FACULDADE DE TECNOLOGIA DE SÃO BERNARDO DO CAMPO
“ADIB MOISÉS DIB”

ANTONIO SAES MONTOIA
HENRIQUE ROSSI
NICOLAS SOUZA DE OLIVEIRA
THIAGO DA SILVA DONAIRE

MANUTENÇÃO PREDITIVA DE TERMOPARES POR MEIO DA IOT

São Bernardo do Campo - SP
Dezembro/2019

ANTONIO SAES MONTOIA
HENRIQUE ROSSI
NICOLAS SOUZA DE OLIVEIRA
THIAGO DA SILVA DONAIRE

MANUTENÇÃO PREDITIVA DE TERMOPARES POR MEIO DA IOT

Trabalho de Conclusão de Curso apresentado à Faculdade de Tecnologia de São Bernardo do Campo “Adib Moisés Dib” como requisito parcial para a obtenção do título de Tecnólogo em Automação Industrial.

Orientador: Prof. Dr. Cláudio Rodrigo Torres

São Bernardo do Campo - SP
Dezembro/2019

ANTONIO SAES MONTOIA
HENRIQUE ROSSI
NICOLAS SOUZA DE OLIVEIRA
THIAGO DA SILVA DONAIRE

MANUTENÇÃO PREDITIVA DE TERMOPARES POR MEIO DA IOT

Trabalho de Conclusão de Curso apresentado à Faculdade de Tecnologia de São Bernardo do Campo “Adib Moisés Dib” como requisito parcial para a obtenção do título de Tecnólogo em Automação Industrial.

Trabalho de Conclusão de Curso apresentado e aprovado em: _____/_____/2019

Banca Examinadora:

_____

Prof. Dr. Cláudio Rodrigo Torres, FATEC SBC – Orientador

_____

Prof. XXXXXXXXXXXXXXXXXX, FATEC SBC – Avaliador

_____

Prof. XXXXXXXXXXXXXXXXXX, FATEC SBC – Avaliador

Dedicamos nosso trabalho e esforços aos nossos amigos, pais, companheiras, e a todos os professores que nos ajudaram durante essa caminhada na trilha do conhecimento. Agradecemos a instituição FATEC por fazer a diferença em nossas vidas pessoais e profissionais. Obrigado.

Agradecemos ao Prof. Dr. Cláudio Rodrigo Torres pela ajuda e orientação durante a elaboração deste trabalho.

“A invenção é o produto mais importante do cérebro criativo do homem. O propósito final é o domínio completo da mente sobre o mundo material, o aproveitamento da natureza humana para as necessidades humanas”.

NIKOLA TESLA

RESUMO

A pesquisa tem como objetivo geral a elaboração de um algoritmo de detecção de falhas em termopares utilizando o Microcontrolador ESP32, no qual pode ser aplicado na viabilização da manutenção preditiva em processos industriais. Com o uso da eletrônica e da lógica de programação o projeto detecta falhas do termopar abrindo e entrando em curto, indicando ao usuário a necessidade da troca de forma preditiva, além de realizar a comunicação por nuvem que disponibiliza os dados na rede utilizando os conceitos de IOT e computação em nuvem. É apresentado também a elaboração do algoritmo, o processo de comunicação e criação da interface em nuvem, e a construção do protótipo de testes que executa o algoritmo e realiza simultaneamente o controle de temperatura PID, que viabiliza a simulação da detecção de falha. Para a elaboração deste trabalho as pesquisas bibliográficas forneceram dados para dar sustentação na elaboração do trabalho, bem como a metodologia cientifica que foi essencial na organização e direção fornecendo métodos e técnicas para a execução com sucesso do protótipo final, o qual se mostrou efetivo em sua operação.

Palavras-chaves: Termopar. Preditiva. Algoritmo. ESP32. IOT.

ABSTRACT

The research aims as a general objective to develop a thermocouple fault detection algorithm using the ESP32 Microcontroller, which can be applied to enable a predictive maintenance in industrial processes. Using electronics and programming logic, the project detects failures on thermocouples opening or shorting, indicating to the user the need for predictive exchange, as well communicating with the cloud and making the data available on the network using the concepts of IOT and cloud computing. It also presents the elaboration of the algorithm, the communication process, the creation of the cloud interface, and the construction of the test prototype that executes the algorithm and simultaneously performs the PID temperature control that enables the simulation of the fault detection. For the preparation of this work, bibliographical research was used, as well as technical and scientific articles that supported the elaboration of the prototype and concept of the objective problem solving, which brought knowledge to successfully execute the final prototype, which proved to be effective on their operation.

Keywords: Thermocouple. Predictive. Algorithm. ESP32. IOT.

LISTA DE FIGURAS

SUMÁRIO

INTRODUÇÃO

Atualmente o mundo tecnológico e produtivo cada vez mais exige a necessidade de controlar variáveis de processo e anteceder eventos para garantir o nível de competitividade em um mercado cada vez mais globalizado e competitivo.

Novas tecnologias como automatização de tarefas, supervisão e comunicação de dados de equipamentos vêm aumentando cada vez mais a eficácia do processo produtivo, afetando diretamente a qualidade dos produtos gerados.

Tendo em vista estes controles essenciais, testa-se a importância de elementos e tecnologias que possibilitem o controle em áreas estratégicas para o processo produtivo, a fim de garantir produtividade e qualidade ao processo.

Um dos componentes mais utilizados na medição de temperatura e controle é o termopar, afinal nele encontram-se possibilidades de inovações por ser um componente elementar do processo produtivo de diversas empresas. A manutenção do termopar é feita de forma planejada utilizando a manutenção preventiva por meio de troca agendada do componente.

Intervenções planejadas feitas por manutenções preventivas é uma estratégia de manutenção eficaz, porém com a troca do componente em intervalos fixos existe uma possibilidade de troca precoce mesmo com o item em boas condições, o que acarreta em interrupções desnecessárias e gastos mais frequentes.

Diante das argumentações apresentadas, o objetivo do projeto que se intitula Manutenção Preditiva de Termopares por meio da IOT é o desenvolvimento de um controlador de temperatura e um algoritmo de detecção das falhas, com suporte a Internet das Coisas (IOT) além do acesso aos dados remotamente e um controle do dispositivo feito pela nuvem. Tal projeto justifica-se pelo ganho da competitividade na utilização e frequência de trocas dos termopares, com redução de custo e redução nas paradas para manutenção.

Para a construção e desenvolvimento do projeto faz-se uso de Microcontrolador, amplificador de instrumentação, termopar, relé de estado sólido, resistência elétrica e IDE do Arduino.

Desta forma, o trabalho é dividido da seguinte maneira:

• Capítulo 1 – Fundamentação teórica: encontra-se as teorias que dão sustentação a construção e desenvolvimento do projeto Manutenção Preditiva de Termopares por meio da IOT;

• Capítulo 2 – Metodologia: é o caminho percorrido para desenvolver a pesquisa. Nela são descritos métodos e técnicas que operacionalizam os instrumentos;

• Capítulo 3 – Desenvolvimento do projeto: descreve passo a passo o desenvolvimento e construção do projeto.

Considerações finais: são descritos o objetivo proposto na introdução e sua justificativa, apontando as relações entre os fatos verificados e as teorias, conquistas alcançadas, pontos fortes e fracos e sugestões para futuros trabalhos.

1 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo são abordadas teorias de autores renomados que dão sustentação ao desenvolvimento e construção do projeto Manutenção Preditiva de Termopares por meio da IOT.

1.1 Breve histórico sobre manutenção

Udop (2007) enfatiza que a “Manutenção” tem origem no latim *manus tenere***,** que significa manter o que se tem. A atividade de manutenção tem seus relatos desde quando o homem deu início à utilização de pequenas ferramentas e equipamentos para fabricar algum produto, garantir o adequado funcionamento de algum sistema e ou preservar o estado de conservação de determinado patrimônio.

Kardec e Nascif (2009) enfatizam que o grande impulso da manutenção veio com a Revolução Industrial do século XVIII, com o surgimento das máquinas a vapor. Neste período, a manutenção era realizada pela própria operação, ainda sem uma separação e divisão de atividades e tarefas, onde pode ser descrito o ciclo de evolução da manutenção, em três fases ou gerações:

• 1ª geração: período anterior de 1930 a 1940, onde se tinha uma indústria com foco no volume de produção e com algumas características:

- Manutenção somente após a falha;
- Não preocupação com a causa raiz das falhas;
- Nenhuma sistematização da manutenção.

• 2ª geração: iniciada após a 2ª Guerra Mundial (1940). Este período desencadeou um grande aumento na demanda por diferentes tipos de produtos, e com isto, a necessidade do bom funcionamento dos equipamentos, com a minimização das quebras, objetivando buscar a maior disponibilidade possível destes recursos. Surgiu assim, uma geração com as seguintes características:

- Instalações mais complexas;
- Busca por maior confiabilidade e disponibilidade dos equipamentos;
- Busca pela causa raiz das falhas dos equipamentos visando eliminá-las para evitar a reincidência;
- Implementação da manutenção planejada;
- Surgimento da Manutenção Preventiva.

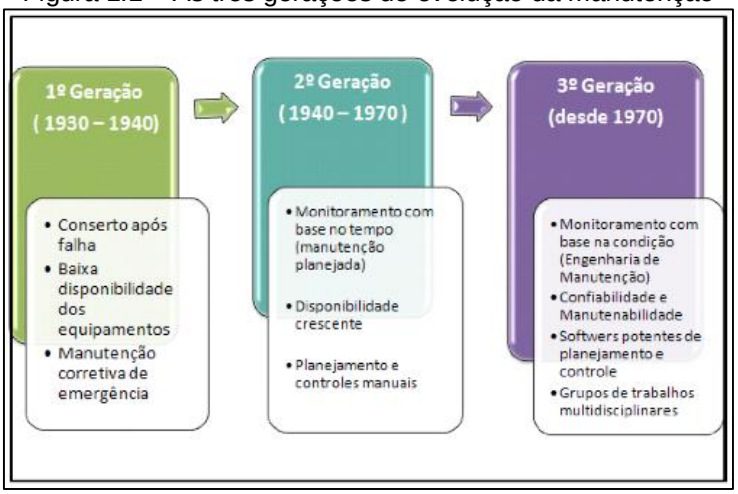
- 3ª geração: década de 70 trouxe mudanças significativas para a indústria.

A visão de controle de custos passa a ser fundamental para o resultado. A implementação do "*JIT- Just in time*" no mundo, com o foco na redução de estoques, como a principal perda das empresas, obriga as organizações a melhorar a gestão da manutenção, garantindo o bom desempenho e disponibilidade dos ativos. Outras características relevantes neste período:

- Busca da excelência na qualidade do produto;
- Maior rigor da legislação em relação à segurança e o meio ambiente;
- Maior nível de automação gerando equipamentos mais complexos;
- Busca pelo conceito "Zero" (defeitos, acidentes, quebras);
- Implementação do JIT;
- Surgimento da manutenção preditiva.

A Figura 1.1 ilustra as três gerações da manutenção:

Figura 1.1 – As três gerações de evolução da manutenção



Fonte: KARDEC E NASCIF, 2009, p.5

1.2 Produtividade e manutenção

Para Gonsalez (2008), produtividade é a relação entre os recursos empregados e os resultados alcançados. Ter alta produtividade é ter alcançado ótimos resultados, aproveitando bem a matéria prima, a capacidade das máquinas, o tempo e as habilidades das pessoas.

A manutenção não pode ser encarada como um setor que gera apenas gastos. Precisa ser vista com uma atividade estratégica que busca aumentar a confiabilidade e disponibilidade dos ativos. Uma falha no início do processo pode desencadear perdas ao longo do processo podendo chegar até o cliente final, comprometendo qualidade ou a entrega.

No Figura 1.2 mostra-se o desempenho de um equipamento genérico em função do tempo de operação, demonstrando a evolução do custo para reparo de acordo com o tipo de manutenção, onde a linha vermelha representa o custo para reparo, e a linha preta o desempenho do equipamento:

Figura 1.2 – Manutenção em função do tempo de intervenção

Fonte: adaptado de https://engeteles.com.br, 2019

A seguir destacamos os principais tipos de Manutenção:

- Corretiva: de acordo com Viana (2002) a manutenção corretiva tem como objetivo intervir nos equipamentos para corrigir falhas, quebras ou defeitos que já ocorreram e colocar o equipamento em operação novamente. A manutenção corretiva nem sempre precisa ter uma atuação de emergência. Esta atividade pode ser também planejada quando se tratar de reestabelecimento de padrões de desempenho, aspectos visuais e também necessidades identificadas em auditorias de inspeção, onde se perceba a deterioração de algum padrão. A manutenção “corretiva” pode então ser dividida em dois tipos:

  - Corretiva não planejada: deve ser executada imediatamente quando o equipamento ou sistema está fora de operação, com interrupção da produção devido à ocorrência de falha, causando, assim, prejuízos financeiros a organização. Nesta situação não há tempo para o planejamento das atividades. A equipe deve ser mobilizada para a resolução do problema;

  - Corretiva planejada ou programada: trata-se da realização da manutenção após a detecção de uma anomalia que pode causar alguma perda, porém sem a interrupção do processo de produção. A solução do problema pode ser programada para um momento onde se tem um intervalo de tempo sem produção. A correção pode ser realizada minimizando as perdas.

Conforme Almeida (2000) as desvantagens na aplicação da manutenção corretiva pode ser:

- Ocorrência da falha de forma inesperada, provocando grandes danos no planejamento das atividades da empresa;
- Aumento drástico nos custos com a geração de perdas com mão de obra, reprogramações, horas extras, fretes extras;
- Perdas de faturamento;
- Necessidade de manutenção de estoques altos de produtos acabados para garantir o abastecimento dos clientes e mercados devido à falta de confiabilidade dos equipamentos;
- Necessidade de manutenção de estoques de componentes de máquinas para as trocas inesperadas;

- Redução drástica na vida útil das máquinas e equipamentos.

- Manutenção Preventiva: conforme Almeida (2000), a correção preventiva é um tipo de manutenção planejada. Suas atividades são programadas para serem realizadas em períodos pré-estabelecidos conforme as necessidades. De acordo com Slack (2002), o objetivo da manutenção preventiva é evitar quebras ou falhas que podem interromper as atividades produtivas ou serviços, reduzir o desempenho do processo, criar riscos de acidentes, gerando alguma forma de perda ambiental. Desta forma para Almeida (2000) aponta que o planejamento da periodicidade desta manutenção deve levar em consideração:

- Regime de trabalho dos equipamentos;
- Informações do fabricante;
- Manutenção de registros e históricos de manutenção do equipamento;
- Consultas no histórico de equipamentos similares.

Conforme Almeida (2000), as vantagens em adotar a manutenção preventiva são as seguintes:

- Aumento da taxa de disponibilidade do equipamento;
- Aumento da vida útil dos equipamentos, com a redução da degradação;
- Redução de custos de produção com a minimização das perdas por paradas;
- Redução de custos com manutenção.
- Redução de custos com estoques de peças de reposição e produtos acabados;
- Redução dos riscos de acidentes.

Contudo, deve-se ter cuidado ao adotar a manutenção preventiva, pois se tende a aumentar o custo com material de reposição, trocas de componentes e intervenções desnecessárias nos equipamentos.

Kardec e Nascif (2009, p.51, grifos do autor), ainda destacam sobre a manutenção preventiva o seguinte: “A Experiência indica que mais intervenções do que o necessário serão feitas e/ou um número elevado de trocas de peças com “meia-vida” ainda em bom estado [...]”.

Manutenção Preditiva: conforme Kardec e Nascif (2009), trata-se de um modelo de manutenção mais moderno que necessita de algumas condições básicas para sua implementação. Seu principal objetivo é através de mecanismo como sensores, softwares, câmeras, e computadores, realizar monitoramento e controle do avanço da degradação ou desgaste de determinado componente ou partes de um equipamento, antevendo o limite aceitável de uso e, também, prever a condição futura. Esta atividade pode ser feita no local ou remotamente.

