UNIVERSIDADE FEDERAL DE VIÇOSA
CENTRO DE CIÊNCIAS EXATAS E TECNOLÓGICAS
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

DANIEL MARTINS MARQUES

SISTEMA DE CONTROLE E SUPERVISÃO DE PROCESSO TÉRMICO UTILIZANDO TELEMETRIA

VIÇOSA
2020

DANIEL MARTINS MARQUES

SISTEMA DE CONTROLE E SUPERVISÃO DE PROCESSO TÉRMICO UTILIZANDO TELEMETRIA

Monografia apresentada ao Departamento de Engenharia Elétrica do Centro de Ciências Exatas e Tecnológicas da Universidade Federal de Viçosa, para a obtenção dos créditos da disciplina ELT 402 – Projeto de Engenharia II – e cumprimento do requisito parcial para obtenção do grau de Bacharel em Engenharia Elétrica.

Orientadora: Profa. Dra. Kétia Soares Moreira.

VIÇOSA
2020

DANIEL MARTINS MARQUES

SISTEMA DE CONTROLE E SUPERVISÃO DE PROCESSO TÉRMICO UTILIZANDO TELEMETRIA

Monografia apresentada ao Departamento de Engenharia Elétrica do Centro de Ciências Exatas e Tecnológicas da Universidade Federal de Viçosa, para a obtenção dos créditos da disciplina ELT 402 – Projeto de Engenharia II e cumprimento do requisito parcial para obtenção do grau de Bacharel em Engenharia Elétrica.

Aprovada em 10 de novembro de 2020.

COMISSÃO EXAMINADORA

Profa. Dra. Kétia Soares Moreira - Orientadora
Universidade Federal de Viçosa

Prof. Dr. André Gomes Tôrres - Membro
Universidade Federal de Viçosa

Prof. Dr. Denilson Eduardo Rodrigues - Membro
Universidade Federal de Viçosa

“Os mais fortes de todos os guerreiros são estes dois: tempo e paciência.”

Liev Tolstói

Agradecimentos

Primeiramente, agradeço a minha família por ter me dado a condição privilegiada de estudar em uma instituição de ensino superior de excelência e todo conforto possível para vencer desafios pessoais e profissionais.

Aos meus colegas de curso, por termos compartilhado conhecimentos e experiências, além das amizades construídas ao longo destes anos.

À equipe UFVBaja, por ser parte relevante da minha construção profissional, além de ter me oferecido conhecimentos técnicos fundamentais para a execução deste trabalho.

À professora Kétia, por sempre ter se apresentado disposta a ajudar nestes anos de monitoria das disciplinas de eletrônica e neste trabalho, além do grande trabalho como orientadora.

Aos professores e técnicos do DEL, pela contribuição no meu desenvolvimento e aprendizado, em especial ao professor Rodolpho pelo apoio, paciência e carinho.

Resumo

Os sistemas de telemetria são amplamente utilizados para o monitoramento de variáveis à distância, se apropriando de diversas vantagens para tornar processos mais robustos e mais produtivos. Esta aplicação já é consolidada na indústria, e vêm continuamente ganhando força ao longo dos anos, para manter a produção mais competitiva em relação ao mercado.

Este trabalho consiste em confeccionar um sistema de telemetria dotado de interface capaz de monitorar e armazenar as variáveis de um processo térmico genérico. Para simular um processo térmico, confeccionou-se um *software* de forma que este e um sistema supervisório programados sejam capazes de comunicar entre si através de uma comunicação serial com um emulador de portas seriais virtuais.

Ambos os *softwares* foram confeccionados na linguagem C# através da IDE *Microsoft Visual Studio 2017*. Para que a comunicação entre os dois programas fosse possível, utilizou-se o programa *Virtual Series Ports Emulator* da *Eterlogic*.

Palavras-chaves: Sistema supervisório, Telemetria, Instrumentação, Comunicação serial, C#, *Microsoft Visual Studio*.

Abstract

Telemetry systems are widely used to monitor variables from a distance. It takes advantages of several aspects in order to improve processes and turn them more robust, therefore satisfying the industry needs to keep its production competitive in relation to the market. This application is already consolidated in the industry, and has been steadily gaining strength over the years.

This work consists of making a telemetry system with an interface capable of monitoring and storing the variables of a generic thermal process. To simulate a thermal process, a software was built in order to interact with the programmed supervisory system using a serial communication with a virtual series ports emulator.

Both softwares were built in the C# language through the *Microsoft Visual Studio 2017* IDE. The *Eterlogic Virtual Series Ports Emulator* program was used in order to these two programs communicate themselves.

Keywords: Supervisory system, Telemetry, Instrumentation, Serial communication, C#, *Microsoft Visual Studio*.

Sumário

Lista de Figuras

Lista de Tabelas

1 Introdução

A telemetria é uma técnica capaz de transportar medições captadas em um processo à distância em função de um dispositivo transmissor. As vantagens e a necessidade da utilização da telemetria e do contínuo processamento de variáveis em sistemas estão relacionados a transmissão à distância destas medições. Consequentemente, é possível destacar que, com a telemetria, é possível agrupar as informações de modo que seja mais fácil e rápida a consulta de instrumentos, possibilitando uma visão conjunta do desempenho de um sistema. Além disso, é possível reduzir o número de operadores em uma unidade em função do aumento de eficiência da mão de obra [1].

A partir da aplicação de sistemas supervisórios, é possível armazenar e analisar dados em um sistema através da telemetria. Na indústria automobilística, esta é frequentemente utilizada no desenvolvimento de veículos e em carros de corrida. Existem diversos sensores captando variáveis das quais são suplementadas pelos dados gerados pelo *software* instalado em unidades de processamento do veículo [2].

O uso da telemetria pode ser útil para muitas aplicações, como é o caso de sistemas térmicos. Para o monitoramento remoto de sistemas de geração de energia elétrica, pode-se citar o caso das termoelétricas, onde utilizam o calor obtido da queima de combustíveis como fonte de energia [3]. Consequentemente, a quantidade apropriada de matéria prima utilizada no processo, bem como o monitoramento das variáveis, é relevante de modo que seja adequado o uso da telemetria.

Os processos térmicos são utilizados na indústria alimentícia com o intuito de reduzir a flora microbiológica presente em alimentos e evitar mudanças ocasionadas por microorganismos patogênicos, portanto a telemetria é amplamente utilizada na indústria para verificar a qualidade dos alimentos [4].

Na indústria automobilística, o controle de sistemas de refrigeração é essencial para o funcionamento de componentes mecânicos dos veículos, como, por exemplo, os radiadores que são controlados por uma ECU (*engine control unit*) através de informações obtidas de sensores e de comandos de atuadores. O ventilador do radiador geralmente é

conectado à um motor elétrico que tem como função resfriar a água do radiador de modo a arrefecer o motor [5].

Também são encontrados trabalhos relacionados à sistemas agrícolas, aonde o monitoramento climático é realizado em função da importância da temperatura do ar para o processo produtivo. Neste contexto, o uso das tecnologias de *software* é favorável para a aquisição e gerenciamento das variáveis coletadas, criando melhores fundamentos com o intuito de contribuir com tomadas de decisões [6].

1.1 Objetivos

Este projeto tem como objetivo geral desenvolver um sistema de baixo custo de supervisão de controle de um processo térmico, que seja capaz de monitorar e enviar comandos para um *software* simulador.

Desta forma, o trabalho apresenta os objetivos específicos de construir um simulador que seja capaz de representar um sistema de controle de refrigeração de processo térmico, desenvolver um sistema supervisório capaz monitorar, comandar e armazenar os dados da simulação do processo térmico e, por fim, através de um emulador de comunicação serial, deve-se realizar a transmissão de dados bidirecional entre os dois *softwares*.

1.2 Impactos do projeto

Caso este trabalho venha a ser aplicado em algum sistema de processo térmico, a utilização da telemetria pode gerar impactos econômicos, principalmente. Isto se deve ao fato de que um *software* como este é capaz de monitorar as variáveis de um processo à distância disponibilizando informações agrupadas. Assim, não dependendo de uma mão de obra responsável por realizar a leitura de medições no local do sistema. Além disso, a utilidade e a eficiência, em função da imediata consulta, crescem, pois, a telemetria torna a inspeção e a manutenção mais acessíveis, confortáveis e protegidas.

2 Materiais e Métodos

Primeiramente, foi necessário desenvolver um *software* capaz de replicar um processo térmico qualquer. A escolha da confecção de um simulador virtual ao invés de um físico se deve ao fato desse mesmo ser mais facilmente manuseável e apresentar um custo de desenvolvimento menor.

Além disso, colocou-se o objetivo de que fosse possível realizar a comunicação entre o processo térmico e o sistema supervisório através do protocolo de comunicação RS-232. Portanto, confeccionou-se o simulador de forma que fosse adaptável a módulos de rádiofrequência de comunicação serial, consequentemente o simulador apresenta *Baud Rate, Data Bits, Stop Bits* e Paridade configuráveis.

Em função da disponibilidade de ferramentas e por apresentar uma licença gratuita, optou-se por utilizar a IDE *Microsoft Visual Studio 2017*, através da linguagem C#. Este ambiente de desenvolvimento integrado apresenta uma série de recursos para código e design que favorecem a confecção destes trabalhos.

Para o sistema de processo térmico, assumiu-se que este necessite de um fator externo para ser arrefecido, consequentemente optou-se por atribuir o resfriamento deste por uma ventoinha, e não pelo próprio controle de aquecimento do sistema, uma vez que isso retrata diversos casos da realidade. Portanto, é necessário que a simulação apresente dois sensores, um de temperatura e outro de rotação. O sensor de temperatura é responsável por captar esta variável diretamente do processo térmico, e o de rotação é capaz de monitorar a velocidade angular (em rotações por minuto) do ventilador.

Logo, projetou-se o simulador de processo térmico assumindo que seja possível ligar ou desligar o processo térmico do sistema de forma independente do sistema de arrefecimento através de botões, vide Figura 1.

