UNIVERSIDADE FEDERAL DE UBERLÂNDIA – UFU

FACULDADE DE ENGENHARIA ELÉTRICA

PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

APLICAÇÃO DE INTERNET DAS COISAS NO MONITORAMENTO DE CORRENTE, TENSÃO E TEMPERATURA EM MOTOR DE INDUÇÃO TRIFÁSICO

ARTUR DE ALMEIDA RIOS

UBERLÂNDIA-MG

JULHO - 2019

UNIVERSIDADE FEDERAL DE UBERLÂNDIA – UFU

FACULDADE DE ENGENHARIA ELÉTRICA

PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

APLICAÇÃO DE INTERNET DAS COISAS NO MONITORAMENTO DE CORRENTE, TENSÃO E TEMPERATURA EM MOTOR DE INDUÇÃO TRIFÁSICO

Dissertação apresentada por **Artur de Almeida Rios** à Universidade Federal de Uberlândia - UFU, como parte dos requisitos para a **obtenção do título de Mestre em Engenharia Elétrica**, na presença da banca examinadora, composta por:

Prof. **Luciano Coutinho Gomes**, Dr. (UFU) – Orientador
Prof. **Henrique José Avelar**, Dr. (CEFET-MG)
Prof. **Darizon Alves de Andrade**, Dr. (UFU)
Prof. **Edgard Afonso Lamounier Jr**, Dr. (UFU)

Ficha Catalográfica Online do Sistema de Bibliotecas da UFU
com dados informados pelo(a) próprio(a) autor(a).

R586 Rios, Artur de Almeida, 1987-
2019 Aplicação de internet das coisas no monitoramento de corrente, tensão e temperatura em motor de indução trifásico [recurso eletrônico] / Artur de Almeida Rios. - 2019.

Orientador: Luciano Coutinho Gomes.
Dissertação (Mestrado) - Universidade Federal de Uberlândia, Pós-graduação em Engenharia Elétrica.
Modo de acesso: Internet.
Disponível em: http://dx.doi.org/10.14393/ufu.di.2019.2044
Inclui bibliografia.
Inclui ilustrações.

1. Engenharia elétrica. I. Gomes, Luciano Coutinho, 1972-, (Orient.). II. Universidade Federal de Uberlândia. Pós-graduação em Engenharia Elétrica. III. Título.

CDU: 621.3

Bibliotecários responsáveis pela estrutura de acordo com o AACR2:
Gizele Cristine Nunes do Couto - CRB6/2091
Nelson Marcos Ferreira - CRB6/3074

UNIVERSIDADE FEDERAL DE UBERLÂNDIA

ATA DE DEFESA

Programa de Pós-Graduação em:	Engenharia Elétrica				
Defesa de:	Dissertação de Mestrado Acadêmico, 718, COPEL				
Data:	24/07/2019	Hora de início:	09:00	Hora de encerramento:	11:10
Matrícula do Discente:	11622EEL019				
Nome do Discente:	Artur de Almeida Rios				
Título do Trabalho:	Aplicação de internet das coisas no monitoramento de corrente, tensão e temperatura em motor de indução trifásico.				
Área de concentração:	Sistemas de energia elétrica				
Linha de pesquisa:	Máquinas e aterramentos elétricos				
Projeto de Pesquisa de vinculação:	Título: Verificação Experimental da Técnica de Transmissão de Energia Sem fios para Atendimento de Cargas Autônomas. Agência Financiadora: FAPEMIG Início: 06/08/2016. Término 05/08/2019. No. do Projeto na agência: APQ-01458-15. Professor Coordenador: Darizon Alves de Andrade				

Reuniu-se no Anfiteatro 1E da Faculdade de Engenharia Elétrica, Campus Santa Mônica, da Universidade Federal de Uberlândia, a Banca Examinadora, designada pelo Colegiado do Programa de Pós-graduação em Elétrica, assim composta: Professores Doutores: Darizon Alves de Andrade - FEELT/UFU; Henrique José Avelar - CEFET-MG; Edgard Afonso Lamounier Júnior - FEELT/UFU; Luciano Coutinho Gomes - FEELT/UFU,orientador(a) do(a) candidato(a).

Iniciando os trabalhos o(a) presidente da mesa, Dr(a). Luciano Coutinho Gomes, apresentou a Comissão Examinadora e o candidato(a), agradeceu a presença do público, e concedeu ao Discente a palavra para a exposição do seu trabalho. A duração da apresentação do Discente e o tempo de arguição e resposta foram conforme as normas do Programa.

A seguir o senhor(a) presidente concedeu a palavra, pela ordem sucessivamente, aos(às) examinadores(as), que passaram a arguir o(a) candidato(a). Ultimada a arguição, que se desenvolveu dentro dos termos regimentais, a Banca, em sessão secreta, atribuiu o resultado final, considerando o(a) candidato(a):

Aprovado(a).

Esta defesa faz parte dos requisitos necessários à obtenção do título de Mestre.

O competente diploma será expedido após cumprimento dos demais requisitos, conforme as normas do Programa, a legislação pertinente e a regulamentação interna da UFU.

Nada mais havendo a tratar foram encerrados os trabalhos. Foi lavrada a presente ata que após lida e achada conforme foi assinada pela Banca Examinadora.

Documento assinado eletronicamente por **Luciano Coutinho Gomes**, **Professor(a) do Magistério**

Superior, em 24/07/2019, às 11:14, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Edgard Afonso Lamounier Junior**, **Professor(a) do Magistério Superior**, em 24/07/2019, às 11:16, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Darizon Alves de Andrade**, **Professor(a) do Magistério Superior**, em 24/07/2019, às 11:17, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Henrique José Avelar**, **Usuário Externo**, em 24/07/2019, às 11:19, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

A autenticidade deste documento pode ser conferida no site https://www.sei.ufu.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **1401592** e o código CRC **47880FB1**.

Referência: Processo nº 23117.063798/2019-15 SEI nº 1401592

UNIVERSIDADE FEDERAL DE UBERLÂNDIA – UFU

FACULDADE DE ENGENHARIA ELÉTRICA

PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

APLICAÇÃO DE INTERNET DAS COISAS NO MONITORAMENTO DE CORRENTE, TENSÃO E TEMPERATURA EM MOTOR DE INDUÇÃO TRIFÁSICO

DISSERTAÇÃO APRESENTADA POR ARTUR DE ALMEIDA RIOS À UNIVERSIDADE FEDERAL DE UBERLÂNDIA - UFU, COMO PARTE DOS REQUISITOS PARA A OBTENÇÃO DO TÍTULO DE MESTRE EM ENGENHARIA ELÉTRICA.

LUCIANO COUTINHO GOMES, Dr.
Orientador

JOSÉ ROBERTO CAMACHO, Dr.
Coordenador do Programa de Pós-graduação *stricto sensu* em Engenharia Elétrica da UFU

DEDICATÓRIA

Dedico este trabalho a minha família, em especial as minhas amadas filhas, Vitória e Alice.

AGRADECIMENTOS

Primeiramente à Deus pela dádiva da vida, e por me conceder saúde e sabedoria.

Ao Prof. Dr. Luciano Coutinho, pela orientação, ensinamentos e apoio.

Ao Prof. Dr. Darizon de Andrade, pela oportunidade e incentivo.

Ao Prof. Dr. Henrique Avelar, pela paciência, ensinamentos, companheirismo e apoio em todas as etapas do trabalho.

A UFU pela oportunidade de acesso ao ensino de qualidade e gratuito.

Aos colegas e professores do LACE e FEELT.

Ao CEFET-MG campus Araxá, por disponibilizar os laboratórios e toda infraestrutura necessária para a elaboração deste trabalho.

Aos amigos pelo incentivo e por tornarem a caminhada menos árdua.

Reservo um agradecimento especial aos meus familiares: Aos meus tios e primos por todo apoio e carinho de sempre. Aos meus avós pelo exemplo de integridade e trabalho. A memória de meu pai, Artur, por ser responsável pelo homem que sou. A minha querida mãe, Sílvia, pela vida e por me incentivar e apoiar sempre nos estudos. A minha querida irmã, Paula, por ser um exemplo e me incentivar a nunca desistir. A minha amada esposa Larissa por todo companheirismo, carinho, paciência, compreensão e acima de tudo segurar as pontas para que eu pudesse estudar e me dedicar a este trabalho. As minhas queridas e amadas filhas, Vitória e Alice, por serem luz em minha vida, me incentivando a levantar todos os dias, trabalhar e estudar para me tornar um ser humano cada dia melhor.

Enfim, agradeço a todos que de alguma maneira fizeram parte desta caminhada.

“Os homens criam as ferramentas, e as ferramentas recriam os homens.”

John Culkin e Marshall McLuhan

RESUMO

A Internet das Coisas (*IoT - Internet of Things*) pode ser definida como a conexão entre o mundo físico e o digital, ou seja, uma rede global entre objetos de uso rotineiros, pessoal ou industrial, que possuem tecnologia embarcada capaz de coletar, transmitir e tomar decisões baseada em banco de dados. O trabalho proposto visa desenvolver e implementar um sistema de aquisição de baixo custo, seguindo as tendências da indústria 4.0, que está inserindo a *IoT* em ambientes industriais. Utilizando microcontroladores ESP32, do fabricante *Espressif Systems*®, para monitorar sobrecorrente, sobretensão e temperatura alta do motor de indução trifásico. As grandezas elétricas são aquisicionadas pelo Módulo de Aquisição e Condicionamento de Sinais de Tensão e Corrente. A temperatura é supervisionada por um sensor de temperatura interno ao motor. Estes dados são lidos, processados e transmitidos por dois ESP32, e aplicando a tecnologia *IoT* os valores de sobrecorrente, sobretensão e temperatura alta são salvos em nuvem. Os dados são disponibilizados para visualização, em forma de gráfico, na plataforma *ThingSpeak*® e alarmes são disparados para *smartphones* de usuários cadastrados, por meio do aplicativo *Telegram*®.

Palavras-chave: ESP32, Indústria 4.0, Internet das Coisas, Microcontroladores.

ABSTRACT

The Internet of Things (IoT) can be defined as the connection between the physical and the digital world, that is, a global network between everyday, personal or industrial objects that have embedded technology capable of collecting, transmitting and make database-based decisions. The proposed work aims to develop and implement a low cost acquisition system, following industry trends 4.0, which is inserting IoT in industrial environments. Using ESP32 microcontrollers from Espressif Systems® to monitor overcurrent, overvoltage and high temperature of the three-phase induction motor. Electrical quantities are acquired by the Voltage and Current Signal Acquisition and Conditioning Module. The temperature is supervised by an internal engine temperature sensor. This data is read, processed and transmitted by two ESP32, and by applying IoT technology the overcurrent, overvoltage and high temperature values are clouded. Data is made available for viewing in graph form on the ThingSpeak® platform and alarms are triggered for registered users' smartphones via the Telegram® application.

Keywords: ESP32, Industry 4.0, Internet of Things, Microcontrollers.

SUMÁRIO

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ABNT – Associação Brasileira de Normas Técnicas

ADC – *Analog to Digital Converter* (Conversor Analógico-Digital)

ANSI – *American National Standards Institute*

AO – Amplificador Operacional

API – *Application Programming Interface* (Interface de Programação de Aplicativos)

BI – *Business Intelligence*

BOT – Diminutivo de *Robot*

CLP – Controlador Lógico Programável

CT – Torque Constante

CV – Cavalo Vapor

DAC – *Digital to Analog Converter* (Conversor Digital-Analógico)

Db – Decibéis

F – Frequência

IA – Inteligência Artificial

IDE – *Integrated Development Environment* (Ambiente de Desenvolvimento Integrado)

IEC – *International Electrotechnical Commission*

IoT – *Internet of Things* (Internet das Coisas)

ISA – *International Society of Automation*

MIT – Motor de Indução Trifásico

Na – Número de amostras por período

NBR – Norma Brasileira

NTC – Coeficiente de Temperatura Negativo

PTC – Coeficiente de Temperatura Positivo

PWM – *Pulse Width Modulation* (Modulação por Largura de Pulso)

RC – Resistivo-Capacitivo

RMS – *Root Mean Square* (Valor eficaz)

RPM – Rotações por Minuto

T – Período

Ta – Taxa de amostragem

V/F – Tensão/Frequência

1. INTRODUÇÃO

Este capítulo apresenta uma visão geral sobre o tema abordado, os recursos utilizados e os objetivos a serem alcançados.

A decisão por elaborar um sistema de aquisição de dados de baixo custo, para monitoramento e aquisição de corrente, tensão e temperatura do motor de indução trifásico, aplicando Internet das Coisas (*IoT - Internet of Things*) neste trabalho foi tomada tendo em vista que *IoT* é um tema relativamente novo no Brasil. Porém o governo brasileiro sinalizou apoio a projetos tecnológicos voltados para *IoT*, inclusive disponibilizando linhas de investimentos atrativos pelo BNDES (Banco Nacional de Desenvolvimento Econômico e Social) (BNDES, 2018).

A *IoT* pode ser definida como a conexão entre o mundo físico e o digital. Tal tecnologia permite que objetos cotidianos, pessoais ou industriais, sejam acionados pela Internet, desde que sejam dotados de computação embarcada e identificados, trocando dados entre si e também com pessoas por meio da rede, permitindo a criação de serviços em diversas áreas, como: energia, segurança, meio ambiente, trânsito, mobilidade, logística, indústria, etc. (OLIVEIRA, NIIZU e BASSETO, 2017).

A *IoT* possui diversas aplicações que estão causando impacto direto ao consumidor, como por exemplo na área da saúde, no setor urbano e rural. Além dessas aplicações, vê-se hoje um grande potencial de ganho através da aplicação de *IoT* na eficiência operacional e no aumento do potencial produtivo.

Trabalhos anteriores, como (CARVALHO, 2010), já demonstravam a importância de métodos de monitoramento em tempo real das condições operacionais de motores de indução trifásicos, como corrente e tensão, e realizando a comunicação sem fio.

É notório o surgimento de trabalhos como de (GOUNDAR, PILLAI, *et al.*, 2015), voltados para a indústria, com o monitoramento em tempo real das condições de motores de indução trifásicos, como vibração e temperatura, utilizando-se de sistemas microcontrolados e *IoT*.

A escolha do Motor de Indução Trifásico (MIT), para ser monitorado pelo sistema de aquisição proposto, se deu por sua posição de liderança em comparação aos demais tipos de motores. O MIT está presente nos mais diversos setores, particularmente na indústria essa liderança é mantida hoje e deverá perdurar por bastante tempo. Além disso, as máquinas de indução são robustas construtivamente, apresentam elevado rendimento e custo inicial baixo (GODOY, SILVA, *et al.*, 2016).

Sistemas de aquisição de baixo custo baseados em microcontroladores, como o proposto nesse trabalho, vem sendo pesquisados nos últimos anos (ANDREOLA, SENTER, *et al.*, 2016).

Pois, segundo (GOUNDAR, PILLAI, *et al.*, 2015), o uso de plataformas microcontroladas é uma solução para adquirir dados dos sensores, comunicar com outros dispositivos, armazenar informações localmente ou na nuvem e alertar o usuário quando falhas forem detectadas.

Microcontrolador é um CI (Circuito Integrado) capaz de efetuar processos lógicos com rapidez, por englobar os circuitos do microprocessador, memória, conversão analógico-digital (AD) e portas digitais de entrada e saída, como também circuitos *PWM* (*Pulse Width Modulation* - Modulação por largura de pulso) e de comunicação dedicados. A facilidade de programação, aliada à quantidade de circuitos internos, permite seu uso em uma vasta gama de tarefas.

Dentre os vários microcontroladores encontradas destacam-se três para fins de comparação, por possuírem *WiFi* integrado, os quais são: ESP8266, ESP32 e MKR1000.

O conhecimento e familiarização com essas tecnologias se tornam de suma importância, assim como o estudo e comparação entre novas plataformas microcontroladas que surgem a cada dia, visando acompanhar as tendências da indústria 4.0 (SOUZA, 2000).

ESP8266 é um microcontrolador projetado pela *Espressif Systems*®, empresa chinesa sediada em Xangai. Ele foi lançado como uma solução para realizar uma ponte entre os microcontroladores existentes e o *WiFi*. O ESP32 é sua versão melhorada, pois além de comunicação *WiFi* também possui comunicação *Bluetooth* (KOLBAN, 2016).

MKR1000 é uma plataforma projetada pelo grupo Arduino®, que iniciou seus projetos na cidade de Ivrea, Itália, em 2005. Essa plataforma de prototipagem eletrônica *open-source* (Código Aberto) combina as funcionalidades de modelos anteriores, além de dispor de *WiFi* integrado. É baseada em *hardware* e *software* flexíveis e fáceis de usar, até mesmo por pessoas que não são especialistas nessa área (ARDUINO HOME PAGE, 2019a).

Os dois microcontroladores da *Espressif Systems*® citados estão inseridos em uma plataforma NodeMCU, consistindo de um kit de desenvolvimento para auxiliar na criação de protótipos de produtos voltados para *IoT*, com códigos abertos (NODEMCU HOME PAGE, 2018).

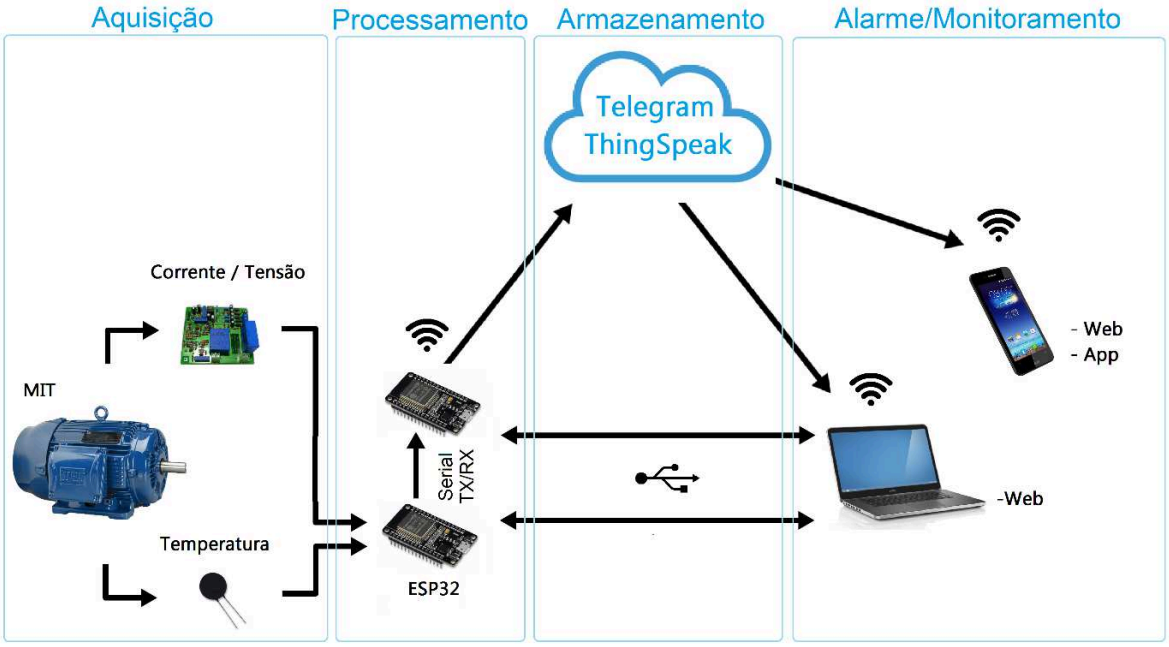
Para a programação das plataformas MKR100, ESP32 e ESP8266, foi utilizado o Ambiente de Desenvolvimento Integrado (*IDE -Integrated Development Environment*), da Arduino®, que é de código aberto, desenvolvido em Java, e projetado para facilitar a escrita de programas e o *upload* para as plataformas de desenvolvimento. Esse *software*, além de ser utilizado para programar qualquer plataforma Arduino® (ARDUINO SOFTWARE PAGE, 2019b), também pode ser utilizado no desenvolvimento de códigos para as plataformas ESP32 ou ESP8266, por esse motivo foi utilizado nesse trabalho.

O principal objetivo desta dissertação é avaliar e conceber um sistema de aquisição de dados de baixo custo, para monitoramento de corrente, tensão e temperatura do motor de indução trifásico, aplicando *IoT*. As grandezas elétricas foram lidas pelo módulo de aquisição de sinais e a temperatura foi monitorada por um sensor. O ESP32 recebeu estes dados, realizou o processamento dos mesmos, e aplicando a tecnologia *IoT,* os parâmetros foram salvos na nuvem e disponibilizados para visualização em forma de gráfico na plataforma *ThingSpeak* (THINGSPEAK HOME PAGE, 2019a). Além disso, alarmes foram disparados via *Telegram* (TELEGRAM HOME PAGE, 2019a) a *smartphones* cadastrados, conforme Figura 1.

Esse objetivo geral é concretizado pelos seguintes objetivos específicos:

- Realizar um estudo sobre a evolução industrial, avanços tecnológicos alcançados ao longo dos anos, seus impactos na indústria e transformação na sociedade.
- Definir os conceitos de Indústria 4.0, seus pilares e sua situação no Brasil.
- Definir os conceitos de *IoT* e sua aplicação na indústria.
- Escolher, através de testes comparativos, a plataforma microcontrolada mais adequada para se trabalhar com aquisição de dados, envolvendo *IoT*.
- Definir plataformas para armazenar os dados em servidores na nuvem.
- Realizar a comunicação entre todos os hardwares e softwares: módulos de aquisição de dados, microcontrolador, internet, plataformas de armazenamento, web e aplicativo de smartphone.

Figura 1 – Diagrama do projeto.



Fonte: Própria autoria.

2. REVISÃO DA LITERATURA

Neste capítulo foi realizado um estudo da arte sobre o tema proposto. Foram mostrados os avanços tecnológicos alcançados ao longo dos anos, seus impactos na indústria e transformação na sociedade, durante quatro revoluções industriais. Foi realizado um estudo sobre a situação do Brasil no contexto mundial, referente a indústria 4.0. E também foi demonstrada a evolução das redes sem fio na indústria e conceituado o que é o motor de indução trifásico (MIT).

2.1. Evolução Industrial

A história da evolução da indústria passa por períodos de revolução, a seguir é apresentada uma breve descrição de cada período, conforme ilustrado na Figura 2.

A Primeira Revolução Industrial originou na Inglaterra, na segunda metade do século XVIII, suas principais características foram o surgimento da máquina a vapor, tendo o carvão como principal fonte de energia, força de trabalho não especializada nem qualificada e manufaturas. Outro marco da primeira revolução industrial foi o aperfeiçoamento da máquina a vapor por James Watt, colocando a indústria têxtil como símbolo da produção excedente, gerando a riqueza da época, criando um novo modelo econômico (COTRIM, 2005).

A Segunda Revolução Industrial aconteceu na Europa, EUA e Japão na segunda metade do século XIX. Iniciou-se quando Henry Ford criou a linha de produção em massa, criando o conceito da produção em escala, reduzindo o custo e popularizando o produto, para que a massa trabalhadora pudesse adquirir, criando um ciclo virtuoso na indústria e na economia. A segunda revolução industrial exigiu uma base técnica mais complexa como, por exemplo, o refino do petróleo, máquinas e motores mais sofisticados movidos a energia elétrica e mão de obra especializada. Os Estados Unidos foram a grande potência econômica e o principal modelo de industrialização dessa fase ou estágio da Revolução Industrial, caracterizada ainda pelo predomínio da indústria automobilística, e outras indústrias a ela ligadas, como petroquímica, siderúrgica, metalúrgica, etc. (ENCICLOPEDIA DE CARACTERÍSTICAS, 2017)

A Terceira Revolução Industrial foi mais abrangente em comparação às anteriores, se iniciou após a segunda guerra mundial. Essa fase foi marcada pela entrada na era da automação, com a implantação de computadores no chão-de-fábrica, gerando um aumento da produção, da qualidade e da segurança dos processos. Tudo isso foi possível graças a controles eletrônicos, que a partir de dados recebidos de sensores e transmissores de campos, tomavam decisões de

forma autônomas. Iniciou-se tanto nos Estados Unidos, sobretudo na Califórnia, como no Japão e Europa ocidental, em particular na Alemanha. É marcada pelo predomínio de indústrias altamente sofisticadas, que exigem maior tecnologia e qualificação do trabalhador (PENA, 2013).

A Quarta Revolução Industrial, dita Indústria 4.0, inaugurou uma nova era ainda em transição. Seu ponto principal é a *Internet*, que está consolidada entre as pessoas como um grande canal de comunicação convergente de todas as tecnologias, e agora está sendo colocado dentro da indústria com seus conceitos adaptados a máquinas e equipamentos. Esse conceito é chamado de *IoT*, onde as máquinas e equipamentos estão conectados em rede e fornecendo informações de forma única, buscando uma diminuição dos custos, aumento de produção, eficiência energética, integração da cadeia produtiva e enfoque na evolução tecnológica (DMD SOLUTIONS, 2019).

Resumindo, as três primeiras revoluções industriais trouxeram a produção em massa, as linhas de montagem, a eletricidade e a tecnologia da informação, elevando a renda dos trabalhadores e fazendo da competição tecnológica o cerne do desenvolvimento econômico. A quarta revolução industrial, que terá um impacto mais profundo e exponencial, se caracteriza, por um conjunto de tecnologias que permitem a fusão do mundo físico, digital e biológico.

Figura 2 – Diagrama com as quatro revoluções industriais.

Fonte: (SORDAN, 2019)

2.2. Indústria 4.0

Segundo (SILVEIRA, 2017) a denominação Indústria 4.0 surgiu a partir de um projeto voltado a tecnologias do governo alemão, e em 2011 a expressão foi usada pela primeira vez, na Feira de Hannover. O grupo responsável pelo projeto, administrado por Siegfried Dais e Kagermann, apresentou um relatório de orientações para o Governo Federal Alemão, com o intuito de planejar sua implantação, em 2012. Foi publicado um trabalho final sobre o desenvolvimento da Indústria 4.0, nessa mesma feira em abril de 2013.

O fundamento básico da indústria 4.0 define que, realizando a conexão entre as máquinas e seus *softwares*, as empresas poderão arquitetar *smart grids* ao longo da cadeia de valor, e com isso, de forma autossuficiente, poderão controlar os módulos da produção. Dessa maneira as manutenções poderão ser agendadas, de forma independente, pelas fábricas inteligentes, e assim ajustar os requisitos e mudanças não planejadas na produção e antever falhas (KAGERMANN, WAHLSTER e HELBIG, 2013).

O fundamento de indústria 4.0 tende para a digitalização da indústria. A definição se refere a uma estrutura modular, que utiliza *hardwares* e *softwares* para monitorar e controlar processos industriais, com o objetivo de melhorar continuamente os múltiplos índices de desempenho dentro do processo. Essa concepção é possível graças a um dos pilares da indústria 4.0, a *IoT*, que cria *smart grids* entre sensores, máquinas, sistemas e internet (CUNHA, ALMEIRA, *et al.*, 2016). A Figura 3 ilustra o conceito de Indústria 4.0.

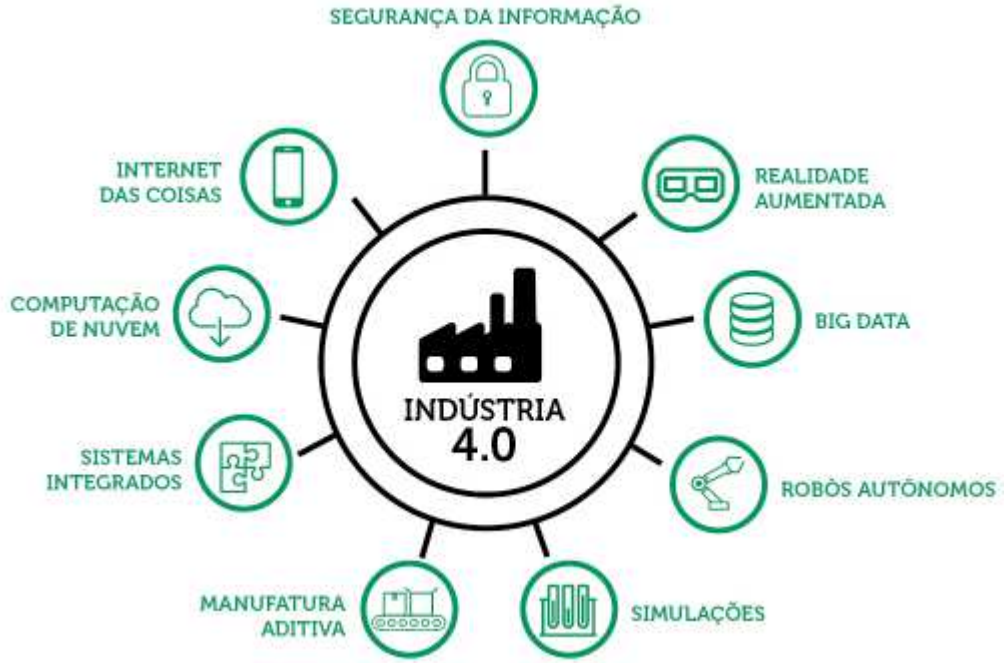
Figura 3 – Representação da Indústria 4.0.

Fonte: (FABIO RODRIGUES, 2018)

2.2.1. Pilares da Indústria 4.0

A Indústria 4.0 possui nove pilares, conforme Figura 4, são eles: *Big Data*, Robôs Autônomos, Simulação, Manufatura Aditiva, Sistemas de Integração, Computação em Nuvem, Internet das coisas, Segurança da Informação, Realidade Aumentada (GERBERT, LORENZ, *et al.*, 2015).

Figura 4 – Os pilares da Indústria 4.0.



Fonte: (NASCIMENTO, 2018)

Big Data: A análise e gestão de grandes quantidades de dados (banco de dados) na indústria permite maior performance, como por exemplo otimizar a qualidade da produção, economizar energia e melhorar o equipamento. Na indústria 4.0, a obtenção e avaliação exaustiva dos dados de muitas fontes diferentes (sistemas e equipamentos) se tornará padrão para suportar a tomada de decisão em tempo real. Sua função é central ao analisar detalhadamente os números e as estatísticas de uma indústria, e com isso identificar falhas nos processos, otimizar a qualidade da produção e poupar energia, tornando mais eficiente o uso de todos recursos (GERBERT, LORENZ, *et al.*, 2015).

Robôs autônomos: Não é novidade a utilização de robôs na indústria, mas na indústria 4.0eles adquirem habilidades para lidar com tarefas mais complexas, com isso eles terão uma maior utilidade por parte das mesmas. Outro ponto forte é que terão maior autonomia, flexibilidade e serão mais cooperativos, na medida em que interagem com outros robôs e trabalham

lado a lado com os seres humanos com segurança, e aprendendo com os mesmos. Além disso, esses robôs vão ter um custo menor, maior segurança e mais capacidades em relação aos utilizados até então nas indústrias (GERBERT, LORENZ, *et al.*, 2015).