Conforme Viana (2002) destaca, as vantagens em adotar a manutenção preditiva são:

- Redução nas intervenções para manutenção corretiva;
- Redução no tempo de manutenção com diagnósticos sem paradas;
- Possibilidades de monitoramento dos equipamentos de forma remota reduzindo custos e obtendo maior eficácia;
- Maior assertividade nos diagnósticos;
- Maior confiabilidade dos equipamentos;
- Redução de custos com aumento da vida útil dos componentes dos equipamentos.

Kardec e Nascif (2009, p.51, grifos do autor), ainda destacam sobre a manutenção preditiva o seguinte: “[...] o número de intervenções cairá drasticamente, o consumo de sobressalentes também e o número de h/h será reduzido [...]”.

Ainda FEMP (2010) indica que a manutenção preditiva pode trazer até 35% de redução de tempo de inatividade, 70% de eliminação de avarias, 25% na otimização de custo com manutenção e 20% no aumento de produtividade da operação.

Conforme Kardec e Nascif (2009), as particularidades para a implementação da manutenção preditiva são:

- Necessidades de investimentos em equipamentos de detecção, os quais são compensados com o maior aproveitamento da vida do componente no equipamento;
- Necessidades de softwares, computadores e adequação dos ativos para possibilitar o monitoramento;
- Necessidade de mão de obra especializada para definição dos parâmetros a serem monitorados e conhecimentos na operação dos respectivos equipamentos;
- Os equipamentos devem estar dotados de recursos tecnológicos que possibilitem seu monitoramento através de medições.

Portanto, não existe um modelo de manutenção ideal, porem existe uma necessidade de análise detalhada do sistema produtivo para identificar qual delas é mais viável economicamente para a empresa.

1.3 Termopar

Alciatore e Histand (2014) destacam que o termopar é um sensor usado para medir a temperatura. Ele consiste em dois fios de arames constituídos de metais diferentes. Os arames são soldados juntos em uma extremidade criando uma junção. Esta junção é onde a temperatura é medida. Quando a junção sofre uma mudança na temperatura, uma tensão é criada.

Aguirre (2015) enfatiza que os termopares fazem com que ocorra uma diferença de tensão quando há mudanças de temperatura. A diferença de potencial produzido é proporcional à diferença de temperatura, podendo ser medida em um voltímetro (em milivolts). A relação da temperatura em função tensão é dada pela equação [1]

$$Emf = \int_{T_1}^{T_2} S_{12} \cdot dT = \int_{T_1}^{T_2} (S_1 - S_2) \cdot dT \qquad [1]$$

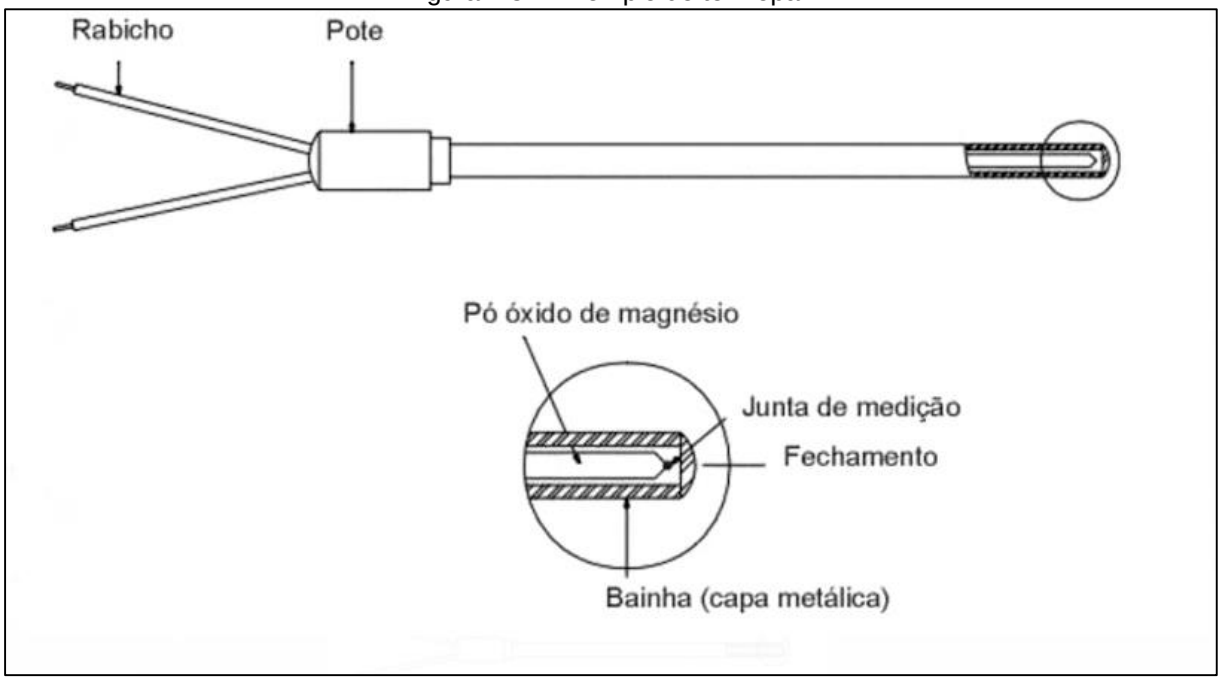
Onde,

Emf é Tensão (V) produzida. T 1 e T 2 são as temperaturas de referência e fim de medição, S12 (mV/°C) é chamado coeficiente de Seebeck do termopar e S1 (mV/°C) e S2 (mV/°C) são os coeficientes Seebeck dos dois termo elementos.

O coeficiente de Seebeck depende do material do qual o termopar é feito. Com a equação é possível notar que uma tensão nula é medida se os dois materiais forem feitos dos mesmos materiais. Logo, diferentes materiais são necessários para fazer um dispositivo sensor de temperatura.

Um exemplo de termopar pode ser verificado na figura 1.3:

Figura 1.3 – Exemplo de termopar



Fonte: https://www.cpilink.com/, 2019

Alciatore e Histand (2014) destacam que existem diversos tipos de termopares, cada um com suas características únicas em termos de faixa de temperatura, durabilidade, resistência à vibração, resistência química e compatibilidade de aplicativos. Sendo os termopares J, K, T e E os tipos mais comuns.

Podemos ver na figura 1.4 a variedade de termopares e suas características.

Figura 1.4 – Tipos de termopares e características

Thermocouple Types			
Type	**Composition**	**Sensitivity**	**Temperature range**
Type B	(+) Platinum - 30% Rhodium (–) Platinum - 6% Rhodium	5 to 10 µV/°C	+250 to +1820 °C
Type E	(+) Chromel (Ni-Cr) (–) Constantan (Cu-Ni)	40 to 80 µV/°C	–270 to +1000 °C
Type J	(+) Iron (–) Constantan (Cu-Ni)	50 to 60 µV/°C	–210 to +1200 °C
Type K	(+) Chromel (Ni-Cr) (–) Alumel (Ni-Al)	28 to 42 µV/°C	–250 to +1250 °C
Type N	(+) Nicrosil (Ni-Cr-Si) (–) Nisil (Ni-Si-Mg)	24 to 38 µV/°C	–250 to +1300 °C
Type R	(+) Platinum (–) Platinum - 13% Rhodium	8 to 14 µV/°C	–50 to +1768 °C
Type S	(+) Platinum (–) Platinum - 10% Rhodium	8 to 12 µV/°C	–50 to +1768 °C
Type T	(+) Copper (–) Constantan (Cu-Ni)	17 to 58 µV/°C	–250 to +400 °C

Fonte: http://www.mosaic-industries.com, 2019

Segundo Aguirre (2015), os termopares são usados em muitas aplicações industriais e científicas. Sendo encontrado em quase todos os mercados industriais, como: geração de energia, petróleo / gás, farmacêutica, biotecnologia, cimento, papel e celulose, etc

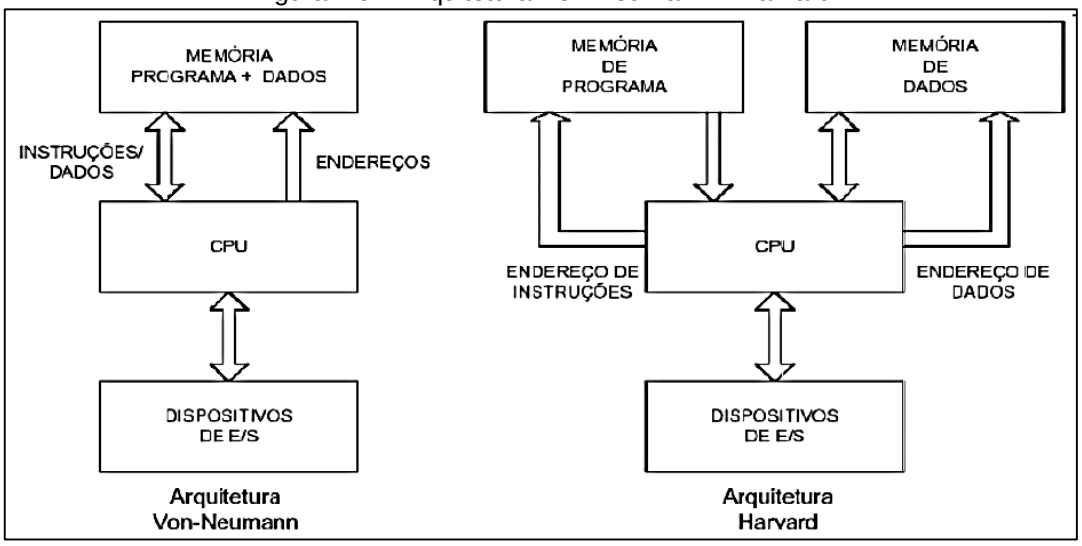
De acordo com Carlson (2017), para manutenções em termopares é indicado verificações de manutenção mensais, deste modo aplica-se comumente a manutenção preventiva e, com isso, tende-se a aumentar trocas de componentes e intervenções desnecessárias, aumentando, assim, o custo com material de reposição enquanto que intervenções planejadas feitas por manutenções preventivas tem-se uma estratégia eficaz, porém sua prática pode reduzir a vida média dos componentes.

1.4 Microcontroladores e arquitetura

Pereira (2010) explica que a base de Microcontroladores e Microprocessadores é a eletrônica digital. No entanto qualquer tipo de computador moderno é uma máquina inteiramente binária, ou seja, ela só é capaz de compreender dois tipos de informações, 0 ou 1, a razão para isso é que a matemática por trás dos computadores modernos é uma álgebra booleana.

Quanto à arquitetura destaca que um Microprocessador pode ser de dois tipos: com memórias separadas para o armazenamento de dados, conhecida como “arquitetura Harvard”, e outro com um espaço de memória unificado, onde são armazenadas tanto as instruções quanto os dados, conhecida como “Von Neumann”, conforme ilustra a Figura 1.5.

Figura 1.5 – Arquitetura Von Neumann x Harvard



Fonte: BORGES, 2009, p. 05

Oliveira (2017) relata que o Microcontrolador é um *chip* programável utilizado para controle de dispositivos eletrônicos e é construído de forma a disponibilizar no mesmo encapsulamento os seguintes dispositivos: CPU, memória RAM, Memória EEPROM, Pinos de I/O e dispositivos auxiliares.

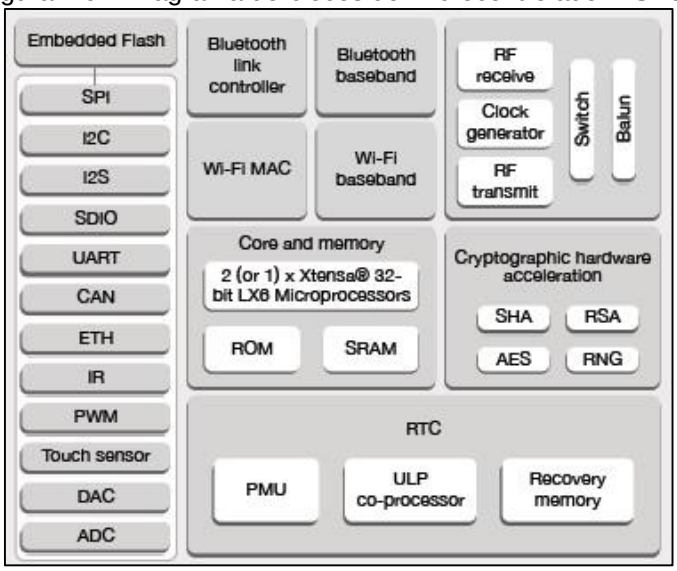
As interfaces de entrada e saída (I/O) tem como principal função em um Microcontrolador a integração diversos dispositivos, sendo as mais comuns conhecidas como GPIO (entrada e saída de propósito geral), podem ser configuradas como entradas ou saídas digitais e analógicas.

Ainda existe a interface de saída PWM que é uma saída digital, porém de pulso variável o que faz ela se comportar como uma saída analógica, sendo normalmente ligada em dispositivos de controle como relé ou transistor.

Espressif (2018), explica que O ESP32 é um Microcontrolador que trabalha com uma arquitetura de 32 bits, podendo conter um ou dois processadores Xtensa

LX6 com arquitetura Harvard. Ele possui *clock* interno de até 240 MHz, suporte a WiFi e Bluetooth, com um transmissor e receptor que trabalha com frequências de 2.4GHz, tendo Memória RAM de 512 KB, memória Flash de 16Mb, tem 48 pinos em sua arquitetura além da terra, 36 GPIO, nove interfaces de saída PWM, dezoito entradas ADC (analógico para digital), além de duas saídas DAC (digital para analógico), conforma ilustra a figura 1.6.

Figura 1.6 – Diagrama de blocos do Microcontrolador ESP32



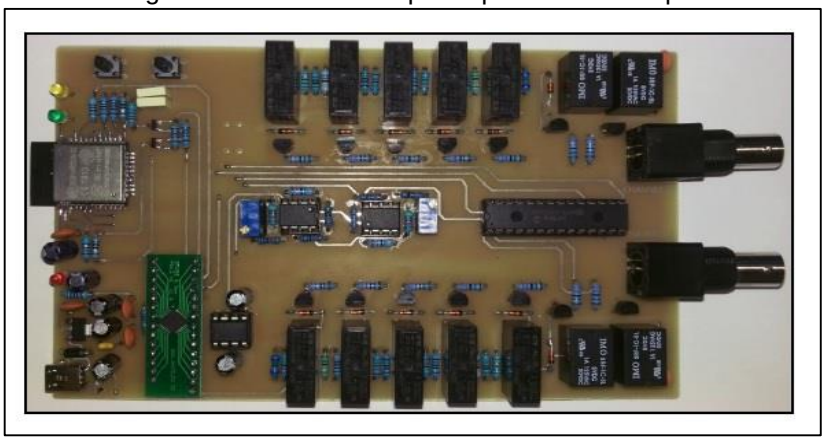
Fonte: ESPRESSIF, 2018, p.5

Quanto à comunicação, têm-se as interfaces seriais síncronas: SPI, I2C e I2S e a assíncrona: USART.

Mayer, Sharp e Vagapov (2017) explicam que a maneira de se programar o ESP32 para uma aplicação de IOT (*Internet of Things* ou internet das coisas) é através do software da Espressif, este que é um software exclusivo para o sistema operacional Linux.

Na Figura 1.7 observa-se um protótipo de um osciloscópio feito com ESP32, que demonstra um pouco da capacidade que o Microcontrolador possui, trata-se da criação de um osciloscópio baseado em um smartphone, onde a tela do osciloscópio seria o próprio smartphone, e o processamento seria feito pelo ESP32, com ajuda de uma placa de circuito e alguns outros componentes.