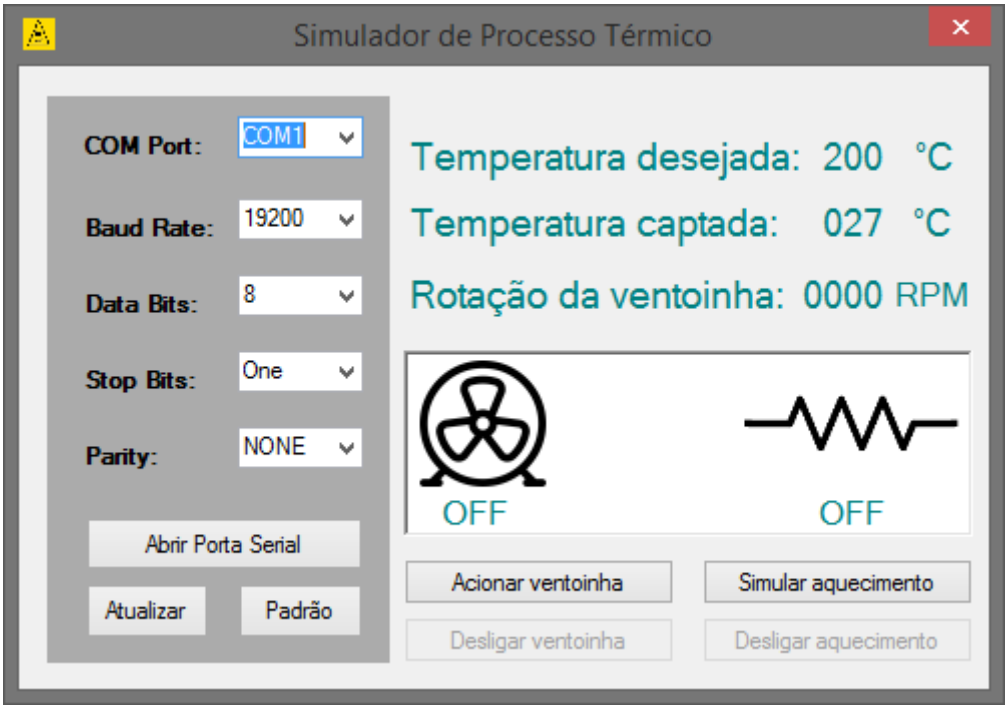

Figura 1. Software simulador do processo térmico.

Como pode-se observar, o simulador apresenta três variáveis na tela, sendo a temperatura desejada pela qual o sistema deve se manter, a temperatura captada e a rotação da ventoinha. Além de apresentar estas no sistema supervisório, optou-se por dispor estas variáveis no simulador para retratar o que acontece na realidade, como em um display LCD ou um monitor, por exemplo. Por este mesmo motivo, o acionamento dos sistemas do processo térmico e de arrefecimento podem ser realizados diretamente através de botões nesta interface, uma vez que é mais seguro garantir estes comandos presencialmente para o caso de falhas na comunicação entre as interfaces.

De forma a se manter fiel à realidade, decidiu-se que o controle do arrefecimento do sistema é de responsabilidade do processo térmico. Entretanto, definiu-se que o comando que estabelece a temperatura desejada no processo térmico se encontra no sistema supervisório, definindo-se então uma comunicação bidirecional entre os sistemas, ou seja, o processo térmico envia as variáveis captadas em tempo real para a telemetria, e recebe os valores desejadas de temperatura. Para manter o processo térmico robusto, definiu-se a temperatura desejada padrão de 200 ºC, para o caso de perda de conexão com a telemetria.

O sistema de arrefecimento, representado pelo ventilador no simulador, é habilitado se o botão “acionar ventoinha” for pressionado, e este será acionado caso a temperatura desejada seja inferior a temperatura real e desligado caso contrário. Consequentemente, é possível

considerar isto como um simples controlador ON/OFF. Optou-se por escolher este tipo de controle dada a sua simplicidade de programação, deixando para desenvolvimentos futuros o comando por meio de um controlador PID.

De forma a aplicar fórmulas simples que sejam fidedignas à realidade no código do simulador, traçou-se as curvas de aquecimento e arrefecimento do processo térmico, sem a ação do ventilador, de acordo com as Figuras 2 e 3, respectivamente.

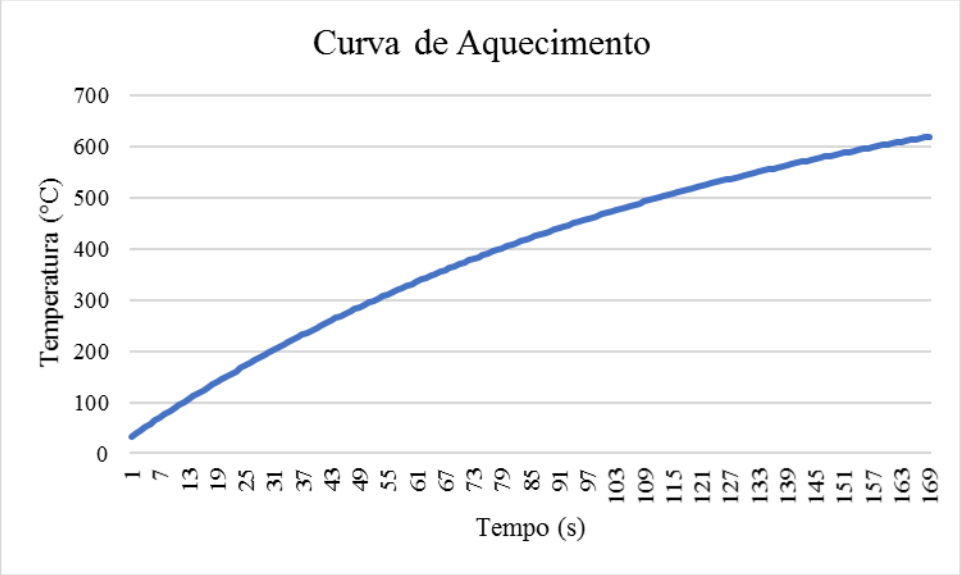

Figura 2. Curva de aquecimento do processo térmico sem ação do ventilador.

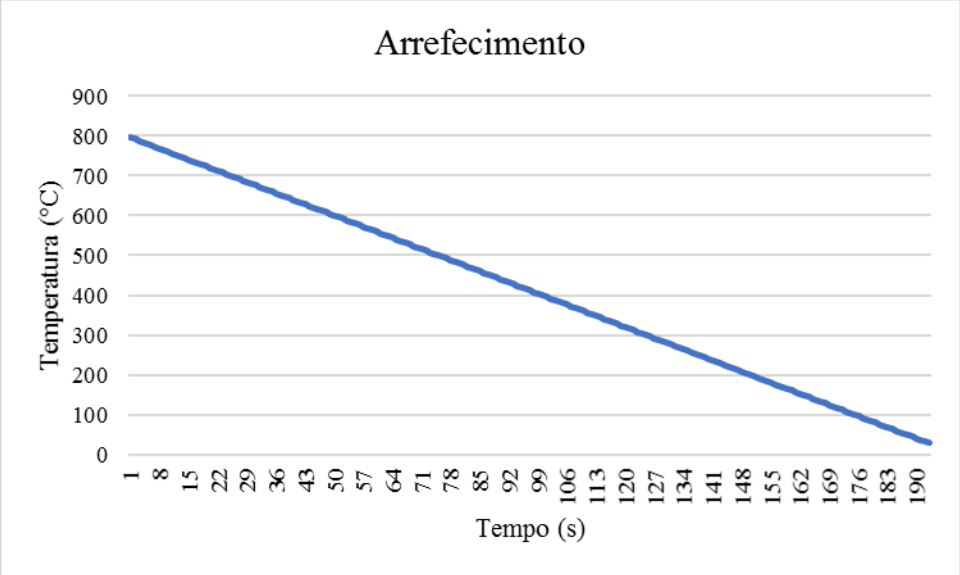

Figura 3. Curva de arrefecimento do processo térmico sem ação do ventilador.

Para programar estas duas curvas no simulador do processo térmico, utilizou-se as fórmulas (1) e (2) [7], para as curvas de aquecimento e arrefecimento respectivamente:

$$T = 800 * (1 - e^{-t/125}) + 27 \quad (1)$$

$$T = 800 - 4 * t \quad (2)$$

Onde "*t*" é o tempo em segundos e "*T*" é a temperatura do processo em ºC. A equação (1) foi confeccionada baseando-se na curva de resposta de um circuito RC com a aplicação súbita de uma fonte CC. Assumiu-se que a temperatura ambiente é de 27 ºC, o que explica a constante somada em (1). O fator de 800 que multiplica parte da função, foi determinado como o máximo valor alcançado pela temperatura obtida no processo térmico, e a constante 125 foi escolhida empiricamente para determinar a suavidade da curva de aquecimento. Estes valores foram determinados baseando-se nas temperaturas que componentes de motores à combustão automotivos possam suportar [8]. Já em (2), construiu-se uma função de primeiro grau assumindo o valor máximo da temperatura a ser captada no simulador, e uma taxa de arrefecimento de 4 °C por segundo.

Para confeccionar o código do simulador e utilizar estas fórmulas, utilizou-se temporizadores com tempo de estouro de 50 ms, ou seja, os temporizadores disparam 20 vezes em um intervalo de 1 segundo. Este tempo foi definido de forma à não tornar o programa pesado em termos de *hardware*, mas ao mesmo tempo apresentar uma resolução satisfatória para não comprometer a obtenção de resultados para o trabalho. Para que fosse possível desligar e imediatamente acionar o processo térmico mas sem que o sistema fosse totalmente arrefecido, aplicou-se no código uma variável auxiliar de forma que fosse utilizada como memória para não restaurar o valor da temperatura todas as vezes que o processo fosse interrompido, com o objetivo de simular casos reais.

