente

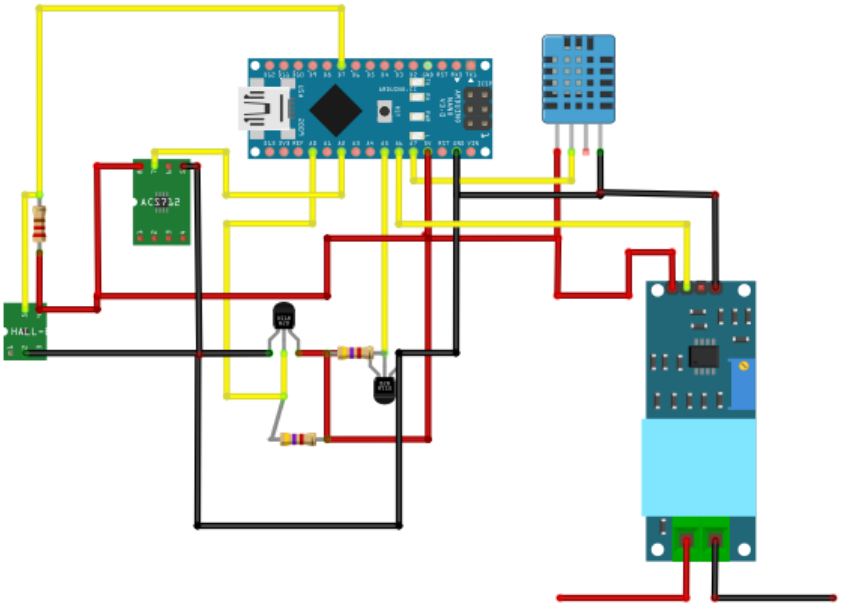

Figura 16- Diagrama elétrico.
Fonte: Autoria própria.

3.2.2.1 Temperatura

O sensor de temperatura utilizado, foi o DS18B20 apresentado na Figura 17. Ele possui as seguintes características:

- 3.0V a 5.5V alimentação/dados
- Precisão de ±0.5°C na faixa de –10 a +85°C
- Gama de temperatura utilizável: -55 a 125 ° C
- ID único de 64-bit gravado em cada sensor
- Vários sensores podem compartilhar um pino
- Tubo de aço inoxidável de 6mm x 35mm
- Diâmetro do cabo: 4mm
- Comprimento: 90cm

Figura 17- Sensor de temperatura DS18B20.
Fonte: (MERCADO LIVRE, 2021).

Foram utilizadas duas unidades deste sensor, sendo um deles posicionado entre as bobinas do estator (Figura 18), e o outro na parte externa da carcaça do motor (Figura 19).

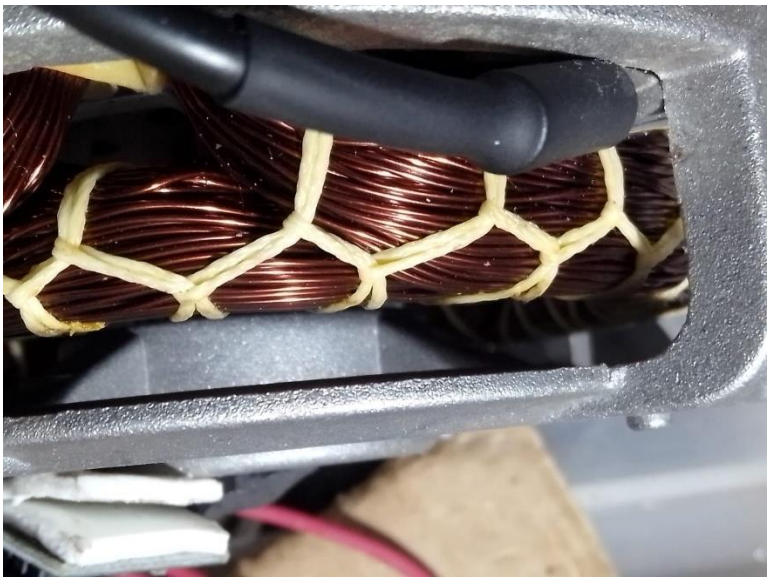

Figura 18- Sensor de temperatura posicionado nas bobinas do estator.
Fonte: Autoria própria.

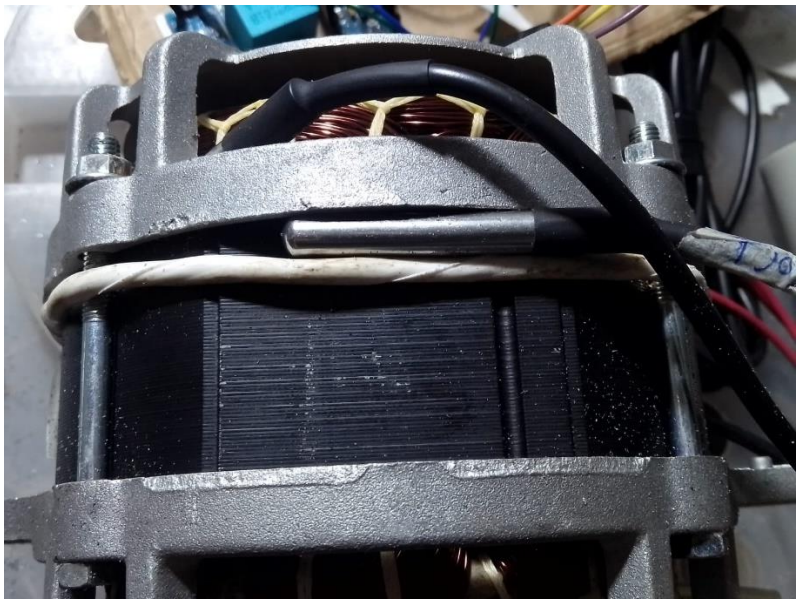

Figura 19- Sensor de temperatura acoplado na carcaça do motor.
Fonte: Autoria própria.

3.2.2.2 Corrente

Para avaliar a corrente consumida pelo motor, foi usado o sensor ACS712, que possuí as seguintes características de funcionamento:

- Tempo de resposta: 5 micro segundos
- Margem de erro: 1,5% a 25ºC
- Tensão de alimentação: 5V
- Dimensões aproximadas: 3,1 x 1,3 cm
- Razão de saída: 100mV por Ampere
- Leitura de corrente: Contínua ou Alternada

Figura 20- Sensor de corrente ACS712.
Fonte: (MERCADO LIVRE, 2021).

Este sensor caracteriza o tipo de sensor de corrente invasivo, o qual foi necessário cortar um dos fios de alimentação do motor, e conectá-los nos conectores do sensor (Figura 21).

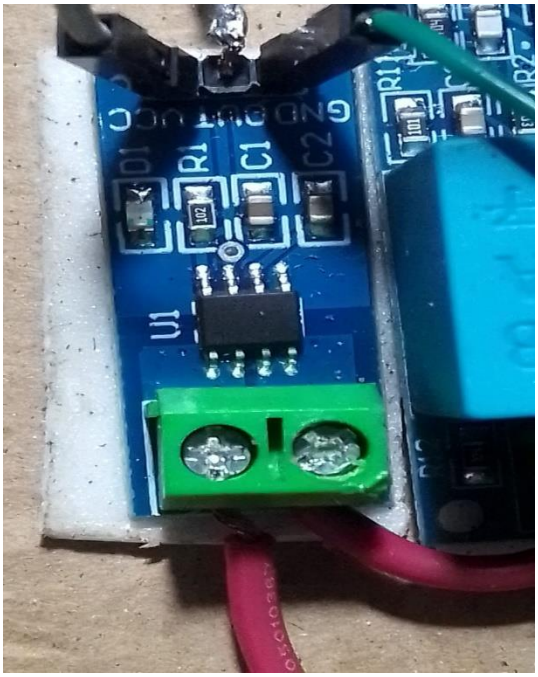

Figura 21- Ligação da alimentação do motor no sensor ACS712.
Fonte: Autoria própria.

3.2.2.3Tensão

A tensão utilizada pelo motor foi avaliada através do sensor ZMPT101B, que possuí as seguintes características:

- Tensão de alimentação do módulo: 5 a 30VDC
- Tensão de entrada: 0 a 250VAC
- Corrente de entrada nominal: 2mA
- Corrente de saída nominal: 2mA
- Proporção: 1000:1000
- Precisão de leitura: 0,5%
- Temperatura de operação: -40 a 70 celsius
- Dimensões: 22mm(L) X 20mm(A) X 51mm(C)

Figura 22- Sensor de tensão ZMPT101B.
Fonte: (MERCADO LIVRE, 2021).

Segundo informações fornecidas pelo próprio fabricante, este sensor necessita calibração. Para realizar a calibração, o sensor foi ligado em paralelo com um multímetro Fluke 302+ (Figura 23) e o ajuste foi realizado ajustando o trimpot de maneira que o valor impresso no Monitor Serial da IDE do Arduino, fosse idêntico ao valor do multímetro.

Figura 23- Multímetro Fluke 302+.
Fonte: Autoria própria.

3.2.2.4 Rotação

Para análise de rotação do motor optou-se pela escolha de um sensor de efeito *hall*, o Ky-003. Esse tipo de sensor detecta a variação de campo magnético, e para o seu funcionamento foi colado um ímã na parte traseira do eixo do motor. O sensor foi posicionado de maneira que fosse capaz de realizar a leitura das inversões de campo magnético. A Figura 24 ilustra o sensor posicionado próximo ao eixo do motor.

Figura 24- Posicionamento do sensor Ky-003.
Fonte: Autoria própria.

A partir da Figura 12 é possível extrair a informação que o motor trabalha com uma rotação nominal de 1620 revoluções por minuto. Portanto o sensor irá receber o dobro de pulsos por minuto. A quantidade de pulsos por unidade de tempo recebido pelo sensor, depende exclusivamente da quantidade de inversões de pólo magnético.

Este sensor apresenta as seguintes características técnicas:

- Tensão de Operação: 4,5 ~ 24V DC;
- Tempo de comutação de saída: 15μs;
- Tamanho: 29mm Largura x 15mm Profundidade x 6mm Altura;
- Peso: 2g.

O tempo de comutação de saída é uma informação crucial, com ele é possível avaliar o tempo entre uma leitura do sensor e a emissão do sinal. Outra informação fundamental é a resolução da porta lógica do microcontrolador utilizado, que para o caso do Arduino Nano utilizado, a resolução é de 10 bits. Para validar a utilização do sensor, foi realizado o cálculo de pulsos emitidos pelo sensor para a porta do Arduino da seguinte maneira:

$$1620[rpm] \cdot 2[polos] = 3240\left[\frac{pulsos}{\min}\right]$$

$$3240\left[\frac{pulsos}{\min}\right] \cdot \frac{1[\min]}{60[s]} = 54\left[\frac{pulsos}{s}\right]$$

$$\frac{1}{54\left[\frac{pulsos}{s}\right]} = 0{,}0185\left[\frac{s}{pulso}\right]$$

Isso implica que para dois pólos, é gerado um pulso para cada 0,0185 segundos, comparando com a resolução das portas do Arduino que permite uma frequência confortável de leituras por volta de 10kHz, ou 10000 leituras por segundo, fica evidente que como o tempo de comutação do sensor é bem inferior a esse tempo requerido por leitura, o sensor atende a necessidade.

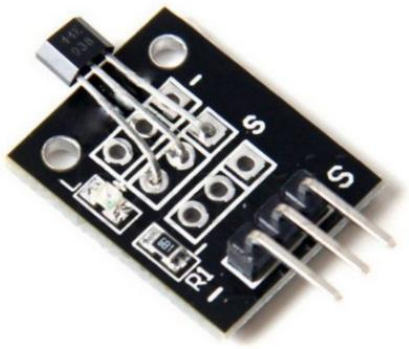

Figura 25- Sensor KY-003 3144.
Fonte: (MERCADO LIVRE, 2021).

3.2.2.5 Temperatura ambiente

Avaliar a temperatura do ambiente é uma tarefa fundamental para verificar a influência do ambiente no funcionamento do motor. Para esta análise foi utilizado o sensor DHT22. Este sensor é capaz de avaliar a temperatura e a umidade do ambiente, e possuí as seguintes características:

- Tensão de operação: 3-5VDC (5,5VDC máximo)
- Faixa de medição de umidade: 0 a 100% UR
- Faixa de medição de temperatura: -40º a +80ºC
- Corrente: 2,5mA durante uso, em *stand by* de 100uA a 150 uA
- Precisão de umidade de medição: ± 2,0% UR
- Precisão de medição de temperatura: ± 0,5 ºC
- Tempo de resposta: 2s
- Dimensões: 25 x 15 x 5mm (sem terminais)

Figura 26- Sensor DHT22
Fonte: (MERCADO LIVRE, 2021).

Este sensor foi ligado na placa com um cabo de aproximadamente 1,5 metros, fazendo com que o mesmo ficasse posicionado a uma boa distância do motor, para que não sofresse influência do insulflamento de ar quente proveniente da ventilação do motor.

3.2.3 Comunicação

A seção anterior apresentou características do sensoriamento montado no motor, todos os dispositivos anteriormente mencionados, ficaram próximos e/ou juntos ao motor, portanto, para enviar os dados obtidos pelo Arduino NANO para o servidor, este que normalmente fica longe dos equipamentos, foi necessário utilizar uma conexão *Wireless*. Para tal conexão foi utilizado o módulo NRF24L01, que é um módulo *transceiver* de 2.4 GHz com baixo consumo de energia.

Apesar de ser um dispositivo capaz de enviar e receber informações, cada um foi comissionado para apenas uma tarefa. O módulo instalado no Arduino NANO (Figura 27) ficou responsável por enviar os dados coletados dos sensores, enquanto o módulo instalado no Arduino MEGA (Figura 28), apenas recebeu os dados. A comunicação possuiu um *delay* entre envio de dados de 10 segundos. Esse valor baixo de *delay* foi definido apenas para facilitar na obtenção de dados e visualização dos resultados, mas em aplicação real o envio de dados pode possuir intervalos maiores.

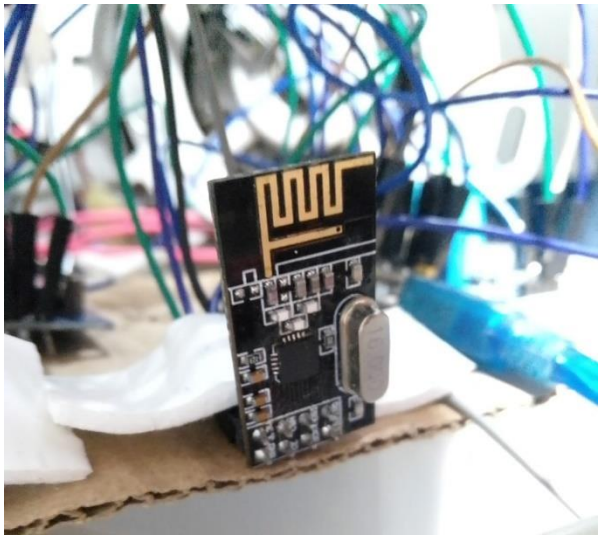

Figura 27- Módulo NRF24L01 transmissor.
Fonte: Autoria própria.

