INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE SANTA CATARINA - CÂMPUS FLORIANÓPOLIS
DEPARTAMENTO ACADÊMICO DE ELETRÔNICA
CURSO SUPERIOR DE ENGENHARIA ELETRÔNICA

RAFAEL HILLER

DESENVOLVIMENTO DE UM SISTEMA DE TESTES TÉRMICOS PARA PRODUTOS ELETRÔNICOS

Florianópolis, 2021

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE SANTA CATARINA - CÂMPUS FLORIANÓPOLIS
DEPARTAMENTO ACADÊMICO DE ELETRÔNICA
CURSO SUPERIOR DE ENGENHARIA ELETRÔNICA

RAFAEL HILLER

DESENVOLVIMENTO DE UM SISTEMA DE TESTES TÉRMICOS PARA PRODUTOS ELETRÔNICOS

Trabalho de conclusão de curso submetido ao Instituto Federal de Educação, Ciência e Tecnologia de Santa Catarina como parte dos requisitos para obtenção do título de engenheiro eletrônico

Orientador:
Dr. Eng.Renan Augusto Starke

Florianópolis, 2021

Ficha de identificação da obra elaborada pelo autor.

Hiller, Rafael
Desenvolvimento de um sistema de testes térmicos para **produtos eletrônicos / Rafael Hiller ; orientação de** Renan **Starke. - Florianópolis, SC,** 2021.
96 p.

Trabalho de Conclusão de Curso (TCC) - Instituto Federal **de Santa Catarina, Câmpus** Florianópolis. Bacharelado **em Engenharia Eletrônica. Departamento** Acadêmico **de** Eletrônica.
Inclui Referências.

1. Testes automatizados. 2. Controlador PI. 3. Funções **de transferência de primeira ordem com atraso.** I. **Starke, Renan . II. Instituto Federal de Santa** Catarina. **Departamento Acadêmico de Eletrônica. III.** Título.

DESENVOLVIMENTO DE UM SISTEMA DE TESTES TÉRMICOS PARA PRODUTOS ELETRÔNICOS

RAFAEL HILLER

Este Trabalho foi julgado adequado para obtenção do Título de Engenheiro Eletrônico em Abril de 2021 e aprovado na sua forma final pela banca examinadora do Curso de Engenharia Eletrônica do instituto Federal de Educação Ciência, e Tecnologia de Santa Catarina.

Florianópolis, 22 de Abril, 2021.

Banca Examinadora:

Renan Augusto Starke, Dr. Eng.

Joabel Moia, Dr. Eng.

Jony Laureano Silveira, Dr. Eng.

AGRADECIMENTOS

Ao Instituto Federal de Santa Catarina (IFSC), que no decorrer de cinco anos proporcionou diversas descobertas, conhecimentos e crescimento pessoal e profissional.

Em especial aos meus avós, seu João Guilherme e dona Aurora, responsáveis por me incentivar no desejo da descoberta e na curiosidade, minha mãe, Susana, e meu Padrastro, Henrique, que me ensinaram o valor do conhecimento e da perseverança. Agradeço a eles por sempre me apoiarem em minhas decisões e nos momentos difíceis.

Aos meus familiares que me apoiaram nesta jornada.

Aos meus colegas de trabalho da Intelbras, por me ajudarem a elaborar os objetivos deste trabalho, atendendo demandas reais em um ambiente de desenvolvimento de produto.

“A gente pode começar
a verificar os ’porque não’ e
começar a criar os ’porque sim’.”
(Ozires Silva)

RESUMO

Diante da sensibilidade do desempenho dos circuitos eletrônicos às suas temperaturas máximas especificadas, este trabalho de conclusão de curso tem como objetivo desenvolver uma sistema que forneça um ambiente onde eles possam ser testados em frente a esta condição, colocando à prova seu funcionamento em temperaturas elevadas. Desta forma, os projetistas de hardware poderão elaborar seus sistemas e fornecer soluções robustas, podendo entregar um produto mais confiável ao mercado. Para tanto, são descritos conceitos de sistemas de controle, identificação de funções de transferências de sistemas à partir de resposta ao degrau e as tecnologias adotadas para tornar o projeto viável. As técnicas abordadas são utilizadas na definição da função de transferência de um forno elétrico a resistência para projetar um controlador de temperatura. Além disso, este trabalho contempla o desenvolvimento de um software de controle e monitoramento para a automatização dos testes, configurando a temperatura desejada e o tempo de duração da mesma. Por último, são apresentadas as considerações finais e recomendações de trabalhos futuros.

Palavras-chave: Testes automatizados. Controlador PI. Funções de transferência de primeira ordem com atraso.

ABSTRACT

Given the sensitivity of electronic circuits performance to high temperatures, this work aims to develop a system that provides an environment where they can be tested in the face of high temperatures. In this way, hardware designers will be able to design their systems, provide robust solutions and be able to deliver a more reliable product to the market. For that, concepts of control systems, identification of system transfer functions based on the step response and the technologies adopted to make the project viable are described. The techniques covered are used to define the transfer function of an electric oven to design a temperature controller. In addition, this work includes the development of control and monitoring software for automation of tests, configuring the desired temperature and its duration. Finally, final considerations and recommendations for future work are presented.

Keywords: Automated tests. PI controller. text editoration. First order transfer function with delay.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ADC	*Analog Digital Converter* - Conversor Analógico Digital
CAN	*Controlle Area Network* - Rede de Área do Controlador
CI	Circuito Integrado
ESP-IDF	*Espressif IoT Development Framework* - Estrutura Espressif de desenvolvimento IoT
GUI	*Graphical User Interface* - Interface Gráfica de Usuário
HTML	*Hyper Text Markup Language* - Linguagem de Marcação de Hiper Texto
HTTP	*Hypertext Transfer Protocol* - Protocolo de Transferência de Hipertexto
I2C	*Inter-integrated Circuit* - Circuito Inter-integrado
IoT	*Internet of Things* - Internet das Coisas
MOSFET	*Metal Oxide Semiconductor Field Effect Transistor* - Transistor de Efeito de Campo Metal-Óxido
PDF	*Portable Document File* - Arquivo de Documento Portátil
PID	*Proportional, integral and derivative* - Proporcional, Integrador e Derivativo
PWM	*Pulse Width Modulation* - Modulação por Largura de Pulso
SPI	*Serial Peripheral Interface* - Interface Periférica Serial
SSR	*Solid State Relay* - Relé de Estado Sólido

SUMÁRIO

1 INTRODUÇÃO

Ao longo das últimas décadas, nota-se que a presença de produtos eletrônicos está cada vez mais presente na rotina das pessoas. Diversifica-se suas funcionalidades de acordo com a demanda do mercado e com a necessidade dos consumidores.

De acordo com Wadhwani (2020), em 2019, o mercado de sistemas embarcados excedeu 100 bilhões de dólares americanos e deverá crescer à uma taxa de crescimento anual maior que 6% entre os anos de 2020 e 2026.

Com este crescimento do número de consumidores, os fabricantes devem asseverar a funcionalidade de suas soluções eletrônicas em diversos cenários, pois, desta forma, garantirão a sua fatia de mercado e se prevalecerão diante os olhos dos consumidores quando comparados aos seus concorrentes. Além disso, há também órgãos reguladores que fiscalizam e homologam os sistemas eletrônicos, com o intuito de certificar o funcionamento dos mesmos, confirmando que o produto é seguro para os usuários.

No tocante aos cenários de aplicação, uma das variáveis que mais influência no desempenho dos sistemas eletro-eletrônicos é a temperatura. Chambers (2016) ainda enfatiza que permitir que os circuitos eletrônicos funcionem por períodos prolongados em altas temperaturas pode diminuir a sua vida útil.

No Brasil, devido à sua localização geográfica, podem ser encontrados seis tipos de clima diferentes, onde sua temperatura pode variar de -10 ℃ no inverno até 40 ℃ no verão, ou seja, apresenta uma grande amplitude térmica e também temperaturas elevadas (FRANCISCO, 2020). Desta forma, projetistas devem desenvolver circuitos eletrônicos que funcionam para uma grande variação de temperatura.

Para empresas que comercializam produtos relacionados ao segmento de telecomunicações no Brasil, estes devem ser homologados pela Agência Nacional de Telecomunicações (ANATEL). O processo de homologação abrange uma série de testes nos produtos, pois eles serão qualificados para atender os consumidores de forma segura e adequada (ANATEL, 2019b). Com isso, um dos testes realizados é o ciclo térmico, onde, dependendo da classe do produto, será submetido à grandes variações de temperatura, podendo chegar até 80 ℃ de temperatura máxima, onde o mesmo deverá manter seu funcionamento (ANATEL, 2019a).

Como os testes de homologação abrangem ciclos térmicos, este trabalho visa desenvolver um ambiente onde possa testar produtos eletrônicos sob altas temperaturas, sendo que a temperatura deste ambiente e o tempo de duração poderão ser definidos previamente pelo projetista.

Este trabalho é organizado da seguinte forma, no Primeiro Capítulo descreve-se a organização e objetivos deste trabalho. Já o Segundo Capítulo é responsável pela

fundamentação teórica, abordando desde os conceitos básicos, como transferência de calor e o funcionamento básico de um forno elétrico, até as teorias matemáticas para modelagem de sistemas, além de apresentar as tecnologias utilizadas no projeto. No Terceiro Capítulo está qualificada a metodologia. O Quarto Capítulo descreve o desenvolvimento do modelo matemático do forno elétrico, o projeto do sistema embarcado responsável por controlar a temperatura do forno elétrico, o desenvolvimento do software de monitoramento e controle, e, por último, os resultados obtidos. Por fim, o Capítulo 5 exibe as considerações finais.

1.1 Justificativa

Uma vez que o mercado de eletrônicos está em constante crescimento, processos e ferramentas que garantem a qualidade e robustez se tornam essenciais. Já que, desta forma, a empresa garantirá qualidade e agilidade em seus produtos para homologação, pois se terá garantia que o sistema funcionará quando submetido ao testes das agências de regulamentação.

Tomando em conta os testes relacionados ciclo térmico durante a homologação de produtos eletrônicos, as temperaturas elevadas de determinadas regiões do Brasil e a susceptibilidade de circuitos eletrônicos às altas temperaturas, o presente trabalho focará no desenvolvimento de um sistema de baixo custo, simples e intuitivo, para que fabricantes de produtos eletrônicos possam testar suas soluções em temperaturas elevadas, certificando-se da robustez e qualidade do projeto.

1.2 Descrição do problema

Para que os fabricantes de produtos eletrônicos relacionados a telecomunicações possam comercializar seus produtos, estes devem ser homologados pela ANATEL, onde serão aplicados diversos testes, como o de elevação de temperatura. Além disso, sabe-se que no Brasil há regiões com temperaturas elevadas, ou seja, as empresas que desenvolvem soluções envolvendo hardware devem se certificar que seus produtos manterão seu funcionamento em ambientes com altas temperaturas.

Baseado nisto, é possível desenvolver um ambiente onde se pode controlar a sua temperatura para testar equipamentos eletrônicos sob a sua temperatura máxima especificada?

1.3 Objetivo geral

Adaptar um forno elétrico para realizar testes em produtos eletrônicos perante temperatura controlada de forma automatizada, de modo que os projetistas consigam integrar testes automatizados com o estresse de temperatura.

1.4 Objetivos específicos

De forma a cumprir os objetivos gerais deste trabalho, foram deliberados os seguintes objetivos específicos:

a) Revisar a bibliografia relacionada às teorias de controle e funcionamento do forno elétrico.

b) Definir o modelo matemático do sistema do forno elétrico.

c) Desenvolver um sistema embarcado para o controle do forno contemplando um servidor HTTP para definição da temperatura desejada e seu monitoramento.

d) Desenvolver software com uma interface gráfica para que o usuário possa realizar o controle e monitoramento da temperatura.

e) Avaliar a implementação do sistema desenvolvido em um ambiente de desenvolvimento de equipamentos eletrônicos.

f) Avaliar os resultados obtidos com a utilização do sistema.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo tem como objetivo fornecer os conceitos teóricos que estão relacionados ao desenvolvimento do projeto, para que cada objetivo específico seja atingido. Além disso, serão revisadas as tecnologias utilizadas para realizar a comunicação dos parâmetros do controlador com aplicações externas e os componentes utilizados.

2.1 Transferência de Calor

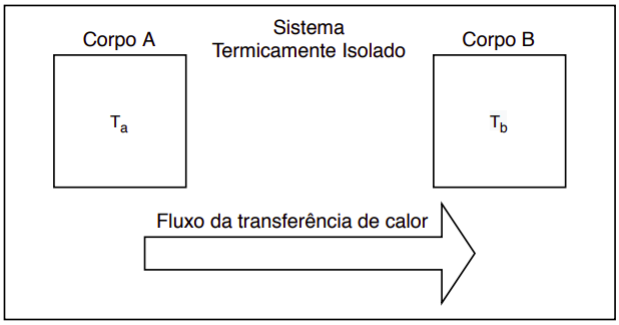
Como o projeto a ser desenvolvido se baseia em uma aplicação de controle de um forno elétrico, é válido fornecer alguns conceitos básicos sobre calor e as formas de que sua transferência pode ocorrer.

Para Schulz (2009), calor é definido como a transferência de energia térmica entre corpos que possuem temperaturas distintas. Além disso, ele ressalta que calor e temperatura não estão diretamente relacionados, porém a diferença de temperatura entre dois corpos gera a transferência de energia térmica, ou seja, calor.

Segundo Curado (2020), define-se temperatura como uma grandeza física que pode ser descrita como o grau de agitação das moléculas que compõem um corpo, sendo responsável pela caracterização térmica de um sistema ou de um corpo

A Figura 1 representa, de forma simplificada, um sistema térmico isolado onde há dois corpos, Corpo A e Corpo B, com temperaturas T_a e T_b, respectivamente, sendo que $T_a > T_b$, desta forma, o fluxo de calor se dá do Corpo A para o Corpo B.

Figura 1 – Representação simplificada da transferência de calor



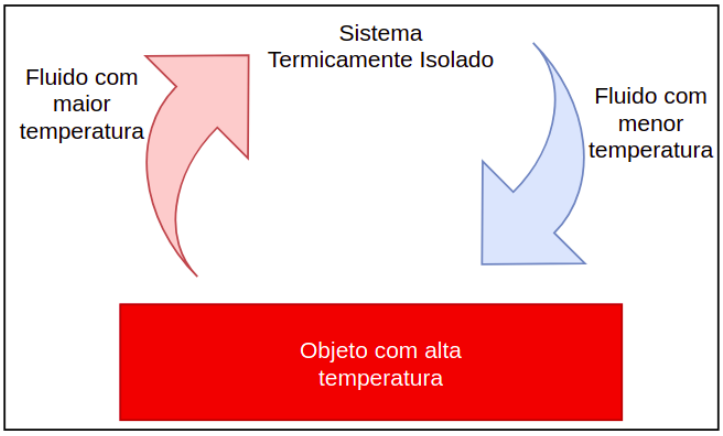
Fonte: Elaboração própria (2020).

A transferência de calor pode ser realizada através de três formas, condução, convecção e radiação.

Convecção se caracteriza pelo tipo de transferência de energia térmica se encontra no interior do forno elétrico, onde um fluido, neste caso o ar, entra em contato com um objeto (resistência) cuja a temperatura é maior que a do fluido. Como

a temperatura do fluido que está em contato com o objeto quente aumenta, esta parte do fluido se expande, tornando-se menos densa. Já que o fluido expandido é mais leve do que o fluido que o cerca, de menor temperatura, a força do empuxo o faz subir. Desta forma, o fluido de menor temperatura, consequentemente, de maior densidade, escoa para tomar o espaço do fluido que se expande, e assim, o processo continua indefinidamente (HALLIDAY; WALKER, 2009).

Figura 2 – Representação simplificada da transferência de calor por convecção



Fonte: Elaboração própria (2020).

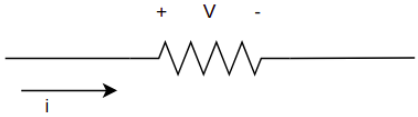
Halliday e Walker (2009) explica que transferência de energia térmica por condução é descrita através da colisões entre os átomos de um objeto que se encontra em alta temperatura. Essas colisões ocorrem devido às altas vibrações dos átomos do material.

Por último, cabe explicar sobre a radiação. Halliday e Walker (2009) relata que esta forma de transferência de calor advém da da troca de energia por ondas eletromagnéticas, sendo que esta é a única forma de transferência de energia no vácuo.

2.2 Resistência

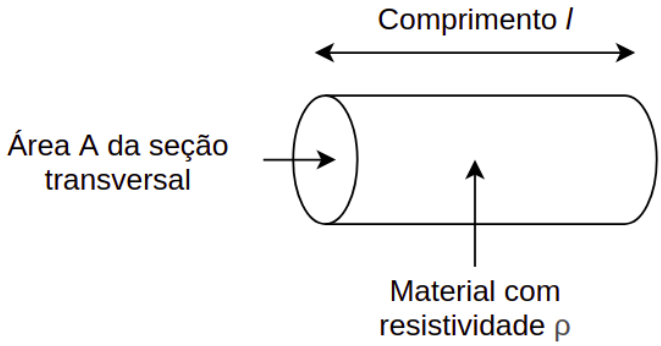
A oposição dos materiais à passagem de corrente, ou de forma mais específica, ao movimento de cargas elétricas, é denominada resistência (NILSSON; RIEDEL, 1999). Em circuitos, elétricos esta propriedade é presentada pelo resistor e seu valor pode ser calculado através da Equação 2.1.

Figura 3 – Representação do resistor em circuitos elétricos



Fonte: Elaboração própria (2021).

Figura 4 – Representação das partes do resistor



Fonte: Elaboração própria (2021).

$$R = \rho * \frac{l}{A} \tag{2.1}$$

Onde:

R : Resistência do elemento (Ω)
ρ : Resistividade do material ($\Omega * m$)
l : Comprimento do elemento (m)
A : Área da seção transversal do elemento (m^2)

Conceitualmente, pode-se entender a resistência se for lembrado de que os elétrons em movimento, que constituem a corrente elétrica, interagem com a estrutura atômica do material o qual atravessam. Com essas interações, parte da energia elétrica se transforma em energia térmica, sendo dissipada em forma de calor (NILSSON; RIEDEL, 1999). Este fenômeno é denominado Efeito Joule.

Para alguns casos, este efeito pode ser indesejado, sendo necessário aplicar componentes e técnicas para dissipar esta energia de modo a não sobreaquecerem os componentes. Porém, para as aplicações que envolvem aquecimento, este efeito é a forma de como estas aplicações funcionam, como, por exemplo, fornos elétricos, estufas elétricas, entre outros.

2.3 Forno Elétrico à Resistência

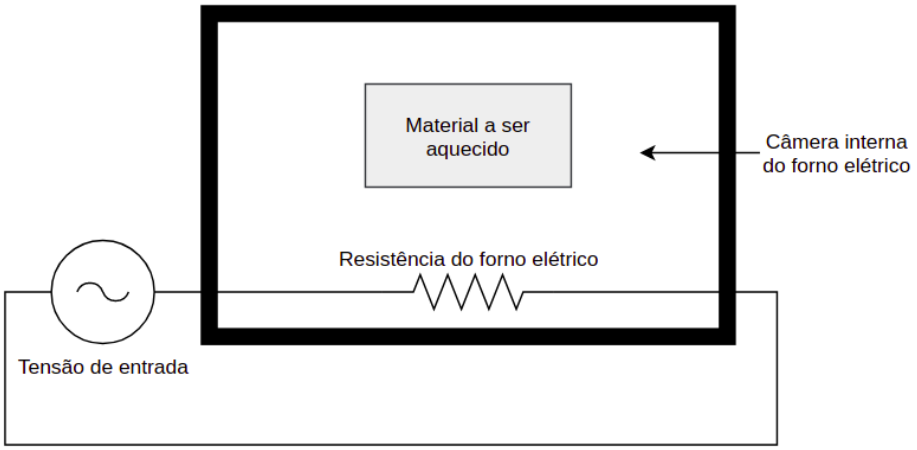
Neste trabalho, será utilizado um forno elétrico à resistência para implementar o ambiente de temperaturas elevadas onde o produto eletrônico será testado.

Como explicado anteriormente, parte da energia resultante da interação das cargas elétricas com a estrutura atômica do material é transformada em energia térmica. Com isso, fornos elétricos à resistência são aqueles que utilizam o calor gerado

por perdas Joule em uma resistência elétrica que é atravessada por uma corrente, que, normalmente, é de alta intensidade (FILHO, 2007).

Dentro do segmento de fornos elétricos, são encontrados dois principais modelos: os de aquecimento direto e aquecimento indireto. O primeiro é utilizado em aplicações específicas, onde o material que será trabalhado é posicionado entre os dois eletrodos do circuito. Já o segundo, o de aquecimento indireto, é o mais comum de ser encontrado. Neste tipo de forno, o material a ser trabalhado é localizado em um câmara com determinado isolamento térmico, e o calor é transferido da resistência através do fenômeno de condução, convecção e irradiação (FILHO, 2007).

Figura 5 – Esquemático do forno elétrico de aquecimento indireto



Fonte: Elaboração própria (2021).

2.4 Identificação de um Sistema por uma Função de Transferência de Primeira Ordem

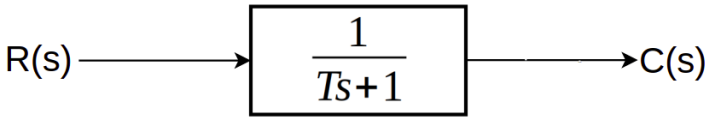
Para um sistema de primeira ordem $G(s)$, sua função de transferência é dada por:

$$\frac{C(s)}{R(s)} = \frac{1}{Ts+1} \tag{2.2}$$

Onde:

C(s) : Sinal de saída do sistema
R(s) : Sinal de entrada do sistema
T : Constante de tempo do sistema

Figura 6 – Diagrama de blocos de um sistema de primeira ordem



Fonte: Elaboração própria (2021).

Com isso, para uma entrada do tipo degrau unitário, ou seja, $R(s) = 1/s$, pode-se descrever a saída do sistema como:

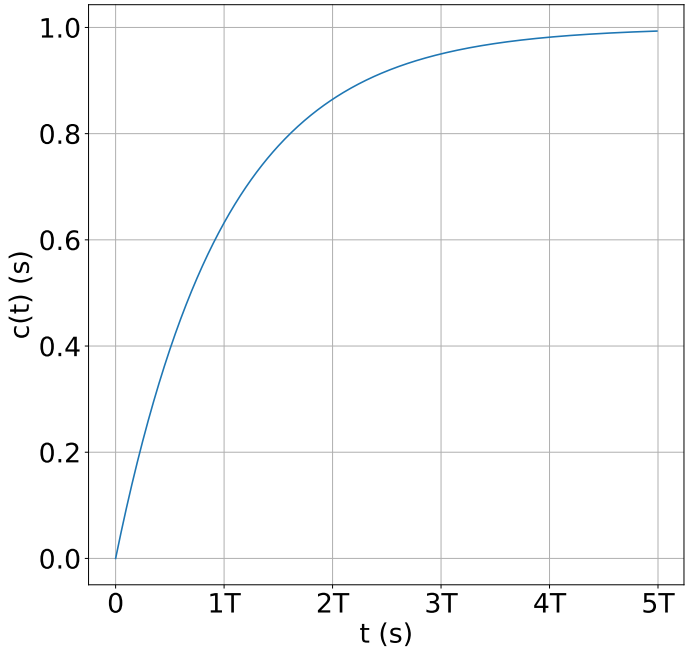
$$C(s) = \frac{1}{Ts+1}\frac{1}{s} = \frac{1}{s} - \frac{T}{Ts+1} = \frac{1}{s} - \frac{1}{s+(1/T)}$$

Realizando a transformada inversa de Laplace do resultado acima, obtém-se a seguinte função da saída do sistema no domínio do tempo:

$$c(t) = \begin{cases} 1 - e^{-t/T}, & \text{para } t \geq 0 \\ 0, & \text{t < 0} \end{cases}$$

Desta forma, a representação gráfica de maneira genérica de um sistema de primeira ordem é mostrada na Figura 7.

Figura 7 – Resposta ao degrau unitário de um sistema de primeira ordem



Fonte: Elaboração própria (2021).

Analisando a imagem acima, sabe-se que o valor final da saída do sistema será um. Baseado nisto, é válido afirmar que, sabendo valor final da saída do sistema, é

possível determinar a constante de tempo, pois, quando o tempo for equivalente a duas constantes de tempo, o valor da saída do sistema será 86,5 % do valor final (OGATA, 2009).