Simulações: Para assegurar a qualidade e efetividade no desenvolvimento de produtos, a utilização de simulação computacional é essencial. Essa ferramenta auxilia as empresas a desenvolverem e aperfeiçoarem seus produtos e processos. Estas simulações exploram os dados em tempo real que refletem o mundo físico em um modelo virtual, o qual irá incluir máquinas, produtos e seres humanos. Isto irá permitir aos operadores testar e otimizar as configurações das máquinas para o próximo produto na linha de produção virtual antes de qualquer mudança no mundo físico, reduzindo assim os tempos de preparação das máquinas e aumento da qualidade (GERBERT, LORENZ, *et al.*, 2015).

Manufatura aditiva: Com a impressão 3D as empresas estão começando a adotar manufatura aditiva, que é usada principalmente para a criação de protótipos e produção de componentes individuais. Com o advento da Indústria 4.0, estes métodos serão amplamente utilizados para produzir pequenos lotes de produtos personalizados que oferecem vantagens de construção e desenhos complexos. As empresas aeroespaciais já estão usando a manufatura aditiva para aplicar novos projetos que diminuem o peso das aeronaves, reduzindo as suas despesas de matérias-primas, como o titânio (GERBERT, LORENZ, *et al.*, 2015).

Sistemas de integrados: Atualmente, nem todos os sistemas são totalmente integrados, faltando uma coesão entre empresas, distribuidores e clientes e até mesmo o processo de produção de uma indústria carece de uma integração plena entre departamentos, como engenharia, manutenção e produção. A indústria 4.0 propõe uma maior sinergia entre todos que façam parte do mesmo ambiente, garantindo uma gestão integral de experiência, para que cadeias de valor sejam realmente automatizadas (GERBERT, LORENZ, *et al.*, 2015).

Computação em Nuvem: O desenvolvimento da indústria 4.0 está sendo sustentado por esse modelo tecnológico. Cada vez mais atividades relacionadas com a produção de bens e serviços requerem o uso de aplicativos e dados compartilhados entre diferentes lugares e sistemas, não se limitando aos servidores de uma empresa. Com as fábricas totalmente digitalizadas, a computação na nuvem servirá para armazenar todas as informações em um banco de dados que poderá ser acessado de qualquer lugar do mundo, a qualquer hora, por meio da internet, e assim reduzir custo, tempo e aumentar a eficiência (GERBERT, LORENZ, *et al.*, 2015).

Internet das coisas: A *IoT* é a conexão lógica de todos os dispositivos e meios relacionados ao ambiente produtivo em uma rede global entre objetos de uso rotineiros, pessoal ou industrial, que possuem tecnologia embarcada capaz de coletar, transmitir e tomar decisões

baseada em dados. Com a *IoT*, um maior número de dispositivos será acrescentado e se conectará por meio de padrões tecnológicos. Isso permitirá que os dispositivos de campo se comuniquem e interajam com os outros, como controladores mais centralizados, conforme necessário. Também descentraliza a análise e tomada de decisões, que permitirá respostas em tempo real (GERBERT, LORENZ, *et al.*, 2015).

Segurança da Informação: Também conhecida por Cibersegurança ou *Cybersecurity*, é um elemento fundamental para proteger sistemas e informações de possíveis ameaças e falhas, que podem vir a causar transtornos na produção. Existe uma norma que estabelece um conjunto de boas práticas para minimizar o risco de redes de sistemas de controle sofrerem ciberataques, é a norma ANSI/ISA99 (ISA99, 2002) (GERBERT, LORENZ, *et al.*, 2015).

Realidade aumentada: Estes sistemas ainda estão em seus estágios iniciais no Brasil, porém a realidade aumentada suporta uma variedade de aplicações e serviços em diferentes campos, como a medicina e educação. Aplicada às necessidades da indústria, é possível ter desde instruções de montagem enviadas via celular para desenvolvimentos para peças de protótipo até o uso de óculos de realidade aumentada para a gestão e operação de determinadas máquinas, melhorando procedimentos de trabalho e tomadas de decisões (GERBERT, LORENZ, *et al.*, 2015).

2.2.2. Situação da Indústria 4.0 no Brasil

Se fizermos uma comparação com o panorama no restante do mundo, a indústria 4.0 no Brasil ainda está um pouco atrasada. Segundo o relatório do Fórum Econômico Mundial, são apenas 25 os países que já aproveitam os benefícios da Indústria 4.0. Os países que lideram as mudanças se concentram em alguns lugares da Europa e do sul da Ásia. A América Latina como um todo ainda está dando seus primeiros passos (WORLD ECONOMIC FORUM, 2018).

Este relatório Econômico Mundial leva em consideração seis *Drivers* de Análise de Produção, são eles:

- Tecnologia e Inovação
- Capital Humano
- Comércio Global e Investimento
- Estrutura Institucional
- Produção Sustentável
- Ambiente de demanda

E dois tópicos relacionados à Estrutura de Produção, são eles:

- Complexidade
- Escala

Na Figura 5 é apresentada a estrutura do modelo de diagnóstico de capacidade de produção, utilizado no relatório do Fórum Econômico Mundial.

Figura 5- Estrutura de Modelo de Diagnóstico

Fonte: Adaptado de (WORLD ECONOMIC FORUM, 2018).

O relatório demonstra uma comparação com 100 países e suas economias, levando em consideração os *drivers* de análise de produção (eixo vertical) e os tópicos relacionados à Estrutura de Produção (eixo horizontal), o resultado foi atribuído a um dos quadrantes, conforme mostrado na Figura 6.

Figura 6 – Quadrante para comparação.

Fonte: Adaptado de (WORLD ECONOMIC FORUM, 2018).

Definição dos quadrantes: Os quatro quadrantes fornecem uma visão para comparação entre os países que possuem perspectivas semelhantes para o futuro da produção, e são definidos da seguinte maneira:

Liderando: Países que possuem hoje uma forte base de produção, e que exibem um alto nível de prontidão para o futuro, por meio do forte desempenho no eixo Drivers de Produção. Esses países também têm o valor econômico mais atual em risco para futuras interrupções.

Legado: Países que hoje possuem uma forte base de produção, mas que se encontram em risco para o futuro, devido ao desempenho inferior do componente Drivers de Produção.

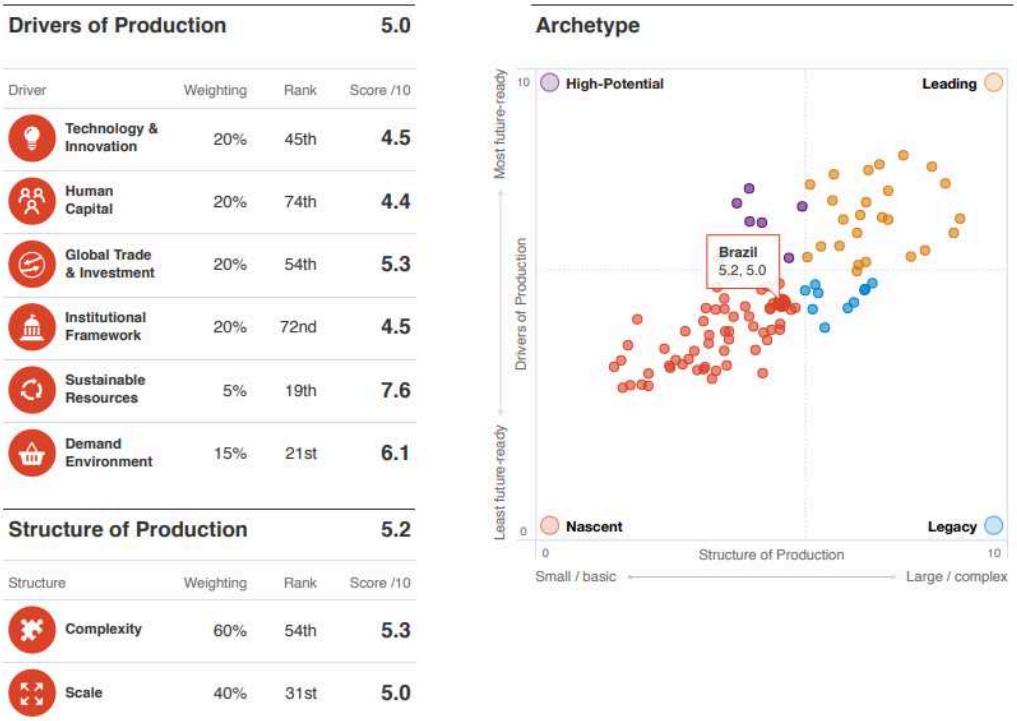
Alto Potencial: Países que hoje trabalham com uma base de produção limitada, mas que pontuam bem em toda a componente Drivers de Produção, indicando que existe capacidade para aumentar a produção no futuro, dependendo das prioridades dentro da economia nacional.

Nascente: Países que hoje possuem uma base de produção limitada e um nível baixo de prontidão para o futuro, devido ao fraco desempenho no eixo dos Drivers de Produção.

A Figura 7(a) apresenta as notas do Brasil, de 0 a 10, nos drivers de produção e na estrutura de produção, e com base na média dessas notas define a posição no quadrante.

Pela média em suas notas o Brasil se encontra no quadrante "Nascente". Contudo, pelo gráfico da Figura 7(b), percebe-se que o Brasil não está longe de chegar ao quadrante ideal.

Figura 7- Posição do Brasil nos quatro quadrantes.



(a) (b)

Fonte: Adaptado de (WORLD ECONOMIC FORUM, 2018).

O relatório apresenta algumas conclusões sobre o Brasil:

- 10% do PIB nacional provêm do setor manufatureiro, que é o 9° maior do mundo.

- Levando em consideração a complexidade, a estrutura de produção do Brasil é relativamente baixa.

- Possui desempenho médio quanto aos drivers de produção. Seus drivers mais bem colocados são os recursos sustentáveis e ambiente de demanda.

- O Brasil deve buscar alavancar as relações globais para facilitar o conhecimento e a transferência de tecnologia, por ser um dos principais destinos dos investimentos estrangeiros diretos e investimentos em novas áreas.

- No quadro institucional o Brasil é o maior desafio. Tratar a eficiência regulatória e a governança orientada para o futuro deve ser prioridade.

O Brasil possui a quinta maior população mundial, e com isso uma grande riqueza de recursos humanos, porém sua capacidade atual de força de trabalho deixa a desejar em relação às habilidades digitais, engenharia, pensamento crítico e outras áreas fundamentais, que são pontos decisivos para o sucesso no futuro.

2.3. Futuro da *IoT* no Brasil

O governo federal brasileiro, na última semana de dezembro de 2018, instituiu o decreto para o Plano Nacional de Internet das Coisas, direcionando esforços para acelerar o avanço da nova tecnologia no Brasil (RAFAEL BITENCOURT, 2018).

A elaboração do plano de *IoT* começou a ser discutida em 2016, quando o MCTIC (Ministério da Ciência, Tecnologia, Inovações e Comunicações), lançou a consulta pública "BNDES/FEP Prospecção nº 01/2016 – Internet das Coisas (*Internet of Things - IoT*)". A consulta foi vencida pelo consórcio McKinsey/Fundação CPqD/ Pereira Neto Macedo, que realizou o estudo "Internet das Coisas: um plano de ação para o Brasil" (BNDES, 2017a).

Os investimentos em iniciativas em *IoT* irão priorizar quatro setores: Cidades, Saúde, Rural e Indústrias. O BNDES está apoiando projetos que priorizem estes setores por meio de um projeto chamado "BNDES Pilotos *IoT*". O objetivo é a seleção de projetos-piloto de testes de soluções tecnológicas de *IoT* para apoio com recursos não reembolsáveis, que poderão chegar a 50% dos itens financiáveis. O orçamento total para projetos de *IoT* é de R$ 20 milhões e cada projeto-piloto receberá um apoio do banco no valor mínimo de R$ 1 milhão. Poderão ser apoiadas soluções executadas por Instituições Tecnológicas públicas ou privadas sem fins lucrativos dentro do foco de cada um dos seguimentos apresentados na Figura 8 (BNDES, 2018).

Figura 8 - Visão do Plano de Ação de *IoT* para o Brasil.

Fonte: Adaptado de (BNDES, 2017b)

2.4. Aplicação da rede sem fio na indústria

Na indústria, a comunicação dos instrumentos de campo com os controladores e sistemas de supervisão é feita através de cabeamento desde que surgiram os primeiros sistemas de automação industrial. São inúmeros cabos saindo das salas de controle e supervisão para o chão-de-fábrica, recebendo e transmitindo os mais diversos tipos de dados.

A principal vantagem do uso de cabos é a confiabilidade na transmissão de dados, ou seja, há poucas perdas nos dados que precisam ser transmitidos e recebidos de um ponto a outro. Outro fator importante é que os instrumentos instalados no campo necessitam de cabeamento para alimentação e, muitas vezes, é viável que a transmissão de dados seja feita através da mesma estrutura e em alguns casos até pelo mesmo cabo.

Há um grande desenvolvimento no setor de transmissão de dados e novas tecnologias surgem abrindo mais opções para as empresas. Uma dessas tecnologias é a transmissão de dados sem fio (*wireless*), em que o transmissor e o receptor se comunicam sem a necessidade de cabeamento. Esse tipo de transmissão já é muito utilizado e substitui bem a comunicação por cabeamento em alguns casos, como na transmissão de voz por rádio frequência, na telefonia e na internet. Inicialmente não se utilizava essa tecnologia na indústria devido à perda relativamente alta de dados em sua transmissão, causada por interferências como campos magnéticos e barreiras físicas.

Atualmente, há opções de tecnologias de transmissão sem fio que prometem baixa interferência, alta confiabilidade na transmissão de dados e trabalham com baixa potência. Podemos citar como exemplo *WirelessHart*, ZigBee, ISA SP100, *Bluetooh* e *WiFi* que vem sendo utilizado em várias aplicações com sucesso.

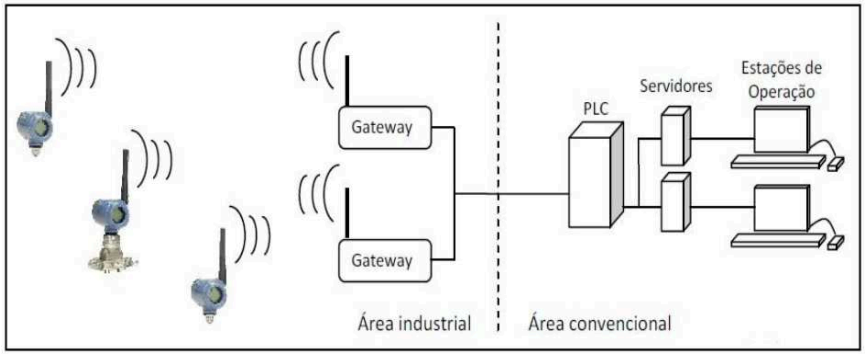
As redes industriais que utilizam transmissão por cabeamento necessitam de grandes investimentos em infraestrutura e um alto custo de mão-de-obra sempre que a rede industrial sofre manutenção ou é expandida.

Já com a utilização de redes sem fio há grande redução do custo da infraestrutura e simplificação das instalações, pois não há necessidade de uso de bandejas e dutos. O custo de manutenção também é reduzido pois, sem o cabeamento, a substituição de dispositivos é facilitada (FAGUNDES, RIOS, *et al.*, 2014).

2.5. Evolução da Rede Sem Fio

Até o início da década de 1990, a utilização de sistemas de comunicação sem fio sofria com poucas opções de hardwares a preços elevados, que não permitiam interoperabilidade (sistemas proprietários) e não possuíam sistemas eficientes de segurança. Porém, a indústria apresentava uma crescente necessidade de sistemas abertos e confiáveis para diversas aplicações, inclusive sem fio. As mudanças das redes de dados ocorridas nos últimos anos foram mais drásticas do que as ocorridas com o rádio no último século. Essa evolução tecnológica, aliada à crescente necessidade de competitividade na indústria, criou um nicho de mercado para a utilização das redes de dados sem fio nos instrumentos de campo, sensores, transmissores e atuadores, conforme ilustrado na Figura 9, o que introduziu uma maior flexibilidade de implementação (RIEGO, 2009).

Figura 9 - Arquitetura da rede sem fio para coleta de dados.



Fonte: (RIEGO, 2009).

2.6. Evolução dos Sistemas de Automação

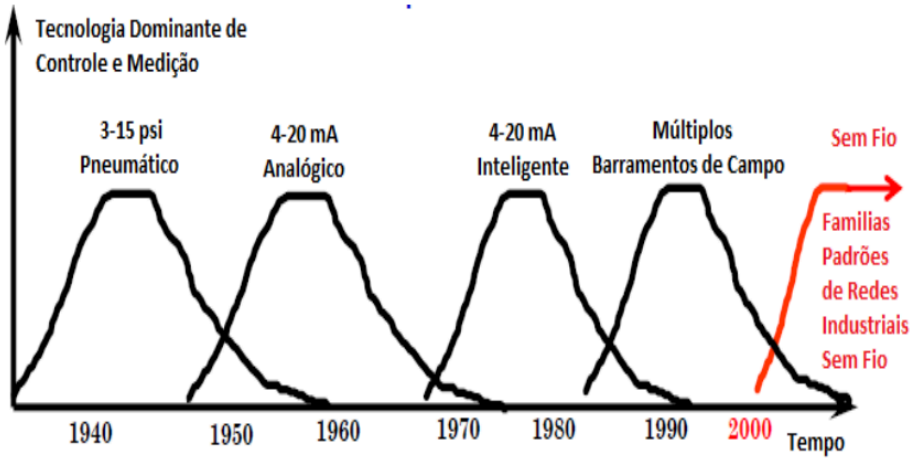
Na época da segunda revolução industrial surgiram os primeiros sistemas de automação, onde trabalhos manuais foram aos poucos substituídos por máquinas dedicadas, que buscavam o aumento e eficiência da produtividade. Os mecanismos possuíam vida útil reduzida e altos índices de manutenção, por serem puramente mecânicos. O surgimento de sistemas mais complexos se deu com o desenvolvimento dos relés, que possibilitaram a criação das primeiras lógicas. Por volta dos anos 40 o chão de fábrica começou a ser controlado por instrumentos de pressão físicos em sinais de 3-15psi. Os transistores surgiram na mesma época, viabilizando a criação de controle de máquinas por comandos numéricos, bem como o controle da instrumentação analógica com referência de tensão. Na década de 60, visando monitorar os instrumentos de campo, surge o sinal analógico de 4-20mA. Na década de 70, os microprocessadores são aprimorados e com isso se manifesta a ideia de aproveitar os computadores para monitorar os processos a partir de um ponto central. Porém, os computadores possuíam alto custo, eram difíceis de programar, eram equipamentos grandes que ocupavam muito espaço e, além disso, eram sensíveis ao ambiente industrial. Na sequência foi desenvolvido o CLP (Controlador Lógico Programável), por necessidade da indústria automobilística (SANTOS e LUGLI, 2013).

Nesta época também se desenvolveram os primeiros sensores inteligentes e os controles digitais associados aos mesmos. Com a evolução dos instrumentos digitais e a necessidade de algo para interligá-los, manifesta-se a ideia da criação de uma rede que conectaria todos os instrumentos e possibilitaria todos os dados do processo em um mesmo meio físico.

Dessa maneira se faz necessário a criação de um padrão, de acordo com as normas, para a rede de controle dos instrumentos inteligentes ficasse padronizada. O padrão IEC 61158, da IEC (*International Electrotechnical Commission*), foi criado no ano de 2000, quando todas as organizações interessadas se juntaram para criar o barramento de campo *fieldbus*. Outras atualizações e padrões surgiram depois, incluindo vários protocolos, como por exemplo o Ethernet (LUGLI e SANTOS, 2011).

A padronização das redes sem fio (*wireless*), foi o desafio seguinte, observando a convergência das redes sem fio para os padrões SP 100 e *WirelessHart*. Com isso surgiram vários estudos para viabilidade e padronizar as redes sem fio, devido aos vários benefícios das mesmas, como flexibilidade, baixo custo de instalação e manutenção. Em 2005 a ISA (*International Society of Automation*) criou o ISA 100 *Committee*, com o objetivo de definir padrões e informações relacionadas ao padrão ISA SP100, determinando a metodologia de implantação de um sistema sem fio para automação (ISA, 2008). A Figura 10 ilustra essa evolução.

Figura 10 – Evolução das Redes Industriais.



Fonte: Adaptado de (ISA, 2008).

2.7. Motor elétrico de indução trifásico (MIT)

Não existe um consenso sobre quem inventou o motor, pois quando os primeiros motores surgiram, diversos pesquisadores trabalhavam em paralelo em busca do mesmo objetivo: a criação de um dispositivo que pudesse gerar energia mecânica quando alimentado por energia elétrica. Sabe-se que por volta de 1820, o dinamarquês Hans Christian Oersted descobriu o eletromagnetismo, que é o princípio do funcionamento dos motores. A indução magnética foi descoberta por volta de 1831 por Michael Faraday, nessa década surgiram as primeiras máquinas elementares, porém o motor elétrico somente apareceria anos mais tarde. O inventor alemão, Erns Werner Siemens, por volta de 1866 criou um dínamo que funcionava como gerador de eletricidade e também como motor de corrente contínua, que pode ser considerado como o primeiro motor da história. No final do século XIX a corrente contínua perdeu lugar para a corrente alternada e novas descobertas foram feitas em relação ao motor. O primeiro a desenvolver um modelo finalizado de motor de corrente alternada foi Mikhail Dolivo-Dobrovolsky, que em 1889 registrou a patente de um motor trifásico com potência de 80 W e rendimento de 80%. A primeira fabricação em série de motores trifásicos assíncronos se iniciou dois anos depois pelo russo, nas potências de 0,4 KW a 7,5 KW, contribuindo para o começo do uso das máquinas elétricas no setor industrial (OLIVEIRA, 2018).

Vários fatores levaram a escolha do MIT, Figura 11, para fornecer a corrente, tensão e temperatura a serem monitoradas neste trabalho. Seguem alguns destes fatores:

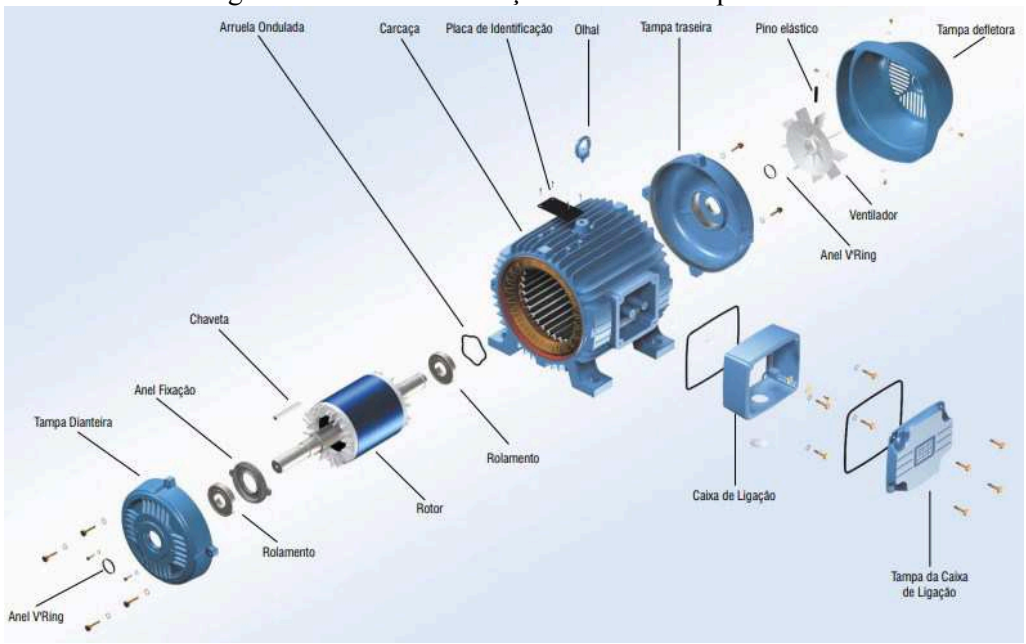
- Em processos industriais é essencial a utilização de motores elétricos, por movimentarem todo tipo de equipamentos e máquinas. Avalia-se que mais de 60% do consumo de energia elétrica encontra-se vinculado a aplicações de força motriz, em países industrializados (ALMEIDA, FERREIRA, *et al.*, 2002).

- O principal equipamento de transformação de energia elétrica em força motriz é o motor de indução trifásico (GODOY, SILVA, *et al.*, 2016).

- Dos motores empregados em processos industriais, 90% são de indução trifásicos, esse fato se deve, principalmente, à relação custo-benefício e robustez mecânica do mesmo, mas a esse tipo de motor estão associadas outras vantagens como: simplicidade de manutenção, versatilidade de aplicações, ampla faixa de potências nominais de operação, construção relativamente simples e adaptável a diferentes situações de carga (HANITSCH, 2002).

- Existem várias classificações para os motores elétricos, a principal está relacionada à alimentação dos motores: acionados por corrente contínua e por corrente alternada. Os de corrente alternada se subdividem em monofásicas, trifásicas e lineares. Os motores de indução trifásicos, ou assíncronos, são amplamente utilizados em aplicações industriais, comerciais e até mesmos residenciais. Com o desenvolvimento de dispositivos conversores de frequência, o emprego desse tipo de motor tende a aumentar, pois torna-se possível a substituição de máquinas de corrente contínua em sistemas onde é necessário o controle de velocidade e/ou posição (FITZGERALD, CHARLES KINGSLEY e UMANS, 2006).

Figura 11 - Motor de indução Trifásico “Explodido”.



Fonte: (CATÁLOGO WEG, 2005)

2.8. Conclusões e Resultados do Capítulo

Após a revisão da literatura, verificou-se que o mundo passou por diversas mudanças, ao que diz respeito às tecnologias envolvidas no setor industrial e na sociedade. Foram várias revoluções até chegarmos no ponto em que estamos, que é a chamada Indústria 4.0. Percebe-se que o Brasil está começando a utilizar dos conceitos e pilares da indústria 4.0, e que o setor industrial brasileiro tem interesse em se modernizar. Também existe um apoio por parte dos órgãos governamentais, porém não sabemos até quando esse apoio será mantido. O que se sabe é que é de suma importância pesquisas nessa área, como a apresentada neste trabalho, para alavancar a quarta revolução industrial aqui no Brasil.

O próximo capítulo irá realizar um comparativo entre plataformas microcontroladas, para definir a mais adequada para se utilizar neste trabalho.

3. DEFINIÇÃODASPLATAFORMAS MICROCONTROLADAS

Neste capítulo foram apresentados os testes comparativos que definiu a plataforma microcontrolada utilizada neste trabalho.

A partir de pesquisas, sabe-se que a velocidade de processamento do microcontrolador, a intensidade do seu sinal sem fio e o consumo são de suma importância para trabalhar com *IoT*, sendo estes os parâmetros utilizados para comparar as plataformas microcontroladas.

Inicialmente foram escolhidas três plataformas microcontroladas para o comparativo, onde o principal quesito foi que a plataforma possuísse *WiFi* integrado, foram escolhidas: MKR1000, ESP32 e ESP8266.

A primeira plataforma é o modelo MKR1000 da Arduino®, cujo *chip* é o ATMEL SAMW25. As outras duas plataformas são da fabricante *Espressif Systems*®, onde a primeira é denominada NodeMCU-32S, a qual utiliza o chip*ESP-WROOM-32*, ou apenas ESP32, e a segunda é o LoLinNodeMCU V3, a qual utiliza o *chipESP8266MOD*, ou ESP8662, conforme Figura 12.

A primeira comparação foi teórica, onde foi confrontado as especificações das três plataformas microcontroladas e seus chips, os resultados estão no tópico 3.2.

Foi realizado o teste de velocidade de processamento, para isso foi desenvolvido um código na IDE da Arduino® e carregado em cada microcontrolador, código este que será explicado no tópico 3.3.

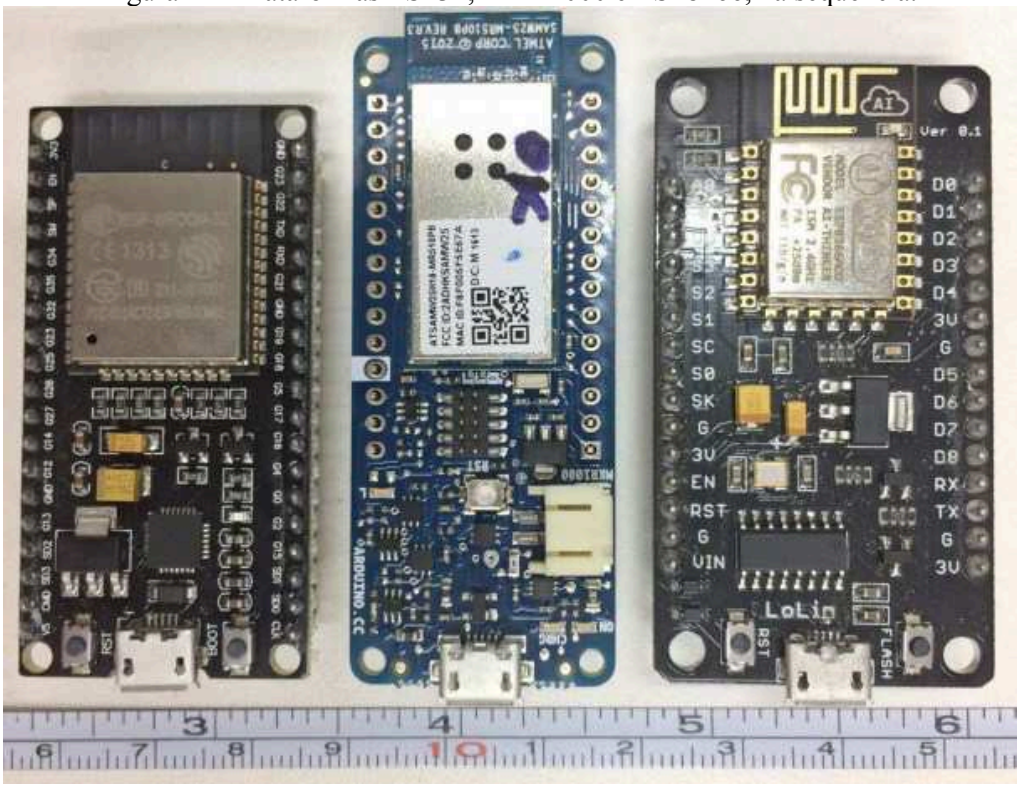
Quanto à comparação da intensidade do sinal, as três plataformas, que possuem *WiFi* integrado, foram programadas como *AccessPoint* para a realização das medições, esse teste será explicado no tópico 3.4.

