SCADA de baixo custo para motores elétricos utilizando o ESP32 e o Scadabr.

Matheus Macena Bastos
Universidade Cândido Mendes
Seropédica, Brasil
mt.macena@gmail.com

***Abstract*— This work presents the efficiency test of a Supervisory Control and Data Acquisition prototype, applied to an electric motor, using free software and low cost hardware components. A bibliographical research was carried out to raise ideas from which, combined with the author's expertise, it was possible to design a functional system that generated satisfactory results.**

***Resumo* — Este trabalho apresenta o teste de eficiência de um protótipo de Sistema de Supervisão e Aquisição de Dados, aplicado a um motor elétrico, utilizando softwares gratuitos e componentes de hardware de baixo custo. Foi realizada uma pesquisa bibliográfica para levantar ideias a partir das quais, aliadas a expertise do autor, foi possível conceber um sistema funcional e que gerou resultados satisfatórios.**

***Palavras-chave*—SCADA; ESP32; Automação; Manutenção.**

I. INTRODUÇÃO

Com o setor produtivo industrial cada vez mais competitivo, as empresas buscam o implemento de variados tipos de tecnologia para o aumento de sua lucratividade através da redução de custos. O setor de manutenção tem grande importância para essa diminuição de gastos e também para a saúde econômica da planta industrial como um todo, garantindo a continuidade dos processos e a disponibilidade das máquinas. (BASTOS, 2019)

De acordo com Pessatti (2020), inúmeros desses processos de fabricação dependem diretamente de máquinas elétricas girantes para seu perfeito funcionamento. Dentre as mais diversas máquinas de uma fábrica, podemos destacar os motores elétricos, sendo estes os responsáveis por transformar energia elétrica em energia mecânica, fornecendo assim os movimentos necessários para desenvolvimento desses processos.

Para Ferreira (2019), a manutenção e o acompanhamento dos motores elétricos são de extrema importância, podendo obter uma relevante redução de desperdícios e uma melhoria na produtividade. A garantia de um bom desempenho dos motores elétricos vem através de um monitoramento e análise de vários aspectos operacionais, feitos por um operador ou técnico, a fim de determinar a necessidade de efetuar a parada do motor, por exemplo, evitando assim os gastos com paradas não planejadas. (MUYNARSK, 2014)

Utilizar tecnologias para monitoramento remoto é uma ação fundamental para garantir a performance das máquinas industriais. Para a *Edge Global Suply* (2020), sem essas tecnologias é impossível a adequação das empresas à era da indústria 4.0. Além disso, existem soluções de *IoT*, ou Internet das Coisas, que tornam esse monitoramento mais barato e assecível, podem ser aplicado desde os processos industriais até problemas do cotidiano das pessoas. (EPICHIN e BASSANI, 2019).

Apoiado nisso, esse trabalho irá verificar a eficiência de um *SCADA*, utilizando uma placa de desenvolvimento de baixo custo e *softwares* gratuitos, visando a aplicação no monitoramento e controle de motores industriais.

II. FUNDAMENTAÇÃO TEÓRICA

A. SCADA

Um SCADA (*Supervisory Control and Data Acquisition*), ou simplesmente Software Supervisório, é uma aplicação que faz a comunicação entre computador e dispositivos inteligentes, sendo esses geralmente *CLP's* (Controladores Lógicos Programáveis), a fim monitorar e controlar os mais variados tipos de processo. (DANEELS, 1990). Os SCADA rodam a partir de um Sistema Operacional, como *Windows*, *Linux*, etc. Dentre as ferramentas de desenvolvimento de SCADA mais famosas podemos citar: *Elipse E3*, *FactoryTalk*, *WinCC*, *EcoStruxure*, dentre outros.