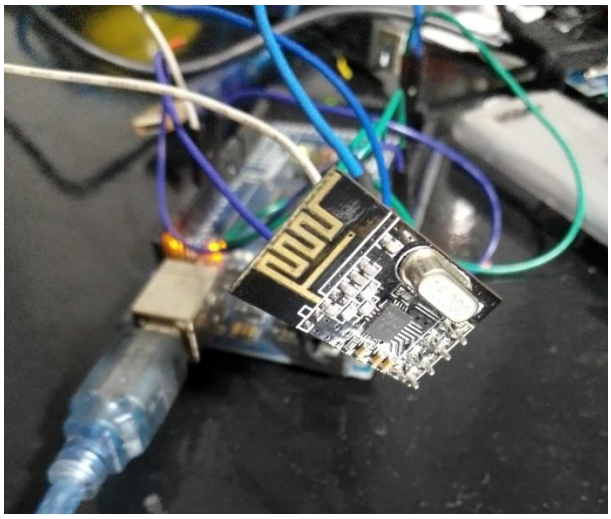

Figura 28- Módulo NRF24L01 receptor.
Fonte: Autoria própria.

A comunicação entre a placa Arduino MEGA e o servidor foi feita via conexão USB.

3.3 PROGRAMAÇÃO

3.3.1 *Software*

A programação do software foi toda desenvolvida em VB.NET, sendo sua principal função coletar os dados enviados do Arduino NANO para o Arduino MEGA, armazená-los em um banco de dados no computador, consultá-los com uma dada periodicidade, e exibi-los em um painel de controle contendo componentes gráficos, para que o usuário pudesse acompanhar a situação operacional do motor.

A estrutura de programação foi dividida entre

- Classe de Conexão com Banco de Dados: responsável por conectar a aplicação ao banco de dados em MySQL (ANEXO A).
- Classe de Design do Painel de Controle: responsável por conter todo o código do *design* da tela do Painel de Controle (ANEXO B).

- Classe do Formulário do Painel de Controle: responsável por conter toda programação da tela do Painel de Controle(ANEXO C).
- Classe de Design da tela de Parâmetros de Funcionamento: responsável por conter todo o código do *design* da tela do Parâmetros de Funcionamento (ANEXO D).
    - Classe do Formulário da tela de Parâmetros de Funcionamento: responsável por conter toda programação da tela da tela de Parâmetros de Funcionamento (ANEXO E).

A leitura dos dados na porta serial fora realizada usando o componente *SerialPort.* Para assegurar que não houvesse o envio desnecessário de dados pelo Arduino MEGA, a requisição foi feita via código no *software,* escrevendo na porta serial a letra “E” para que o Arduino MEGA escrevesse na porta serial os dados recebidos pelo Arduino NANO. Sem a requisição os dados não eram escritos.

Para garantir que toda vez que o usuário fechasse o software o mesmo não tivesse que imputar os parâmetros ideais de funcionamento novamente, esses dados ficaram salvos no banco de dados, e podem ser alterados de acordo com a necessidade do usuário.

Na estrutura de programação, parâmetros do motor em funcionamento, foram constantemente comparados com os parâmetros ideais de funcionamento do motor, cadastrados no banco de dados.

Foi realizado uma verificação condicional pra cada parâmetro que o usuário julgasse importante, e caso um dos parâmetros excedesse o limite operacional ideal, o usuário fora notificado na tela do *software* com a mudança de cor das *panels* e um *beep* sonoro.

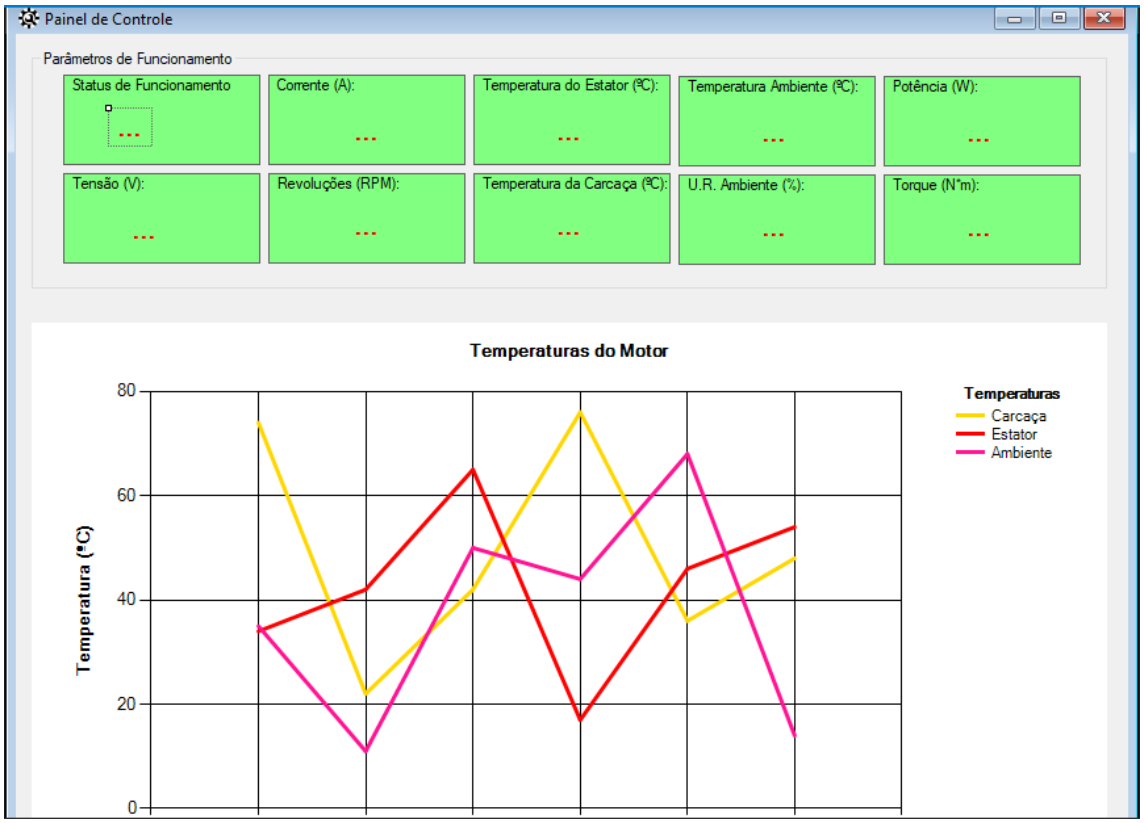

Figura 29- Tela de desenvolvimento do *software* em VB.NET.
Fonte: Autoria própria.

3.3.2 Banco de Dados

Para que os dados coletados pelo Arduino NANO através dos sensores fossem armazenados e posteriormente consultados, foi criado um banco de dados em MySql no software MySQL-Front.

O MySQL-Front é um software de gerenciamento de banco dados, com ele, é possível criar uma *database* com tabelas e relacionamento entre elas, permitindo que sejam armazenadas informações de diferentes tipos e formatos. Os dados coletados pelo Arduino NANO foram armazenados em uma tabela chamada “dados”. Essa tabela foi desenvolvida com 9 colunas de armazenamento, sendo elas cod_dados, carcaca, estator, tensao, corrente, rpm, umidade, temp_ambiente, data_dados, sendo que a primeira dessas colunas foi destinada ao código das informações. Esse código tem o objetivo de identificar cada informação que preenche o banco dados, ele nunca se repete, e tem

caráter de chave primária auto incremental, o ANEXO F apresenta o código para criar o banco.

Como o banco de dados é criado e manipulado através de linhas de código, foi possível manipulá-lo através de comandos enviados por uma "*sql*" do *software* em VB.NET.

Para que o *software* em VB.NET acessasse o banco de dados, foi utilizado o conector *mysql-connector-net* e o *software* XAMPP.

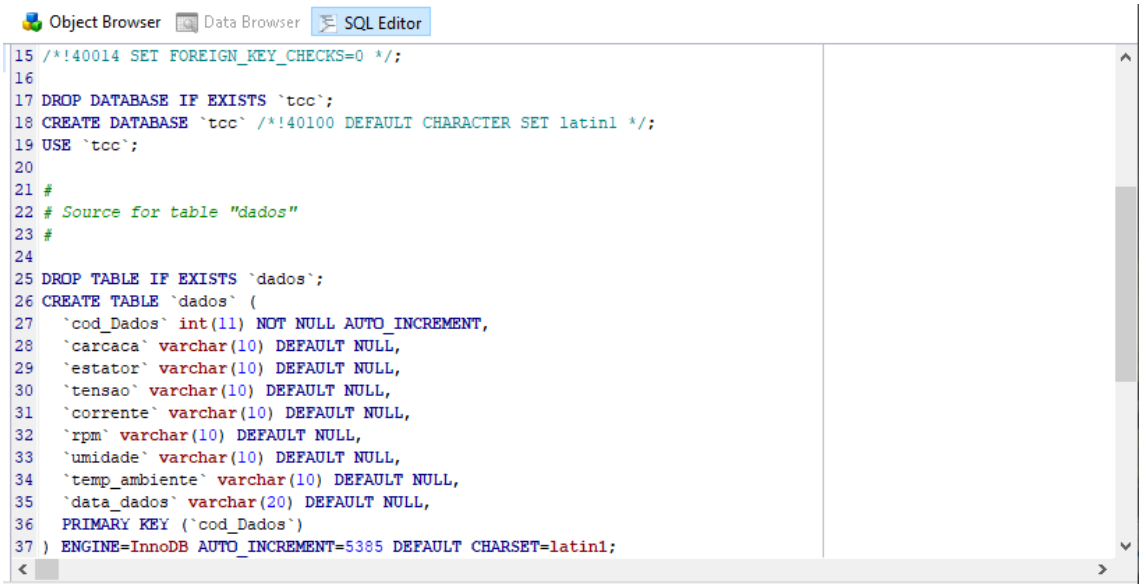

Figura 30- Interface do MySql-Front.
Fonte: Autoria própria.

4 RESULTADOS

Após a fixação do motor na base de apoio, a realização de toda instrumentação e programação, foi possível extrair os dados para avaliar o desempenho de todo sistema em funcionamento.

4.1 CALIBRAÇÃO DO SENSOR DE TENSÃO

No momento da leitura dos dados do sensor ZMPT101B, o mesmo apresentou dados incorretos, por volta de 60 a 62 volts, evidenciando a descalibragem do sensor. O *trimpot* foi então ajustado por volta de 6 giros no sentido horário até que seu valor igualasse com o multímetro. A medição realizada pelo multímetro, e os valores impressos pelo sensor no Monitor Serial são apresentados nas Figuras 31 e 32.

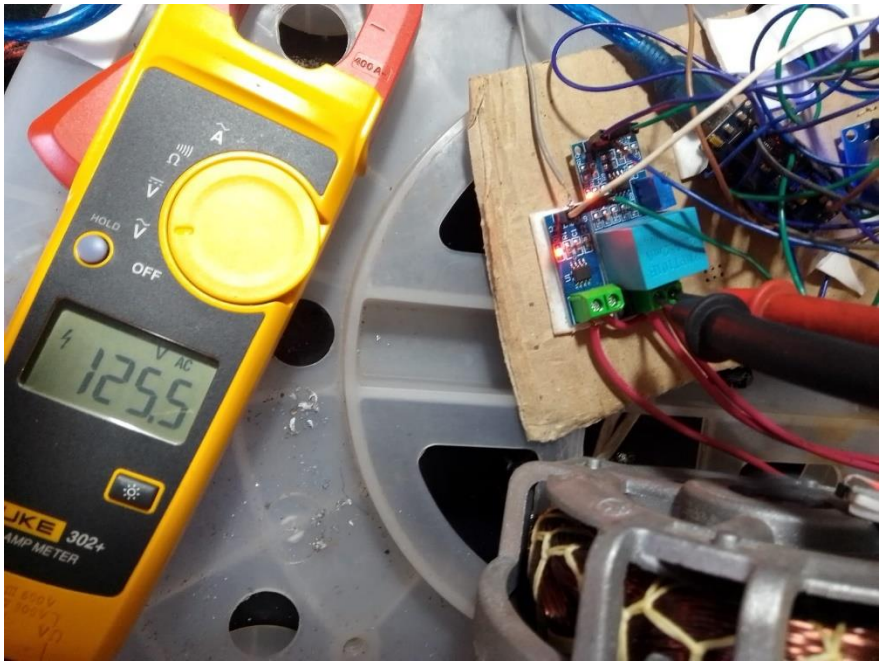

Figura 31- Calibração do sensor de tensão ZMPT101B.
Fonte: Autoria própria.

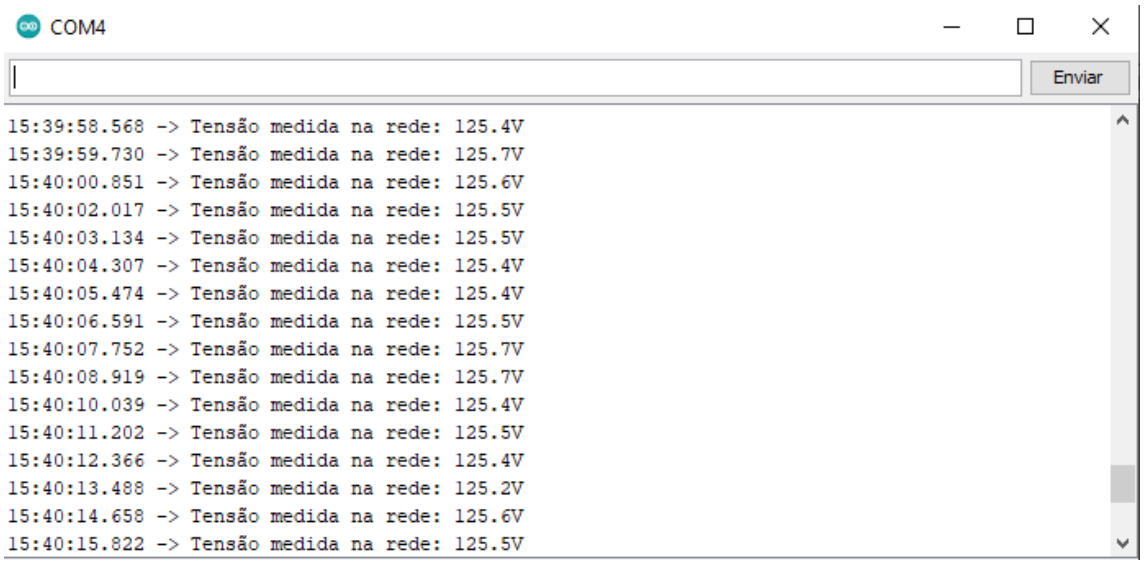

Figura 32- Valores de tensão lidos pelo sensor ZMPT101B.
Fonte: Autoria própria.