Nó tópico 3.5 foram realizados os testes de consumo. Para tal foi utilizado um adaptador *USB* (*USB Tester*), que realiza a leitura da tensão e do consumo em amperes da porta *USB* à qual está conectada a plataforma sob teste.

Também foram realizados testes com o sensor *hall* do ESP32, a fim de verificar a possibilidade de utilização desse sensor para adquirir variáveis de campo magnético em motores sem utilização de sensores adicionais, os resultados estão no tópico 3.6.

Além dos testes citados anteriormente, também foi realizado o teste com o sensor de temperatura, interno ao ESP32, os resultados se encontram no tópico 3.7.

Figura 12 - Plataformas ESP32, MKR1000 e ESP8266, na sequência.



Fonte: Própria autoria.

3.1. Instalando o Software de programação

As plataformas microcontroladas utilizadas neste trabalho foram programadas na IDE da Arduino®. Porém, por não se tratar de um software nativo, para utilizar o ESP32 e ESP8266 algumas configurações se tornam necessárias.

Primeiro passo é instalar a IDE da Arduino no computador, sua instalação é padrão, bem simples e intuitiva. O software para download é disponibilizado para Windows, Mac OS X e Linux, de maneira gratuita na página da Arduino (ARDUINO SOFTWARE PAGE, 2019b).

Com a IDE da Arduino instalada no computador, é necessário prepará-la para se tornar compatível com o ESP32, para isso seguiremos algumas etapas que estão demonstradas graficamente no fluxograma da Figura 13.

De maneira parecida ao ESP32, para utilizarmos o ESP8266 no IDE da Arduino® também é necessário realizar uma preparação. O passo-a-passo de todo o processo é demonstrado em (RANDOM NERD TUTORIALS, 2015).

Figura 13 – Fluxograma com etapas para preparação da IDE da Arduino®.

Fonte: (KOYANAGI, 2018)

3.2. Especificações das Plataformas

Foram feitas comparações entre as principais especificações e características de cada plataforma, baseadas nos *datasheets* de seus respectivos fabricantes (ESPRESSIF SYSTEMS, 2019a), (ESPRESSIF SYSTEMS, 2016) (ARDUINO STORE PAGE, 2019c) conforme apresentada na Tabela 1. Já as principais especificações dos microcontroladores utilizados em cada plataforma são apresentadas na Tabela 2, também baseadas nos *datasheets* de seus fabricantes (ESPRESSIF SYSTEMS, 2018) (ESPRESSIF SYSTEMS, 2019b) (ARDUINO UPLOADS, 2016).

Tabela 1 - Comparativo entre plataformas.

Plataformas			
Nome Comercial	ESP32	ESP8266	MKR1000
Prototipagem	NodeMCU-32S	NodeMCU v3 Lolin	Arduino MKR1000
GPIO pinos	32	17	8
FLASH	Típico 16Mb Max 32Mb	Típico 512K Max 16 MB	256 Kb
ADC pinos	16	1	7
DAC pinos	2	0	1
Vcc	3.3V	3.3V	3.3V
Consumo	80 mA	80 mA	20 mA
Dimensões	54,0 x 27,5 mm	58,0 x 31,0 mm	61,5 x 25,0 mm
Preço médio	R$ 42,45	R$ 38,16	R$ 135,12

Fonte: (Espressif, 2019a), (Espressif, 2016) e (Arduino, 2019c)

Tabela 2 - Comparativo entre microcontroladores.

Microcontroladores			
Nome comercial	ESP32	ESP8266	MKR1000
Chip	ESPWROOM-32	ESP8266MOD	ATMEL ATSAMW25
N° Processadores	2	1	1
Processador	Tensilica Xtensa LX6	Tensilica Xtensa LX106	SAMD21 ARM® Cortex -M0+
Arquitetura	32 Bit	32 Bit	32 Bit
CPU Freq. (Max)	240 MHz	160 MHz	48 MHz
Bluetooth	sim	não	não
SRAM	520 Kb	50 Kb	32 Kb
FLASH	4 MB	4 MB	256 Kb
GPIO pinos	36	17	8
ADC pinos	16	1	7
DAC pinos	2	-	1
ADC Resolução	12 bits	10 bits	12 bits
DAC Resolução	8 bits	-	10 bits
Interfaces	SPI,I2C,UART,I2S,CAN	SPI,I2C,UART,I2S	SPI, I2C, UART
Sensor Touch	10	-	-
Sensor Hall	1	-	-
Sensor Temperatura	1	-	-

Fonte: (Espressif, 2018), (Espressif, 2019b) e (Arduino, 2016).

Na internet são encontradas muitas informações sobre as especificações das plataformas, porém algumas não batem devido ao fato de existirem plataformas diferentes com os mesmos chips. Desta forma, é bom deixar claro que o comparativo acima se refere às plataformas apresentadas neste capítulo.

Com base nesse comparativo, foi verificado que o microcontrolador ESP32 é superior em vários quesitos, além de possuir uma maior frequência de *clock*, entre os mais notórios é que ele possui dois processadores, enquanto o ESP8266 e o MKR1000 possuem apenas um. Em poucas categorias o ESP8266 e o MKR1000 são similares ao ESP32. Em termos gerais, o ESP8266 sobressai frente ao MKR1000, assim, pode-se inferir que, para a aplicação que o trabalho propõe, as plataformas da *Espressif Systems*® são superiores às plataformas da Arduino®.

Por outro lado, o MKR1000 tem a indicação de menor consumo de energia, o que será verificado em testes práticos nesse trabalho, no tópico 3.5

Além disso, o ESP32 possui sensores *on-chip* que os demais não possuem, como um sensor *hall*, um sensor de temperatura e dez entradas para sensores de toque (*touch)*.

Para a aplicação em questão, até o momento o ESP32 mostrou-se mais promissor, no entanto foram realizados alguns testes para comprovar a escolha.

3.3. Teste de velocidade de processamento

O objetivo deste teste foi calcular o tempo de resposta de cada microcontrolador, executando códigos com a mesma finalidade. É importante salientar que os microcontroladores possuem arquiteturas diferentes, e com isso bibliotecas e funções diferentes, sendo assim foram desenvolvidos programas distintos, porém semelhantes em relação aos objetivos a serem alcançados.

Contudo, foi utilizado o mesmo software de desenvolvimento, Arduino IDE, para todos os microcontroladores, a fim de obter uma comparação mais coerente. Inclusive é importante salientar que foi utilizado apenas um núcleo do ESP32.

O código realiza os seguintes comandos: primeiramente conecta o microcontrolador à rede sem fio, depois realizada a leitura de um sensor de temperatura externo (LM35), do fabricante *Texas Instruments* (TEXAS INSTRUMENTS, 2017) pela entrada analógica do microcontrolador.

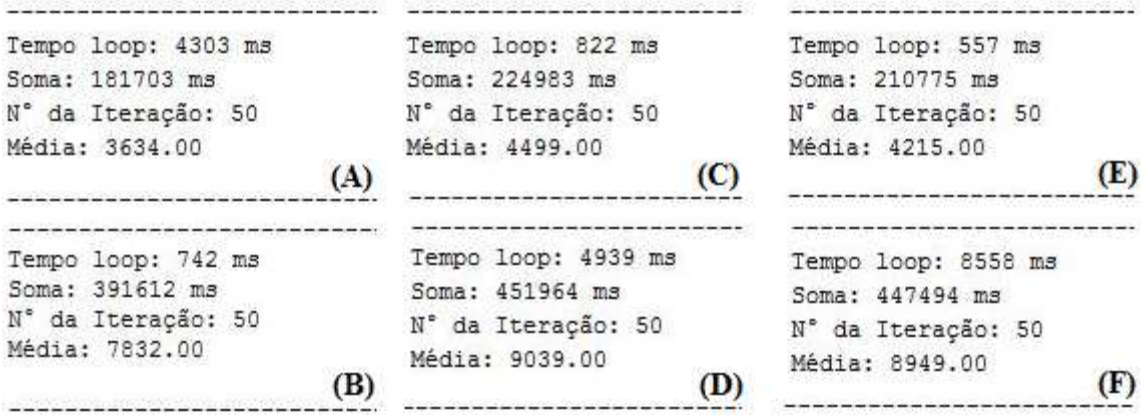
Dentro de um "*loop"* é realizado a função "*if*" e de acordo com os níveis de temperaturas pré-estabelecidos (alta ou muito alta) o microcontrolador irá enviar mensagens via *Telegram*®

para um *bot* (abreviação de "*robot*") cadastrado em um *smartphone* e também irá disponibilizar os dados em forma de gráfico na plataforma *ThingSpeak* (ThingSpeak, 2019), que pode ser acessada via navegador *Web*. Tanto o *Telegram*, quanto o *ThingSpeak* são plataformas gratuitas que salvam os dados em nuvem. Dentre outras funções do código, como reconhecimento de alarmes, ele também executa um código para buscar a informação do horário via protocolo de sincronização de relógios NTP (NTP, 2019). Ao final, o código mede o tempo em milissegundos que cada microcontrolador necessitou para executar todas as iterações e comandos dentro de um *loop*. É importante observar que foi utilizado no código um *delay* de 500 milissegundos.

Os tempos medidos, Figura 14, foram inseridos na Tabela 3, onde consta o tempo medido com o programa sendo executado sem nenhuma iteração e os tempos medidos com o programa realizando iterações. Foram executadas por volta de 40 iterações (envio e recebimento de mensagens via Telegram), onde aproximadamente 30 mensagens foram enviadas ao Telegram pelo microcontrolador e 10 mensagens recebidas no microcontrolador e enviadas via Telegram.

Neste teste não foi possível monitorar outras variáveis analógicas, como corrente e tensão, pois o ESP8266 possui apenas uma entrada analógica, causando uma limitação.

Figura 14 – (a) ESP32 sem iteração (b) ESP32 com iteração (c) ESP8266 sem iteração (d) ESP8266 com iteração (e) MKR1000 sem iteração (f) MKR1000 com iteração



Fonte: Própria autoria.

Tabela 3- Tempo dos microcontroladores.

	Tempo gasto (ms)		
Plataforma	**ESP32**	**ESP8266**	**MKR1000**
Sem Iterações	3634	4499	4215
Com Iterações	7832	9039	8949

Fonte: Própria autoria.

3.4. Testes de intensidade do sinal *WiFi*

Os objetivos destes testes foram: medir a intensidade do sinal das três plataformas que possuem *WiFi* integrado; verificar a intensidade do sinal de cada um deles em diferentes condições; e fazer uma análise comparativa após o levantamento dos dados. Para isso, os três microcontroladores foram programados com a função de *Access Point* (Ponto de Acesso), cada um transmitindo em um canal diferente para evitar interferências, ligados de maneira simultânea e realizando as medições. A primeira medição foi feita sem obstáculos a 2 metros de distância das plataformas em um ambiente interno, onde foi registrado na Figura 15(a) as amplitudes dos sinais em certo instante de tempo, e na Figura 15(b) as oscilações dos sinais durante um intervalo de tempo de aproximadamente 3 minutos.

A segunda medição seguiu o mesmo modelo da primeira, porém agora a uma distância de 10 metros das plataformas, com uma parede como obstáculo, a medição foi realizada com a plataforma em um ambiente interno e o *smartphone* em um ambiente externo, onde se registrou na Figura 16(a) as amplitudes dos sinais em certo instante de tempo e na Figura 16(b) as oscilações dos sinais durante um intervalo de tempo de aproximadamente 3 minutos.

Para a realização dos testes foi utilizado o aplicativo para *smartphone*, *WiFiAnalyzer*, que apresenta a intensidade do sinal em *dBm* (unidade de medida de intensidade do sinal). É importante observar que, quanto menor o valor encontrado em *dBm*, maior a potência do sinal.

Cada categoria de teste foi repetida em média dez vezes, a fim de se verificar a constância dos resultados obtidos. A seguir são relatados, os pontos levantados sobre cada plataforma:

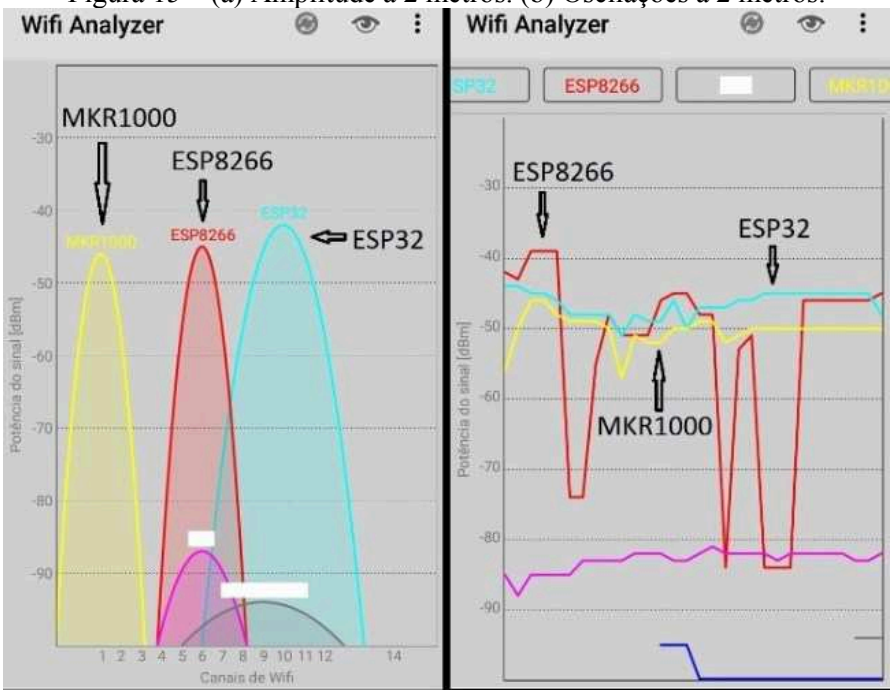
- No teste a 2 metros de distância, o microcontrolador ESP32 demonstrou melhores resultados, sem oscilações e apresentou um nível de amplitude de sinal superior aos demais.

- No teste a 10 metros e com obstáculos, o ESP32 foi ligeiramente inferior ao MRK1000, mantendo a constância e uma amplitude satisfatória do sinal.

- O MKR1000, no teste a 10 metros de distância e com obstáculo, obteve os melhores resultados, não apresentando muitas oscilações, além de obter um nível de amplitude superior aos demais. No teste a 2 metros e com obstáculos, apresentou amplitude de sinal próxima àquela do ESP32, mantendo uma constância e amplitude desejáveis.

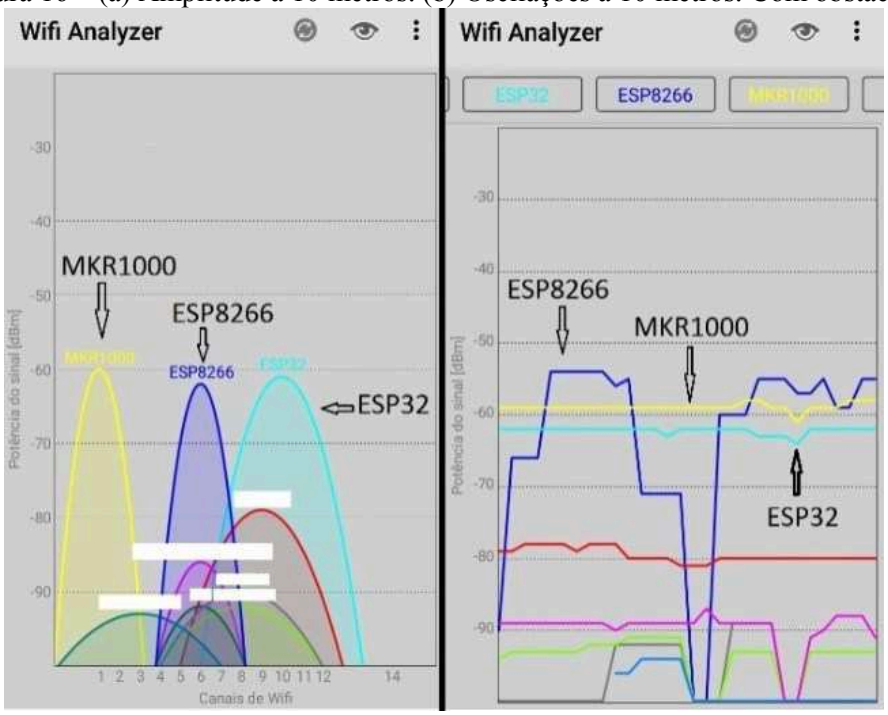
- Quanto ao ESP8266, este apresentou comportamento semelhante aos demais, tanto a 2 metros quanto a 10 metros de distância com obstáculo, observando que a 10 metros a potência do sinal era naturalmente menor. Também apresentou resultados adequados no teste de amplitude, porém apresentou resultados bastante insatisfatórios na constância do sinal, devido a grandes oscilações de amplitude.

Figura 15 – (a) Amplitude a 2 metros. (b) Oscilações a 2 metros.



Fonte: Própria autoria.

Figura 16 – (a) Amplitude a 10 metros. (b) Oscilações a 10 metros. Com obstáculo.



Fonte: Própria autoria.

Resumindo, neste teste de intensidade do sinal o ESP32 e o MKR1000 apresentaram bons resultados. Já o ESP8266 apresentou grandes oscilações na intensidade do seu sinal, comportamento não desejado para a aplicação em questão.

3.5. Testes de consumo

Com o intuito de verificar os dados de consumo inseridos na Tabela 1, foram realizados testes para verificação do consumo das plataformas, sob as mesmas condições de uso, utilizando um dispositivo (USB Tester) que mede a tensão e a corrente da porta USB à qual as plataformas foram conectadas. Para medir o consumo de cada plataforma, os três microcontroladores foram programados com a função de Access Point e realizada a leitura de consumo, Figura 17.

No teste para verificação do consumo, utilizando o USB Tester, as plataformas MKR1000 e ESP8266 apresentaram a mesma leitura de corrente, 50 mA. Já a plataforma ESP32 apresentou uma leitura mais alta, 110 mA.

Esses valores não coincidem exatamente com os valores dos *datasheets*, já que o consumo de cada plataforma depende do programa que está senso executado. Os resultados observados mostram que, programados como *Acces Point*, o MKR1000 e o ESP8266 apresentam menor consumo, enquanto que o ESP32 apresenta um maior consumo, condizente com a informação do *datasheet* deste último.

Figura 17 – Teste de consumo utilizando o USB Tester.

Fonte: Própria autoria.

3.6. Testes sensor hall ESP32

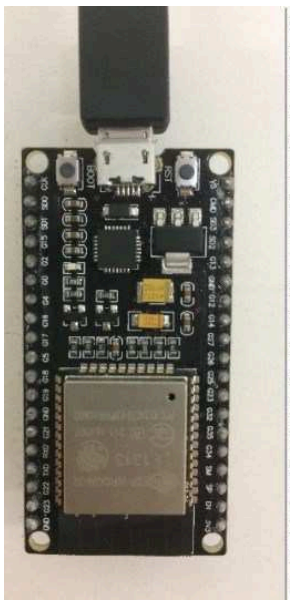
Este teste foi realizado somente na plataforma ESP32, pois é a única entre as três que possui sensor *hall*. A finalidade do teste foi verificar se esse sensor seria capaz que captar campos magnéticos gerados durante o funcionamento do MIT.

O sensor hall é um dispositivo que, quando sob o efeito de um campo magnético, produz uma tensão proporcional à intensidade do campo (HONEYWELL, s.d.).

Foi utilizado o *software* Arduino IDE para carregar o programa que faz a leitura do sensor hall, no ESP32. O programa executa a leitura do sensor, através da função *hallRead(),* e envia os dados para o monitor serial da própria IDE. Com esse programa em funcionamento foram realizados vários testes para verificar como o sensor *hall* do ESP32 se comporta na presença de campos magnéticos, inclusive se consegue captar interferências de campos magnéticos de motores. Para este teste foi utilizado um intervalo de 1 segundo entre medições.

As primeiras leituras foram realizadas sem a aplicação de campo magnético ao ESP32, e verificou-se que o sensor capta um pequeno valor, porém esse valor pode ser desprezado. Os resultados dessa leitura podem ser observados na Figura 18. Em seguida foi aplicado um campo magnético ao sensor *hall*, utilizando um imã permanente. Ao se aproximar do sensor, o polo norte do imã, nota-se que os valores das leituras variam positivamente. Ao se aproximar o polo sul do imã, nota-se que os valores das leituras se tornam negativos. Em ambos os casos um maior valor em módulo representa um maior valor de campo aplicado, conforme Figura 19.

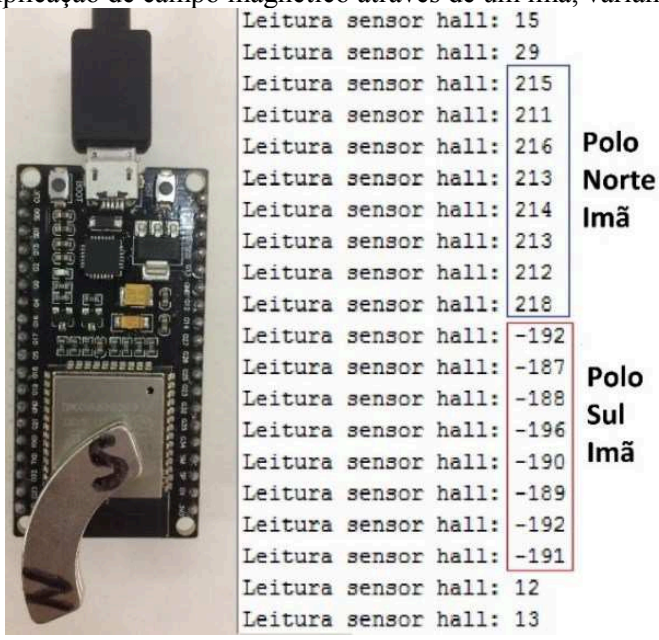
Figura 18 – Leituras do sensor hall sem a aplicação de campo magnético.



```
Leitura sensor hall: 10
Leitura sensor hall: 12
Leitura sensor hall: 11
Leitura sensor hall: 9
Leitura sensor hall: 10
Leitura sensor hall: 10
Leitura sensor hall: 12
Leitura sensor hall: 16
Leitura sensor hall: 14
Leitura sensor hall: 11
Leitura sensor hall: 15
Leitura sensor hall: 16
Leitura sensor hall: 11
Leitura sensor hall: 14
Leitura sensor hall: 12
Leitura sensor hall: 13
Leitura sensor hall: 12
Leitura sensor hall: 10
Leitura sensor hall: 12
Leitura sensor hall: 16
```

Fonte: Própria autoria.

Figura 19 – Aplicação de campo magnético através de um imã, variando seus polos.



Fonte: Própria autoria.

Um último teste foi realizado para verificar a interferência do campo magnético de motores sobre a leitura do ESP32. Foi constatado que, mesmo com o acionamento do motor sob a plataforma de desenvolvimento, as leituras de campo magnético não foram alteradas. Desta forma foi verificado que não houve interferência dos motores na leitura do campo magnético. Logo, para ocorrer uma variação na leitura do sensor *hall*, um campo magnético tem que ser aplicado diretamente sobre o mesmo, e caso seja necessário a leitura de campo magnético gerado pelo motor é necessário um sensor externo.

3.7. Testes sensor de temperatura ESP32

Este teste também foi realizado somente na plataforma ESP32, pois é a única que possui um sensor de temperatura *on-chip*, das três plataformas em questão.

Para realizar a leitura do sensor de temperatura do ESP32, utiliza-se a função, *temprature_sens_read()* e os resultados podem ser observados no monitor serial. Assim foram realizados alguns testes, para verificar as leituras do sensor de temperatura em diferentes condições.

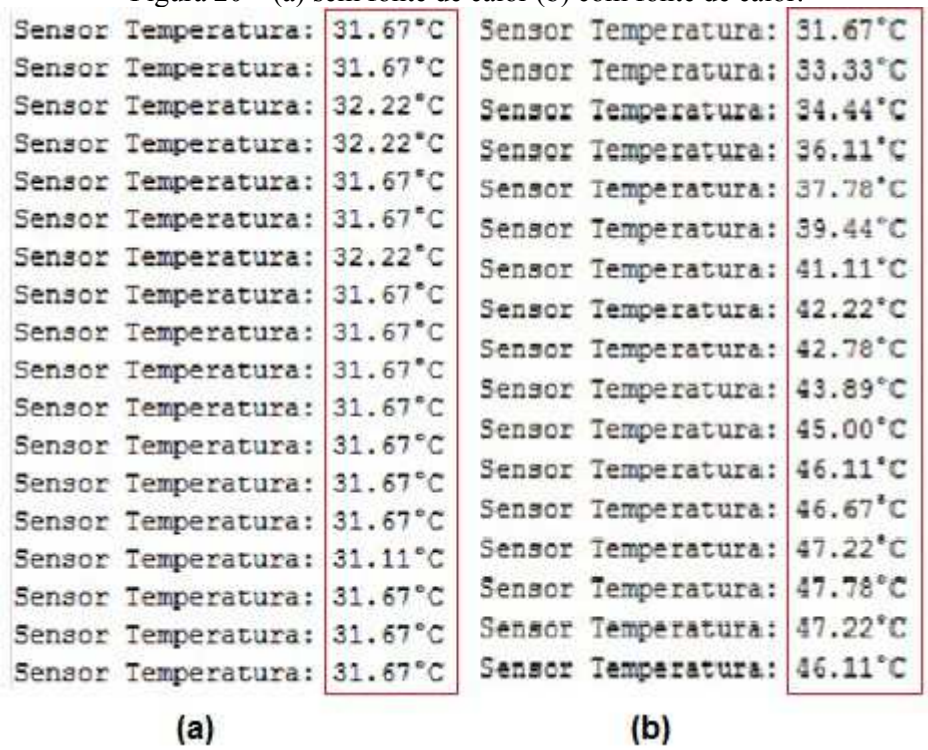
O sensor de temperatura do ESP32 realiza a leitura em Fahrenheit, sendo necessária a conversão para graus Celsius. Entretanto foi observado que o valor medido não condiz com o valor real, que foi medido utilizando um multímetro. A diferença entre as medições deve-se ao fato do sensor do ESP32 não vir calibrado. Assim, deve-se fazer a calibração do sensor de temperatura para poder utilizá-lo corretamente.

Após a calibração do sensor foi realizado um teste sem a aplicação de fonte de calor, cujos resultados podem ser observados na Figura 20(a), sendo a temperatura apresentada superior à temperatura ambiente e correspondendo à temperatura interna do ESP32.

Em seguida foi aplicar a fonte de calor ao ESP32, direcionada ao sensor de temperatura *on-chip*, notando-se um aumento gradativo da temperatura. Para este teste foi utilizado um intervalo de 4 segundos entre medições, Figura 20(b).

Dessa maneira verificou-se que o sensor de temperatura funciona, porém é necessário aplicar uma fonte de calor bem próxima ao chip da plataforma microcontrolada.

Figura 20 – (a) sem fonte de calor (b) com fonte de calor.



Fonte: Própria autoria.

3.8. Conclusões e Resultados do Capítulo

Baseado nos resultados dos testes e análises feitas entre as três plataformas *WiFi* (ESP32, ESP8266 e MKR100), o ESP32 mostrou-se mais promissor para ser utilizado em aplicações com *IoT*. Além de possuir melhores especificações na maioria dos quesitos, este apresentou melhor desempenho no teste de velocidade, mesmo utilizando apenas um processador, e demonstrou ótimos resultados no teste de intensidade do sinal, semelhante aos resultados do MKR1000, sendo este último inferior nos testes de velocidade de processamento e nos recursos disponíveis na plataforma.

Os testes com o sensor hall e de temperatura do ESP32 não apresentaram resultados satisfatórios que justifiquem sua utilização para adquirir variáveis de campo magnético e temperatura de motores. Assim, para a leitura de campo magnético e temperatura no monitoramento destas variáveis em motores deve-se utilizar sensores externos.

Em se tratando de disponibilidades de informações técnicas, o MKR1000 apresenta um número maior de fontes para pesquisa, além de maior facilidade de programação. Já as informações das plataformas da *Espressif Systems* são mais escassas e divergentes, dificultando a busca por informações sobre os recursos disponíveis para essas plataformas.

Assim, conclui-se que a partir dos testes e estudos realizados a plataforma microcontrolada mais indicada ao presente trabalho é o ESP32, pois apresentou um conjunto satisfatório de recursos e um ótimo desempenho relativo para uso em aplicações de *IoT*.

Novas plataformas e microcontroladores são desenvolvidas a cada ano, e novos testes podem ampliar as comparações obtidas nesse trabalho.

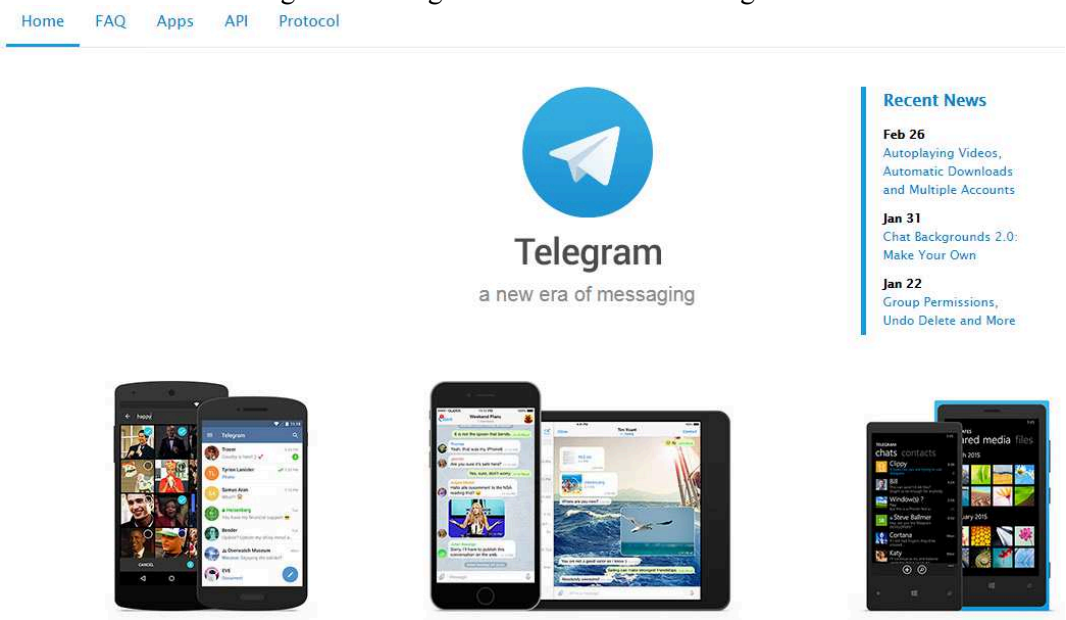
4. PLATAFORMAS ARMAZENAMENTO DE DADOS EM NUVEM

Dentre as diversas plataformas disponíveis, se optou por trabalhar com a plataforma Telegram e a ThingSpeak. Um dos principais motivos da escolha se dá por elas serem plataformas grátis. Neste capítulo foi explicado o passo-a-passo para instalação, configuração e utilização das mesmas, assim como as particularidades de cada plataforma.