A implementação de um SCADA nas empresas requer um custo elevado tanto para instalação como para a manutenção da licença de software. Uma alternativa gratuita é o Scadabr, que é um software livre de código aberto. Atualmente, é atualizado por qualquer usuário que pertença a comunidade do Scadabr, que não tem fins lucrativos. Os principais protocolos de comunicação suportados são: *Modbus*, *ASCII*, *Bacnet*, entre outros. (Marlon Ramos Silva, 2013). O Scadabr permite a criação de aplicativos personalizados, em qualquer linguagem de programação atual (*JAVA*, *C++*, *VB*, *PHP*, *JavaScript*, *MS Excel* entre outras), a partir do código-fonte disponibilizado ou de sua API "*web-services*". (ROCHA, 2011). Com o Scadabr é possível desenvolver softwares supervisório gratuitamente, a partir de um desktop com *Windows* por exemplo.

B. Placas de Desenvolvimento

Placas de desenvolvimento consistem em um circuito impresso, um microcontrolador, uma conexão *USB* para programação e diversas *GPIO's* (Entradas e saídas). (SAMPAIO 2014). Os microcontroladores são dispositivos

inteligentes (MIYADAIRA, 2009). Podemos dizer que são o "cérebro" de uma placa, portanto esses dispositivos podem se integrar a um SCADA através do seu protocolo de comunicação. Esses dispositivos são produzidos por empresas como *Atmel Corporation, Espressif Systems, RS Components, Intel*, etc.. Dentre as mais variadas placas de desenvolvimento presentes no mercado, podemos destacar as linhas mais populares: *Arduino, Raspeberry Pi e NodeMCU*.

Foi feita uma pesquisa através do google acadêmico para levantar a quantidade de trabalhos, dentre artigos e livros, envolvendo essas plataformas:

TABELA I
QUANTIDADE DE ARTIGOS. GOOGLE ACADÊMICO. 15/03/2021

Arduino	700.000
Raspberry PI	529.000
NodeMCU	4.880

Nota-se que o Arduino e o Raspberry PI têm o uso muito difundido no meio acadêmico, enquanto que o NodeMCU não tem grande expressão. Contudo, uma placa NodeMCU equipada com um microcontrolador *ESP32* da *Espressif Systems*, apresenta um hardware altamente superior as principais placas Arduino e Raspberry. A tabela a seguir mostra um comparativo entre algumas plataformas de mesma faixa de preço (em torno de R$ 60,00) o NodeMCU ESP32, o Arduino Uno R3 e o recém lançado Raspberry Pi Pico:

TABELA II
COMPARATIVO DE HARDWARE ENTRE PLACAS DE DESENVOLVIMENTO.

PLACA	CLOCK	RAM	FLASH	GPIO
Arduino Uno R3	16 MHz	2Kb	32Kb	14
Raspberry Pi Pico	133 MHz	256Kb	2 Mb	**40**
ESP32	**160 MHz**	**512Kb**	**16Mb**	38

Como é possível visualizar na tabela, o ESP32 é superior as demais placas na maioria dos aspectos de hardware, tendo um preço de compra praticamente igual. Por não ser muito difundido no meio acadêmico, como o Arduino UNO R3 ou o Raspberry Pi Pico, e ser significantemente mais potente do que os outros, o ESP32 foi escolhido para o desenvolvimento desse protótipo.

C. *Motores Elétricos*

Partindo do princípio de que a energia não é criada nem destruída: ela simplesmente muda de forma (UMANS**,** 2014), um motor elétrico é uma máquina que tem a propriedade de transformar energia elétrica em energia mecânica. Sua aplicação se dá mais comumente em acionamentos de cargas mecânicas, como ventiladores, compressores, guinchos, esteiras, etc, estando presente principalmente no ambiente fabril. (GUEDES, 1994),

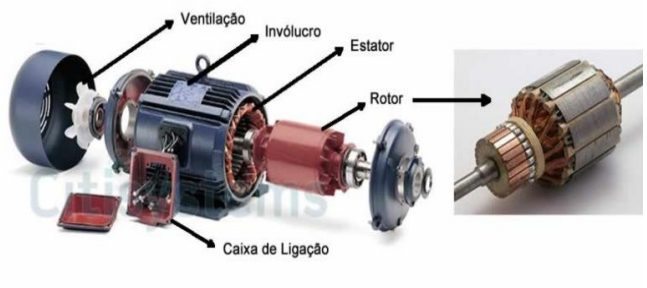


Fig. 1. Partes de um motor elétrico de corrente alternada.