2.5 Identificação de um Sistemas com função de Transferência de Primeira Ordem com Atraso

Como será visto no Capítulo 4, o comportamento de um forno pode ser aproximado com uma função de transferência de primeira ordem com atraso. Sendo assim, será necessário apresentar como esta identificação será realizada.

A função de transferência de um sistema de primeira ordem $G(s)$, com atraso α, é descrito por:

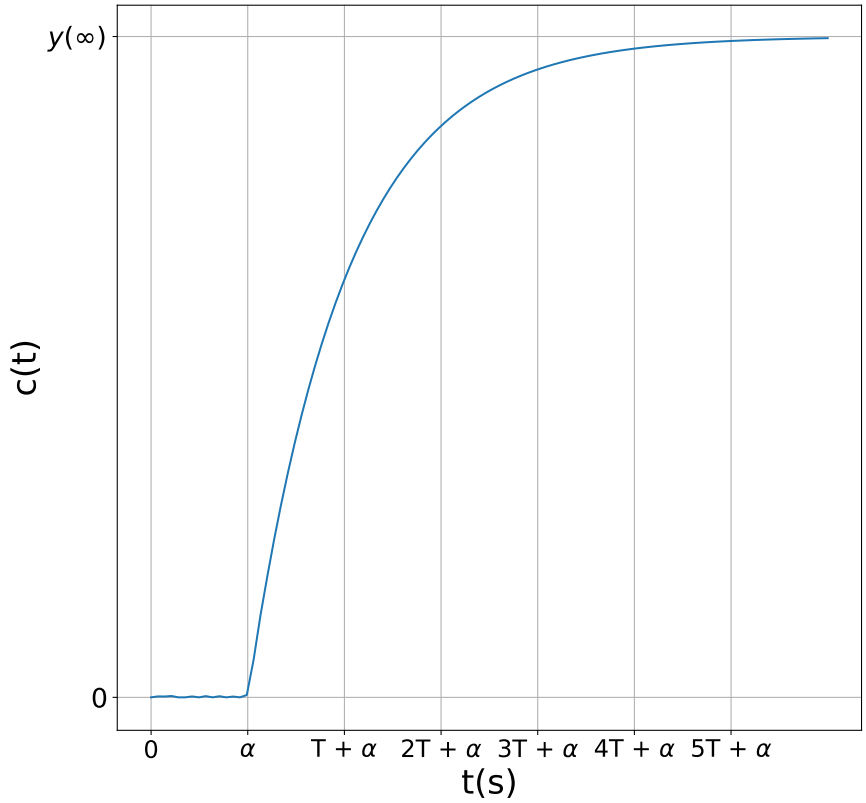
$$G(s) = \frac{C(s)}{R(s)} = \frac{k}{Ts+1} e^{-\alpha s} \tag{2.3}$$

Com base na equação acima, a corresponde resposta de G(s) para um degrau de amplitude A, inciado em $t = 0$, é descrita por:

$$c(t) = \begin{cases} kA(1 - e^{-t/T}), & t \geq \alpha \\ 0, & t < \alpha \end{cases} \tag{2.4}$$

Desta forma, a curva será representada de forma semelhante à Figura 7, porém com um determinando intervalo onde a saída do sistema não será alterada, conforme mostra a Figura 8

Figura 8 – Resposta ao degrau de amplitude *A* de um sistema de primeira ordem com atraso



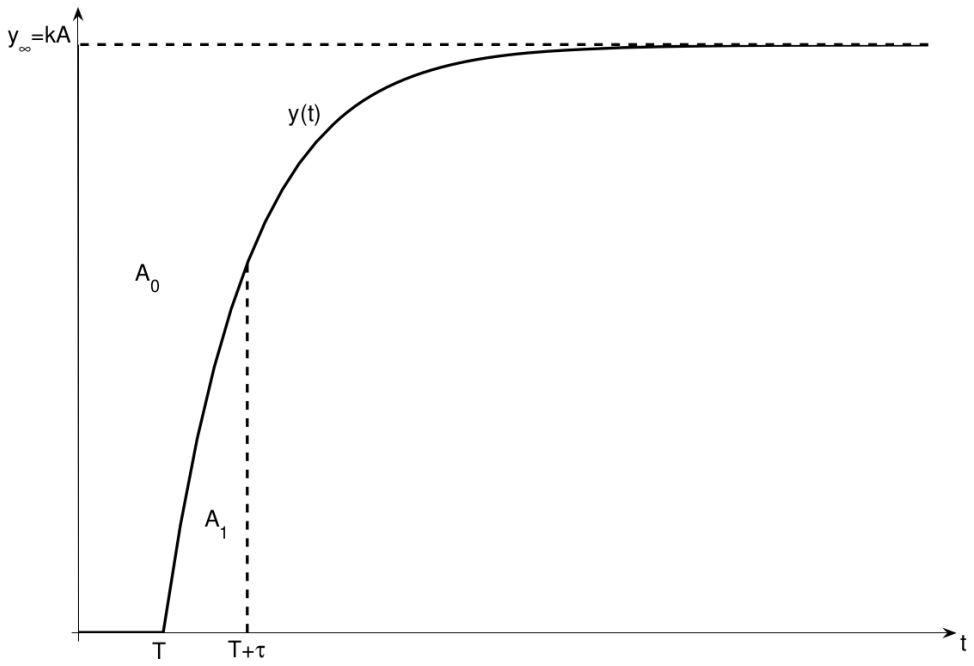
Fonte: Elaboração própria (2021).

Esta análise pode ser feita para sistemas de segunda ordem superamortecidos, permitindo uma simplificação da função de transferência com apenas um polo.

Observando a Equação 2.3, pode-se concluir que será necessário determinar três parâmetros da função de transferência: o ganho k, a constante de tempo T e o tempo de atraso α.

O método para o definir estes parâmetros será baseado no cálculo das duas áreas, A_0 e A_1, no gráfico da Figura 9.

Figura 9 – Indicação das áreas que deverão ser calculadas através da resposta ao degrau



Fonte: Guerra (2006).

Sabendo que os parâmetros k, T e α da Equação 2.3 são desconhecidos e tomando como base a Figura 9 e a equação 2.3, é possível afirmar que o ganho será a razão entre o valor de regime permanente da saída do sistema e o valor da amplitude do sinal de entrada, conforme mostra a Equação 2.5 (ÅSTRÖM; HÄGGLUND, 1995).

$$k = \frac{c(\infty)}{A} \tag{2.5}$$

Onde:

k : O ganho da equação de transferência do sistema
$c(\infty)$: O valor da saída do sistema em regime permanente
A : Amplitude do sinal de entrada

Após a determinação do ganho k, a constante de tempo e o atraso da resposta do sistema podem ser calculados através das áreas A_0 e A_1, de acordo com as equações 2.6 e 2.7, respectivamente (ÅSTRÖM; HÄGGLUND, 1995).

$$A_0 = \int_0^\infty [kA - c(t)]dt = kA(T + \alpha) \tag{2.6}$$

Onde:

A_0 : Área entre à linha da saída do sistema o valor de regime permanente
c(t) : Saída do sistema
k : O ganho da equação de transferência do sistema
A : Amplitude do sinal de entrada
T : Constante de tempo do da função de transferência do sistema
α : Tempo de atraso do sistema

$$A_1 = \int_0^{\alpha+T} c(t)dt = \int_0^{\alpha+T} kA(1 - e^{-t/T})dt = kATe^{-1} \rightarrow T = \frac{A_1 e}{kA} \tag{2.7}$$

Onde:

A_1 : Área entre à linha da saída do sistema e a abscissa
c(t) : Saída do sistema
k : O ganho da equação de transferência do sistema
A : Amplitude do sinal de entrada
T : Constante de tempo do da função de transferência do sistema
α : Tempo de atraso do sistema
e : Número de Euler
t : tempo (s)

Por fim, substituindo-se α na Equação 2.6, obtêm-se:

$$A_0 = kA\alpha + kA\frac{A_1 e}{kA} \rightarrow \alpha = \frac{A_0 - A_1 e}{kA} \tag{2.8}$$

Com as equações para a determinação dos parâmetros da equação de transferência já definidas, é possível modelar um sistema de segunda ordem superamortecido de forma muito similar a um sistema de primeira ordem com atraso, como será visto no Capítulo 4.

2.6 Aproximação de Padé de uma Função de Transferência com Atraso

Visto que a função de transferência de um sistema de primeira ordem com atraso é dado pela Equação 2.3, nota-se que ela é composta por uma função irracional, sendo esta parte termo que compõe o atraso, descrito por $e^{-\alpha s}$.

Algumas técnicas de sistemas de controle requerem uma função racional. Com este objetivo, a aproximação de Padé é usada com frequência para este caso (BAQUETTE, 2002).

A aproximação de Padé consiste na aproximação de uma função não racional por uma função racional (BAKER; GRAVES-MORRIS, 1996). Assim, é possível

aproximar funções mais complexas usando uma função racional do tipo $R_{m,n}(x)$, representada pela Equação 2.9 (HANTA; PROCHáZKA, 2009):

$$R_{m,n} = \frac{Q_m(x)}{P_m(x)} = \frac{b_0 + b_1 x + b_2 x^2 + ... + b_n x^n}{a_0 + a_1 x + a_2 x^2 + ... + a_n x^n} \tag{2.9}$$

Os coeficientes para ambos os polinômios são calculados de maneira que os primeiros $m + n + 1$ termos desapareçam na séries de Taylor das funções aproximadas. Primeiro, a função exponencial é desenvolvida em séries de Taylor na origem em $m + n + 1$ termos, e, então, aproximada por uma função racional ($q_0 = 0$) (HANTA; PROCHáZKA, 2009):

$$e^{\alpha s} \approx \sum_{i=0}^{m+n} (-1)^i \frac{(\alpha s)^i}{i!} = \frac{\sum_{i=0}^{m} p_i (\alpha s)^i}{\sum_{i=0}^{n} q_i (\alpha s)^i} \tag{2.10}$$

Removendo o denominador e considerando que a ordem dos polinômios são iguais, um conjunto de equações para definição dos coeficientes pode ser obtido. Baseando-se nisso, os termos p_i e q_i são determinados de acordo com as equações 2.11 e 2.12, respectivamente (HANTA; PROCHáZKA, 2009):

$$p_i = (-1)^i \frac{(2n-1)!n!}{(2n)!i!(n-i)!} \qquad \text{i} = 0, 1, ..., n \tag{2.11}$$

$$q_i = \frac{(2n-1)!n!}{(2n)!i!(n-i)!} \qquad \text{i} = 0, 1, ..., n \tag{2.12}$$

Onde:

p_i : Coeficiente do numerador da Equação 2.10
q_i : Coeficiente do denominador da Equação 2.10
n : Ordem dos polinômios da aproximação de Padé
i : Valor do índice da somatória da Equação 2.10

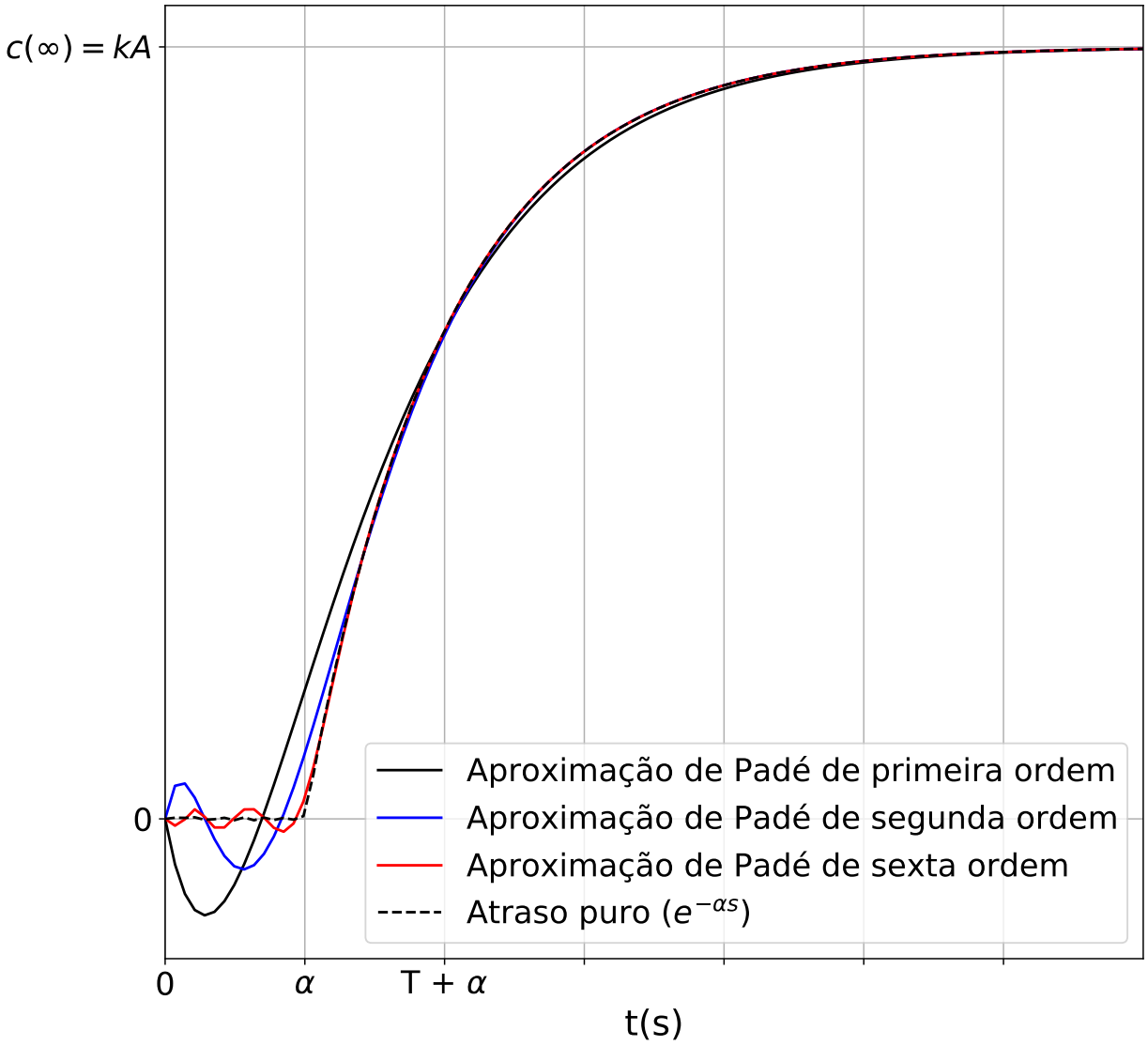
Sendo assim, para o tempo de atraso α do sistema de primeira ordem com atraso, a aproximação de Padé de primeira e segunda ordem são expressas pelas equações 2.13 e 2.14, respectivamente (BAQUETTE, 2002).

$$e^{-\alpha s} \approx \frac{\frac{-\alpha}{2} s + 1}{\frac{\alpha}{2} s + 1} \tag{2.13}$$

$$e^{-\alpha s} \approx \frac{\frac{\alpha^2}{12} s^2 - \frac{\alpha}{2} s + 1}{\frac{\alpha^2}{12} s^2 + \frac{\alpha}{2} s + 1} \tag{2.14}$$

Com a aproximação da função exponencial $e^{-\alpha s}$ em uma função racional, pode-se representar graficamente a resposta do sistema de primeira ordem substituindo a função exponencial por suas aproximações, conforme mostra a Figura 10.

Figura 10 – Comparação da resposta ao degrau com sistema de primeira ordem com atraso puro e as aproximações de Padé



Fonte: Elaboração própria (2021).

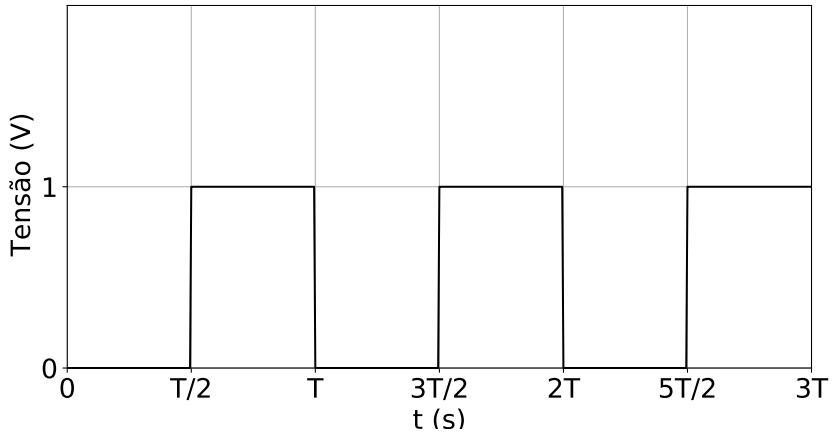
2.7 Modulação por Largura de Pulso

A técnica de Modulação por Largura de Pulso (*Pulse Width Modulation* - PWM) utiliza uma saída digital de um sistema microcontrolado para controlar o valor médio de uma aplicação. O PWM é utilizado em diversos setores e aplicações, desde instrumentação e comunicação, até controle de potência e conversão de energia (GUERRA, 2006).

O uso dos sinais de PWM é baseado no valor médio de uma forma de onda periódica. Considerando um sinal digital, ou seja, possui amplitudes finitas, seu valor médio é controlado pelo instante de tempo em que o mesmo fica em nível lógico

alto. Esta relação entre o período do sinal e período que o sinal se encontra em nível lógico alto é definido como razão cíclica, também conhecido como *Duty Cycle* (LIMA; VILLAÇA, 2012).

Figura 11 – Exemplo de PWM com razão cíclica de 50%



Fonte: Elaboração própria (2021).

Para qualquer sinal digital, seu valor médio pode ser encontrado de acordo com a Equação 2.15 (LIMA; VILLAÇA, 2012).

$$V_{med} = \frac{T_{on}}{T} * V \tag{2.15}$$

Onde:

V_{med} : Valor médio da tensão de saída (V)
V : Tensão máxima do PWM (V)
T : Período do sinal (s)
T_{on} : Tempo ativo no período (s)

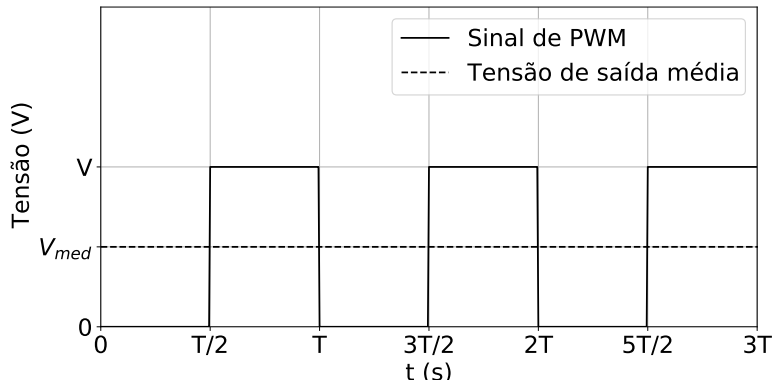
De forma geral, a Equação 2.15 pode ser obtida a partir da Equação 2.16, sendo que a primeira pode ser utilizada para calcular o valor médio de qualquer função periódica (GUERRA, 2006).

$$V_{med} = \frac{1}{T} \int_0^T v(t)dt \tag{2.16}$$

Onde:

V_{med} : Valor médio da tensão de saída (V)
$v(t)$: Função que representa a tensão do sinal periódico (V)
T : Período do sinal (s)

Figura 12 – Exemplo de PWM com ciclo ativo de 50% do período e o valor médio da tensão de saída

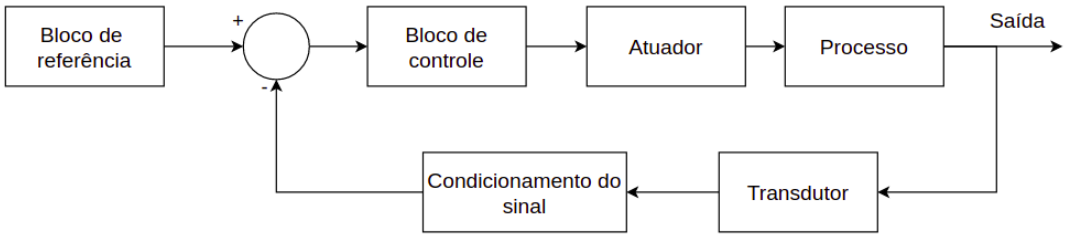


Fonte: Elaboração própria (2021).

2.8 Planta de controle

Antes de aprofundar na explicação dos termos que compõem o controlador Proporcional, Integrador e Derivativo (*Proportional, integral and derivative* - PID), é válido demostrar os componentes que formam uma planta de controle, sendo estes representados na figura abaixo:

Figura 13 – Diagrama dos componentes principais da planta de controle



Fonte: Adaptado de Johson e Moradi (2005).

Com base Figura 13, o bloco Processo é o próprio sistema em que determinadas variáveis físicas precisam ser controladas. Em processos industriais, alguns exemplos são: fornos, motores e torres de destilação (BAQUETTE, 2002).

Já o bloco Atuador, é a unidade responsável por fornecer material ou potência para a entrada do sistema. Análogo a este, o bloco Transdutor será quem transformará as grandezas físicas da saída do sistema para o domínio físico do bloco de controle. Em conjunto ao Transdutor, tem-se o bloco de Condicionamento de Sinal, realizando a preparação do sinal que será computado pelo bloco de controle (BAQUETTE, 2002).

Por fim, há o bloco de Controle, que tem como objetivo processar o valor do bloco de Referência e o valor da saída do sistema e providenciar um novo valor que será atribuído à entrada do sistema. O bloco de Controle é onde será projetado o controlador.

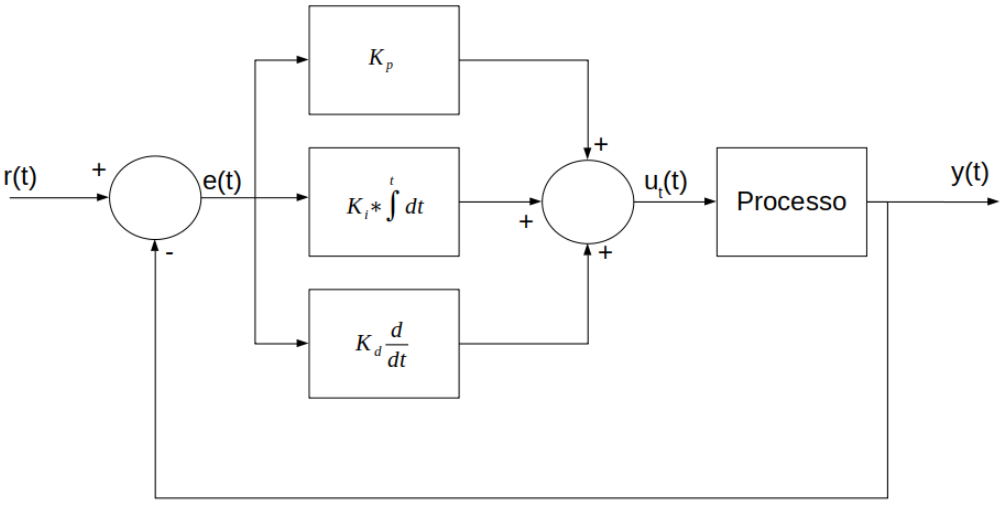
2.9 Controle PID

Para Johson e Moradi (2005), controle PID é um nome utilizado para controle com três termos. A sigla PID se refere a primeira letra dos nomes que compõem este controle, sendo P para o Proporcional, I para o integrador e D para o derivativo.

Johson e Moradi (2005) ainda informa que os controladores PID são os mais usados em aplicações industriais. Até mesmo sistemas de controle mais complexos, onde é empregado uma rede de controle, o bloco principal de controle é um controlador PID.

Ogata (2009) cientifica que a grande utilidade de controladores PID se deve a sua aplicabilidade a maioria dos sistemas de controle. Especialmente os sistemas onde não se há o modelo matemático da planta e não é possível utilizar os métodos de projetos analíticos, os controladores PID se mostram como a melhor opção. Sabe-se que no segmento de controle de sistemas, os controladores PID provaram sua proficiência fornecendo um controle satisfatório, porém, em algumas situações, eles podem não garantir um controle ótimo.

Figura 14 – Diagrama de blocos dos componentes do controlador PID



Fonte: Elaboração própria (2021).