4.1. Telegram

O Telegram é um serviço de mensagens instantâneas baseado na nuvem. Ele foi fundado em 2013, pelos irmãos russos Nikolai e Pavel Durov, atualmente sua base está em Dubai, Emirados Árabes Unidos. O aplicativo está disponível para smartphones ou tablets (Android, iOS, Windows Phone, Ubuntu Touch, Firefox OS), computadores (Windows, OS X, Linux) e também como Aplicação web. Os usuários podem enviar mensagens e trocar fotos, vídeos, stickers e arquivos de qualquer tipo. O Telegram também possui criptografia ponta-a-ponta opcional. Os clientes do Telegram possuem código aberto, porém seus servidores são proprietários. O serviço também providencia APIs para desenvolvedores independentes, ou seja, possui ferramentas que facilitam a construção de sistemas com *IoT*, possibilitando um controle por troca de mensagens entre plataformas móveis, como tablets e/ou smartphones, e plataformas microcontroladas com por exemplo Node MCU ESP32 (TELEGRAM HOME PAGE, 2019a).

Figura 21 – Página inicial do site do Telegram.



Fonte: (TELEGRAM HOME PAGE, 2019a)

4.1.1. Configurando o BOT do Telegram

A palavra Bot é o diminutivo de *robot*, que também é conhecido como *internet bot* ou *web robot*. Criada para simular ações humanas repetidas vezes de maneira padrão, essa aplicação de software atua da mesma maneira que faria um robô.

Esse aplicativo disponibiliza a seus usuários uma inteligência artificial (IA) capaz de auxiliá-los na construção de seu próprio robô virtual (Bot), que na aplicação desse trabalho será utilizado para enviar mensagens a usuários cadastrados com alertas de sobrecorrente, sobretensão e temperatura alta.

Abaixo segue passo a passo para a criação do *chatBot*, após ter instalado o aplicativo em seu smartphone:

- Conforme Figura 22, abra a tela inicial do aplicativo e pesquise por: @BotFather.

Figura 22 – BotFather.

Fonte: (TELEGRAM, 2019b)

O comando irá direcionar a um chat particular com o "*Father*", o robô virtual do Telegram, que apresentará diversas ferramentas e funções exclusivas.

- Clique em "Começar".

- Digite /*help* se precisar de ajuda. Todos os comandos do BOT devem começar com uma / “barra”. Veja alguns dos principais comandos na Figura 23.

- Para criar um novo *BOT*, digite o comando /*newbot*.

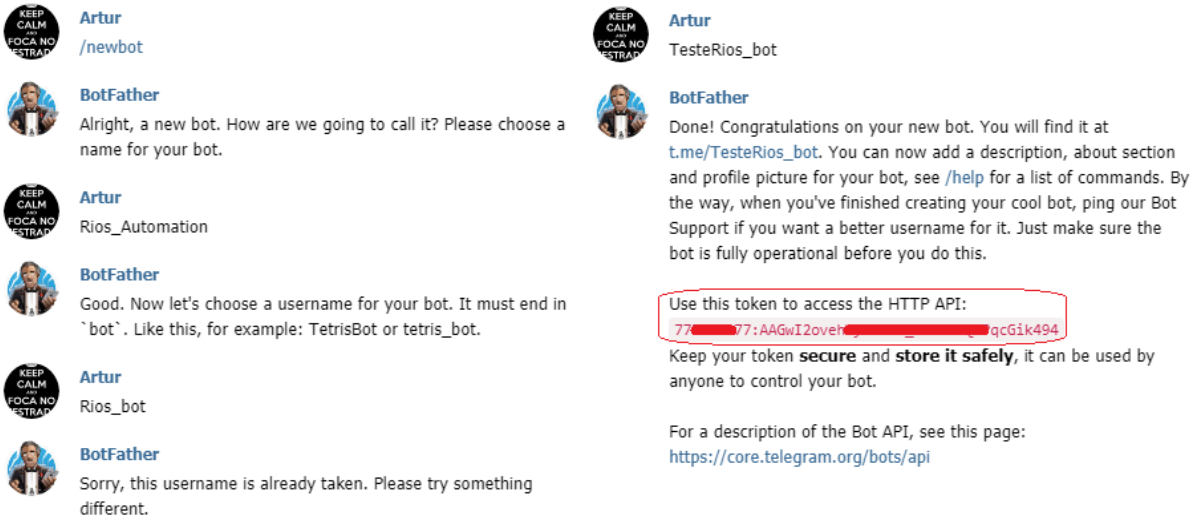
Como resposta, o *Father* irá lhe solicitar um nome para o novo *Bot*, e um usuário finalizado com a palavra "*bot*", este será utilizado como forma de localizá-lo na barra de pesquisa.

Na mensagem de confirmação da construção do novo *bot*, será enviado um Token de acesso, uma chave de API composta por diversas letras e números, com a função de estabelecer conexão entre seu chat particular no Telegram e o ESP32. Segue exemplo na Figura 24.

Figura 23 – Comandos do Telegram.

Fonte: Própria autoria.

Figura 24 – Exemplo de criação de um bot e de um usuário.

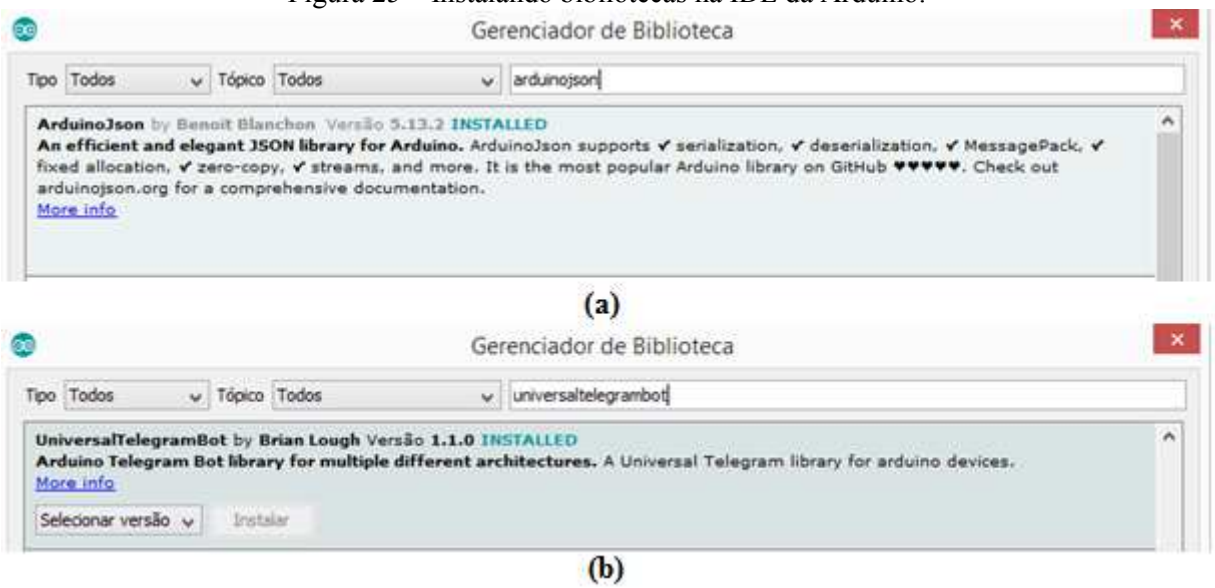


Fonte: Própria autoria.

4.1.2. Bibliotecas necessárias ao Telegram

No momento de programar o ESP32, na IDE da Arduino, para comunicar com o Telegram é necessário instalar algumas bibliotecas, além das bibliotecas padrão, são elas: ArduinoJson e Universal Arduino Telegram Bot, conforme Figura 25(a) e Figura 25(b) e, respectivamente.

Figura 25 – Instalando bibliotecas na IDE da Arduino.



4.1.3. Teste do Telegram

Com o software Arduino IDE preparado e o *bot* do Telegram criado, foi elaborado um código para realizar a comunicação entre o ESP32 e o Telegram.

Alguns pontos importantes para o funcionamento do código são:

- Declarar as bibliotecas corretamente.
- Definir a chave “BOT_TOKEN” e número do “CHAT_ID”.
- Ter conexão com a internet.

Foram testados vários códigos mais simples antes de chegar ao código final do trabalho, porém não é o intuito demonstrá-los aqui, pois os parâmetros do Telegram estão presentes e identificados no código responsável pela transmissão dos dados para a internet, que se encontra no APÊNDICE B.

Pode-se concluir a partir desse teste que a comunicação entre o ESP32 e o Telegram ocorreu da maneira esperada.

4.2. ThingSpeak

O ThingSpeak® é uma plataforma analítica *open source* que fornece vários serviços exclusivamente direcionados para projetos que envolvam *IoT*, que permite agregar, visualizar e analisar fluxos de dados, em forma de gráficos, salvos em nuvem em tempo real, conforme descreve sua *homepage*, Figura 26 (THINGSPEAK HOME PAGE, 2019a).

Figura 26 – Home page site ThingSpeak.

Fonte: (THINGSPEAK HOME PAGE, 2019a)

O ThingSpeak permite visualizações instantâneas via HTTP de dados postados na plataforma pelos mais diversos dispositivos. Com a capacidade de executar o código MATLAB® no ThingSpeak, é possível realizar a análise e o processamento *online* dos dados conforme eles são recebidos, porém também é possível receber códigos advindos de outras plataformas e *softwares*. O ThingSpeak é frequentemente usado para prototipagem e sistemas *IoT* de verificação de conceito que exigem análise (THINGSPEAK LEARN MORE, 2019b).

O ThingSpeak foi originalmente lançado pela *ioBridge*[1] em 2010 como um serviço de suporte a aplicativos *IoT*. Integrou o suporte do *software* de computação numérica MATLAB da MathWorks, permitindo que os usuários do ThingSpeak analisassem e visualizassem os dados carregados usando o Matlab sem a necessidade da compra de uma licença.

[1]A *ioBridge* é fabricante de hardware de monitoração e controle baseado em internet e fornecedora de serviços baseados em nuvem e serviços de API online. <https://www.iobridge.com/> Acessado em 08 de Maio de 2019.

A plataforma é vinculada a Mathworks, com isso toda a documentação do ThingSpeak é incorporada no site de documentação Mathworks, e até mesmo permite que as contas de usuário registradas da Mathworks sejam credenciais de *login* válidas no site ThingSpeak (MATHWORKS, 2019).

O ThingSpeak tem sido utilizado em trabalhos especializados de "*maker*", que utilizam o "Canal ThingSpeak". Esse canal armazena os dados que enviamos para o ThingSpeak e compreende os elementos abaixo:

- 8 campos para armazenar dados de qualquer tipo - Pode ser usados para armazenar os dados de um sensor ou de um dispositivo incorporado.
- 3 campos de localização - Pode ser usados para armazenar a latitude, longitude e a elevação. Estes são muito úteis para rastrear um dispositivo em movimento.
- 1 campo de status - Uma mensagem curta para descrever os dados armazenados no canal.

Para usar o ThingSpeak é necessário se inscrever e criar um canal. Como canal criado é possível enviar os dados, onde os mesmos serão processados pelo ThingSpeak e disponibilizados para visualização em forma de gráficos. A licença gratuita disponibiliza até 4 canais (THINGSPEAK LEARN MORE, 2019b).

O upload de dados se dá através de requisições HTTP/HTTPS contendo os dados, canais onde devem ser escritos e uma chave de autenticação de escrita do canal (LEMOS, BARBOSA, *et al.*, 2016).

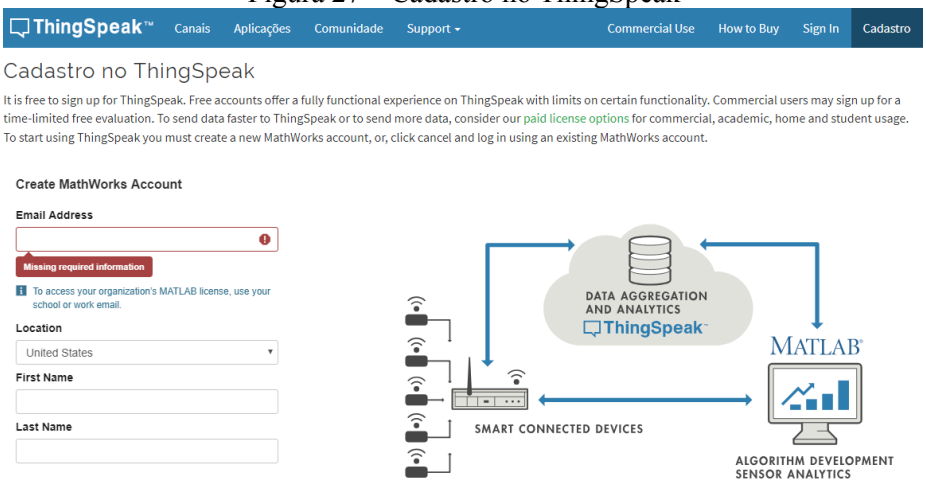
4.2.1. Criando um canal ThingSpeak

Primeiro passo pra utilizar o ThingSpeak é se cadastrar no site, conforme Figura 27. Com o cadastro feito deve-se *logar* no site e clicar na aba “Canais”, “Meus Canais”, “*NewChanel*” e configure o nome e os campos do seu canal, conforme Figura 28.

Após criado o canal, clicando na aba “*Sharing*” é possível mantê-lo privado, torná-lo público ou compartilhá-lo com usuários específicos.

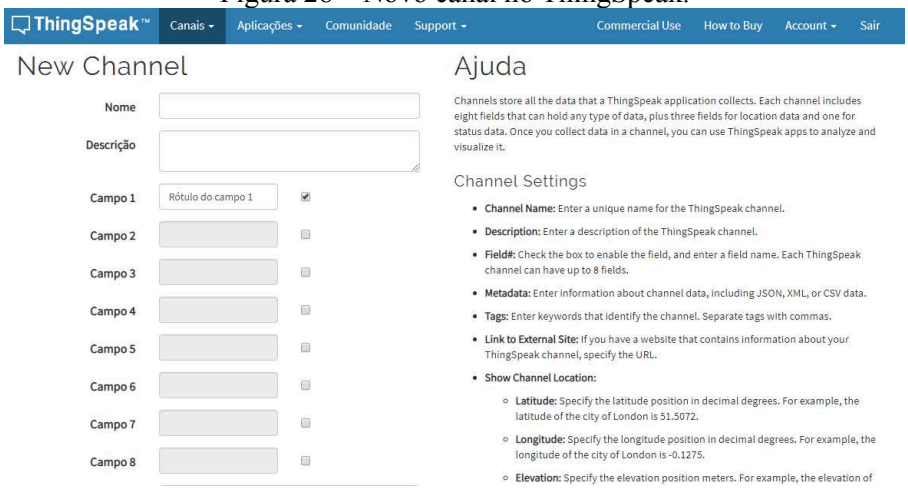
Na aba “Chaves” ficam disponíveis as chaves de escrita e leitura, essas chaves são exclusivas de cada canal. Neste projeto a chave de escrita será utilizada no código do ESP32, para realizar a comunicação entre a plataforma microcontrolada e o ThingSpeak.

Figura 27 – Cadastro no ThingSpeak



Fonte: (THINGSPEAK, 2019c)

Figura 28 – Novo canal no ThingSpeak.



Fonte: (THINGSPEAK, 2019d)

4.2.2. Teste ThingSpeak

De maneira análoga ao Telegram, também foram testados vários códigos antes de chegar ao código final do programa, disponível no APÊNDICE B. Esse código está com suas linhas de programação comentadas, para um melhor entendimento do funcionamento.

Seguem alguns pontos importantes para que a comunicação entre o ESP32 e o ThingSpeak funcione corretamente:

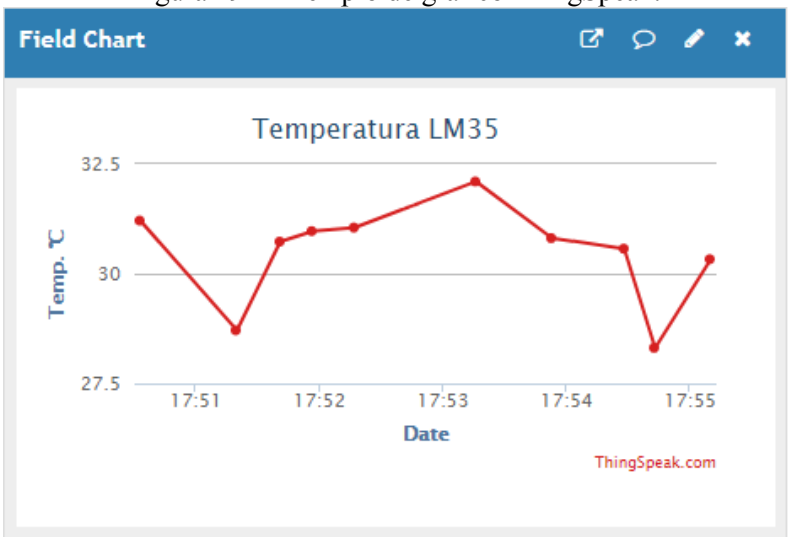
- Declarar as bibliotecas corretamente;
- Definir a chave "*writeAPIKey*" e o endereço de servidor.
- Ter conexão com a internet.

Com os testes realizados também se conclui que a comunicação entre o ESP32 e o ThingSpeak funcionam corretamente.

Um ponto importante a ser informado sobre o ThingSpeak é que seus gráficos são atualizados automaticamente a cada quinze segundos.

Segue na Figura 29um exemplo de gráfico gerado pelo ThingSpeak, onde foi medido a temperatura a partir de um sensor LM35.

Figura 29 – Exemplo de gráfico ThingSpeak.



Fonte: Própria autoria.

4.3. Conclusões e Resultados do Capítulo

Neste capítulo foi definida a plataforma para armazenamento e transmissão dos dados em nuvem (Telegram e ThingSpeak). Verificou-se que são plataformas de fácil obtenção, instalação, configuração e uso. Encontram-se disponíveis para diversos sistemas operacionais e possuem uma interface intuitiva de fácil utilização.

No próximo capítulo será apresentada a bancada didática com o motor de indução trifásico utilizado neste trabalho, assim como os sensores utilizados para monitoramento e aquisição de dados, como a corrente, tensão e temperatura.

5. AQUISIÇÃO DE CORRENTE, TENSÃO E TEMPERATURA

Neste capítulo foi apresentada a Bancada Didática da WEG que contém o motor de indução trifásico utilizado neste trabalho, os sensores para a aquisição e condicionamento de corrente, tensão e temperatura, e os critérios de alarmes.

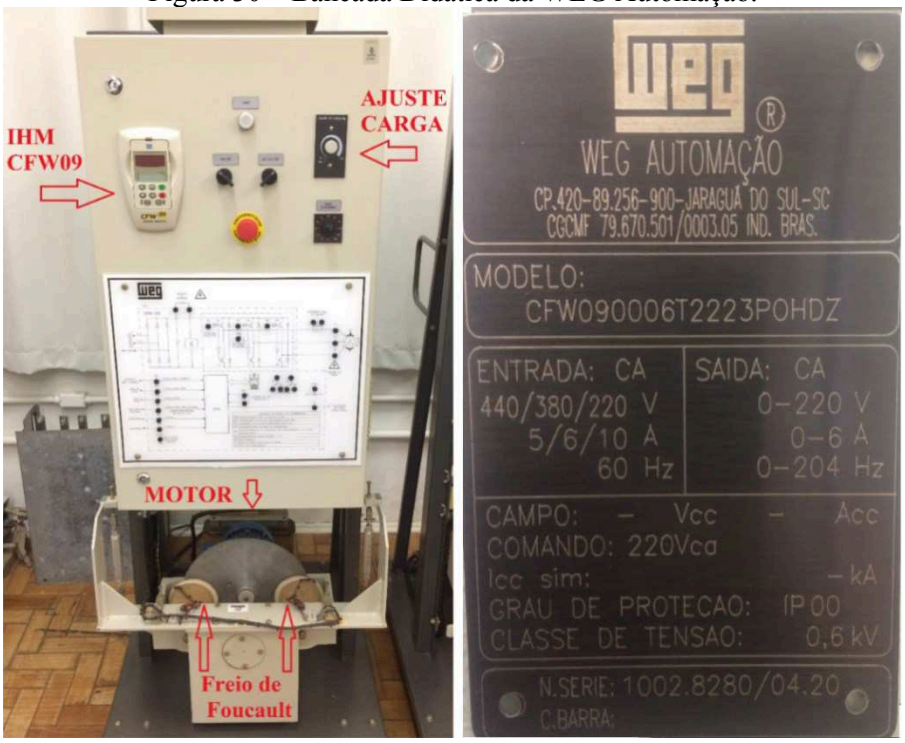
5.1. Bancada Didática WEG

Foi utilizada no projeto a bancada didática da WEG Automação, que fica localizada no Laboratório de Acionamentos do CEFET-MG campus Araxá, modelo CFW090006T2223POHDZ, conforme Figura 30.

A bancada é composta por um motor de indução trifásico de corrente alternada tipo gaiola da fabricante WEG, Figura 31, com as seguintes características:

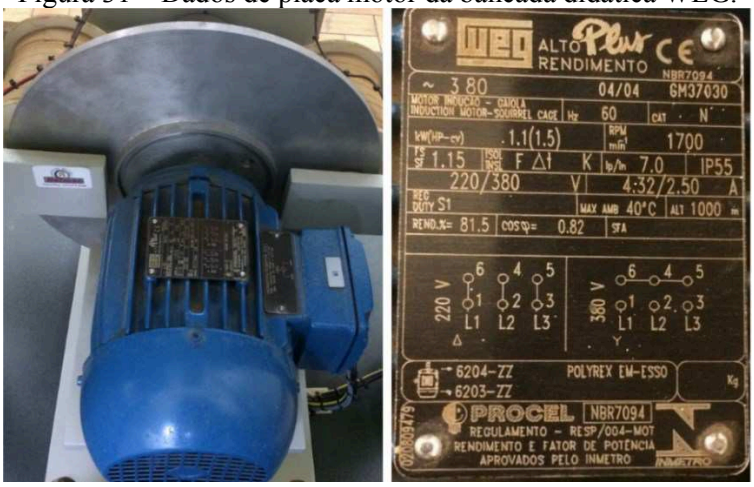
- Modelo Alto Rendimento Plus | Categoria N | IP55
- 1,5 CV | 1700 RPM | FS 1,15
- 220/380 V | 4,32/2,50 A | Ip/In 7,0
- Rend. 81,5% | Cosφ 0,82

Figura 30 – Bancada Didática da WEG Automação.



Fonte: Própria autoria.

Figura 31 – Dados de placa motor da bancada didática WEG.



Fonte: Própria autoria.

A bancada também possui um inversor CFW09 da WEG para acionamento, controle de velocidade de rotação do eixo, monitoramento e proteção do motor (WEG CFW09, 2012).

Esse inversor possui cinco tipos de controle, configuráveis pelo parâmetro P202, conforme Figura 32.Para este trabalho, o mesmo foi configurado para trabalhar com controle escalar V/F (Tensão/Frequência), ou seja, código 0 (V/F 60Hz) de acordo com a Figura 33. Nesse modo de controle, varia-se a relação V/F de acordo com valores predefinidos, possibilitando uma partida suave em rampa de tensão.

Figura 32 – Parâmetro para definir tipo de controle do CFW09.

Fonte: (WEG CFW09, 2006)

Figura 33 – Inversor configurado para controle V/F 60Hz.

Fonte: Própria autoria.

A bancada também possui um ajuste de carga que aumenta e diminui a carga do motor utilizando os Freios de Foucault.

Freio de Foucault tem como premissa básica, a utilização de um disco de alumínio para interpolar uma resistência ao torque de máquinas rotativas com o objetivo de reduzir sua velocidade, simular uma variação de carga acoplada ao eixo, ou mesmo suavizar a frenagem no momento do desligamento, dissipando sua energia inercial na forma de efeito Joule, pela ação das correntes parasitas circulantes no disco (PINHEIRO FILHO, 2014).

5.2. Sensores

Neste tópico estão definidos os tipos de sensores, descrevendo o princípio de funcionamento de cada um, para medir corrente, tensão e temperatura.

5.2.1. Sensores de Corrente e Tensão

Para realizar as leituras de corrente e tensão do motor de indução trifásico foi utilizado o "Módulo de Aquisição e Condicionamento de sinais de Tensão e Corrente", conforme Figura 34. Esse módulo foi desenvolvido por Marcelo Rodrigues S. Brito, Henrique José Avelar e Ernane A.A. Coelho no NUPEP-UFU (Núcleo de Pesquisa em Eletrônica de Potência da Universidade Federal de Uberlândia).

Figura 34 – Módulo Aquisição/Condicionamento de sinais de Tensão e Corrente.

Fonte: Guia de uso sensores Hall (NUPEP-UFU, 2011)

A função do módulo é a leitura, utilizando sensores hall, e conversão de sinais de corrente e tensão para níveis baixos de tensão (0 V – 3,3 V), filtrar altas frequências com o intuito de evitar ruídos e efeitos de *aliasing*. Dessa maneira é possível que os sinais possam ser lidos por um microcontrolador (NUPEP, 2011).

O módulo de aquisição e condicionamento de sinais de tensão e corrente pode ser dividido em três circuito básicos:

1° Circuito–Filtro e Regulação da Alimentação: Nesse estágio há uma separação entre as alimentações dos AOs (Amplificadores Operacionais) do estágio de filtro de tensão com o de corrente, assim havendo um desacoplamento para evitar interferência entre sinais. Como uma parte anexa a este estágio, existe um divisor de tensão, cuja função é fornecer um nível de tensão que será somado ao sinal para criar um offset e elevar o zero do sinal (NUPEP, 2011).

2° Circuito– Tratamento Inicial do Sinal: Nessa etapa existem dois estágios iguais, uma para o sinal de corrente e um para o sinal de tensão. Esse circuito é um filtro *anti-aliasing* responsável por limitar a frequência de sinal amostrado, devendo este ter seus valores recalculados de acordo com a frequência do sinal de interesse (NUPEP, 2011).

3° Circuito – Circuito Somador: Soma o sinal advindo do circuito anterior a um nível DC próximo de 1,5V (dependendo da plataforma microcontrolada e da calibração utilizada) de maneira que o nível zero do sinal de entrada fique neste valor (NUPEP, 2011).

Para testar os módulos, os mesmos foram ligados em um motor de indução trifásico de 0,25CV localizado na bancada do Laboratório de Acionamentos do CEFET-MG campus Araxá, conforme Figura 35. Foram realizados os seguintes procedimentos:

- O motor foi ligado em um circuito de potência e comando composto por fusíveis, botoeiras, contatores e relé de proteção, com o fechamento em triângulo (Δ), ou seja, 220V.

- As três fases que alimentam o motor foram interrompidas e enroladas em três sensores *hall* de corrente e também conectadas em três sensores *hall* de tensão, de módulos diferentes.

- Os módulos foram alimentados com uma fonte externa em +15V e -15V.

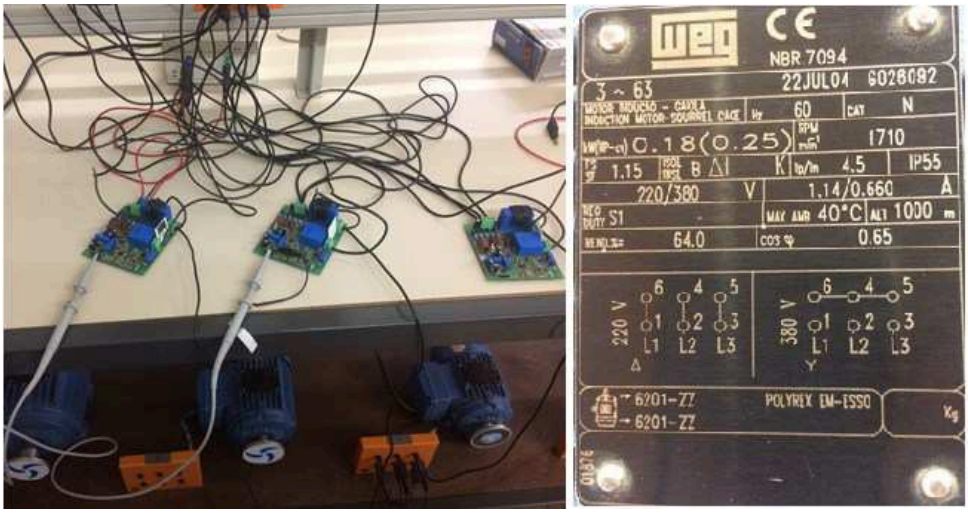
- Com o auxílio do osciloscópio foram medidos os valores de corrente e tensão de duas fases (devido ao osciloscópio utilizado) e levantadas as formas de onda, conforme Figura 36.

Dessa maneira foi verificado que os Módulos de Aquisição e Condicionamento de sinais de Tensão e Corrente, por meio de sensores *hall*, atendem ao projeto em questão.

Definido a utilização dos módulos, foi utilizada uma caixa para alocar os mesmos e também as plataformas microcontroladas ESP32, conforme Figura 37(a).

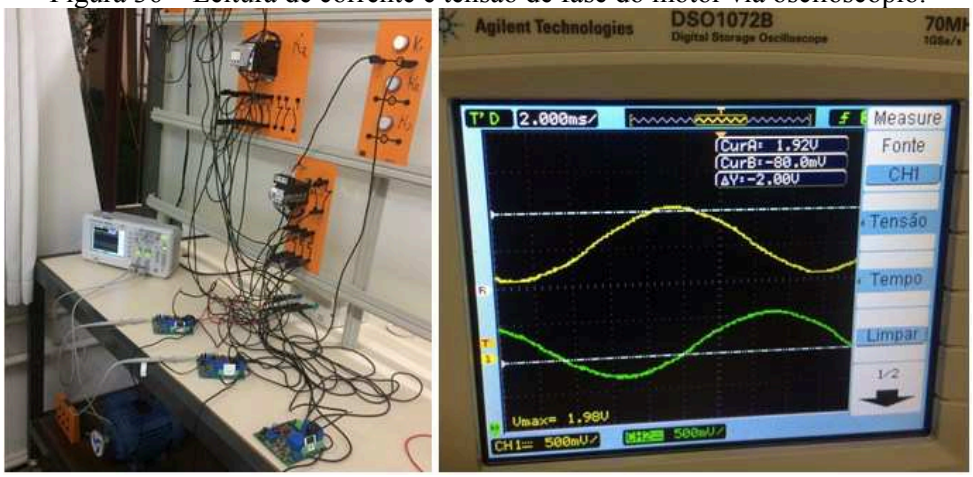
Para alimentação dos módulos foi utilizada uma fonte de tensão contínua de +15 e -15 Volts, conforme Figura 37(b).