Na construção dos motores, são aplicadas várias técnicas e materiais apropriados para garantir sua performance e diminuição de perdas de energia (JOÃO GABRIEL, 2009). Além disso, um motor elétrico possui diversos parâmetros a serem definidos no momento da sua especificação (SILVEIRA, 2016), como:

- Temperatura de operação;
- Classificação de isolamento;
- Classificação de vibração;
- Proteção de poeira;
- Proteção de gotejamento;

Quando o motor opera dentro das faixas especificadas em cada um desses parâmetros, há a garantia de sua alta performance, portanto, tornam-se indicadores importantes para o monitoramento do equipamento. Esse controle geralmente é feito por instrumentos industriais de vários tipos.

III. MATERIAIS E MÉTODOS

D. *Revisão Teórica*

Para o início do desenvolvimento da proposta, foram realizadas pesquisas bibliográficas e bibliométricas visando levantar ideias e avaliar a relevância do tema proposto. Também foi pesquisado em torno do preço de compra dos

materiais e softwares, a fim de conceber um sistema de baixo custo.

E. Hardwares

O principal componente de hardware escolhido foi o ESP32 modelo Wroom-32, por conter WIFI de fábrica, bom poder de processamento e grande quantidade de GPIO's.

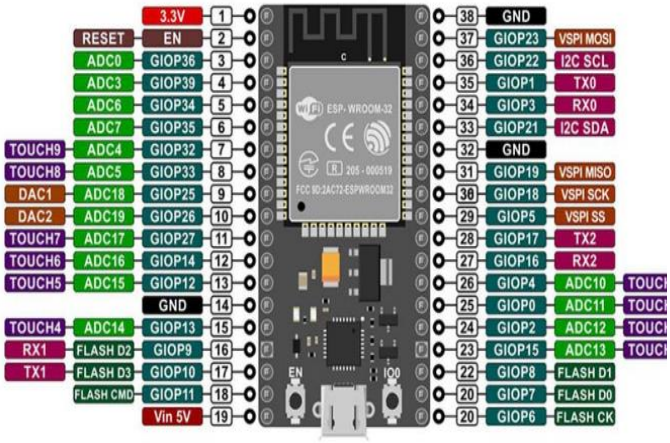
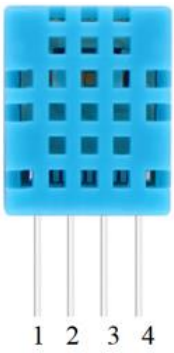


Fig. 2. Pinagem do ESP32

A parte de sensores foram escolhidos o sensor de temperatura e umidade DHT11, o sensor acelerômetro GY521 e o sensor de corrente elétrica SCT0-13-30a. Para fazer as conexões foi utilizado duas protoboards e fios jump. A alimentação do circuito se dará através de uma bateria de 5V e 1A. O comando do motor se dá através de um relé. O ESP32 alimenta a protoboard com 3.3V e consequentemente os componentes ligados a ela. A ligação de cada componente com o ESP32 é apresentada a seguir:



1	3.3v ESP32
2	GPIO 27
3	-
4	GND ESP32

Fig. 3. Ligações do sensor DHT11

Vcc	3.3v ESP32
GND	GND ESP32
SCL	GPIO 22
SDA	GPIO 21

Fig. 4. Ligações do sensor GY521

Vcc	3.3v ESP32
GND	GND ESP32
IN	GPIO 32
NC	Fase
NO	A1 Contator

Fig. 5. Ligações do relé

O sensor de corrente SCT0-13-30 em especial, precisa de componentes adicionais antes de ser ligado a energia. Um capacitor de 10µF e dois resistores exatamente iguais, no caso foram usados de 10kΩ.