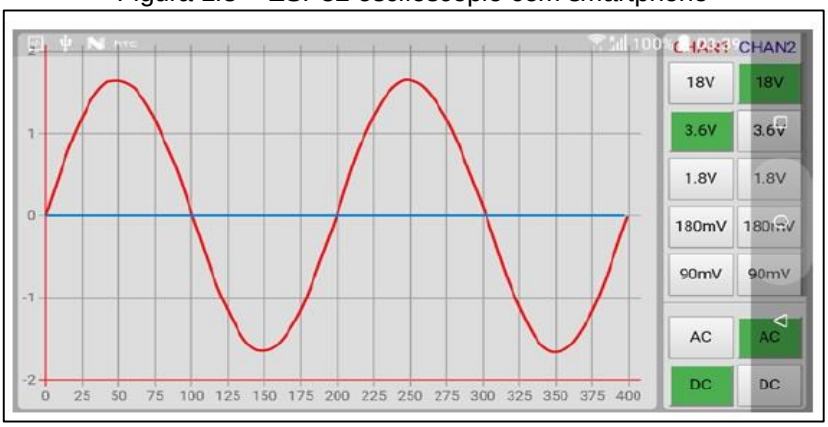
Figura 1.7 – ESP32 em protótipo de osciloscópio



Fonte: MAYER, SHARP E VAGAPOV, 2017 p.6

A Figura 1.8 foi desenvolvida através do Visual Studio usando o framework do Cordoba, representa a tela do smartphone com o osciloscópio, podendo ser utilizada em múltiplos sistemas operacionais de smartphones.

Figura 1.8 – ESP32 osciloscópio com smartphone



Fonte: MAYER, SHARP E VAGAPOV, 2017 p.6

Conclui-se que o ESP32 é um Microcontrolador versátil, tanto para aplicações domesticas, quanto para aplicações industriais e de IOT. Sua capacidade de controle e integração é muito grande, o que o torna capaz de controlar processos de todos os tipos.

1.5 Sistema de controle

Ogata (2010) destaca que um processo é composto por uma série de ações sistêmicas, sendo que estas ações devem ser controladas e tem um objetivo em comum a ser atingido, ou seja, controlar significa medir o valor de uma variável e

controlar o valor de outra variável, com a finalidade de fazer com que a variável medida atinja o valor necessário pelo processo naquela aplicação.

Os dispositivos de controle são os elementos responsáveis por tomar as decisões. Podem ser de vários tipos, dentre eles, elétricos, mecânicos, pneumáticos ou óticos. Na maioria dos casos são os Microcontroladores.

Novus (2003), explica que o controle PID consiste em calcular um valor de atuação sobre um processo, a partir do valor desejado e do valor atual da variável do processo. O Controle é composto por três ações: P é a correção proporcional ao erro; I é a correção proporcional ao produto erro x tempo; D é a correção proporcional à taxa de variação do erro.

Parâmetro P: ao aumentar, o processo, torna-se mais lento, geralmente torna se mais estável com menos oscilações, tem menos overshoot (valores que excedem seu valor alvo). Ao diminuir, o processo, fica mais rápido, torna se mais instável ou oscilante, tem mais overshoot.

Parâmetro I: ao aumentar, o processo, torna-se mais rápido, atingindo rapidamente o setpoint (valor alvo), fica mais instável ou oscilante, tem mais overshoot. Ao diminuir, o processo, torna-se mais lento demorando em atingir o setpoint, fica mais estável ou menos oscilante, tem menos overshoot.

Parâmetro D: ao aumentar, o processo, torna-se mais lento, tem menos overshoot. Ao diminuir, o processo, torna-se mais rápido, tem mais overshoot.

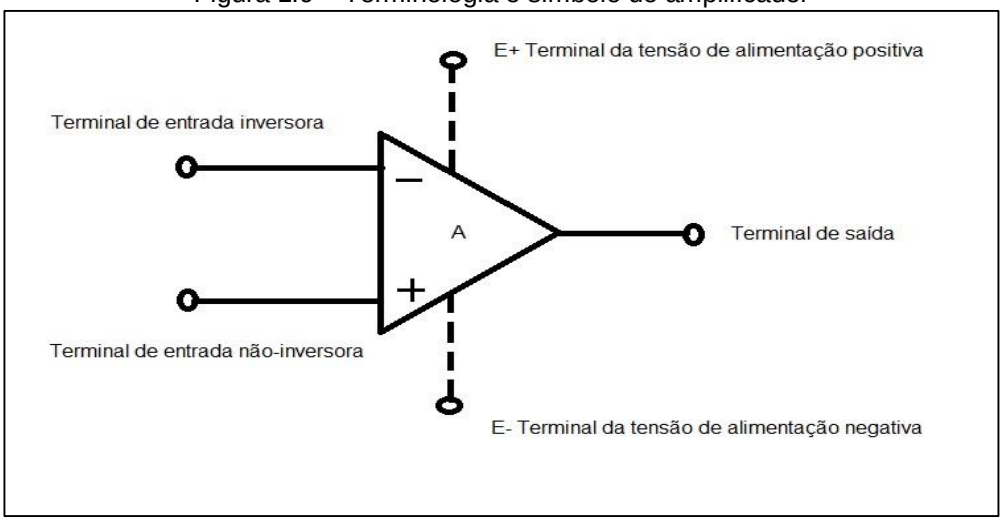
Ao unir se as três técnicas tem-se o controle PID, aproveitando-se das vantagens de cada tipo, o controle básico do P, a eliminação de erros do I, e a redução das oscilações do D.

1.6 Amplificadores operacionais

Segundo Gruiter (1988), o amplificador operacional é um circuito integrado (C.I.) comum em aplicações lineares, sendo muito versátil e de fácil aplicação. Desta forma, possui um campo muito amplo de suas aplicações devido suas características.

Um amplificador operacional é um amplificador de tensão contínua, linear, com um ganho elevado, utilizando uma rede de realimentação negativa para controlar suas características de operações. Internamente é composto por duas ou mais entradas de amplificadores diferenciais, seguidos por mais alguns amplificadores. A Figura 1.9 ilustra a terminologia e símbolo de um amplificador.

Figura 1.9 – Terminologia e símbolo do amplificador



Fonte: GRUITER, 1988, p.2

Um sinal no sentido positivo, aplicado na entrada não inversora produzirá, um sinal também positivo na saída, mas se o mesmo sinal de entrada for aplicado na entrada inversora, produzirá um sinal no sentido negativo, é importante dizer que os terminais de alimentação costumam ser simétricos. Ou seja: E+ é +15 V e E- é -15 V.

Ainda segundo o autor Gruiter (1988), os principais fatores que determinam as diferenças entre os amplificadores reais e ideais: o ganho bastante elevado quando em malha aberta, a resistência finita de entrada e a resistência de saída que deve tender a zero, deve-se ressaltar ainda alguns erros que podem ocorrer no uso de amplificadores, tais como, o erro no ganho de malha fechada devido ao valor finito da

resistência de entrada, o erro no ganho de malha fechada devido a valores não tão próximos de zero na resistência de saída, sinais de erros gerados na entrada, tensão *offset* na entrada, corrente *offset* de entrada, e o *slew rate* (tempo de resposta).

1.7 Internet das coisas e sua estrutura

Santos et al. (2017) esclarecem que a Internet das Coisas é uma extensão da Internet comum, possibilitando que dispositivos com capacidade computacional e de comunicação sejam conectados à Internet. A conexão com a Internet possibilita controlar remotamente os dispositivos e também que os mesmos sejam acessados como provedores de serviços.

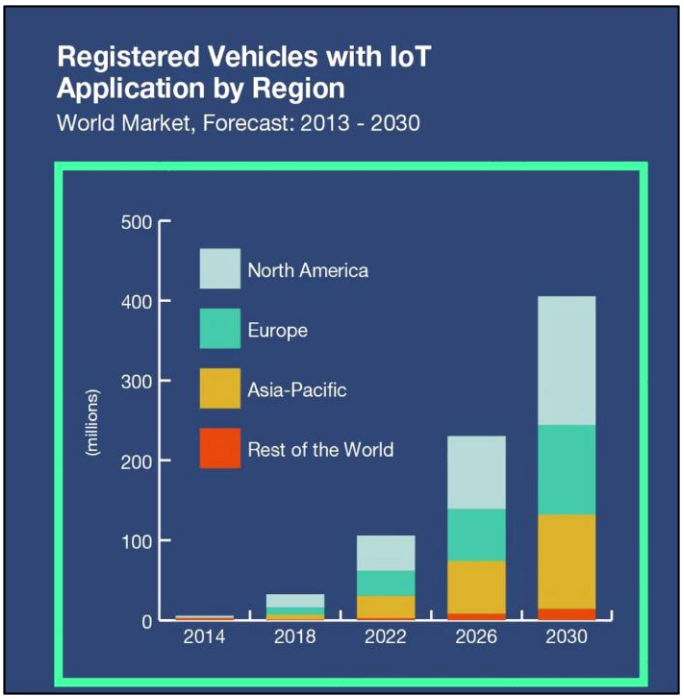
A Internet das Coisas teve seu surgimento possibilitado pelo avanço de áreas como a Microeletrônica, sensoriamento, sistemas embarcados e comunicação. Devido ao seu enorme potencial de aplicação em diversas atividades da vida humana, vem recebendo grande destaque nos meios acadêmicos e industriais.

Press (2014) esclarece que atualmente não só os computadores convencionais possuem acesso à grande rede, cada vez mais o acesso vem sendo viabilizado para outros aparelhos como: TV's, *smartphones, laptops*, automóveis, *videogames* e a lista aumenta a cada dia.

Utilizando os recursos destes dispositivos é possível controlar, realizar troca de informações, acessar serviços e interagir com outros usuários. Ao mesmo tempo surge uma gama de novas possibilidades aliadas a novos desafios como: regulamentações, segurança e padronizações.

Um dos grandes obstáculos à Internet das Coisas é a padronização de tecnologias, fator que permite a heterogeneidade de dispositivos conectados à Internet. Entretanto existe uma expectativa que até 2030 existam 400 milhões de dispositivos registrados com aplicações IOT, conforme mostra a Figura 1.10:

Figura 1.10 – Gráfico dos dispositivos com aplicação de IOT no mundo



Fonte: www.abiresearch.com, 2019

Santos et al. (2017) destacam que a IOT é considerada um conjunto de diversas tecnologias, das quais atuam para possibilitar a integração dos dispositivos do espaço físico ao virtual, sendo:

- Identificação: um dos blocos mais importantes, tendo em vista que é essencial identificar os dispositivos individualmente para conecta-los à rede. Podem-se utilizar tecnologias como RFID, NFC (Comunicação por campo de proximidade) e endereçamento IP para realizar a identificação;

- Comunicação: relacionada às diversas técnicas que podem ser utilizadas para conectar os dispositivos. Sendo um dos fatores determinantes para gestão do consumo de energia dos dispositivos. Algumas das tecnologias utilizadas para comunicação são WiFi, Bluetooth e RFID;

- Serviços: possuem diversas classes de serviços, como Identificação que mapeiam entidades físicas em virtuais, serviços de agregação de dados, de colaboração, inteligência e de ubiquidade;

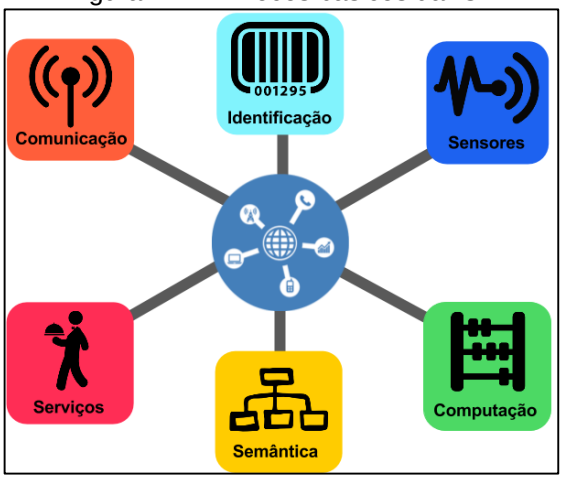
- Semântica: relaciona à habilidade de extração de conhecimento dos objetos, associando a descoberta de conhecimentos ao uso eficiente dos recursos provenientes da IOT;

- Computação: acrescenta uma unidade de processamento como um Microcontrolador ou Microprocessador;

- Sensores: os sensores colhem informações sobre o contexto em que os dispositivos se encontram, em seguida essas informações são armazenadas ou encaminhadas para centros de armazenamento ou uma nuvem.

A Figura 1.11 apresenta os blocos básicos de construção da IOT:

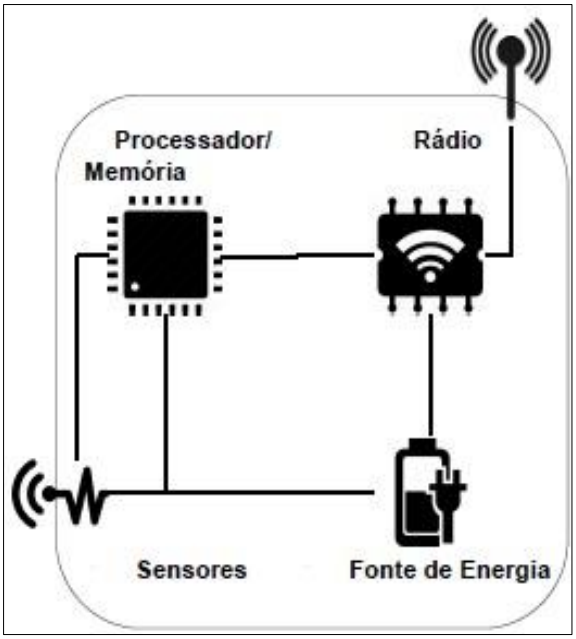
Figura 1.11 – Blocos básicos da IOT



Fonte: SANTOS et al, 2017, p.7

Segundo Santos et al. (2017) a arquitetura básica dos dispositivos IOT é composta por quatro blocos: processador/memória, comunicação, energia e sensores/atuadores. A figura 1.12 ilustra a visão geral da arquitetura de um dispositivo IOT:

Figura 1.12 – Arquitetura IOT



Fonte: SANTOS et al, 2017, p.8

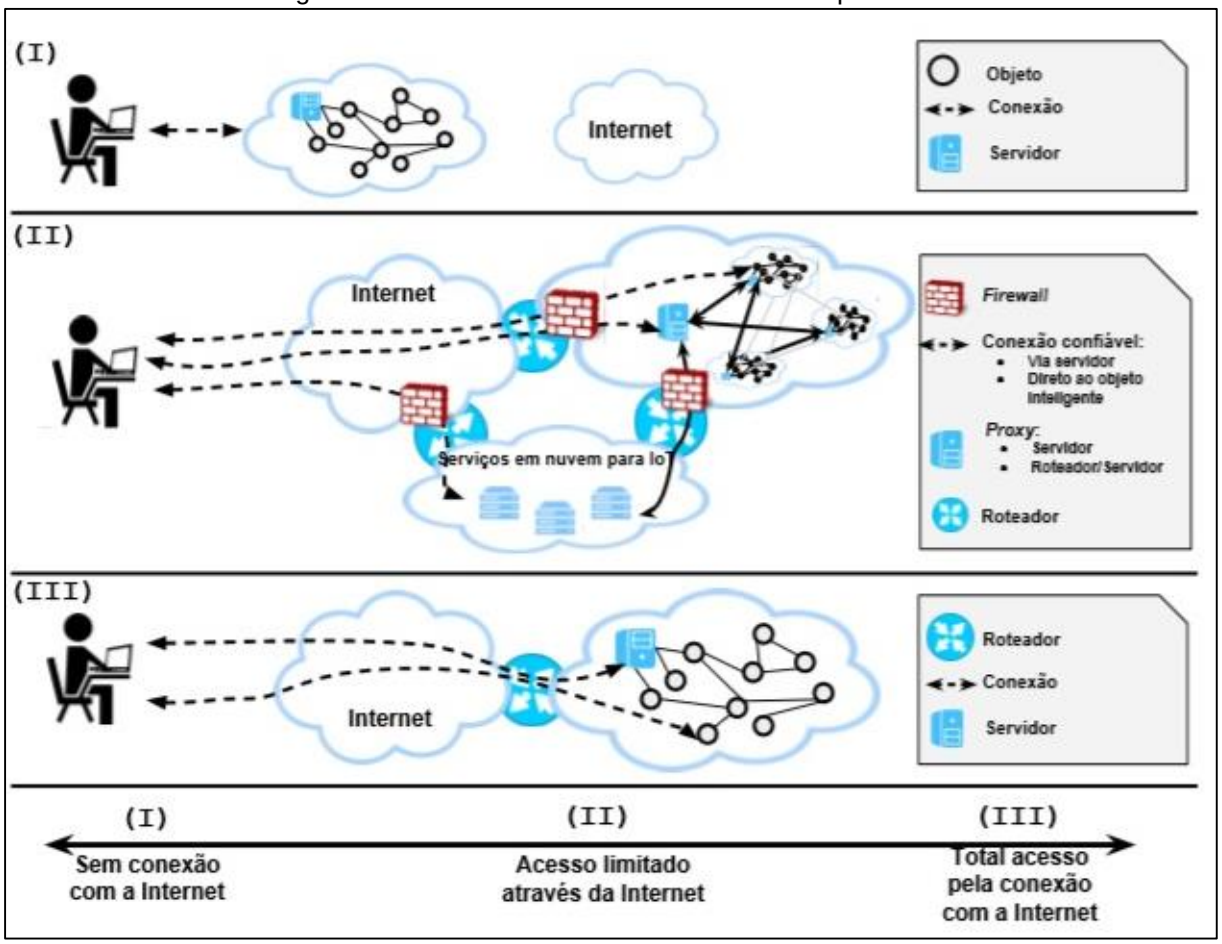
1.8 Computação em nuvem

Oliveira (2017) define que nuvem é o ambiente em que os serviços essenciais da rede são hospedados em servidores externos. De modo que esses serviços são terceirizados, para que os mesmos possuam alta disponibilidade a baixo custo.