2.9.1 Controle Proporcional

Considerando o sinal de erro da Equação 2.17, este componente é utilizado quando a saída do controlador deve ser proporcional ao valor do erro (JOHSON; MORADI, 2005).

$$e(t) = r(t) - y(t) \tag{2.17}$$

Onde:

$e(t)$: Valor do erro do sinal
$r(t)$: Valor de referência
$y(t)$: Valor da saída do sistema

A representação do controle proporcional estão descritas pelas equações 2.18 e 2.19, sendo a primeira no domínio do tempo e a segunda no domínio da frequência (JOHSON; MORADI, 2005).

$$u_c(t) = k_p e(t) \tag{2.18}$$

$$U_c(s) = k_p E(s) \tag{2.19}$$

Onde:

$u_c(t)$ e $U_c(s)$: Sinal de saída do controlador
k_p : Ganho do componente Proporcional
$e(t)$ e $E(s)$: Sinal de erro

2.9.2 Controle Integrador

A ação integral é utilizada quando deseja-se um erro de regime permanente nulo (JOHSON; MORADI, 2005).

Segundo Vasconcellos (2017), como o controle Integrador é a soma de todos os erros anteriores da entrada do sistema, pode-se, então, interpretar esta ação como armazenadora de energia do sistema. Assim, quando o sinal de erro em sua entrada for nula, a saída do bloco integral será um valor proporcional a esta energia armazenada.

As representações no domínio do tempo e de Laplace do controle integral estão representadas pelas equações 2.20 e 2.21, respectivamente (JOHSON; MORADI, 2005).

$$u_c(t) = k_i \int^t e(t)dt \tag{2.20}$$

$$U_c(s) = \left[\frac{k_i}{s}\right] E(s) \tag{2.21}$$

Onde:

$u_c(t)$ e $U_c(s)$: Sinal de saída do controlador
k_i : Ganho do componente Integrador
$e(t)$ e $E(s)$: Sinal de erro

Para a utilização do componente integrador do controlador em sistemas discretos, Microchip (2016) diz que a integral pode ser aproximada através do somatório, resultando em:

$$\int_0^t e(t)dt \approx T \sum_{k=0}^{n} e(k) \tag{2.22}$$

Onde:

T : Período de amostragem (s)
k : Índice da amostra atual
n : Número de amostras

2.9.3 Controle Derivativo

Este componente tem como entrada a variação do sinal de erro, atuando com previsões. Com isso, para implementar o controle derivativo deve haver mais cautela do que quando utilizado o controle integral ou proporcional, pois, para a maioria das aplicações reais, um controle derivativo puro não pode ser implementado devido à possibilidade da amplificação do ruído do sinal de erro medido. (JOHSON; MORADI, 2005).

As equações que representam o comportamento do componente Derivativo no domínio do tempo e da frequência são expressas através das equações 2.23 e 2.24 (JOHSON; MORADI, 2005).

$$u_c(t) = k_d \frac{de}{dt} \tag{2.23}$$

$$U_c(s) = [k_d s]E(s) \tag{2.24}$$

Onde:

$u_c(t)$ e $U_c(s)$: Sinal de saída do controlador
k_d : Ganho do componente Derivativo
$e(t)$ e $E(s)$: Sinal de erro

O comportamento do componente Derivativo no domínio discreto pode ser aproximado pela razão da diferença entre os sinais de erro o período de amostragem, conforme a Equação 2.25 (MICROCHIP, 2016).

$$\frac{de}{dt} \approx \frac{e(n) - e(n-1)}{T} \tag{2.25}$$

Onde:

T : Período de amostragem (s)
$e(n)$: Valor do erro atual
$e(n-1)$: Valor do erro anterior

2.9.4 Topologias do Controle PID

A implementação do controle PID pode-se implementar de diversas formas, sendo as três principais a ISA, Série e Paralelo.

Para ÅSTRÖM e HÄGGLUND (1995), a arquitetura ISA tem como característica as ações proporcional, integral e derivativa não interagirem com o domínio do tempo. Esta topologia admite zeros complexos, tornando-se útil para controlar sistemas com polos oscilatórios. A Equação 2.26 representa o a função de transferência do controle PID com esta topologia.

$$G(s) = K(1 + \frac{1}{sT_i} + sT_d) \tag{2.26}$$

Onde:

$G(s)$: Saída do sistema
K : Ganho do controlador
T_i : Constante de tempo do controle integral
T_d : Constante de tempo do controle derivativo

Em relação à topologia série, também conhecida como clássica, ÅSTRÖM e HÄGGLUND (1995) afirmam que esta arquitetura é obtida, naturalmente, quando um controlador é implementado como um dispositivo analógico baseado em um sistema de equilíbrio de força pneumática. Por este motivo esta topologia é referenciada como clássica. ÅSTRÖM e HÄGGLUND (1995) ainda explicam que a topologia série tem uma interpretação atrativa no domínio da frequência, devido ao zeros corresponderem aos valores inversos das constantes de tempo integrai e derivativo. Sua função de transferência é representada pela Equação 2.27.

$$G(s) = K(1 + \frac{1}{sT_i})(1 + sT_d) \tag{2.27}$$

Onde:

$G(s)$: Saída do sistema
K : Ganho do controlador
T_i : Constante de tempo do controle integral
T_d : Constante de tempo do controle derivativo

Por fim, a topologia Paralela, com a sua função de transferência representada pela Equação 2.28, é a forma mais geral, pois as ações proporcional, integral e derivativa puras são obtidas com parâmetros finitos. Assim, esta topologia se torna a mais flexível e, por isso, será a topologia escolhida para implementar o controlador de temperatura (ÅSTRÖM; HÄGGLUND, 1995).

$$G(s) = K_P + \frac{K_I}{s} + sK_d \tag{2.28}$$

Onde:

$G(s)$: Saída do sistema
K_p : Ganho da ação proporcional
K_i : Ganho da ação integral
K_d : Ganho da ação derivativa

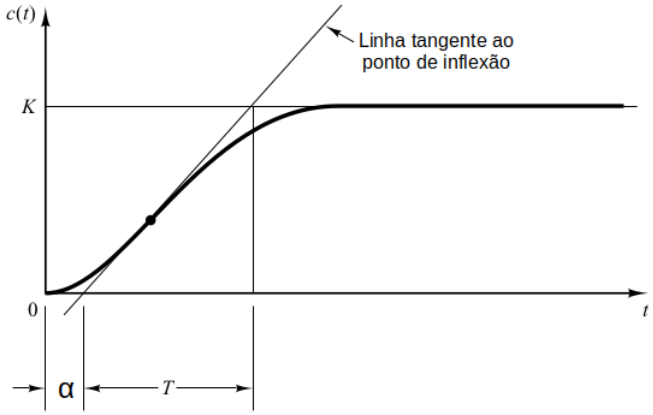
2.9.5 Regras de Ziegler-Nichols para Definição dos Ganhos dos Controladores PID

Para determinar os valores do ganho proporcional K_p, o ganho do componente integrador K_i e o ganho do componente derivativo K_d, Ziegler e Nichols propuseram uma regra onde estes termos são calculados de acordo com o a reposta transiente de uma determinada planta (OGATA, 2009).

Ogata (2009) informa que existem dois métodos para ajustar os ganhos dos componentes do controlador PID, método de Malha Aberta e de Malha Fechada, porém, como será visto no Capítulo 4, o tipo de resposta do forno elétrico enquadra-se no tipo de resposta para utilizar o método de Malha Aberta, por isso somente este será abordado.

Para determinar os parâmetros do controlador utilizando o método de Malha Aberta, deve-se obter uma reposta ao degrau da planta, em malha aberta, experimentalmente ou dinamicamente através de uma simulação, conforme a Figura 15. Este tipo de resposta implica que não envolve integradores nem polos complexos conjugados dominantes. Com isso, o sinal da saída do sistema exibirá uma curva com forma de S (OGATA, 2009).

Figura 15 – Resposta ao degrau de um sistema em formato "S"



Fonte: Adaptado de Ogata (2009).

Analisando a Figura 15, pode-se considerar que a curva com forma de “S“ é possível de ser caracterizada através de duas constantes, o tempo de atraso α e a constante de tempo T. Com o tempo de atraso e a constante do sistema, os ganhos dos componentes do controlador PI podem ser encontrados com as equações 2.29 e 2.30. Já para o controlador PID, os termos podem ser encontrados com as equações 2.31, 2.32 e 2.33 (OGATA, 2009).

$$K_p = 0.9\frac{T}{\alpha} \tag{2.29}$$

$$K_i = \frac{0,27T}{\alpha^2} \tag{2.30}$$

$$K_p = 1,2\frac{T}{\alpha} \tag{2.31}$$

$$K_i = 0,6\frac{T}{\alpha^2} \tag{2.32}$$

$$K_d = 0,6T \tag{2.33}$$

Com estas equações, será possível projetar um controlador para um sistema que apresenta um comportamento de primeira ordem com atraso. As variáveis T e α poderão ser calculadas através das equações mostradas na Seção 2.5.

O sistema a ser controlado vai ser modelado por uma função de primeira ordem com atraso não dominante e, como será visto na Seção 4.2, os requisitos do

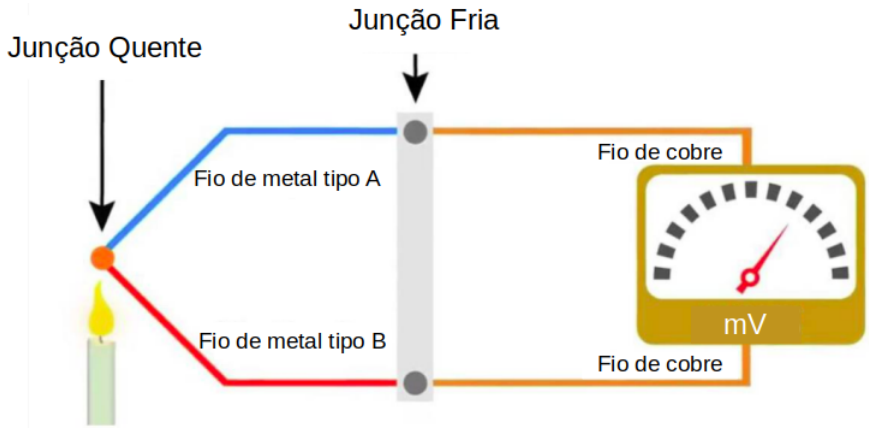
projeto do controlador é não haver sobressinal e erro em regime permanente nulo. Para isso, foi escolhido o controlador PI para controlar a temperatura no interior do forno elétrico. Assim, deverão ser definidos os ganhos K_i e K_p representados pelas equações 2.30 e 2.29 respectivamente.

2.10 Termopar

No desenvolvimento deste projeto, será utilizado um termopar como sensor de temperatura para realizar a medição no interior do forno elétrico.

Termopares são sensores compostos por dois metais que, para uma diferença de temperatura nestes metais, uma tensão será gerada. O valor desta tensão é dependente do valor da temperatura. Assim, o princípio físico que define o funcionamento do termopar é o efeito Seebeck. Além disso, como estes sensores são, normalmente, empregados em ambientes severos, os termopares precisam ser projetados de forma robusta. (PURWAR; DEEP, 2018).

Figura 16 – Esquemático das partes do termopar e seu funcionamento



Fonte: Adaptado de Omega (2019b).

Sobre a classificação dos tipos de termopares, Omega (2019a) afirma que os termopares podem ser encontrados em diferentes combinações de metais e calibrações, sendo que os mais comuns são os termopares de "*Metal Base*", conhecidos como Tipo J, K, T, E e N. Além desses modelos, ainda há as calibrações de altas temperaturas, denominados de Termopares de Metais Nobre, sendo o Tipo R, S, C e GB relacionados a esta categoria.

Guerra (2006) demonstra a equação para o modelo matemático do termopar, sendo esta representada pela Equação 2.34.

$$v(t) = K_t\theta(t) + b \tag{2.34}$$

Onde:

$v(t)$: Tensão fornecida pelo termopar no instante t (V)
K_t : Ganho
$\theta(t)$: Temperatura no interior do forno no instante t (℃)
b : Constante

O tipo de termopar escolhido para este trabalho é o Tipo K e o modelo é o *part number* F0028-10. Os motivos da sua escolha se dão por, primeiro, o sensor já possui uma estrutura de fixação para chapas metálicas, podendo ser perfeitamente posicionado em qualquer posição no interior do forno elétrico. Segundo, no mercado, encontram-se algumas opções de circuitos integrados dedicados para a leitura do seu valor de tensão e fornecendo uma interface de comunicação. Assim, o projeto se torna mais fácil por não ter de projetar um circuito para fazer o condicionamento do sinal para o conversor A/D do microcontrolador realizar a leitura. Desta forma, o microcontrolador lerá o valor da temperatura no CI dedicado através um protocolo de comunicação. Quanto à escala de temperatura do termopar, o modelo escolhido poderá operar em temperaturas entre 0 ℃ e 400 ℃.

2.11 MAX6675

O MAX6675 é um CI (Circuito Integrado) utilizado para digitalização de leitura de tensões de termopares Tipo K. Este CI pode ser aplicado em conjunto de um microcontrolador. Entre as suas principais características estão, conversão de temperatura, compensação de junção fria, digitalização e fornecimento de uma interface serial para leitura dos dados (MAXIM INTEGRATED, 2014).

Durante o período de digitalização, o MAX6675 incluí *hardware* para condicionamento do sinal para converter o sinal do termopar em uma tensão compatível com as entradas do Conversor Analógico Digital (*Analog Digital Converter* - ADC). Antes de converter as tensões termoelétricas em valores de temperatura equivalentes, é necessário realizar a compensação da região de junção fria (MAXIM INTEGRATED, 2014).

Como mencionado na Seção 2.10, o termopar produz uma tensão relacionada a diferença de temperatura entre duas junções, uma que está realizando a medida, denominada de junção quente, e outra que é denominada como junção fria, conforme mostra a Figura 16. Isso implica que, caso a temperatura da junção fria varie e a da junção quente, não, a tensão no terminais do termopar sofrerá uma variação indesejada. Então, idealmente, as medições devem ser realizadas onde a temperatura da junção fria estará sempre estável. Como isso não é possível de ser feito, já que é praticamente impossível de manter a temperatura da junção fria sempre estável, foi criada a tecnologia de compensação de junção fria. Este processo se caracteriza pela compensação das variações da temperatura da junção fria de forma eletrônica,

através de sensores de temperatura ou termistores. Sendo assim, caso a temperatura da junção quente não varie e a da junção fria sim, o valor não será alterado, devido à compensação de junção fria.(JACKSON, 2019).

No caso do MAX6675, esta compensação é realizada utilizando um diodo sensível a temperatura. Após isso, a circuitaria interna do dispositivo passa a tensão do diodo (representando a temperatura ambiente) e a do termopar (representando a temperatura medida menos a temperatura ambiente) para o ADC, que este calculará a temperatura da junção quente do termopar através de uma função de conversão, representada pela Equação 2.35 (MAXIM INTEGRATED, 2014).

$$V_{out} = (41\mu V/°C)5(T_R - T_{AMB}) \tag{2.35}$$

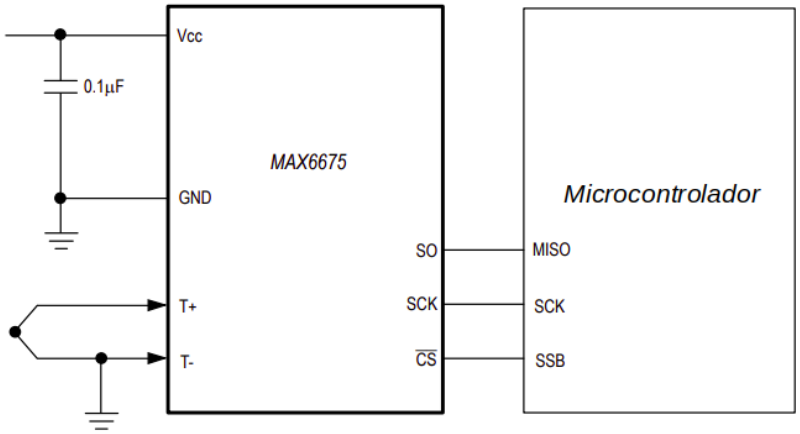
Onde:

V_{out} : Tensão de saída do termopar (μV)
T_R : Temperatura da junção quente do termopar (°C)
T_{AMB} : Temperatura ambiente (°C)

Em relação à digitalização, o ADC do MAX6675 adiciona a medida da temperatura de junção realizada pelo diodo com a tensão do termopar amplificada e efetua a leitura, resultando em um valor de 12 bits ponto fixo, sendo 10 para representação do valor inteiro e 2 para o valor decimal. Desta forma, com a precisão de 0,25 °C, sabe-se que a temperatura mínima é 0 °C e a máxima é 1023,75 °C (MAXIM INTEGRATED, 2014).

Outra importante característica do CI MAX6675 é o fornecimento de uma interface serial. Com o protocolo Interface Periférica Serial (*Serial Peripheral Interface* - SPI), um microcontrolador poderá se comunicar com o CI, captando os valores de temperatura obtidos através da leitura da tensão do termopar (MAXIM INTEGRATED, 2014).

Figura 17 – Esquemático do circuito de aplicação do MAX6675



Fonte: Adaptado de Maxim Integrated (2014).

Por realizar todo o tratamento do sinal analógico, possuir a compensação de junção fria e fornecer uma interface SPI para disponibilizar seu dados, este componente foi escolhido para ser integrado no conjunto de leitura da saída do sistema.

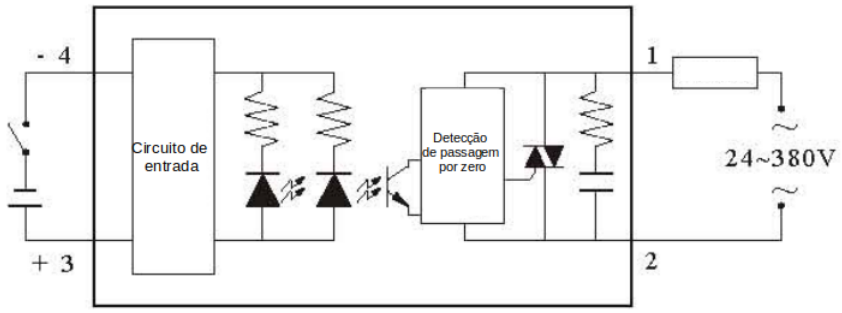
2.12 Relé de Estado Sólido

O Relé de Estado Sólido (*Solid State Relay* - SSR) é um dispositivo responsável de realizar o chaveamento de cargas através da comutação de uma chave eletrônica, diferente do relé mecânico, que utiliza uma bobina magnetizada para realizar o contato.

O funcionamento básico de um relé de estado sólido consiste na ativação de uma chave eletrônica através de seu fechamento, e, consequentemente, alimentando a carga, onde a tensão, normalmente, pode variar de 24 V até 380 V.

Outra característica do SSR é a detecção por passagem por zero (*zero-cross*), ou seja, a carga só será alimentada no momento em que a tensão alternada passar por 0 V (GUERRA, 2006). Assim, evitará possíveis danos à chave estática, esta podendo ser um Transistor de Efeito de Campo Metal-Óxido (*Metal Oxide Semiconductor Field Effect Transistor* - MOSFET) ou TRIAC, dependerá da topologia do fabricante.

Figura 18 – Esquemático do circuito do relé de estado sólido



Fonte: Adaptado de FOTEK (20–?).

O modelo escolhido para este trabalho é o SSR-40 DA. O primeiro motivo da escolha são as suas especificações, suportando o valor da tensão da rede (220 V), corrente (aproximadamente 6 A) e a tensão de acionamento (3 V). O segundo é a sua detecção de passagem por zero já estar implementada no relé, ou seja, o seu controle poderá ser realizado com apenas com o PWM.

2.13 ESP32

O microcontrolador ESP32 tem como grande característica fornecer hardware e software para o desenvolvimento de aplicações que utilizam a Internet das Coisas (*Internet of Things* - IoT). Desta forma, possui interface para Wi-Fi 2,4 GHz e diversos periféricos, como os protocolos de comunicação SPI, Circuito Inter-integrado (*Inter-integrated Circuit* - I2C) e Rede de Área do Controlador (*Controlle Area Network* - CAN). Com isso, ele poderá ser integrado com o CI MAX6675 para leitura das temperaturas através do protocolo SPI e, além disso, comunicar-se via Wi-Fi. Isso proporcionará o monitoramento e controle da temperatura desejada remotamente (ESPRESSIF SYSTEMS, 2020).

O fabricante Espressif Systems fornece um *framework* para desenvolvimento de software embarcado denominado Estrutura Espressif de desenvolvimento IoT (*Espressif IoT Development Framework* - ESP-IDF), que consiste em um conjunto de bibliotecas em linguagem C, onde o desenvolvedor terá acesso aos recursos de hardware, podendo desenvolver a sua aplicação embarcada (ESPRESSIF SYSTEMS, 2020).

Alguns fatores foram decisivos para a escolha deste microcontrolador. O primeiro deles é a sua popularidade, desta forma, pode ser encontrado diversas bibliotecas e exemplos de aplicações na internet, além do amplo suporte pela comunidade. Outro fator é a sua interface com Wi-Fi. Como o componente já possui uma antena interna para o uso desta tecnologia, a implementação do Wi-Fi no microcontrolador não

necessita da compra de componentes extras. Por fim, o seu custo benefício também teve grande participação na sua escolha. Por possuir duas unidades de processamento, frequência do *clock* até 240 MHz e diversos periféricos, o ESP32 se torna uma opção viável.

2.14 Hypertext Transfer Protocol (HTTP)

Utilizado para transmissão de documentos hipermídia, como HTML, o Protocolo de Transferência de Hipertexto (*Hypertext Transfer Protocol* - HTTP) encontra-se na camada de aplicação do modelo OSI. Foi desenvolvido para comunicações entre navegadores web e servidores da web, porém possui aplicações com outras finalidades. Seguindo o modelo clássico de cliente-servidor, onde o cliente é responsável por abrir uma conexão para fazer uma requisição ao servidor e esperar pelo retorno de uma resposta, o HTTP é um protocolo sem estado, ou seja, o servidor não mantém nenhum dado entre duas requisições (MOZILLA, 2020).

A utilização deste protocolo se caracteriza pela implementação de um servidor, disponibilizando os dados relacionados à temperatura e a definição da temperatura desejada. Com isso, o sistema poderá ter seus parâmetros controlados e lidos por aplicações externas, através de requisições HTTP.

2.15 Qt Framework

Como será visto na Sessão 4.4 do Capítulo 4, será desenvolvida uma interface gráfica para o usuário controlar e monitorar o forno elétrico. Desta forma, utilizou-se o *framework* Qt para desenvolvê-la.

Esta ferramenta se trata de um *framework* para desenvolvimento de aplicações de desktop, sistemas embarcados e sistemas móveis. Apesar de ser desenvolvido em linguagem C++, o Qt oferece extensões para outras linguagens, como Java e Python, sendo esta última a que será utilizada para o desenvolvimento do software deste trabalho (QT, 2019).

Além disso, o Qt possui uma ferramenta iterativa gráfica denominada Qt Designer, que permite desenvolver interfaces gráficas de forma simples e, após isso, exportar o código resultante (QT, 2019).

A escolha desta ferramenta se deu pela facilidade em criar Interface Gráfica de Usuário (*Graphical User Interface* - GUI) através do Qt Designer e haver a sua extensão para linguagem Python, que foi a escolhida para a elaboração do software.

2.16 Relação dos Dispositivos e Tecnologias Adotados

Por fim, serão apresentados os componentes e tecnologias que foram utilizados neste projeto e suas respectivas funções, como mostra a Tabela 1.

Tabela 1 – Relação dos componentes e dispositivos adotados e sua respectiva função

Componente ou tecnologia	Função
Forno elétrico	Dispositivo central deste trabalho, pois ele será responsável por fornecer o ambiente e, também, por gerar a troca de energia térmica com o sistema eletrônico que será testado
Termopar Tipo K	Será utilizado para medir a temperatura no interior do forno para realizar a ação de controle.
MAX6675	CI responsável por interpretar as leituras analógicas de tensão do termopar que representam a temperatura. Além disso, fornecerá uma interface SPI para o microcontrolador adquirir seus dados.
ESP32	Microcontrolador que será utilizado para realizar a ação de controle e fornecer seus dados para aplicações externas.
Servidor HTTP	O microcontrolador criará um servidor HTTP para que aplicações externas possam obter seus dados e poder definir a temperatura desejada no interior do forno.
Relé de estado sólido SSR-40 DA	Este dispositivo terá como objetivo realizar a ação de controle do sistema, permitindo e bloqueando a passagem de corrente para a resistência no interior do forno.
Qt *framework*	Com o auxílio da ferramenta Qt Designer, esta tecnologia foi adotada para a criação das interfaces gráficas que o software de controle e monitoramento contemplará.