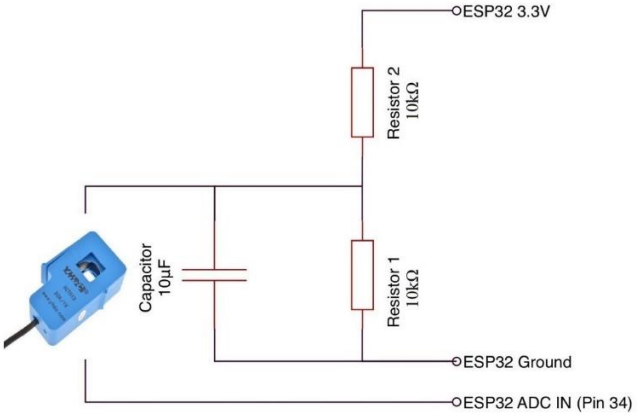


Fig. 6. Ligações do sensor SCT013-30. (DECUYPER, 2019)

O diagrama de potência consiste em um motor elétrico, um contator e um disjuntor. O ESP32 envia sinal para o relé

ativando a bobina do contator, que por sua vez faz o motor ligar. A ligação pode ser visualizada na imagem abaixo:

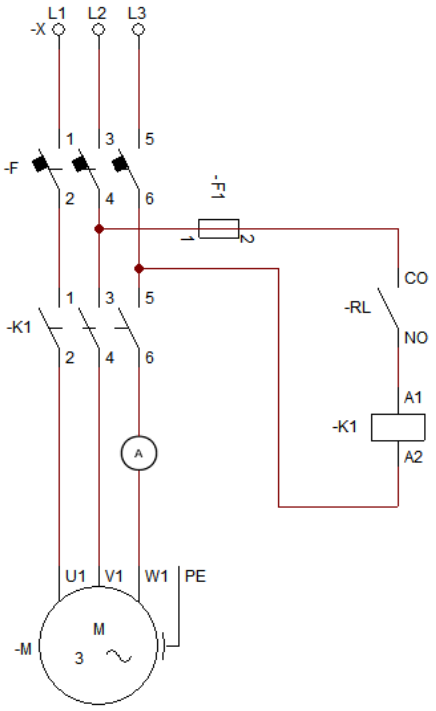


Fig. 7. Diagrama de comando e potência

F. Software

A programação do ESP32 foi realizada através da Arduino IDE, que usa a linguagem de programação c/c++. A principal biblioteca utilizada foi a "modbus.h", permitindo que o ESP se comunique através do protocolo Modbus. O Modbus é um protocolo do tipo mestre-escravo, utilizado em automação industrial, podendo ser utilizado em outras áreas, como por exemplo, na automação residencial. (BARBOSA, 2015). O código fonte completo é muito extenso e não poderá ser apresentado nesse artigo, porém é possível acessa-lo gratuitamente através do link: https://github.com/MatheusMacena/ESP32_Scadabr_Motor

O ponto chave do código fonte foi a criação dos registradores do protocolo Modbus para posterior comunicação com o Scadabr. A tabela a seguir apresenta cada um deles:

TABELA II

LISTA DE REGISTRADORES MODBUS

Registrador	Tipo	Função	Valores
100	Status do Coil	Comando o motor.	0 = Liga 1 = Desliga
101	Registrador de Entrada	Lê a temperatura do DHT11	De 0 à 50
102	Registrador de Entrada	Lê a umidade do DHT11	De 0 à 100
103	Registrador Holding	Armazena o status do motor	0 = desligado 1 = ligado 2 = Falha no commando 3 = Alta temperature 4 = Alta vibração 5 = Sobrecorrente 6 = Alta umidade
104	Registrador Holding	Reset de falhas	0 = Reset 1 = Resetando
105	Registrador de Entrada	Lê a corrente elétrica do SCT013	De 0 à 10
106	Registrador de Entrada	Lê a vibração do GY521	De 0 à 500

O código fonte segue a seguinte lógica:

- O motor inicia desligado, com o registrador 103 (Status do Motor) assumindo o valor "1" (desligado). Paralelo a isso ocorre a aquisição de dados de status, temperatura, umidade, vibração e corrente elétrica em seus respectivos registradores
- Quando o registrador 100 (Comando do motor) tem seu estado alterado para "0" (ligar) e não houver nenhuma falha, o motor entrará em funcionamento.
- Caso seja enviado o comando e o sensor de corrente marque um valor ">0", o registrador 103 receberá o valor "1" (ligado).