4.2 VALIDAÇÃO DO SENSOR DE CORRENTE

Para validar a leitura de corrente realizada pelo sensor ACS712-20A, foi novamente realizada a aferição pelo multímetro. Neste caso, o sensor não precisou ser ajustado, pois as leituras realizadas pelo mesmo, pouco destoavam do valor aferido pelo multímetro. Pequenas flutuações ocorreram esporadicamente devido a própria rede, porém foi possível notar o mesmo comportamento no sensor e no multímetro. As Figuras 33 e 34 mostram a aferição de corrente pelo multímetro e os valores de corrente provenientes do ACS712 impressos no Monitor Serial.

Figura 33- Leitura de corrente pelo multímetro Fluke 302+.
Fonte: Autoria própria.

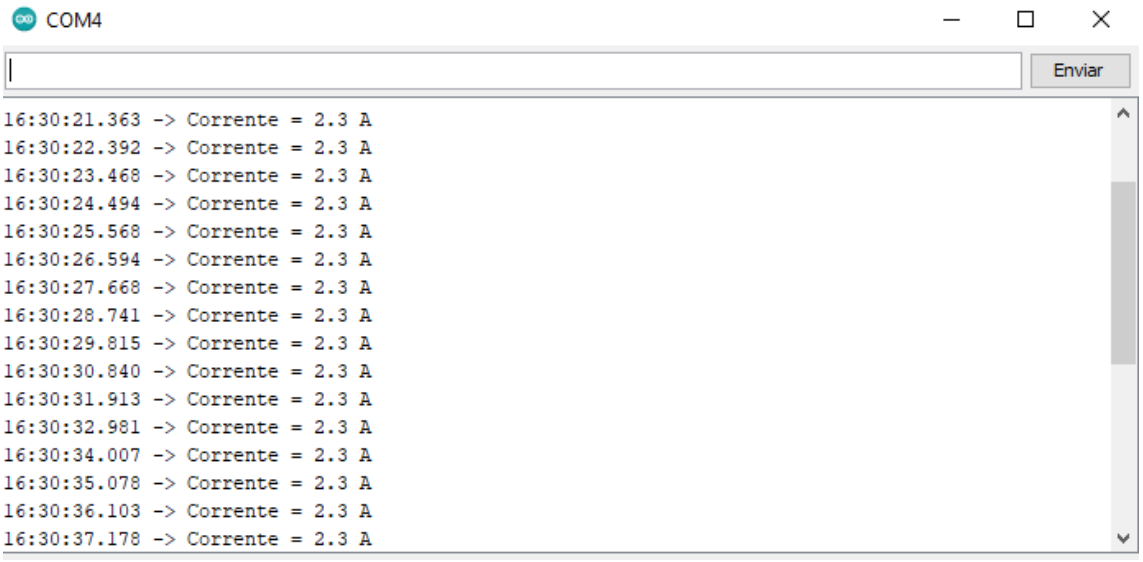

Figura 34- Valores de corrente lidos pelo sensor ACS712.
Fonte: Autoria própria.

4.3 COMUNICAÇÃO

Devido a ausência da implementação do valor de RSSI (*Received Sinal Strength Indicator*) na biblioteca do módulo NRF24L01, não foi possível realizar medições de intensidade de sinal, portanto para verificar a cobertura de sinal do módulo, ambos foram colocados para comunicar-se e afastados entre si em espaço aberto, e com barreiras de paredes. A distância média de comunicação em espaço aberto foi da ordem de 30 metros apresentando falha ou demora de resposta para distâncias maiores. Em região com obstáculos a distância foi da ordem de 8 a 10 metros.

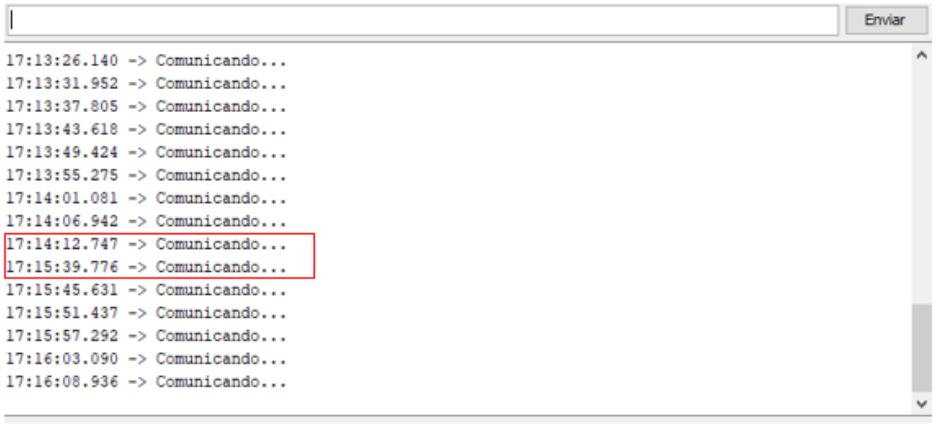

Figura 35- Comunicação entre transmissor e receptor.
Fonte: Autoria própria.

A Figura 35 ilustra o momento de testes de comunicação entre os dois módulos. É possível notar que em algum momento a comunicação para (destacado em vermelho na Figura 35), e o Monitor Serial não exibe mais os dados que antes eram recebidos de 5 em 5 segundos, isso mostra a perda de comunicação entre os módulos. Portanto, para evitar perdas de comunicação durante o funcionamento do *software*, o motor em conjunto com o Arduino NANO e toda instrumentação foram posicionados a cerca de 5 metros de distância do servidor.

A Figura 36 mostra os dados recebidos no Monitor Serial da placa Arduino MEGA. É possível notar que os dados foram multiplicados por 100 para remover casas decimais, facilitando a manipulação desses dados.

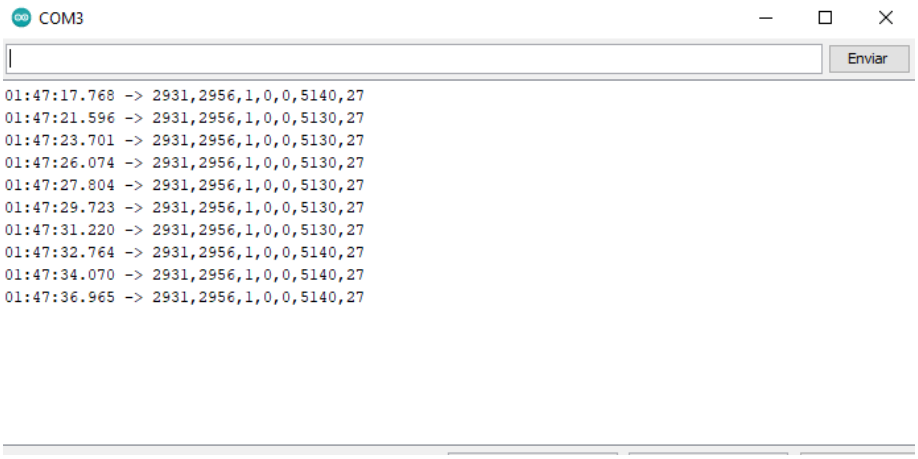

Figura 36- Comunicação dos dados do motor para o módulo receptor.
Fonte: Autoria própria.

4.4 SOFTWARE

A Figura 37 mostra a tela do Painel de Controle do *software.* O objetivo foi deixar os dados mais evidentes possível, de maneira que o usuário não necessitasse alternar entre abas para conseguir visualizar os parâmetros de funcionamento do motor. Todos os dados são consultados com um intervalo de 10 em 10 segundos, e atualizados na tela automaticamente. No gráfico foi representado os dados de temperatura do estator, da carcaça e a temperatura ambiente. Os dados do gráfico também seguiram os intervalos de consulta de 10 em 10 segundos.

É possível notar na Figura 37 o comportamento do perfil de temperatura momentos após ligar o motor, que inicialmente estava à temperatura ambiente, e atingiu quase 80ºC em apenas 10 minutos com comportamento crescente, sem carga na ponta do eixo.

Nota-se também que devido a ausência de carga na ponta do eixo o motor atinge revoluções maiores que a informada pelo fabricante na Figura 12, consumindo uma corrente menor.

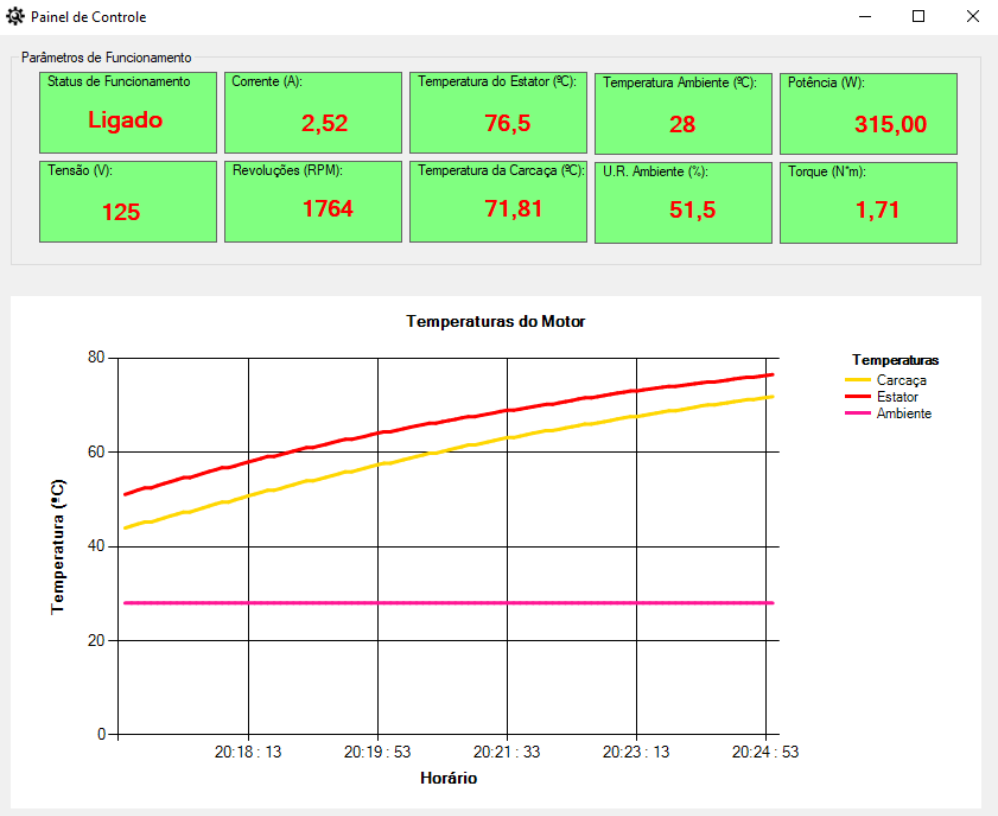

Figura 37- Tela do painel de controle do *software* desenvolvido.
Fonte: Autoria própria.

Após cerca de 45 minutos, a temperatura do motor estabilizou-se, mantendo um delta de aproximadamente 2 graus entre a temperatura do estator, e a temperatura da carcaça (Figura 38).

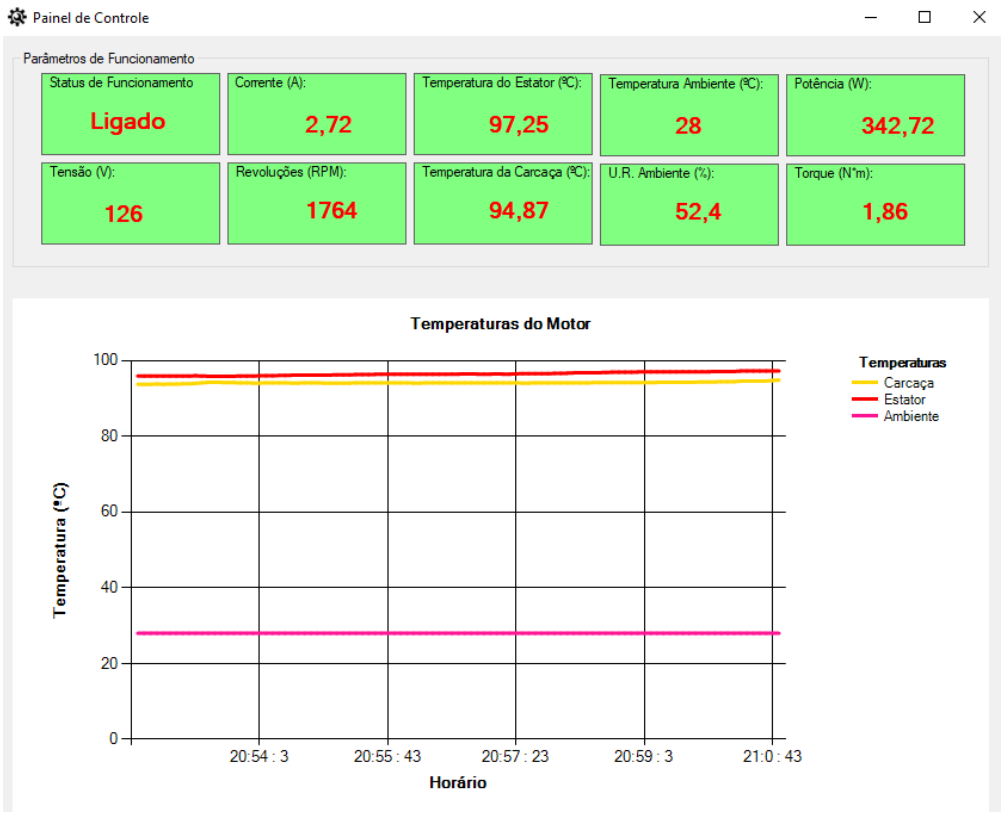

Figura 38- Condições operacionais após estabilização da temperatura.
Fonte: Autoria própria.

Ao passar o *mouse* sobre as *panels* informativas, o cursor muda para *hand* e permite que o usuário clique e configure as informações de limite máximo de operação (Figura 39). Vale observar que na frente das caixas de texto, foram colocadas *checkbox* para permitir que o usuário escolha se deseja que ao exceder o valor, toque o alarme. Para critério de testes do sistema foram estabelecidos os valores apresentados na Figura 39.