Fonte: Elaboração própria (2020).

3 METODOLOGIA

Este trabalho tem como objetivo desenvolver um sistema para testar produtos eletrônicos sob a temperatura máxima especificada, sendo aplicado para resolver uma necessidade da validação da robustez do projeto. Com isso, esta pesquisa pode ser classificada como aplicada, descritiva com abordagem qualitativa e bibliográfica.

A pesquisa aplicada tem como característica o objetivo prático, ou seja, seu resultado será aplicado ou utilizado para resolver um interesse específico (MARCONI; LAKATOS, 2017). Baseando-se nisso, esta monografia utilizou a pesquisa aplicada para desenvolver uma ambiente de estresse térmico a partir de um forno elétrico.

Além da pesquisa aplicada, outro tipo de pesquisa utilizada foi a abordagem qualitativa, sendo que esta tem como alvo a análise da compreensão de um tema, não somente a análise numérica (GERHARDT; SILVEIRA, 2009).

Já em relação à pesquisa descritiva, Gil (2002) descreve que este modelo de pesquisa baseia-se em retratar as características de algum fenômeno ou a correlação entre variáveis. Neste trabalho foram identificados os blocos do sistema, assim, pode-se relacionar este processo à este tipo de pesquisa.

Por último, foi realizada a pesquisa bibliográfica, ou seja, aquela que se aproveita de artigos científicos e livros publicados (GIL, 2002). Neste Trabalho de Conclusão de Curso, utilizou-se artigos científicos, livros, monografias e materiais do meio eletrônico para construir a base de dados e a fundamentação teórica para realizar seus objetivos.

O desenvolvimento deste trabalho foi iniciado pela definição da função de transferência do forno elétrico, a qual se deu por quatro partes. A primeira: a aplicação de um sinal quadrado com *duty-cycle* na entrada do forno por meio do relé de estado sólido e adquisição das medições dos termopares periodicamente; a segunda: a criação de um *script* em linguagem Python para exportar estes dados para um arquivo contendo o tempo em segundos que as temperaturas foram adquiridas, a temperatura atual e a temperatura desejada; a terceira: a criação de um algoritmo a fim de realizar os cálculos e definir o valor dos parâmetros que compõem a função de transferência de um sistema de primeira ordem com atraso; e, por fim, verificar a similaridade entre a resposta do forno com os valores adquiridos e a resposta calculada através do modelo matemático elaborado.

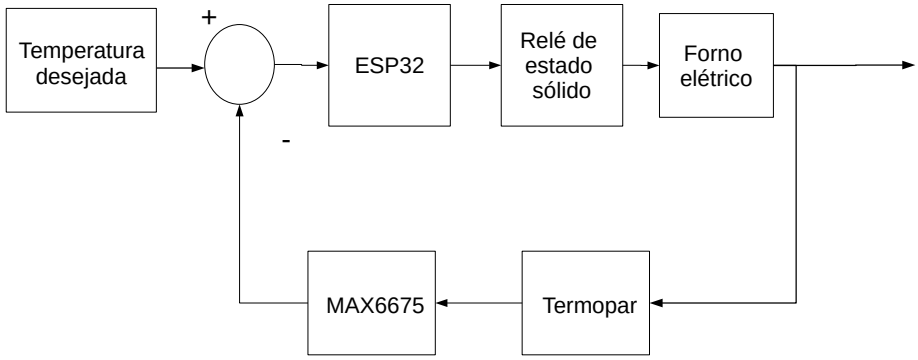
Em seguida, foi projetado o controlador responsável por controlar a temperatura do forno. Como os parâmetros da função de transferência já foram definidos previamente, o projeto do controlador utilizou a técnica de Ziegler-Nichols para encontrar os valores das constantes que o compõem. Após isso, implementou-se controlador PI no microcontrolador.

Um servidor HTTP, responsável por realizar a comunicação entre o controlador e aplicações externas, foi também embarcado no microcontrolador ESP32. Sua criação utilizou-se de bibliotecas do próprio ESP-IDF. Sua funcionalidade foi testada com a utilização da ferramenta *Postman*, que criou requisições HTTP e suas respostas foram verificadas.

Um dos sistemas que compõem este projeto é o software de monitoramento e controle do forno elétrico. Desenvolvido em linguagem de programação Python, ele terá três recursos principais: interface gráfica intuitiva, automatização dos testes com o forno elétrico e exportação dos dados sobre os processos que ocorreram. Para a elaboração da interface gráfica foi adotado o *framework* Qt. Ainda no tocante à interface gráfica, utilizou-se a ferramenta Qt Designer para o posicionamento dos componentes que a formam, como botões e gráficos, e a geração do código em Python. Em seguida, foram criadas as classes que são responsáveis pelas funcionalidades do código, conforme será visto na Sessão 4.4.

Por fim, todas as partes do sistema foram integradas. Sua funcionalidade foi aprovada analisando se a temperatura do forno estabeleceu-se de acordo com a temperatura desejada durante o período previamente estabelecido. Além disso, os dados exportados após os testes devem estar de acordo com os eventos que aconteceram, comprovando o controle de temperatura, a comunicação entre cliente-servidor e a automação das rotinas criadas pelo usuário. O diagrama de blocos da planta de controle resultante deste trabalho é representado pela Figura 19.

Figura 19 – Planta de controle com os componentes utilizados neste trabalho



Fonte: Elaboração própria (2021).

4 PROJETO E RESULTADOS

Este Capítulo será responsável por abordar o projeto dos componentes deste trabalho e os resultados obtidos. Primeiro, será descrito o processo de definição da função de transferência do forno elétrico. Após isso, o desenvolvimento do controlador PI e a sua implementação no microcontrolador ESP32 será abordada. Já a próxima sessão, descreverá a criação do servidor HTTP responsável pela interface entre o software de monitoramento e controle e o controlador. Em seguida, será retratado a elaboração do software, comentado do Capítulo 1, e suas funcionalidades. Por fim, será demonstrado a integração de todas as partes do projeto desenvolvida e os resultados obtidos.

Durante a elaboração do sistema, é possível notar que diversas técnicas abordadas no Capítulo 2 serão utilizadas, principalmente na definição da função de transferência do forno elétrico e no projeto do controlador PI.

4.1 Modelagem Matemática do Forno Elétrico

O início do desenvolvimento do projeto se dá pela modelagem matemática do forno elétrico, representado pela Figura 20, por uma função de transferência de primeira ordem com atraso. Ainda em relação ao forno, este possui potência máxima de 1500 W.

Este processo divide-se em três partes, sendo a primeira responsável pela aquisição das temperaturas, a segunda o processamento destas amostras para a definição dos parâmetros da função de transferência e, por fim, a simulação da função de transferência para comprovar a similaridade entre o modelo real do forno elétrico e o modelo obtido matematicamente. Desta forma, os parâmetros obtidos nesta sessão serão utilizados para calcular o ganho dos componentes do controlador PI, conforme será descrito na Sessão 4.2.

Figura 20 – Forno elétrico escolhido para o projeto.

Fonte: Elaboração própria (2021).

Para obter a temperatura do forno, foi utilizado o termopar com o CI MAX6675, sendo que este segundo se comunicará com o ESP32 através do protocolo SPI. O termopar utilizado foi posicionado na parte superior do forno, conforme mostra a 21.

Figura 21 – Termopar utilizado (esquerda) posicionado na parte superior do forno junto com termopar reserva (direita).

Fonte: Elaboração própria (2021).

Além disso, o ESP32 será responsável por controlar a razão cíclica do PWM na entrada do relé de estado sólido. A Figura 22 representa o posicionamento SSR na lateral forno.

Figura 22 – Relé de estado sólido utilizado.

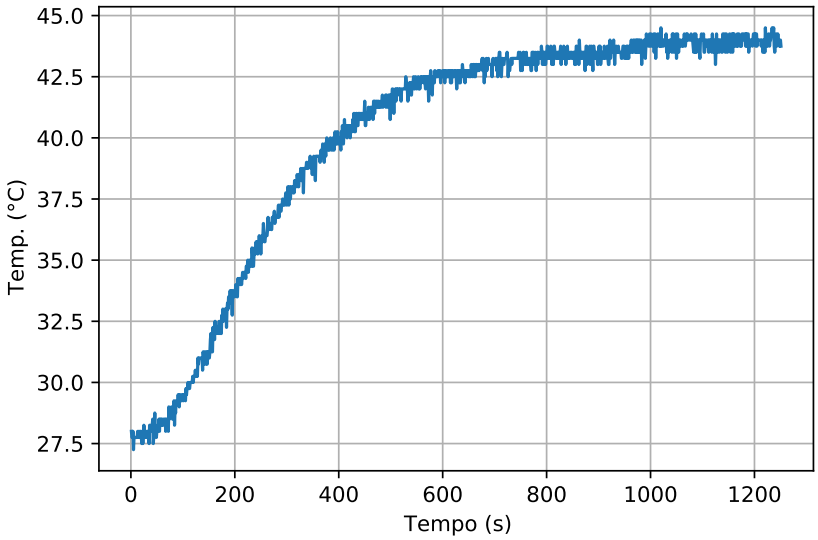
Fonte: Elaboração própria (2021).

Como sistemas térmicos costumam apresentar uma resposta muito lenta, foi arbitrada uma frequência de 1 Hz para o sinal PWM de entrada do sistema. No ESP32, o sinal de PWM foi implementado através de uma biblioteca fornecida pelo ESP-IDF (ESPRESSIF SYSTEMS, 2021).

Já para visualização dos dados, foi criado um programa para exportar as amostras de temperatura para um arquivo, de forma que, posteriormente, estes dados possam ser lidos e o gráfico da temperatura em função do tempo gerado.

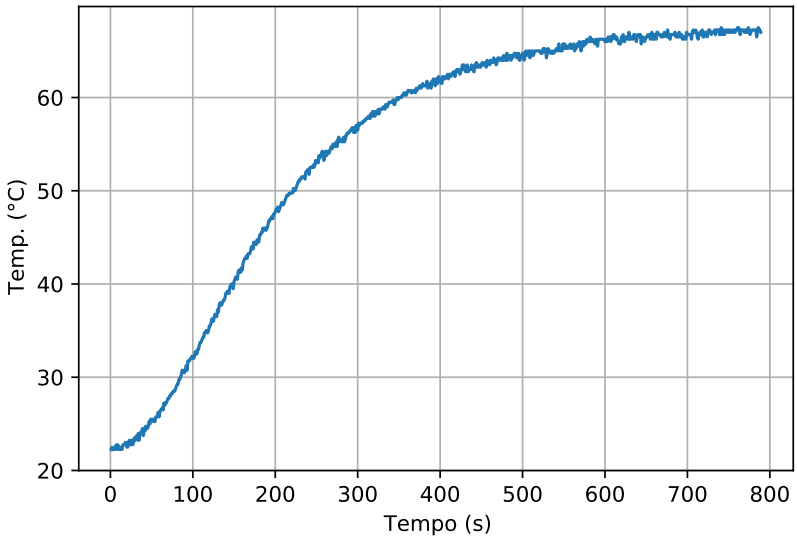
Como o forno elétrico trata-se de um sistema lento, a reposta ao degrau com amplitudes variadas pode ser emulada com um sinal de PWM com diferentes razões cíclicas. Assim, inicialmente foram aplicados 4 sinais de PWM com *duty cycles* destintos: 10 %, 36 %, 50 % e 100 %, conforme mostram as Figuras 23:, 24, 25 e 26:

Figura 23 – Resposta ao degrau do forno elétrico com razão cíclica de 10% .



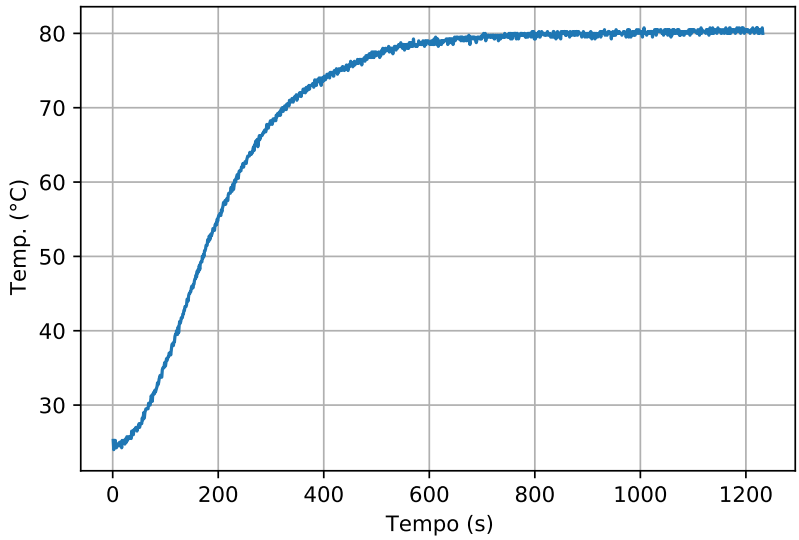
Fonte: Elaboração própria (2021).

Figura 24 – Resposta ao degrau do forno elétrico com razão cíclica de 36% .



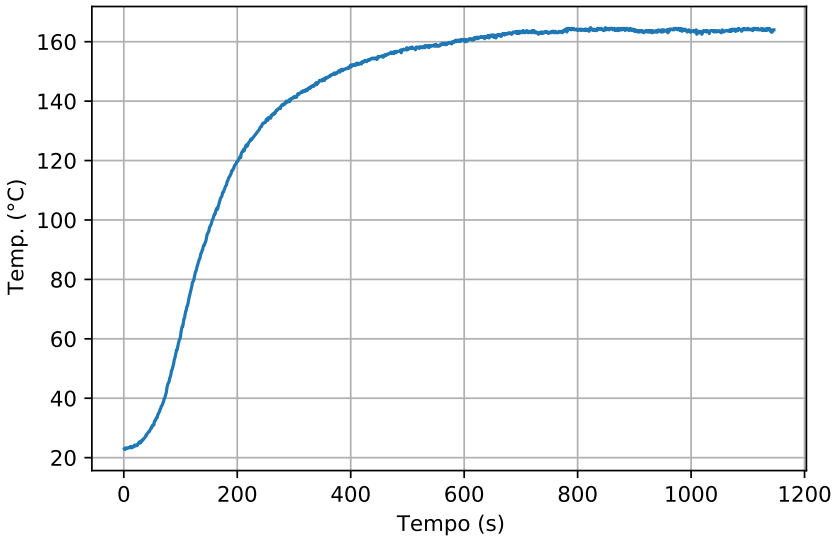
Fonte: Elaboração própria (2021).

Figura 25 – Resposta ao degrau do forno elétrico com razão cíclica de 50% .



Fonte: Elaboração própria (2021).

Figura 26 – Resposta ao degrau do forno elétrico com razão cíclica de 100% .

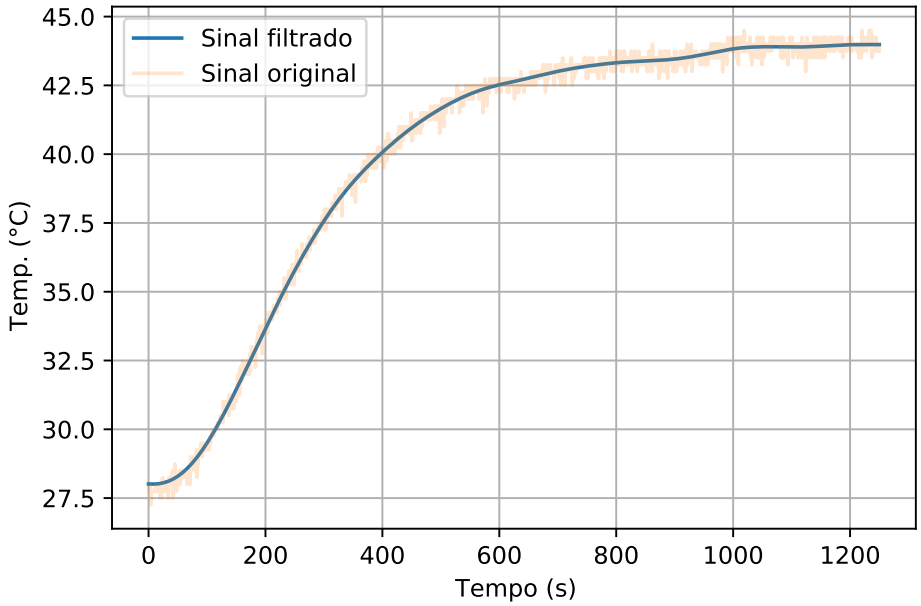


Fonte: Elaboração própria (2021).

Analisando as imagens acima, percebe-se que a temperatura de regime permanente da Figura 23 é a mais próxima do que será usado para os testes com os produtos eletrônicos.

Com a resposta escolhida para o cálculo dos parâmetros da função de transferência, aplicou-se um filtro digital do tipo passa baixas no sinal para melhorar a sua visualização, conforme mostra a Figura 27.

Figura 27 – Resposta ao degrau do forno elétrico com razão cíclica de 10 % filtrada.



Fonte: Elaboração própria (2021).

Em seguida, foram calculadas as áreas A_0 e A_1, e o ganho k, descritas na Sessão 2.5, sendo elas descritas novamente neste capítulo pelas Equações 4.2, 4.3 e 4.1. Além disso, a função de transferência do sistema de primeira ordem com atraso é representada pela Equação 4.4.

$$k = \frac{c_\infty}{A} \tag{4.1}$$

$$A_0 = \int_0^\infty [kA - c(t)]dt = kA(T + \alpha) \tag{4.2}$$

$$A_1 = \int_0^{\alpha+T} c(t)dt = \int_0^{\alpha+T} kA(1 - e^{-t/T})dt = kATe^{-1} \rightarrow T = \frac{A_1 e}{kA} \tag{4.3}$$

$$G(s) = \frac{C(s)}{R(s)} = \frac{k}{Ts+1}e^{-\alpha s} \tag{4.4}$$

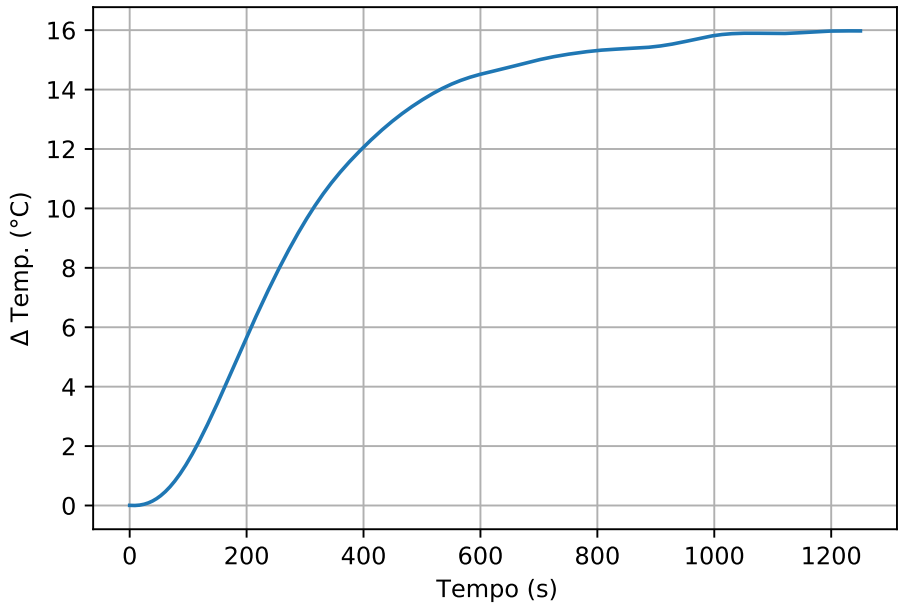
Como a resposta do sistema amostrada é um sinal digital obtido periodicamente a cada um segundo, utilizou-se o somatório para o cálculo aproximado da integral do sinal das equações 4.2 e 4.3. Com isso, elas podem ser reescritas pelas equações 4.5 e 4.6, respectivamente.

$$A_0 = \int_0^\infty [kA - c(t)]dt \approx \sum_{k=0}^{n} \frac{kA - c(k)}{1/f_s} \tag{4.5}$$

$$A_1 = \int_0^{\alpha+T} c(t)dt \approx \sum_{k=0}^{\alpha+T} \frac{c(k)}{1/f_s} \tag{4.6}$$

As equações do ganho k e das áreas A_0 e A_1 contemplam respostas ao degrau com condição inicial igual a zero. Desta forma, subtraiu-se a temperatura de cada amostra pelo valor da temperatura inicial, obtendo a resposta ao degrau representada pela Figura 28

Figura 28 – Resposta ao degrau do forno elétrico com *duty cycle* de 10 % filtrada com condição inicial zero.



Fonte: Elaboração própria (2021).

Para o cálculo dos parâmetros α, T e k da função de transferência, iniciou-se com o ganho k, já pode ser obtido diretamente com a Equação 4.1. Porém, antes de definir o valor de k, a amplitude do sinal de entrada, A, deve ser calculada com a equação da tensão média do sinal de PWM. Sabendo que a tensão eficaz da rede elétrica é 220 V, o período do PWM é um segundo e o *duty cycle* é 10 % do período, o valor da tensão eficaz na entrada do forno é calculado abaixo:

$$V_{eficaz} = A = \frac{T_{on}}{T} V = 22\,V \tag{4.7}$$

Para determinar o valor de temperatura no regime permanente, utilizou-se uma função da *max()* do Python para se obter o maior valor sinal da resposta ao degrau com condição inicial zero, representado pela Figura 28. Com isso, o valor retornado da temperatura de regime permanente são aproximadamente 16 ℃.

Com os parâmetros da Equação 4.1 estabelecidos, o valor de k pôde ser determinado, conforme os cálculos abaixo:

$$k = \frac{c_\infty}{A} \rightarrow k = 0,29 \tag{4.8}$$

Em seguida, obteve-se a área A_0 através da Equação 4.5.

$$A_0 \approx \sum_{k=0}^{n} \frac{kA - c(k)}{1/f_s} \tag{4.9}$$

$$A_0 \approx 4.933,79 \tag{4.10}$$

Utilizando o valor de A_0, a soma $T + \alpha$ foi definida, possibilitando o cálculo da Equação 4.6.

$$kA(T + \alpha) = 4.933,79 \tag{4.11}$$

$$T + \alpha = 308,84 \approx 309 \tag{4.12}$$

Após ter definido o limite superior do somatório da Equação 4.6, ou seja, o valor de $T + \alpha$, a área A_1 foi obtida, conforme os cálculos abaixo:

$$A_1 \approx \sum_{k=0}^{\alpha+T} \frac{c(k)}{1/f_s} \tag{4.13}$$

$$A_1 = 1.231,83 \tag{4.14}$$

Seguidamente, o valor de T foi definido:

$$A_1 = kATe^{-1} \rightarrow T = \frac{A_1 e}{kA} \tag{4.15}$$

$$T = 209,60 \tag{4.16}$$

Por fim, o último parâmetro da função de transferência, o tempo de atraso α, foi determinado utilizando a Equação 4.2:

$$A_0 = kA(T + \alpha) \rightarrow \alpha = \frac{A_0}{kA} - T \tag{4.17}$$

$$\alpha = 99,23 \tag{4.18}$$

Com isso, finalizou-se o cálculo dos parâmetros da função de transferência. A Equação 4.19 mostra sua representação.

$$G(s) = \frac{0,29}{209,6s + 1} e^{-99,23s} \tag{4.19}$$

Observando a Equação 4.19, nota-se que há um componente irracional, que é o atraso do sistema, representado pelo termo $e^{-99,23s}$. Desta forma, utilizou-se as

Aproximações de Padè para torná-lo racional. Através da função *Pade*() da biblioteca Python-Control do Python, criou-se uma função de transferência com o atraso desejado (PYTHON-CONTROL, 2019a).