A tabela abaixo mostra a lógica de geração de falhas:

TABELA 4

LÓGICA DE GERAÇÃO DE FALHAS

SE	FALHA
Motor ligar e o valor da corrente é igual a zero	Falha no comando

Motor está ligado e a temperatura é maior ou igual a 40°C	Alta temperatura
Motor está ligado e a umidade é maior ou igual a 95%	Alta umidade
Motor está ligado e a vibração é maior ou igual a 2000 pontos	Alta vibração
Motor está ligado e a corrente elétrica é maior ou igual a 10A	Sobrecorrente

Sempre que o motor entra em qualquer estado de falha, o controle imediatamente envia um comando de parada e mostra o nome do defeito na tela.

G. SCADABR

Para o SCADA, foi escolhido o Scadabr. As principais configurações do Scadabr são o *data source*, onde se deve configurar como o software se comunica com o hardware, os *data points*, que são basicamente os registradores e por fim a tela do supervisório em si.

O data source foi configurado com o endereço IP do ESP32 e foram acrescentados data points de acordo com os registradores apresentados na tabela criados do código fonte.

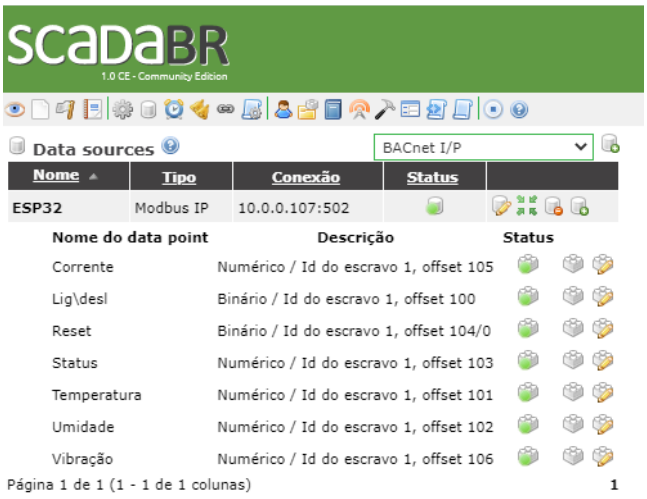


Fig. 8. Data source e data points do Scadabr

Com os data points criados, iniciou-se a criação da tela do SCADA, inserindo as imagens criadas e atribuindo cada um dos data points a elas. Como sua biblioteca de imagens é limitada, foi feito o uso de editores como o *GIMP* e o *Inkscape*, para criar novos conteúdos.

Fig. 9. Tela principal do SCADA com o motor em operação

IV. CONTROLE E AQUISIÇÃO DE DADOS

Feita a comunicação entre Scadabr e ESP32, via wi-fi, iniciaram os testes de funcionamento. Cada comando foi registrado via print screen, assim como todas as falhas.

- O comando liga e desliga foi realizado diretamente via Scadabr
- As falhas de temperatura e umidade foram forçadas utilizando um soprador térmico
- Alta vibração foi gerada forçando o GY521
- Sobrecorrente alterando o setpoint de falha
- Falha no comando retirando o sensor SCT13-30 do circuito

A lista completa das telas é muito extensa, mas pode ser visualizada no endereço: https://github.com/MatheusMacena/ESP32_Scadabr_Motor

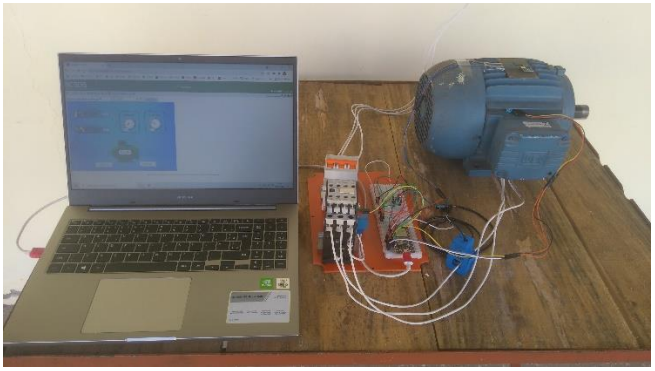


Fig. 10. Bancada de testes.