Oliveira (2017) enfatiza que em aplicações críticas como medições de grandezas físicas, por exemplo, os dados não devem ficar hospedados nos dispositivos. As informações devem ser enviadas para um elemento centralizador, com maior poder de processamento e disponibilidade. Se esse elemento estiver na nuvem, as aplicações possuem maior flexibilidade.

Santos et al. (2017) destacam que os modelos de conectividade das redes IOT são classificados como um espectro formado por uma rede de dispositivos sem conexão com a Internet, uma rede de conexão entre os dispositivos inacessível para meios externos e uma rede de plena conexão a Internet; da qual os dispositivos são conectados efetivamente. Na Figura 1.13 podemos verificar os modelos de conectividade para dispositivos IOT em rede IPv6 sendo eles:

Figura 1.13 – Modelos de conectividade dos dispositivos



Fonte: SANTOS et al, 2017, p.16

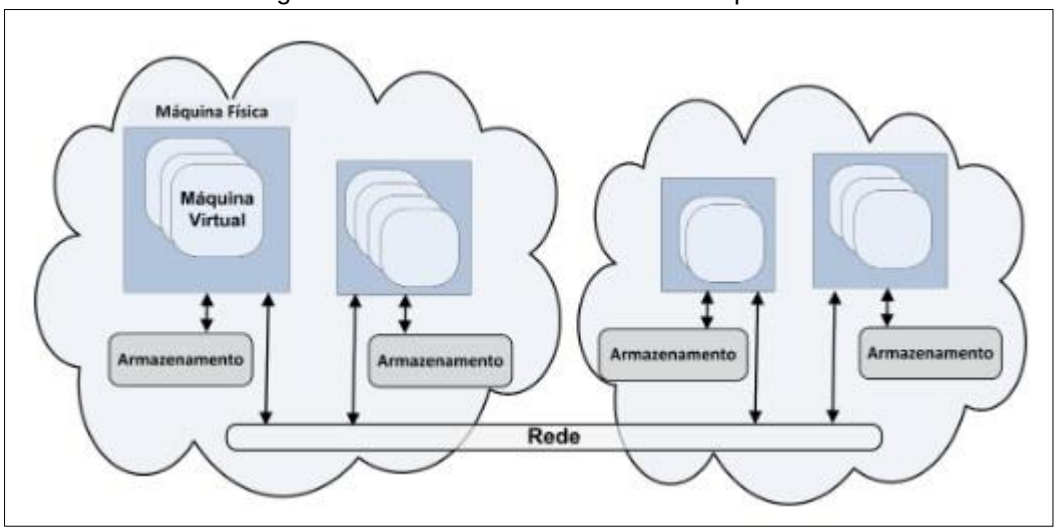
I. Rede autônoma, na qual os dispositivos inteligentes não possuem conexão com a Internet;

II. Rede de dispositivos inteligentes limitada, pois o acesso aos dispositivos é restrito;

III. IOT “autêntica”, na qual os objetos estão conectados à Internet.

Segundo Soror et al. (2010) a infraestrutura do ambiente em nuvem é composta por diversas máquinas ou nós físicos de baixo custo, ligados em rede. Sendo que cada máquina possui as mesmas configurações de software, mas há diferenças de *hardware* em termos de CPU, memória e armazenamento em disco. A figura 1.14 ilustra a conexão em rede das máquinas:

Figura 1.14 – Conexão em rede das máquinas



Fonte: SOUSA et al, 2009, p.4

1.9 Tecnologias de comunicação

Santos et al (2017) destacam os principais protocolos de comunicação utilizados em IOT:

- Ethernet: utiliza os cabos de par trançado ou fibra óptica, onde cada um proporciona diferentes taxas de comunicação. Tais os cabos atingem taxas de até um Gbps, limitados a 100 metros. Já os cabos de fibra óptica atingem taxas de até 10 Gbps, limitados a dois quilômetros;

- Bluetooth Low Energy: definido no padrão IEEE 802.15.1, é um dos protocolos de comunicação mais utilizados para IOT. Ao contrário das versões antigas do Bluetooth, esta possui especificação voltada para baixo consumo de energia, permitindo dispositivos que usam baterias do tamanho de moedas;

- WiFi: seguindo o padrão IEEE 802.15.1 esse padrão é uma solução de comunicação extremamente popular, ainda mais para IOT, devido à possibilidade de conexão em qualquer lugar que possua uma rede WiFi. Possui taxas de comunicação de 600 Mbps ou 1300 Mbps;

- 3G/4G: padrões de telefonia celular são muito utilizados em aplicações IOT em projetos que necessitam alcançar longas distâncias. Entretanto, o consumo de

energia das tecnologias 3G/4G é elevado em comparação a outras. A taxa de comunicação atingida pelo 3G é 1 Mbps e o 4G atinge 10 Mbps;

- SigFox: possui raio de cobertura entre 3 e 10 Km em áreas urbanas e entre 30 Km e 50 Km em áreas rurais, com taxa de comunicação que varia entre 10 e 1000 bps. Possui baixo consumo de energia e opera na faixa de 900 MHz;

- LoRaWAN: é uma rede focada em acessos a dispositivos de até 15 quilômetros, utilizando uma estação de rádio de longo alcance. Possui frequências entre 433 e 915 MHz. Possuem taxas de comunicação entre 300 bps e 50 Kbps e o consumo de energia é relativamente pequeno;

- ZigBee: desenvolvida a partir de protocolos definidos no padrão IEEE 802.15.4, com alcance máximo limitado a 100 metros. Opera nas frequências 868 MHz e 915 MHz, além 2.4 GHz (exclusiva da faixa ISM). Possui taxa máxima de 250 Kbps para comunicação e um baixo consumo.

Ainda em comunicações, destacam-se os principais protocolos da camada de aplicações:

- HTTP: utilizado nas redes com computadores do tipo PC, sendo que na Internet realiza o acesso de informações seguindo a estratégia requisição/resposta no paradigma cliente/servidor. O HTTP é amplamente utilizado em plataformas nuvem como o “thinger.io”. Entretanto seu uso é limitado em dispositivos IOT, devido à capacidade computacional baixas dos mesmos;

- CoAP: O *Constrained Application Protocol* (Protocolo de Aplicação Restrita) realiza transferência de dados semelhante ao REST (Representational State Transfer - Transferência de Estado Representacional) com funcionalidades semelhantes ao HTTP, permitindo que clientes e servidores acessem e consumam serviços via Internet facilmente;

- MQTT: O *Message Queue Telemetry Transport* (Transporte de Telemetria da Fila de Mensagens) é um protocolo voltado para dispositivos extremamente limitados e utiliza a estratégia publish / subscribe (publicar / assinar) para transferência de dados. Tendo como principal vantagem a minimização no uso de largura da banda de rede e recursos dos dispositivos.

2 METODOLOGIA

Neste capítulo encontra-se a trajetória para o desenvolvimento e construção do projeto que se intitula Manutenção Preditiva de Termopares por meio da IOT. Trata-se de uma pesquisa aplicada que é desenvolvida nas dependências da FATEC São Bernardo do Campo e nas residências dos integrantes do grupo.

Prodanov e Freitas (2013) destacam que a metodologia é o caminho a percorrer para o desenvolvimento de uma pesquisa. Enfocam que os métodos são procedimentos amplos do raciocínio e as técnicas são procedimentos que operacionalizam os métodos mediante instrumentos adequados.

Severino (2016) enfatiza que a preparação metódica e planejada de um trabalho científico supõe uma sequência de etapas que compreende: determinação do tema-problema e justificativa, levantamento da bibliografia referente ao tema, leitura e documentação dessa bibliografia após seleção, construção lógica do trabalho e redação do texto.

A construção da redação do TCC tem como base o Manual de Normalização de Projeto de Trabalho de Graduação da FATEC - SBC (2017) que se encontra embasado nas normas da ABNT.

2.1 O tema-problema com justificativa e descrição do projeto

O termopar é um componente elementar no processo produtivo de diversas empresas, sendo frequentemente adotado em um regime de substituição planejada, utilizando a manutenção preventiva ou não planejada, como a manutenção corretiva do componente. Assim não existe um regime de manutenção preditiva que deve trazer uma vantagem competitiva para o processo produtivo, por tanto viabilizar um projeto neste sentido e é uma abordagem inovadora no mercado.

A manutenção preditiva de termopares não é encontrada no meio produtivo conforme aponta a literatura especializada, podendo gerar desperdícios como a não

utilização completa do componente quando trocado precocemente pela manutenção preventiva ou a parada de produção causada pela quebra do sensor.

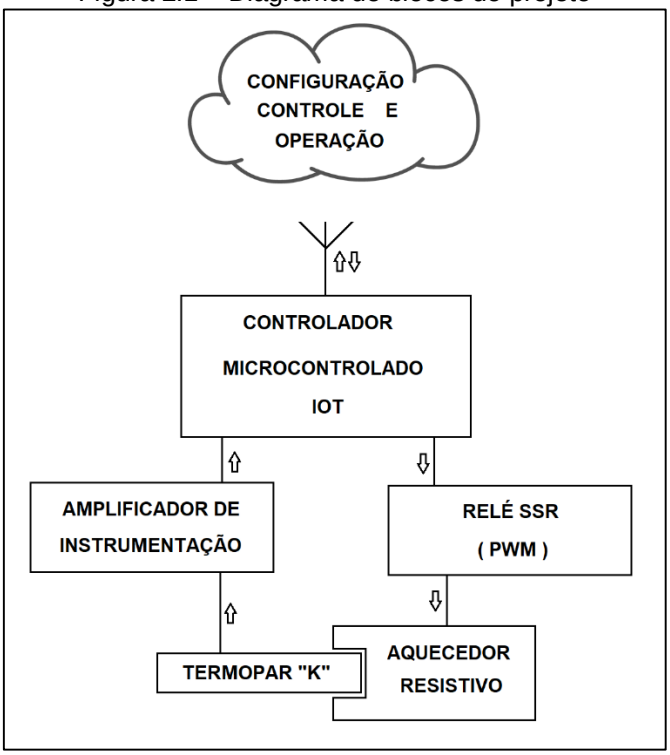
Desta forma, principal justificativa para este trabalho se baseia no ganho competitivo no processo de manutenção e substituição do componente de termopar.

Através de monitoramento do comportamento do Termopar é possível detectar quando o termopar inicia o comportamento de falha, e indicando sua troca neste momento.

Implementando a comunicação IOT junto ao projeto, faz-se com que os dados possam ser acessados remotamente e o controle do dispositivo possa ser feito pela nuvem ou de qualquer ponto de acesso.

Para a melhor visualização do projeto diagrama de blocos mostra a sua estrutura, conforme ilustra a Figura 2.1.

Figura 2.1 – Diagrama de blocos do projeto



Fonte: Autoria própria, 2019

Conforme o diagrama de blocos, o controlador com IOT é o foco principal do diagrama, conectando a todos os blocos que funcionam da seguinte forma:

Configuração, controle e operação: é parte de IOT do projeto, ficando na “nuvem”. O supervisório, o armazenamento de dados e o controle podem ser acessados remotamente, por meio da internet. Os dados são atualizados em tempo real;

Controlador microcontrolado com IOT: o tipo de comunicação com a nuvem é feito via WiFi. O controlador como ligação principal é responsável pelo envio e recepção dos dados da nuvem para controle e supervisão, receber os dados do amplificador de instrumentação para o monitoramento do termopar, e controlar o relé que realiza controle de potência do aquecedor resistivo;

Amplificador de Instrumentação: é um amplificador do sinal do termopar com compensação de junta fria para que o controlador possa ler a temperatura do termopar.

Termopar: o sensor que vai monitorar o processo industrial desejado e, ao mesmo tempo, monitorado pela controladora;

Aquecedor resistivo: um tipo de aquecedor que é controlado, estando em contato direto com o termopar e servindo no projeto para simular um processo industrial.

2.2 Etapas teóricas e práticas para o desenvolvimento do projeto

Após o esclarecimento do tema-problema e uma breve descrição do projeto através do diagrama parte-se para as seguintes etapas:

Primeira etapa: reunião dos componentes do grupo, para escolha do orientador. Após o convite, o orientador se colocou à disposição do grupo e concordou nos orientar com o tema;

Segunda etapa: reunião do grupo com o orientador, para tratar de temas como métodos de pesquisa e possíveis referencias, além de uma explicação geral sobre o tema principal do projeto. O orientador colocou-se à disposição para suprimir algumas dúvidas e marcou, obrigatoriamente, um dia por semana para lhe apresentar o andamento da pesquisa;

Terceira etapa: levantamento de objetivos e cronograma de atividades a serem desenvolvidas, conforme ilustrado a Figura 2.2;

Figura 2.2 – Cronograma de atividades

Semanas	Seleção Das fontes de pesquisa	Redação dos textos da fundamentação Teorica	Elaboração dos elemento pré e pós textuais	Concepção da introdução	Criação dos Slides	Apresentação do TCC a banca
1º Fevereiro						
2º Fevereiro						
3º Fevereiro						
4º Fevereiro						
1º Março						
2º Março						
3º Março						
4º Março						
5º Março						
1º Abril						
2º Abril						
3º Abril						
4º Abril						
5º Abril						
1º Maio						
2º Maio						
3º Maio						
4º Maio						
5º Maio						

Fonte: Autoria própria, 2019

Quarta etapa: levantamento bibliográfico ocorreu na biblioteca da FATEC S.B Campo, em sites especializados e manuais de empresas. Os mesmos foram lidos e relidos e selecionado aquelas teorias que melhor se enquadram para o desenvolvimento do projeto. Em seguida, constrói-se o Capitulo 1 – Fundamentação teórica e referencias.