Figura 35 – Teste dos módulos para leitura de correntes e tensões de um motor



Fonte: Própria autoria.

Figura 36 – Leitura de corrente e tensão de fase do motor via osciloscópio.

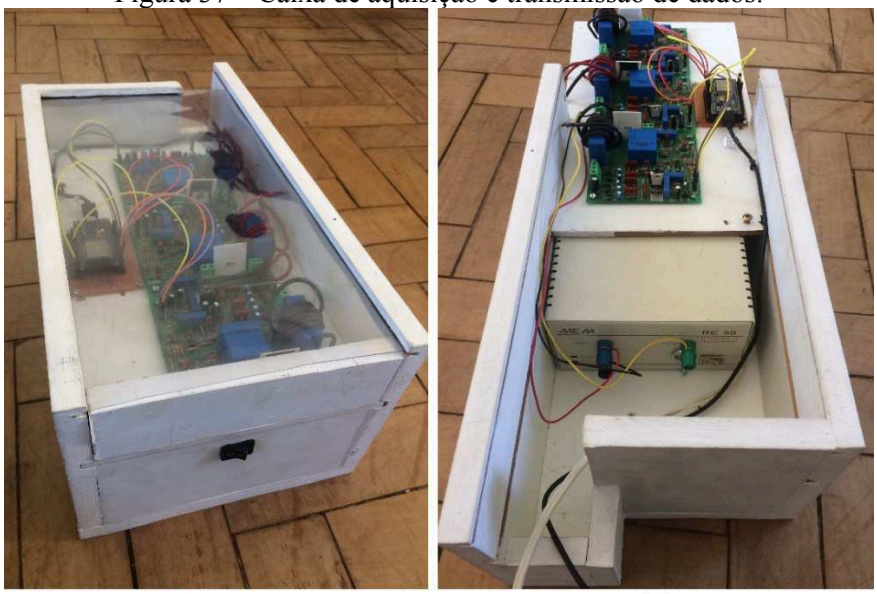


Fonte: Própria autoria.

Os ESP32 foram alimentados por suas entradas micro *USB* através de carregadores de celular bifásico, com saída de tensão contínua de 5 V e corrente de 2 A.

A caixa foi fixada no suporte da bancada didática, conforme Figura 38, em um ponto próximo à caixa de ligação do motor, pois nesse ponto foi interrompido o circuito de alimentação do motor de modo que cada fase proveniente do circuito de acionamento passe pelos sensores hall das placas de medições antes de ir para o motor.

Figura 37 – Caixa de aquisição e transmissão de dados.



(a) **(b)**

Fonte: Própria autoria.

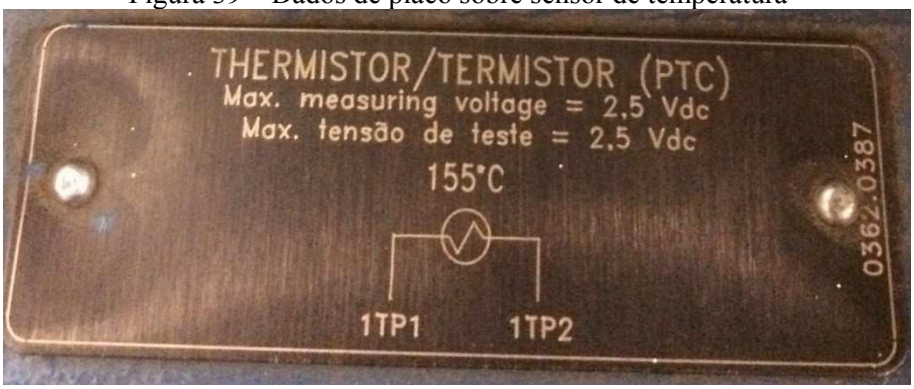
Figura 38 – Caixa fixada na bancada didática da WEG.

Fonte: Própria autoria.

5.2.2. Sensor de Temperatura

Para medir temperatura alta foi utilizado um sensor de temperatura existente no motor. Na carcaça do motor, junto aos dados de placa, há uma informação que indica que o motor possui um termistor PTC em seu interior, conforme Figura 39.

Figura 39 – Dados de placo sobre sensor de temperatura



Fonte: Própria autoria.

De acordo com Catálogo WEG (2002), os termistores PTC e NTC utilizados em seus motores são detectores térmicos compostos de sensores semicondutores que alteram sua resistência subitamente ao atingirem uma determinada temperatura. O termistor PTC (Coeficiente de Temperatura Positivo) é um termistor cuja resistência aumenta subitamente para um valor bem definido de temperatura, especificado para cada tipo. Já o termistor NTC (Coeficiente de Temperatura Negativo), ao contrário do PTC, diminui sua resistência subitamente, entretanto sua aplicação não é comum em motores elétricos, pois os circuitos eletrônicos de controle disponíveis, normalmente são para o PTC. Segue abaixo Figura 40 com a visualização do aspecto externo dos termistores.

Figura 40 – Aspecto externo de um termistor.

Fonte: (CATÁLOGO WEG, 2005).

Porém, durante os testes, ao variar a temperatura do motor notou-se que o sensor de temperatura presente no motor se comporta como um NTC, ou seja, ao aumentar a temperatura do motor, sua resistência diminui, e quando a temperatura diminui sua resistência aumenta.

A variação da temperatura do motor foi realizada aumentando a carga do motor, e para confirmar esse aumento da carga foi verificado o aumento da corrente nas fases do motor.

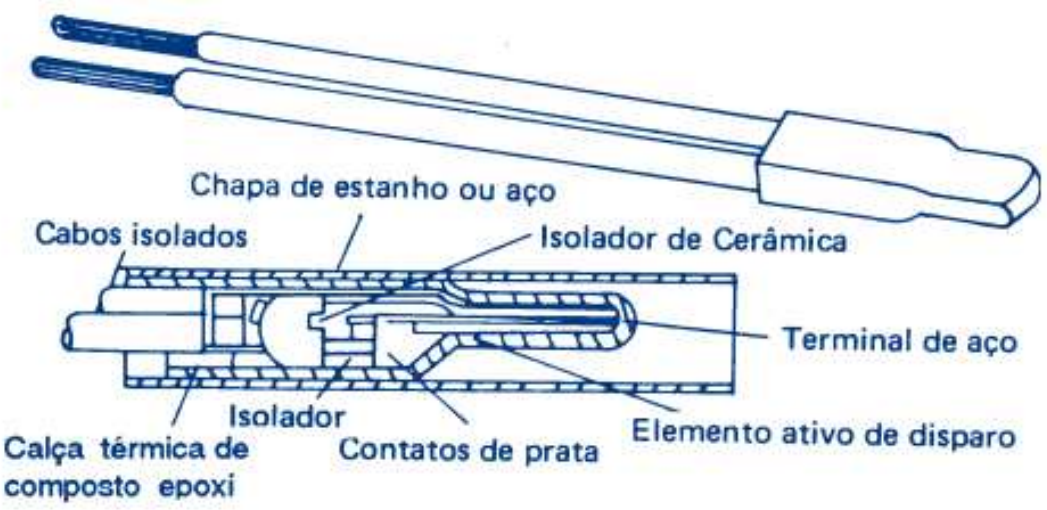
Pesquisas realizadas com profissionais da área levam a crer, pelo comportamento apresentado, que o sensor de temperatura seria um termostato.

Termostatos são sensores térmicos do tipo bimetálico, com contatos de prata normalmente fechados, que se abrem quando ocorre determinada elevação de temperatura. Este volta à sua forma original quando a temperatura de atuação do bimetálico baixar, possibilitando o fechamento dos contatos novamente, conforme ilustrado na Figura 41.

Os termostatos podem ser destinados para sistemas de alarme e/ou desligamento de motores elétricos trifásicos. Os de desligamento deverão atuar na temperatura máxima do material isolante, já os de alarme devem ser apropriados para atuar na elevação de temperatura prevista do motor. Podem ser utilizados três termostatos (um por fase) ou seis termostatos (dois por fase), dependendo do grau de segurança. São ligados em série com a bobina do contator (CATÁLOGO WEG, 2005).

Os termostatos são instalados nas cabeças de bobinas de fases diferentes, conforme ilustrado na Figura 42.

Figura 41 - Aspecto externo e interno de um termostato.



Fonte: (CATÁLOGO WEG, 2005)

Figura 42 – Termostato sendo instalado na cabeça da bobina do motor.

Fonte: (CATÁLOGO WEG, 2005)

5.3. Critérios para Alarmes

Nesse tópico foram apresentados os critérios de alarmes para sobrecorrente, sobretensão e temperatura alta, pois serão as condições a serem monitoradas neste trabalho.

O inversor CFW09 possui algumas proteções para garantir que o motor funcione em segurança, conforme especificações na Figura 43.

Com isso foi necessário realizar simulações de sobrecorrente, sobretensão e temperatura alta dentro dos critérios de funcionamento do inversor, de forma que os alarmes no momento da simulação de defeitos sejam disparados antes que o inversor desarme o motor.

Figura 43 – Proteções inversor CFW09.

SEGURANÇA	Proteções	Sobretensão no circuito intermediário	Curto-circuito na saída
		Subtensão no circuito intermediário	Curto-circuito fase-terra na saída
		Sobretemperaturas no inversor e no motor	Erro externo
		Sobrecorrente na saída	Erro de autodiagnose e de programação
		Sobrecarga no motor (i x t)	Erro de comunicação serial
		Sobrecarga no resistor de frenagem	Ligação Invertida Motor/Encoder
		Erro na CPU (Watchdog) / EPROM	Falta de fase na alimentação (modelos > mecânica 3)
		Falha de encoder incremental	Falha de conexão da interface HMI – CFW09

5.3.1. Sobrecorrente

De acordo com (WEG CFW09, 2006) o inversor de frequência CFW09, presente na bancada, possui em seus critérios de segurança uma proteção contra sobrecorrente. Essa proteção de sobrecorrente atua quando a corrente nas fases do motor for duas vezes maior que a corrente nominal, com a ressalva de ser em aplicações para Torque Constante (CT), conforme Figura 44. Caso aconteça uma situação de sobrecorrente, o inversor desligará o motor e mostrará em seu display o código de erro (E00), que conforme Figura 45, corresponde a sobrecarga.

Figura 44 – Critérios para Sobrecorrente.

SEGURANÇA	PROTEÇÃO	☑ Sobrecorrente/curto-circuito na saída (atuação: >2xInominal para aplicações de Torque Constante (CT)) ☑ Sub./sobretensão na potência ☑ Subtensão/falta de fase na alimentação (1) ☑ Sobretemperatura na potência ☑ Sobrecarga no resistor de frenagem ☑ Sobrecarga na saída (IxT) ☑ Defeito externo ☑ Erro na CPU/EPROM ☑ Curto-circuito fase-terra na saída ☑ Erro de programação

Fonte: (WEG CFW09, 2006)

Figura 45 – Código de erro do CFW09 da bancada didática da WEG.

CÓDIGOS DE ERRO DO CONVERSOR	
E00	Sobrecorrente/curto-circuito na saída
E01	Sobretensão no circuito intermediário (link CC)
E02	Subtensão no circuito intermediário (link CC)
E03	Subtensão/Falta de fase na alimentação
E04	Sobretemperatura no dissipador da potência/Falha no circuito de pré-carga
E05	Sobrecarga na saída (função I x t)
E06	Erro externo
E11	Curto-circuito fase-terra na saída
E24	Erro de programação

Fonte: Acervo do autor.

A sobrecarga de um motor é proporcional à corrente do mesmo, portanto, para sinalizar a sobrecarga iremos utilizar de dois critérios que estão em destaque na Figura 46, que citam as seguintes condições de sobrecarga admissíveis:

- 150% durante 60 seg. a cada 10 min. (1,5 x I nom. – CT)

Nesse critério é tolerado que a corrente chegue a150% do valor da corrente nominal durante 60 segundos, uma única vez em um intervalo de 10 minutos, com o Torque constante. Caso a corrente chegue a 150% da nominal, e permaneça por pelo menos 60 segundos, pela segunda vez o inversor desligará o motor e mostrará em seu display o código de erro de sobrecarga (E05), conforme Figura 45.

Um novo ciclo do contador de tempo se inicia sempre que o tempo de 10 minutos for atingido ou o inversor for inicializado.

- 180 % durante 1 seg. a cada 10 min. (1,8 x I nom. – CT)

Nesse critério é tolerado que a corrente chegue a 180% do valor da corrente nominal durante um 1 segundo, uma única vez em um intervalo de 10 minutos, com o Torque constante.

Nesse caso também se inicia um novo ciclo do contador de tempo sempre que o tempo de 10 minutos for atingido ou o inversor for inicializado.

Figura 46 – Dados de sobrecarga admissível no motor.

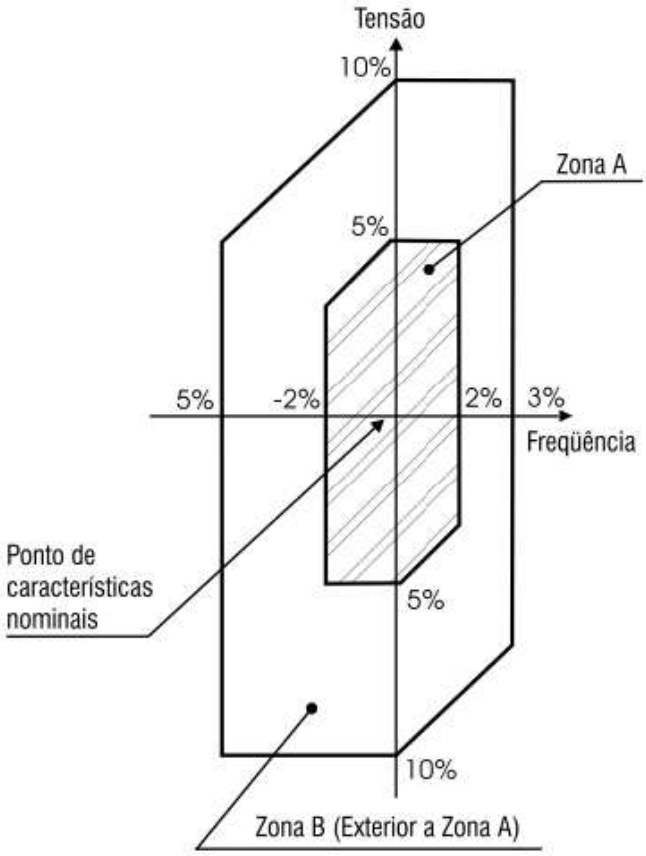
CONTROLE	Tipo de alimentação	Fonte Chaveada
	Microcontrolador	Tipo RISC 32 bits
	Método de controle	PWM Senoidal SVM (Space Vector Modulation)
		Reguladores de Corrente, Fluxo e Velocidade implementados em software (Full Digital)
	Tipos de controle	Escalar (Tensão Imposta – V / F)
		Vetorial Sensorless (sem encoder)
		Vetorial com Encoder
	Chaveamento	Transistores IGBT – Frequências Selecionáveis : 1,25 / 2,5 / 5,0 / 10 kHz
	Variação de frequência	0 a 204 Hz (para rede em 60 Hz)
		0 a 170 Hz (para rede em 50 Hz)
		Acima de 204 Hz (sob consulta)
	Sobrecarga admissível	150% durante 60 seg. a cada 10 min. (1,5 x I nom. – CT)
		180 % durante 1 seg. a cada 10 min. (1,8 x I nom. – CT)
	Rendimento	97%

Fonte: (WEG CFW09, 2012)

5.3.2. Sobretensão

Segundo o (CATÁLOGO WEG, 2005), que se referencia na (NBR 7094, 1996), os motores de indução possuem uma margem de tolerância para a tensão e frequência, e essas combinações das variações de ambas são classificadas como Zona A ou B, conforme Figura 47.

Figura 47 - Limites das variações de tensão e de frequência em funcionamento.



Fonte: (CATÁLOGO WEG, 2005)

Conforme Figura 47, os motores são projetados para desempenharem suas funções principais, no que tange a tensão e frequência, no "Ponto de características nominais", ou seja, trabalhar com tensão nominal e frequência nominal. Mas podem não atender completamente às características de desempenho em relação a estas grandezas e apresentar alguns desvios. Estes desvios possuem duas zonas de classificação toleráveis, são elas:

Zona A – O motor deve ser capaz de desempenhar sua função nesta zona, porém pode não atender às características de desempenho a tensão e frequência nominais, apresentando alguns desvios. Outra característica é que as elevações de temperatura provavelmente serão superiores em relação à quando o mesmo trabalha no "Ponto de características nominais".

Zona B – O motor também deve ser capaz de desempenhar sua função nesta zona, porém pode apresentar desvios superiores em relação à Zona A, no que se refere às características de desempenho a tensão e frequência nominais. Em relação às elevações de temperatura, possuem boa chance de serem superiores em relação à quando o mesmo trabalha no "Ponto de características nominais" e muito provavelmente serão superiores àquelas da Zona A.

5.3.3. Temperatura Alta

Existem vários fatores que podem causar uma temperatura alta em motores, a Figura 48apresenta as causas mais comuns e as proteções habitualmente utilizadas. Observa-se que a figura possui uma legenda que indica se o método adotado realiza uma proteção total, parcial ou não protege contra cada causa de sobretensão.

O motor utilizado, Figura 31, possui um sensor de temperatura, conforme discutido no tópico 5.2.2, que se romperá caso perceba uma variação abrupta de temperatura, que deverá ser por volta de 155 °C.

Sendo assim o alarme da temperatura foi definido para quando o sensor se romper, ou seja, quando esse fato acontecer será emitido um alarme avisando que a temperatura está alta, e assim a proteção contra sobreaquecimento foi acionada.

Figura 48– Causas de sobreaquecimento e sistemas de proteção de motores.

Causas de sobreaquecimento	Proteção em função da corrente: Só fusível ou disjuntor	Proteção em função da corrente: Fusível e protetor térmico	Proteção com sondas térmicas no motor
Sobrecarga com corrente 1.2 vezes a corrente nominal	○	●	●
Regimes de carga S1 a S10	○	◑	●
Frenagens, reversões e funcionamento com partida freqüentes	○	◑	●
Funcionamento com mais de 15 partidas por hora	○	◑	●
Rotor bloqueado	◑	◑	●
Falta de fase	○	◑	●
Variação de tensão excessiva	○	●	●
Variação de freqüência na rede	○	●	●
Temperatura ambiente excessiva	○	○	●
Aquecimento externo provocado por rolamentos, correias, polias, etc	○	○	●
Obstrução da ventilação	○	○	●

Legenda: ○ não protegido
◑ semi-protegido
● totalmente protegido

Fonte: (CATÁLOGO WEG, 2005)

5.4. Conclusões e Resultados do Capítulo

Após montagem e testes realizados, referente ao sensoriamento, foi verificado que os sensores e critérios utilizados para alarmes, Tabela 4, se mostraram satisfatórios para o projeto em questão. Porém é necessário realizar uma calibração dos sensores e também do ESP32, para um melhor resultado. O capítulo seguinte descreve as calibrações necessárias.

Tabela 4 - Critérios para Alarmes

Alarmes	Condições para Disparo	Observações
Sobrecorrente	Corrente > 6,48 A	150% da corrente nominal (4,32 A)
	Corrente > 7,77 A	180% da corrente nominal (4,32 A)
Sobretensão	Tensão > 242 V	Tensão 10% acima da nominal (220 V)
Temperatura Alta	Temperatura > 155 °C	Temperatura de ruptura do sensor

Fonte: Própria autoria.

6. CALIBRAÇÃO E COMUNICAÇÃO

Neste capítulo foi descrito as calibrações necessárias para um melhor funcionamento da plataforma ESP32 e do Módulo de Aquisição e Condicionamento de Sinais de Tensão e Corrente. Também foi apresentada a comunicação realizada entre as plataformas microcontroladas ESP32.

6.1. Módulo de aquisição de tensão e corrente

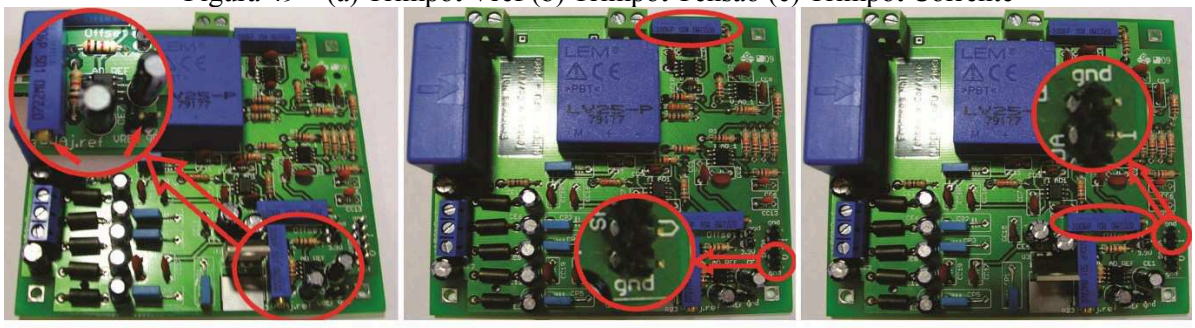
Para realizar as medidas de corrente e tensão do MIT, são utilizados os Módulos de Aquisições de Sinais, conforme descrito no tópico5.2.1. Os mesmos possuem uma calibração padrão, sendo acrescido um circuito passivo para ajuste do nível de tensão e filtragem do sinal a ser aquisitado pelo ESP32.

6.1.1. Calibração do Módulo

O módulo deve ser calibrados antes de sua utilização, por seus trimpots. Um deles é para ajuste de "Vref", Figura 49(a), que convencionalmente é regulado em 1,5V. Os outros dois são para regular o offset dos sinais de saída, sendo um para tensão, Figura 49(b), e o outro para corrente, Figura 49(c). Estes devem ser calibrados para o mesmo valor do "Vref", sem estarem recebendo nenhum sinal de entrada (NUPEP, 2011).

O sensor de corrente por efeito *hall*, LA55P, possui uma corrente nominal de 50A $_{RMS}$. Para medir correntes de menor valor devemos utilizar um número maior de voltas no sensor, sendo que cada volta multiplica por dois o valor da corrente lida pelo módulo.

Figura 49 – (a) Trimpot Vref (b) Trimpot Tensão (c) Trimpot Corrente



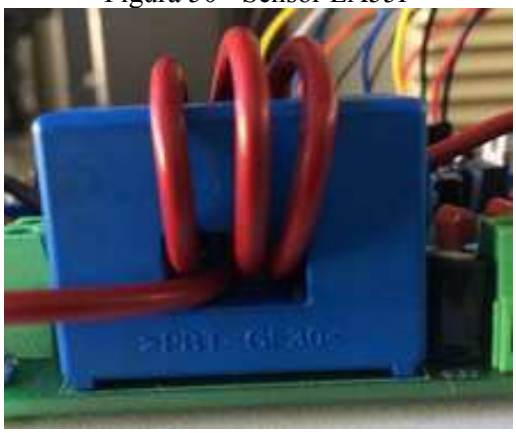
(a) (b) (c)

Fonte: Adaptado de (NUPEP, 2011).

No caso deste trabalho foram utilizadas três voltas (50/3 = 16,67 A_{RMS}), sendo assim conseguimos ler correntes até 16,67 A e assim ter uma melhor precisão, conforme Figura 50.

As saídas de corrente e tensão possuem um condicionamento de sinal com um range de 0 a 3,0 V, de modo que o sinal possui nível zero no valor de 1,5 V.

Figura 50 - Sensor LA55P



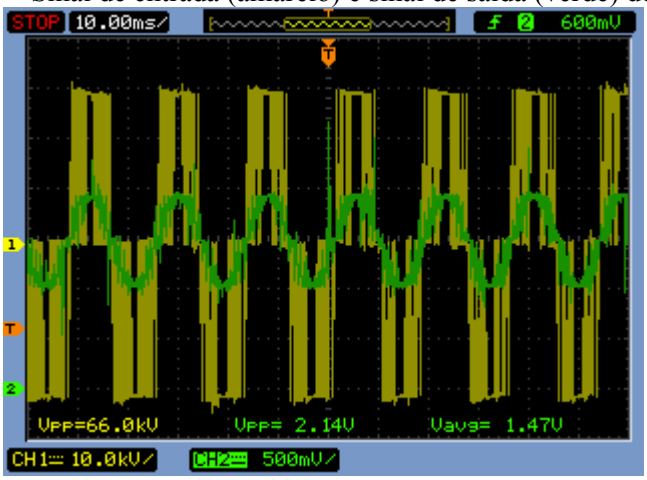
Fonte: Própria autoria.

6.1.2. Condicionadores de níveis de sinal e filtro

Com o auxílio de um osciloscópio foi medido o sinal na entrada do Módulo de Aquisição e Condicionamento de Sinais (sinal do inversor), representado em amarelo na Figura 51. E também foi medido o sinal de saída do módulo, representado em verde na Figura 51.

Pela imagem é possível observar que o sinal de saída do módulo necessita ser filtrado para obter uma forma mais próxima da senoidal.

Figura 51 – Sinal de entrada (amarelo) e sinal de saída (verde) do módulo.



Fonte: Própria autoria.

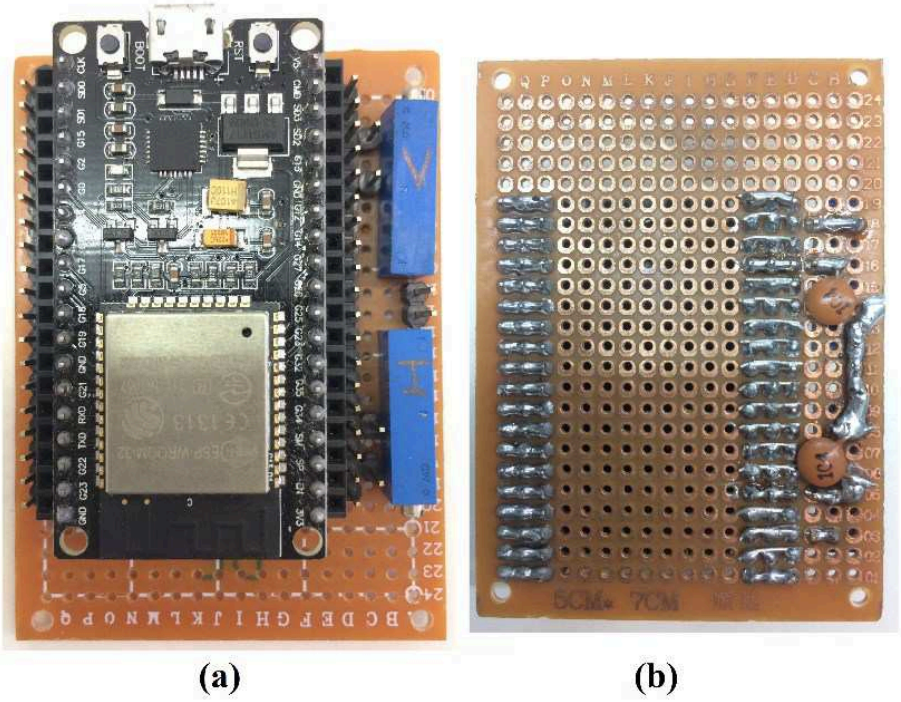
Como explicado no tópico 5.2.1, o Módulo de Aquisição e Condicionamento de Sinais possui um filtro *anti-aliasing,* porém esse filtro foi projetado para uma faixa de frequência diferente do inversor utilizado neste trabalho, com isso esse filtro anti-*aliasing* não filtrou corretamente o sinal, sendo necessário adicionar filtro RC (Resistivo-Capacitivo) auxiliar para melhorar a aquisição dos sinais de corrente e tensão.

Para os dois circuitos RC auxiliares (corrente e de tensão), foram instalados dois trimpots de 20k ohms cada conforme Figura 52(a), associados a dois capacitores cerâmicos de 100nF cada, conforme Figura 52(b).

Com a instalação dos filtros RC, verificou-se que o sinal se aproximou mais de uma onda senoidal, conforme ilustra a Figura 53 sem o filtro RC e a Figura 54 com o filtro RC. Nessa figura são apresentadas as formas de onda equivalentes a corrente (azul) e tensão (vermelho), nas entradas ADC do ESP32, ou seja, depois dos filtros. É importante observar que nesse momento não consideramos os ganhos de corrente e tensão, e que foi adicionado uma carga no motor, o que fez com que o sinal de corrente seja maior que o sinal de tensão.

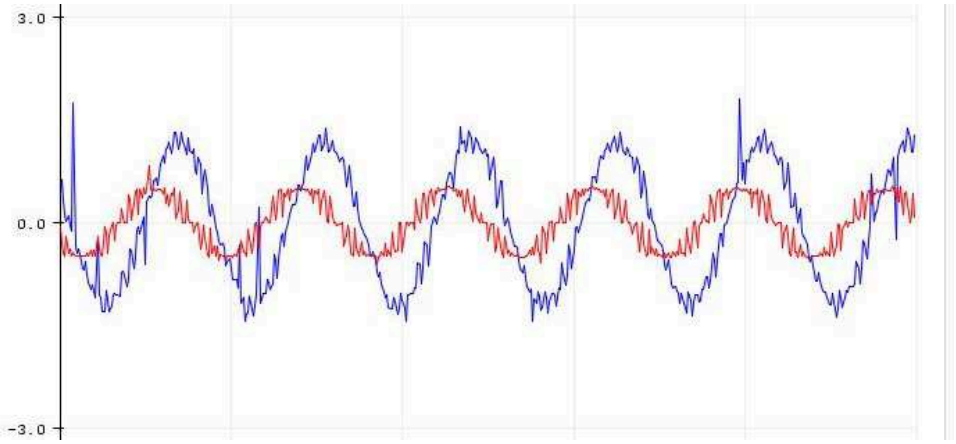
Outra função dos *trimpots* foi a de ajustar a saída do módulo para valores de 0 a 1 V, ao invés de 0 a 3,0 V, que corresponde à faixa linear de tensão de entrada no conversor analógico-digital (AD) do ESP32.

Figura 52 – Trimpots e capacitores dos filtros RC.