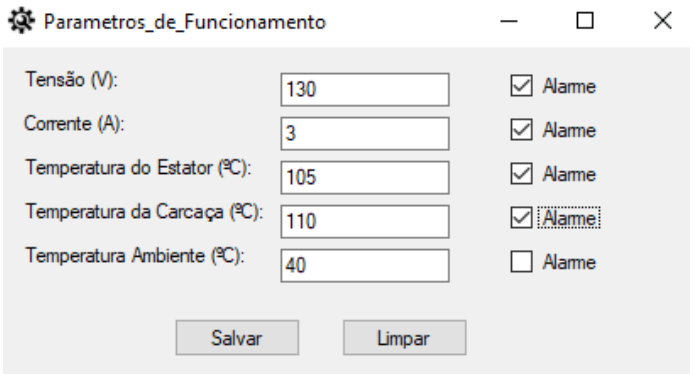

Figura 39- Tela de inserção dos parâmetros limites de funcionamento.
Fonte: Autoria própria.

Segundo Weg (2021), a temperatura máxima permitida de funcionamento de um motor elétrico, é limitada pela classe de isolamento das bobinas do estator. Ao verificar o selo do fabricante colado na carcaça do motor (Figura 12), é possível notar que a classe do motor é H, e comparando com os dados fornecidos na Figura 40, a temperatura máxima permitida é de 180ºC para uma temperatura ambiente de 40ºC.

Classe de isolamento		A	E	B	F	H
Temperatura ambiente	°C	40	40	40	40	40
Δt = elevação de temperatura (método da resistência)	°C	60	75	80	105	125
Diferença entre o ponto mais quente e a temperatura média	°C	5	5	10	10	15
Total: temperatura do ponto mais quente	°C	105	120	130	155	180

Figura 40- Temperaturas máximas permitidas por categorias de isolamento.
Fonte: (WEG, 2021).

O critério de temperatura máxima é calculado com base, na temperatura ambiente adicionado a elevação de temperatura máxima permitida, e a diferença de temperatura do ponto mais quente em relação à média, que para classe H é estabelecido por norma um valor de 15ºC. Portanto para verificar a temperatura máxima permitida no momento de funcionamento do motor bastou acrescer os valores da Figura 40 com a temperatura ambiente coletada pelo sensor DHT22.

$$T_{max}\,[^{o}C] = T_{amb} + 125 + 15$$

$$T_{max}[^{o}C] = 28 + 125 + 15$$

$$T_{max}[^{o}C] = 28 + 125 + 15$$

$$T_{max} = 168\,^{o}C$$

Comparando o valor informado pelo *software* (Figura 38) com a temperatura máxima calculada, é possível notar que o motor estava dentro dos limites de temperatura estabelecidos pelo isolamento do motor.

Após o funcionamento sem carga na ponta do eixo, foi aplicado um torque de atrito na polia da ponta do eixo do motor através da alavanca de

frenagem, causando um aumento imediato da corrente do consumida pelo motor, que subiu para valores acima do estabelecido nos parâmetros de funcionamento, fazendo com que a temperatura do motor começasse a subir também. Após poucos minutos a temperatura do motor excedeu os valores máximos estabelecidos para teste, disparando o alarme do software, e indicando em vermelho os valores fora do limite (Figura 41).

Figura 41- Parâmetros de operação fora dos limites estabelecidos.
Fonte: Autoria própria.

4.5 BANCO DE DADOS

Durante o funcionamento do *software*, todos os dados lidos na porta serial foram cadastrados no banco de dados, garantindo que os dados de funcionamento do motor não fossem perdidos. A Figura 42 mostra parte dos dados que foram populados no banco de dados durante os testes, cerca de pouco mais de 8 mil dados foram registrados (destacado em vermelho). Vale observar que os dados que continham valores decimais foram salvos no banco de dados multiplicados por 100, para facilitar o trânsito de variáveis entre as três plataformas (Arduino, VB.NET e MySQL).

cod_Dados	carcaca	estator	tensao	corrente	rpm	umidade	temp_ambie...	data_dados
518	7775	7993	126	258	1770	5960	28	20:7:28
519	7793	8012	126	262	1769	5970	28	20:7:40
520	7825	8037	126	262	1770	5930	28	20:7:52
521	7843	8056	126	258	1769	5940	28	20:8:4
522	7862	8081	125	258	1764	5930	28	20:8:16
523	7900	8131	126	258	1770	5940	28	20:8:28
524	7931	8156	126	255	1770	5920	28	20:8:40
525	7956	8175	126	262	1770	5920	28	20:8:52
526	7981	8187	126	265	1769	5920	28	20:9:4
527	8006	8206	126	265	1770	5910	28	20:9:16
528	8031	8225	126	262	1764	5930	28	20:9:28
529	8056	8243	126	262	1764	5930	28	20:9:40
530	8081	8268	126	262	1770	5920	28	20:9:52
531	8106	8281	126	262	1770	5920	28	20:10:4
532	8131	8300	126	262	1764	5930	28	20:10:16
533	8143	8325	126	262	1770	5930	28	20:10:28
534	8506	8625	125	262	1769	5900	28	20:15:25
535	8512	8637	125	265	1770	5890	28	20:15:37
536	8531	8650	124	258	1770	5890	28	20:15:49
537	8543	8662	126	269	1770	5880	28	20:16:1
538	8556	8668	126	265	1770	5880	28	20:16:13
539	8575	8675	122	255	1770	5880	28	20:16:25
540	8581	8681	126	265	1764	5870	28	20:16:37
541	8581	8687	126	265	1769	5860	28	20:16:49
542	8600	8693	126	258	1770	5880	28	20:17:1
543	8612	8700	126	265	1770	5910	28	20:17:13
544	8618	8693	126	265	1763	5890	28	20:17:25

531:3 8369 Record(s)

Figura 42- Dados aferidos pelo sistema salvos no banco dados.
Fonte: Autoria própria.

5 CONCLUSÃO

Portanto, após o desenvolvimento deste presente trabalho, foi possível obter informações de funcionamento do motor em tempo real de operação, permitindo o acompanhamento de variáveis que são de suma importância para a confiabilidade do equipamento. Apesar da aplicação em pequena escala (protótipo), o princípio de funcionamento do *software* continuará sendo o mesmo, limitando-se apenas aos tipos de sensores e controladores utilizados, que para motores maiores, exigirão dispositivos mais robustos.

Como observado durante a aquisição dos dados, a temperatura normal de funcionamento do motor atinge valores elevados, o que não permite saber por meio do tato se determinado equipamento encontra-se em condições normais de operação. O *software* por sua vez, após configurado com parâmetros definidos pelo usuário, que podem ser extraídos de fichas técnicas ou manuais de fabricantes, permite que o operador de máquinas e/ou o gerente de manutenção acompanhe as condições operacionais dos equipamentos com uma riqueza muito maior de detalhes.

O armazenamento dos dados no banco de dados, permite que o usuário crie um acervo de condições operacionais de seus equipamentos, que pode ser correlacionado com horário de produção e demanda produtiva. A avaliação das curvas de temperatura, pode levar por exemplo, a previsão de uma possível falha do isolamento do estator, perda de capacitância do capacitor, ou má lubrificação dos rolamentos, proporcionando assim a otimização do processo como um todo, garantindo manutenções programadas e mais assertivas.

Por fim, após a elaboração de todo o sistema, desde a aquisição dos dados até a exibição no *software,* atendeu as condições esperadas, permitindo uma avaliação minuciosa do funcionamento do motor, garantindo que o usuário possa prever possíveis falhas futuras através da análise dos valores exibidos na tela do painel de controle, auxiliado também pela indicação de alerta visual e sonoro.

REFERÊNCIAS

ALMEIDA, Marcio Tadeu. **Manutenção preditiva: benefícios e lucratividade**, 2008.

ARDUINO. **Arduino Software (IDE)**. Disponível em: https://www.arduino.cc/en/Guide/Environment. Acesso em: 30 jul. 2021.

Cavalcante, Cristiano Alexandre Virgínio e Almeida, Adiel Teixeira de. **Modelo multicritério de apoio a decisão para o planejamento de manutenção preventiva utilizando PROMETHEE II em situações de incerteza**. Pesquisa Operacional [online]. 2005, v. 25, n. 2 [Acessado 25 Julho 2021] , pp. 279-296. Disponível em: <https://doi.org/10.1590/S0101-74382005000200007>. Epub 13 Set 2005. ISSN 1678-5142. https://doi.org/10.1590/S0101-74382005000200007.

CPFL ENERGIA. Bônus Motor: **Saída ideal para a eficiência energética nas empresas:** projeto tem foco nos equipamentos que mais consomem energia elétrica no brasil e representam uma grande oportunidade para indústria, comércio e serviços. Projeto tem foco nos equipamentos que mais consomem energia elétrica no Brasil e representam uma grande oportunidade para indústria, comércio e serviços. Disponível em: https://www.canalenergia.com.br/noticias/53155142/bonus-motor-saida-ideal-para-a-eficiencia-energetica-nas-empresas. Acesso em: 04 jul. 2021.

FRANCHI, Claiton Moro. **Acionamentos Elétricos**. 3. ed. São Paulo: Editora Érica Ltda., 2008. 250 p.

GUEDES, Fernanda Lopes. **Programação Estruturada**. Disponível em: http://tics.ifsul.edu.br/matriz/conteudo/disciplinas/_pdf/apostila_prog_estruturada.pdf. Acesso em: 28 jul. 2021.

GUIA CNC. **Motor Monofásico**. 2003. Disponível em: http:// http://www.guiacnc.com.br/motores-em-geral/motor-monofasico/. Acesso em: 25 jul. 2021.

KARDEC, Alan; NASCIF, Júlio Aquino. **MANUTENÇÃO**: função estratégica. 3. ed. Rio de Janeiro: Qualitymark, 2009. 384 p.

MERCADO LIVRE. **Hall sensor magnético**. Disponível em: https://produto.mercadolivre.com.br/MLB-830456073-hall-sensor-magnetico-modulo-ky003-esp8266-arduino-nodemcu-_JM?matt_tool=18956390&utm_source=google_shopping&utm_medium=organic. Acesso em: 08 ago. 2021.

MERCADO LIVRE. **Módulo sensor DHT11**. Disponível em: https://produto.mercadolivre.com.br/MLB-1898223884-modulo-sensor-dht11-temperatura-e-umidade-arduino-pic-_JM?matt_tool=18956390&utm_source=google_shopping&utm_medium=organic. Acesso em: 08 ago. 2021.

MERCADO LIVRE. **Módulo sensor tensão**. Disponível em: https://produto.mercadolivre.com.br/MLB-1669365908-modulo-sensor-tensao-ac-0-a-250v-voltimetro-zmpt101b-arduino-_JM?matt_tool=18956390&utm_source=google_shopping&utm_medium=organic. Acesso em: 08 ago. 2021.

MERCADO LIVRE. **Sensor corrente módulo arduino**. Disponível em: https://produto.mercadolivre.com.br/MLB-1343788105-5a-acs712-sensor-corrente-modulo-arduino-esp8266-_JM?matt_tool=18956390&utm_source=google_shopping&utm_medium=organic. Acesso em: 08 ago. 2021.

MERCADO LIVRE. **Sensor de temperatura a prova dagua**. Disponível em: https://produto.mercadolivre.com.br/MLB-688214480-sensor-de-temperatura-a-prova-d-agua-ds18b20-pic-arduino-_JM?matt_tool=18956390&utm_source=google_shopping&utm_medium=organic. Acesso em: 08 ago. 2021.

MICROSOFT. **O que há de novo no Visual Basic**. 2018. Disponível em: https://docs.microsoft.com/en-us/dotnet/visual-basic/whats-new/. Acesso em: 28 jul. 2021.

MINISTÉRIO DA CIÊNCIA E TECNOLOGIA - MCT (Brasil). **ESTUDO DA COMPETITIVIDADE DA INDÚSTRIA BRASILEIRA**. Campinas: Frfr, 1993. 334 p. Disponível em: https://www.bibliotecaagptea.org.br/administracao/agroindustria/livros/ESTUDO%20DA%20COMPETITIVIDADE%20DA%20INDUSTRIA%20BRASILEIRA.pdf. Acesso em: 01 ago. 2021.

MORO, Norberto; AURAS, André Paegle. **INTRODUÇÃO À GESTÃO DA MANUTENÇÃO**. 2007. Disponível em: https://norbertocefetsc.pro.br/downloads/manutencao.pdf. Acesso em: 14 jul. 2021.

MULTILÓGICA-SHOP. **Guia Arduino Iniciante**. Disponível em: https://multilogica-shop.com/guia-arduino-iniciante. Acesso em: 30 jul. 2021.

PINHEIRO, Diego da Silva. **ANÁLISE DOS DISTÚRBIOS NO TRANSITÓRIO DE PARTIDA DE MOTORES DE INDUÇÃO TRIFÁSICOS**. 2015. 177 f. TCC (Graduação) - Curso de Engenharia Elétrica, Universidade Federal do Pará, Tucuruí, 2015.

PINTO, Alan Kardec; XAVIER, Júlio Aquino Nascif. **Manutenção: função estratégica**. 2. ed. Rio de Janeiro: Editora Qalitymark, 2003.

PROCEL. **PROCEL INDÚSTRIA**: motor elétrico guia básico. Brasília: Editoração Eletrônica Link Design, 2009. 190 p.

SCHILDT, Herbert. **C Completo e Total**. 3. ed. São Paulo: Makron Books do Brasil Editora Ltda, 1997. 811 p. Tradução de Roberto Carlos Mayer.

WEG. **Electric Machinery Company**: Motores Síncronos. Motores Síncronos. Disponível em: https://www.weg.net/catalog/weg/BR/pt/Motores-El%C3%A9tricos/Motores-S%C3%ADncronos/Electric-Machinery-Company---Motores-S%C3%ADncronos/Electric-Machinery-Company---Motores-S%C3%ADncronos/p/MKT_WEN_MOTOR_SYNC_WEM. Acesso em: 25 jul. 2021.

WEG. **GUIA DE ESPECIFICAÇÃO**: motores elétricos. Motores Elétricos. Disponível em: https://static.weg.net/medias/downloadcenter/h32/hc5/WEG-motores-eletricos-guia-de-especificacao-50032749-brochure-portuguese-web.pdf. Acesso em: 16 nov. 2021.

WEG. **Motores Síncronos**. Disponível em: https://static.weg.net/medias/downloadcenter/h13/h7e/WEG-motores-sincronos-50005369-catalogo-portugues-br.pdf. Acesso em: 26 jul. 2021.