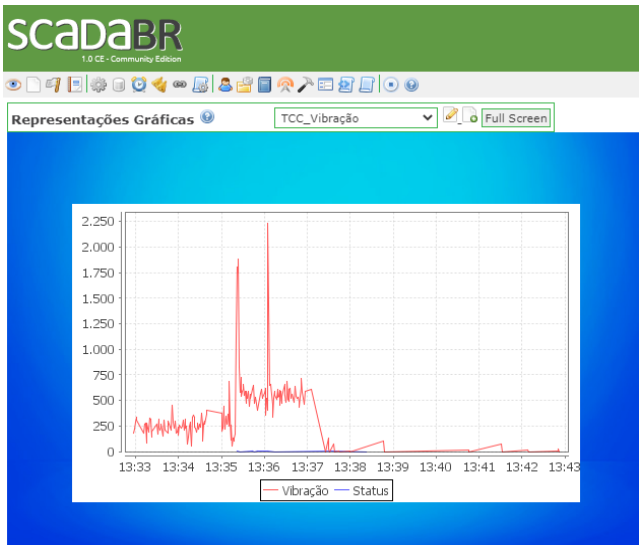


Fig. 11. Gráfico de Vibração.

Fig. 12. Gráficos de temperatura e umidade.

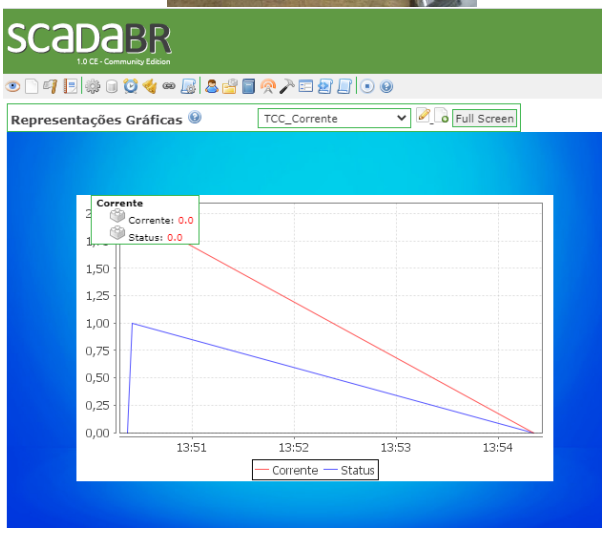


Fig. 13. Comparação de leitura de corrente elétrica.

V. CONCLUSÃO

Diante dos resultados apresentados, pode-se afirmar que o protótipo alcançou seu objetivo: ser um sistema de controle e aquisição de dados de baixo custo e confiável. O ESP32 garantiu uma proteção a mais para o equipamento, tendo em vista que ele automaticamente monitora diversos parâmetros de funcionamento e efetua paradas por segurança. As telas com gráficos permitem ao operador se antecipar nas ações de manutenção, prevenindo paradas inesperadas e reduzindo os custos para a sua organização.

Contudo existem diversos pontos de melhoria que podem ser implementados a esse projeto como:

- Acrescentar um sensor de corrente elétrica para cada uma das fases;
- Acrescentar um sensor de temperatura e umidade ambientes, fazendo a comparação do meio externo com a caixa de ligação do motor;
- Acrescentar a possibilidade do operador alterar os setpoints de geração de falha, permitindo que o sistema se adeque aos mais variados ambientes;
- Integrar o SCADA com um software de banco de dados como o MySQL.

VI. REFERÊNCIAS

[1] Barbosa, André Sarmento. Biblioteca Modbus para ESP8266. GitHub, 2015. Disponível em< https://github.com/andresarmento/modbus-esp8266/blob/master/README_pt_BR.md>. Acesso em: 9 abril. 2021

[2] Bastos, Matheus Macena. Desenvolvimento de um sistema de informação para apoiar a manutenção produtiva total. *In: XV Congresso Nacional de Excelência em Gestão*. Rio de Janeiro, RJ. 2019.