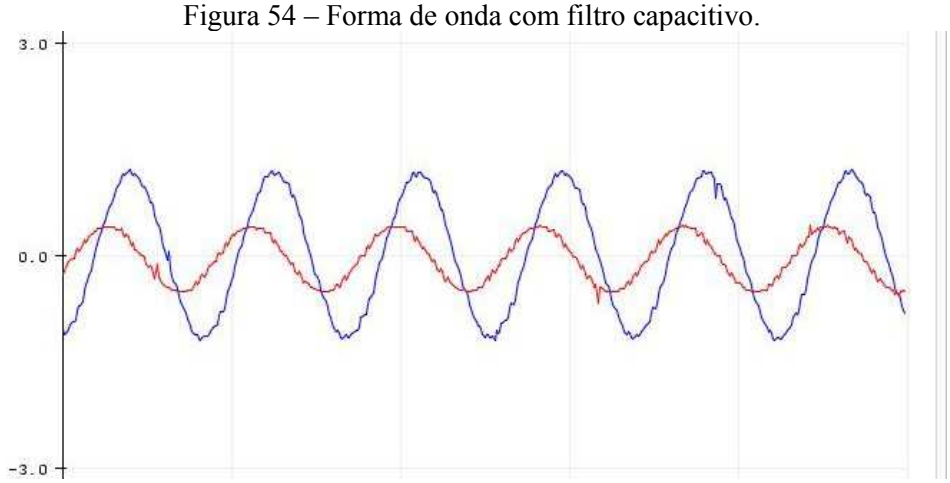
Fonte: Própria autoria.

Figura 53 – Forma de onda sem filtro capacitivo.



Fonte: Própria autoria.

Figura 54 – Forma de onda com filtro capacitivo.



Fonte: Própria autoria.

6.1.3. Taxa de Amostragem

Neste tópico foi definida a taxa de amostragem utilizada neste trabalho, que é o número de amostras de um sinal analógico coletadas em um certo intervalo de tempo, para conversão em um sinal digital. Foi utilizada a seguinte fórmula para definir a taxa de amostragem:

$$Ta = \frac{T}{Na} \quad (1)$$

Onde: Ta = Taxa de amostragem

T = Período = 1/f

f = frequência = 60Hz

Na = Número de amostras por período = 85

O número de amostras por período foi determinado, por meio de testes, de forma que o tempo gasto durante o serviço de interrupção do ESP32 para aquisição e cálculo dos valores RMS de corrente e tensão não fosse demasiado em relação ao tempo disponível ao programa principal, entre chamadas de interrupção. Substituindo os valores acima na equação (2), encontramos o seguinte valor para taxa de amostragem:

$$Ta = \frac{1/60}{85} = 1{,}9607 * 10^{-4} \quad (2)$$

Portanto, a taxa de amostragem é de **196µs**.

A função que utiliza a taxa de amostragem, no código do ESP32 é a seguinte:

$$timerAlarmWrite(timer, 196, true); \quad (3)$$

6.2. ESP 32

Como dito anteriormente, para o processamento dos valores de corrente, tensão e temperatura, iremos utilizar o ESP32, o mesmo realizará a leitura de corrente e tensão em suas entradas analógicas e a leitura de temperatura alta em uma de suas entradas digitais.

6.2.1. Entradas Analógicas

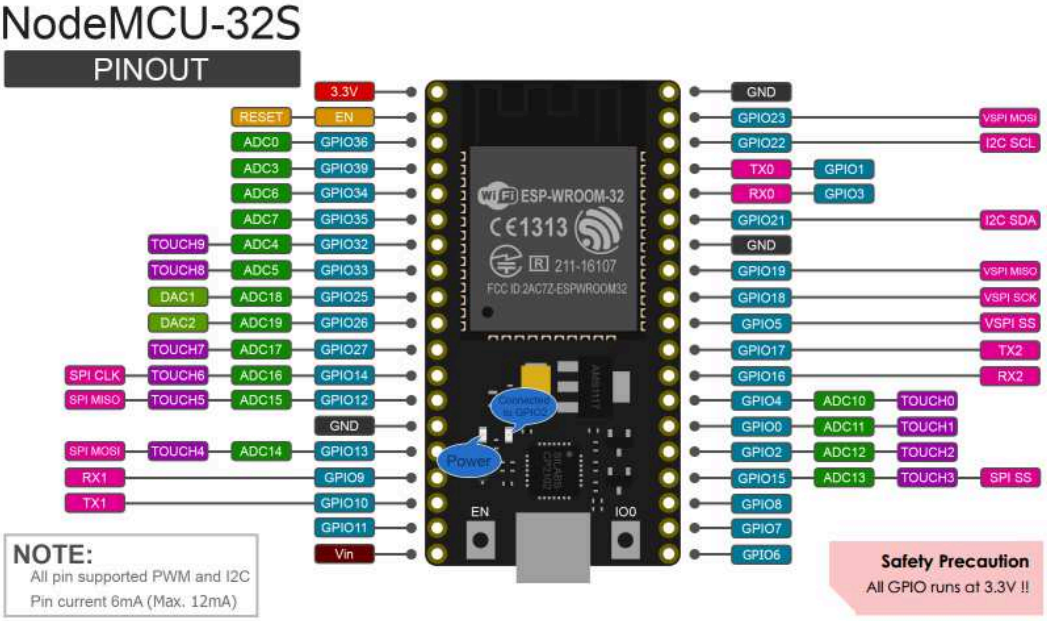
A plataforma ESP32 possui dezesseis entradas para conversão analógico digital (*ADC - Analog-to-Digital Converter*), conforme Figura 55.

Das dezesseis entradas, seis estão ligadas internamento no ADC1 do chip do ESP32, e as outras dez estão ligadas no ADC2, conforme PINOUT do chip do ESP32, Figura 56.

Fazendo uma ressalva, é importante observar que, após vários testes, constatou-se um conflito entre as funções *Wifi* e *analogRead*(), ou seja, quando a função *WiFi* está ativada algumas entradas analógicas deixam de funcionar.

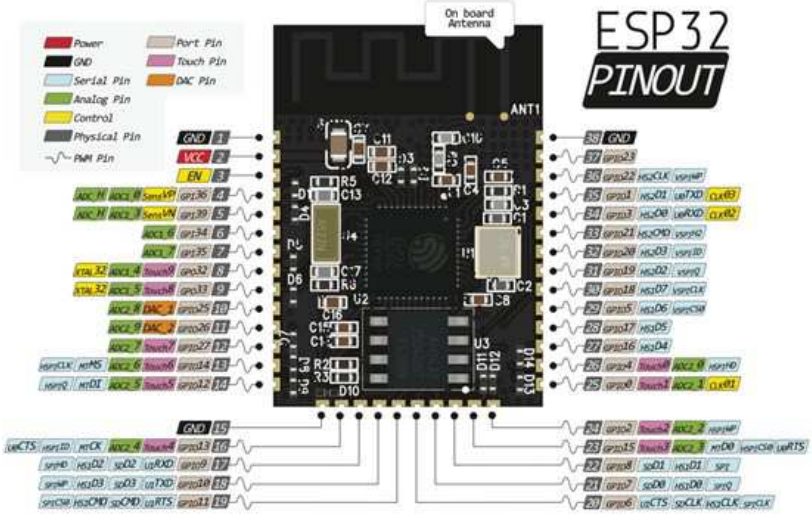
Após pesquisas, verificou-se que as dez entradas analógicas do ADC2 não funcionam ao mesmo tempo que a função *WiFi*, sendo assim, quando se utiliza o ESP32 conectado ao *Wifi*, apenas as seis entradas analógicas conectadas ao ADC1 ficam disponíveis ao uso. Durante as pesquisas foi observado que o ESP32 também poderá apresentar esse mesmo comportamento quando utilizar a função *Bluetooth* (ESPRESSIF ESP32, 2017) (GITHUB, 2017).

Figura 55 – PINOUT plataforma ESP32



Fonte: (CATALÓGO NODEMCU ESP32, 2017).

Figura 56 - PINOUT chip ESP32



Fonte: (KOYANAGI, 2019).

Uma solução para esse problema é realizar as leituras analógicas com o *WiFi* desconectado, e conectar apenas para transmitir as informações. Outra alternativa é utilizar dois ESP32 se comunicando, o primeiro responsável pela aquisição dos dados (não se conecta a internet) e o segundo responsável pela transmissão dos dados para internet. No caso deste trabalho, como a ideia é realizar um monitoramento em tempo real, é importante estar o tempo todo conectado ao *WiFi*, desta forma, foram utilizados dois ESP32.

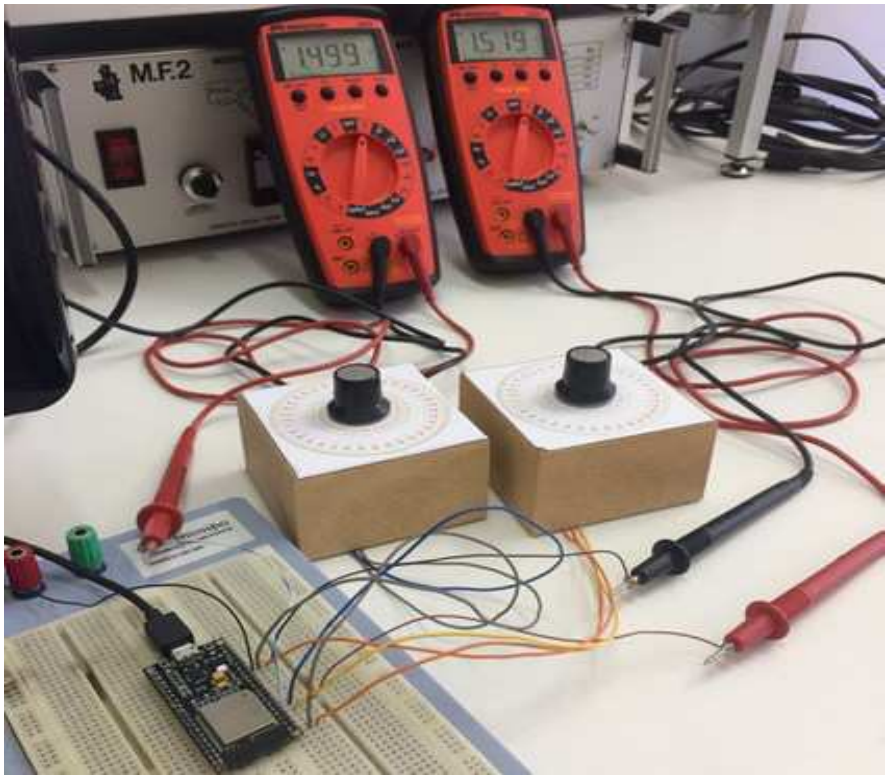
Cada uma das entradas ADC do ESP32 possuem 12 bits de resolução, ou seja, a conversão apresenta valores entre 0 e 4095, correspondendo às tensões de entrada que possuem

valor padrão de 0 a 3,3 V. Uma precaução que se deve tomar é que os pinos de entrada do ESP32 suportam no máximo uma tensão contínua de 3,3 Volts.

6.2.2. Curva de calibração

Foi constatado que a variação do ESP32 não é linear, dessa forma foi necessário realizar uma calibração a partir da curva de variação de tensão pelos valores equivalentes lidos em sua serial. Para levantar a curva, foram utilizadas duas entradas ADC, uma simulando a leitura de corrente e a outra a de tensão, conforme Figura 57.

Figura 57 – Simulação com potenciômetros.



Fonte: Própria autoria.

Com o auxílio de um potenciômetro foram injetados diferentes níveis de tensão em uma entrada analógica. Estes foram ligados da seguinte maneira: um dos terminais fixos foi ligado ao pino de saída de tensão de 3,3 V, o outro terminal fixo no pino de GND e o cursor (sinal de saída) foi ligado em uma entrada ADC.

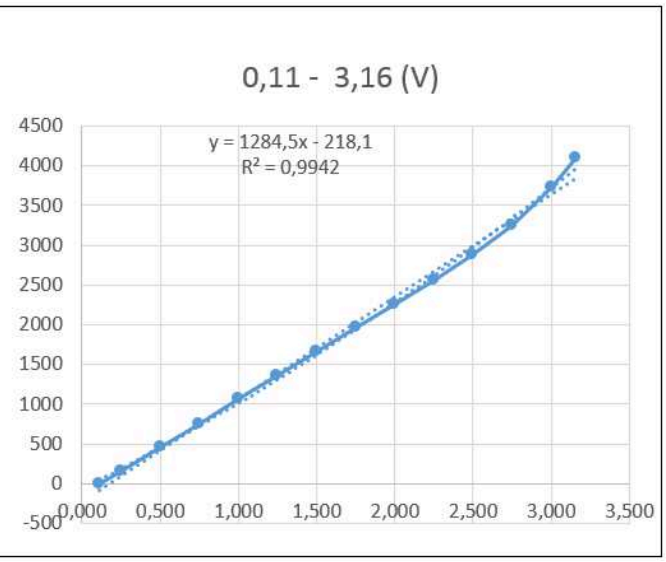
No momento do teste verificou-se que, com uma tensão de 110 mV o ESP32 apresentava 0 em sua serial, e que com um valor de 3,16 V apresentava 4095. Os resultados intermediários, assim como a acentuação maior da curva apresentada nos valores máximos de tensão, estão representados no gráfico, conforme Figura 58.

Outro teste realizado foi desconsiderar os valores finais para plotar o gráfico, com isso verificou-se que o mesmo apresenta uma curva mais linear e o valor do R^2 se aproxima mais do valor um, conforme Figura 59.

Com o intuito de melhorar a calibração foi realizado um condicionamento do sinal, conforme descrito no próximo tópico.

Figura 58 – Curva de calibração de 0,11 a 3,36 V.

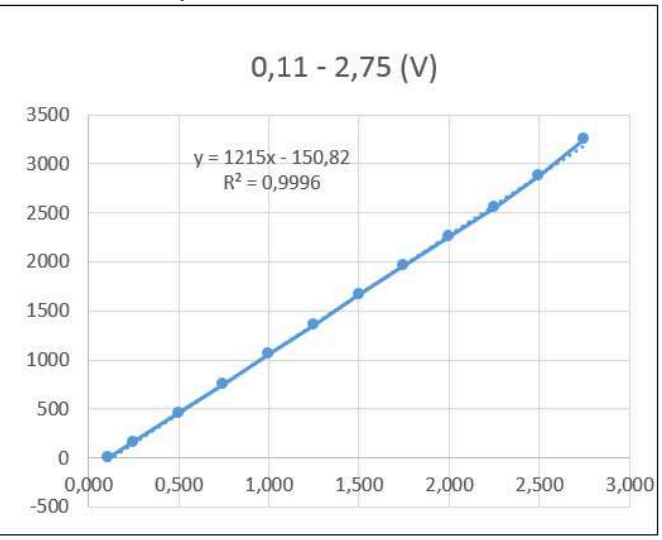
Multímetro (V)	ESP32 (serial)
0,110	0
0,250	163
0,500	460
0,750	755
1,000	1062
1,250	1360
1,500	1664
1,750	1962
2,000	2261
2,250	2559
2,500	2878
2,750	3248
3,000	3728
3,160	4095



Fonte: Própria autoria.

Figura 59 - Curva de calibração de 0,11 a 2,75 V.

Multímetro (V)	ESP32 (serial)
0,110	0
0,250	163
0,500	460
0,750	755
1,000	1062
1,250	1360
1,500	1664
1,750	1962
2,000	2261
2,250	2559
2,500	2878
2,750	3248
3,000	3728
3,160	4095



Fonte: Própria autoria.

6.2.3. Ajuste da curva de calibração

Devido à não linearidade nos valores próximos a 3,3V optou-se por trabalhar em uma faixa de valores entre 0,1 e 1,1 V. Para que isso fosse possível, foi utilizada uma função para ler as entradas analógicas que possibilita zerar a atenuação.

Normalmente para realizar a leitura de uma entrada analógica, pino 35 por exemplo, se utiliza a função:

$$le_{adc} = analogRead(35); \quad (4)$$

A função acima utiliza uma atenuação, de forma a conseguir ler valores na faixa de 0 a 3,3V.

Para utilizar atenuação zero, para realizar a leitura do mesmo pino 35, são utilizadas as funções:

$$le_{adc} = adc1\_get\_raw(ADC1\_CHANNEL\_7); \quad (5)$$

$$adc1\_config\_width(ADC\_WIDTH\_BIT\_12); \quad (6)$$

$$adc1\_config\_channel\_atten(ADC1\_CHANNEL\_7, ADC\_ATTEN\_DB\_0); \quad (7)$$

Para utilizar as funções acima, é necessário instalar a seguinte biblioteca:

$$\#include\ "adc.h" \quad (8)$$

Esta biblioteca está disponível para download em (GITHUB, 2018).

Enfim, com o código pronto foram realizados os testes com o potenciômetro, conforme explicado no tópico anterior. Os resultados seguem na Figura 60, de posse desses dados foi gerado o gráfico e levantado a equação da curva utilizando a primeira e segunda coluna, conforme apresentado na Figura 61.

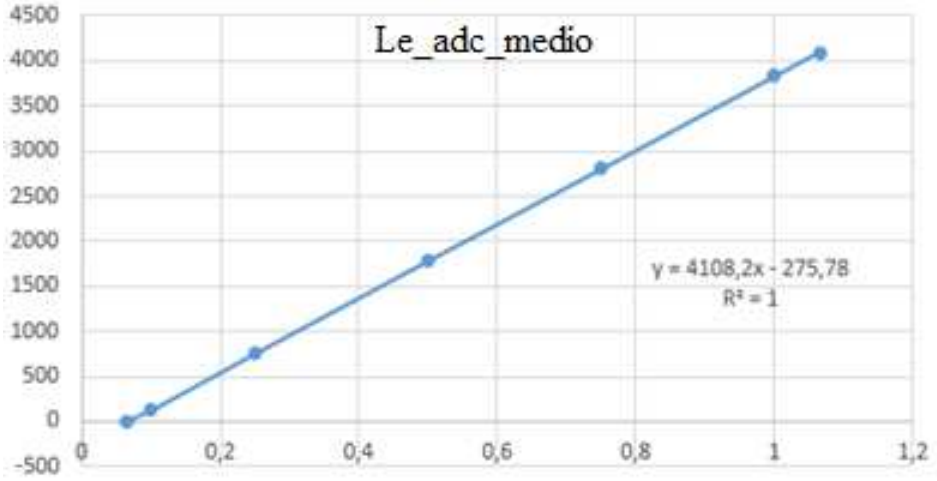
Nota-se que os valores de tensão variaram de 0,065 a 1,065. Obtendo desta forma uma resolução de 1/4095.

Figura 60 – Tabela com os dados do teste com potenciômetro.

Multímetro (V)	Le_adc_medio	Va_max	Va_rms
0,065	0	0,050	0,010
0,100	130	0,190	0,110
0,250	748	0,700	0,600
0,500	1776	1,520	1,430
0,750	2804	2,380	2,260
1,000	3840	3,210	3,100
1,065	4095	3,300	3,300

Fonte: Própria autoria.

Figura 61 – Curva de calibração após ajuste.



Fonte: Própria autoria.

A partir da forma geral da função polinomial de 1° grau, conforme equação abaixo:

$$f(x) = ax + b \tag{9}$$

Foi encontrado o valor de **"a"** (coeficiente de x) e o valor de **"b"** (termo constante):

$$A = 4.108{,}2 \text{ e } B = 275{,}78 \tag{10}$$

Esses dados são utilizados no código do programa, conforme equação abaixo:

$$va = \frac{(float)(le_{adc} + 275{,}78)}{4.108{,}2}; \tag{11}$$

6.2.4. Calibração do valor máximo e rms

Após o ajuste da curva de calibração foi feito o cálculo no código para se encontrar o valor máximo, conforme equações abaixo:

$$if\ (va\_max < va); \tag{12}$$

$$va\_max = va; \tag{13}$$

E o valor rms, conforme equações abaixo:

$$va\_rms = va\_rms + va * va; \tag{14}$$

$$vrms = sqrt(va\_rms / int\_count); \tag{15}$$

Com o intervalo de tensão mínima (0,065 mV) e máxima (1,065 mV) encontrados no tópico anterior, foi obtida uma forma de onda, conforme Figura 62, para alimentar a entrada analógica do ESP32.

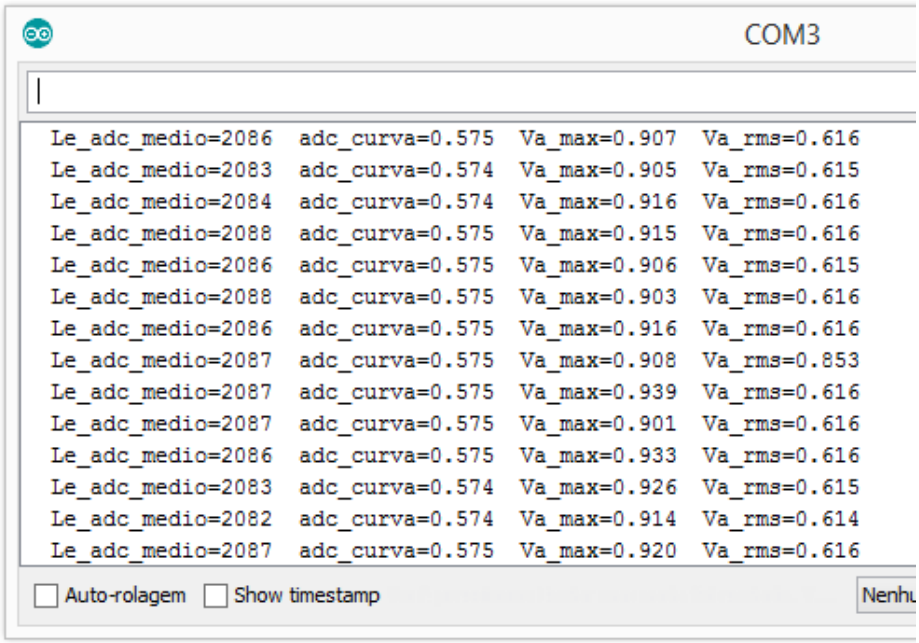
Figura 62 – Forma de onda de 0,065 mV a 1,065 mV.

Fonte: Própria autoria.

Carregado o código e alimentado a entrada analógica do ESP32 com um sinal de forma de onda igual ao da Figura 62, encontramos os valores de tensão máxima e tensão rms no monitor serial, conforme Figura 63.

Note que os valores de tensão máxima e tensão rms encontrados na Figura 63, batem com os valores encontrados na Figura 62, comprovando assim que o código está calculando esses valores corretamente.

Figura 63 – Saída de valores no monitor serial.



Fonte: Monitor serial da IDE da Arduino.

6.3. Comunicação entre Microcontroladores ESP32

Neste trabalho, devido a incompatibilidades entre bibliotecas utilizadas no código do programa, foi necessário utilizar duas plataformas microcontroladas ESP32.

Um dos conflitos encontrados foi entre a biblioteca *WiFi* da plataforma microcontrolada ESP32 e as função de Interrupção do Timer, ou seja, a interrupção não funciona da maneira esperada quando o ESP32 está conectado na internet.

O primeiro ESP32 fica responsável por realizar a leitura dos dados aquisitados pelo módulo de aquisição e condicionamento de dados (de acordo com uma taxa de amostragem definida), realiza os cálculos para encontrar os valores médios, máximos e rms de tensão e corrente e transmiti os valores rms para o segundo ESP32.

O segundo ESP32 recebe os dados do primeiro, aplica um ganho de acordo com cada grandeza, realiza as condições de alarme para sobrecorrente, sobretensão e temperatura alta, se conecta à internet, salva os resultados em forma de gráfico no ThingSpeak e envia alarmes para o aplicativo Telegram.

Foram testados alguns protocolos para realizar a comunicação entre os ESP32, e dentre eles se optou por utilizar a comunicação serial (TX/RX)[2], por se tratar de uma comunicação simples e que cumpre com o que é proposto no trabalho.

Foi utilizado o sentido de transmissão simplex, ou seja, comunicação unidirecional, pois o primeiro ESP32 enviará os dados e o segundo ESP32 receberá os dados, não sendo necessário o segundo enviar dados ao primeiro.

As plataformas microcontroladas ESP32, utilizadas neste trabalho, possuem três seriais cada e seus respectivos pinos estão identificados na Tabela 5 (CATALÓGO NODEMCU ESP32, 2017).

Tabela 5 - Comunicação serial e seus respectivos pinos.

Comunicação Serial ESP32						
Seriais	TX0	RX0	TX1	RX1	TX2	RX2
Pinos	1	3	10	9	17	16

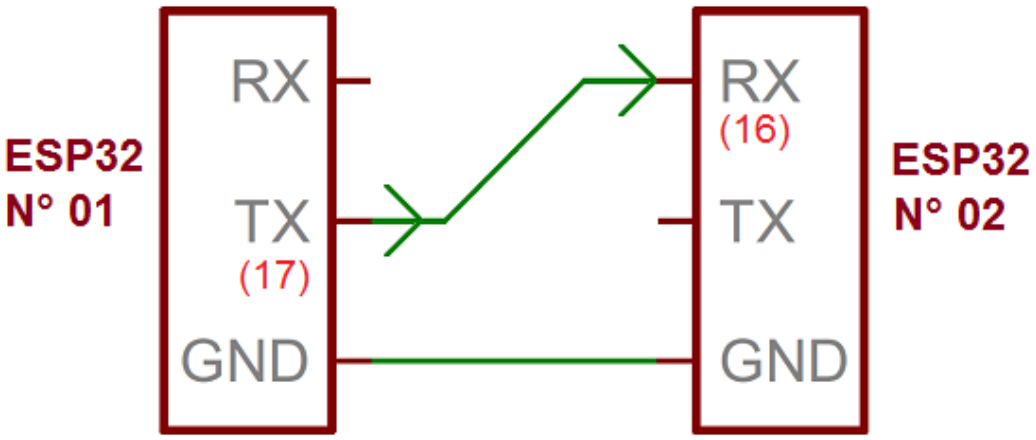
Fonte: Própria autoria.

Uma observação a ser feita sobre a comunicação serial, é que foi encontrado um conflito entre a serial 0 da plataforma microcontrolada ESP32 e a biblioteca do Telegram "Universal-TelegramBot.h" pois a biblioteca utiliza a serial 0, ou seja, quando a função Telegram é chamada ocorre um atraso na transmissão de dados pela serial 0. Para resolver esse conflito foi utilizado a serial 2 do ESP32 para transmitir os dados de um ESP32 para o outro, e assim eliminar o atraso no envio de dados.

Com isso, foi realizado a comunicação entre os dois microcontroladores, ligando o pino TX (17) do primeiro ESP32 ao pino RX (16) do segundo, além disso também foram conectados os pinos de terra (GND) dos dois ESP32, conforme Figura 64.

[2] RX é o termo usado para representar o pino receptor de uma comunicação serial e TX representa o transmissor (SPARKFUN, 2016).

Figura 64 – Comunicação Serial 2 entre ESP32



Fonte: Adaptado de (SPARKFUN, 2016).

6.4. Conclusões e Resultados do Capítulo

Neste capítulo foram realizadas as calibrações necessárias ao módulo de aquisição e condicionamento de corrente e tensão e a plataforma microcontrolada ESP32. Foi verificada a importância de se realizar a calibração e os ajustes necessários, para um bom funcionamento do sistema. Também foi apresentado o filtro RC adicional ao sistema, com o intuito de condicionar o sinal de saída para aquisição e cálculo dos valores rms.

Algumas incompatibilidades e conflitos foram verificados, como por exemplo:

- Foi verificado que algumas entradas analógicas da plataforma microcontrolada ESP32 param de funcionar quando o ESP32 está conectado à internet, devido a um conflito entre a função *WiFi* e a função *analogRead*.

- Foi verificado um conflito entre a Biblioteca *WiFi* do ESP32 e as função de Interrupção do Timer.

- Foi verificado um conflito entre a serial 0 do ESP32 e a biblioteca do Telegram, pois a biblioteca utiliza a serial 0.

Com o intuito de resolver todos esses conflitos e incompatibilidades foram utilizados dois ESP32 com funções distintas, onde somente o segundo se conecta à internet e a comunicação entre os dois é feita pela serial 2.

Nó próximo capítulo foram apresentadas as simulações dos alarmes e os resultados obtidos.

7. RESULTADOS EXPERIMENTAIS

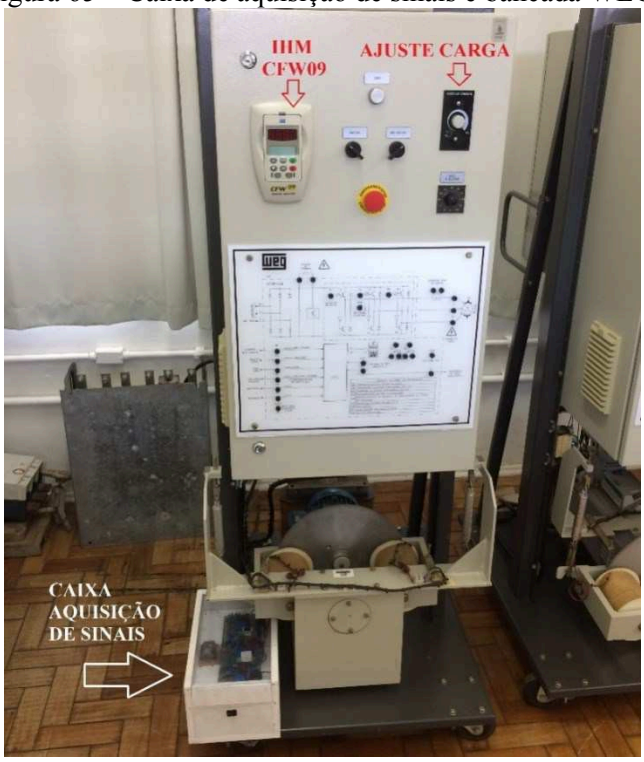
Neste capítulo foi apresentado os resultados experimentais advindos dos testes de sobrecorrente, sobretensão e temperatura alta.

Para realizar os testes, foi utilizada a bancada didática da WEG, apresentada no tópico 5.1 deste trabalho, a qual possui um ajuste de carga por meio de um potenciômetro que possibilita alterar a corrente, e possui um inversor CFW09 que possibilita alterar a velocidade de rotação do motor e consequentemente a valor da tensão.

A caixa projetada para este trabalho, contendo o módulo de aquisição de sinais e as plataformas microcontroladas ESP32, foi instalada na bancada didática, para conexão dos sensores ao motor, conforme Figura 65.

É importante salientar que os testes foram realizados utilizando apenas uma fase do motor, pois apenas um módulo dos três disponíveis, estava funcionando corretamente. Porém o monitoramento de apenas uma fase foi suficiente para a realização dos testes necessários para a obtenção dos resultados, sem prejudicar o objetivo do trabalho.

Figura 65 – Caixa de aquisição de sinais e bancada WEG.



Fonte: Própria autoria.