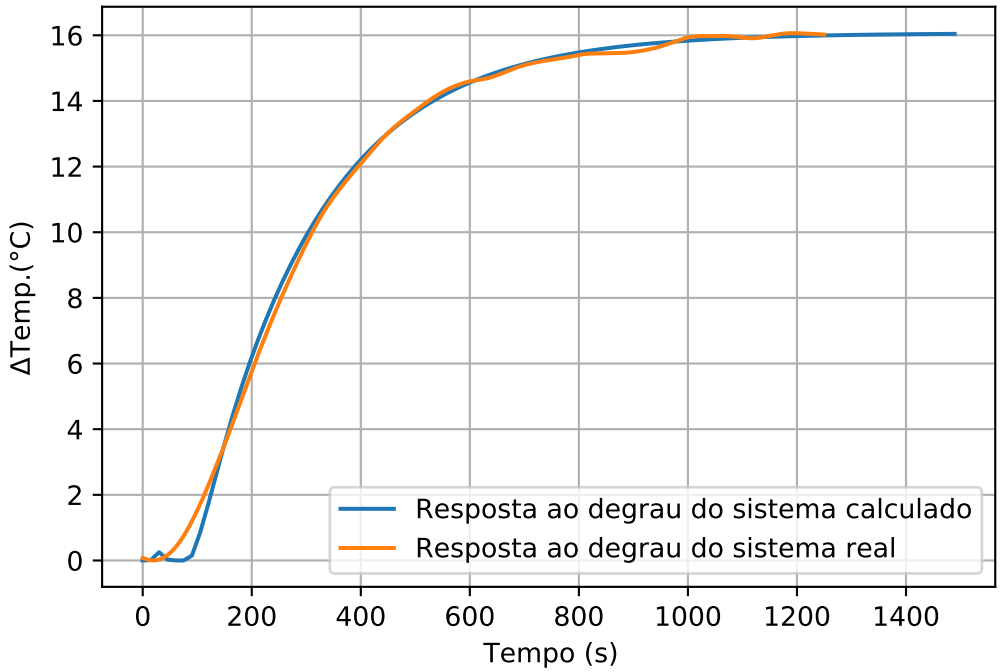
$$G(s) = \frac{s^2 - 0,06046s + 0,001219}{s^2 + 0,06046s + 0,001219} \tag{4.20}$$

Após isso, empregou-se a classe *TransferFunction* para obter a função de transferência da parte racional da Equação 4.19, e, seguidamente, a função *series* para obter a função de transferência racional do termo de primeira ordem com o termo que representa o atraso, formando assim, a função de transferência descrita pela Equação 4.21 (PYTHON-CONTROL, 2019c).

$$G(s) = \frac{0,29s^2 - 0,01756s + 0,0003539}{209,6s^3 + 13,67s^2 + 0,3159s + 0,001219} \tag{4.21}$$

Já para simulação da resposta ao degrau, foi implementada a função *step_response* (PYTHON-CONTROL, 2019b). Com o resultado, elaborou-se um gráfico comparando a resposta ao degrau obtida através do modelo real do forno com a resposta ao degrau do modelo representado pela Equação 4.21. Esta comparação está caracterizada pela Figura 29.

Figura 29 – Comparação da resposta ao degrau entre o sistema real e o sistema obtido matematicamente



Fonte: Elaboração própria (2021).

A Figura 29 demonstra a grande similaridade entre o sistema real do forno elétrico e o modelo calculado através das equações descritas nesta sessão. Com

isso, confirma-se o valor dos parâmetros da função de transferência, ou seja, eles representam o comportamento do modelo real. Estes parâmetros serão utilizados para o projeto do controlador PI, conforme mostra a Sessão 4.2.

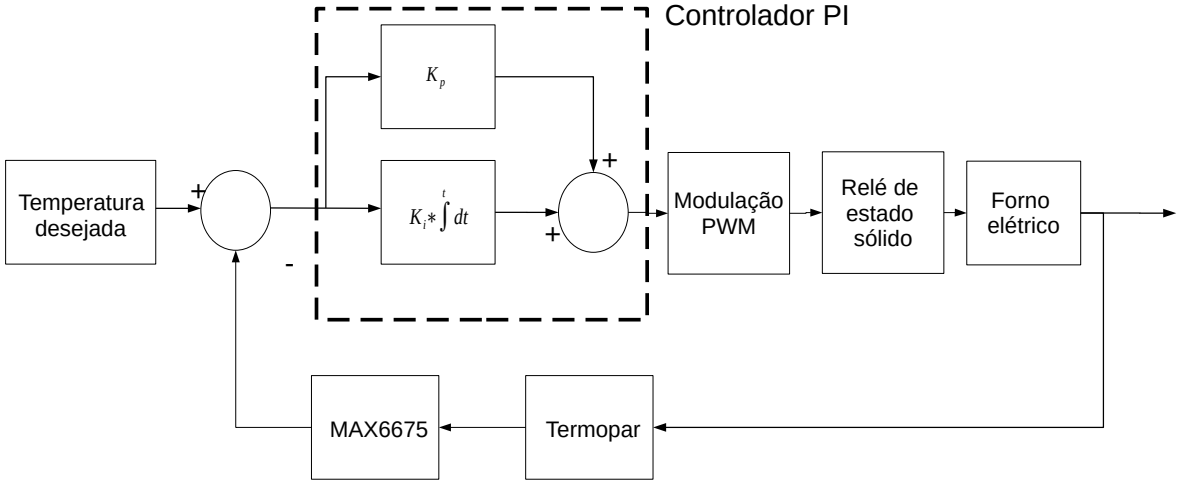
4.2 Projeto do Controlador PI

Dentre os componentes que compõe a aplicação desenvolvida neste trabalho, o controlador PI será responsável por manter a temperatura no interior do forno elétrico de acordo com o valor desejado.

Como mencionando na Seção 2.9.5, os motivos para a escolha do controlador PI para controlar o sistema são: a função de transferência do forno elétrico é representada por uma de primeira ordem com atraso não dominante e, em relação às especificações do projeto, não há um tempo de acomodação determinado, a únicas características a serem levadas em conta é não haver sobressinal e o erro ser nulo em regime permanente.

No tocante à planta de controle, pode-se afirmar que o atuador do sistema será o relé de estado sólido, já utilizado na sessão anterior para a aplicação do sinal de PWM, o sensor para obter o valor a saída do sistema será o termopar com seu sinal condicionado pelo CI MAX6675, realimentado a malha de controle. O bloco de referência é composto pela temperatura desejada configurada pelo servidor HTTP, com projeto descrito na Sessão 4.3. Desta forma, o diagrama da planta de controle está representado pela Figura 30.

Figura 30 – Diagrama da planta de controle do forno elétrico.



Fonte: Elaboração própria (2021).

Os ganhos dos termos que compõem o controlador PI, ou seja, o ganho proporcional e o ganho integral, foram calculados através das equações de Zeigler-Nichols

mencionadas na Sessão 2.9.5, sendo elas reescritas novamente pelas equações 4.22 e 4.23.

$$K_p = 0.9\frac{T}{\alpha} \tag{4.22}$$

$$K_i = \frac{0,27T}{\alpha^2} \tag{4.23}$$

A Tabela 2 relaciona os parâmetros da equação de transferência do segundo e o ganho dos componentes do controlador com seus respectivos valores.

Tabela 2 – Relação entre os parâmetros da função de transferência e os termos do controlador com

Parâmetro	Valor
T	209,60
α	99,23
K_p	1,988
K_i	0,0062

Fonte: Elaboração própria (2021).

Após, os termos K_i e K_p terem sido definidos, o controlador foi implementado no ESP32. Para isso, foram utilizadas as Equações 4.24 e 4.25 para implementar o comportamento da ação proporcional e integral, e, logo em seguida, a soma do resultado delas, conforme demonstra a Equação 4.26.

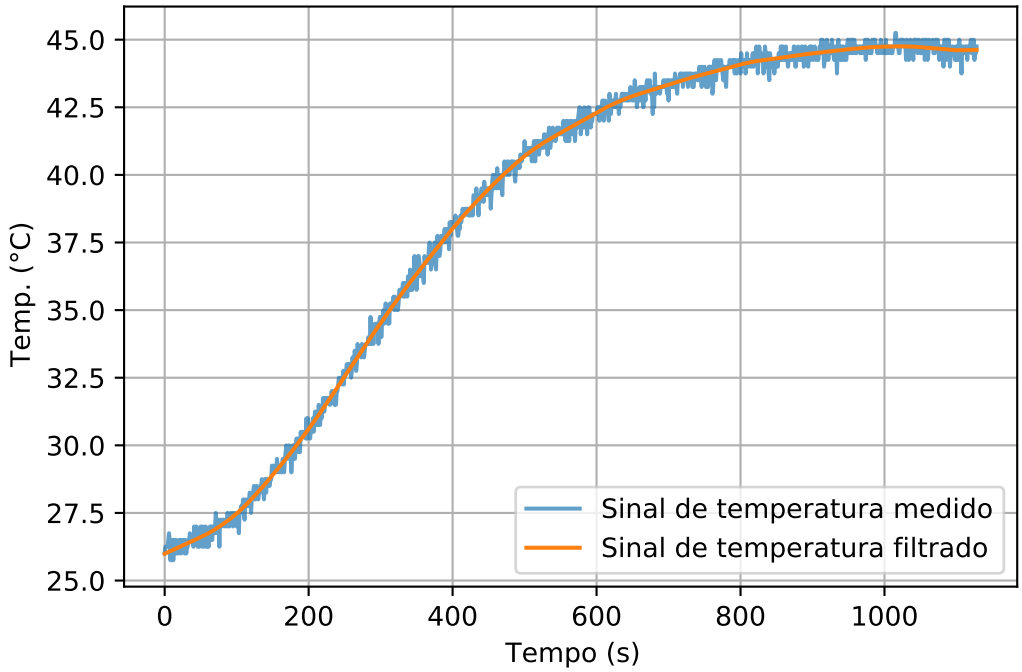
$$u_i(k) = T\sum_{k=0}^{n} e(k) \tag{4.24}$$

$$u_p(k) = k_p e(k) \tag{4.25}$$

$$u(k) = u_i(k) + u_p(k) = k_i T\sum_{k=0}^{n} e(k) + k_p e(k) \tag{4.26}$$

A programa em linguagem C da implementação do controlador PI no microcontrolador é descrito pelo Apêndice A. Para comprovar seu funcionamento, foi configurado uma temperatura desejada de 44 ℃ no interior do forno elétrico e analisado a saída do sistema, conforme mostra a Figura 31.

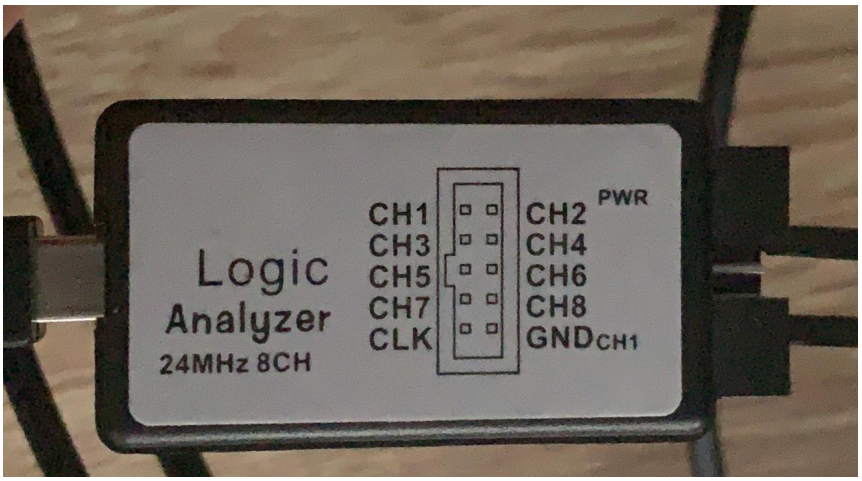
Figura 31 – Temperatura no forno em função do tempo para temperatura desejada de 44 ℃.



Fonte: Elaboração própria (2021).

Durante a tarefa do ESP32 de realizar a ação de controle, de acordo com o valor na saída do controlador, o *duty cycle* do sinal de PWM é alterado. As figuras 33 e 35 mostram o sinal de PWM nos terminais de comando do relé de estado sólido obtidos através do analisador lógico (Figura 32), sendo que a primeira foi obtida em um período de transição, ou seja, com sinal de erro maior. Já a segunda, foi capturada no período de estabilização, ou seja, quando a temperatura já atingiu o valor desejado, desta forma, só há a ação do componente integrador do controlador, já que o sinal de erro é praticamente zero:

Figura 32 – Analisador lógico utilizado para leitura do sinal de comando do ESP32.

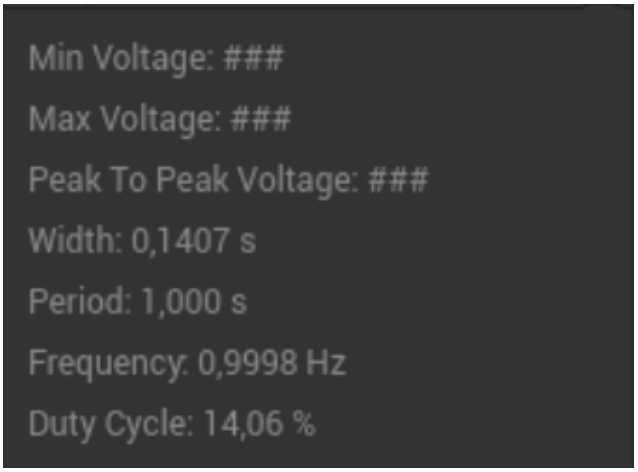


Fonte: Elaboração própria (2021).

Figura 33 – Sinal de PWM com *duty cycle* de 14 % na entrada do relé de estado sólido.

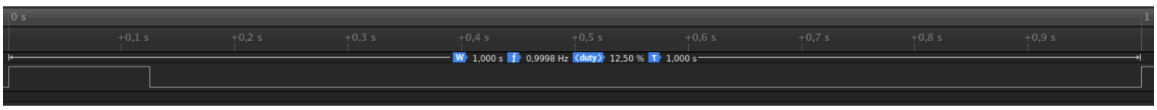
Fonte: Elaboração própria (2021).

Figura 34 – Informações do sinal PWM com *duty cycle* de 14 % na entrada do relé de estado sólido



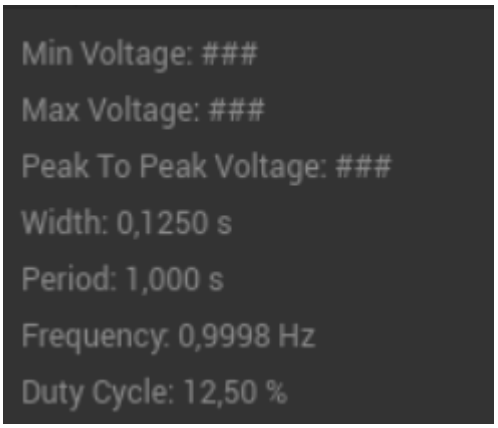
Fonte: Elaboração própria (2021).

Figura 35 – Sinal de PWM com *duty cycle* de 12,5 % na entrada do relé de estado sólido



Fonte: Elaboração própria (2021).

Figura 36 – Informações do sinal de PWM com *duty cycle* de 12,5 % na entrada do relé de estado sólido.



Fonte: Elaboração própria (2021).

Com isso, a Figura 31 mostra que o comportamento do forno se tornou satisfatório, uma vez que não há o erro em regime permanente, devido a ação do componente integrador do controlador PI. Além disso, não houve presença de sobressinal significativa, já que os valores que ultrapassam 44 ℃ tratam-se de ruídos do próprio termopar. Desta forma, concluiu-se o projeto do controlador PI.

4.3 Desenvolvimento do Servidor HTTP

Um dos componentes do sistema deste trabalho é o servidor HTTP, para que ocorra a troca de informações entre o controlador PI e o software de controle e monitoramento. Com isso, esta sessão detalhara a elaboração de uma interface de programação de aplicação (*Application Programming Interface*, API) servida pelo microcontrolador e o software atuando como cliente.

Antes do desenvolvimento do servidor, foram estabelecidos quais dados poderiam ser configurados ou adquiridos. Desta forma, estabeleceu-se que os valores de temperatura atual, desejada e do ambiente externo poderão ser adquiridos, enquanto somente o valor da temperatura desejada poderá ser configurada.

Sabendo os parâmetros que a API disponibilizará, em seguida foram definidos os *endpoints* para o cliente realizar as requisições e qual o tipo de requisição que poderá ser realizada. A Tabela 3 descreve as especificações estabelecidas para o desenvolvimento da API.

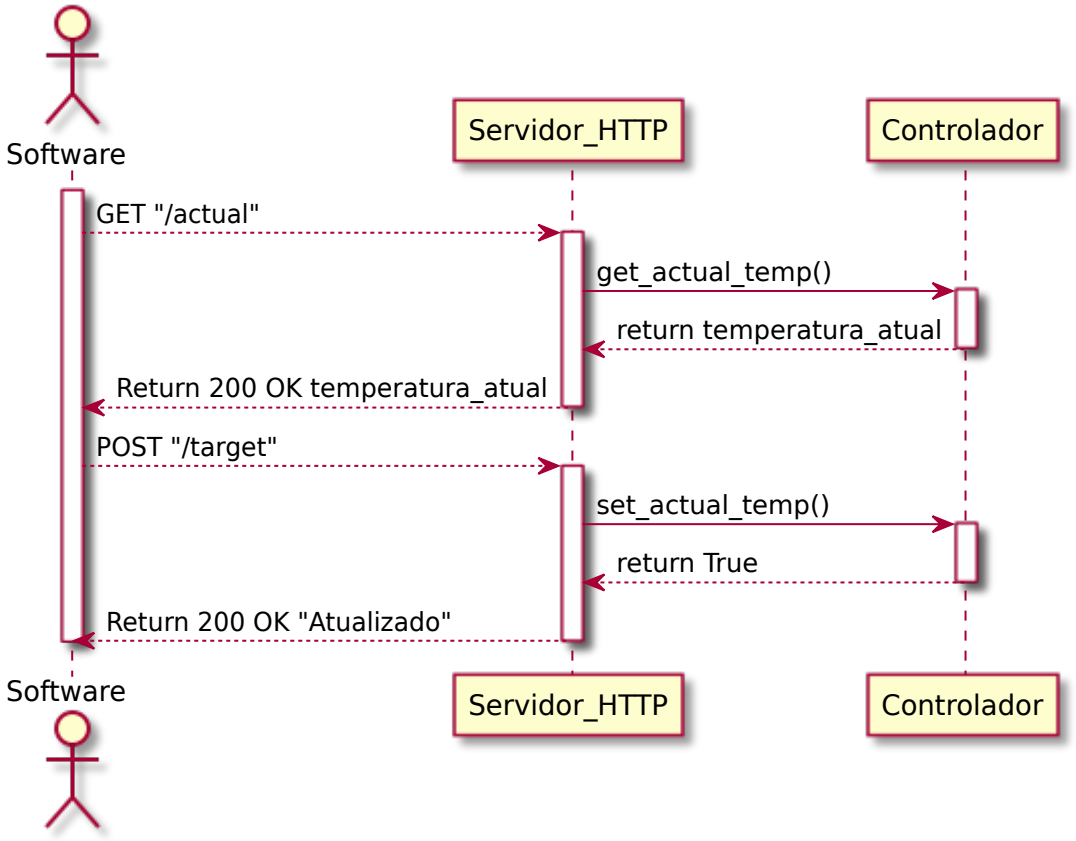
Tabela 3 – Relação entre os parâmetros fornecidos pela API com seus respectivos *endpoints* e tipos de requisição.

Parâmetro	*Endpoint*	Tipo de requisição
Temperatura desejada	/target	GET/POST
Temperatura atual	/actual	GET
Temperatura ambiente	/room	GET

Fonte: Elaboração própria (2021).

Com as especificações já estabelecidas, foi implementado o servidor no microcontrolador ESP32 utilizando o ESP-IDF, como mencionado no Capítulo 3. O código fonte elaborado está contido no Apêndice E. Ainda em relação ao desenvolvimento do servidor, ele não armazenará os dados trocados com aplicações externas, ou seja, os dados recebidos pelas requisições POST serão configurados no controlador através de métodos fornecidos por esta entidade. E para requisições GET, o servidor lerá os parâmetros do controlador também através de métodos disponibilizados pelo controlador. A Figura 37 representa o diagrama de sequência entre o software, controlador PI e o servidor HTTP.

Figura 37 – Diagrama de sequência entre o software, controlador PI e o servidor HTTP.

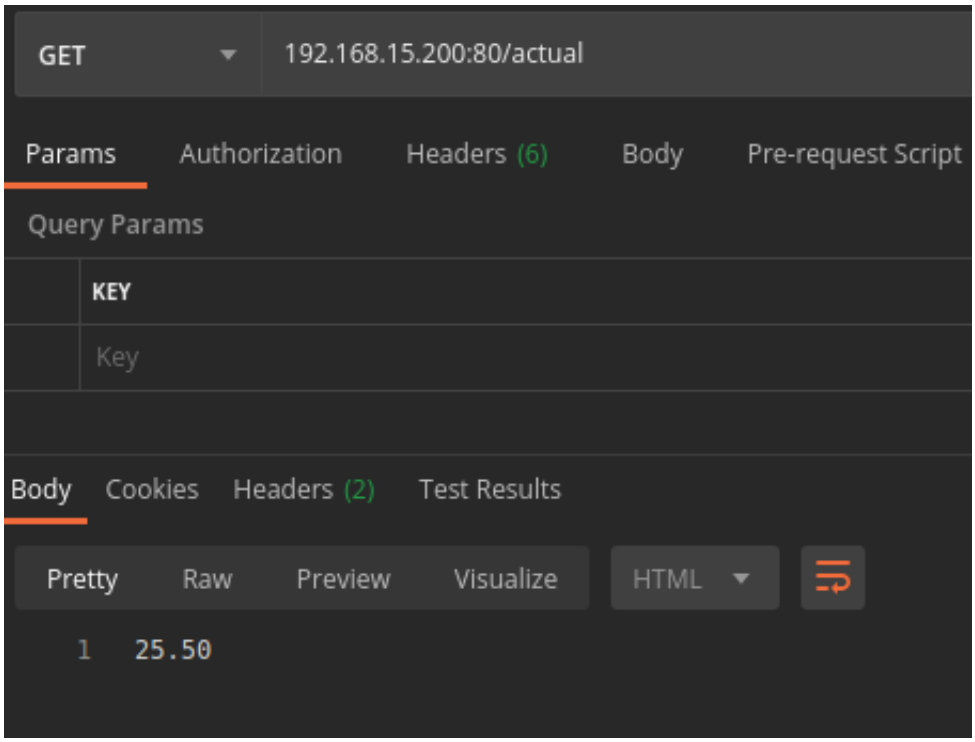


Fonte: Elaboração própria (2021).

Para verificar o comportamento do servidor, foi utilizada a ferramenta *Postman* para realizar as requisições para cada *endpoint*, analisando o seu código HTTP retornado e o valor recebido. Para a temperatura desejada, primeiramente, foi realizada uma requisição POST e, em seguida, uma requisição GET para verificar se o valor foi armazenado pelo controlador.

No primeiro caso, foi verificado o método GET comparando o valor da temperatura atual lido pelo microcontrolador e o valor obtido através da requisição GET no *Postman*. A mensagem impressa no terminal pelo microcontrolador é a seguinte: `main: Current: 25.50 Target 0.0`. Já a Figura 38, representa a resposta obtida no *Postman*.

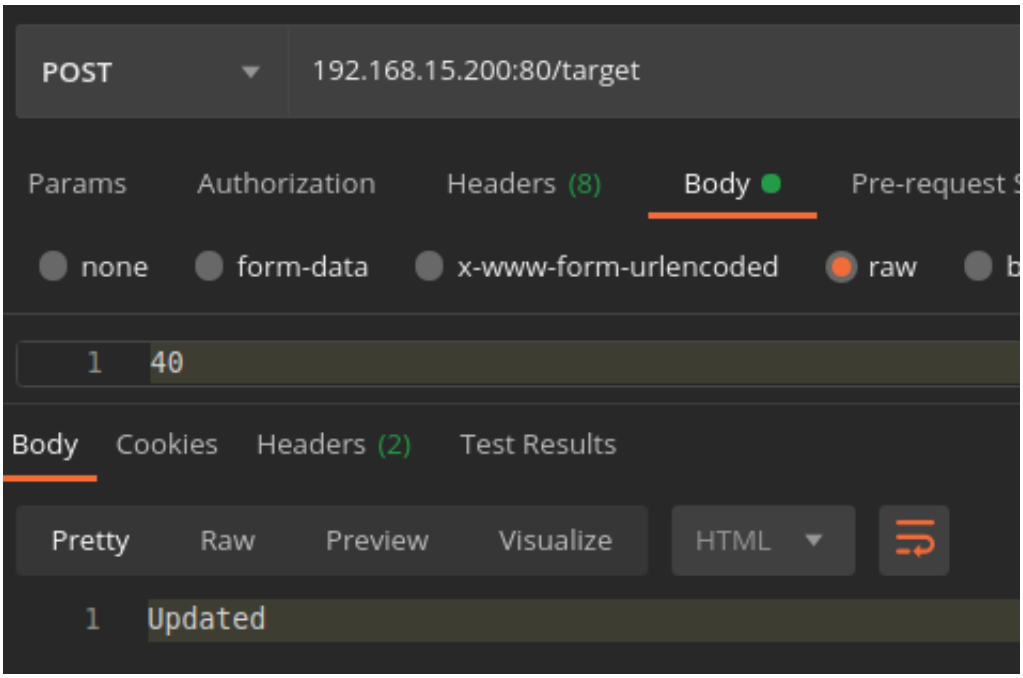
Figura 38 – Requisição GET realizada pela ferramenta Postman para o *endpoint* "/current".