Quinta etapa: levantamento dos materiais e componentes que são utilizados na construção do projeto. Pesquisa em sites e lojas especializadas, buscando a melhor viabilidade de preço. Aquisição dos materiais conforme Tabela 2.1

Tabela 2. 1 - Materiais utilizados para a confecção do projeto

Produto	Qtde.	Valor em Reais
Relé 8 pinos 5 V	1	11,73
Relé de estado sólido 5 V	1	27,41
Amplificador de precisão	1	27,32
Placa padrão 10x10 cm	1	6,50
Placa ESP32	1	36,03
Conector PCB 40 pinos	2	11,95
Módulo de Alimentação	1	21,25
Potenciômetro 470 Ω	1	18,00
Potenciômetro 1 KΩ	1	2,00
Chapa de madeira MDF 30X20 cm	1	12,50
Knob para potenciômetro	2	8,00
Ferro de solda 30W	1	21,90
Termopar tipo K	1	15,00
Trilho de alumínio 20x4,5 cm	1	8,35
LED’s RGB	5	1,50
TOTAL		229,44

Fonte: Autoria própria, 2019

Sexta etapa: configuração da plataforma do Arduíno para utiliza o ESP32.

Sétima etapa: elaboração do diagrama elétrico. Montagem da prototipagem em placa de teste. Verificação do funcionamento do protótipo e montagem final.

Oitava etapa: configuração da comunicação entre Microcontrolador (ESP32) e nuvem. Download da biblioteca para utilizar a programação do Microcontrolador.

Nona etapa: desenvolvimento do algoritmo para detecção de falha do termopar Monitoramento temporal da resistência elétrica do termopar comprando com o valor adotado como o padrão.

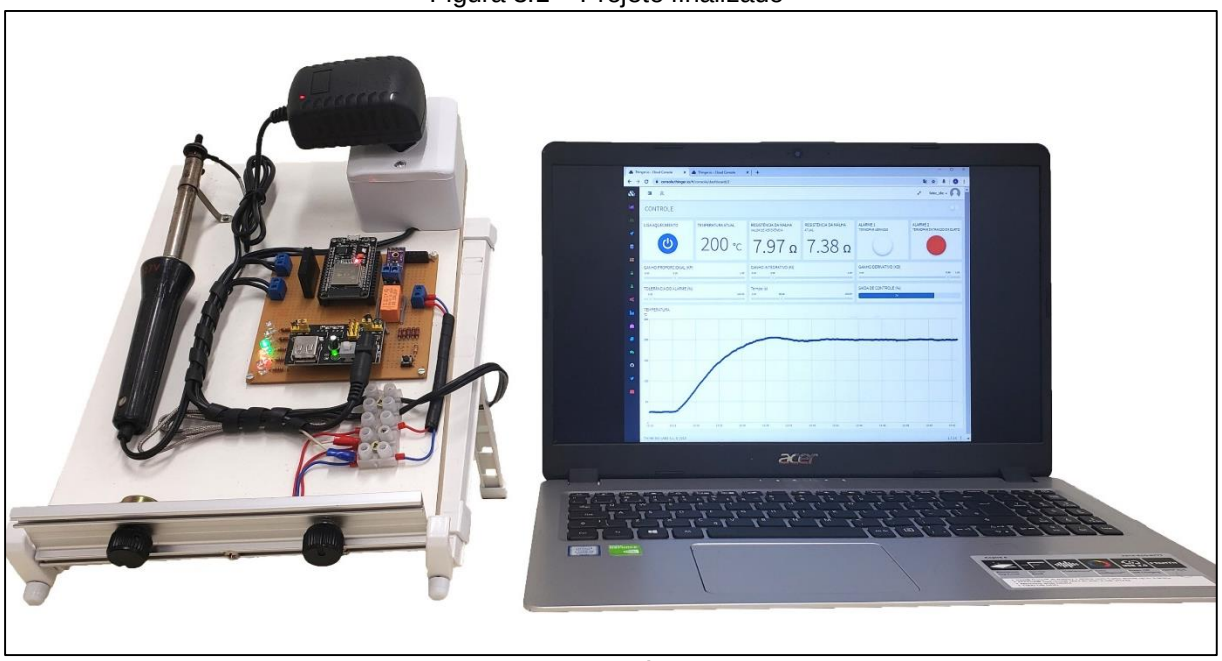
Décima etapa: criação de talas de Operador e Controle. Relações de variáveis criadas na nuvem e locais. Gráfico de monitoramento de temperatura. Registro de alarmes. Resistencia da malha do termopar. Criação de alarmes via e-mail.

Décima primeira etapa: Obstáculos de soluções. Concluído o desenvolvimento do projeto fazem-se as Considerações finais, Resumo e Abstract.

3 DESENVOLVIMENTO DO PROJETO

Neste capitulo encontra- se passo a passo o desenvolvimento e construção lógica do projeto intitulado Manutenção Preditiva de Termopares por meio da IOT. Para melhor entendimento a Figura 3.1 ilustra-o finalizado.

Figura 3.1 – Projeto finalizado



Fonte: Autoria própria, 2019

Seu funcionamento dá-se da seguinte maneira: o ESP32 abriga o algoritmo de detecção de falha do termopar, o Microcontrolador se comunica com a interface na nuvem enviando e recebendo dados, em nuvem existem alguns alarmes, bem como telas de controle.

No protótipo de testes há um amplificador operacional que será responsável por amplificar o sinal do termopar tipo K, além de um relé de estado sólido e um aquecedor resistivo, ambos controlados pelo ESP32. O aquecedor faz a simulação de um processo real de um aquecimento do termopar, possibilitando a ativação dos alarmes, tanto no protótipo quanto na interface em nuvem, quando ocorrerem as falhas.

O desenvolvimento do projeto encontra-se fundamentado nos seguintes tópicos:

- Configuração da interface de programação do Microcontrolador;
- Construção do protótipo de testes do termopar;
- Configuração de comunicação entre o Microcontrolador e a nuvem;
- Desenvolvimento do algoritmo para detecção de falha do termopar e controle PID;
- Desenvolvimento da interface na nuvem.

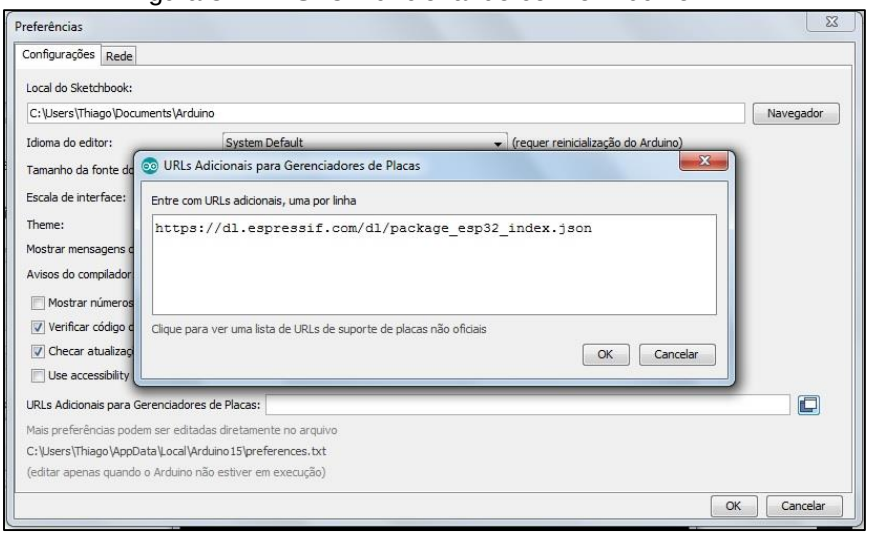
3.1 Configuração da interface de programação do Microcontrolador

A primeira etapa do desenvolvimento refere-se à configuração da interface de programação do Microcontrolador ESP32, com intuito de utilizar o Arduino IDE. Faz-se uso de um conversor USB-serial para fazer a comunicação entre o Microcontrolador e o computador. Em seguida baixa-se o framework mais recente do Arduino, compatível com o computador que se utiliza. Realiza a instalação do framework do Arduino, abrindo o programa para iniciar a configuração.

Com a janela do Arduino IDE aberta, clica-se na aba “arquivo”, e logo após em “preferências”, um novo menu é aberto. Na parte inferior há uma caixa de texto descrita como “URLs adicionadas para gerenciamento de placas”. Em seguida adiciona o endereço “https://dl.espressif.com/dl/package_esp32_index.json”. Este endereço é disponibilizado pela Espressif para que o Microcontrolador seja utilizado na plataforma do Arduino. Clicando em ok fecha-se a tela e retorna-se ao início.

Na tela principal clica-se em ferramentas para instalar a biblioteca ESP32, em seguida clica-se em “Arduino \ genuino uno” e em “gerenciador de placas”, um novo menu é aberto. Nele há uma barra de busca, selecionando-a, digita-se “ESP32”. Clicando em instalar a biblioteca ESP32 está disponível para utilização. A Figura 3.2 ilustra o ESP32 funcionando com o Arduino IDE.

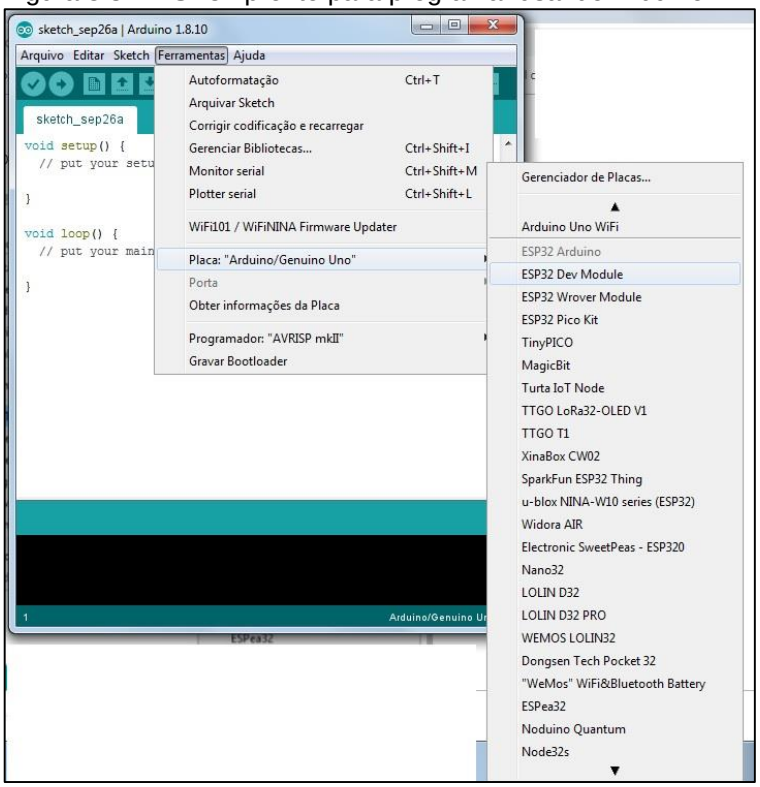
Figura 3.2 – ESP32 funcionando com o Arduino IDE



Fonte: Autoria própria 2019

Em seguida seleciona-se a aba “ferramentas” e “Arduino \ genuino uno”. Uma lista é apresentada. Seleciona-se “ESP32 dev module” e em seguida faz-se a programação do Microcontrolador, conforme ilustra a Figura 3.3:

Figura 3.3 – ESP32 pronto para programar usando Arduino IDE



Fonte: Autoria própria, 2019

3.2 Construção do protótipo de testes do termopar

Depois das configurações da IDE do Arduino, dá-se início a construção do protótipo de testes. O primeiro passo é criar o diagrama elétrico com a finalidade da construção física de um protótipo padrão.

Para a construção do diagrama elétrico há necessidade de cinco status do Microcontrolador sinalizados por *led's*. Utilizam-se os pinos 18 e 19 para os alarmes do termopar em processo de curto e abrindo, o *led* do pino 21 indica quando o teste de malha está sendo realizado, o 22 mostra que a saída de controle está ligada e 23 representa que o Microcontrolador está conectado com a internet.

Para fazer a leitura dos valores de tensão proveniente do termopar e adequá-los à faixa de leitura do Microcontrolador, usa-se um amplificador de instrumentação, cuja relação de saída é 5 mv / °C, conectado no GPIO 35, fixando os limites de leitura de 0°C a 660°C.

Constrói-se um divisor de tensão utilizando um resistor de 20 Ohm (R2) e o termopar, que representa o R1, a tensão de saída do divisor é lida através do pino GPIO 34. O divisor é alimentado com 3,3 V, o cálculo da resistência do termopar é dado pela equação a seguir [1].

$$R1 = (R2 / VR2 \times VE) - R2 \qquad [1]$$

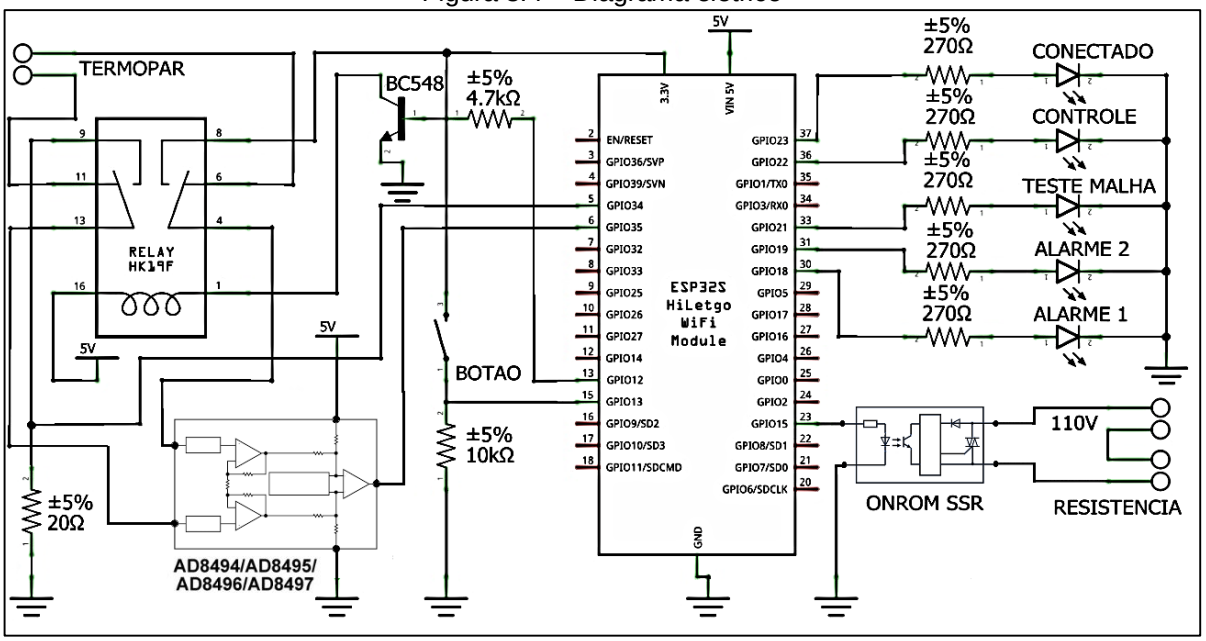
Onde,

R1 é a resistência da malha em teste, e R2 é a resistência de 20 Ohms, VR2 é a tensão sobre o resistor R2 medida pelo Microcontrolador, VE é a tensão de alimentação do divisor.

Alterna-se entre a leitura da temperatura e a medição da resistência da malha do termopar através de um relé com dois contatos reversíveis, em estado desligado mede-se a temperatura e ligado faz-se o teste de malha.