Em função da utilização de temporizadores para construir as curvas de aquecimento e arrefecimento do processo térmico sem a utilização do controlador, foi aplicada uma lógica para a ação de resfriamento do controlador de arrefecimento e não simplesmente uma equação. Desta forma, as curvas mencionadas nas Figuras 2 e 3 param de ser perfeitamente descritas pelas equações (1) e (2), respectivamente, e passam a ser impactadas por mais um temporizador com intervalo de estouro a cada um segundo. Esta ação de resfriamento é diretamente proporcional à velocidade da rotação do ventilador que subtrai no módulo do valor da temperatura.

De forma a estabelecer a comunicação serial com o sistema supervisório, o simulador de processo térmico envia os dados à uma taxa de 10 informações por segundo. Cada informação é transmitida através de uma linha de caracteres que contém os valores da temperatura do processo e a rotação do ventilador captados e enviados através de um padrão de envio.

Os dados enviados pelo simulador são transmitidos na forma “ba0bv0t000r000&%$” sendo que os zeros correspondem aos valores das variáveis captadas. Caso a telemetria receba uma linha de caracteres fora deste padrão, a informação é completamente descartada.

Este padrão foi aplicado com o objetivo de ser fiel à aplicações reais, aonde a segurança de dados confidenciais é considerada para que não exista qualquer tipo de vazamento de informações. Uma vez que este não é o foco do trabalho, o padrão de envio de informações utilizado apresenta lógica simples, mas dada as condições de aplicações no mercado, optou-se por implementá-lo.

Para simular uma conexão *wireless* entre os *softwares*, utilizou-se o programa *Virtual Series Ports Emulator* da Eterlogic, que é capaz de emular de forma gratuita portas COM do computador utilizado. Desta forma, configurou-se este programa para criar um par de portas COM que comunicam entre si. Segue a interface do *software* na Figura 4.

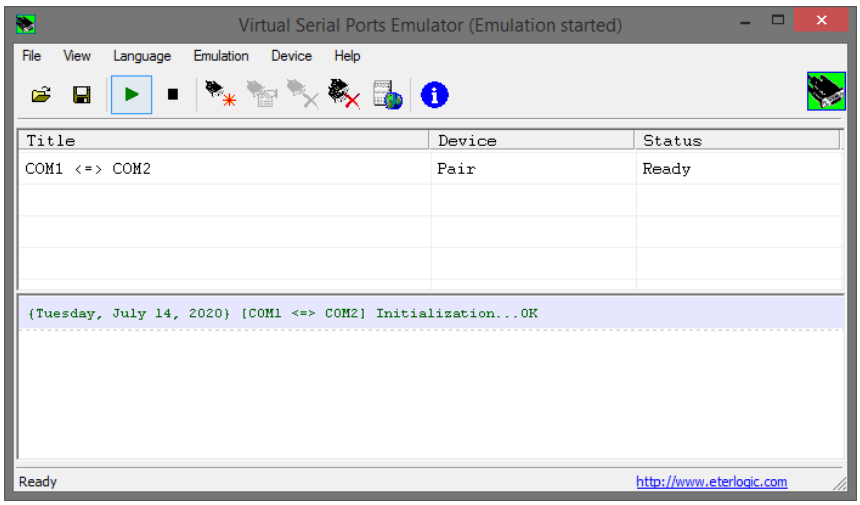

Figura 4. Interface do programa VSPE.

Como pode-se observar na figura, emulou-se as portas COM 1 e 2, sendo uma a ser aberta no simulador de processo térmico e a outra no sistema supervisório. Com isso, é possível que exista a comunicação serial entre os dois *softwares* de forma independente.

Segue na Figura 5 a interface da tela inicial do sistema supervisório.

Figura 5. Tela inicial do sistema supervisório.

No canto esquerdo, destacado com fundo cinza, segue a área das configurações da comunicação serial, aonde na parte superior se encontra a data e hora atuais, pouco abaixo são os campos de seleção das configurações, das quais são exatamente as mesmas do simulador do processo térmico, que são disponibilizados através de listas *dropdown*. Logo abaixo há um campo em branco do qual, no momento em que a abertura serial é realizada, é apresentado o nome do arquivo txt criado automaticamente para armazenar todo o histórico de dados recebidos e enviados durante a operação da telemetria. O nome gerado contém a data e horário atuais. Por fim, neste mesmo campo há o botão de abertura da porta serial, um botão para abrir o diretório do local que o arquivo txt foi armazenado, um terceiro botão responsável por atualizar as opções disponíveis nas listas *dropdown*, e o último botão responsável por colocar preencher as configurações para o padrão do software, que seriam:

Tabela 1. Lista das configurações padrão do sistema supervisório.

Campo	Valor
COM Port	COM2
Baud Rate	9600 bps
Data Bits	8
Stop Bits	One
Parity	None

As configurações padrão do simulador do processo térmico são as mesmas do sistema supervisório, com exceção da COM Port, pois não seria possível utilizar os dois softwares simultaneamente pela mesma porta. Essas configurações foram escolhidas como padrão em função de serem facilmente obtidas em módulos RF obtidos no mercado [9][10]. É possível observar na Figura 6 os campos das configurações da comunicação serial.

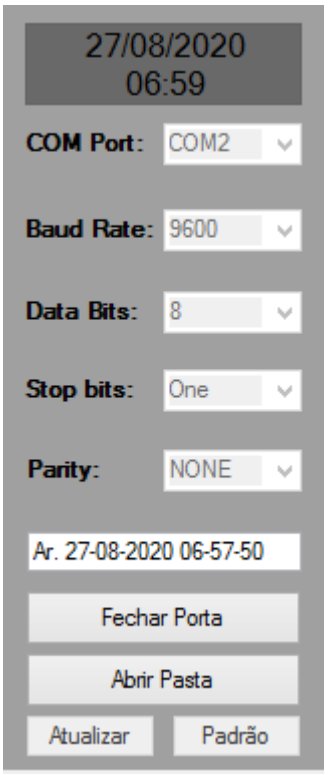

Figura 6. Configurações da comunicação serial no sistema supervisório.

Ainda na Figura 5, pode-se visualizar no canto inferior esquerdo da interface o monitor de alarmes (destacados na Figura 7), destacado em preto. Este monitor é responsável por piscar em amarelo quando há algum evento de alerta, e em vermelho para o caso de alguma emergência. Nestes dois casos, aparecem no monitor o horário da ocorrência do evento e o próprio evento. Para o caso de alarmes, irá aparecer um botão para ignorar o alerta e apagar os textos do monitor de alarme caso este seja um desejo do usuário. Em casos de emergência, os textos vão apagar somente se a emergência for corrigida, sendo que o campo “Emergências com correções pendentes” é atualizado no centro do supervisório. Ainda no monitor de alarmes, há um botão indicado pelo caractere “>” do qual é capaz de expandir o monitor e fornecer uma descrição completa do evento conjuntamente com uma sugestão de correção.

Figura 7. Tela do monitor de alarmes (a) em um caso de emergência, (b) em um caso de alerta e (c) com a descrição detalhada de uma emergência.

No canto inferior da interface do sistema supervisório, existe um campo com as guias "Medições" e "Lista de Eventos". O campo de medições apresenta, a partir do primeiro dado recebido de forma adequada, um gráfico que apresenta a temperatura desejada e a temperatura obtida no processo térmico, e outro gráfico que apresenta a velocidade do ventilador do sistema de arrefecimento. Ambos gráficos apresentam os vinte últimos dados recebidos, sendo isto equivalente aos dois últimos segundos, desta forma o dado mais antigo do gráfico é substituído pelo dado mais novo recebido, pelo modo FIFO (*First In, First Out*). É possível observar esta guia na Figura 8.

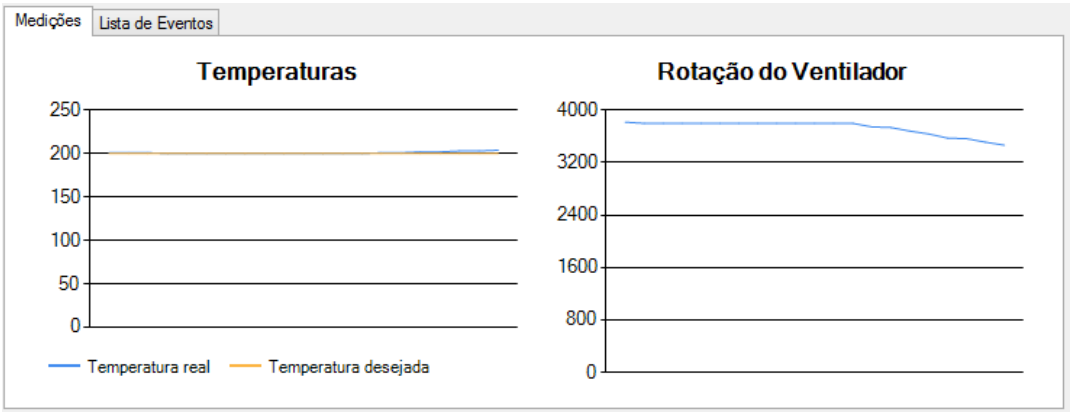

Figura 8. Gráficos das temperaturas e rotação do ventilador disponíveis na aba de medições.

A guia de lista de eventos apresenta todas as ocorrências durante o funcionamento do sistema supervisório. Estes dados são dispostos na forma de uma planilha com três colunas, sendo elas o horário do acontecimento do evento, uma pequena descrição do evento e a

gravidade deste, que pode ser classificada como normal, alerta ou emergência. Todos os eventos desta lista são disponibilizados no aquivo txt, mas apenas os alertas e as emergências aparecem no monitor de alarmes. Segue na Tabela 2 todos os possíveis eventos durante uma operação do sistema supervisório, e na Figura 9 a apresentação da lista de eventos destacada após um funcionamento aleatório da telemetria.

Tabela 2. Possíveis eventos durante o funcionamento do sistema supervisório.