Fonte: Adaptado de Postman (2021).

Para o segundo caso, foi realizada uma requisição POST para configurar o valor da temperatura deseja, conforme mostra a Figura 39.

Figura 39 – Requisição POST realizada pela ferramenta Postman para o *endpoint* "/target".

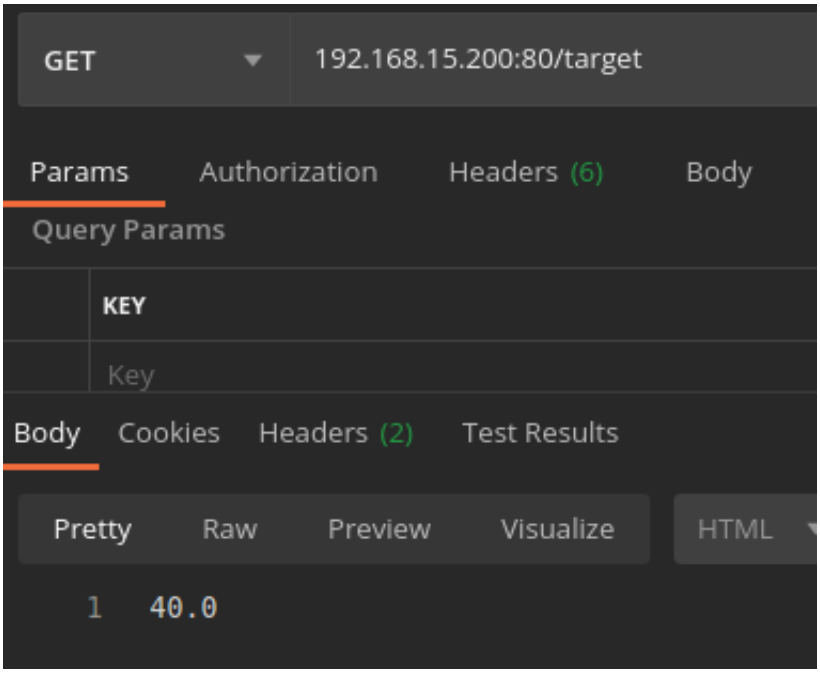


Fonte: Adaptado de Postman (2021).

Após isso, foi verificado o valor impresso no terminal pelo microcontrolador, que forneceu a mensagem: `Target 40.0`.

E, por último, realizado uma requisição GET para analisar a temperatura desejada retornada pelo servidor, como mostra a Figura 40.

Figura 40 – Requisição GET realizada pela ferramenta Postman para o *endpoint* "/target".



Fonte: Adaptado de Postman (2021).

Analisando as três figuras acima, pode-se afirmar que o desempenho do servidor atendeu os requisitos. Desta forma, ele intermediará a comunicação entre o software de controle e monitoramento e o controlador.

4.4 Desenvolvimento do Software de Controle e Monitoramento

O software de controle e monitoramento terá uma grande significância para a usabilidade deste projeto, pois ele deverá ser intuitivo e prático de usar. Desta forma, a interface gráfica desenvolvida deve disponibilizar os dados sobre a rotina em andamento, como período restante, temperatura atual, temperatura desejada e o gráfico da temperatura no interior do forno em função do tempo.

Além disso, o software deverá possibilitar a programação de rotinas de temperaturas automatizadas, ou seja, o usuário deverá poder escolher a temperatura desejada e o tempo de duração.

Por fim, o software exportará os dados dos processos de controle de temperatura que ocorreram, como o gráfico da temperatura em função do tempo e uma tabela com as principais informações. Desta forma, o projetista poderá conferir por quanto tempo seu circuito eletrônico funcionou e qual foi o seu desempenho sob determinada temperatura.

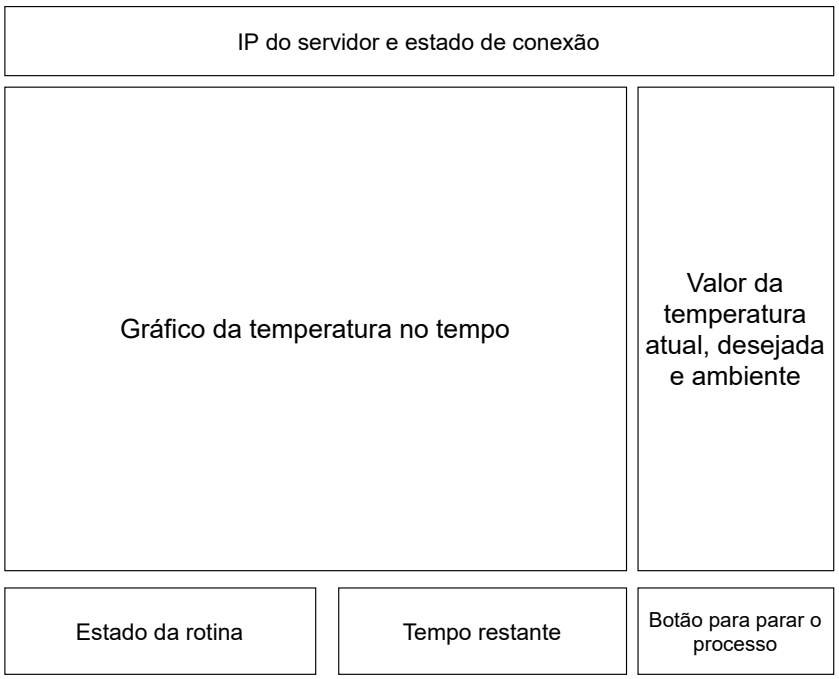
4.4.1 Interface Gráfica Principal

Como a interface gráfica da janela principal deve ser simples e intuitiva, ela terá que de disponibilizar os dados necessários sobre o processo que está ocorrendo. Para isso, foi elaborada uma lista de especificações com os principais dados que deverão ser mostrados para o usuário:

- Gráfico que disponibiliza a evolução da temperatura no interior do forno através do tempo.
- Tempo restante da rotina atual.
- Valores de temperatura desejada, temperatura atual e temperatura ambiente, caso deseja-se adicionar este dado futuramente.
- IP do servidor HTPP e seu estado de conexão, ou seja, se etá conectado ou desconectado.
- Botão para interromper a rotina e abortar o processo.
- Estado da rotina, isto é, estável ou se está em transição de temperatura.

Com as especificações estabelecidas, a próxima etapa foi a criação de um esboço para o posicionamento dos dados, procurando a melhor forma de disponibilizá-los ao usuário. Com isso, chegou-se no seguinte esquema:

Figura 41 – Diagrama dos componentes gráficos da interface gráfica da janela principal

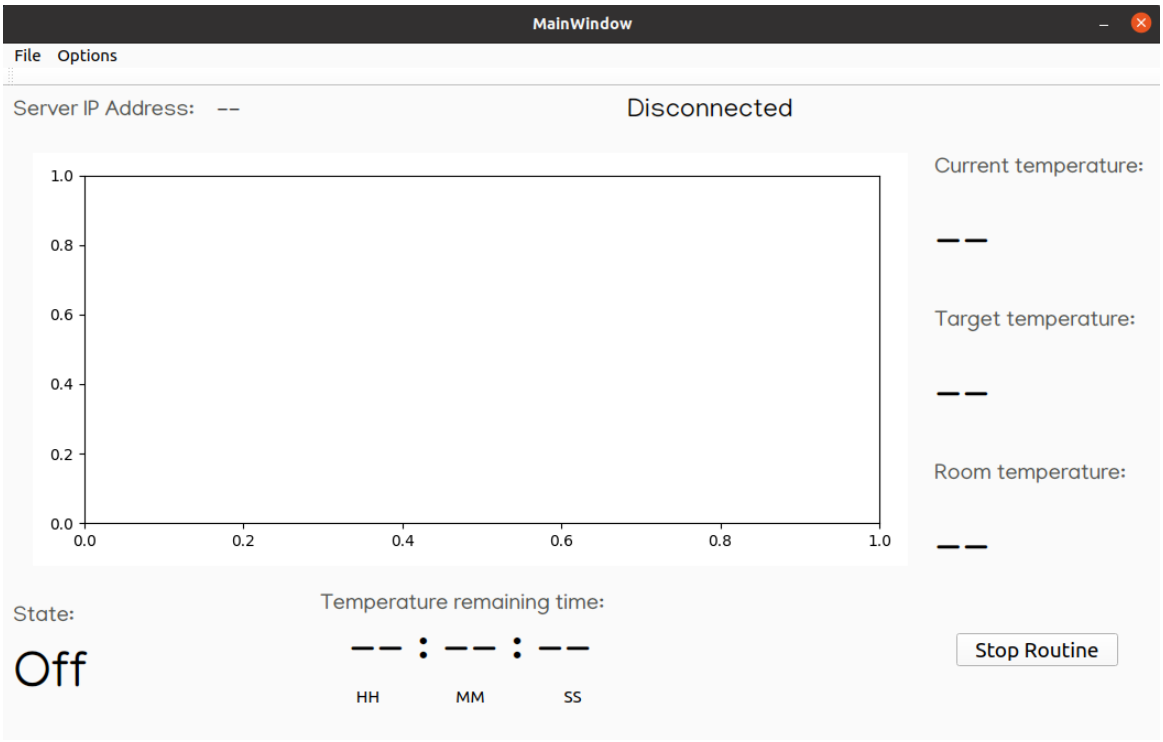


Fonte: Elaboração própria (2021).

Após da criação do esboço da interface, utilizou-se a ferramenta *Qt Designer* para o posicionamento dos elementos gráficos. Para geração do código em Python da interface elaborada, aplicou-se o programa *pyuic5* (PYTHON BASICS, 2020). Com isso, a Figura 42 representa o resultado da interface da janela principal.

A escolha do posicionamento dos componentes da interface principal baseou-se em testes com usuários eventuais. Isso contribuiu para adição de legendas para os valores, informando seus respectivos significados, tornando a interface mais compreensível.

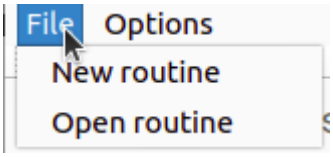
Figura 42 – Interface gráfica da janela principal



Fonte: Elaboração própria (2021).

No menu *File* no canto superior esquerdo na Figura 42 foram desenvolvidos dois botões, um para criar uma rotina e outro para abrir uma já criada e salva, conforme mostra a Figura 43.

Figura 43 – Aba para criação ou abertura de rotinas.



Fonte: Elaboração própria (2021).

Por fim, com o código em Python exportado a partir do Qt Designer, foram criadas classes para poder configurar os valores exibidos na tela periodicamente, conforme o Apêndice B.

4.4.2 Criação de Rotinas

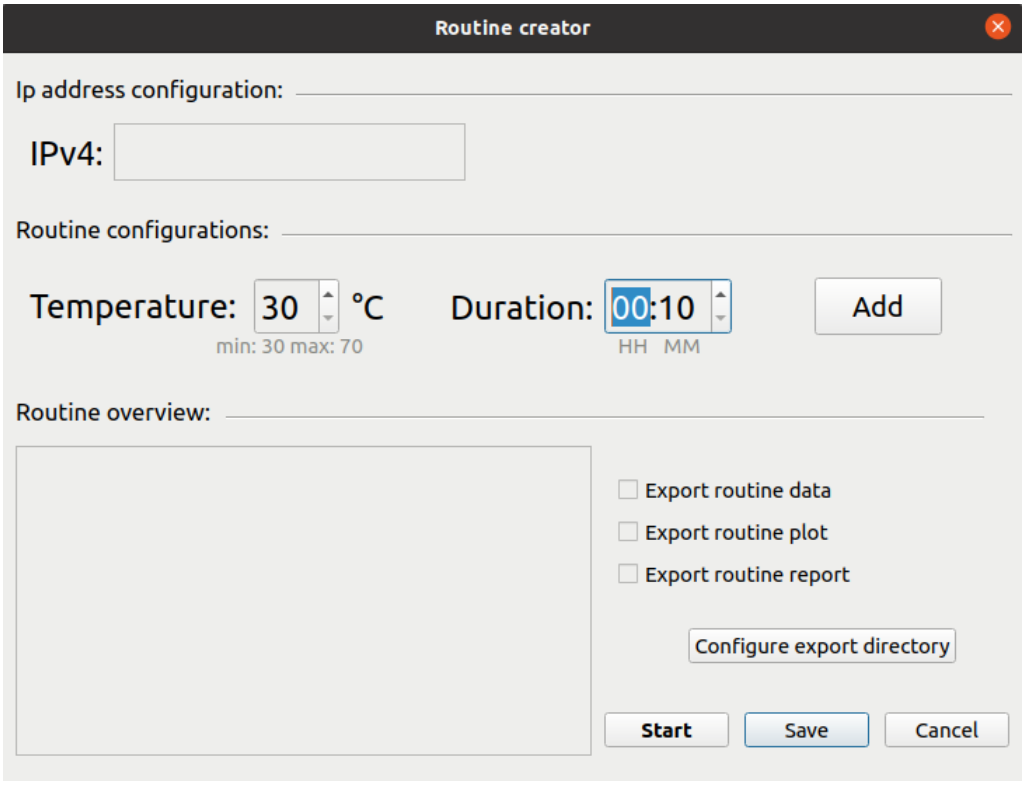
Uma das funcionalidades principais do software é a criação das rotinas de temperatura. Este recurso é composto por dois elementos, a temperatura e o período. Ou seja, o usuário pode configurar uma sequência de rotinas, sendo que cada uma tem um valor de temperatura e o período em que forno manterá esta temperatura.

O início do desenvolvimento deste recurso se deu pela especificação das principais características e funcionalidades que ele deve contemplar. Baseado nisto, os requisitos relacionados às rotinas estão citados abaixo:

- Interface gráfica que possibilita a configuração do servidor HTTP do forno, criação de uma sequência de rotinas, salvar a rotina criada, opções de exportar os dados relacionados ao processo de teste e mostrar as rotinas que estão sendo configuradas.
- Cálculo do tempo decorrido a partir do momento que a temperatura escolhida é atingida.
- Realizar as requisições ao servidor para configurar a temperatura desejada e obter a temperatura atual.

A interface gráfica para a criação de rotinas é aberta quando a aba *New Routine* é clicada na janela principal. O resultado da elaboração da interface denominada *Routine Creator* está representado pela Figura 44.

Figura 44 – Interface gráfica para criação de rotinas.



Fonte: Elaboração própria (2021).

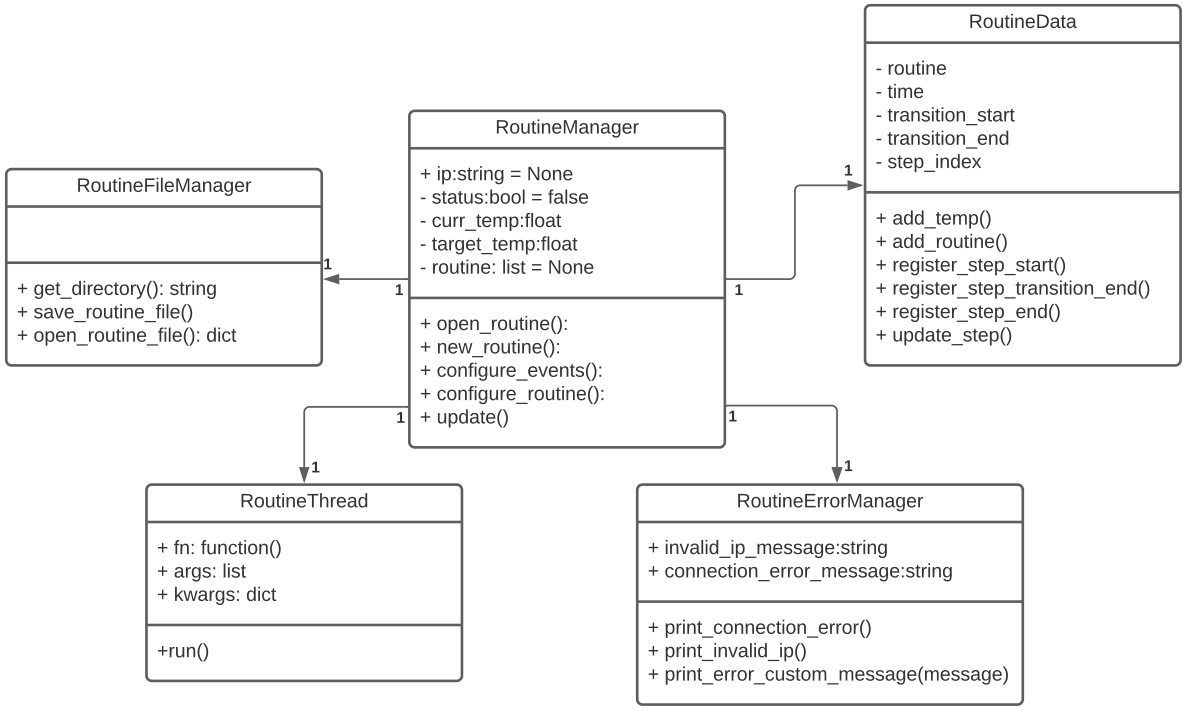
Além da inclusão das especificações, citadas anteriormente, na interface gráfica foi adicionado um botão para o usuário configurar o diretório de destino dos ar-

quivos que serão exportados após as rotinas finalizarem. Semelhante à janela principal, após o desenvolvimento da interface, o código Python foi exportado.

Quanto ao desenvolvimento do código deste recurso, foi criada uma classe para assumir o papel central, atuando como gerenciador. Com isso, esta classe tem responsabilidades de iniciar a rotina, atualizar o valor de temperatura desejado, cronometrar o tempo em que a rotina se encontra e monitorar a temperatura atual, conforme mostra o Apêndice C.

Como o gerenciamento das rotinas envolve muitas funcionalidades, algumas classes adicionais foram criadas de forma deixar o código mais modular e escalável. O diagramas das classes envolvidas com seus respectivos métodos e atributos principais é representado pela Figura 45.

Figura 45 – Diagrama UML de classes das classes relacionadas ao gerenciador de rotinas.



Fonte: Elaboração própria (2021).

A classe relacionada à exportação dos arquivos, apesar de se relacionar com o gerenciador de rotinas, ela será apresentada na Sessão 4.4.3. Já no tocante às classes representadas na Figura 45, a Tabela 4 exibe as principais responsabilidades de cada uma.

Tabela 4 – Relação das classes do gerenciamento das rotinas e suas respectivas responsabilidades.

Classe	Responsabilidade
RoutineManager	Esta é a classe central do gerenciamento de rotinas, tem como responsabilidades fornecer os dados das rotinas que serão exibidos na janela principal, realizar requisições ao servidor, inciar e finalizar as rotinas e criar as rotinas.
RoutineData	Tem como objetivo armazenar os dados das rotinas, como o tempo de início, fim e período de transição entre rotinas.
RoutineFileManager	Esta classe é utilizada para obter os diretórios onde deseja-se salvar ou abrir um arquivo.
RoutineErrorManager	Sua responsabilidade é mostrar mensagens usuário alertando-o sobre algum erro que tenha ocorrido, como, por exemplo, entrada de dados inválidos ou desconexão com o servidor.
RoutineThread	Esta classe criará *threads* durante as requisições HTTP, pois, assim, a aplicação não dependerá da resposta do servidor HTTP para realizar as suas demais funções, ou seja, a requisição é realizada de forma concorrente com o resto da aplicação.

Fonte: Elaboração própria (2021).

4.4.3 Exportação dos Dados da Rotina

O terceiro e último principal recurso do software de controle e monitoramento é a exportação dos dados das rotinas que ocorreram. Isso possibilita o projetista correlacionar o desempenho do seu sistema eletrônico de acordo com a temperatura que foi configurada, além da oportunidade de ter os testes documentados.

Os dados relacionados as rotinas são gerenciados pela classe *RoutineData*, como mostrado na Tabela 4. Com isso, foram estabelecidos dois principais objetivos para a exportação dos dados. O primeiro é a criação de uma tabela em código Linguagem de Marcação de Hiper Texto (*Hyper Text Markup Language* - HTML) contendo os principais dados, conforme a tabela abaixo:

Tabela 5 – Dados contidos na tabela de exportação dos dados das rotinas e seu significado.

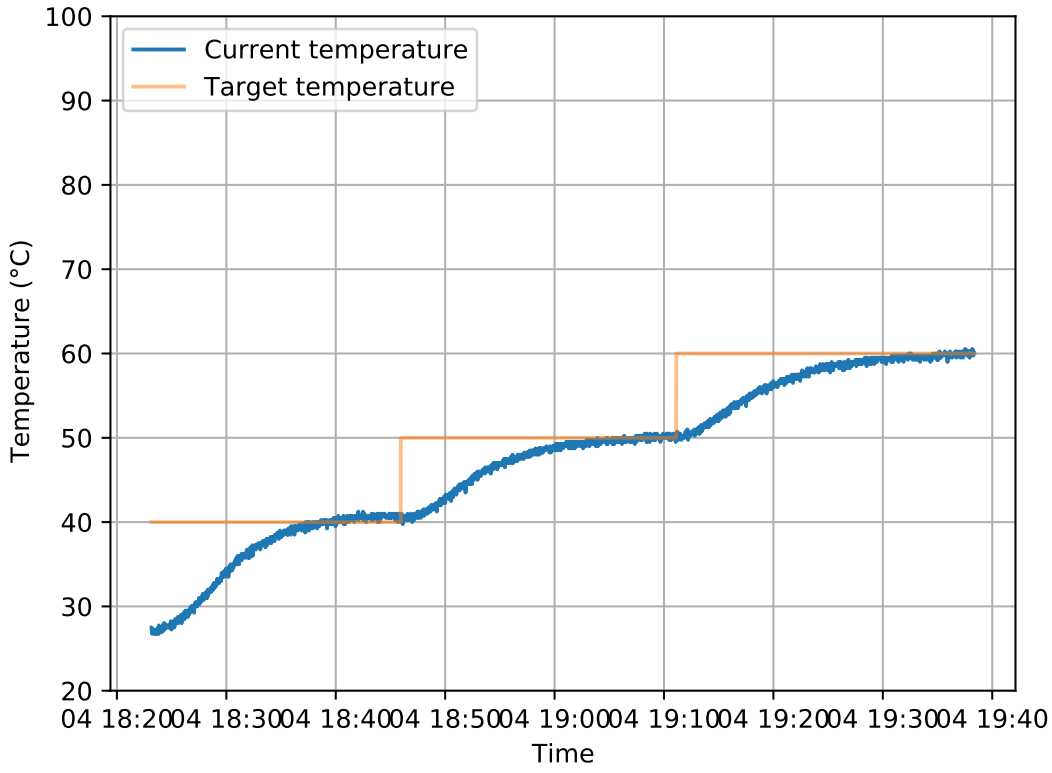
Nome da coluna	Significado
Temperatura Alvo	Temperatura desejada no interior do forno elétrico.
Tempo de início	Instante em que a rotina iniciou.
Tempo de fim	Instante em que a rotina foi finalizada.
Duração	Tempo em que a temperatura desejada foi mantida no interior do forno elétrico.
Tempo de transição	Tempo em que a temperatura do forno ficou entre a valor definido na rotina anterior até se estabilizar na temperatura desejada da rotina atual.

Fonte: Elaboração própria (2021).

O segundo objetivo na exportação de dados é elaboração de um gráfico com todo o decorrer das rotinas em um arquivo Arquivo de Documento Portátil (*Portable Document File* - PDF).