7.1. Sobrecorrente

Para o teste de sobrecorrente, primeiramente foi acionado o motor via inversor CFW09 e aguardado um tempo suficiente para o mesmo atingir a rotação nominal, que de acordo com os dados de placa é de 1700 rpm. Na sequência, através do potenciômetro presente na bancada didática da Weg, foi aumentando a carga gradativamente e, com o auxílio do alicate amperímetro, foi monitorada a corrente.

Baseado nos critério de alarmes para sobrecorrente, apresentado no tópico5.3.1, será utilizado neste trabalho a condição mais crítica, conforme abaixo:

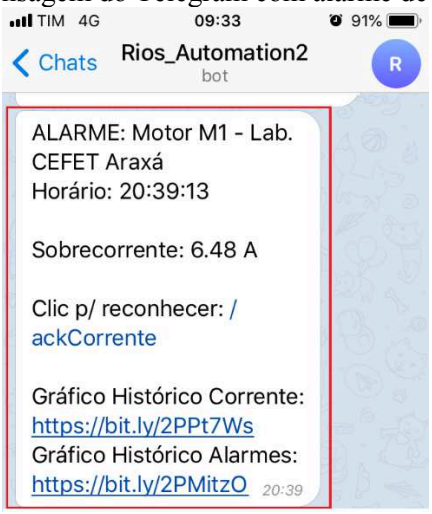
- Alarmar quando a corrente do motor atingir um valor de 1,5 a corrente nominal, ou seja, **6,48 A** (4,32 x 1,5 = 6,48 A).

Quando essa condição for atingida será enviado uma mensagem via Telegram aos smartphones cadastrados e registrado o evento no gráfico do ThingSpeak.

7.1.1. Telegram

A Figura 66, mostra a mensagem recebida no aplicativo Telegram, referente ao alarme de sobrecorrente. Observe que a informação com o valor da corrente se encontra no centro da mensagem, e complementando, no corpo da mensagem também se encontra informações sobre a *tag* do motor, o local que o mesmo se encontra, o horário do alarme, e também disponibiliza uma opção para reconhecer o alarme e um link para acessar o histórico com os valores de corrente e alarmes de sobrecorrente, na página da plataforma ThingSpeak.

Figura 66 - Mensagem do Telegram com alarme de Sobrecorrente.

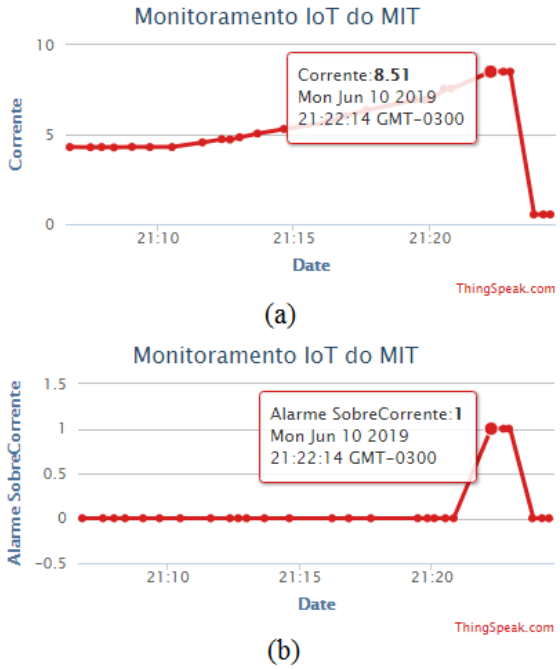


Fonte: Própria autoria.

7.1.2. ThingSpeak

O ThingSpeak apresenta o histórico com os valores de leitura da corrente, Figura 67(a), e o histórico de alarmes de sobrecorrente, Figura 67(b). Também são disponibilizadas as informação do horário, dia, mês, ano e também fuso horário. No gráfico de alarme, quando o valor estiver em "1" significa que o alarme de sobrecorrente foi ativado.

Figura 67- Registro de Sobrecorrente, no ThingSpeak.



Fonte: Própria autoria.

7.2. Sobretensão

Para o teste de sobretensão, nos baseamos no controle escalar V/F, onde variando a tensão varia-se a frequência e consequentemente a velocidade. Foi utilizado deste princípio, e assim ao variar a rotação do motor via inversor CFW09, consequentemente a tensão também varia, e com isso foi possível simular uma sobretensão.

Verificou-se que, para que o motor trabalhe em um tensão constante de 220 V, sua rotação deve ser de 1.410 rpm a plena carga, ou seja, corrente nominal de 4,32 A. A bancada didática da WEG possui bornes para medir tensão entre as fases e também bornes para se medir corrente a partir de um transformador de corrente. Dessa maneira o multímetro da direita mostra o valor de tensão em volts, a IHM do CFW09 que está no centro mostra o valor da rotação em rpm e o multímetro da esquerda mostra a corrente em amperes, conforme Figura 68.

Figura 68 – Leitura de rotação com corrente nominal e tensão de 220V.

Fonte: Própria autoria.

Baseado no tópico 5.3.2deste trabalho, onde são apresentadas as zonas com as margens de tolerância de trabalho do motor em relação a tensão e frequência, verificou-se que com uma tensão 10% acima da nominal o motor passa a trabalhar fora das zonas de tolerância.

Com isso foi considerado que um valor de tensão 10% acima de 220V é considerado sobretensão, ou seja, o motor atingindo uma tensão de **242V** (220 x 1,10 = 242 V) será disparado o alarme de sobretensão.

Com o auxílio do inversor e dos multímetros foi variada a rotação até um valor de tensão de 242 V, nesse momento a IHM do inversor indicou um rotação de 1700 rpm com corrente a plena carga, conforme Figura 69.

Assim, seguindo os parâmetros citados foram realizados os testes no Telegram e no ThingSpeak, conforme explicado nos próximos tópicos.

Figura 69 - Leitura de rotação com corrente nominal e tensão de 242V.

Fonte: Própria autoria.

7.2.1. Telegram

Para simular o alarme e receber a notificação via Telegram, inicialmente foi ligado o motor com uma rotação de 1410 rpm, após o mesmo se estabelecer foi aumentada a sua rotação para 1700 rpm, dessa maneira houve uma variação no valor de tensão de 220 V para 242 V, dessa maneira foi enviado um alarme de sobretensão e consequentemente enviado uma mensagem ao Telegram, conforme Figura 70.

Observa-se que, além do valor de sobretensão, a mensagem do Telegram apresenta a *tag* do motor, o local que o mesmo se encontra, o horário do alarme, disponibiliza uma opção para reconhecer o alarme e também disponibiliza um link para acessar o histórico com os valores de tensão e os alarmes de sobretensão, na página da plataforma ThingSpeak.

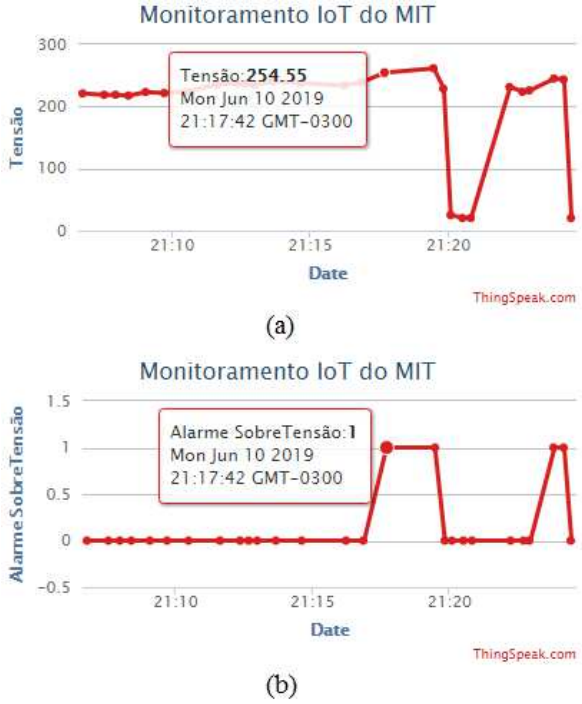
Figura 70 - Mensagem do Telegram com alarme de Sobretensão.

Fonte: Própria autoria

7.2.2. ThingSpeak

Foram disponibilizados no ThingSpeak o gráfico contendo os valores de tensão, Figura 71(a), e o gráfico com o histórico dos alarme de sobretensão, Figura 71(b). No gráfico de alarme, quando o valor estiver em “1” significa que o alarme de sobretensão foi ativado.

Figura 71– Registro de Sobretensão, no ThingSpeak.



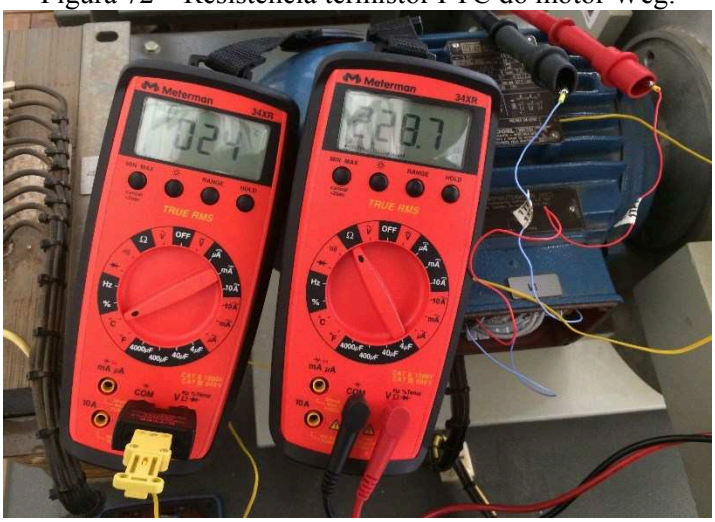
Fonte: Própria autoria.

7.3. Temperatura Alta

Para o teste de Temperatura Alta foi utilizado o termistor PTC presente nos enrolamentos do motor, conforme descrito no tópico 5.2.2.

O termistor PTC possui uma resistência em torno de 230 Ω, em temperatura ambiente (24°C), conforme medições realizadas e registradas na Figura 72.

Figura 72 – Resistência termistor PTC do motor Weg.



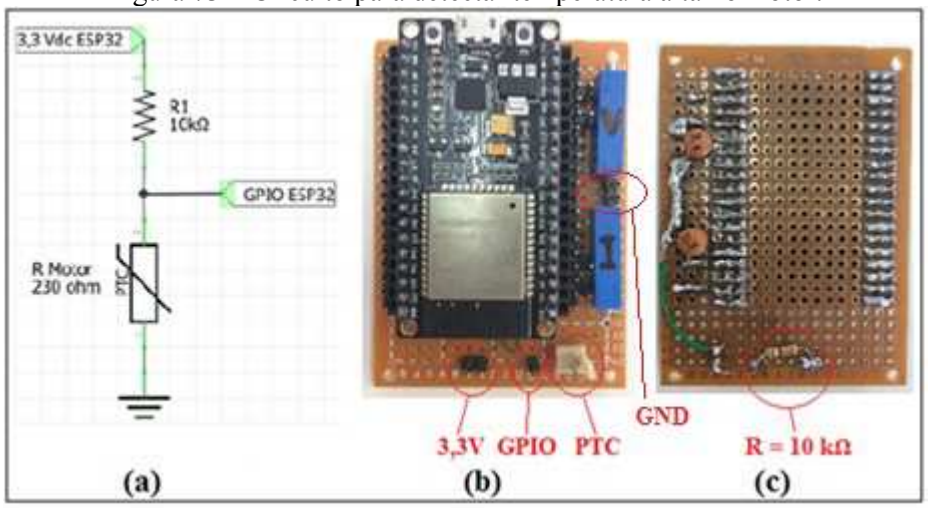
Fonte: Própria autoria.

Uma característica desse termistor é que, quando ocorre uma elevação de temperatura acima de seu limite de ruptura, ocorrerá uma variação abrupta na resistência do sensor PTC e com isso irá interromper a corrente no circuito (CATÁLOGO WEG, 2005).

Baseando-se nessas informações, foi projetado um circuito para detectar temperatura alta no motor, com o ESP32, conforme Figura 73.

Em situação normal (motor na faixa de temperatura aceitável) será setado “0” em uma entrada digital do ESP32, e caso ocorra uma elevação de temperatura acima do limite, será setado “1” na entrada digital e consequentemente será enviado um alarme via Telegram ao usuário informando a situação, e também o alarme será registrado no ThingSpeak.

Figura 73 – Circuito para detectar temperatura alta no motor.



Fonte: Própria autoria.

7.3.1. Telegram

A mensagem enviada ao aplicativo Telegram, conforme Figura 74, informa no centro da mensagem que o alarme foi de Temperatura Alta. Além disso, também informa a *tag* do motor, o local que o mesmo se encontra, o horário do alarme, disponibiliza uma opção para reconhecer o alarme e também disponibiliza um link para acessar o histórico de alarmes de temperatura alta, na página da plataforma ThingSpeak.

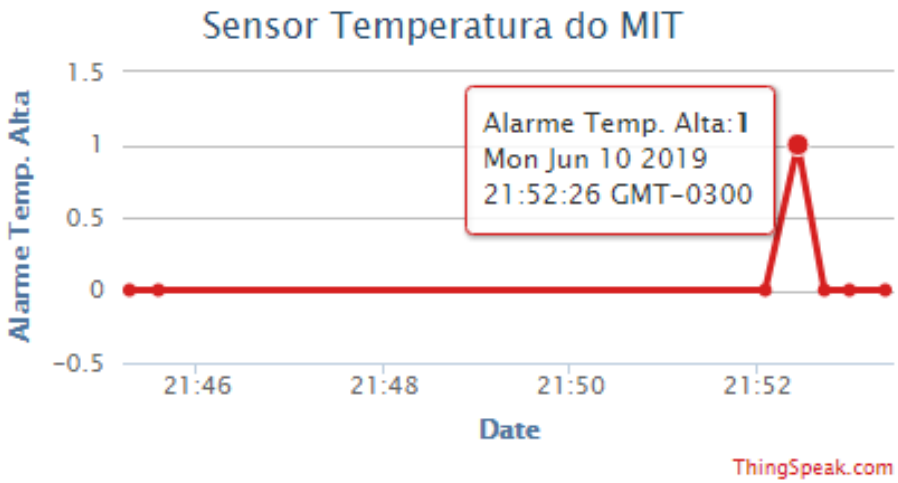
Figura 74 – Mensagem do Telegram com alarme de Temperatura Alta.

Fonte: Própria autoria.

7.3.2. ThingSpeak

A plataforma ThingSpeak registra e mantém um histórico das vezes que foi enviado um alarme de temperatura alta, com a informação do horário, dia, mês, ano e também fuso horário. Quando o gráfico estiver em “1” significa que o alarme foi ativado, e em “0” que a temperatura do motor está normal, conforme Figura 75.

Figura 75 – Registro de Temperatura Alta, no ThingSpeak.



Fonte: Própria autoria.

7.4. Conclusões e Resultados do Capítulo

Neste capítulo foram reforçados e definidos os critérios de alarmes para sobrecorrente, sobretensão e temperatura alta. Também foram apresentados os resultados obtidos em seus respectivos testes.

Os testes práticos finais foram feitos utilizando as duas plataformas microcontroladas ESP32 em conjunto, com isso não apresentaram os conflitos encontrados nos primeiros testes, quando foi utilizado apenas um ESP32.

Portanto, após resolver os conflitos e realizar vários testes verificou-se pelas imagens das mensagens recebidas pelo aplicativo Telegram e pelos gráficos da plataforma ThingSpeak, apresentadas neste capítulo, que os testes realizados confirmaram o que propõe o trabalho, que é o monitoramento de corrente, tensão e temperatura aplicando *IoT*.

8. CONSIDERAÇÕES FINAIS

Neste capítulo foram apresentadas as conclusões sobre o trabalho e também sugeridas algumas propostas para trabalhos futuros, que possam advir do trabalho em questão.

8.1. Conclusões

Neste trabalho foi apresentado e desenvolvido um sistema de aquisição e monitoramento de corrente, tensão e temperatura em motor de indução trifásico, aplicando a tecnologia *IoT*, projetado de forma a ser um sistema de baixo custo.

A plataforma microcontrolada ESP32, utilizada neste trabalho, mostrou-se eficiente e cumpriu com o que lhe era designado, como aquisitar os dados, executar o código, processar e realizar a transmissão pela internet utilizando o *WiFi*. Porém, é importante observar que apareceram alguns conflitos quando os testes foram feitos utilizando apenas um ESP32 para rodar todo o código, e que esses conflitos foram resolvidos quando se utilizou dois ESP32 e dividiu o código entre eles, ficando o primeiro ESP32 responsável pela aquisição e cálculos iniciais, e o segundo ESP32 para aplicação das condições de alarme e transmissão dos dados para internet.

O Módulo de Aquisição e Condicionamento de Sinais de Tensão e Corrente, também se mostrou eficaz com o seu propósito, que era o de condicionar os sinais de corrente e tensão para ao ESP32. Contudo foi necessário a instalação de um circuito auxiliar com um filtro RC, para um melhor tratamento dos níveis de sinais.

A bancada didática da WEG atendeu muito bem o propósito do trabalho, pois a mesma é composta por um motor de indução trifásico e dispositivos como inversor de frequência, ajuste de carga por meio do freio de Foucault e sensor de temperatura interno ao motor, que possibilitaram simular sobretensão, sobrecorrente e temperatura alta. Observando que a bancada conta com um inversor PWM (não senoidal) e, por isso, foi utilizado um filtro RC adicional como uma solução para o monitoramento.

As plataformas para armazenamento dos dados em nuvem também atenderam aos seus propósitos: o ThingSpeak em registrar e disponibilizar, em forma de gráfico, os valores medidos de tensão e corrente, e também os alarmes de sobretensão, sobrecorrente e temperatura alta; e o Telegram em enviar alertas a smartphones de usuários cadastrados, informando quando um evento de sobrecorrente, sobretensão ou temperatura alta ocorrer. O Telegram atendeu muito bem, se mostrando uma excelente plataforma, disponibilizando opções para criar “bots” de acordo com a aplicação necessária ao projeto. Plataforma segura, prática, disponível para vários

sistemas operacionais, além de ser *opensource*. O ThingSpeak também é uma excelente plataforma, porém possui algumas limitações em sua versão grátis, como o número de canais que é limitado a quatro e o registro de eventos que é limitado a um intervalo mínimo de quinze segundos.

Conclui-se que, a aplicação de internet das coisas no monitoramento de corrente, tensão e temperatura em motor de indução trifásico, utilizando sistemas de aquisição de baixo custo, composto por *hardwares* acessíveis e *softwares open source*, se mostrou eficaz em seu propósito. Dessa maneira, é possível considerar que há potencial nesse projeto, e que o mesmo pode ser melhorado com o intuito de se tornar um produto comercial para atender pequenas empresas, por exemplo.

8.2. Propostas para trabalhos futuros

Baseado nos estudos realizados, e observando que melhorias podem ser realizadas com o intuito de aprimorar o projeto, apresentam-se como sugestões para trabalhos futuros:

- Implementar o monitoramento das três fases do motor, para uma segurança extra do sistema;

- Estudo de outras plataformas para visualização de gráficos com servidores em nuvem, como por exemplo, Grafana®, AWS da Amazon® e Azure da Microsoft®;

- Estudo para implementar o monitoramento com um ESP32 apenas;

- Estudo de um sensor e sistema de monitoramento de temperatura contínuo, e não somente alarmes;

- Implementação de sensores para monitoramento e aquisição de outras condições do MIT como, por exemplo, vibração;

- Estudo de outro tipo de sensor de corrente e tensão, que seja menor por exemplo, visando a criação de um protótipo mais comercial;

- Estudo e elaboração de uma fonte de tensão menor, para alimentar os módulos de aquisição e a plataforma microcontrolada, também visando um protótipo que possa ser comercializado.

REFERÊNCIAS

ALMEIDA, A. T. et al. Comparative analysis of IEEE 112-B and IEC 34-2 efficiency testing standards using stray load losses in low voltage three-phases, cage inductionmotors. **IEEE Trans. Ind. Appl.**, 38, n. 2, 2002. 608 - 614. https://doi.org/10.1109/28.993186.

ANDREOLA, A. T. et al. Low cost data acquisition system for wind prospecting. **12th IEEE International Conference on Industry Applications (INDUSCON)**, 2016. 1-6. https://doi.org/10.1109/INDUSCON.2016.7874463.

ARDUINO HOME PAGE. What is Arduino?, 2019a. Disponivel em: <https://www.arduino.cc/>. Acesso em: 21 Março 2019.

ARDUINO SOFTWARE PAGE. Software, 2019b. Disponivel em: <https://bit.ly/2kdak6c>. Acesso em: 22 Março 2019.

ARDUINO STORE PAGE. Arduino MKR1000 WiFi, 2019c. Disponivel em: <https://bit.ly/2zC7V0U>. Acesso em: 22 Março 2019.

ARDUINO UPLOADS. MKR1000 Schematic, 2016. Disponivel em: <https://bit.ly/2HGNdkh>. Acesso em: 22 Março 2019.

BEZERRA, J. Revolução Industrial, 2018. Disponivel em: <https://www.todamateria.com.br/revolucao-industrial/>. Acesso em: 30 Julho 2018.

BNDES. Estudo “Internet das Coisas: um plano de ação para o Brasil”, 2017a. Disponivel em: <https://bit.ly/2kfqRJ4>. Acesso em: 27 Fevereiro 2019.

BNDES. **Produto 8: Relatório do Plano de Ação**. BNDES. [S.l.], p. 65. 2017b.

BNDES. BNDES Pilotos IoT, 2018. Disponivel em: <https://bit.ly/2JMZCEK>. Acesso em: 27 Fevereiro 2019.

CARVALHO, D. P. D. **Monitoramento wireless de eficiência e condição de operação de motores de indução trifásicos**. Universidade Federal de Uberlândia. Uberlândia, MG, p. 116. 2010.

CATALÓGO NODEMCU ESP32. **Introduction to NodeMcu ESP32**. Einstronic. [S.l.], p. 4. 2017.

CATÁLOGO WEG. **Motores Elétricos - Linhas de Produtos - Características - Especificações - Instalações - Manutenções**. Jaraguá do Sul, SC: WEG, 2005.

COTRIM, G. **História Global - Brasil e Geral**. 8°. ed. São Paulo: Saraiva, 2005.

CUNHA, A. F. Sistemas Embarcados. **Saber Eletrônica**, n. 414, 2007.

CUNHA, M. J. D. et al. Proposal for an IoT architecture in industrial processes. **12th IEEE International Conference on Industry Applications (INDUSCON)**, 2016. 1-7.

DMD SOLUTIONS. Indústria 4.0: Uma Visão da Automação Industrial, 2019. Disponivel em: <https://dmdsolutions.com.br/2019/05/01/industria-4-0-uma-visao-da-automacao-industrial/>. Acesso em: 10 Maio 2019.

ENCICLOPEDIA DE CARACTERÍSTICAS. Segunda Revolución Industrial, 2017. Disponivel em: <https://www.caracteristicas.co/segunda-revolucion-industrial/>. Acesso em: 30 Julho 2018.

ESPRESSIF ESP32. ADC2 support, 2017. Disponivel em: <https://www.esp32.com/viewtopic.php?t=1822#p10097>. Acesso em: 12 Fevereiro 2019.

ESPRESSIF SYSTEMS. Datasheet: NodeMCU v3 LoLin, 2016. Disponivel em: <https://bit.ly/2HzKARb>. Acesso em: 22 Março 2019.

ESPRESSIF SYSTEMS. ESP8266EX Datasheet, 2018. Disponivel em: <https://bit.ly/2ERZctm>. Acesso em: 22 Março 2019.

ESPRESSIF SYSTEMS. Datasheet: ESP-32 DevKitC, 2019a. Disponivel em: <https://bit.ly/2Tz22qC>. Acesso em: 22 Março 2019.

ESPRESSIF SYSTEMS. ESP32 WROOM Series, 2019b. Disponivel em: <https://bit.ly/2CBvK8U>. Acesso em: 20 Março 2019.

FABIO RODRIGUES. A jornada rumo à maturidade da Internet das Coisas na Indústria 4.0. **Novida**, 2018. Disponivel em: <https://novida.com.br/industria/jornada-rumo-maturidade-da-internet-das-coisas-na-industria-4-0/>. Acesso em: 16 Agosto 2018.

FAGUNDES, F. D. et al. Simulação de controle de processo industrial utilizando protocolo de comunicação Zigbee. **Educação & Tecnologia**, v. 19, n. 2, p. 59-74, 2014.

FILIPEFLOP. **Guia IoT para iniciantes em Eletrônica**. Florianópolis/SC, p. 55. 2019.

FITZGERALD, A. E.; CHARLES KINGSLEY, J.; UMANS, S. D. **Máquinas Elétricas**. 6. ed. [S.l.]: Bookman, 2006.

GERBERT, P. et al. Industry 4.0: The Future of Productivity and Growth in Manufacturing Industries, 2015. Disponivel em: <https://www.bcg.com/pt-br/publications/2015/engineered_products_project_business_industry_4_future_productivity_growth_manufacturing_industries.aspx>. Acesso em: 24 Agosto 2018.

GITHUB. ADC2 Channel cannot be used when WiFi is in use, 2017. Disponivel em: <https://github.com/espressif/arduino-esp32/issues/440>. Acesso em: 13 Fevereiro 2019.

GITHUB. Espressif IoT Development Framework, 2018. Disponivel em: <https://github.com/espressif/esp-idf/blob/master/components/driver/include/driver/adc.h>. Acesso em: 14 Maio 2019.

GODOY, W. F. et al. Application of intelligent tools to detect and classify broken rotor bars in three-phase induction motors fed by an inverter. **IET Electric Power Applications**, 10, n. 5, 02 June 2016. 430 - 439. https://doi.org/10.1049/iet-epa.2015.0469.

GOUNDAR, S. S. et al. Real time condition monitoring system for industrial motors. **2nd Asia-Pacific World Congress on Computer Science and Engineering (APWC on CSE)**, 23 May 2015. 1-9. https://doi.org/10.1109/APWCCSE.2015.7476232.

HANITSCH, R. Energy Efficient Electric Motors. **RIO 02 - World Climate & Energy Event, Rios de Janiero**, 2002. 6 - 11.

HONEYWELL. **Hall Book: Hall effect sensing and application**. Illinois, USA. s.d.

ISA. **The ISA100 Standards: Overview & Status**. [S.l.]. 2008.

ISA99. **Industrial Automation and Control Systems Security**. ISA. EUA. 2002.

KAGERMANN, H.; WAHLSTER, W.; HELBIG, J. **Recommendations for implementing the strategic initiative INDUSTRIE 4.0**. National Academy Of Science and Enginneering. Frankfurt, p. 82. 2013.

KOLBAN, N. **Kolban's Book on ESP32 & ESP8266**. Texas, USA: [s.n.], 2016.

KOYANAGI, F. ESP32: Como instalar na IDE do Arduino. **Fernando K Tutoriasis, Tecnologia e Tendências**, 2018. Disponivel em: <https://www.fernandok.com/2018/02/esp32-como-instalar-na-ide-do-arduino.html>. Acesso em: 14 Junho 2018.

KOYANAGI, F. ESP32: Detalhes internos e pinagem. **Fernando K Tutoriais, Tecnologia e Tendências**, 2019. Disponivel em: <https://www.fernandok.com/2018/03/esp32-detalhes-internos-e-pinagem.html>. Acesso em: 20 Abril 2019.

LAMPKIN, V. et al. **Building Smarter Planet Solutions with MQTT and IBM WebSphere MQ Telemetry**. [S.l.]: Redbooks, 2012.

LEMOS, B. S. et al. Estudo Comparativo de Middlewares para Internet das Coisas usando Plataformas Livres no Monitoramento de Ambientes. **XXXIV Simpósio Brasileiro de Telecomunicações – SBrT2016**, SANTARÉM, PA, 30 Agosto 2016.

LIBERO TECNOLOGIA. Come aiutare i lavoratori a reggere l'impatto dell'Industria 4.0, 2017. Disponivel em: <https://tecnologia.libero.it/come-aiutare-i-lavoratori-a-reggere-limpatto-dellindustria-4-0-14980>. Acesso em: 10 Maio 2019.

LUGLI, A. B.; SANTOS, M. M. D. Redes Industriais: Evolução, Motivação e Funcionamento. **InTech**, p. 5p, 2011.

LWT SISTEMAS. Terceira Revolução Industrial, 2018. Disponivel em: <https://www.lwtsistemas.com.br/industria-40-quarta-revolucao-industrial/terceira-revolucao-industrial/>. Acesso em: 30 Julho 2018.

MATHWORKS. ThingSpeak Support from Desktop MATLAB, 2019. Disponivel em: <https://www.mathworks.com/hardware-support/thingspeak.html>. Acesso em: 08 Maio 2019.

NASCIMENTO, L. D. O. Indústria 4.0 – Transformação e Desafios para o Cenário Brasileiro. **Unespciência**, 2018. Disponivel em: <http://unespciencia.com.br/2018/02/01/industria-93/>. Acesso em: 03 Junhi 2019.

NBR 7094. **Máquinas elétricas girantes - Motores de indução - Especificação**. ABNT. Rio de Janeiro, RJ, p. 50. 1996.

NODEMCU HOME PAGE. NodeMcu Connect Things EASY, 2018. Disponivel em: <https://www.nodemcu.com/index_en.html>. Acesso em: 22 Março 2019.

NUPEP. **Módulos de Aquisição e Condicionamento de sinais de Tensão e Corrente**. UFU. Uberlândia, MG, p. 5. 2011.

OLIVEIRA, C. A. D. **Plataforma para Ensaios de Motores de INdução Trifásicos e Simulação de Cargas Mecânicas: Acionamento, Operação e Monitoramento com Auxílio de Fonte Programável**. Universidade Federal de Uberlândia (UFU). Uberlândia, p. 95. 2018.

OLIVEIRA, V. G.; NIIZU, F. Y.; BASSETO, F. Internet das Coisas. **O Setor Elétrico**, n. 132, p. 32-35, 2017. ISSN 1983-0912.

PENA, R. A. Trabalho na Terceira Revolução Industrial, 2013. Disponivel em: <https://brasilescola.uol.com.br/geografia/trabalho-na-terceira-revolucao-industrial.htm>. Acesso em: Julho 30 2018.

PINHEIRO FILHO, R. F. **Estudo de um Sistema de Frenagem Eletromagnética Empregando Correntes Parasitas**. UFRN. Natal, RN. 2014. (267).

RAFAEL BITENCOURT. Valor Econômico, 27 Dezembro 2018. Disponivel em: <https://www.pressreader.com/brazil/valor-economico/20181227/281917364196938>. Acesso em: 27 Fevereiro 2019.

RANDOM NERD TUTORIALS. How to Install the ESP8266 Board in Arduino IDE, 2015. Disponivel em: <https://randomnerdtutorials.com/how-to-install-esp8266-board-arduino-ide/>.