Evento	Gravidade
Erro no recebimento de uma das informações	Alerta
Sistema de arrefecimento preparado com processo iniciado	Normal
Sistema de arrefecimento preparado	Normal
Sistema de arrefecimento desligado por completo	Alerta
Ventilador Girando	Normal
Ventilador parou de girar pois temperatura está equilibrada	Normal
Ventilador parou de girar pois sistema de arrefecimento está despreparado	Alerta
Processo térmico inciado	Normal
Sistema de arrefecimento despreparado e processo térmico incializado	Emergência
Processo térmico finalizado	Alerta
Temperatura obtida 10% superior à desejada	Emergência
Temperatura obtida volta ao normal	Normal
Abertura de porta serial	Normal
Falha na tentativa de abertura da porta serial	Alerta
Fechamento da porta serial	Alerta
Falha na tentativa do fechamento da porta serial	Alerta
Comando de alteração da temperatura desejada	Normal
Comando do acionamento do ventilador realizado	Normal
Comando de desligamento do ventilador realizado	Normal
Erro no envio do comando para o ventilador	Alerta
Comando de iniciar processo realizado	Normal
Comando de finalizar processo realizado	Normal
Erro no envio do comando para a inicialização/finalização do processo térmico	Alerta

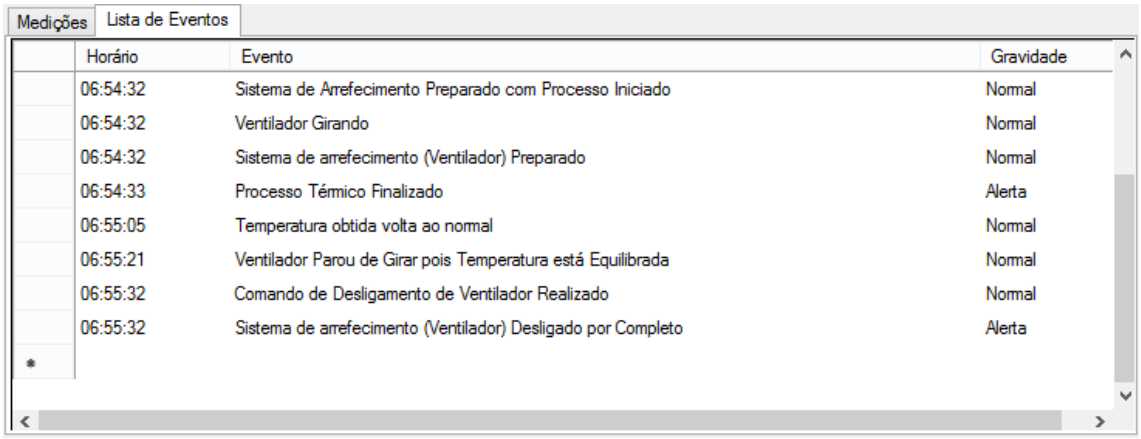

Figura 9. Lista de Eventos do supervisório durante a operação do simulador de processo térmico.

A Figura 10 mostra parte da Figura 5 destacada. Nesta é possível visualizar os dados das temperaturas obtida e desejada, a rotação do ventilador do sistema de arrefecimento e a quantidade de emergências com correções pendentes, sendo esta já mencionada anteriormente. No canto direito da Figura 10, verifica-se uma barra colorida indicadora da temperatura instantânea obtida no processo térmico. A barra tem seus valores alterados conforme a temperatura desejada, desta forma esta acende as luzes verdes para temperaturas obtidas superiores à 90% e 100% da desejada, a luz vermelha para valores superiores à 110% e as luzes azuis para valores inferiores à 90%.

No canto superior da mesma figura, é possível visualizar uma barra de rolagem que corresponde ao valor da temperatura desejada no processo térmico. Desta forma, o valor desejado pode ser ajustado pela preferência do usuário. Para enviar este comando, basta o usuário selecionar o valor de temperatura desejado e aguardar alguns instantes. Quando o comando for enviado, um painel verde irá indicar que a mensagem foi enviada com o valor selecionado pelo usuário. Esta lógica foi aplicada para que a quantidade de informações enviadas do sistema supervisório para o simulador seja reduzida e também para dar a possibilidade do usuário confirmar a sua requisição.

Ainda na Figura 10, existem outros quatro painéis coloridos além da barra colorida. O painel destacado na cor verde com o texto “Mensagem OK” indica o correto recebimento dos últimos dados enviados pelo simulador. Caso a mensagem seja incorreta, este painel ficará vermelho. Os painéis verde e azul, próximos um do outro, ficam alternando as suas cores entre azul-verde e verde-azul a cada recebimento dos dados correto. Por fim, o outro painel é responsável por indicar o envio correto do valor da temperatura desejada do sistema

supervisório para o processo térmico, assim como mencionado anteriormente. Caso exista algum erro na transmissão, o painel ficará vermelho e a mensagem de erro será apresentada.

Figura 10. Display de informações instantâneas do supervisório e barra de ajuste da temperatura desejada.

Através da Figura 11, é possível visualizar a área de envio dos comandos de acionamento do sistema supervisório. Desta forma, o usuário é capaz de ligar/desligar remotamente o ventilador do sistema de arrefecimento e ligar/desligar o processo térmico através do botões da figura. Além disso, o texto em cima dos botões, que na figura estão na forma “ON”, indicam se os sistemas estão ligados ou desligados. É importante considerar também que, para acionar remotamente tanto o processo térmico quanto o sistema de arrefecimento, o programa solicita uma confirmação para o usuário de forma a evitar falhas humanas. Quanto às figuras, caso o ventilador apresente um valor de rotação maior que zero, uma imagem indicando um vento irá aparecer, analogamente se a temperatura do processo térmico for maior que a temperatura ambiente, a imagem da resistência irá mudar a cor de preto para vermelho. É possível comparar estas imagens entre as figuras 5 e 11.

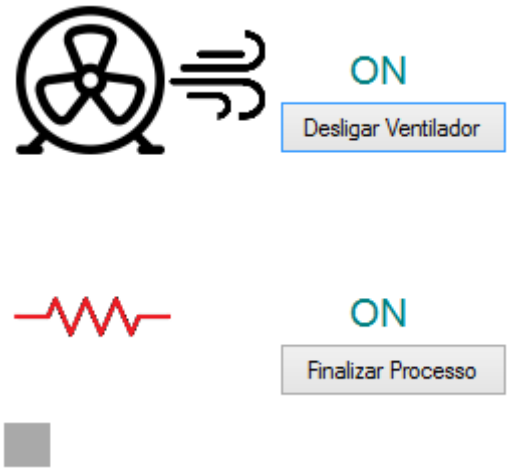

Figura 11. Comandos do sistema supervisório relacionados ao processo térmico simulado.

Após qualquer operação do superivsório, ou seja, quando a sua porta serial é aberta e posteriormente fechada, é salvo na máquina um arquivo txt do qual apresenta todos os dados coletados durante o funcionamento da telemetria. Este arquivo apresenta os valores das temperaturas obtida e desejada, a rotação do ventilador do sistema de arrefecimento, todos os eventos ocorridos e as suas devidas gravidades, e a data e hora do armazenamento destas informações. Este aquivo txt foi confeccionado pensando na simplicidade em visualizar ele assim que for aberto, mas também pensando na facilidade de manupulá-lo em alguma ferramenta de planilhas, como o *Microsoft Excel*. Segue na Figura 12 o arquivo txt após uma operação aleatória do sistema supervisório.

```
Data e Hora         Temperatura[°C]   Temperatura Desejada[°C]   Rotação da ventoinha[RPM]   Eventos, Alarmes e Gravidade
27/08/2020 06:52:48                                                                       Abertura de Porta Serial/Normal
27/08/2020 06:52:51        027                200                        0000
27/08/2020 06:52:51        027                200                        0000
27/08/2020 06:52:51        027                200                        0000
27/08/2020 06:52:51        027                200                        0000
27/08/2020 06:52:51        027                200                        0000
27/08/2020 06:52:51        027                200                        0000
27/08/2020 06:52:51        027                200                        0000
27/08/2020 06:52:52        027                200                        0000
27/08/2020 06:52:52        027                200                        0000
27/08/2020 06:52:52        027                200                        0000
27/08/2020 06:52:52        027                200                        0000
27/08/2020 06:52:52        027                200                        0000
27/08/2020 06:52:52        027                200                        0000
27/08/2020 06:52:52        027                200                        0000
27/08/2020 06:52:52        027                200                        0000
```

Figura 12. Arquivo txt gerado durante uma operação do sistema supervisório.

Como a comunicação entre o processo térmico e o sistema supervisório foi emulada de forma virtual, esta não foi impactada pela ação de ruídos externos. Mesmo assim, dadas as

condições da realidade, foi preciso tratar os dados recebidos de forma que os dados incorretos sejam descartados. Assim como mencionado anteriormente, o sistema supervisório recebe os dados através do padrão “ba0bv0t000r000&%$” por questões de segurança. Os dois primeiros zeros (após os caracteres “ba” e “bv”) estão associados aos acionamentos do processo térmico e do sistema de arrefecimento, respectivamente. Os zeros após o caractere “t” correspondem ao valor de temperatura obtido e após o “r” são referentes à rotação do ventilador.

A telemetria envia dois tipos de dados para o processo térmico, sendo estes “ca0cv0&%$” e “t000&%$”, também padronizados por questões de segurança. O primeiro dado está relacionado aos comandos de acionamento do processo térmico e do sistema de arrefecimento, que funciona de forma análoga aos caracteres “ba0bv0” recebidos pelo sistema supervisório, conforme foram previamente citados. O segundo tipo de envio de dados está relacionado ao envio do valor de temperatura desejada para o processo térmico.