ANEXO A – CÓDIGO DA CLASSE DE CONEXÃO

```
Imports MySql.Data.MySqlClient
Imports System.IO
Public Class Classe_de_Conexao


    Public con As New MySqlConnection
    Public cmd As MySqlCommand = con.CreateCommand
    Public da As MySqlDataAdapter = New MySqlDataAdapter(cmd)
    Public ds As New DataSet


    Dim fs As FileStream
    Dim br As BinaryReader


    Public Sub conectar()
        Try
            con.ConnectionString = "server=localhost;user id=root;
password=;database=tcc"
            con.Open()
            'MsgBox("Conexão realizada com sucesso!")
        Catch ex As Exception
            MsgBox(ex.Message & "Erro no Método conectar!!")
        End Try
    End Sub


    Public Sub Operar(ByVal sql)

        conectar()
        Try
            cmd.CommandText = sql
            cmd.ExecuteNonQuery()
            'MsgBox("Operação realizada com sucesso!", MsgBoxStyle.Information,
"Informação")
        Catch ex As Exception
            MsgBox(ex.Message & ex.ToString, MsgBoxStyle.Critical, "Erro no
método operar")
        Finally
            con.Close()

        End Try
    End Sub
    Public Function listar(ByVal sql As String)
        conectar()
        Try
            ds.Clear()
            cmd.CommandText = sql
            da.Fill(ds)
        Catch ex As Exception
            MsgBox(ex.Message & "Erro no método listar")
        Finally
            con.Close()
        End Try
        Return ds
    End Function

End Class
```