Com isso, foi desenvolvida uma classe responsável por exportar os arquivos, denominada *RoutineExporter* (Apêndice D). O modelo de resultado dos arquivos exportados estão representados pelas Figuras 47 e 46.

Figura 46 – Exemplo de gráfico exportado após a finalização das rotinas.



Fonte: Elaboração própria (2021).

Figura 47 – Exemplo de tabela de dados após a finalização das rotinas.

Target temperature (°C)	Start time	End time	Duration (s)	Transition time (s)
40	2021-01-04 18:23:08.153839	2021-01-04 18:45:55.428006	1367.3	766.3
50	2021-01-04 18:45:55.428234	2021-01-04 19:11:04.433780	1509.0	908.0
60	2021-01-04 19:11:04.433864	2021-01-04 19:38:21.434715	1637.0	1037.0

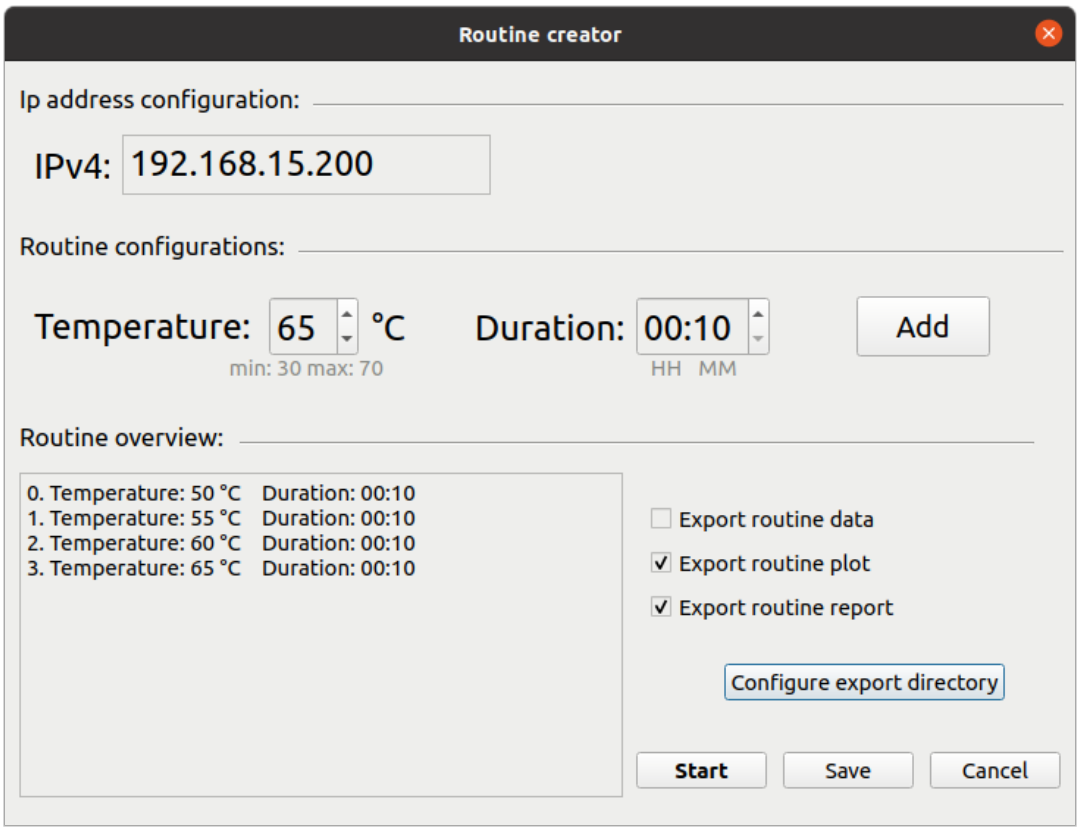
Fonte: Elaboração própria (2021).

4.5 Resultados Obtidos

Após o desenvolvimento das partes que compõe este trabalho, ou seja, o controlador PI, o servidor HTTP e o software de controle e monitoramento, deve-se realizar a integração de todos estes componentes para executar o teste de rotinas com temperaturas variadas.

Inicialmente, foram criadas quatro rotinas com temperaturas distintas e 10 minutos de duração em cada temperatura, conforme mostra a Figura 48.

Figura 48 – Janela de criação de rotinas com a criação das rotinas de teste.

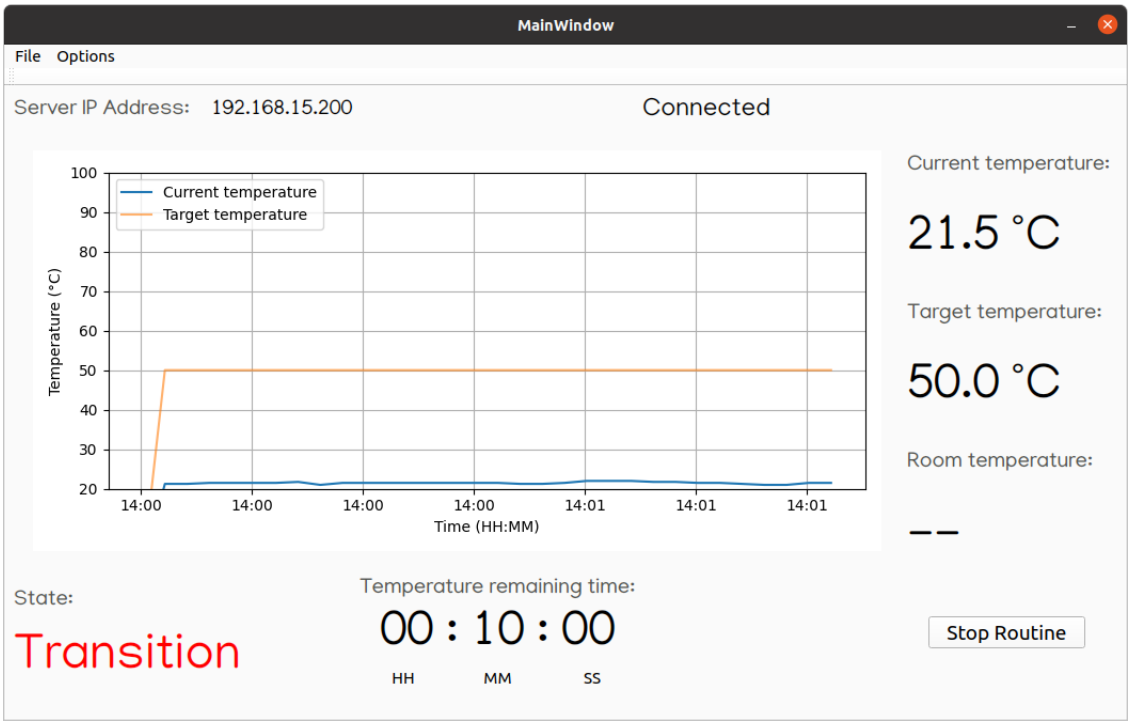


Fonte: Elaboração própria (2021).

Nota-se na figura acima que foram selecionadas as opções de exportar o gráfico da temperatura em função do tempo e a tabela com as principais informações das rotinas após a sua finalização. Seguinte à inicialização do teste, nota-se que o

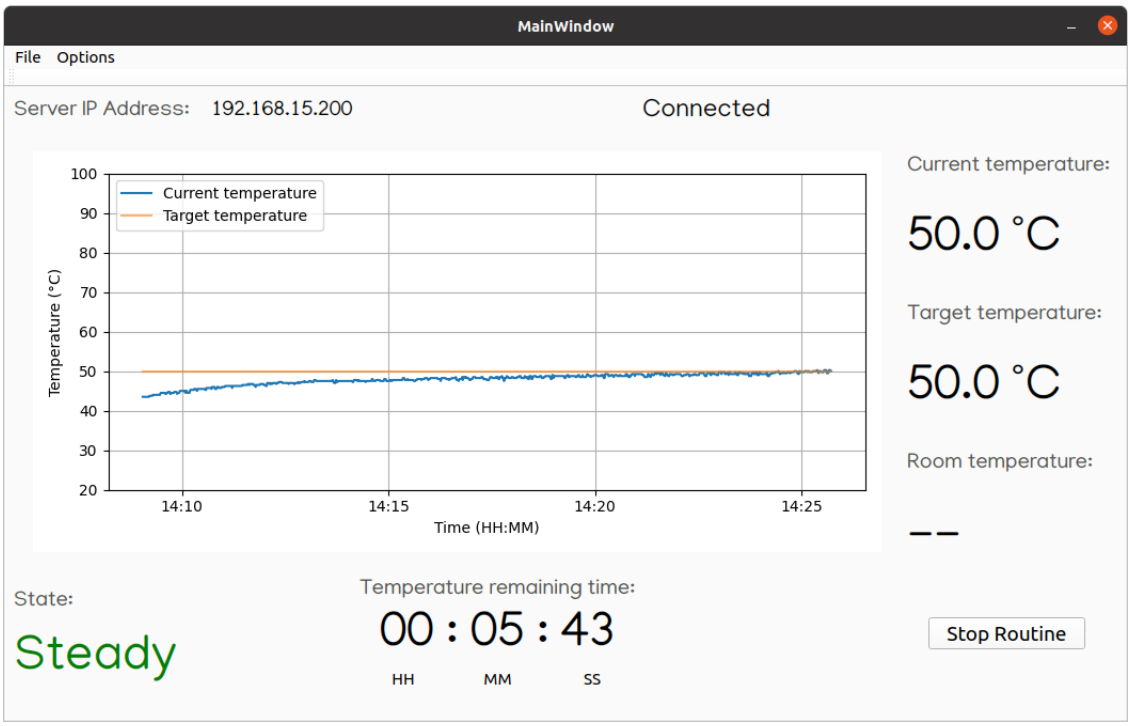
software configura o valor da temperatura desejada no interior do forno, conforme mostra a Figura 49. Já a Figura 50 mostra a temperatura no interior do forno já estável depois de aproximadamente cinco minutos da primeira rotina configurada.

Figura 49 – Interface principal logo após a inicialização do teste.



Fonte: Elaboração própria (2021).

Figura 50 – Interface principal após a temperatura estável da primeira rotina.

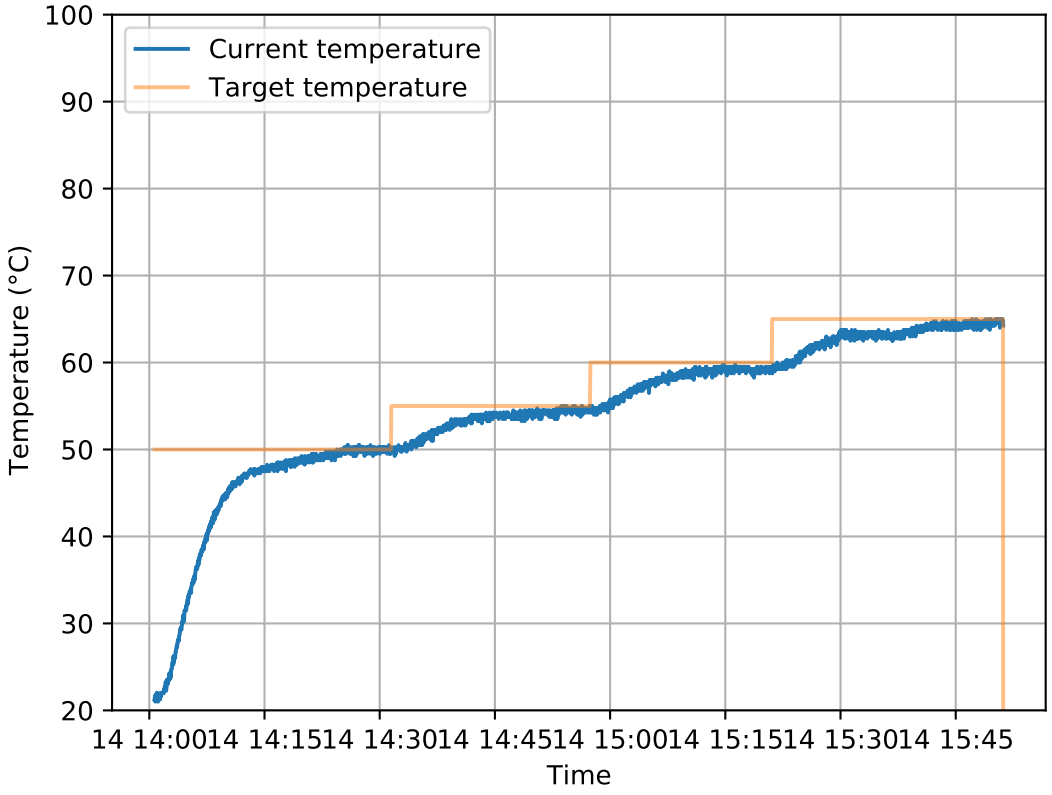


Fonte: Elaboração própria (2021).

Como o termopar possuí um erro de leitura de 0,25 ℃, foi arbitrado que pode haver um erro de 0,5 ℃ para iniciar a contagem regressiva da rotina, ou seja, quando a temperatura atual do forno estive 0,5 ℃ abaixo da temperatura desejada, já será considerado início da rotina e a contagem regressiva inicializará.

Com as rotinas finalizadas, os arquivos selecionados no início do teste foram exportados, como mostram as figuras 51 e 52.

Figura 51 – Gráfico da temperatura em função do tempo exportado após a finalização das rotinas.



Fonte: Elaboração própria (2021).

Figura 52 – Tabela com os dados das rotinas.

Target temperature (°C)	Start time	End time	Duration (s)	Transition time (s)
50	2021-03-14 14:00:39.929313	2021-03-14 14:31:28.090268	1848.2	1247.2
55	2021-03-14 14:31:28.090377	2021-03-14 14:57:23.090598	1555.0	955.0
60	2021-03-14 14:57:23.090726	2021-03-14 15:21:03.094091	1420.0	820.0
65	2021-03-14 15:21:03.094276	2021-03-14 15:51:11.090833	1808.0	1208.0

Fonte: Elaboração própria (2021).

Analisando a Figura 52, nota-se que o tempo de duração está de acordo com o que foi planejado, ou seja, a diferença entre a duração e o tempo de transição resulta no tempo que foi configurado que se mantivesse na temperatura desejada de cada rotina, neste caso, dez minutos.

Já a Figura 51, mostra que não houve sobressinal e, após o período de transição em cada rotina, a temperatura estabilizou-se em torno da temperatura dese-

jada, havendo apenas um pequeno desvio devido aos erros de leitura da medição do termopar.

Desta forma, pode-se afirmar que o comportamento da temperatura no interior do forno elétrico assemelhou-se com o que foi planejado através do software de controle e monitoramento. Com isso, confirmou-se que o projeto atingiu os requisitos de projeto, habilitando o projetista de sistemas eletrônicos testar suas soluções sob estresse de altas temperaturas de forma automatizada.

5 CONSIDERAÇÕES FINAIS

Neste trabalho apresentou-se o desenvolvimento de um sistema para realizar testes em produtos eletrônicos sob temperatura controlada de forma automatizada. Com isso, projetistas de hardware poderão verificar se seus circuitos estão aptos para funcionar em temperaturas especificadas pelo produto, mantendo seu funcionamento dentro do que é esperado.

O modelo matemático obtido para este forno elétrico mostrou-se satisfatório para a definição dos parâmetros do controlador, uma vez que, quando comparada a resposta real com a resposta da simulação, os resultados foram semelhantes.

Quanto ao projeto do controlador PI, foi realizado de maneira simples, utilizando as equações de Ziegler-Nichols para definição dos ganhos do termos que o compõe. Não apresentando sobressinais significativos, ou seja, as variações de leitura dos termopares foram responsáveis por sobressinais abaixo de 1 ℃, o controlador teve um comportamento satisfatório.

Por último, este trabalho demonstrou a elaboração do software de controle e monitoramento, tornando a utilização da aplicação simples, intuitiva e automatizada. Nele, mostrou-se que os principais dados das rotinas que ocorreram poderão ser exportados, podendo ser anexados à documentação dos testes de validação, por exemplo.

Com isso, os objetivos específicos foram atingidos. Assim, possibilitou-se realizar testes em produtos eletrônicos em uma empresa de tecnologia.

Em relação à bibliografia utilizada, esta foi suficiente para o desenvolvimento deste projeto, apesar de muitas delas serem livros em inglês.

Para pesquisas futuras, sugere-se a utilização de sensores de temperatura mais precisos, ou seja, não atendendo uma faixa de temperatura tão abrangente quanto ao do termopar. Sugere-se, também, a implantação de materiais isolantes térmicos nas paredes externas do forno elétrico, isso deixará o interior do forno mais próximo de um sistema adiabático. Além disso, propõe-se a implementação de dois novos recursos, o controle da umidade de a possibilidade de resfriar o interior do forno.

Ainda no tocante à pesquisas futuras, recomenda-se projetar um controlador mais eficiente, com uma ação proporcional e integral mais agressiva e o uso da ação da derivativa para diminuir o sobressinal e o tempo de acomodação.

REFERÊNCIAS

ANATEL. *Ato nº 14098, de 23 de novembro de 2017*. 2019. Disponível em: <https://www.anatel.gov.br/legislacao/atos-de-certificacao-de-produtos/2017/1231-ato-14098>. Acesso em: 24 nov. 2020. Citado na página 14.

ANATEL. *Certificação de Produtos*. 2019. Disponível em: <https://antigo.anatel.gov.br/setorregulado/apresentacao-certificacao>. Acesso em: 24 nov. 2020. Citado na página 14.

BAKER, G. A.; GRAVES-MORRIS, P. *Padé Approximants*. Cambridge: Cambridge University Press, 1996. 764 p. Citado na página 25.

BAQUETTE, B. W. *Process Control*: Modeling, design and simulation. Ohio: Prentice Hall, 2002. 769 p. Citado 3 vezes nas páginas 25, 26 e 29.

CHAMBERS, P. *Heat Has a Negative Affect on Electronics*. Element5, 2016. Disponível em: <https://element5digital.com/heat-has-a-negative-affect-on-electronics/#:~:text=Allowing%20systems%20to%20run%20for,functional%20the%20machine%20will%20become.> Acesso em: 24 nov. 2020. Citado na página 14.

CURADO, A. *O que é temperatura – Conceito, tipos de medição e escalas*. Conhecimento Científico, 2020. Disponível em: <https://conhecimentocientifico.r7.com/o-que-e-temperatura/>. Acesso em: 1 dez. 2020. Citado na página 17.

ESPRESSIF SYSTEMS. *ESP-IDF*: Esp32. [S.l.], 2020. Disponível em: <https://docs.espressif.com/projects/esp-idf/en/latest/esp32/get-started/>. Citado na página 40.

ESPRESSIF SYSTEMS. *LED Control*. 2021. Disponível em: <https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/peripherals/ledc.html>. Acesso em: 20 de dez. 2020. Citado na página 47.

FILHO, J. M. *Instalações Elétricas Industriais*. 7. ed. Rio de Janeiro: Livros Técnicos e Científicos Editora, 2007. 948 p. Citado na página 20.

FOTEK. *SSR Series*: Solid state relay. [S.l.], 20–? Disponível em: <https://cdn.sparkfun.com/datasheets/Components/General/SSR40DA.pdf>. Citado na página 40.

FRANCISCO, W. de C. *Climas do Brasil*. Brasil Escola, 2020. Disponível em: <https://brasilescola.uol.com.br/brasil/os-climas-brasil.html>. Acesso em: 24 nov. 2020. Citado na página 14.

GERHARDT, T. E.; SILVEIRA, D. T. *Métodos de Pesquisa*. Porto Alegre: Universidade Federal do Rio Grande do Sul, 2009. Citado na página 43.

GIL, A. C. *Como Elaborar Projetos de Pesquisa*. 4. ed. São Paulo: Atlas, 2002. Citado na página 43.

GUERRA, L. N. de A. *USO DE COMPENSADOR PID NO CONTROLE DA TAXA DE VARIAÇÃO DE TEMPERATURA EM UM FORNO ELÉTRICO A RESISTÊNCIA*. Trabalho de conclusão de curso (Graduação) — Universidade Federal do Rio de Janeiro, Rio de Janeiro, Out. 2006. Citado 5 vezes nas páginas 24, 27, 28, 36 e 39.

HALLIDAY, R. R. D.; WALKER, J. *Fundamentos da Física*: Volume 2 - gravitação, ondas e termodinâmica. Rio de Janeiro: LTC, 2009. Citado na página 18.

HANTA, V.; PROCHáZKA, A. Rational approximation of time delay. 10 2009. Disponível em: <https://www2.humusoft.cz/www/papers/tcp09/035_hanta.pdf>. Acesso em: 10 jan. 2021. Citado na página 26.

JACKSON, D. *What Exactly is Cold Junction Compensation? How Does It Relate to the Use of Thermocouples as Temperature Sensors?* TEGAM Inc., 2019. Disponível em: <https://www.tegam.com/what-exactly-is-cold-junction-compensation/>. Acesso em: 13 jan. 2021. Citado na página 38.

JOHSON, M. A.; MORADI, M. H. *PID Control*: New identification and design methods. London: Springer, 2005. 572 p. Citado 4 vezes nas páginas 29, 30, 31 e 32.

LIMA, C. B. de; VILLAÇA, M. V. M. *AVR e Arduino*: Técnicas de projeto. 2. ed. Florianópolis: Clube dos Autores, 2012. 612 p. Citado na página 28.

MARCONI, M. de A.; LAKATOS, E. M. *Técnicas de Pesquisa*. São Paulo: Atlas, 2017. Citado na página 43.

MAXIM INTEGRATED. *MAX6675*: Cold-junction-compensated k-thermocoupleto-digital converter (0°c to +1024°c). [S.l.], 2014. Disponível em: <https://datasheets.maximintegrated.com/en/ds/MAX6675.pdf>. Acesso em: 15 jan 2021. Citado 3 vezes nas páginas 37, 38 e 39.

MICROCHIP. *AVR221*: Discrete pid controller on tinyavr and megaavr devices. [S.l.], 2016. Disponível em: <http://ww1.microchip.com/downloads/en/Appnotes/Atmel-2558-Discrete-PID-Controller-on-tinyAVR-and-megaAVR_ApplicationNote_AVR221.pdf>. Acesso em: 17 jan 2021. Citado 2 vezes nas páginas 32 e 33.

MOZILLA. *HTTP*. 2020. Disponível em: <https://developer.mozilla.org/en-US/docs/Web/HTTP>. Acesso em: 21 jan. 2021. Citado na página 41.

NILSSON, J. W.; RIEDEL, S. A. *Circuitos Elétricos*. 5. ed. Rio de Janeiro: Livros Técnicos e Científicos Editora, 1999. 539 p. Citado 2 vezes nas páginas 18 e 19.

OGATA, K. *Modern Control Engineering*. 5. ed. New Jersey: Prentice Hall, 2009. 912 p. Citado 4 vezes nas páginas 22, 30, 34 e 35.

OMEGA. *Thermocouple types*. Omega Engineering Inc., 2019. Disponível em: <https://www.omega.com/en-us/resources/thermocouple-types>. Acesso em: 13 jan. 2021. Citado na página 36.

OMEGA. *Working principle of thermocouples*. Omega Engineering Inc., 2019. Disponível em: <https://www.omega.com/en-us/resources/how-thermocouples-work>. Acesso em: 13 jan. 2021. Citado na página 36.

PURWAR, A.; DEEP, S. A novel thermocouple for ultra high temperature applications: Design and computational analysis. Jun. 2018. Disponível em: <https://www.researchgate.net/publication/323647209_A_novel_thermocouple_for_ultra_high_temperature_applications_Design_and_computational_analysis/link/5b3c8a8f4585150d23f6a305/download>. Acesso em: 10 jan. 2021. Citado na página 36.

PYTHON BASICS. *Qt Designer Python*. 2020. Disponível em: <https://pythonbasics.org/qt-designer-python/>. Acesso em: 15 nov. 2020. Citado na página 63.

PYTHON-CONTROL. *control.pade*. 2019. Disponível em: <https://python-control.readthedocs.io/en/0.8.3/generated/control.pade.html>. Acesso em: 20 de dez. 2020. Citado na página 53.

PYTHON-CONTROL. *control.step*. 2019. Disponível em: <https://python-control.readthedocs.io/en/0.8.4/generated/control.step_response.html#control-step-response>. Acesso em: 20 de dez. 2020. Citado na página 53.

PYTHON-CONTROL. *control.TransferFunction*. 2019. Disponível em: <https://python-control.readthedocs.io/en/0.8.4/generated/control.TransferFunction.html#control.TransferFunction>. Acesso em: 20 de dez. 2020. Citado na página 53.