Para avaliar o resultado da aplicação do sistema supervisório sobre o processo térmico, realizou-se seis simulações variando-se a temperatura desejada do processo térmico em passos de 100 °C partindo-se de 200 °C até 700 °C. Assim, analisou-se a disposição das informações na telemetria, tanto nos gráficos dispostos, quanto no banco de dados no arquivo txt, e verificou-se se as informações foram dispostas de forma que o objetivo deste sistema supervisório foi atingido. Também foi analisado o histórico de falhas do software ao longo das simulações a partir de intervalos de tempo previamente definidos.

3 Resultados e Discussão

Os valores captados pela telemetria em função da simulação do processo térmico estão dispostos na Tabela 3.

Tabela 3. Valores coletados durante as execuções do processo térmico simulado e do sistema supervisório.

Temperatura desejada (ºC)	Temperatura captada média (ºC)	Desvio padrão (ºC)	Erro máximo (%)
200	201,5738	±1,4544	2,50
300	301,5311	±1,5054	1,67
400	401,3715	±1,3202	1,00
500	501,2423	±1,3068	0,80
600	601,4508	±1,5094	0,83
700	701,2667	±1,3599	0,57

Esta tabela foi construída utilizando os dados disponibilizados no arquivo txt. Além disso, a telemetria foi capaz de transmitir em tempo real todas as variáveis do processo térmico simulado (Figura 13). O maior responsável pelos erros e o desvio padrão de cada simulação, como podem ser visualizados na mesma tabela, ocorreram em função do sistema de controle do simulador de processo térmico, e não do sistema supervisório.

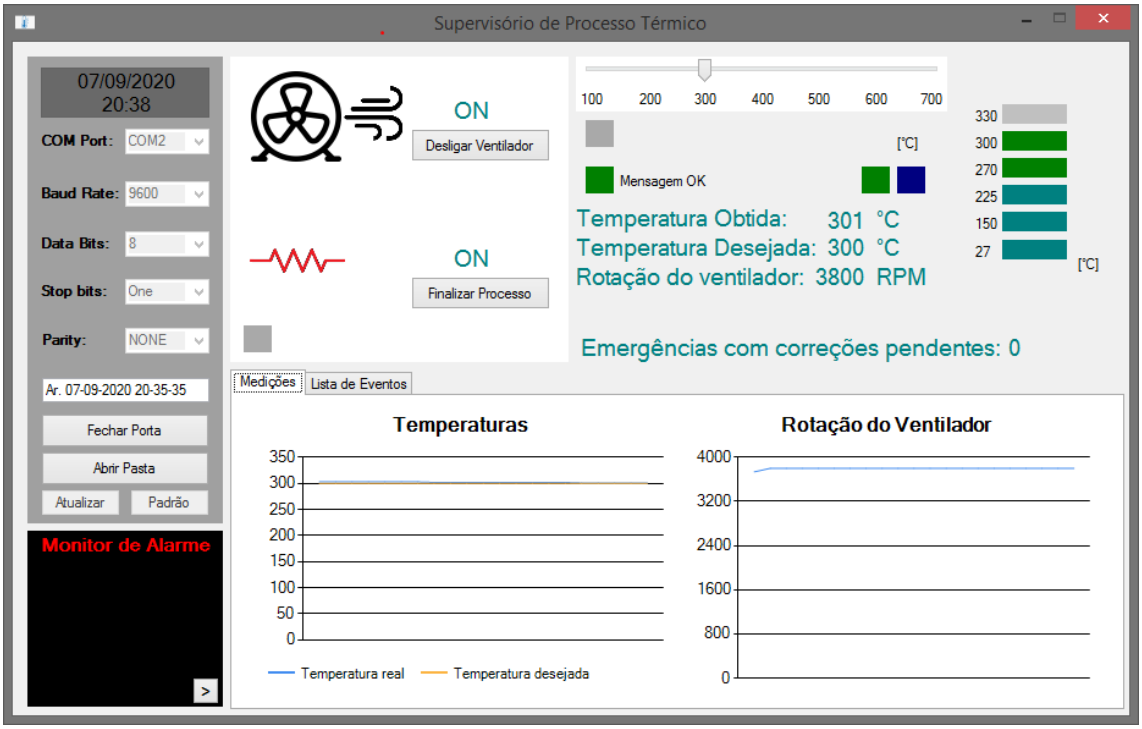

Figura 13. Sistema supervisório operando durante as simulações (temperatura desejada ajustada para 300 °C).

Além dos dados coletados de forma adequada, a lista de eventos apresentou todas as ocorrências corretamente, tanto no arquivo txt quanto na aba dedicada do sistema supervisório, como exemplificado na Figura 14.

Medições | Lista de Eventos

Horário	Evento	Gravidade
20:35:35	Abertura de Porta Serial	Normal
20:35:39	Processo Térmico Iniciado	Normal
20:35:39	Sistema de Arrefecimento Despreparado e Processo Térmico Inicializado	Emergência
20:35:43	Sistema de Arrefecimento Preparado com Processo Iniciado	Normal
20:35:43	Sistema de arrefecimento (Ventilador) Preparado	Normal

(a)

Medições | Lista de Eventos

Horário	Evento	Gravidade
20:42:19	Ventilador Parou de Girar pois Temperatura está Equilibrada	Normal
20:42:32	Ventilador Girando	Normal
20:43:53	Comando de Alteração de Temperatura Desejada (700 °C)	Normal
20:44:03	Ventilador Parou de Girar pois Temperatura está Equilibrada	Normal
20:44:16	Ventilador Girando	Normal
20:45:41	Comando de Finalizar Processo Realizado	Normal
20:45:41	Processo Térmico Finalizado	Alerta
20:45:48	Sistema de arrefecimento (Ventilador) Desligado por Completo	Alerta
20:45:52	Ventilador Parou de Girar pois Sistema de Arrefecimento está Despreparado	Alerta
20:46:14	Fechamento da Porta Serial	Alerta

(b)

Figura 14. Lista de eventos no (a) início da simulação e (b) final da simulação.

Junto com a lista de eventos, o monitor de alarmes também funcionou corretamente durante as simulações, como pode ser visualizado na Figura 15 o alarme do fechamento da porta serial.

Figura 15. Monitor de alarmes no término das simulações (a) com visualização simples e (b) visualização expandida.

Os botões de comando, tanto o de ajuste de temperatura desejada, quanto os de acionamento do sistema de arrefecimento e de inicialização do processo térmico funcionaram corretamente. Os botões de configuração da comunicação serial, de forma geral, têm resultado inconclusivo uma vez que o real teste destas opções ocorreria somente no caso de uma aplicação real, e não simulada. Mesmo assim, o ajuste da porta COM da máquina utilizada, por exemplo, é determinante para a correta operação do *software*.

Durante as simulações, o sistema supervisório apresentou duas falhas. Uma delas foi a apresentação dos painéis intermitentes verde e azul, que são responsáveis por indicar o recebimento de uma nova informação. Durante certos momentos não era visual esta alternância entre cores destes componentes gráficos. A outra falha foi em relação à apresentação do indicador de emergências com correções pendentes, que funcionou corretamente para todos os eventos ocorridos com exceção de um caso, aonde o contador foi de zero para um quando desligou-se o sistema de arrefecimento durante a simulação com o processo térmico inicializado, mas que não retornou a zero quando o problema foi corrigido. Além disso, o *software* não tem nenhuma forma de ajustar esse contador manualmente.

4 Conclusões

A partir deste trabalho, foi possível construir um sistema supervisório de baixo custo capaz de monitorar em tempo real e armazenar as variáveis de um processo térmico simulado. Além disso, a telemetria também é capaz de realizar comandos dedicados ao processo de remotamente de forma adequada.

Foram realizadas simulações para verificar a correta operação de ambos *softwares* e percebeu-se a existência de duas falhas. Uma delas está relacionada a uma ferramenta gráfica responsável por apresentar o recebimento de novos dados do processo térmico. Para identificar a causa e corrigir este problema, é válida a ideia de realizar uma aplicação real deste sistema supervisório para melhorar a disposição de informações para o usuário.

A outra falha do sistema supervisório foi um caso aonde o indicador de emergências com correções pendentes não voltava para zero quando algum problema no processo térmico era corrigido. Para este caso em específico, deve-se identificar e corrigir este problema, além de ser plausível a ideia de incluir um botão capaz de ajustar manualmente este indicador, conforme a vontade do usuário.

Apesar das falhas, constata-se que o sistema supervisório confeccionado está apto a realizar uma aplicação real dentro do seu escopo de operação, uma vez que ele foi capaz de realizar as tarefas incumbidas a este, e os problemas ocasionados apresentaram apenas sintomas gráficos, o que não impossibilita o seu funcionamento e dá a liberdade de corrigir estes casos. Portanto, conclui-se que os objetivos deste projeto foram alcançados.

Referências Bibliográficas

[1] GONÇALVES, M. G. Monitoramento e Controle de Processos. SENAI, Rio de Janeiro – Petrobrás. DN, 2003.

[2] "Telemetria" Magneti Marelli, 01 de dezembro de 2019 [Online]. Disponível em: "https://www.magnetimarelli.com/pt/business_areas/motorsport/excel%C3%AAncias-tecnol%C3%B3gicas/telemetria" [Acesso em 01 dezembro 2019].

[3] PAGOTTI, L. F. "A dimensão Técnica e Ambiental da Produção de Energia". PUC Goiás – Pontifícia Universidade Católica de Goiás – Departamento de Engenharia.

[4] "Operação do Processamento Térmico em Alimentos" Alimentos Tecnologia e Operações, 02 de dezembro de 2019. [Online]. Disponível em: "http://abgtecalim.yolasite.com/resources/Processamento%20T%C3%A9rmico%20e%20Trocadores%20de%20Calor" [Acesso em 02 dezembro 2019].

[5] "Eletroventilador do radiador" APS Distribuidora, 01 de dezembro de 2019 [Online]. Disponível em: "http://www.apsdistribuidora.com.br/noticias/Saiba-tudo-sobre-o-eletroventilador-do-radiador" [Acesso em 01 dezembro 2019].