O relé de estado sólido ligado a GPIO 15 do Microcontrolador faz o controle do aquecimento de acordo com cálculo do algoritmo PID. De acordo com os componentes necessários para o funcionamento do algoritmo de testes, chega-se no seguinte diagrama elétrico, conforme a Figura 3.4:

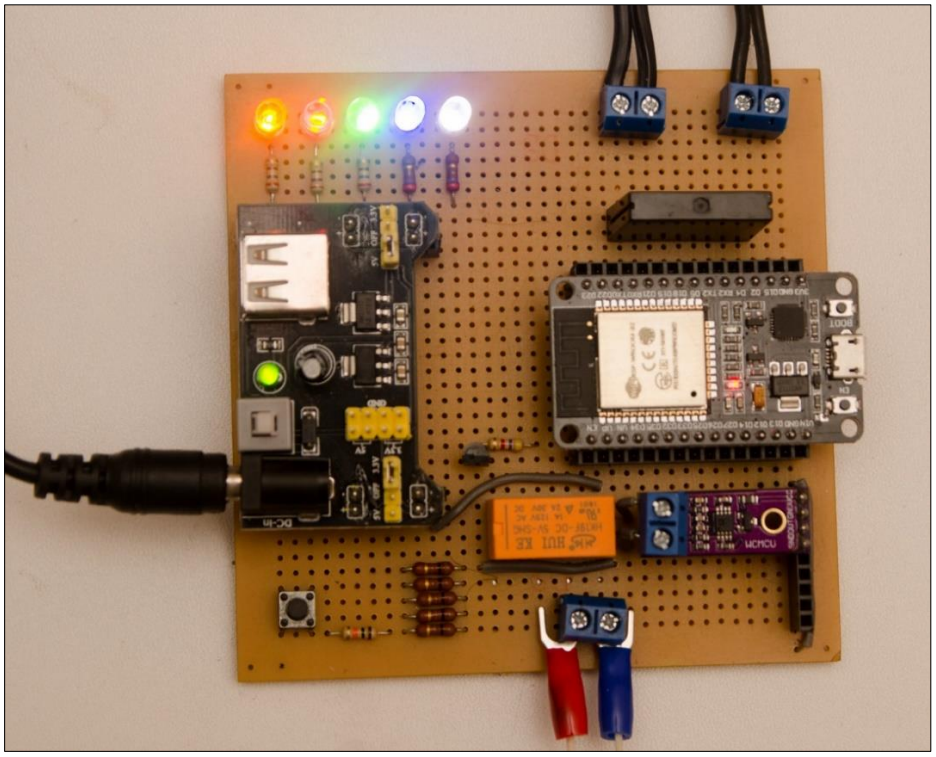
Figura 3.4 – Diagrama elétrico



Fonte: Autoria própria, 2019

Utiliza-se o diagrama elétrico da Figura 3.4 e uma placa padrão para realizar a montagem eletrônica do protótipo, o resultado pode ser conferido na Figura 3.5:

Figura 3.5 – Placa eletrônica do protótipo



Fonte: Autoria própria, 2019.

3.3 Configuração de comunicação entre o Microcontrolador e a nuvem

Depois da configuração da interface de programação do Microcontrolador e da construção do protótipo, parte-se para a configuração de comunicação entre o Microcontrolador e a nuvem. Na primeira fase acessa-se através de um computador o site “https://console.thinger.io/#/signup” para fazer o cadastro de um novo usuário da plataforma e fazer o download da biblioteca “thinger.io”. Mantém-se o site aberto para ser usado posteriormente.

Após o término do download, abre-se a janela da IDE do Arduino, clica-se na aba “sketch” e seleciona-se a opção “incluir arquivo” buscando a pasta “thinger.io” e escolhe o arquivo “ESP32”, dentro de uma pasta do mesmo nome. Com esta biblioteca, alguns comandos são inseridos na tela principal.

Retorna-se a tela principal do site “thinger.io”, clica-se na opção “Device” e no botão “Add Device” para criar um novo dispositivo para ser conectado à nuvem, no caso, o Microcontrolador ESP32. É adicionada uma descrição no campo “Device description” e digita-se uma credencial de conexão no campo “credentials”.

Em seguida retorna-se a IDE do Arduino, sendo necessário inserir o nome usuário da plataforma, nome do dispositivo e a credencial; nos comandos “#define USERNAME”, “#define DEVICE_ID” e “#define DEVICE_CREDENTIAL”, respectivamente. Os dados devem estar de acordo com o que foi criado em nuvem para evitar erros no programa e problemas de comunicação.

O último passo da configuração é definir a rede Wi-Fi para que o Microcontrolador se conecte a nuvem. Para isto, digita-se o nome da rede no comando “#define SSID” e a sua respectiva senha de acesso no comando “#define SSID_PASSWORD”. Ambos devem estar inseridos corretamente para haver comunicação com a rede de Wi-Fi. A Figura 3.6 exemplifica o uso da respectiva biblioteca:

Figura 3.6 – Exemplo de uso da biblioteca do thinger.io

```
#include "EEPROM.h"
#include "freertos/FreeRTOS.h"
#include "freertos/queue.h"
#include "freertos/task.h"
#include <ThingerESP32.h>

// informações de login na platafotma IOT www.thinger.io
#define USERNAME "fatec_sbc"
#define DEVICE_ID "1"
#define DEVICE_CREDENTIAL "tcc_fatec"

// informações das redes WiFi
#define SSID3 "REDE_1"
#define SSID_PASSWORD3 "senha_01"
#define SSID2 "rede_2"
#define SSID_PASSWORD2 "senha_02"
#define SSID4 "rede_3"
#define SSID_PASSWORD4 "senha_03"
#define SSID "rede_04"
#define SSID_PASSWORD "senha_04"
ThingerESP32 thing(USERNAME, DEVICE_ID, DEVICE_CREDENTIAL);
```

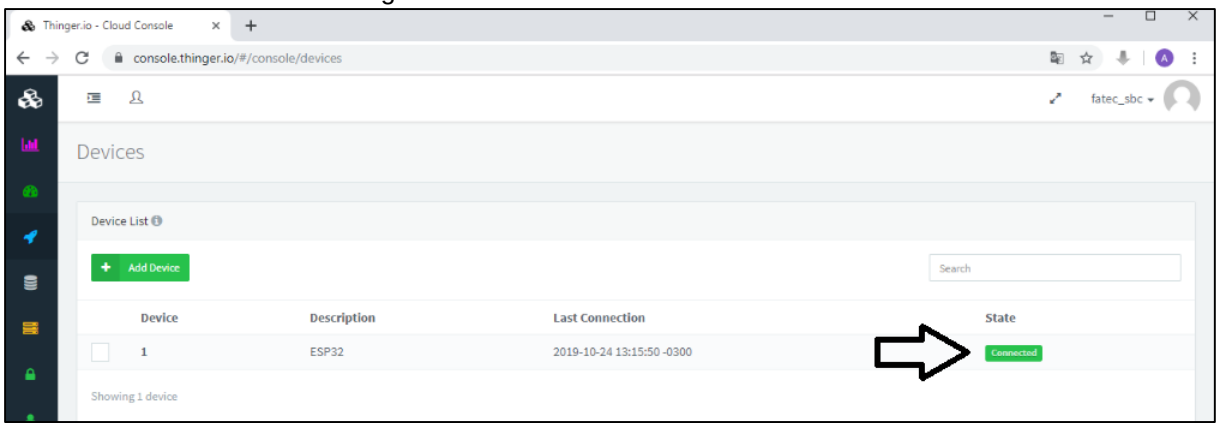
Fonte: Autoria própria, 2019

O processo de comunicação dá-se da seguinte maneira: conecta-se o Microcontrolador ao computador numa porta USB, via cabo. Na IDE clica-se na aba “Ferramentas” e em “Placa” aparece uma lista de dispositivos, seleciona-se “ESP32 Dev Module”, que foi instalado junto com a biblioteca “package_esp32”, e também seleciona a porta de conexão em “Ferramentas” para não haver erros de conexão.

Em seguida, se salva o programa criado na IDE e se clica no ícone “Carregar” para enviar o programa ao Microcontrolador. O processo leva alguns segundos e se

não houver erros, retorna à janela da nuvem. Em “Devices” aparece o dispositivo criado anteriormente e “Connected” no seu status. A Figura 3.7 ilustra a plataforma da nuvem conectada ao Microcontrolador:

Figura 3.7 – Plataforma nuvem conectada



Fonte: Autoria própria, 2019

A programação completa da configuração de comunicação entre o Microcontrolador e a nuvem encontra-se no Apêndice A.

3.4 Desenvolvimento do algoritmo - Detecção de falha e controle PID

O algoritmo de detecção de falha do termopar baseia-se conforme seu uso, ao aquecer e resfriar o termopar começa a apresentar fissuras nos arames, alterando assim sua resistência elétrica, para detectar esta variação de resistividade circula-se corrente elétrica pela malha. O valor da corrente é inversamente proporcional ao da resistência da malha, que é comparado com um valor padrão acrescido de uma tolerância.

Se a resistência medida for maior que o valor de referência, seta o alarme 1 que representa o termopar entrando em curto, quando menor seta-se o alarme 2 que representa o do termopar abrindo.

O teste é cíclico e o intervalo de tempo entre eles é ajustado em uma interface de controle na nuvem. Ao trocar um termopar o programa também guarda o primeiro valor adquirido e considera-o como referência, conforme ilustra a Figura 3.8:

Figura 3.8 – Algoritmo dos alarmes

```
if (rm > (ra * (1.0 + (tol / 100.0)))) {
        al1 = 1;
        ral1 = 1;
        digitalWrite(18, HIGH);
}

if (rm < (ra / (1.0 + (tol / 100.0)))) {
        al2 = 1;
        ral2 = 1;
        digitalWrite(19, HIGH);
}
```

Fonte: Autoria própria, 2019

Na ocorrência de falhas, liga-se um led localizado na placa e este alarme é sinalizado na interface da nuvem. Um e-mail é enviado para um destinatário configurado em nuvem, indicando a ocorrência do alarme e seu tipo e causa. O status do alarme permanece ligado até que a manutentor intervenha, ao reestabelecer o sistema, o mesmo deve acionar o botão que reseta os alarmes e realiza um novo teste na malha, adotando o valora da primeira leitura como referência.

Estes valores de resistência são visualizados na interface da nuvem. O histórico da resistência medida é armazenado em nuvem para análises futuras.

O controle de aquecimento baseia-se no algoritmo PID. Para fins de controle de temperatura os ajustes e o gráfico de aquecimento por tempo podem ser acessados através da interface na nuvem, conforme a Figura 3.9:

Figura 3.9 – Algoritmo PID

```
erro = sp - terc;
p = kp * erro;
i += ki * erro * ct;
d = kd * erro / ct;
pid = p + i + d;
```

Fonte: Autoria própria, 2019

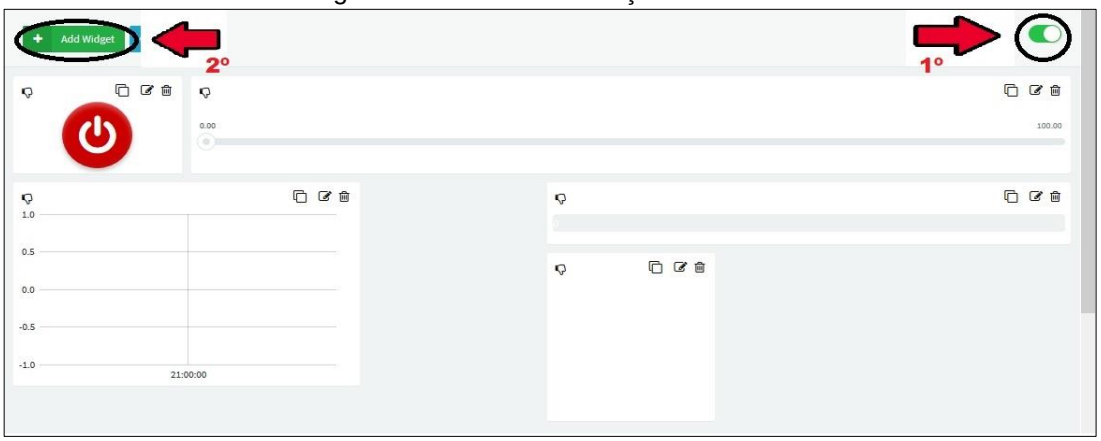
A programação completa da do algoritmo de detecção de falha e do controle PID encontram-se nos Apêndices C e D.

3.5 Desenvolvimento da interface na nuvem

Para finalizar o projeto, faz-se o desenvolvimento da interface na nuvem. Através de um computador, realiza-se o *login* no site “thinger.io”, clica-se na aba “Dashboards” e no botão “Add Dashboard”. Preenchem-se os campos “Dashboard id”, “Dashboard name” e “Dashboard description”, respectivamente. Este procedimento é realizado para criar as telas de “OPERADOR” e “CONTROLE” para interface para uso do operador e da configuração.

Com as telas criadas, são inseridas e configuradas as ferramentas para controle e monitoramento do processo. Para isto, habilita-se o modo de edição, clicando no botão superior ao lado direito e em “Add Widget” conforme a Figura 3.10.

Figura 3.10 – Modo de edição na nuvem



Fonte: Autoria própria, 2019

A janela “Widget Settings” é aberta e no campo “Type”, escolhe-se o objeto para ser inserido na tela. Para salvar as alterações realizadas em uma tela, deve-se desabilitar o modo de edição.

Na tela “OPERADOR” é inserido um gráfico para o usuário verificar a variação de temperatura no decorrer do tempo. Para isto, dentro da janela “Widget Settings” seleciona-se “Time Series Chart” no campo “Type”, “From Device” em “Chart Input”. Seleciona-se o “ESP32” em “Select Device” que é o dispositivo que foi criado anteriormente.

No campo “Enter Resource Name” é criada uma variável da nuvem que na programação é relacionada à respectiva variável de temperatura já existente, tal relação deve ser feita conforme a Figura 3.11 para as variáveis de todos os objetos criados na nuvem.

Figura 3.11 – Relacionamento de variáveis da nuvem na programação.

```
//Relacionamento entre variáveis da thinger.io e variáveis do ESP32.
thing["tliga"]  << inputValue(liga);
thing["tsp"]    << inputValue(sp);
thing["tkp"]    << inputValue(kp);
thing["tki"]    << inputValue(ki);
thing["tkd"]    << inputValue(kd);
thing["tct"]    << inputValue(ct);
thing["ttick"]  << inputValue(tick);
thing["ttol"]   << inputValue(tol);
thing["tterm"]  >> outputValue(term);
thing["tterc"]  >> outputValue(terc);
thing["tpid"]   >> outputValue(pid);
thing["tpout"]  >> outputValue(pout);
thing["tal1"]   >> outputValue(al1);
thing["tal2"]   >> outputValue(al2);
thing["tral1"]  >> outputValue(ral1);
thing["tral2"]  >> outputValue(ral2);
thing["trm"]    >> outputValue(rm);
thing["tra"]    >> outputValue(ra);
```

Fonte: Autoria própria, 2019

Por último é definido o tempo de atualização do gráfico no item “Time Period”.

Após o gráfico é inserido um botão para ligar o aquecimento e um indicador para mostrar o valor atual de temperatura em graus Celsius para o usuário. No botão de aquecimento criado, seleciona-se “On/Off State” na lista de *widgets* e cria-se uma variável para ser associada à programação. Para o indicador, seleciona-se o objeto “Text / Value” e cria-se a variável para exibir o valor no indicador. Na aba “Display

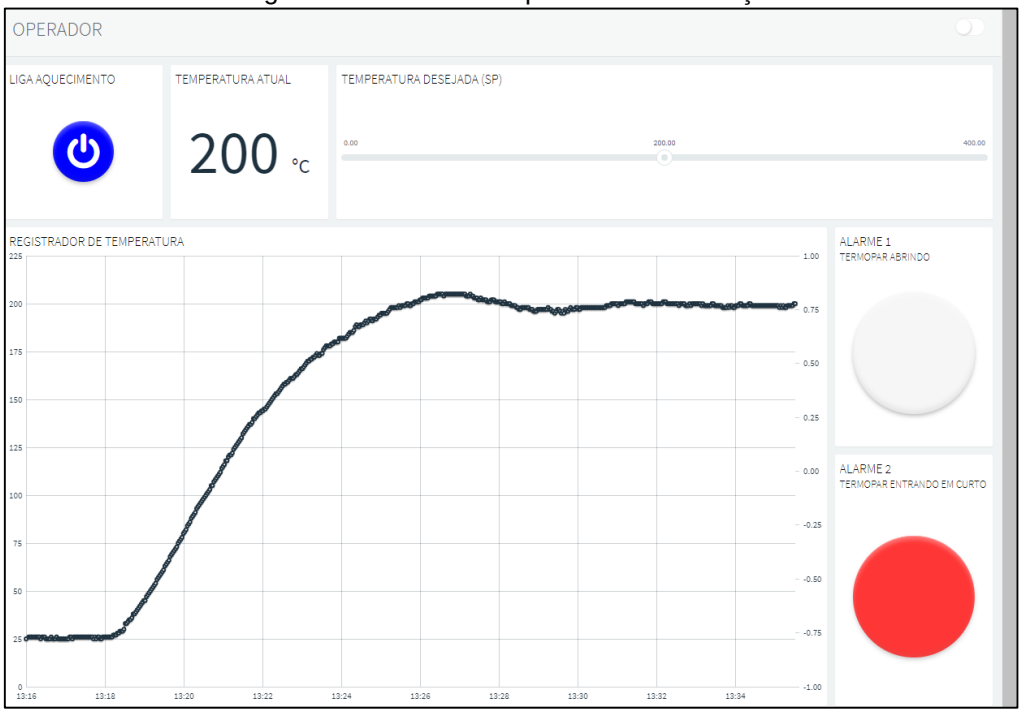
Options” digita-se “ºC” no campo “Units” para temperatura ser exibida em graus Celsius.