[3] Daneels, Axel; Salter, Wayne. What is SCADA? *International Conference on Accelerator and Large Experimental Physics Control Systems*. Trieste, Itália. 1999.

[4] Decuyper, Xavier; DIY Home Energy Monitor: ESP32 + CT Sensors + Emonlib, 2019. Disponível em: <https://savjee.be/2019/07/Home-Energy-Monitor-ESP32-CT-Sensor-Emonlib/>. Acesso em: 18 mai. 2021.

[5] Edge Global Supply. Análise de vibração e temperatura: saiba como e por que implantar, 2020. *Edge Global Supply*. Disponível em: <https://blog.edgeglobalsupply.com.br/analise-de-vibracao/>. Acesso em: 25 jan. 2021.

[6] Epichin, Gabriel; Bassani, Lorena. Projeto de Interfaces e Periféricos. *Universidade Federal do Espírito Santo*, 2019. Disponível em: <https://erus.ufes.br/wp-content/uploads/2020/05/Atividade_de_I_P.pdf>. Acesso em: 15 jun. 2021.

[7] Ferreira, Carlos André. Sistema de monitoramento em motores de indução trifásico utilizando a plataforma Arduino. 45 f. *Projeto de pesquisa - Curso superior de Tecnologia em manutenção industrial*. Universidade Tecnológica Federal do Paraná, Guarapuava, 2019.

[8] Guedes, Manuel Vaz. Motor de indução trifásico: seleção e aplicação. *Faculdade de Engenharia da Universidade do Porto, 1994*. Disponível em: <http://www.estgv.ipv.pt/paginaspessoais/vasco/textos/MI_sel&aplic.pdf>. Acesso em: 20 fev. 2021.

[9] Miyadaira, Alberto Noboru. Microcontroladores pic18: aprenda e programe em linguagem c. *Saraiva educação SA*, 2009.

[10] Muynarsk, Oscar Gomes; Garcia, Marcus V. Rocha. Sistema de monitoramento e controle de máquinas elétricas, utilizando microcontrolador Arduino e supervisório Elipse SCADA para diminuição de parada não programadas para a manutenção, *Anais do ENIAC, 2014*. Disponível em: <https://ojs.eniac.com.br/index.php/anais/article/view/191>. Acesso em: 25 jan. 2021.

[11] Oliveira, João Gabriel Souza Martins De. Materiais usados na construção de motores elétricos. *Pontifícia universidade católica do rio grande do Sul, 2009*. Disponível em: <http://www.motoreletrico.net/upload/materiais_motores.pdf>. Acesso em: 11 mar. 2021.

[12] Pessatti, Otávio Bohn Et Al. Um panorama sobre o monitoramento de condições de operação de motores elétricos, 2020. Disponível em: <https://repositorio.ufsc.br/handle/123456789/209320>. Acesso em: 20 fev. 2021.

[13] Rocha, Victor. Automação e Sensoreamento Remoto utilizando Software Livre "SCADA". Viva o Linux, 2011. Disponível em: <https://www.vivaolinux.com.br/artigo/Automacao-e-Sensoreamento-Remoto-utilizando-Software-Livre-SCADA> Acesso em: 14 mar. 2021.

[14] Sampaio, Claudio. ARM para hobbyistas – Parte 1: Placas de desenvolvimento. Disponível em: <https://www.embarcados.com.br/arm-para-hobbyistas-parte-1/>. Acesso em: 13 mar. 2021.

[15] Silveira, Cristiano Bertulucci. Motor elétrico CA: Quais os tipos e como especificar? Citisystems, 2016. Disponível em: <https://www.citisystems.com.br/motor-eletrico/>. Acesso em: 13 mai. 2021.

[16] Umans, Stephen D. Máquinas elétricas de Fitzgerald e Kingsley; tradução: Anatólio Laschuk. – 7. ed. – Dados eletrônicos. – Porto Alegre: AMGH, 2014.