ANEXO B – CÓDIGO DO DESIGN DO PAINEL DE CONTROLE

```
<Global.Microsoft.VisualBasic.CompilerServices.DesignerGenerated()> _
Partial Class Painel_de_Controle
    Inherits System.Windows.Forms.Form

    'Descartar substituições de formulário para limpar a lista de componentes.
    <System.Diagnostics.DebuggerNonUserCode()> _
    Protected Overrides Sub Dispose(ByVal disposing As Boolean)
        Try
            If disposing AndAlso components IsNot Nothing Then
                components.Dispose()
            End If
        Finally
            MyBase.Dispose(disposing)
        End Try
    End Sub

    'Exigido pelo Windows Form Designer
    Private components As System.ComponentModel.IContainer

    'OBSERVAÇÃO: o procedimento a seguir é exigido pelo Windows Form Designer
    'Pode ser modificado usando o Windows Form Designer.
    'Não o modifique usando o editor de códigos.


      <System.Diagnostics.DebuggerStepThrough()> _


Private Sub InitializeComponent()
        Me.components = New System.ComponentModel.Container()
        Dim ChartArea1 As
System.Windows.Forms.DataVisualization.Charting.ChartArea = New
System.Windows.Forms.DataVisualization.Charting.ChartArea()
        Dim Legend1 As System.Windows.Forms.DataVisualization.Charting.Legend =
New System.Windows.Forms.DataVisualization.Charting.Legend()
        Dim Series1 As System.Windows.Forms.DataVisualization.Charting.Series =
New System.Windows.Forms.DataVisualization.Charting.Series()
        Dim Series2 As System.Windows.Forms.DataVisualization.Charting.Series =
New System.Windows.Forms.DataVisualization.Charting.Series()
        Dim Series3 As System.Windows.Forms.DataVisualization.Charting.Series =
New System.Windows.Forms.DataVisualization.Charting.Series()
        Dim Title1 As System.Windows.Forms.DataVisualization.Charting.Title = New
System.Windows.Forms.DataVisualization.Charting.Title()
        Dim resources As System.ComponentModel.ComponentResourceManager = New
System.ComponentModel.ComponentResourceManager(GetType(Painel_de_Controle))
        Me.Chart1 = New System.Windows.Forms.DataVisualization.Charting.Chart()
        Me.Timer1 = New System.Windows.Forms.Timer(Me.components)
        Me.Timer2 = New System.Windows.Forms.Timer(Me.components)
        Me.Timer3 = New System.Windows.Forms.Timer(Me.components)
        Me.Timer4 = New System.Windows.Forms.Timer(Me.components)
        Me.Timer5 = New System.Windows.Forms.Timer(Me.components)
        Me.Timer6 = New System.Windows.Forms.Timer(Me.components)
        Me.Timer7 = New System.Windows.Forms.Timer(Me.components)
        Me.Timer8 = New System.Windows.Forms.Timer(Me.components)
        Me.SerialPort1 = New System.IO.Ports.SerialPort(Me.components)
        Me.Label1 = New System.Windows.Forms.Label()
        Me.Label2 = New System.Windows.Forms.Label()
```

```
        Me.Label3 = New System.Windows.Forms.Label()
        Me.Label4 = New System.Windows.Forms.Label()
        Me.Label5 = New System.Windows.Forms.Label()
        Me.Label6 = New System.Windows.Forms.Label()
        Me.p_tensao = New System.Windows.Forms.Panel()
        Me.p_corrente = New System.Windows.Forms.Panel()
        Me.p_rpm = New System.Windows.Forms.Panel()
        Me.p_potencia = New System.Windows.Forms.Panel()
        Me.p_torque = New System.Windows.Forms.Panel()
        Me.p_status = New System.Windows.Forms.Panel()
        Me.lbl_tensao = New System.Windows.Forms.Label()
        Me.lbl_corrente = New System.Windows.Forms.Label()
        Me.lbl_potencia = New System.Windows.Forms.Label()
        Me.lbl_torque = New System.Windows.Forms.Label()
        Me.lbl_rpm = New System.Windows.Forms.Label()
        Me.lbl_status = New System.Windows.Forms.Label()
        Me.GroupBox1 = New System.Windows.Forms.GroupBox()
        Me.p_estator = New System.Windows.Forms.Panel()
        Me.lbl_estator = New System.Windows.Forms.Label()
        Me.Label8 = New System.Windows.Forms.Label()
        Me.p_carcaca = New System.Windows.Forms.Panel()
        Me.lbl_carcaca = New System.Windows.Forms.Label()
        Me.Label10 = New System.Windows.Forms.Label()
        Me.p_amb = New System.Windows.Forms.Panel()
        Me.lbl_amb = New System.Windows.Forms.Label()
        Me.Label12 = New System.Windows.Forms.Label()
        Me.p_ur = New System.Windows.Forms.Panel()
        Me.lbl_ur = New System.Windows.Forms.Label()
        Me.Label14 = New System.Windows.Forms.Label()
        CType(Me.Chart1, System.ComponentModel.ISupportInitialize).BeginInit()
        Me.p_tensao.SuspendLayout()
        Me.p_corrente.SuspendLayout()
        Me.p_rpm.SuspendLayout()
        Me.p_potencia.SuspendLayout()
        Me.p_torque.SuspendLayout()
        Me.p_status.SuspendLayout()
        Me.GroupBox1.SuspendLayout()
        Me.p_estator.SuspendLayout()
        Me.p_carcaca.SuspendLayout()
        Me.p_amb.SuspendLayout()
        Me.p_ur.SuspendLayout()
        Me.SuspendLayout()
        '
        'Chart1
'
        ChartArea1.AxisX.Title = "Horário"
        ChartArea1.AxisX.TitleFont = New System.Drawing.Font("Microsoft Sans
Serif", 10.0!, System.Drawing.FontStyle.Bold)
        ChartArea1.AxisY.Title = "Temperatura (ºC)"
        ChartArea1.AxisY.TitleFont = New System.Drawing.Font("Microsoft Sans
Serif", 10.0!, System.Drawing.FontStyle.Bold)
        ChartArea1.Name = "ChartArea1"
        Me.Chart1.ChartAreas.Add(ChartArea1)
        Legend1.Name = "Legend1"
        Legend1.Title = "Temperaturas"
        Me.Chart1.Legends.Add(Legend1)
        Me.Chart1.Location = New System.Drawing.Point(12, 220)
        Me.Chart1.Name = "Chart1"
        Me.Chart1.Palette =
System.Windows.Forms.DataVisualization.Charting.ChartColorPalette.Fire
        Series1.BorderWidth = 3
        Series1.ChartArea = "ChartArea1"
```

```
        Series1.ChartType =
System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Line
        Series1.Legend = "Legend1"
        Series1.Name = "Carcaça"
        Series2.BorderWidth = 3
        Series2.ChartArea = "ChartArea1"
        Series2.ChartType =
System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Line
        Series2.Legend = "Legend1"
        Series2.Name = "Estator"
        Series3.BorderWidth = 3
        Series3.ChartArea = "ChartArea1"
        Series3.ChartType =
System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Line
        Series3.Legend = "Legend1"
        Series3.Name = "Ambiente"
        Me.Chart1.Series.Add(Series1)
        Me.Chart1.Series.Add(Series2)
        Me.Chart1.Series.Add(Series3)
        Me.Chart1.Size = New System.Drawing.Size(818, 432)
        Me.Chart1.TabIndex = 0
        Me.Chart1.Text = "Chart1"
        Title1.Font = New System.Drawing.Font("Microsoft Sans Serif", 10.0!,
System.Drawing.FontStyle.Bold)
        Title1.Name = "Title1"
        Title1.Text = "Temperaturas do Motor"
        Me.Chart1.Titles.Add(Title1)
        '
        'Timer1
        '
        Me.Timer1.Interval = 3000
        '
        'Timer2
        '
        Me.Timer2.Interval = 5000
        '
        'Timer3
        '
        Me.Timer3.Interval = 1500
        '
        'Timer4
        '
        Me.Timer4.Interval = 1000
        '
        'Timer5
        '
        Me.Timer5.Interval = 1000
        '
        'Timer6
        '
        Me.Timer6.Interval = 1000
        '
        'Timer7
        '
        Me.Timer7.Interval = 1000
        '
        'Timer8
        '
        Me.Timer8.Interval = 1000
        '
        'SerialPort1
        '
```

```
        Me.SerialPort1.PortName = "COM3"
        '
        'Label1
        '
Me.Label1.AutoSize = True
        Me.Label1.Location = New System.Drawing.Point(3, 0)
        Me.Label1.Name = "Label1"
        Me.Label1.Size = New System.Drawing.Size(62, 13)
        Me.Label1.TabIndex = 16
        Me.Label1.Text = "Tensão (V):"
        '
        'Label2
        '
        Me.Label2.AutoSize = True
        Me.Label2.Location = New System.Drawing.Point(3, 0)
        Me.Label2.Name = "Label2"
        Me.Label2.Size = New System.Drawing.Size(66, 13)
        Me.Label2.TabIndex = 17
        Me.Label2.Text = "Corrente (A):"
        '
        'Label3
        '
        Me.Label3.AutoSize = True
        Me.Label3.Location = New System.Drawing.Point(3, 0)
        Me.Label3.Name = "Label3"
        Me.Label3.Size = New System.Drawing.Size(100, 13)
        Me.Label3.TabIndex = 18
        Me.Label3.Text = "Revoluções (RPM):"
        '
        'Label4
        '
        Me.Label4.AutoSize = True
        Me.Label4.Location = New System.Drawing.Point(3, 0)
        Me.Label4.Name = "Label4"
        Me.Label4.Size = New System.Drawing.Size(72, 13)
        Me.Label4.TabIndex = 19
        Me.Label4.Text = "Potência (W):"
        '
        'Label5
        '
        Me.Label5.AutoSize = True
        Me.Label5.Location = New System.Drawing.Point(3, 0)
        Me.Label5.Name = "Label5"
        Me.Label5.Size = New System.Drawing.Size(73, 13)
        Me.Label5.TabIndex = 20
        Me.Label5.Text = "Torque (N*m):"
        '
        'Label6
        '
        Me.Label6.AutoSize = True
        Me.Label6.Location = New System.Drawing.Point(3, 0)
        Me.Label6.Name = "Label6"
        Me.Label6.Size = New System.Drawing.Size(128, 13)
        Me.Label6.TabIndex = 21
        Me.Label6.Text = "Status de Funcionamento"
        '
        'p_tensao
        '
        Me.p_tensao.BackColor = System.Drawing.Color.FromArgb(CType(CType(128,
Byte), Integer), CType(CType(255, Byte), Integer), CType(CType(128, Byte),
Integer))
        Me.p_tensao.BorderStyle = System.Windows.Forms.BorderStyle.FixedSingle
```

```
        Me.p_tensao.Controls.Add(Me.lbl_tensao)
        Me.p_tensao.Controls.Add(Me.Label1)
        Me.p_tensao.Location = New System.Drawing.Point(24, 94)
        Me.p_tensao.Name = "p_tensao"
        Me.p_tensao.Size = New System.Drawing.Size(150, 69)
        Me.p_tensao.TabIndex = 22
        '
        'p_corrente
        '
        Me.p_corrente.BackColor = System.Drawing.Color.FromArgb(CType(CType(128,
Byte), Integer), CType(CType(255, Byte), Integer), CType(CType(128, Byte),
Integer))
        Me.p_corrente.BorderStyle = System.Windows.Forms.BorderStyle.FixedSingle
        Me.p_corrente.Controls.Add(Me.lbl_corrente)
        Me.p_corrente.Controls.Add(Me.Label2)
        Me.p_corrente.Location = New System.Drawing.Point(180, 19)
        Me.p_corrente.Name = "p_corrente"
        Me.p_corrente.Size = New System.Drawing.Size(150, 69)
        Me.p_corrente.TabIndex = 23
        '
        'p_rpm
        '
        Me.p_rpm.BackColor = System.Drawing.Color.FromArgb(CType(CType(128,
Byte), Integer), CType(CType(255, Byte), Integer), CType(CType(128, Byte),
Integer))
        Me.p_rpm.BorderStyle = System.Windows.Forms.BorderStyle.FixedSingle
        Me.p_rpm.Controls.Add(Me.lbl_rpm)
        Me.p_rpm.Controls.Add(Me.Label3)
        Me.p_rpm.Location = New System.Drawing.Point(180, 94)
        Me.p_rpm.Name = "p_rpm"
        Me.p_rpm.Size = New System.Drawing.Size(150, 69)
        Me.p_rpm.TabIndex = 23
        '
        'p_potencia
        '
        Me.p_potencia.BackColor = System.Drawing.Color.FromArgb(CType(CType(128,
Byte), Integer), CType(CType(255, Byte), Integer), CType(CType(128, Byte),
Integer))
        Me.p_potencia.BorderStyle = System.Windows.Forms.BorderStyle.FixedSingle
        Me.p_potencia.Controls.Add(Me.lbl_potencia)
        Me.p_potencia.Controls.Add(Me.Label4)
        Me.p_potencia.Location = New System.Drawing.Point(648, 20)
        Me.p_potencia.Name = "p_potencia"
        Me.p_potencia.Size = New System.Drawing.Size(150, 69)
        Me.p_potencia.TabIndex = 23
        '
        'p_torque
        '
        Me.p_torque.BackColor = System.Drawing.Color.FromArgb(CType(CType(128,
Byte), Integer), CType(CType(255, Byte), Integer), CType(CType(128, Byte),
Integer))
        Me.p_torque.BorderStyle = System.Windows.Forms.BorderStyle.FixedSingle
        Me.p_torque.Controls.Add(Me.lbl_torque)
        Me.p_torque.Controls.Add(Me.Label5)
        Me.p_torque.Location = New System.Drawing.Point(648, 95)
        Me.p_torque.Name = "p_torque"
        Me.p_torque.Size = New System.Drawing.Size(150, 69)
        Me.p_torque.TabIndex = 24
        '
        'p_status
        '
```

```
        Me.p_status.BackColor = System.Drawing.Color.FromArgb(CType(CType(128,
Byte), Integer), CType(CType(255, Byte), Integer), CType(CType(128, Byte),
Integer))
        Me.p_status.BorderStyle = System.Windows.Forms.BorderStyle.FixedSingle
        Me.p_status.Controls.Add(Me.lbl_status)
        Me.p_status.Controls.Add(Me.Label6)
        Me.p_status.Location = New System.Drawing.Point(24, 19)
        Me.p_status.Name = "p_status"
        Me.p_status.Size = New System.Drawing.Size(150, 69)
        Me.p_status.TabIndex = 22
        '
        'lbl_tensao
        '
        Me.lbl_tensao.AutoSize = True
        Me.lbl_tensao.Font = New System.Drawing.Font("Microsoft Sans Serif",
14.0!, System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, CType(0,
Byte))
        Me.lbl_tensao.ForeColor = System.Drawing.Color.Red
        Me.lbl_tensao.Location = New System.Drawing.Point(47, 30)
        Me.lbl_tensao.Name = "lbl_tensao"
        Me.lbl_tensao.Size = New System.Drawing.Size(28, 24)
        Me.lbl_tensao.TabIndex = 17
        Me.lbl_tensao.Text = "..."
        '
        'lbl_corrente
        '
        Me.lbl_corrente.AutoSize = True
        Me.lbl_corrente.Font = New System.Drawing.Font("Microsoft Sans Serif",
14.0!, System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, CType(0,
Byte))
        Me.lbl_corrente.ForeColor = System.Drawing.Color.Red
        Me.lbl_corrente.Location = New System.Drawing.Point(60, 30)
        Me.lbl_corrente.Name = "lbl_corrente"
        Me.lbl_corrente.Size = New System.Drawing.Size(28, 24)
        Me.lbl_corrente.TabIndex = 18
        Me.lbl_corrente.Text = "..."
        '
        'lbl_potencia
        '
        Me.lbl_potencia.AutoSize = True
        Me.lbl_potencia.Font = New System.Drawing.Font("Microsoft Sans Serif",
14.0!, System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, CType(0,
Byte))
        Me.lbl_potencia.ForeColor = System.Drawing.Color.Red
        Me.lbl_potencia.Location = New System.Drawing.Point(58, 30)
        Me.lbl_potencia.Name = "lbl_potencia"
        Me.lbl_potencia.Size = New System.Drawing.Size(28, 24)
        Me.lbl_potencia.TabIndex = 19
        Me.lbl_potencia.Text = "..."
        '
        'lbl_torque
        '
        Me.lbl_torque.AutoSize = True
        Me.lbl_torque.Font = New System.Drawing.Font("Microsoft Sans Serif",
14.0!, System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, CType(0,
Byte))
        Me.lbl_torque.ForeColor = System.Drawing.Color.Red
        Me.lbl_torque.Location = New System.Drawing.Point(58, 27)
        Me.lbl_torque.Name = "lbl_torque"
        Me.lbl_torque.Size = New System.Drawing.Size(28, 24)
        Me.lbl_torque.TabIndex = 20
        Me.lbl_torque.Text = "..."
```

```
        '
        'lbl_rpm
        '
Me.lbl_rpm.AutoSize = True
        Me.lbl_rpm.Font = New System.Drawing.Font("Microsoft Sans Serif", 14.0!,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, CType(0, Byte))
        Me.lbl_rpm.ForeColor = System.Drawing.Color.Red
        Me.lbl_rpm.Location = New System.Drawing.Point(60, 27)
        Me.lbl_rpm.Name = "lbl_rpm"
        Me.lbl_rpm.Size = New System.Drawing.Size(28, 24)
        Me.lbl_rpm.TabIndex = 21
        Me.lbl_rpm.Text = "..."
        '
        'lbl_status
        '
        Me.lbl_status.AutoSize = True
        Me.lbl_status.Font = New System.Drawing.Font("Microsoft Sans Serif",
14.0!, System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, CType(0,
Byte))
        Me.lbl_status.ForeColor = System.Drawing.Color.Red
        Me.lbl_status.Location = New System.Drawing.Point(36, 27)
        Me.lbl_status.Name = "lbl_status"
        Me.lbl_status.Size = New System.Drawing.Size(73, 24)
        Me.lbl_status.TabIndex = 22
        Me.lbl_status.Text = "Ligado"
        '
        'GroupBox1
        '
        Me.GroupBox1.Controls.Add(Me.p_tensao)
        Me.GroupBox1.Controls.Add(Me.p_carcaca)
        Me.GroupBox1.Controls.Add(Me.p_ur)
        Me.GroupBox1.Controls.Add(Me.p_torque)
        Me.GroupBox1.Controls.Add(Me.p_estator)
        Me.GroupBox1.Controls.Add(Me.p_status)
        Me.GroupBox1.Controls.Add(Me.p_amb)
        Me.GroupBox1.Controls.Add(Me.p_potencia)
        Me.GroupBox1.Controls.Add(Me.p_corrente)
        Me.GroupBox1.Controls.Add(Me.p_rpm)
        Me.GroupBox1.Location = New System.Drawing.Point(12, 12)
        Me.GroupBox1.Name = "GroupBox1"
        Me.GroupBox1.Size = New System.Drawing.Size(818, 183)
        Me.GroupBox1.TabIndex = 25
        Me.GroupBox1.TabStop = False
        Me.GroupBox1.Text = "Parâmetros de Funcionamento"
        '
        'p_estator
        '
        Me.p_estator.BackColor = System.Drawing.Color.FromArgb(CType(CType(128,
Byte), Integer), CType(CType(255, Byte), Integer), CType(CType(128, Byte),
Integer))
        Me.p_estator.BorderStyle = System.Windows.Forms.BorderStyle.FixedSingle
        Me.p_estator.Controls.Add(Me.lbl_estator)
        Me.p_estator.Controls.Add(Me.Label8)
        Me.p_estator.Location = New System.Drawing.Point(336, 19)
        Me.p_estator.Name = "p_estator"
        Me.p_estator.Size = New System.Drawing.Size(150, 69)
        Me.p_estator.TabIndex = 23
        '
        'lbl_estator
        '
        Me.lbl_estator.AutoSize = True
```

```
        Me.lbl_estator.Font = New System.Drawing.Font("Microsoft Sans Serif",
14.0!, System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, CType(0,
Byte))
        Me.lbl_estator.ForeColor = System.Drawing.Color.Red
        Me.lbl_estator.Location = New System.Drawing.Point(58, 30)
        Me.lbl_estator.Name = "lbl_estator"
        Me.lbl_estator.Size = New System.Drawing.Size(28, 24)
        Me.lbl_estator.TabIndex = 19
        Me.lbl_estator.Text = "..."
        '
        'Label8
        '
        Me.Label8.AutoSize = True
        Me.Label8.Location = New System.Drawing.Point(3, 0)
        Me.Label8.Name = "Label8"
        Me.Label8.Size = New System.Drawing.Size(141, 13)
        Me.Label8.TabIndex = 19
        Me.Label8.Text = "Temperatura do Estator (ºC):"
        '
        'p_carcaca
        '
        Me.p_carcaca.BackColor = System.Drawing.Color.FromArgb(CType(CType(128,
Byte), Integer), CType(CType(255, Byte), Integer), CType(CType(128, Byte),
Integer))
        Me.p_carcaca.BorderStyle = System.Windows.Forms.BorderStyle.FixedSingle
        Me.p_carcaca.Controls.Add(Me.lbl_carcaca)
        Me.p_carcaca.Controls.Add(Me.Label10)
        Me.p_carcaca.Location = New System.Drawing.Point(336, 94)
        Me.p_carcaca.Name = "p_carcaca"
        Me.p_carcaca.Size = New System.Drawing.Size(150, 69)
        Me.p_carcaca.TabIndex = 24
        '
        'lbl_carcaca
        '
        Me.lbl_carcaca.AutoSize = True
        Me.lbl_carcaca.Font = New System.Drawing.Font("Microsoft Sans Serif",
14.0!, System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, CType(0,
Byte))
        Me.lbl_carcaca.ForeColor = System.Drawing.Color.Red
        Me.lbl_carcaca.Location = New System.Drawing.Point(58, 27)
        Me.lbl_carcaca.Name = "lbl_carcaca"
        Me.lbl_carcaca.Size = New System.Drawing.Size(28, 24)
        Me.lbl_carcaca.TabIndex = 20
        Me.lbl_carcaca.Text = "..."
        '
        'Label10
        '
        Me.Label10.AutoSize = True
        Me.Label10.Location = New System.Drawing.Point(3, 0)
        Me.Label10.Name = "Label10"
        Me.Label10.Size = New System.Drawing.Size(148, 13)
        Me.Label10.TabIndex = 20
        Me.Label10.Text = "Temperatura da Carcaça (ºC):"
        '
        'p_amb
        '
        Me.p_amb.BackColor = System.Drawing.Color.FromArgb(CType(CType(128,
Byte), Integer), CType(CType(255, Byte), Integer), CType(CType(128, Byte),
Integer))
        Me.p_amb.BorderStyle = System.Windows.Forms.BorderStyle.FixedSingle
        Me.p_amb.Controls.Add(Me.lbl_amb)
        Me.p_amb.Controls.Add(Me.Label12)
```

```
        Me.p_amb.Location = New System.Drawing.Point(492, 20)
        Me.p_amb.Name = "p_amb"
        Me.p_amb.Size = New System.Drawing.Size(150, 69)
        Me.p_amb.TabIndex = 23
        '
        'lbl_amb
        '
        Me.lbl_amb.AutoSize = True
        Me.lbl_amb.Font = New System.Drawing.Font("Microsoft Sans Serif", 14.0!,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, CType(0, Byte))
        Me.lbl_amb.ForeColor = System.Drawing.Color.Red
        Me.lbl_amb.Location = New System.Drawing.Point(58, 30)
        Me.lbl_amb.Name = "lbl_amb"
        Me.lbl_amb.Size = New System.Drawing.Size(28, 24)
        Me.lbl_amb.TabIndex = 19
        Me.lbl_amb.Text = "..."
        '
        'Label12
        '
        Me.Label12.AutoSize = True
        Me.Label12.Location = New System.Drawing.Point(3, 0)
        Me.Label12.Name = "Label12"
        Me.Label12.Size = New System.Drawing.Size(137, 13)
        Me.Label12.TabIndex = 19
        Me.Label12.Text = "Temperatura Ambiente (ºC):"
        '
        'p_ur
        '
        Me.p_ur.BackColor = System.Drawing.Color.FromArgb(CType(CType(128, Byte),
Integer), CType(CType(255, Byte), Integer), CType(CType(128, Byte), Integer))
        Me.p_ur.BorderStyle = System.Windows.Forms.BorderStyle.FixedSingle
        Me.p_ur.Controls.Add(Me.lbl_ur)
        Me.p_ur.Controls.Add(Me.Label14)
        Me.p_ur.Location = New System.Drawing.Point(492, 95)
        Me.p_ur.Name = "p_ur"
        Me.p_ur.Size = New System.Drawing.Size(150, 69)
        Me.p_ur.TabIndex = 24
        '
        'lbl_ur
        '
        Me.lbl_ur.AutoSize = True
        Me.lbl_ur.Font = New System.Drawing.Font("Microsoft Sans Serif", 14.0!,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, CType(0, Byte))
        Me.lbl_ur.ForeColor = System.Drawing.Color.Red
        Me.lbl_ur.Location = New System.Drawing.Point(58, 27)
        Me.lbl_ur.Name = "lbl_ur"
        Me.lbl_ur.Size = New System.Drawing.Size(28, 24)
        Me.lbl_ur.TabIndex = 20
        Me.lbl_ur.Text = "..."
        '
        'Label14
        '
        Me.Label14.AutoSize = True
        Me.Label14.Location = New System.Drawing.Point(3, 0)
        Me.Label14.Name = "Label14"
        Me.Label14.Size = New System.Drawing.Size(96, 13)
        Me.Label14.TabIndex = 20
        Me.Label14.Text = "U.R. Ambiente (%):"
        '
        'Painel_de_Controle
        '
        Me.AutoScaleDimensions = New System.Drawing.SizeF(6.0!, 13.0!)
```

```
        Me.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font
        Me.BackColor = System.Drawing.SystemColors.Control
        Me.ClientSize = New System.Drawing.Size(846, 664)
        Me.Controls.Add(Me.GroupBox1)
        Me.Controls.Add(Me.Chart1)
        Me.Icon = CType(resources.GetObject("$this.Icon"), System.Drawing.Icon)
        Me.Name = "Painel_de_Controle"
        Me.StartPosition = System.Windows.Forms.FormStartPosition.CenterScreen
        Me.Text = "Painel de Controle"
        CType(Me.Chart1, System.ComponentModel.ISupportInitialize).EndInit()
        Me.p_tensao.ResumeLayout(False)
        Me.p_tensao.PerformLayout()
        Me.p_corrente.ResumeLayout(False)
        Me.p_corrente.PerformLayout()
        Me.p_rpm.ResumeLayout(False)
        Me.p_rpm.PerformLayout()
        Me.p_potencia.ResumeLayout(False)
        Me.p_potencia.PerformLayout()
        Me.p_torque.ResumeLayout(False)
        Me.p_torque.PerformLayout()
        Me.p_status.ResumeLayout(False)
        Me.p_status.PerformLayout()
        Me.GroupBox1.ResumeLayout(False)
        Me.p_estator.ResumeLayout(False)
        Me.p_estator.PerformLayout()
        Me.p_carcaca.ResumeLayout(False)
        Me.p_carcaca.PerformLayout()
        Me.p_amb.ResumeLayout(False)
        Me.p_amb.PerformLayout()
        Me.p_ur.ResumeLayout(False)
        Me.p_ur.PerformLayout()
        Me.ResumeLayout(False)

    End Sub
Friend WithEvents Chart1 As DataVisualization.Charting.Chart
    Friend WithEvents Timer1 As Timer
    Friend WithEvents Timer2 As Timer
    Friend WithEvents Timer3 As Timer
    Friend WithEvents Timer4 As Timer
    Friend WithEvents Timer5 As Timer
    Friend WithEvents Timer6 As Timer
    Friend WithEvents Timer7 As Timer
    Friend WithEvents Timer8 As Timer
    Friend WithEvents SerialPort1 As IO.Ports.SerialPort
    Friend WithEvents Label1 As Label
    Friend WithEvents Label2 As Label
    Friend WithEvents Label3 As Label
    Friend WithEvents Label4 As Label
    Friend WithEvents Label5 As Label
    Friend WithEvents Label6 As Label
    Friend WithEvents p_tensao As Panel
    Friend WithEvents p_corrente As Panel
    Friend WithEvents p_rpm As Panel
    Friend WithEvents p_potencia As Panel
    Friend WithEvents p_torque As Panel
    Friend WithEvents p_status As Panel
    Friend WithEvents lbl_tensao As Label
    Friend WithEvents lbl_corrente As Label
    Friend WithEvents lbl_rpm As Label
    Friend WithEvents lbl_potencia As Label
    Friend WithEvents lbl_torque As Label
    Friend WithEvents lbl_status As Label
```

```
    Friend WithEvents GroupBox1 As GroupBox
    Friend WithEvents p_carcaca As Panel
    Friend WithEvents lbl_carcaca As Label
    Friend WithEvents Label10 As Label
    Friend WithEvents p_ur As Panel
    Friend WithEvents lbl_ur As Label
    Friend WithEvents Label14 As Label
    Friend WithEvents p_estator As Panel
    Friend WithEvents lbl_estator As Label
    Friend WithEvents Label8 As Label
    Friend WithEvents p_amb As Panel
    Friend WithEvents lbl_amb As Label
    Friend WithEvents Label12 As Label
End Class
```