RIEGO, H. B. **Redes sem fio na indústria de processos: oportunidades e desafios**. USP. São Paulo, SP. 2009. (101).

SANTOS, C. M. D. O. **Uma análise do papel dos designers na Internet das Coisas**. Dissertação Mestrado, PUC-Rio. Rio de Janeiro, RJ, p. 108. 2017.

SANTOS, J. V. D.; LUGLI, A. B. Comparativo e Estudo das Tecnologias Wireless para Automação Industrial. **Revista C & I (Controle & Instrumentação)**, São Paulo, SP, v. 15, p. 40-46, 2013.

SILVEIRA, C. B. O Que é Indústria 4.0 e Como Ela Vai Impactar o Mundo, 2017. Disponivel em: <https://www.citisystems.com.br/industria-4-0/>. Acesso em: 27 Julho 2018.

SORDAN, J. E. Fatec Sertãozinho. **O desafio da indústria 4.0 para as empresas brasileiras**, 2019. Disponivel em: <https://www.fatecsertaozinho.edu.br/blog/o-desafio-da-industria-4-0-para-as-empresas-brasileiras>. Acesso em: 03 Junho 2019.

SOUZA, D. J. D. **Desbravando o PIC:** Baseado no microcontrolador PIC 16F84. 5°. ed. São Paulo: Érica, 2000.

SPARKFUN. Serial Communication, 2016. Disponivel em: <https://learn.sparkfun.com/tutorials/serial-communication>. Acesso em: 24 Junho 2019.

TELEGRAM. Bots: An introduction for developers, 2019b. Disponivel em: <https://core.telegram.org/bots>. Acesso em: 06 Maio 2019.

TELEGRAM HOME PAGE. Telegram: a new era “Telegram: a new era, 2019a. Disponivel em: <https://telegram.org/>. Acesso em: 20 March 2019.

TEXAS INSTRUMENTS. Datasheet: LM35 Precision Centigrade Temperature Sensors. **TI**, 2017. Disponivel em: <http://www.ti.com/lit/ds/symlink/lm35.pdf>. Acesso em: 22 March 2019.

THINGSPEAK. Cadastro no ThingSpeak, 2019c. Disponivel em: <https://thingspeak.com/users/sign_up>. Acesso em: 20 March 2019.

THINGSPEAK. Sign In to ThingSpeak, 2019d. Disponivel em: <https://thingspeak.com/login>. Acesso em: 20 March 2019.

THINGSPEAK HOME PAGE. Understand Your Things: The open IoT platform with MATLAB analytics, 2019a. Disponivel em: <https://thingspeak.com/>. Acesso em: 20 March 2019.

THINGSPEAK LEARN MORE. Learn More About ThingSpeak, 2019b. Disponivel em: <https://thingspeak.com/pages/learn_more>. Acesso em: 20 March 2019.

WEG CFW09. **Manual Inversor de Frequência**. WEG. Jaraguá do Sul, SC, p. 329. 2006.

WEG CFW09. **CFW09 - Inversores de Frequência**. WEG. Jaraguá do Sul - SC, p. 24. 2012.

WORLD ECONOMIC FORUM. **Readiness for the Future of Production Report 2018**. Committed To Improving The State Of The World. Geneva. 2018. (254).

APÊNDICE A

Código utilizado no primeiro ESP32, que possui a atribuição de aquisitar e tratar os dados para envio ao segundo ESP32 via serial. O código segue comentado.

```
/* CÓDIGO 01: O código 01, executado no primeiro ESP32, é responsável pela
aquisição e condicionamento dos dados. O mesmo realiza a leitura das gran-
dezas elétricas por meio de interrupção do timer, em seguida é realizado
cálculos para encontrar os valores médios, máximos e rms de tensão e cor-
rente. Os valores rms são transmitidos para o segundo ESP32 pela serial 2,
ou seja comunicação serial TX/RX.

Programa desenvolvido por: Artur de Almeida Rios e Prof. Dr. Henrique José
Avelar (2019)*/


//-----Declarando bibliotecas e definindo parâmetros----------------------
#include"adc.h"//biblioteca para configurar os canais ADC do ESP32
//(necessária para atualizar atenuação)
#define analog_pin_ten 35 //define o pino 35 como entrada de tensão
#define analog_pin_cor 32 //define o pino 32 como entrada de corrente
#define Na 85.0 //número de amostras por período
#define Nper 59 //número de períodos entre cada apresentação de Vrms
#define TTimer 196 //período do timer (1/60Hz)/Na
#define B 275.78 //termo constante da função da curva de calibração
#define A 4108.2 //coeficiente de x da função da curva de calibração
#define invA 1.0/A //substitui divisão por multiplicação nos cálculos
#define RXD2 16 //pino RX da serial 2
#define TXD2 17 //pino TX da serial 2
//------------------------------------------------------------------------

//-----Declarando as variáveis de tensão----------------------------------
volatileint le_adc_t, int_count_t = 0, cont_per_t = 0, media_adc_t = 0;
volatilefloat va_t, va_rms_t = 0, va_max_t = 0, vrms_t, adc_curva_t;
//------------------------------------------------------------------------

//-----Declarando as variáveis de corrente--------------------------------
volatileint le_adc_c, int_count_c = 0, cont_per_c = 0, media_adc_c = 0;
volatilefloat va_c, va_rms_c = 0, va_max_c = 0, vrms_c, adc_curva_c;
//------------------------------------------------------------------------

//-----Declarando os ponteiros do timer-----------------------------------
hw_timer_t * timer =NULL;   //ponteiropara controlar o temporizador do
//timer (Tensão)
hw_timer_t * timer2 =NULL;  //ponteiro para controlar o temporizador do
//timer2 (Corrente)
//------------------------------------------------------------------------

//-----Interrupção para Tensão--------------------------------------------
void IRAM_ATTR ISR_Ten() {   //IRAM_ATTR aloca a rotina na DRAM
  //le_adc_t = analogRead(analog_pin_ten); //função lê ADC cru
  le_adc_t = adc1_get_raw(ADC1_CHANNEL_7); //função que permite configurar
//a atenuação ao ler a ADC- ADC1_CHANNEL_7 = GPIO35

  va_t = (float) (le_adc_t + B) * invA; //ajustando o valor lido no adc a
//partir da curva de calibração
```

```
if (int_count_t == 0) {
    va_rms_t = 0;
    media_adc_t = 0;
  }
  va_rms_t = va_rms_t + va_t * va_t; //cálculo do valor RMS para tensão
  media_adc_t = media_adc_t + le_adc_t;
  int_count_t++;
  if (int_count_t >= Na) {
    vrms_t =sqrt(va_rms_t / int_count_t);
    media_adc_t = media_adc_t / int_count_t;
    adc_curva_t = (media_adc_t + B) * invA;
    cont_per_t++;
    int_count_t = 0;
  }
  if (va_max_t < va_t) {
    va_max_t = va_t;
  }
}
//-------------------------------------------------------------------------

//-----Interrupção para Corrente-------------------------------------------
void IRAM_ATTR ISR_Cor() {   //IRAM_ATTR aloca a rotina na DRAM
  //le_adc_c = analogRead(analog_pin_cor); //função lê ADC cru
  le_adc_c = adc1_get_raw(ADC1_CHANNEL_4); //função que permite configurar
//a atenuação ao ler a ADC - ADC1_CHANNEL_4 = GPIO32

  va_c = (float) (le_adc_c + B) * invA; //ajustando o valor lido no adc a
//partir da curva de calibração
  if (int_count_c == 0) {
    va_rms_c = 0;
    media_adc_c = 0;
  }
  va_rms_c = va_rms_c + va_c * va_c; //cálculo do valor RMS para corrente
  media_adc_c = media_adc_c + le_adc_c;
  int_count_c++;
  if (int_count_c >= Na) {
    vrms_c =sqrt(va_rms_c / int_count_c);
    media_adc_c = media_adc_c / int_count_c;
    adc_curva_c = (media_adc_c + B) * invA;
    cont_per_c++;
    int_count_c = 0;
  }
  if (va_max_c < va_c) {
    va_max_c = va_c;
  }
}
//-------------------------------------------------------------------------

voidsetup() {
  Serial.begin(115200); //configura serial
  Serial2.begin(115200, SERIAL_8N1, RXD2, TXD2); //configura serial 2

  //-----Configura ADC-------------------------------------------------------
  adc1_config_width(ADC_WIDTH_BIT_12); //configura resolução das ADC1
  adc1_config_channel_atten(ADC1_CHANNEL_7, ADC_ATTEN_DB_0);
//atribui atenuação 0 ao pino 35
  adc1_config_channel_atten(ADC1_CHANNEL_4, ADC_ATTEN_DB_0);
//atribui atenuação 0 ao pino 32
  //-------------------------------------------------------------------------
```

```
//-----Configura timer1 para tensão-------------------------------------
  timer = timerBegin(1, 80,true);  //timer 1, div 80 => 1/(80MHZ/80) = 1us
  timerAttachInterrupt(timer,&ISR_Ten,true); //anexa a função ISR_Ten ao
//timer 1
  timerAlarmWrite(timer, TTimer,true); //define o alarme de acordo com o
//TTimer
  timerAlarmEnable(timer); //habilitar a interrupção
  //----------------------------------------------------------------------

//-----Configura timer2 para corrente-----------------------------------
  timer2 = timerBegin(2, 80,true); // 1/(80MHZ/80) = 1us
  timerAttachInterrupt(timer2,&ISR_Cor,true); //anexa a função ISR_Cor ao
//timer 2
  timerAlarmWrite(timer2, TTimer,true); //define o alarme de acordo com o
//TTimer
  timerAlarmEnable(timer2); //habilitar a interrupção
  //----------------------------------------------------------------------
} //fim do void setup()


voidloop() {
  //-----Declara variáveis-------------------------------------------------
  float v_t, vmax_t, v_c, vmax_c; //define variáveis de tensão e corrente
tipo float
  int conta_t, n_adc_t, conta_c, n_adc_c; //define variáveis de tensão e
corrente tipo int
  //----------------------------------------------------------------------

  //---Cáculos Tensão------------------------------------------------------
  noInterrupts(); //desativa a interrupção
  v_t = vrms_t; //não pode ficar dentro do "if" abaixo para não "resetar" o
//ESP32
  conta_t = cont_per_t;
  interrupts(); //reativa a interrupção
  if (conta_t == Nper) {
    noInterrupts(); //desativa a interrupção
    n_adc_t = media_adc_t;
    cont_per_t = 0;
    vmax_t = va_max_t;
    va_max_t = 0;
    interrupts(); //reativa a interrupção
    //Serial.print("le_adc_t_medio=");
    //Serial.print(n_adc_t); //lê a valor recebido do ADC, de tensão
    //Serial.print("adc_curva_t=");
    //Serial.print(adc_curva_t, 3); //valor calibrado de tensão
    //Serial.print("va_max_t=");
    //Serial.print(vmax_t, 3); //valor máximo da tensão
    Serial.print("va_rms_t= ");
    Serial.println(v_t); //valor da tensão rms
    Serial2.println(v_t); //valor da tensão rms enviado para a serial 2
  }
  //----------------------------------------------------------------------

  //-----Cálculos Corrente-------------------------------------------------
  noInterrupts(); //desativa a interrupção
  v_c = vrms_c; //não pode ficar dentro do "if" abaixo para não "resetar" o
//ESP32
  conta_c = cont_per_c;
  interrupts(); //reativa a interrupção
  if (conta_c == Nper) {
    noInterrupts(); //desativa a interrupção
```

```
    n_adc_c = media_adc_c;
    cont_per_c = 0;
    vmax_c = va_max_c;
    va_max_c = 0;
    interrupts(); //reativa a interrupção
    //Serial.print("le_adc_c_medio=");
    //Serial.print(n_adc_c); //lê a valor recebido do ADC, de corrente
    //Serial.print("adc_curva_c=");
    //Serial.print(adc_curva_c, 3); //valor calibrado de tensão
    //Serial.print("va_max_c=");
    //Serial.print(vmax_c, 3); //valor máximo da corrente
    Serial.print("va_rms_c= ");
    Serial.println(v_c); //valor da corrente rms
    Serial2.println(v_c); //valor da corrente rms enviado para a serial 2
  }
  //--------------------------------------------------------------------------
} //fim void loop()
```

APÊNDICE B

Código utilizado no segundo ESP32, que possui a atribuição de receber os dados do primeiro ESP32, realizar as condições de alarme, se conectar na internet e salvar os dados em plataformas na nuvem. O código segue comentado.

```
/* CÓDIGO 02: O código 02, executado no segundo ESP32, é responsável por
avaliar as condições de alarmes e transmitir os dados para a internet. O
código recebe os dados pela serial 2 (TX/RX), aplica um ganho na tensão e
na corrente, verifica as condições de sobretensão, sobrecorrente e tempera-
tura alta, e realiza a comunicação com servidores em nuvem, no caso o
ThingSpeak e o Telegram, para que esses dados possam ser visualizados em
forma de gráfico e alarmes possam ser recebidos em forma de mensagens pelo
aplicativo Telegram nos smartphones de usuários cadastrados.

Programa desenvolvido por: Artur de Almeida Rios e Prof. Dr. Henrique José
Avelar (2019)*/

//------BIBLIOTECAS-----------------------------------------------------------
#include<UniversalTelegramBot.h>//Biblioteca Telegram
#include<WiFi.h>                //Biblioteca WiFi
#include<WiFiClientSecure.h>    //Biblioteca Cliente SSL conexões seguras
#include<NTPClient.h>           //Biblioteca Relógio NTP
#include"esp_system.h"          //Biblioteca WatchgogTimer
//----------------------------------------------------------------------------

//------AJUSTES/ALARMES-------------------------------------------------------
#define ganhoV 320.0  //valor do ganho aplicado a tensão
#define ganhoI 10.0   //valor do ganho aplicado a corrente
#define sobretensao 242.0  //valor de sobretensão (1.1 * Vnominal)
#define sobrecorrente 6.48  //valor de sobrecorrente (1.5 * Inominal)
//----------------------------------------------------------------------------

//------Pinos Serial 02-------------------------------------------------------
#define RXD2 16 //pino RX da serial 2
#define TXD2 17 //pino TX da serial 2
//----------------------------------------------------------------------------

//------Variáveis Tensão------------------------------------------------------
int alarmeTriggered_t = 0; //disparo para o alarme de sobretensão
volatilebool telegramSensorPressedFlag_t =false; //flag sobretensão
int alarmeReconhecido_t = 0; //reconhece alarme de sobretensão
float le_adc_t, tensao ; //valor cru recebido e valor final de tensão
//----------------------------------------------------------------------------

//------Variáveis Corrente----------------------------------------------------
int alarmeTriggered_c = 0; //disparo para o alarme de sobrecorrente
volatilebool telegramSensorPressedFlag_c =false; //flag sobrecorrente
int alarmeReconhecido_c = 0; //reconhece o alarme de sobrecorrente
float corrente, le_adc_c; //valor cru e valor final de corrente
//----------------------------------------------------------------------------

//------Variáveis Temperatura-------------------------------------------------
int alarmeTriggered_temp = 0; //disparo para o alarme de temperatura alta
volatilebool telegramSensorPressedFlag_temp =false; //flag temp. alta
int alarmeReconhecido_temp = 0;  //reconhece alarme de temperatura alta
```

```
int pino_temp = 27;  //pino 27 que recebe o bit do sensor de temperatura
int le_gpio = 0;  //armazena o bit referente a temperatura
int status_temp;  //armazena o estado do alarme de temperatura alta
//--------------------------------------------------------------------------

//-------WiFi---------------------------------------------------------------
constchar* ssid ="iPhone"; //Definir o SSID da rede WiFi
constchar* password ="cruzeiro"; //Definir a senha da rede WiFi
//--------------------------------------------------------------------------

//------Variáveis Telegrama-----------------------------------------------
int Bot_mtbs = 1000;     //tempo médio entre a leitura das mensagens
long Bot_lasttime;       //a varredura das últimas mensagens foi concluída
bool Start =false;
//--------------------------------------------------------------------------

//--------ThingSpeak.com-----------------------------------------------
String writeAPIKey ="9YGD9WOX7T8YA60N";  //chave de escrita
constchar* server ="api.thingspeak.com"; //endereço servidor thingspeak
int cont = 0;//zera cont
//--------------------------------------------------------------------------

//--------Telegram----------------------------------------------------------
#define BOT_TOKEN "743208042:AAGK6eUc1ltDM0krNmZharovv1mESi_nGds"
//Token do Bot telegram "chave"
#define CHAT_ID "714608621"// Para obter o Chat ID, acesse Telegram
=>//usuario IDBot => comando /getid
WiFiClientSecureclient; // cliente SSL para conexões seguras
UniversalTelegramBot bot(BOT_TOKEN,client); //Objeto com os métodos para
//comunicação pelo Telegram
//--------------------------------------------------------------------------

//-------- Configurações de relógio on-line----------------------------------
WiFiUDP udp; //Cria um objeto da classe "UDP"
NTPClient ntp(udp,"a.st1.ntp.br",-3*3600,60000); //Cria um objeto "NTP"
String horario;  //Variável que armazenara a hora
//--------------------------------------------------------------------------

//------WatchdogTimer-----------------------------------------------------
constint wdtTimeout = 30000;  //tempo em ms para acionar o watchdog
hw_timer_t *timer =NULL; //ponteiro de controledo temporizador WDT
void IRAM_ATTR resetModule() { //IRAM_ATTR aloca a rotina na DRAM
  ets_printf("reboot\n");  //imprime na serial
  esp_restart();  //reinicia o chip
}
//--------------------------------------------------------------------------

voidsetup() {
  Serial.begin(115200);  //configura serial
  Serial2.begin(115200, SERIAL_8N1, RXD2, TXD2);  //configura serial 2
  pinMode(pino_temp,INPUT); //define “pino_temp” como entrada

  //---Necessário ao Telegram-----------------------------------------------
  WiFi.mode(WIFI_STA);      //Configura WIFI do ESP32 para modo estação
  WiFi.disconnect();        // desconecta o WIFI
  delay(100);               // atraso de 100 milisegundos
  //--------------------------------------------------------------------------

  //---WiFi-----------------------------------------------------------------
  WiFi.begin(ssid, password);  //Inicia o WiFi
  while (WiFi.status() != WL_CONNECTED) {  //Espera a conexão no router
```

```
    delay(500);  // atraso de 500 milisegundos
    Serial.print(".");  //imprime na serial
  }
  Serial.println("");
  Serial.print("Conectado na rede: ");
  Serial.println(ssid); //apresenta na serial a SSID da rede
  Serial.print("IP: ");
  Serial.println(WiFi.localIP()); //apresenta na serial o IP da rede
  //-------------------------------------------------------------------------

  //-----Atualizando as horas-----------------------------------------------
  ntp.begin();               // Inicia o NTP
  ntp.forceUpdate();         // Força o Update
  //-------------------------------------------------------------------------

  //------Watchdog---------------------------------------------------------
  timer = timerBegin(0, 80,true);                  //timer 0, div 80
  timerAttachInterrupt(timer,&resetModule,true);  //anexa a função
//resetModule ao timer
  timerAlarmWrite(timer, wdtTimeout * 1000,false); //define o alarme de
//acordo com o TTimer
  timerAlarmEnable(timer);                        //habilitar a interrupção
  //-------------------------------------------------------------------------
} //fim void setup()

//------Envio MSG Telegram-----------------------------------------------
void sendTelegramMessage(int TipoMsg) { //configura as mensagens a serem
//enviadas ao Telegram
  String message = ("ALARME: Motor M1 - Lab. CEFET Araxá"); //cabeçalho MSG
  message.concat("\n");
  message.concat("Horário: ");
  message.concat(horario); //envia horário atual do alarme na mensagem
  message.concat("\n");
  message.concat("\n");

  switch (TipoMsg) //controla o fluxo do programa, permitindo a execução de
//diferentes condições
  {
    case 1://Mensagem de Sobretensão
      message.concat("Sobretensão: ");
      message.concat(tensao); //valor da tensão
      message.concat(" V");
      message.concat("\n");
      message.concat("\n");
      message.concat("Clic p/ reconhecer: /ackTensao"); //comando para
//reconhecer alarme de sobretensão
      message.concat("\n");
      message.concat("\n");
      message.concat("Gráfico Histórico Tensão:");
      message.concat("\n");
      message.concat("https://bit.ly/2vA2P11"); //link para gráfico no
//ThingSpeak
      message.concat("\n");
      message.concat("Gráfico Histórico Alarmes:");
      message.concat("\n");
      message.concat("https://bit.ly/2DNs1p8"); //link para gráfico no
//ThingSpeak
      message.concat("\n");
      telegramSensorPressedFlag_t =false;
      break;
    case 2://Mensagem de Sobrecorrente
```

```
      message.concat("Sobrecorrente: ");
      message.concat(corrente); //valor da corrente
      message.concat(" A");
      message.concat("\n");
      message.concat("\n");
      message.concat("Clic p/ reconhecer: /ackCorrente"); //comando para
//reconhecer alarme de sobrecorrente
      message.concat("\n");
      message.concat("\n");
      message.concat("Gráfico Histórico Corrente:");
      message.concat("\n");
      message.concat("https://bit.ly/2PPt7Ws"); //link para gráfico no
//ThingSpeak
      message.concat("\n");
      message.concat("Gráfico Histórico Alarmes:");
      message.concat("\n");
      message.concat("https://bit.ly/2PMitzO"); //link para gráfico no
//ThingSpeak
      message.concat("\n");
      telegramSensorPressedFlag_c =false;
      break;
    case 3://Mensagem de Temperatura Alta
      message.concat("Temperatura Alta! ");
      message.concat("\n");
      message.concat("\n");
      message.concat("Clic p/ reconhecer: /ackTemp"); //comando para
//reconhecer alarme de temperatura alta
      message.concat("\n");
      message.concat("\n");
      message.concat("Gráfico Histórico Alarmes:");
      message.concat("\n");
      message.concat("https://bit.ly/2YwQjfG"); //link para gráfico no
//ThingSpeak
      message.concat("\n");
      telegramSensorPressedFlag_temp =false;
      break;
    default:
      message.concat("Msg inválida!\n");
  }
  if (bot.sendMessage(CHAT_ID, message,"Markdown")) //Markdown é uma //lin-
guagem simples de marcação,que converte o texto em HTML válido
  {
    //Serial.println("Mensagem Telegram enviada com sucesso");
  }
}
//------------------------------------------------------------------------

//------Condições para Envio Telegram-----------------------------------
void handleNewMessages(int numNewMessages)
{
  //Serial.print("Mensagem recebida = ");
  //Serial.println(String(numNewMessages));

  for (int i = 0; i < numNewMessages; i++)
  {
    String chat_id =String(bot.messages[i].chat_id);
    String text = bot.messages[i].text;

    String from_name = bot.messages[i].from_name;
    if (from_name =="") from_name ="Guest";
    {
```

```
      if (text =="/ackTensao") //comando Desativa Alarme de sobretensão
      {
        alarmeReconhecido_t = 1;  // desliga disparo do Alarme
        bot.sendMessage(chat_id,"Alarme Sobretensão Reconhecido!","");
//envia mensagem
      }
      elseif (text =="/ackCorrente")  //desativa Alarme de sobrecorrente
      {
        alarmeReconhecido_c = 1;  //desliga disparo do Alarme
        bot.sendMessage(chat_id,"Alarme Sobrecorrente Reconhecido!","");
//envia mensagem
      }
      elseif (text =="/ackTemp")  //desativa Alarme de temperatura alta
      {
        alarmeReconhecido_temp = 1;  //desliga disparo do Alarme
        bot.sendMessage(chat_id,"Alarme Temp. Alta Reconhecido!",""); //en-
via mensagem
      }
      elseif (text =="/start")  //comando para iniciar a comunicação
      {
        String welcome ="Alarme Motor IoT, "+ from_name +".\n\n";
        welcome +="Comandos:\n";
        welcome +="/start : Inicia a comunicação\n";
        welcome +="/ackCorrente : Reconhece Alarme Sobrecorrente\n";
        welcome +="/ackTensao : Reconhece Alarme Sobretensão\n";
        welcome +="/ackTemp : Reconhece Alarme Temperatura Alta\n";
        bot.sendMessage(chat_id, welcome,"Markdown");
      }
      else
      {
        bot.sendMessage(chat_id,"Comando inválido! Digite ou clic em /start
para ver as opções de comando!","");
      }
    }
  }
}
//------------------------------------------------------------------------


voidloop() {

  //------LEITURA DADOS SERIAL-----------------------------------------------
  if (Serial2.available() > 0) {
    le_adc_t =Serial2.parseFloat(); //ler dados enviados referente a //ten-
são do primeiro ESP32
    le_adc_c =Serial2.parseFloat(); //ler dados enviados referente a //cor-
rente do primeiro ESP32
    tensao = ganhoV * (le_adc_t);    //aplicando o ganho de tensão
    corrente = ganhoI * (le_adc_c);  //aplicando o ganho de corrente
    Serial.print(le_adc_t);       //imprime na serial o valor de tensão cru
    Serial.print(" ");
    Serial.print(tensao);       //imprime na serial o valor de tensão final
    Serial.print(" ");
    Serial.print(le_adc_c);     //imprime na serial o valor de corrente cru
    Serial.print(" ");
    Serial.print(corrente);   //imprime na serial o valor de corrente final
    Serial.print("   ");
    Serial.println(cont);     //imprime na serial o valor o número de loops
  }
  //------------------------------------------------------------------------
```

```
//-----THINGSPEAK-------------------------------------------------------------
//Obs. É necessário adicionar um "delay(500)" para que o ThingSpeak plote
//os dados no gráfico. Para que esse tempo não gere um atraso em todo o
//programa foi criado uma condição para que os dados sejam plotados de 1 em
//1 minuto ou quando acontecer um alarme, conforme abaixo:

  if ((cont >= 60)||(tensao > sobretensao)||(corrente > sobrecor-
rente)||(le_gpio == 1)) { //Condições para executar a função ThingSpeak
    WiFiClientclient; //Inicializa a biblioteca do cliente
    //Cria uma string de dados para enviar ao ThingSpeak
    Stringdata="field1="+String(tensao) +"&field2="+String(alarmeTrig-
gered_t) +"&field3="+String(corrente) +"&field4="+String(alarmeTriggered_c)
+"&field5="+String(status_temp);
    //Posta os dados para o ThingSpeak
    if (client.connect(server, 80)) {
      client.println("POST /update HTTP/1.1");
      client.println("Host: api.thingspeak.com");
      client.println("Connection: close");
      client.println("User-Agent: ESP32WiFi/1.1");
      client.println("X-THINGSPEAKAPIKEY: "+ writeAPIKey);
      client.println("Content-Type: application/x-www-form-urlencoded");
      client.print("Content-Length: ");
      client.print(data.length());
      client.print("\n\n");
      client.print(data);
    }
    delay(500); //E necessário um atraso antes de fechar o "client" para
//que os dados sejam plotados
    client.stop();//para o client
    Serial.println("ThingSpeak"); //Verificação se entrou dentro do if
    cont = 0; //zera o contador
  }
  //-----------------------------------------------------------------------

  //-----ALARME TENSÃO----------------------------------------------------
  if (tensao > sobretensao) {  //condição para alarme de sobretensão
    alarmeTriggered_t = 1;  //dispara o alarme de sobretensão

    if (alarmeReconhecido_t == 0)  //zera o alarme reconhecido
      telegramSensorPressedFlag_t =true; //seta flag como verdadeiro
  }
  else
  {
    alarmeTriggered_t = 0;  //zera o disparo de alarme de sobretensão
   alarmeReconhecido_t = 0;  //zera o reconhecimento de alarme
  }
  //-----------------------------------------------------------------------

  //-----ALARME CORRENTE--------------------------------------------------
  if (corrente > sobrecorrente) {  //condição para alarme de sobrecorrente
    alarmeTriggered_c = 1;  //dispara o alarme de sobrecorrente

    if (alarmeReconhecido_c == 0)  //zera o alarme reconhecido
      telegramSensorPressedFlag_c =true; //seta a flag como verdadeiro
  }
  else
  {
    alarmeTriggered_c = 0;  //zera o disparo de alarme de sobrecorrente
    alarmeReconhecido_c = 0; //zera o reconhecimento de alarme
  }
  //-----------------------------------------------------------------------
```

```
  //--------Temperatura-------------------------------------------------------
  le_gpio =digitalRead(pino_temp);//lê o estado do sensor de temperatura
  if (le_gpio == 1)  //condição para alarme de temperatura alta
  {
    Serial.println("Temperatura alta!"); //imprime msg temperatura alta
    status_temp = 1;  //armazena o estado de temperatura alta
    alarmeTriggered_temp = 1; //Dispara o alarme de temperatura alta

    if (alarmeReconhecido_temp == 0)  //zera o alarme reconhecido
      telegramSensorPressedFlag_temp =true;  //seta a flag como verdadeiro
  }
  else
  {
    Serial.println("Normal!");  //imprime mensagem de temperatura normal
    status_temp = 0;  //armazena o estado de temperatura normal
    alarmeTriggered_temp = 0;  //garante que o alarme não foi reconhecido
    alarmeReconhecido_temp = 0;  //sensor do Alarme foi acionado
  }
  //--------------------------------------------------------------------------

  //--------Controle Mensagem Telegram----------------------------------------
  if ( telegramSensorPressedFlag_t ) //realiza a condição quando a flag do
//alarme de sobretensão for verdadeiro
  {
    sendTelegramMessage(1);  //envia mensagem de sobretensão
  }
  if ( telegramSensorPressedFlag_c ) //realiza a condição quando a flag do
//alarme de sobrecorrente for verdadeiro
  {
    sendTelegramMessage(2);  //envia mensagem de sobrecorrente
  }
  if ( telegramSensorPressedFlag_temp ) //realiza a condição quando a flag
//do alarme de temperatura alta for verdadeiro
  {
    sendTelegramMessage(3);  //envia mensagem de temperatura alta
  }
  if (millis() > Bot_lasttime + Bot_mtbs)    //controlando as mensagens
  {
    int numNewMessages = bot.getUpdates(bot.last_message_received + 1);

    while (numNewMessages)                  //número de novas mensagens
    {
      //Serial.println("Mensagem recebida do Telegram");
      handleNewMessages(numNewMessages);
      numNewMessages = bot.getUpdates(bot.last_message_received + 1);
    }
    Bot_lasttime =millis();
  }
  //--------------------------------------------------------------------------

  //--------Outros Comandos---------------------------------------------------
  horario = ntp.getFormattedTime(); //Armazena na variável "horario", o ho-
rário atual.
  timerWrite(timer, 0); //alimenta o watchdog para não resetar o programa
(feed watchdog)
  cont = cont + 1; //incrementa o contador
}//fim void loop
```