[6] BARBOSA, R.Z.[1] PEREA MARTINS, J. E. M.[2] "Desenvolvimento de um Sistema de Telemetria para Monitoramento Térmico em Casas de Vegetação". 1.Faculdade de Ciências Agronômicas, UNESP - Univ Estadual Paulista, Campus de, Botucatu, SP, Brasil. 2.Faculdade de Ciências, Departamento de Computação, UNESP - Univ Estadual Paulista, Campus de Bauru, SP, Brasil, 2014.

[7] SADIKU, Matthew N. O., ALEXANDER, Charles K. Fundamentos de Circuitos Elétricos. McGrawHill, Bookman. Tradução: José Lucimar do Nascimento; revisão técnica: Antônio Pertence Júnior. 5. Ed. AMGH Editora Ltda, Porto Alegre, 2013.

[8] VARELLA, C. A. A., SANTOS, G. S. "Noções Básicas de Motores Diesel". UFRRJ – Universidade Federal Rural do Rio de Janeiro. Seropédica – RJ, julho de 2010.

[9] "HC-12 Wireless Serial Port Communication Module User Manual Product V1.18" [Online]. Disponível em: "https://www.elecrow.com/download/HC-12.pdf" [Acesso em 02 de setembro de 2020].

[10] “XBee/XBee-PRO ZB RF Modules User Guide” [Online]. Disponível em: “https://www.digi.com/resources/documentation/digidocs/PDFs/90000976.pdf” [Acesso em 02 de setembro de 2020].

Apêndice A – Código do sistema supervisório

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.IO;
using System.IO.Ports;
using System.Threading;
using System.Diagnostics;


namespace Supervisório_de_Processo_Térmico
{
   public partial class Form1 : Form
   {
      public delegate void Fdelegate(string a);

      #region Declarar variáveis
      private DirectoryInfo pasta;
      private StreamWriter arquivo;
      string diretorio = (@"C:\\Arquivos Supervisório TCC\\" + DateTime.Now.ToString("dd-MM-yyyy"));
      string recebe = string.Empty;
      string msg = string.Empty;
      string temp = string.Empty;
      string rot = string.Empty;
      string nome = string.Empty;
      bool teste1 = true;
      int segundos = 4;
      int segundosLuzAlarme = 0;
      int tempdes = 200;
      int v_temp;
      int qtd_emerg_pendentes = 0;
      long tempox = 0;
      bool flagAlerta_v_temp_alta = false;
      bool flagAlerta_proc_iniciado = false;
      bool flagAlerta_ventilador_ligado = false;
      bool flagAlerta_ventilador_girando = false;
      bool flagAlerta_arrefecimento_preparado = false;
      bool flagAlerta_arrefecimento_preparado2 = false;
      #endregion

      public Form1()
      {
         InitializeComponent();
         serialPort1.DataReceived += new SerialDataReceivedEventHandler(serialPort1_DataReceived);
      }

      private void Form1_Load(object sender, EventArgs e)
      {
         Leitura_porta();  //chama a fç abaixo para configurar portas
         Padrao();         //Configurações padrão
         lbl_Data.Text = DateTime.Now.ToString("dd/MM/yyyy");  //Relógio
         lbl_Horas.Text = DateTime.Now.ToString("HH:mm");

         label17.Text = (tempdes * 0.5).ToString("N0");   //Display de temperatura instantâneo
         label18.Text = (tempdes * 0.75).ToString("N0");
         label20.Text = (tempdes * 0.9).ToString("N0");
```

```
    label21.Text = tempdes.ToString();
    label25.Text = (tempdes * 1.1).ToString("N0");
}

public void Leitura_porta()
{
    #region Configurar Portas
    String[] valoresPort = SerialPort.GetPortNames();//Comando para pegar as entradas
    for (int i = 0; i < valoresPort.Length; i++)     // do pc
    {
        combo_port.Items.Add(valoresPort[i]);
    }
    #endregion

    #region Configurar Baud Rate
    int[] valoresBaud = { 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200 };

    for (int i = 0; i < valoresBaud.Length; i++)  //Pegando os valores da lista valoresBaud
    {
        combo_baud.Items.Add(valoresBaud[i].ToString());
    }
    #endregion

    #region Configurar Data Bits
    combo_data.Items.Add("7");  //Disponibilizando duas opções de Data Bits
    combo_data.Items.Add("8");
    #endregion

    #region Configurar Stop Bits
    combo_stop.Items.Add("None"); //Disponibilizando 3 opções de Stop Bits
    combo_stop.Items.Add("One");
    combo_stop.Items.Add("two");
    #endregion

    #region Configurar Parity
    combo_parity.Items.Add("NONE"); //Disponibilizando 5 opções de Paridade
    combo_parity.Items.Add("EVEN");
    combo_parity.Items.Add("ODD");
    combo_parity.Items.Add("MARK");
    combo_parity.Items.Add("SPACE");
    #endregion

}

public void Padrao()
{
    combo_port.Text = "COM2";  //Definindo as configurações padrão da comunicação serial
    combo_baud.Text = "9600";
    combo_data.Text = "8";
    combo_stop.Text = "One";
    combo_parity.Text = "NONE";
}

private void serialPort1_DataReceived(object sender, SerialDataReceivedEventArgs args)
{    //Método acionado quando receber informações
    try
    {
        recebe = serialPort1.ReadLine();  //Leitura do que é recebido antes de quebrar linha
    }
    catch (IOException)
    {
    }
    this.BeginInvoke(new Fdelegate(Recepcao_serial), new object[] { recebe }); //Chama a fç Recepção Serial
}

public void Recepcao_serial(string a)
{
```

```
        this.label_Hora_Alarme.TextChanged += new System.EventHandler(this.label_Hora_Alarme_TextChanged);
//Relógio

        msg = a;
        //verificação de informações recebidas
        if (msg.Length == 18 && msg.Substring(15, 3).Equals("&%$") && msg.Substring(0,2).Equals("ba")
           && msg.Substring(3,2).Equals("bv") && msg.Substring(6, 1).Equals("t")
           && msg.Substring(10, 1).Equals("r"))
           {
           tratamento_strings(msg); //string adequada vai direto para aqui
        }
        else
        {
           label_msg.Text = ("ERRO");  //Aviso na interface de informação recebida fora do padrão
           msg_ok.BackColor = Color.Red;
           string[] evento = new string[3] { DateTime.Now.ToString("HH:mm:ss"), "Erro no recebimento de uma das
informações", "Alerta" };
           dataGridView1.Rows.Add(evento); //Avisando sobre o erro na lista de eventos
           label_Hora_Alarme.Text = DateTime.Now.ToString("HH:mm:ss"); //Preenchendo os textos no monitor de alarmes
           label_razao_alarme.Text = evento[2];
           label_90.Visible = true;
           lbl_descricao.Text = evento[1];
           lbl_sugestao.Text = "Caso seja um problema recorrente, verificar conexão";
           arquivo.WriteLine(DateTime.Now + "                                                    " + evento[1] + "/" +
evento[2]);
           //Preenchendo no arquivo txt sobre o erro
        }
     }

     public void tratamento_strings(string msg)
     {
        //decompondo a string recebida
        temp = msg.Substring(7, 3);
        rot = msg.Substring(11, 4);
        v_temp = int.Parse(temp);
        msg_ok.BackColor = Color.Green; //msg recebida com sucesso
        lbl_emerg_pendentes.Text = qtd_emerg_pendentes.ToString();

        chart_temp.Visible = true;  //Deixando os gráficos visíveis assim que alguma informação for recebida
        chart_rot.Visible = true;

        //Indicando que as informações recebidas estão OK
        if (panel_test1.BackColor == Color.Navy) panel_test1.BackColor = Color.Green;
        else panel_test1.BackColor = Color.Navy; //navy=azul
        if (panel_test2.BackColor == Color.Green) panel_test2.BackColor = Color.Navy;
        else panel_test2.BackColor = Color.Green;
        //test1 e test2 são as luzes intermitentes ao receber string
        label_msg.Text = ("Mensagem OK");

        try
        {
           label_temperatura.Text = temp;
           lbl_rot.Text = rot;  //Passando os valores dos dados recebidos para a interface

           if (chart_temp.Series[0].Points.Count > 20)   //Informações nos gráficos (temperatura)
           {
              chart_temp.Series[0].Points.RemoveAt(0);  //Atualizando pontos(Last in, First out)
              chart_temp.Update();
           }
           chart_temp.Series[0].Points.AddXY(tempox, temp);

           if (chart_temp.Series[1].Points.Count > 20) //Informações nos gráficos (tempdesejada)
           {
              chart_temp.Series[1].Points.RemoveAt(0); //Atualizando pontos(Last in, First out)
              chart_temp.Update();
           }
```