ANEXO C – CÓDIGO DO FORMULÁRIO DO PAINEL DE CONTROLE.

```
Public Class Painel_de_Controle
    Dim Horario, Horario_fim, data As DateTime
    Dim dados As New Dados

    Dim ds As DataSet
    Dim sql As String
    Dim con As New Classe_de_Conexao

    ' Dim valores
    Dim data_x, data_
    Dim carcaca, estator, tensao, corrente, rpm, umidade, temp_ambiente,
data_dados As String

    Private Sub Timer3_Tick(sender As Object, e As EventArgs) Handles Timer3.Tick
        System.Console.Beep(1500, 1000)
    End Sub

    Dim carcaca_limite, estator_limite, tensao_limite, corrente_limite,
rpm_limite, umidade_limite, temp_ambiente_limite, data_dados_limite As Double



    Dim valores_carcaca(100) As Double
    Dim valores_interna(100) As Double
    Dim valores_ambiente(100) As Double
    'Dim valores_carcaca(10) As Double
    Dim datas(100) As String
    Dim i = 0
    Private Sub Timer2_Tick(sender As Object, e As EventArgs) Handles Timer2.Tick

        Try
            SerialPort1.Open()
            SerialPort1.Write("E")
            Dim a(7) As String
            Dim valor = SerialPort1.ReadLine
            SerialPort1.Close()


            a = Split(valor, ",")
            carcaca = a(0)
            estator = a(1)
            tensao = a(2)
            corrente = a(3)
            rpm = a(4)
            umidade = a(5)
            temp_ambiente = a(6)
            data_dados = Date.Now.Hour & ":" & Date.Now.Minute & ":" &
Date.Now.Second

            If CType(tensao, Double) > 90 Then
                lbl_status.Text = "Ligado"
            Else
                lbl_status.Text = "Desligado"
            End If

            Try
                lbl_tensao.Text = CType(tensao, Double)
                lbl_corrente.Text = CType(corrente, Double) / 100
```

```
                lbl_rpm.Text = CType(rpm, Double)
                lbl_estator.Text = CType(estator, Double) / 100
                lbl_carcaca.Text = CType(carcaca, Double) / 100
                lbl_amb.Text = CType(temp_ambiente, Double)
                lbl_ur.Text = CType(umidade, Double) / 100
                lbl_potencia.Text = Format((CType(tensao, Double)) *
(CType(corrente, Double) / 100), "0.00")
                lbl_torque.Text = Format(9550 * ((CType(tensao, Double)) *
(CType(corrente, Double) / 100)) / (1000 * CType(rpm, Double)), "0.00")

                If CType(tensao, Double) > tensao_limite Or CType(corrente,
Double) / 100 > corrente_limite Or CType(estator, Double) / 100 > estator_limite
Or CType(carcaca, Double) / 100 > carcaca_limite Then
                    Timer3.Enabled = True
                Else
                    Timer3.Enabled = False
                End If
                If CType(tensao, Double) > tensao_limite Then
                    p_tensao.BackColor = Color.FromArgb(255, 192, 192)

                Else
                    p_tensao.BackColor = Color.FromArgb(128, 255, 128)

                End If
                If CType(corrente, Double) / 100 > corrente_limite Then
                    p_corrente.BackColor = Color.FromArgb(255, 192, 192)

                Else
                    p_corrente.BackColor = Color.FromArgb(128, 255, 128)

                End If
                If CType(estator, Double) / 100 > estator_limite Then
                    p_estator.BackColor = Color.FromArgb(255, 192, 192)

                Else
                    p_estator.BackColor = Color.FromArgb(128, 255, 128)

                End If
                If CType(carcaca, Double) / 100 > carcaca_limite Then
                    p_carcaca.BackColor = Color.FromArgb(255, 192, 192)

                Else
                    p_carcaca.BackColor = Color.FromArgb(128, 255, 128)

                End If

            Catch ex As Exception

            End Try

            '255,192,192
            Try
                If carcaca IsNot Nothing And estator IsNot Nothing And tensao
IsNot Nothing And corrente IsNot Nothing And rpm IsNot Nothing And umidade IsNot
Nothing And temp_ambiente IsNot Nothing And data_dados IsNot Nothing Then


                    sql = "Insert into
dados(carcaca,estator,tensao,corrente,rpm,umidade,temp_ambiente,data_dados)values
('" & carcaca & "','" & estator & "','" & tensao & "','" & corrente & "','" & rpm
& "','" & umidade & "','" & temp_ambiente & "','" & data_dados & "')"
                    con.Operar(sql)
```

```
                End If
            Catch ex As Exception
                'MsgBox(ex.Message & " Erro no método Cadastrar")
            End Try




        Catch ex As Exception

        End Try

        If i < 101 Then
            datas(i) = Date.Now.Hour & ":" & Date.Now.Minute & " : " &
Date.Now.Second
            valores_carcaca(i) = CType(carcaca, Integer) / 100
            valores_interna(i) = CType(estator, Integer) / 100
            valores_ambiente(i) = CType(temp_ambiente, Integer)
        ElseIf i >= 100 Then
            For j As Integer = 1 To 100
                datas(j - 1) = datas(j)
                valores_carcaca(j - 1) = valores_carcaca(j)
                valores_interna(j - 1) = valores_interna(j)
                valores_ambiente(j - 1) = valores_ambiente(j)
            Next
            datas(100) = Date.Now.Hour & ":" & Date.Now.Minute & " : " &
Date.Now.Second
            valores_carcaca(100) = CType(carcaca, Integer) / 100
            valores_interna(100) = CType(estator, Integer) / 100
            valores_ambiente(100) = CType(temp_ambiente, Integer)
        End If
        Chart1.Series(0).Points.DataBindXY(datas, valores_carcaca)
        Chart1.Series(1).Points.DataBindXY(datas, valores_interna)
        Chart1.Series(2).Points.DataBindXY(datas, valores_ambiente)
        i = i + 1

    End Sub

    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        carcaca_limite = 100
        estator_limite = 105
        tensao_limite = 130
        corrente_limite = 3

        'imer1.Enabled = True
        Timer2.Enabled = True
        'Timer3.Enabled = True

    End Sub

    Private Sub Timer1_Tick(sender As Object, e As EventArgs) Handles Timer1.Tick
        '''' fazer leitura dos dados que foi realizado a leitura do banco de
dados
        '''
        'valores = Buscar_Dados_Temperatura_Estator()

        'Chart1.Series(0).Points.DataBindXY(ds)
    End Sub

    Public Function Buscar_Dados_Temperatura_Estator()
```

```
        sql = "select hora_leitura,temp_estator from dados_motor where
data_leitura='" & data & "'  and hora_leitura >'" & Horario & "'  and
hora_leitura<'" & Horario_fim & "'"
        ds = con.listar(sql)
        Return ds
    End Function


  End Class
```