Em seguida é inserida uma barra para o usuário selecionar a temperatura desejada. Para cria-la, seleciona-se “Slider” na lista de *widgets*, define-se uma variável e na aba “Display Options” digitam-se os valores de mínimo e máximo de temperatura: 0 ºC e 400 ºC, nos campos “Min Value” e “Max Value”.

Por último são colocados os alarmes que indicam as falhas de termopar abrindo e entrando em curto. Para isto, seleciona-se a opção “Led Indicator” e cria-se uma variável para cada alarme. Na aba “Display Options” seleciona-se as cores para alarme ligado e desligado no campo “Color”. Na aba “Widget” se define título e subtítulo dos alarmes, nos campos “Tilte” e “Subtiltle”.

No modo edição de telas também se realizam os ajustes de nome, tamanho, cores e posição dos objetos. Após os ajustes conecta-se o Microcontrolador para executar a comunicação com a nuvem. A Figura 3.12 ilustra a tela “OPERADOR” em funcionamento.

Figura 3.12 – Tela de Operador em execução



Fonte: Autoria própria, 2019

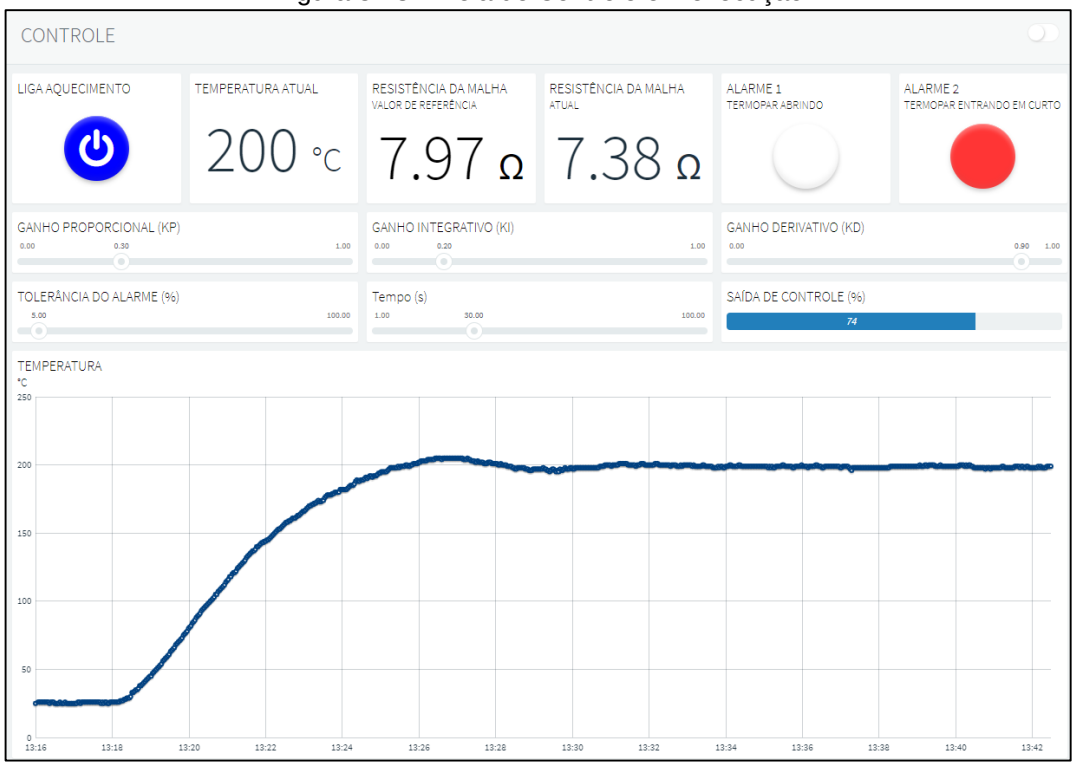
A tela “CONTROLE” é desenvolvida para ser utilizada por técnicos responsáveis pela realização manutenções, possuindo recursos exclusivos para estas ações, além dos que são disponíveis ao operador.

De forma análoga à tela “OPERADOR” são realizado novamente todos os passos descritos.

Em seguida colocam-se mais dois indicadores “Text / Value”. Ambos devem registrar o valor de resistência de malha medido em ohms (Ω), sendo que um deles apresenta um valor de referência e o outro o valor real medido na malha do termopar. Cria-se uma variável para o indicador do valor de referência e medido. Por último define-se a unidade de medida de resistência, clicando na aba “Display Options” e digitando “Ω” no campo “Units”.

São inseridas barras “Slider” para que o técnico realize ajustes nos parâmetros do controle PID, como: ganho proporcional (KP), integrativo (KI), derivativo (KD), o tempo de leitura do valor do termopar em segundos e a tolerância de operação dos alarmes em (%). Insere-se também uma barra do tipo “Progressbar” e uma variável para indicar a situação de saída do controle em (%). Organizam-se todas as barras e inserem-se seus respectivos nomes. Conecta-se o Microcontrolador novamente, para executar a comunicação e o controle via nuvem, conforme a Figura 3.13:

Figura 3.13 – Tela de Controle em execução



Fonte: Autoria própria, 2019

Após a criação das telas na aba “Dashboards”, clica-se na aba “Data Buckets” para a criação de registradores de operações das telas. Para isto, clica-se no botão “Add Data Bucket” e cria-se um código de registrador, nome e descrição, nos campos “Bucket id”, “Bucket name” e “Bucket description”, respectivamente. Em seguida seleciona-se a fonte de dados no campo “Data Source”, dispositivo utilizado e a variável da nuvem a ser utilizada pelo registrador.

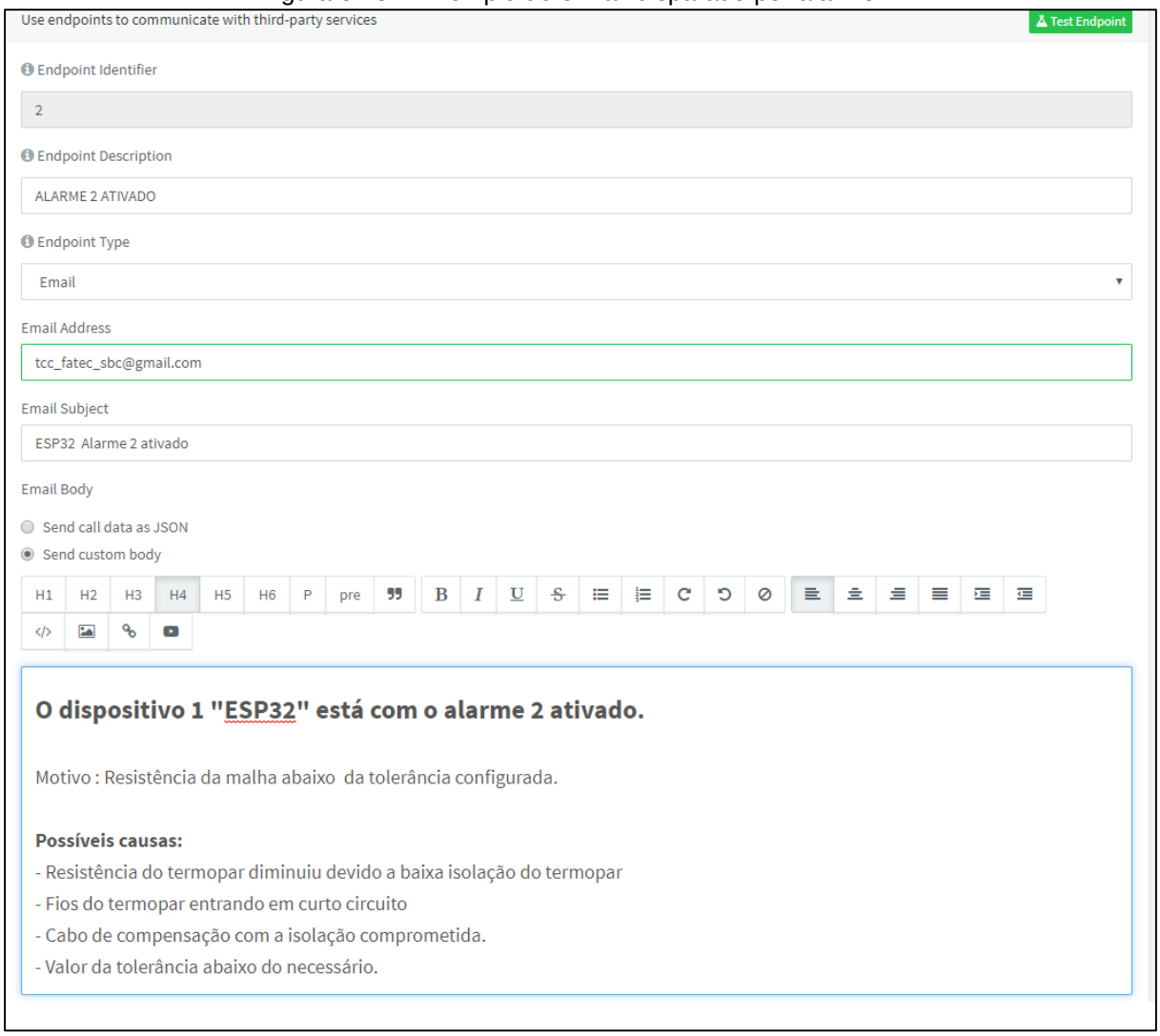
Cria-se um registrador para cada um dos alarmes e um para a medição de resistência na malha. Dentro de um determinado período de tempo os valores são registrados, gerando um histórico para o técnico consultar posteriormente. A Figura 3.14 ilustra a atuação de um dos registradores:

Figura 3.14 – Registrador de resistência da malha

Fonte: Autoria própria, 2019

Após a criação dos registradores, clica-se na aba “Endpoints” para a elaboração de *e-mails* que são enviados ao técnico no momento em que ocorre alguma falha no termopar ou quando é feito *reset* de um dos alarmes. Para isto, clica-se em “Add Endpoint”, digita-se o nome e a descrição do *endpoint* nos campos “Endpoint Identifier” e “Endpoint Description”. No campo “Endpoint Type” seleciona-se “Email” e digita-se o endereço de *e-mail* no qual a mensagem é enviada. Aparece uma caixa de mensagem na qual se digita a respectiva mensagem que é enviada ao técnico como demonstra a Figura 3.15:

Figura 3.15 – Exemplo de e-mail disparado por alarme 2



Fonte: Autoria própria, 2019

Deste modo conclui-se o desenvolvimento e construção do protótipo e desenvolvimento do algoritmo com sucesso.

3.6 Obstáculos e soluções

Durante o desenvolvimento do projeto surgiram alguns obstáculos e os mesmos foram solucionados através de pesquisas, conhecimento dos integrantes do grupo e consulta a professores da área.

Obstáculo 1: É possível através do ângulo da reta de aquecimento, variação de tempo por variação de temperatura, verificar se o termopar está abrindo ou está em curto, este método demonstrou não funcionar, devido aos valores de alteração do

ângulo serem mínimos, e facilmente confundidos com alterações normais em um processo envolvendo termopar.

Solução: foi desenvolvido um novo algoritmo de detecção de falha, utilizando uma corrente que passa pelo termopar periodicamente, esta que serve para verificar se o mesmo está abrindo ou em curto.

CONSIDERAÇÕES FINAIS

O projeto intitulado Manutenção Preditiva de Termopares por meio da IOT tem como objetivo desenvolver um controlador de temperatura e algoritmo de detecção de falhas com suporte a Internet das Coisas (IOT). Tal objetivo foi concluído com êxito. O projeto se justificou o pelo ganho da competitividade na utilização e frequência de trocas dos termopares, com redução de custo e paradas para manutenção.

O desenvolvimento se deu no Microcontrolador ESP32, assim como o amplificador de instrumentação, monitoramento da malha do termopar, plataforma “thinger.io” e IDE do Arduino.

Dentre as fontes pesquisadas necessárias para execução do projeto, as que mais agregaram conhecimento foram sobre a comunicação em nuvem e programação e configuração do ESP32, juntamente com os conhecimentos obtidos nos tópicos de manutenção preditiva, termopar, amplificadores e controle PID, que foram de suma importância para a concretização do objetivo proposto.

Os métodos e técnicas obtidas pela metodologia científica deram suporte para organizar e planejar as etapas que direcionam o caminho para o desenvolvimento do projeto. Deste modo foi possível trazer uma nova abordagem de manutenção, no caso a preditiva, para o componente termopar, podendo futuramente ser reaplicado o conceito em diversos outros componentes no processo produtivo.

O projeto trouxe para os integrantes do grupo conhecimentos agregador aos obtidos durante a formação durante o curso, complementando os conhecimentos técnicos e incentivando à pesquisa e desenvolvimento de tecnologia. Trouxe também a possibilidade de pesquisa e conhecimento de novos componentes, como o ESP32, o qual foi compartilhado com os docentes trazendo nova referência na elaboração de projetos e prototipagem mais efetiva.

Foram encontrados desafios durante a elaboração do projeto, porém foram solucionados com êxito com ajuda do embasamento bibliográfico e também com ajuda dos conhecimentos dos docentes.

O projeto traz para a comunidade cientifica a possibilidade de estudo de caso, assim como abre a possibilidade para implementação e desenvolvimento de novos algoritmos para detecção de falhas de forma preditiva em componentes industriais, sendo necessário um estudo de caso para cada novo projeto.

Como desvantagem destaca-se a não possibilidade atual de integração com Controladores Lógicos Programáveis (CLP's), que pode ser desenvolvida futuramente em projetos de pesquisa.

Tendo em vista todo o desenvolvimento, conclui-se que o projeto foi desenvolvido com êxito, trazendo novos conhecimentos para os integrantes do grupo, principalmente sobre o Microcontrolador ESP32, a elaboração do algoritmo de detecção de falha e comunicação com a nuvem.

Propõem como ações de melhorias futuras a comunicação do protótipo com Controladores Lógicos Programáveis, assim como desenvolvimento de novos algoritmos para detecção de falhas de outros tipos de sensores e a aplicação de *Machine Learning* e conceitos de *Big Data*.

REFERÊNCIAS BIBLIOGRÁFICAS

AGUIRRE, Luis Antonio. **Fundamentos de Instrumentação**. 1. ed. ISBN: 978-85-8143-183-3. Pearson: 2015.

ALCIATORE, DAVID G.; HISTAND, Michael B. **Introdução à Mecatrônica e aos sistemas de medições**. 4. ed. ISBN: 978-85-8055-341-3. São Paulo: Bookman, 2014.

ALMEIDA, M. T. **Manutenção preditiva: confiabilidade e qualidade**. 2000. Disponível em:< http://www.mtaev.com.br>. Acesso em: 06 abr. 2019.