```
            chart_temp.Series[1].Points.AddXY(tempox, tempdes);

            if (chart_rot.Series[0].Points.Count > 20) //Informações nos gráficos (rotação)
            {
               chart_rot.Series[0].Points.RemoveAt(0);  //Atualizando pontos(Last in, First out)
               chart_rot.Update();
            }
            chart_rot.Series[0].Points.AddXY(tempox, rot);
            chart_rot.ChartAreas[0].AxisY.Maximum = 4000;  //Definindo máximos e mínimos no gráfico de rotação
            chart_rot.ChartAreas[0].AxisY.Minimum = 0;

            if (msg.Substring(5, 1) == "1")   //Status dos sistemas
            {
               but_vent.Text = "Desligar Ventilador";
               lbl_vent.Text = "ON";  //Configurando um dos botões de comando para desligar ventilador
               if (flagAlerta_ventilador_ligado == false)
               {
                  if (msg.Substring(2, 1) == "1" && flagAlerta_arrefecimento_preparado2 == true && msg.Substring(5, 1) ==
"1")
                  {  //Caso a emergência do arrefecimento estar despreparado e o processo iniciado, ele entra nesta condição
informando que o problema foi corrigido
                     string[] evento1 = new string[3] { DateTime.Now.ToString("HH:mm:ss"), "Sistema de Arrefecimento
Preparado com Processo Iniciado", "Normal" };
                     dataGridView1.Rows.Add(evento1); //Avisando sobre o erro na lista de eventos
                     flagAlerta_arrefecimento_preparado2 = false;
                     label_Hora_Alarme.Text = "     "; //Apagando os textos no monitor de alarmes
                     label_razao_alarme.Text = "    "; label_90.Visible = false;
                     lbl_descricao.Text = "    ";
                     lbl_sugestao.Text = "   ";

                     qtd_emerg_pendentes--;
                     arquivo.WriteLine(DateTime.Now + "                                                    " + evento1[1] + "/" +
evento1[2]);
                  } //Preenchendo no arquivo txt sobre a correção
                  else
                  {
                     string[] evento = new string[3] { DateTime.Now.ToString("HH:mm:ss"), "Sistema de arrefecimento
(Ventilador) Preparado", "Normal" };
                     dataGridView1.Rows.Add(evento); //Avisando sobre o evento na lista de eventos
                     flagAlerta_ventilador_ligado = true;
                     arquivo.WriteLine(DateTime.Now + "                                                    " + evento[1] + "/" +
evento[2]);
                  }   // Preenchendo o evento no arquivo txt
               }

               flagAlerta_arrefecimento_preparado = true;
            }
            else
            {
               but_vent.Text = "Acionar Ventilador";
               lbl_vent.Text = "OFF";  //Configurando um dos botões de comando para acionar ventilador
               if (flagAlerta_ventilador_ligado == true)
               {
                  string[] evento = new string[3] { DateTime.Now.ToString("HH:mm:ss"), "Sistema de arrefecimento
(Ventilador) Desligado por Completo", "Alerta" };
                  dataGridView1.Rows.Add(evento); //Avisando sobre o alerta na lista de eventos
                  label_Hora_Alarme.Text = DateTime.Now.ToString("HH:mm:ss");
                  label_razao_alarme.Text = evento[2]; //preenchendo os textos no monitor de alarmes
                  flagAlerta_ventilador_ligado = false;
                  flagAlerta_arrefecimento_preparado = false;
                  label_90.Visible = true;
                  lbl_descricao.Text = evento[1];
                  lbl_sugestao.Text = "Caso deseje preparar novamente, clique no comando";
                  arquivo.WriteLine(DateTime.Now + "                                                    " + evento[1] + "/" +
evento[2]);
               }   // Preenchendo o evento no arquivo txt
            }
```

```
            if (int.Parse(rot) > 1)
            {
               pictureBox3.Visible = true; //Imagem de vento visivel
               if (flagAlerta_ventilador_girando == false && msg.Substring(5, 1) == "1")
               {
                  string[] evento = new string[3] { DateTime.Now.ToString("HH:mm:ss"), "Ventilador Girando", "Normal" };
                  dataGridView1.Rows.Add(evento); //Avisando sobre o evento na lista de eventos
                  flagAlerta_ventilador_girando = true;
                  arquivo.WriteLine(DateTime.Now + "                                                    " + evento[1] + "/" +
evento[2]);
               }  // Preenchendo o evento no arquivo txt
            }
            else
            {
               pictureBox3.Visible = false;  //Tirando o "vento" da imagem do ventilador

               if (flagAlerta_ventilador_girando == true && msg.Substring(5, 1) == "1")
               {
                  string[] evento = new string[3] { DateTime.Now.ToString("HH:mm:ss"), "Ventilador Parou de Girar pois
Temperatura está Equilibrada", "Normal" };
                  dataGridView1.Rows.Add(evento); //Avisando sobre o evento na lista de eventos
                  flagAlerta_ventilador_girando = false;
                  arquivo.WriteLine(DateTime.Now + "                                                    " + evento[1] + "/" +
evento[2]);
               } // Preenchendo o evento no arquivo txt

               if (flagAlerta_ventilador_girando == true && msg.Substring(5, 1) == "0")
               {
                  string[] evento = new string[3] { DateTime.Now.ToString("HH:mm:ss"), "Ventilador Parou de Girar pois
Sistema de Arrefecimento está Despreparado", "Alerta" };
                  dataGridView1.Rows.Add(evento); //Avisando sobre o evento na lista de eventos
                  label_Hora_Alarme.Text = DateTime.Now.ToString("HH:mm:ss");
                  label_razao_alarme.Text = evento[2]; //preenchendo os textos no monitor de alarmes
                  flagAlerta_ventilador_girando = false;
                  label_90.Visible = true;
                  lbl_descricao.Text = evento[1];
                  lbl_sugestao.Text = "Acione o Sistema de Arrefecimento no Comando!";
                  arquivo.WriteLine(DateTime.Now + "                                                    " + evento[1] + "/" +
evento[2]);
               } // Preenchendo o evento no arquivo txt
            }
            if (msg.Substring(2, 1) == "1")
            {
               but_aque.Text = "Finalizar Processo";
               lbl_aque.Text = "ON"; //Configurando um dos botões de comando para finalizar processo

               if (flagAlerta_proc_iniciado == false)
               {
                  string[] evento = new string[3] { DateTime.Now.ToString("HH:mm:ss"), "Processo Térmico Iniciado",
"Normal" };
                  dataGridView1.Rows.Add(evento); //Avisando sobre o evento na lista de eventos
                  flagAlerta_proc_iniciado = true;
                  arquivo.WriteLine(DateTime.Now + "                                                    " + evento[1] + "/" +
evento[2]);
                  // Preenchendo o evento no arquivo txt
               }

               if (msg.Substring(2,1) == "1" && flagAlerta_arrefecimento_preparado == false && msg.Substring(5, 1) == "0")
               {
                  string[] evento1 = new string[3] { DateTime.Now.ToString("HH:mm:ss"), "Sistema de Arrefecimento
Despreparado e Processo Térmico Inicializado", "Emergência" };
                  dataGridView1.Rows.Add(evento1); //Avisando sobre o evento na lista de eventos
                  flagAlerta_arrefecimento_preparado = true;
                  flagAlerta_arrefecimento_preparado2 = true;
                  label_Hora_Alarme.Text = DateTime.Now.ToString("HH:mm:ss"); //preenchendo os textos no monitor de
alarmes
```

```
                label_razao_alarme.Text = evento1[2]; label_90.Visible = true;
                lbl_descricao.Text = evento1[1];
                lbl_sugestao.Text = "Verifique o acionamento do Sistema de Arrefecimento!";
                qtd_emerg_pendentes++;
                arquivo.WriteLine(DateTime.Now + "                                                    " + evento1[1] + "/" +
evento1[2]);
            } // Preenchendo o evento no arquivo txt

        }
        else
        {
            but_aque.Text = "Iniciar Processo";
            lbl_aque.Text = "OFF"; //Configurando um dos botões de comando para iniciar processo
            if (flagAlerta_proc_iniciado == true)
            {
                string[] evento = new string[3] { DateTime.Now.ToString("HH:mm:ss"), "Processo Térmico Finalizado",
"Alerta" };
                dataGridView1.Rows.Add(evento); //Avisando sobre o evento na lista de eventos
                label_Hora_Alarme.Text = DateTime.Now.ToString("HH:mm:ss");
                label_razao_alarme.Text = evento[2]; //preenchendo os textos no monitor de alarmes
                flagAlerta_proc_iniciado = false;
                flagAlerta_arrefecimento_preparado = false;
                label_90.Visible = true;
                lbl_descricao.Text = evento[1];
                lbl_sugestao.Text = "Caso deseje iniciar novamente, clique no comando Iniciar Processo";
                arquivo.WriteLine(DateTime.Now + "                                                    " + evento[1] + "/" +
evento[2]);
            } // Preenchendo o evento no arquivo txt
        }

        if (v_temp >= 27)  //Colocando o simbolo da resistencia na cor vermelha
        {
            pictureBox2.Image = Supervisório_de_Processo_Térmico.Properties.Resources.simbolo_resistencias_quente;
        }

        if (v_temp <= 27)   //Display de temperatura instantanea até 27C
        {
            panel2.BackColor = Color.Silver;
            panel3.BackColor = Color.Silver;
            panel4.BackColor = Color.Silver;
            panel6.BackColor = Color.Silver;
            panel7.BackColor = Color.Silver;
            panel11.BackColor = Color.Silver;   //Colocando o simbolo da resistencia preto
            pictureBox2.Image = Supervisório_de_Processo_Térmico.Properties.Resources.simbolo_resistencias;
        }
        else if (v_temp > 27 && v_temp < tempdes*0.5)  //Ajustando display de temperatura entre 27C e 50% da temp des
        {

            #region cores3
            panel2.BackColor = Color.Teal;
            panel3.BackColor = Color.Silver;
            panel4.BackColor = Color.Silver;
            panel6.BackColor = Color.Silver;
            panel7.BackColor = Color.Silver;
            panel11.BackColor = Color.Silver;
            #endregion
        }
        else if (v_temp >= tempdes*0.5 && v_temp < tempdes*0.75)
        { //Ajustando display de temperatura entre 50% da temp des e 75% da temp des

            #region cores4
            panel2.BackColor = Color.Teal;
            panel3.BackColor = Color.Teal;
            panel4.BackColor = Color.Silver;
            panel6.BackColor = Color.Silver;
            panel7.BackColor = Color.Silver;
            panel11.BackColor = Color.Silver;
```

```
            #endregion
        }
        else if (v_temp >= tempdes*0.75 && v_temp < tempdes*0.9)
        { //Ajustando display de temperatura entre 75% da temp des e 90% da temp des

            #region cores5
            panel2.BackColor = Color.Teal;
            panel3.BackColor = Color.Teal;
            panel4.BackColor = Color.Teal;
            panel6.BackColor = Color.Silver;
            panel7.BackColor = Color.Silver;
            panel11.BackColor = Color.Silver;
            #endregion
        }
        else if (v_temp >= tempdes*0.9 && v_temp < tempdes)
        {  //Ajustando display de temperatura entre 90% da temp des e 100% da temp des