ANEXO D – CÓDIGO DO DESIGN DA TELA DE PARÂMETROS DE FUNCIONAMENTO

```
<Global.Microsoft.VisualBasic.CompilerServices.DesignerGenerated()> _
Partial Class Parametros_de_Funcionamento
    Inherits System.Windows.Forms.Form

    'Descartar substituições de formulário para limpar a lista de componentes.
    <System.Diagnostics.DebuggerNonUserCode()> _
    Protected Overrides Sub Dispose(ByVal disposing As Boolean)
        Try
            If disposing AndAlso components IsNot Nothing Then
                components.Dispose()
            End If
        Finally
            MyBase.Dispose(disposing)
        End Try


      End Sub


'Exigido pelo Windows Form Designer
    Private components As System.ComponentModel.IContainer

    'OBSERVAÇÃO: o procedimento a seguir é exigido pelo Windows Form Designer
    'Pode ser modificado usando o Windows Form Designer.
    'Não o modifique usando o editor de códigos.
    <System.Diagnostics.DebuggerStepThrough()> _
    Private Sub InitializeComponent()
        Dim resources As System.ComponentModel.ComponentResourceManager = New
System.ComponentModel.ComponentResourceManager(GetType(Parametros_de_Funcionament
o))
        Me.Label1 = New System.Windows.Forms.Label()
        Me.Label2 = New System.Windows.Forms.Label()
        Me.Label8 = New System.Windows.Forms.Label()
        Me.Label10 = New System.Windows.Forms.Label()
        Me.Label12 = New System.Windows.Forms.Label()
        Me.CheckBox1 = New System.Windows.Forms.CheckBox()
        Me.CheckBox2 = New System.Windows.Forms.CheckBox()
        Me.CheckBox3 = New System.Windows.Forms.CheckBox()
        Me.CheckBox4 = New System.Windows.Forms.CheckBox()
        Me.CheckBox5 = New System.Windows.Forms.CheckBox()
        Me.TextBox1 = New System.Windows.Forms.TextBox()
        Me.TextBox2 = New System.Windows.Forms.TextBox()
        Me.TextBox3 = New System.Windows.Forms.TextBox()
        Me.TextBox4 = New System.Windows.Forms.TextBox()
        Me.TextBox5 = New System.Windows.Forms.TextBox()
        Me.TableLayoutPanel1 = New System.Windows.Forms.TableLayoutPanel()
        Me.Button1 = New System.Windows.Forms.Button()
        Me.Button2 = New System.Windows.Forms.Button()
        Me.TableLayoutPanel1.SuspendLayout()
        Me.SuspendLayout()
        '
        'Label1
        '
        Me.Label1.AutoSize = True
```

```
        Me.Label1.Location = New System.Drawing.Point(3, 0)
        Me.Label1.Name = "Label1"
        Me.Label1.Size = New System.Drawing.Size(62, 13)
        Me.Label1.TabIndex = 21
        Me.Label1.Text = "Tensão (V):"
        '
        'Label2
        '
        Me.Label2.AutoSize = True
        Me.Label2.Location = New System.Drawing.Point(3, 26)
        Me.Label2.Name = "Label2"
        Me.Label2.Size = New System.Drawing.Size(66, 13)
        Me.Label2.TabIndex = 22
        Me.Label2.Text = "Corrente (A):"
        '
        'Label8
        '
        Me.Label8.AutoSize = True
        Me.Label8.Location = New System.Drawing.Point(3, 52)
        Me.Label8.Name = "Label8"
        Me.Label8.Size = New System.Drawing.Size(141, 13)
        Me.Label8.TabIndex = 23
        Me.Label8.Text = "Temperatura do Estator (ºC):"
        '
        'Label10
        '
        Me.Label10.AutoSize = True
        Me.Label10.Location = New System.Drawing.Point(3, 78)
        Me.Label10.Name = "Label10"
        Me.Label10.Size = New System.Drawing.Size(148, 13)
        Me.Label10.TabIndex = 25
        Me.Label10.Text = "Temperatura da Carcaça (ºC):"
        '
        'Label12


          '


Me.Label12.AutoSize = True
        Me.Label12.Location = New System.Drawing.Point(3, 104)
        Me.Label12.Name = "Label12"
        Me.Label12.Size = New System.Drawing.Size(137, 13)
        Me.Label12.TabIndex = 24
        Me.Label12.Text = "Temperatura Ambiente (ºC):"
        '
        'CheckBox1
        '
        Me.CheckBox1.AutoSize = True
        Me.CheckBox1.Location = New System.Drawing.Point(294, 3)
        Me.CheckBox1.Name = "CheckBox1"
        Me.CheckBox1.Size = New System.Drawing.Size(58, 17)
        Me.CheckBox1.TabIndex = 26
        Me.CheckBox1.Text = "Alarme"
        Me.CheckBox1.UseVisualStyleBackColor = True
        '
        'CheckBox2
        '
        Me.CheckBox2.AutoSize = True
        Me.CheckBox2.Location = New System.Drawing.Point(294, 29)
        Me.CheckBox2.Name = "CheckBox2"
```

```
Me.CheckBox2.Size = New System.Drawing.Size(58, 17)
Me.CheckBox2.TabIndex = 27
Me.CheckBox2.Text = "Alarme"
Me.CheckBox2.UseVisualStyleBackColor = True
'
'CheckBox3
'
Me.CheckBox3.AutoSize = True
Me.CheckBox3.Location = New System.Drawing.Point(294, 55)
Me.CheckBox3.Name = "CheckBox3"
Me.CheckBox3.Size = New System.Drawing.Size(58, 17)
Me.CheckBox3.TabIndex = 28
Me.CheckBox3.Text = "Alarme"
Me.CheckBox3.UseVisualStyleBackColor = True
'
'CheckBox4
'
Me.CheckBox4.AutoSize = True
Me.CheckBox4.Location = New System.Drawing.Point(294, 81)
Me.CheckBox4.Name = "CheckBox4"
Me.CheckBox4.Size = New System.Drawing.Size(58, 17)
Me.CheckBox4.TabIndex = 29
Me.CheckBox4.Text = "Alarme"
Me.CheckBox4.UseVisualStyleBackColor = True
'
'CheckBox5
'
Me.CheckBox5.AutoSize = True
Me.CheckBox5.Location = New System.Drawing.Point(294, 107)
Me.CheckBox5.Name = "CheckBox5"
Me.CheckBox5.Size = New System.Drawing.Size(58, 17)
Me.CheckBox5.TabIndex = 30
Me.CheckBox5.Text = "Alarme"
Me.CheckBox5.UseVisualStyleBackColor = True
'
'TextBox1
'
Me.TextBox1.Location = New System.Drawing.Point(158, 3)
Me.TextBox1.Name = "TextBox1"
Me.TextBox1.Size = New System.Drawing.Size(100, 20)
Me.TextBox1.TabIndex = 31
'
'TextBox2
'
Me.TextBox2.Location = New System.Drawing.Point(158, 29)
Me.TextBox2.Name = "TextBox2"
Me.TextBox2.Size = New System.Drawing.Size(100, 20)
Me.TextBox2.TabIndex = 32
'
'TextBox3
'
Me.TextBox3.Location = New System.Drawing.Point(158, 55)
Me.TextBox3.Name = "TextBox3"
Me.TextBox3.Size = New System.Drawing.Size(100, 20)
Me.TextBox3.TabIndex = 33
'
'TextBox4
'
Me.TextBox4.Location = New System.Drawing.Point(158, 81)
Me.TextBox4.Name = "TextBox4"
Me.TextBox4.Size = New System.Drawing.Size(100, 20)
Me.TextBox4.TabIndex = 34
```

```
        '
        'TextBox5
        '
        Me.TextBox5.Location = New System.Drawing.Point(158, 107)
        Me.TextBox5.Name = "TextBox5"
        Me.TextBox5.Size = New System.Drawing.Size(100, 20)
        Me.TextBox5.TabIndex = 35
        '
        'TableLayoutPanel1
          '
Me.TableLayoutPanel1.ColumnCount = 3
        Me.TableLayoutPanel1.ColumnStyles.Add(New
System.Windows.Forms.ColumnStyle(System.Windows.Forms.SizeType.Percent,
39.14141!))
        Me.TableLayoutPanel1.ColumnStyles.Add(New
System.Windows.Forms.ColumnStyle(System.Windows.Forms.SizeType.Percent,
34.59596!))
        Me.TableLayoutPanel1.ColumnStyles.Add(New
System.Windows.Forms.ColumnStyle(System.Windows.Forms.SizeType.Percent,
26.26263!))
        Me.TableLayoutPanel1.Controls.Add(Me.Label12, 0, 4)
        Me.TableLayoutPanel1.Controls.Add(Me.CheckBox1, 2, 0)
        Me.TableLayoutPanel1.Controls.Add(Me.CheckBox2, 2, 1)
        Me.TableLayoutPanel1.Controls.Add(Me.CheckBox3, 2, 2)
        Me.TableLayoutPanel1.Controls.Add(Me.CheckBox4, 2, 3)
        Me.TableLayoutPanel1.Controls.Add(Me.CheckBox5, 2, 4)
        Me.TableLayoutPanel1.Controls.Add(Me.TextBox1, 1, 0)
        Me.TableLayoutPanel1.Controls.Add(Me.TextBox2, 1, 1)
        Me.TableLayoutPanel1.Controls.Add(Me.TextBox3, 1, 2)
        Me.TableLayoutPanel1.Controls.Add(Me.TextBox4, 1, 3)
        Me.TableLayoutPanel1.Controls.Add(Me.TextBox5, 1, 4)
        Me.TableLayoutPanel1.Controls.Add(Me.Label10, 0, 3)
        Me.TableLayoutPanel1.Controls.Add(Me.Label8, 0, 2)
        Me.TableLayoutPanel1.Controls.Add(Me.Label2, 0, 1)
        Me.TableLayoutPanel1.Controls.Add(Me.Label1, 0, 0)
        Me.TableLayoutPanel1.Location = New System.Drawing.Point(12, 12)
        Me.TableLayoutPanel1.Name = "TableLayoutPanel1"
        Me.TableLayoutPanel1.RowCount = 5
        Me.TableLayoutPanel1.RowStyles.Add(New
System.Windows.Forms.RowStyle(System.Windows.Forms.SizeType.Percent, 20.0!))
        Me.TableLayoutPanel1.RowStyles.Add(New
System.Windows.Forms.RowStyle(System.Windows.Forms.SizeType.Percent, 20.0!))
        Me.TableLayoutPanel1.RowStyles.Add(New
System.Windows.Forms.RowStyle(System.Windows.Forms.SizeType.Percent, 20.0!))
        Me.TableLayoutPanel1.RowStyles.Add(New
System.Windows.Forms.RowStyle(System.Windows.Forms.SizeType.Percent, 20.0!))
        Me.TableLayoutPanel1.RowStyles.Add(New
System.Windows.Forms.RowStyle(System.Windows.Forms.SizeType.Percent, 20.0!))
        Me.TableLayoutPanel1.Size = New System.Drawing.Size(396, 131)
        Me.TableLayoutPanel1.TabIndex = 36
        '
        'Button1
        '
        Me.Button1.Location = New System.Drawing.Point(106, 160)
        Me.Button1.Name = "Button1"
        Me.Button1.Size = New System.Drawing.Size(75, 23)
        Me.Button1.TabIndex = 37
```

```
        Me.Button1.Text = "Salvar"
        Me.Button1.UseVisualStyleBackColor = True
        '
        'Button2
        '
        Me.Button2.Location = New System.Drawing.Point(205, 160)
        Me.Button2.Name = "Button2"
        Me.Button2.Size = New System.Drawing.Size(75, 23)
        Me.Button2.TabIndex = 38
        Me.Button2.Text = "Limpar"
        Me.Button2.UseVisualStyleBackColor = True
        '
        'Parametros_de_Funcionamento
        '
        Me.AutoScaleDimensions = New System.Drawing.SizeF(6.0!, 13.0!)
        Me.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font
        Me.ClientSize = New System.Drawing.Size(418, 196)
        Me.Controls.Add(Me.Button2)
        Me.Controls.Add(Me.Button1)
        Me.Controls.Add(Me.TableLayoutPanel1)
        Me.Icon = CType(resources.GetObject("$this.Icon"), System.Drawing.Icon)
        Me.Name = "Parametros_de_Funcionamento"
        Me.StartPosition = System.Windows.Forms.FormStartPosition.CenterScreen
        Me.Text = "Parametros_de_Funcionamento"
        Me.TableLayoutPanel1.ResumeLayout(False)
        Me.TableLayoutPanel1.PerformLayout()
        Me.ResumeLayout(False)

    End Sub

    Friend WithEvents Label1 As Label
    Friend WithEvents Label2 As Label
    Friend WithEvents Label8 As Label
    Friend WithEvents Label10 As Label
    Friend WithEvents Label12 As Label
    Friend WithEvents CheckBox1 As CheckBox
    Friend WithEvents CheckBox2 As CheckBox
    Friend WithEvents CheckBox3 As CheckBox
    Friend WithEvents CheckBox4 As CheckBox
    Friend WithEvents CheckBox5 As CheckBox
    Friend WithEvents TextBox1 As TextBox
    Friend WithEvents TextBox2 As TextBox
    Friend WithEvents TextBox3 As TextBox
    Friend WithEvents TextBox4 As TextBox
    Friend WithEvents TextBox5 As TextBox
    Friend WithEvents TableLayoutPanel1 As TableLayoutPanel
    Friend WithEvents Button1 As Button
    Friend WithEvents Button2 As Button


End Class
```

ANEXO E – CÓDIGO DO FORMULÁRIO DA TELA DE PARÂMETROS DE FUNCIONAMENTO

```
Public con As New MySqlConnection
    Public cmd As MySqlCommand = con.CreateCommand
    Public da As MySqlDataAdapter = New MySqlDataAdapter(cmd)
    Public ds As New DataSet


    Dim fs As FileStream
    Dim br As BinaryReader


    Public Sub conectar()
        Try
            con.ConnectionString = "server=localhost;user id=root;
password=;database=tcc"
            con.Open()
            'MsgBox("Conexão realizada com sucesso!")
        Catch ex As Exception
            MsgBox(ex.Message & "Erro no Método conectar!!")
        End Try
    End Sub


    Public Sub Operar(ByVal sql)

        conectar()
        Try
            cmd.CommandText = sql
            cmd.ExecuteNonQuery()
            'MsgBox("Operação realizada com sucesso!", MsgBoxStyle.Information,
"Informação")
        Catch ex As Exception
            MsgBox(ex.Message & ex.ToString, MsgBoxStyle.Critical, "Erro no
método operar")
        Finally
            con.Close()

        End Try


      End Sub
```

ANEXO F – CÓDIGO DO BANCO DE DADOS

```
DROP DATABASE IF EXISTS tcc;
CREATE DATABASE tcc ;
USE tcc;

DROP TABLE IF EXISTS dados;
CREATE TABLE dados(
cod_Dados int primary key AUTO_INCREMENT,
carcaca varchar(10),
estator varchar(10),
tensao varchar(10),
corrente varchar(10),
rpm varchar(10),
umidade varchar(10),
temp_ambiente varchar(10),
data_dados varchar(20));
```

ANEXO G – CÓDIGO DO ARDUINO MEGA

```
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>

RF24 radio(7, 8);
const byte address[6] = "00002";
int text[8];
void setup() {
  Serial.begin(9600);
  radio.begin();
  radio.openReadingPipe(0, address);
  radio.setPALevel(RF24_PA_MAX);
  radio.startListening();
}
void imprimir(){

     Serial.print(text[1]);
      Serial.print(",");
     Serial.print(text[2]);
     Serial.print(",");
     Serial.print(text[3]);
     Serial.print(",");
    Serial.print(text[4]);
    Serial.print(",");
    Serial.print(text[5]);
    Serial.print(",");
    Serial.print(text[6]);
    Serial.print(",");
    Serial.println(text[7]);

  }

void loop() {
  if (radio.available()) {
    //char text[32] = "";
    //String text="";
    text[1]="";
    radio.read(text, sizeof(text));
   //imprimir();
  }
  if (Serial.read()=='E'){
   imprimir();
```

```
    }
    //delay(1000);
}
```

ANEXO H – CÓDIGO DO ARDUINO NANO

```
#include "EmonLib.h"
#define VOLT_CAL 211.6
#include <OneWire.h>
#include <DallasTemperature.h>
#define DS18B20_OneWire 8
#include <DS1307.h>
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>
#include <DHT.h>
#define DHTPIN 3
#define DHTTYPE DHT22

DHT dht(DHTPIN, DHTTYPE);

RF24 radio(7,10);

const byte address[6] = "00002";

DS1307 rtc(A4, A5);

OneWire oneWire(DS18B20_OneWire);
DallasTemperature sensortemp(&oneWire);

int ndispositivos = 0;
float grausC;
int dados[8];
String hora;
String data;
EnergyMonitor emon1;
float vetCorrente[100];

float revolutions=0;
int rpm=0;
long  startTime=0;
long  elapsedTime;


void setup()
{

sensortemp.begin();
radio.begin();
rtc.halt(false);
rtc.setSQWRate(SQW_RATE_1);
rtc.enableSQW(true);

Serial.begin(9600);
dht.begin(); //INICIALIZA A FUNÇÃO
  delay(2000); //INTERVALO DE 2 SEGUNDO ANTES DE INICIAR
pinMode(A0, INPUT);
```

```
pinMode(2, INPUT_PULLUP);
//LEITURA DE TENSÃO
emon1.voltage(1, VOLT_CAL, 1.7);



radio.openWritingPipe(address);
radio.setPALevel(RF24_PA_MIN);
radio.stopListening();

}


int tensao(){
  emon1.calcVI(17,2000);

  float supplyVoltage   = emon1.Vrms;

  Serial.print("Tensão medida na rede AC: ");
  Serial.print(supplyVoltage, 0);
  Serial.println("V");
  return supplyVoltage;
  }

int corrente(){
  double media=0;
  double maior_Valor = 0;
  float valor_Corrente = 0;
  for(int i = 0; i < 100; i++)
  {
    vetCorrente[i] = analogRead(A0);
    delayMicroseconds(100);
  }
  for(int i = 0; i < 100; i++)
  {
    if(maior_Valor < vetCorrente[i])
    {
      maior_Valor = vetCorrente[i];
    }
  }
  maior_Valor = maior_Valor * 0.004882812;
  valor_Corrente = maior_Valor - 2.5;
  if (valor_Corrente<0){
    valor_Corrente=0;
    }
  valor_Corrente = valor_Corrente * 1000;
  valor_Corrente = valor_Corrente / 100;
  valor_Corrente = valor_Corrente / 1.41421356;
  Serial.print("Corrente = ");
  Serial.print(valor_Corrente,2);
  Serial.println(" A");
  return valor_Corrente*100;
  }

  int temperatura_carcaca(){
    sensortemp.requestTemperatures();
    Serial.print("SENSOR DA CARCAÇA: ");
    grausC = sensortemp.getTempCByIndex(0);
    Serial.print(grausC);
```

```
    Serial.println("ºC");
    return grausC*100;
  }
   int temperatura_estator(){
    Serial.print("SENSOR DO ESTATOR: ");
    grausC = sensortemp.getTempCByIndex(1);
    Serial.print(grausC);
    Serial.println("ºC");
  Serial.println("");
  return grausC*100;
    }

    void hora_rtc(){
      Serial.print("Hora : ");
  Serial.print(rtc.getTimeStr());
hora=rtc.getTimeStr();
  Serial.print(" ");
  Serial.print("Data : ");
  Serial.print(rtc.getDateStr(FORMAT_SHORT));
  data=(rtc.getDateStr(FORMAT_SHORT));
  Serial.print(" ");
  Serial.println(rtc.getDOWStr(FORMAT_SHORT));
      }

  void loop()
{

revolutions=0; rpm=0;
startTime=millis();
attachInterrupt(digitalPinToInterrupt(2),interruptFunction,RISING);
delay(5000);
detachInterrupt(2);

elapsedTime=millis()-startTime;

if(revolutions>0)
{
  rpm=(max(1, revolutions) * 60000) / elapsedTime;
}



  int carcaca=0;
  int estator=0;
  int tensao1=0;
  int corrente1=0;

  carcaca=temperatura_carcaca();
  estator=temperatura_estator();
  tensao1=tensao();
  corrente1=corrente();

dados[1]=carcaca;
dados[2]=estator;
dados[3]=tensao1;
dados[4]=corrente1;
dados[5]=rpm;
dados[6]=dht.readHumidity()*100;
dados[7]=dht.readTemperature();
```

```
radio.write(dados, sizeof(dados));
}

void interruptFunction() //interrupt service routine
{
  revolutions++;
}
```