BORGES, Charles **Os poderosos µControladores avr.** Santa Catarina: Departamento acadêmico de eletrônica, 2009.

CARLSON, jan. **Maintenance tips: thermocouples**. 2017. Catalytic Products needs. Disponível em: < https://www.cpilink.com/blog/maintenance-tips-thermocouples>. Acesso em: 18 mar. 2019.

ESPRESSIF. **ESP32 Datasheet.** Shanghai: Espressif Systems, 2018. 29 p.

FEMP - Federal Energy Management Program - U.S. Department of Energy. 2010. **Operations & Maintenance Best Practices**. Disponível em: < https://www.energy.gov/sites/prod/files/2013/10/f3/omguide_complete.pdf>. Acesso em 11 out. 2019.

GONSALEZ, Wagner P. A **administração da produção**. 2008. Disponível em: <http://www.administradores.com.br/artigos/carreira/a-administracao-da-producao/23401/>. Acesso em 07 abr. 2019.

GRUITER, Arthur François de. **Amplificadores operacionais fundamentos e aplicações.** ed.1, São Paulo: McGraw-Hill, 1988.

KARDEC, Alan.; NASCIF J. **Manutenção: função estratégica**. 3. ed. Rio de Janeiro: Qualitymark: Petrobrás, 2009. 384 p.

MANUAL DE NORMALIZAÇÃO DE PROJETO DE TRABALHO DE GRADUAÇÃO – FATEC SBCAMPO. **Material didático para utilização nos projetos de trabalho de graduação dos cursos de tecnologia em automação industrial e informática.** São Bernardo do Campo: Fatec, 2017.

MAYER, Alexander; SHARP, Andrew; VAGAPOV, Yuriy **Comparative Analysis and Practical Implementation of the ESP32 Microcontroller Module for the Internet of Things.** País de Gales: Glyndwr University, publicado em 2017. 07 p

NOVUS. **Controle PID básico.** Novus produtos eletrônicos Ltda, publicado em 2003, 06 P. Disponível em: <www.novus.com.br/artigosnoticias/arquivos/ArtigoPIDBasicoNovus.pdf>. Acesso em 11 out. 2019.

OGATA, Katsushiko. **Engenharia de controle moderno.** 5. ed., São Paulo: Pearson, 2010.

OLIVEIRA, Sergio. **Internet da Coisas com ESP8266, Arduino e Raspberry Pi.** ed.1, São Paulo: Novatec, 2017.

PEREIRA, Fábio. **Microcontrolador pic18 detalhado: Hardware e Software.** ed.7, São Paulo: Erica, 2010.

PRESS, G. **Internet of Things By The Numbers: Market Estimates And Forecasts**. New York: Forbes Publications, 2014

PRODANOV, C.C.; FREITAS, E.C. **Metodologia do trabalho científico:** métodos e técnicas da pesquisa e do trabalho científico. 2. ed. Rio Grande do Sul: Universidade Feevale, 2013.

SANTOS, B. et al. (2017). **Internet das Coisas:** da teoria à prática. Universidade Federal de Minas Gerais. Belo Horizonte: UFMG, 2017.

SEVERINO, A. J. **Metodologia do trabalho científico**, 23. ed. São Paulo: Cortez, 2017.

SOROR, A. et al. (2010). **Automatic virtual machine configuration for databases worloads**. Chicago: ACM Trans, 2010.

SOUSA, F. et al. (2009). **Computação em Nuvem:** Conceitos, tecnologias, Aplicações e Desafios. Escola Regional de Computação. Ceará: UFC, 2009.

SLACK, N.; CHAMBERS, S.; JOHNSTON, R. **Administração da Produção**. São Paulo: Atlas, 2002.

UDOP - União dos Produtores de Bioenergia. 2007. **O que é Manutenção Industrial**. Disponível em: <https://www.udop.com.br/index.php?item=noticias&co d=65080>. Acesso em 07 abr. 2019.

VIANA H. R. G., **PCM – Planejamento e controle de manutenção.** Rio de Janeiro: Qualitymark, 2002.

APÊNDICES

APÊNDICE A - CONFIGURAÇÃO DE CONEXÃO ENTRE O MICROCONTROLADOR E NUVEM

```
#include "EEPROM.h"
#include "freertos/FreeRTOS.h"
#include "freertos/queue.h"
#include "freertos/task.h"
#include <ThingerESP32.h>

// informações de login na platafotma IOT www.thinger.io
#define USERNAME "fatec_sbc"
#define DEVICE_ID "1"
#define DEVICE_CREDENTIAL "tcc_fatec"

// informações das redes WiFi
#define SSID3 "REDE_1"
#define SSID_PASSWORD3 "senha_01"
#define SSID2 "rede_2"
#define SSID_PASSWORD2 "senha_02"
#define SSID4 "rede_3"
#define SSID_PASSWORD4 "senha_03"
#define SSID "rede_04"
#define SSID_PASSWORD "senha_04"
ThingerESP32 thing(USERNAME, DEVICE_ID, DEVICE_CREDENTIAL);
```

APÊNDICE B - VARIÁVEIS E TAREFAS DE EXECUÇÃO

```
// Declaração das variáveis globais
float term = 0;       // Temperatura Real do Termopar.
float terc = 0;       // Temperatura Considerada no Processo.
float test = 0;       // Teste do Termopar
float erro = 0;       // Erro.
float kp = 0.8;       // Ganho Proporcional.
float ki = 0.3;       // Ganho Integrativo.
float kd = 0.0;       // Ganho Derivativo.
float ct = 1;         // Tempo de Ciclo.
float p = 0;          // Valor Proporcional Calculado.
float i = 0;          // Valor Integral Calculado.
float d = 0;          // Valor Derivativo Calculado.
float pid = 0;        // Saída PID.
float pout = 0;       // Percentual da Saida de Controle.
float vr2 = 0;        // Tensão no resistor 2 do divisor de tensão.
float rm = 0;         // Resistência da Malha
float ra = 0;         // Resistência Adotata
int sp = 0;           // Set Point.
int tol = 5;          // Tolerância dos alarmes
int freq = 1;         // Frequencia do PWM
int ledChannel = 0;   // Canal do PWM
int resolution = 10;  // Resolução do PWM
int al1 = 0;          // Alarme 1 - Termopar Abrindo.
int al2 = 0;          // Alarme 2 - Termopar Em Curto.
int ral1 = 0;         // Alarme 1 - Registro Termopar Abrindo.
int ral2 = 0;         // Alarme 2 - Registro Termopar Emtrando em Curto.
int tick = 5;         // Tempo do Teste
bool liga = 0;        // Liga aquecimento.

SemaphoreHandle_t myMutex;
QueueHandle_t xQueueTeste;

void vTesteSensor(void *pvParameters);
void vControleTemperatura(void *pvParameters);
void vTrocaSensor(void *pvParameters);
void vMonitoraFlash(void *pvParameters);
void vResetAlarme(void *pvParameters);
void lerFlash(void);
void gravarFlash(void);
```

APÊNDICE C - ALGORITMO DE DETECÇÃO DE FALHAS

```
// Task Teste da malha do termopar
void vTesteSensor(void *pvParameters) {
        int acum = 0;
        int teste = 0;
        BaseType_t xStatus;
        uint32_t xData = 0;
        TickType_t xTicksToDelay;

        while (1) {
                xSemaphoreTake(myMutex, portMAX_DELAY);
                digitalWrite(12, HIGH);
                digitalWrite(23, HIGH);
                vTaskDelay(300 / portTICK_PERIOD_MS);

                rm = analogRead(34);
                digitalWrite(12, LOW);
                digitalWrite(23, LOW);
                vTaskDelay(300 / portTICK_PERIOD_MS);
                xSemaphoreGive(myMutex);

                if (rm > (ra * (1.0 + (tol / 100.0)))) {
                        al1 = 1;
                        ral1 = 1;
                        digitalWrite(18, HIGH);
                }

                if (rm < (ra / (1.0 + (tol / 100.0)))) {
                        al2 = 1;
                        ral2 = 1;
                        digitalWrite(19, HIGH);
                }
        }
                if (xStatus == pdPASS) {
                        if (xData == 1) {
                                ra = rm;
                                al1 = 0;
                                digitalWrite(18, LOW);
                                al2 = 0;
                                digitalWrite(19, LOW);
                          xTaskCreatePinnedToCore(vResetAlarme, "vResetAlarme", 1024, NULL, 1, NULL, 0);
                        }
                }

                xTicksToDelay = ((tick * 1000) - 700) / portTICK_PERIOD_MS;
                xStatus = xQueueReceive(xQueueTeste, &xData, xTicksToDelay);
        }
}
```

APÊNDICE D - CONTROLE PID

```
// Task controle PID de temperatura
void vControleTemperatura(void *pvParameters) {

        while (1) {
                vTaskDelay(800 / portTICK_PERIOD_MS);
                xSemaphoreTake(myMutex, portMAX_DELAY);

                term = analogRead(35);

                xSemaphoreGive(myMutex);

                terc = int((term - 1390) / 5);

                if (liga == HIGH) {
                        erro = sp - terc;
                        p = kp * erro;
                        i += ki * erro * ct;
                        d = kd * erro / ct;
                        pid = p + i + d;
                        if (pid < 0) {
                                pid = 0;
                        }
                        if (pid > 1023) {
                                pid = 1023;
                        }

                        pout = int(pid / 1023 * 100);
                        digitalWrite(22, HIGH);
                        ledcWrite(ledChannel, pid);
                } else {
                        digitalWrite(22, LOW);
                        erro = 0;
                        p = 0;
                        i = 0;
                        d = 0;
                        pid = 0;
                        pout = 0;
                        ledcWrite(ledChannel, 0);
                }
        }
}
```

APÊNDICE E - ROTINA PARA O BOTÃO DE TROCA DO TERMOPAR

```
// Task botão e troca de termopar
void vTrocaSensor(void *pvParameters) {
        int botao = 0;
        while (1) {
                vTaskDelay(50 / portTICK_PERIOD_MS);
                botao = digitalRead(13);
                if (botao == HIGH) {
                        vTaskDelay(50 / portTICK_PERIOD_MS);
                        botao = digitalRead(13);
                        if (botao == HIGH) {
        xQueueSend(xQueueTeste, &botao, pdMS_TO_TICKS(100));
                                vTaskDelay(60000 / portTICK_PERIOD_MS);
                        }
                }
        }
}
```

APÊNDICE F - GRAVAMENTO DE DADOS NA MEMÓRIA DO MICROCONTROLADOR

```
 // Task Grava parâmetros na flash
void gravarFlash(void) {
        int address = 0;

        EEPROM.writeInt(address, 1234);
        address += sizeof(int);

        EEPROM.writeFloat(address, kp);
        address += sizeof(float);
        EEPROM.writeFloat(address, ki);
        address += sizeof(float);
        EEPROM.writeFloat(address, kd);
        address += sizeof(float);
        EEPROM.writeFloat(address, ra);
        address += sizeof(float);

        EEPROM.writeInt(address, tol);
        address += sizeof(int);
        EEPROM.writeInt(address, tick);
        address += sizeof(int);
        EEPROM.writeInt(address, al1);
        address += sizeof(int);
        EEPROM.writeInt(address, al2);
        address += sizeof(int);
        EEPROM.writeInt(address, sp);
        address += sizeof(int);
        EEPROM.commit();

  digitalWrite(2, HIGH);
  vTaskDelay(200 / portTICK_PERIOD_MS);
  digitalWrite(2, LOW);

        fkp = kp;
        fki = ki;
        fkd = kd;
        fra = ra;
        ftol = tol;
        ftick = tick;
        fal1 = al1;
        fal2 = al2;
        fsp = sp;
}
```

APÊNDICE G - LEITURA DE DADOS

```
// Task leitura de parâmetros na flash
void lerFlash(void) {
        int address = 0;
        int controle = 0;

        controle = EEPROM.readInt(address);
        address += sizeof(int);

        if (controle == 1234) {
                // Serial.println("controle = 1234");
                kp = EEPROM.readFloat(address);
                address += sizeof(float);
                ki = EEPROM.readFloat(address);
                address += sizeof(float);
                kd = EEPROM.readFloat(address);
                address += sizeof(float);
                ra = EEPROM.readFloat(address);
                address += sizeof(float);

                tol = EEPROM.readInt(address);
                address += sizeof(int);
                tick = EEPROM.readInt(address);
                address += sizeof(int);
                al1 = EEPROM.readInt(address);
                address += sizeof(int);
                al2 = EEPROM.readInt(address);
                address += sizeof(int);
                sp = EEPROM.readInt(address);
                address += sizeof(int);
        }

        fkp = kp;
        fki = ki;
        fkd = kd;
        fra = ra;
        ftol = tol;
        ftick = tick;
        fal1 = al1;
  fal1 = ral1;
        fal2 = al2;
  fal2 = ral2;
        fsp = sp;

 if ( al1 == 1 ) {
    digitalWrite(18, HIGH);
 }
 if ( al1 == 2 ) {
    digitalWrite(19, HIGH);
 }
}
```

APÊNDICE H - VERIFICAÇÃO DE DADOS

```
// Task verifica alteração nos parâmetros e grava na flash
void vMonitoraFlash(void *pvParameters) {
        while (1) {
                vTaskDelay(10000 / portTICK_PERIOD_MS);
                if ((fkp != kp) || (fki != ki) || (fkd != kd) || (fra != ra) ||
                        (ftol != tol) || (ftick != tick) || (fal1 != al1) ||
                        (fal2 != al2) || (fsp != sp)) {
                        if (ftick > tick) {
                                const int AlteraTick = 2;
                                xQueueSend(xQueueTeste, &AlteraTick, pdMS_TO_TICKS(100));
                        }
                        gravarFlash();
                }
        }
}
```

APÊNDICE I - RELACIONAMENTO DE VARIÁVEIS ENTRE A NUVEM E O MICROCONTROLADOR

```
// Task principal
void setup() {
        myMutex = xSemaphoreCreateMutex();
        xQueueTeste = xQueueCreate(3, sizeof(uint32_t));

        thing.add_wifi(SSID1, SSID_PASSWORD1);
        thing.add_wifi(SSID2, SSID_PASSWORD2);
        thing.add_wifi(SSID3, SSID_PASSWORD3);
        thing.add_wifi(SSID4, SSID_PASSWORD4);

        //Relacionamento entre variáveis da thinger.io e variáveis do ESP32.
        thing["tliga"]  << inputValue(liga);
        thing["tsp"]    << inputValue(sp);
        thing["tkp"]    << inputValue(kp);
        thing["tki"]    << inputValue(ki);
        thing["tkd"]    << inputValue(kd);
        thing["tct"]    << inputValue(ct);
        thing["ttick"]  << inputValue(tick);
        thing["ttol"]   << inputValue(tol);
        thing["tterm"]  >> outputValue(term);
        thing["tterc"]  >> outputValue(terc);
        thing["tpid"]   >> outputValue(pid);
        thing["tpout"]  >> outputValue(pout);
        thing["tal1"]   >> outputValue(al1);
        thing["tal2"]   >> outputValue(al2);
        thing["tral1"]  >> outputValue(ral1);
        thing["tral2"]  >> outputValue(ral2);
        thing["trm"]    >> outputValue(rm);
        thing["tra"]    >> outputValue(ra);
        |
}

void loop() {

        thing.handle();

        digitalWrite(21, LOW);
        vTaskDelay(20 / portTICK_PERIOD_MS);

        if (WiFi.status() == WL_CONNECTED) {
                digitalWrite(21, HIGH);
        } else {
                digitalWrite(21, LOW);
        }
        vTaskDelay(980 / portTICK_PERIOD_MS);
}
```

APÊNDICE J - TEMPO DE ALARME PARA REGISTRO EM NUVEM

```
// Task temporiza alarme para histórico em nuvem
void vResetAlarme(void *pvParameters){
  vTaskDelay(70000 / portTICK_PERIOD_MS);
  ral1 = 0;
  ral2 = 0;
  vTaskDelete(NULL);
  }
```