QT. *About Qt*. 2019. Disponível em: <https://wiki.qt.io/About_Qt>. Acesso em: 27 de fev. 2021. Citado na página 41.

SCHULZ, D. *Calor*. UFRGS, 2009. Disponível em: <https://www.if.ufrgs.br/~dschulz/web/calor.htm>. Acesso em: 1 dez. 2020. Citado na página 17.

VASCONCELLOS, A. P. M. de. *PROJETO DE CONTROLADORES PI E PID PARA UM FORNO AQUECEDOR DE OLEO DE UMA PLANTA DE TRATAMENTO DE HIDROCARBONETOS*. Trabalho de conclusão de curso (Graduação) — Universidade Federal do Rio de Janeiro, Rio de Janeiro, Fev. 2017. Citado na página 31.

WADHWANI, S. Y. P. *Embedded systems Market Size By Component*. Global Market Insights, 2020. Disponível em: <https://www.gminsights.com/industry-analysis/embedded-system-market>. Acesso em: 24 nov. 2020. Citado na página 14.

ÅSTRÖM, K. J.; HÄGGLUND, T. *PID Controllers: theory, design and tuning*. 2. ed. North Carolina: Instrument Society of America, 1995. 343 p. Citado 3 vezes nas páginas 24, 33 e 34.

APÊNDICES

APÊNDICE A – CÓDIGO DA IMPLEMENTAÇÃO DO CONTROLADOR PI NO ESP32

```
1   static void ssr_control_task(void *pvParameters)
2   {
3       double Kp = 1.988;
4       double Ki = 0.0062;
5       double error = 0, curr_decimal, target_decimal;
6
7       double proportional, integral = 0, pi, derivative = 0;
8       int fs = 1;
9       int duty;
10
11      sensor_read_t curr, target, last_target;
12
13      last_target.integer = 0;
14
15      ssr_init();
16
17      while(1){
18
19          curr = get_actual_temp();
20          target = get_target_temp();
21
22          if(target.integer){
23
24              target_decimal = ((double) target.decimal)/100;
25              curr_decimal = ((double) curr.decimal)/100;
26
27              if (target.integer < last_target.integer){
28                  if (curr.integer < target.integer)
29                      integral = 0;
30
31                  last_target = target;
32              }
33
34              error = (double) target.integer - (double) curr.integer;
35              error = error + (target_decimal - curr_decimal);
36
37              proportional = Kp * error;
38              integral += Ki * error;
39
40          }
41          else{
42              proportional = 0;
43              integral = 0;
```

```
44                  }
45
46                  last_target = target;
47
48                  pi = proportional + integral + derivative;
49
50                  duty = (pi * 16/62);
51
52                  if (duty > 64) duty = 64;
53                  if (duty < 0) duty = 0;
54
55                  duty = duty * 256 * 4;
56
57                  ESP_LOGI(TAG, "Current: %d.%d\tTarget: %d.%d\tError: %.2f\tControl:
      %.2f\tK: %.2f\tI: %.2f\tD: %.2f\tDuty: %d",
58                          curr.integer, curr.decimal,
59                          target.integer, target.decimal,
60                          error, pi, proportional, integral, derivative, duty);
61
62
63                  ssr_set_duty(duty);
64                  vTaskDelay(1000 / portTICK_RATE_MS);
65              }
66          }
67
```

APÊNDICE B – CÓDIGO DAS CLASSES CRIADAS PARA MANIPULAÇÃO DA INTERFACE GRÁFICA PRINCIPAL

```
68          class ChamberMainWindow(QtWidgets.QMainWindow, Ui_MainWindowSetup):
69              def __init__(self):
70                  super(QtWidgets.QMainWindow, self).__init__()
71
72                  self.target_temp = None
73                  self.curr_temp = None
74                  self.time_remaining = {
75                      "hour": "--",
76                      "minute": "--",
77                      "second": "--"
78                  }
79                  self.routine_manager = RoutineManager()
80
81                  self.setupUi(self)
82                  self.init_widget()
83                  self.configure_events()
84
85
86              def init_widget(self):
87                  self.matplotlib_widget = MatplotWidget()
88                  self.layoutvertical = QtWidgets.QVBoxLayout(self.plot_widget)
89                  self.layoutvertical.addWidget(self.matplotlib_widget)
90
91
92              def configure_events(self):
93                  self.actionNew_routine.triggered.connect(self.routine_manager.
    new_routine)
94                  self.actionOpen_routine.triggered.connect(self.routine_manager.
    open_routine)
95                  self.stop_button.clicked.connect(self.routine_manager.end_routine)
96                  self.configure_timers()
97
98
99              def configure_timers(self):
100                 self.timer = QtCore.QTimer()
101                 self.timer.setInterval(1000)
102                 self.timer.timeout.connect(self.update)
103                 self.timer.start()
104
105
106             def update(self):
107                 if self.routine_manager.status == True:
108                     self.routine_manager.update()
```

```
109                     self.curr_temp = self.routine_manager.get_current_temp()
110                     self.target_temp = self.routine_manager.get_target_temp()
111
112                     if self.routine_manager.status == True:
113                         self.time_remaining = self.routine_manager.
    get_period_remaining()
114
115                     self.update_plot()
116                     self.update_temp_labels()
117                     self.update_state()
118                     self.update_remaining_time()
119
120                     self.conn_status_value.setText("Connected")
121                     if self.server_ip_label_value.text() == "--":
122                         self.server_ip_label_value.setText(self.routine_manager.ip)
123
124
125             def update_temp_labels(self):
126                 self.update_target_temp()
127                 self.update_current_temp()
128
129
130             def update_plot(self):
131                 self.matplotlib_widget.update_plot(self.curr_temp, self.target_temp
    )
132
133
134             def update_current_temp(self):
135                 self.curr_temp_value.setText(f"{self.curr_temp} C")
136
137
138             def update_target_temp(self):
139                 self.target_temp_value.setText(f"{self.target_temp} C")
140
141
142             def update_state(self):
143                 if self.routine_manager.transient:
144                     self.state_value.setText("Transition")
145                     self.state_value.setStyleSheet("Color: Red")
146
147                 elif self.routine_manager.transient == False:
148                     self.state_value.setText("Steady")
149                     self.state_value.setStyleSheet("Color: Green")
150
151                 elif self.routine_manager.status == False:
152                     self.state_value.setText("Off")
153                     self.state_value.setStyleSheet("Color: Gray")
```

```
154
155
156             def update_remaining_time(self):
157                 self.time_remaining_value.setText(
158                     f"{self.time_remaining['hour']:02d} : {self.time_remaining['
    minute']:02d} : {self.time_remaining['second']:02d}"
159                 )
160
161
162             def update_room_temp(self):
163                 pass
164
165
166             def disconnect(self):
167                 self.conn_status_value.setText("Disconnected")
168                 self.conn_status_value.setStyleSheet("color: Red")
169
170
171         class MatplotWidget(QtWidgets.QWidget):
172
173             def __init__(self, parent=None, max_samples=1000, ylim=[20, 100]):
174                 super(MatplotWidget, self).__init__(parent)
175                 self.figure = Figure(tight_layout=True)
176
177                 self.canvas = FigureCanvas(self.figure)
178                 self.axis = self.figure.add_subplot(111)
179                 self.layoutvertical = QtWidgets.QVBoxLayout(self)
180                 self.layoutvertical.addWidget(self.canvas)
181                 self.max_samples = max_samples
182                 self.time = []
183                 self.curr_temp = []
184                 self.target_temp = []
185                 self.ylim = ylim
186                 self.formatter = dates.DateFormatter("%H:%M")
187
188
189             def update_plot(self, current, target):
190                 now = datetime.datetime.now()
191
192                 if len(self.time) > self.max_samples:
193                     self.curr_temp = self.curr_temp[1:] + [current]
194                     self.target_temp = self.target_temp[1:] + [target]
195                     self.time = self.time[1:] + [now]
196                 else:
197                     self.curr_temp.append(current)
198                     self.target_temp.append(target)
199                     self.time.append(now)
```

```
200
201                 self.axis.clear()
202                 self.axis.xaxis.set_major_formatter(self.formatter)
203                 self.axis.plot(self.time, self.curr_temp, label="Current
    temperature")
204                 self.axis.plot(self.time, self.target_temp, alpha=0.6, label="
    Target temperature")
205                 self.axis.grid()
206                 self.axis.legend(loc="upper left")
207                 self.axis.set_ylim(self.ylim)
208                 self.axis.set_xlabel("Time (HH:MM)")
209                 self.axis.set_ylabel("Temperature (C)")
210                 self.canvas.draw()
211
```

APÊNDICE C – CÓDIGO DA CLASSE CRIADAS PARA O GERENCIAMENTO DAS ROTINAS

```
212         class RoutineManager(QtWidgets.QDialog, Ui_RoutineCreator):
213 
214             port = 80
215 
216             def __init__(self, routine=[], ip=None):
217                 super(QtWidgets.QDialog, self).__init__()
218                 self.upper_limit = 2
219                 self.lower_limit = .5
220                 self.curr_temp = 0
221                 self.target_temp = 0
222                 self.transient = False
223                 self.status = False
224                 self.routine = routine
225                 self.curr_step = None
226                 self.last_step = None
227                 self.initial_time = None
228                 self.ip = ip
229                 self.routine_error_manager = RoutineErrorManager()
230                 self.routine_data = RoutineData()
231                 self.exporter = RoutineExporter()
232                 self.file_manager = RoutineFileManager()
233 
234                 self.climatic_chamber = None
235                 self.threadpool = QtCore.QThreadPool()
236 
237                 self.setupUi(self)
238                 self.configure_events()
239 
240 
241             def new_routine(self):
242                 if self.status:
243                     self.routine_error_manager.print_error_custom_message(
244                         "You need to stop the current routine in order to start a
    new one."
245                     )
246                     return
247 
248                 self.show()
249                 self.exec_()
250 
251 
252             def open_routine(self):
253                 if self.status:
```

```
254                     self.routine_error_manager.print_error_custom_message(
255                         "You need to stop the current routine in order to open a
    routine."
256                     )
257                     return
258
259                 configs = self.file_manager.open_routine_file()
260
261                 if configs is not False:
262                     self.show()
263
264                     self.configure_routine(configs)
265                     self.exec_()
266
267
268             def configure_events(self):
269                 self.r_button_add.clicked.connect(self.add_clicked)
270                 self.r_button_next.clicked.connect(self.next_clicked)
271                 self.r_button_cancel.clicked.connect(self.cancel_clicked)
272                 self.r_button_save.clicked.connect(self.save_clicked)
273                 self.r_plot_checkbox.stateChanged.connect(self.plot_export_clicked)
274                 self.r_report_checkbox.stateChanged.connect(self.
    report_export_clicked)
275                 self.configure_path_button.clicked.connect(self.
    configure_dir_clicked)
276
277
278             def configure_routine(self, configs):
279                 for step in configs['routine']:
280                     self.add_routine(
281                         temp=step['temperature'],
282                         hour=step['hour'],
283                         minute=step['minute']
284                     )
285
286                 self.ip = configs['ip']
287                 self.r_text_ip.setText(self.ip)
288
289
290             def update(self):
291                 if self.status:
292                     worker = RoutineThread(self.update_server)
293                     self.threadpool.start(worker)
294                     self.update_routine()
295
296
297             def update_server(self):
```

```
298                 try:
299                     if self.target_temp != self.curr_step['temperature']:
300                         self.climatic_chamber.set_target_temp(self.curr_step["
    temperature"], 2)
301 
302                     self.target_temp = self.climatic_chamber.get_target_temp(2)
303                     self.curr_temp = self.climatic_chamber.get_current_temp(2)
304                     self.routine_data.add_temp(self.curr_temp, self.target_temp)
305 
306                 except requests.exceptions.Timeout:
307                     print("Timeout")
308                     #self.routine_error_manager.print_error_custom_message("Server
    connection timeout")
309 
310 
311             def get_target_temp(self):
312                 return self.target_temp
313 
314 
315             def get_current_temp(self):
316                 return self.curr_temp
317 
318 
319             def get_period_remaining(self):
320                 time_remaining = {}
321                 if not self.status:
322                     time_remaining = {
323                         'hour': '--',
324                         'minute': '--',
325                         'second': '--'
326                     }
327                 elif self.transient:
328                     time_remaining = {
329                         'hour': int(self.curr_step['total_time'] / 3600),
330                         'minute': int(self.curr_step['total_time'] / 60),
331                         'second': int(self.curr_step['total_time'] % 60)
332                     }
333 
334                 else:
335                     time_seconds = self.curr_step['total_time'] - (time.time() -
    self.initial_time)
336                     time_remaining = {
337                         'hour': int(time_seconds / 3600),
338                         'minute': int(time_seconds / 60),
339                         'second': int(time_seconds % 60)
340                     }
341 
```

```
342                 return time_remaining
343
344
345             def display_routine(self, temp, minute, hour):
346                 self.r_text_browser.append(f"{len(self.routine)}. Temperature: {
    temp} C\tDuration: {hour:02d}:{minute:02d}")
347
348
349             def update_routine(self):
350                 # Verify if it's in temperature transition
351                 if self.transient:
352                     if self.curr_temp >= self.curr_step['temperature'] - self.
    lower_limit:
353                         self.transient = False
354                         self.initial_time = time.time()
355                         self.routine_data.register_step_transition_end()
356
357                 # Verify if it's step temperature needs to be updated
358                 elif self.curr_step['total_time'] <= (time.time() - self.
    initial_time):
359                     print('here')
360                     if self.curr_step != self.last_step:
361                         self.routine_data.register_step_end()
362                         self.curr_step = next(self.routine)
363                         print(self.curr_step)
364                         self.routine_data.register_step_start()
365                         self.transient = True
366
367                     else:
368                         self.routine_data.register_step_end()
369                         self.end_routine()
370                         self.curr_step = None
371
372
373             def end_routine(self):
374                 if self.status:
375                     self.status = False
376                     self.climatic_chamber.set_target_temp(0, 3)
377                     self.exporter.export(
378                         routine = self.routine_data.routine,
379                         time = self.routine_data.time,
380                         curr_temps = self.routine_data.curr_temp,
381                         target_temps = self.routine_data.target_temp
382                         )
383                     self.routine = []
384                     self.r_text_browser.clear()
385
```

```
386
387             def connect(self):
388                 try:
389                     self.climatic_chamber = ClimaticChamber(self.ip, self.port)
390                 except ValueError:
391                     self.routine_error_manager.print_invalid_ip()
392                     return False
393
394                 if not self.climatic_chamber.is_connected():
395                     self.routine_error_manager.print_connection_error()
396                     return False
397
398                 else:
399                     return True
400
401
402             def add_routine(self, temp, minute, hour):
403                 self.display_routine(temp, minute, hour)
404                 self.routine.append(
405                     {
406                         "hour": hour,
407                         "minute": minute,
408                         "temperature": temp,
409                         "total_time": (minute*60 + hour*3600)
410                     }
411                 )
412
413
414             def add_clicked(self):
415                 temp = self.r_temp_spinbox.value()
416                 time = self.r_period_time.time()
417                 minute = time.minute()
418                 hour = time.hour()
419
420                 self.add_routine(temp, minute, hour)
421
422
423             def next_clicked(self):
424                 self.ip = self.r_text_ip.toPlainText()
425
426                 if self.connect():
427
428                     self.initial_time = time.time()
429                     self.status = True
430                     self.transient = True
431                     self.last_step = self.routine[-1]
432
```

```
433                     self.routine_data.add_routine(self.routine)
434                     self.routine_data.register_step_start()
435
436                     self.routine = iter(self.routine)
437                     self.curr_step = next(self.routine)
438                     self.close()
439
440
441             def cancel_clicked(self):
442                 self.close()
443                 self.routine = []
444                 self.status = False
445
446
447             def save_clicked(self):
448                 ip = self.r_text_ip.toPlainText()
449
450                 if len(self.routine) == 0:
451                     self.routine_error_manager.print_error_custom_message(
452                         "You must add some temperature and its duration in order to
     save the routine."
453                     )
454                 elif not ip:
455                     self.routine_error_manager.print_error_custom_message(
456                         "You must set the server IP address in order to save the
    routine."
457                     )
458                 else:
459                     self.file_manager.save_routine_file(self.routine, ip)
460
461
462             def plot_export_clicked(self):
463                 if self.r_plot_checkbox.isChecked():
464                     self.exporter.enable_export_plot()
465                 else:
466                     self.exporter.disable_export_plot()
467
468
469             def report_export_clicked(self):
470                 if self.r_report_checkbox.isChecked():
471                     self.exporter.enable_export_report()
472
473                 else:
474                     self.exporter.disable_export_report()
475
476
477             def configure_dir_clicked(self):
```

```
478                 export_path = self.file_manager.get_directory("Select export files
    directory")
479                 self.exporter.configure_path(export_path)
480
```

APÊNDICE D – CÓDIGO DA CLASSE RESPONSÁVEL PELA EXPORTAÇÃO DOS DADOS DAS ROTINAS

```
481         import matplotlib.pyplot as plt
482         from matplotlib.backends.backend_pdf import PdfPages
483         import pandas as pd
484         from pretty_html_table import build_table
485
486
487         class RoutineExporter:
488             def __init__(self):
489                 self.plot_file_name = "routine_plot.pdf"
490                 self.report_file_name = "routine_report.html"
491                 self.data_file_name ="routine_data.csv"
492                 self.export_path = "."
493                 self.plot_status = False
494                 self.report_status = False
495
496
497             def configure_path(self, path):
498                 if len(path) > 0:
499                     self.export_path = path
500
501
502             def enable_export_report(self):
503                 self.report_status = True
504
505
506             def disable_export_report(self):
507                 self.report_status = True
508
509
510             def enable_export_plot(self):
511                 self.plot_status = True
512
513
514             def disable_export_plot(self):
515                 self.plot_status = False
516
517
518             def is_activated(self):
519                 if self.plot_status or self.report_status:
520                     return True
521                 else:
522                     return False
523
```

```
524
525             def export(self, routine, time, curr_temps, target_temps):
526                 self.export_temp_plot(time, curr_temps, target_temps)
527                 self.export_report(routine, time, curr_temps, target_temps)
528
529
530             def export_temp_plot(self, time, curr_temps, target_temps):
531                 if self.plot_status:
532                     fig = plt.figure()
533                     plt.plot(time, curr_temps, label="Current temperature")
534                     plt.plot(time, target_temps, alpha=.5, label="Target
    temperature")
535                     plt.legend(loc="upper left")
536                     plt.xlabel("Time")
537                     plt.ylabel("Temperature (C)", rotation='vertical')
538                     plt.ylim([20, 100])
539                     plt.grid()
540                     pp = PdfPages(f"{self.export_path}/{self.plot_file_name}")
541                     pp.savefig(fig)
542                     pp.close()
543
544
545             def export_report(self, routine, time, curr_temps, target_temps):
546                 if self.report_status:
547                     data = {
548                         "Target temperature (C)": [],
549                         "Start time": [],
550                         "End time": [],
551                         "Duration (s)": [],
552                         "Transition time (s)": []
553                     }
554
555                     for i in routine:
556                         data["Target temperature (C)"].append(i["temperature"])
557                         data["Start time"].append(i["start_time"])
558                         data["End time"].append(i["end_time"])
559                         data["Transition time (s)"].append(round(i["transition_time
    "], 1))
560                         data["Duration (s)"].append(round(i["duration"], 1))
561
562                     df = pd.DataFrame(data)
563                     html_pretty = build_table(df, color='green_dark')
564                     f = open(f"{self.export_path}/{self.report_file_name}", "w")
565                     f.write(html_pretty)
566                     f.close()
```

APÊNDICE E – CÓDIGO DO SERVIDOR HTTP

```
567         #include "chamber_server.h"
568
569
570         static const char *TAG = "http_server";
571
572         static esp_err_t actual_get_handler(httpd_req_t *req)
573         {
574             char actual_temp[7];
575
576             sensor_read_t actual = get_actual_temp();
577
578             sprintf(actual_temp, "%u.%u", actual.integer, actual.decimal);
579
580             httpd_resp_send(req, (const char*) actual_temp, HTTPD_RESP_USE_STRLEN);
581
582             return ESP_OK;
583         }
584
585         static esp_err_t target_get_handler(httpd_req_t *req)
586         {
587             char target_temp[7];
588
589             sensor_read_t target = get_target_temp();
590
591             sprintf(target_temp, "%u.%u", target.integer, target.decimal);
592
593             httpd_resp_send(req, (const char*) target_temp, HTTPD_RESP_USE_STRLEN);
594
595             return ESP_OK;
596         }
597
598         static esp_err_t target_post_handler(httpd_req_t *req)
599         {
600             char content[3];
601             sensor_read_t value = {
602                 .integer = 0,
603                 .decimal = 0,
604                 .raw_value = 0
605             };
606
607             /* Truncate if content length larger than the buffer */
608             size_t recv_size = MIN(req->content_len, sizeof(content));
609
610             int ret = httpd_req_recv(req, content, recv_size);
```

```
611
612             if (ret <= 0) {
613                 if (ret == HTTPD_SOCK_ERR_TIMEOUT) {
614
615                     httpd_resp_send_408(req);
616                 }
617
618                 return ESP_FAIL;
619             }
620
621             uint32_t temp = atoi(content);
622             value.integer = temp;
623             esp_err_t res = set_target_temp(value);
624
625             /* Log data received */
626             ESP_LOGI(TAG, "Updating target temperature to %s", content);
627
628             if (res != ESP_OK){
629                 const char resp[] = "Failed";
630                 httpd_resp_send(req, resp, HTTPD_RESP_USE_STRLEN);
631             }
632             else {
633                 const char resp[] = "Updated";
634                 httpd_resp_send(req, resp, HTTPD_RESP_USE_STRLEN);
635             }
636
637             return ESP_OK;
638         }
639
640         static const httpd_uri_t actual = {
641             .uri       = "/actual",
642             .method    = HTTP_GET,
643             .handler   = actual_get_handler,
644             .user_ctx  = NULL
645         };
646
647         static const httpd_uri_t target_get = {
648             .uri       = "/target",
649             .method    = HTTP_GET,
650             .handler   = target_get_handler,
651             .user_ctx  = NULL
652         };
653
654         static const httpd_uri_t target_post = {
655             .uri       = "/target",
656             .method    = HTTP_POST,
657             .handler   = target_post_handler,
```

```
658             .user_ctx  = NULL
659         };
660
661         static httpd_handle_t start_webserver(void)
662         {
663             httpd_handle_t server = NULL;
664             httpd_config_t config = HTTPD_DEFAULT_CONFIG();
665
666             // Start the httpd server
667             ESP_LOGI(TAG, "Starting server on port: '%d'", config.server_port);
668             if (httpd_start(&server, &config) == ESP_OK) {
669                 // Set URI handlers
670                 ESP_LOGI(TAG, "Registering URI handlers");
671                 httpd_register_uri_handler(server, &actual);
672                 httpd_register_uri_handler(server, &target_get);
673                 httpd_register_uri_handler(server, &target_post);
674                 return server;
675             }
676
677             ESP_LOGI(TAG, "Error starting server!");
678             return NULL;
679         }
680
681         static void stop_webserver(httpd_handle_t server)
682         {
683             // Stop the httpd server
684             httpd_stop(server);
685         }
686
687         static void disconnect_handler(void* arg, esp_event_base_t event_base,
688                                        int32_t event_id, void* event_data)
689         {
690             httpd_handle_t* server = (httpd_handle_t*) arg;
691             if (*server) {
692                 ESP_LOGI(TAG, "Stopping webserver");
693                 stop_webserver(*server);
694                 *server = NULL;
695             }
696         }
697
698         static void connect_handler(void* arg, esp_event_base_t event_base,
699                                     int32_t event_id, void* event_data)
700         {
701             httpd_handle_t* server = (httpd_handle_t*) arg;
702             if (*server == NULL) {
703                 ESP_LOGI(TAG, "Starting webserver");
704                 *server = start_webserver();
```

```
705             }
706         }
707
708         void http_server_start(void)
709         {
710             static httpd_handle_t server = NULL;
711
712             ESP_ERROR_CHECK(wifi_connect());
713             ESP_ERROR_CHECK(esp_event_handler_register(IP_EVENT,
    IP_EVENT_STA_GOT_IP, &connect_handler, &server));
714             ESP_ERROR_CHECK(esp_event_handler_register(WIFI_EVENT,
    WIFI_EVENT_STA_DISCONNECTED, &disconnect_handler, &server));
715
716             server = start_webserver();
717         }
718
```