            #region cores6
            panel2.BackColor = Color.Teal;
            panel3.BackColor = Color.Teal;
            panel4.BackColor = Color.Teal;
            panel6.BackColor = Color.Green;
            panel7.BackColor = Color.Silver;
            panel11.BackColor = Color.Silver;
            #endregion
        }
        else if (v_temp >= tempdes && v_temp < tempdes*1.10)
        {   //Ajustando display de temperatura entre 100% da temp des e 110% da temp des

            #region cores7
            panel2.BackColor = Color.Teal;
            panel3.BackColor = Color.Teal;
            panel4.BackColor = Color.Teal;
            panel6.BackColor = Color.Green;
            panel7.BackColor = Color.Green;
            panel11.BackColor = Color.Silver;
            #endregion
        }
        else if (v_temp >= tempdes*1.10)
        {    //Ajustando display de temperatura para maior que 110% da temp des

            #region cores10
            panel2.BackColor = Color.Teal;
            panel3.BackColor = Color.Teal;
            panel4.BackColor = Color.Teal;
            panel6.BackColor = Color.Green;
            panel7.BackColor = Color.Green;
            panel11.BackColor = Color.Maroon;

            if (flagAlerta_v_temp_alta == false)
            {

                string[] evento = new string[3] { DateTime.Now.ToString("HH:mm:ss"), "Temperatura obtida 10% superior à
desejada ", "Emergência" };
                dataGridView1.Rows.Add(evento); //Avisando sobre o evento na lista de eventos
                flagAlerta_v_temp_alta = true;
                label_Hora_Alarme.Text = DateTime.Now.ToString("HH:mm:ss");
                label_razao_alarme.Text = evento[2];  //preenchendo os textos no monitor de alarmes
                label_90.Visible = true;
                lbl_descricao.Text = evento[1];
                lbl_sugestao.Text = "Sistema de arrefecimento com algum problema!";
                qtd_emerg_pendentes++;
                arquivo.WriteLine(DateTime.Now + "                                                    " + evento[1] + "/" +
evento[2]);
            }  // Preenchendo o evento no arquivo txt
            #endregion
        }
```

```
            if (v_temp < 1.1 * tempdes && flagAlerta_v_temp_alta == true)
            {
               string[] evento = new string[3] { DateTime.Now.ToString("HH:mm:ss"), "Temperatura obtida volta ao normal",
"Normal" };
               dataGridView1.Rows.Add(evento);  //Avisando sobre o evento na lista de eventos
               flagAlerta_v_temp_alta = false;
                //Apagando todos os textos no monitor de alarmes
               label_Hora_Alarme.Text = "     ";
               label_razao_alarme.Text = "    "; label_90.Visible = false;
               lbl_descricao.Text = "    ";
               lbl_sugestao.Text = "   ";

               qtd_emerg_pendentes--; //Subtraindo 1 das emergencias pendentes
               arquivo.WriteLine(DateTime.Now + "                                                                    " + evento[1] + "/" +
evento[2]);
            }   // Preenchendo o evento no arquivo txt
            tempox++;

            arquivo.WriteLine(DateTime.Now + "        " + temp + "                " + tempdes.ToString("000")
               + "                    "+rot); //escrever no txt

            label17.Text = (tempdes * 0.5).ToString("N0"); //Ajuste de temperatura no display
            label18.Text = (tempdes * 0.75).ToString("N0");
            label20.Text = (tempdes * 0.9).ToString("N0");
            label21.Text = tempdes.ToString();
            label25.Text = (tempdes * 1.1).ToString("N0");

         }
         catch
         {
            label_msg.Text = ("String não decimal");
            msg_ok.BackColor = Color.Red; //Informação de erro ao receber dados
         }
      }

      private void but_porta_Click(object sender, EventArgs e)
      {
         pasta = new DirectoryInfo(diretorio);
         pasta.Create(); //Assim q se clica no botão abrir porta, um arquivo é criado
         if (teste1)
         {
            if (serialPort1.IsOpen == true) serialPort1.Close();
            else
            {
            //fç do VS que recebe as informações do módulo RF (configuração)
            serialPort1.PortName = combo_port.Text;
            serialPort1.BaudRate = int.Parse(combo_baud.Text);
            serialPort1.DataBits = int.Parse(combo_data.Text);
            serialPort1.StopBits = (StopBits)(combo_stop.SelectedIndex);
            serialPort1.Parity = (Parity)(combo_parity.SelectedIndex);
            }
            try
            {

            #region Criar arquivo
            //contrução do txt com nome criado automaticamente
            nome = ("Ar. " + DateTime.Now.ToString("dd-MM-yyyy HH-mm-ss"));
            textBox_nome.Text = nome;
            arquivo = new StreamWriter(diretorio + "\\" + nome + ".txt");

               arquivo.WriteLine("Data e Hora       " + "  Temperatura[°C]" + "   Temperatura Desejada[°C]" + "   Rotação da
ventoinha[RPM]" + "   Eventos, Alarmes e Gravidade ");
               #endregion  //Escrevendo 1a linha do arquivo txt

               teste1 = false;  //não perimitir os botões ficaram "clicaveis" ao abrir a porta
            serialPort1.Open();
```

```
            but_porta.Text = "Fechar Porta";  //Comandos para alterar a interface quando a porta for aberta
            but_padrao.Enabled = false;
            combo_port.Enabled = false;
            combo_baud.Enabled = false;
            combo_data.Enabled = false;
            combo_parity.Enabled = false;
            combo_stop.Enabled = false;
            trackBar_tempdes.Enabled = true;
            but_atualizar.Enabled = false;
            timer1.Enabled = true;
            but_aque.Enabled = true;
            but_vent.Enabled = true;
            serialPort1.WriteLine("t" + tempdes.ToString("000") + "&%$"); //Enviar comando de temp des assim que a porta
for aberta (medidas de segurança no simulador)
            string[] evento = new string[3] { DateTime.Now.ToString("HH:mm:ss"),"Abertura de Porta Serial","Normal"};
            dataGridView1.Rows.Add(evento);  //Escrevendo evento no txt e na lista de eventos
            arquivo.WriteLine(DateTime.Now + "                                                    " + evento[1] + "/" +
evento[2]);


            }
            catch
            {
               MessageBox.Show("Não foi possível abrir a porta selecionada");  //Msg de erro quando não for possivel abrir
porta
               string[] evento = new string[3] { DateTime.Now.ToString("HH:mm:ss"), "Falha na Tentativa de Abertura da
Porta Serial", "Alerta" };
               dataGridView1.Rows.Add(evento); //Avisando sobre o evento na lista de eventos
               label_Hora_Alarme.Text = DateTime.Now.ToString("HH:mm:ss");
               label_razao_alarme.Text = evento[2]; //preenchendo os textos no monitor de alarmes
               label_90.Visible = true;
               lbl_descricao.Text = evento[1];
               lbl_sugestao.Text = "Aguarde alguns instantes e tente novamente";
               lbl_emerg_pendentes.Text = qtd_emerg_pendentes.ToString();
            }
            }
            else
            {
               try
               {
                  teste1 = true; //deixar os botões "clicaveis" quando o programa estiver de porta fechada
                  arquivo.Close();
                  serialPort1.Close();
                  but_porta.Text = "Abrir Porta"; //Comandos para alterar a interface quando a porta estiver fechada
                  msg_ok.BackColor = Color.Silver;
                  panel_test1.BackColor = Color.Silver;
                  panel_test2.BackColor = Color.Silver;
                  label_msg.Text = ("Aguardando Conexão");
                  but_padrao.Enabled = true;
                  combo_port.Enabled = true;
                  combo_baud.Enabled = true;
                  combo_data.Enabled = true;
                  combo_parity.Enabled = true;
                  combo_stop.Enabled = true;
                  trackBar_tempdes.Enabled = false;
                  but_atualizar.Enabled = true;
                  but_vent.Enabled = false;
                  but_aque.Enabled = false;
                  timer1.Enabled = false;
                  label_temperatura.Text = "000";
                  //label_tempdes.Text = "200";
                  lbl_rot.Text = "0000";
                  label_comando.Text = "            ";
                  label_valor_comando.Text = "            ";
                  panel_comando.BackColor = Color.DarkGray;
                  panel2.BackColor = Color.Silver;
                  panel3.BackColor = Color.Silver;
```

```
                panel4.BackColor = Color.Silver;
                panel6.BackColor = Color.Silver;
                panel7.BackColor = Color.Silver;
                panel11.BackColor = Color.Silver;

                label_comandoBut.Text = "            ";
                panel_comandoBut.BackColor = Color.DarkGray;
                //chart_temp.Series.Clear();
                pictureBox3.Visible = false;

                pictureBox2.Image = Supervisório_de_Processo_Térmico.Properties.Resources.simbolo_resistencias;

                but_vent.Text = "Acionar Ventilador";
                but_aque.Text = "Iniciar Processo";
                but_vent.Enabled = false;
                but_aque.Enabled = false;
                lbl_aque.Text = "OFF";
                lbl_vent.Text = "OFF";

                label17.Text = (tempdes * 0.5).ToString("N0");
                label18.Text = (tempdes * 0.75).ToString("N0");
                label20.Text = (tempdes * 0.9).ToString("N0");
                label21.Text = tempdes.ToString();
                label25.Text = (tempdes * 1.1).ToString("N0");

                string[] evento = new string[3] { DateTime.Now.ToString("HH:mm:ss"), "Fechamento da Porta Serial",
"Alerta" };
                dataGridView1.Rows.Add(evento); //Avisando sobre o evento na lista de eventos
                label_Hora_Alarme.Text = DateTime.Now.ToString("HH:mm:ss");
                label_razao_alarme.Text = evento[2]; //preenchendo os textos no monitor de alarmes
                label_90.Visible = true;
                lbl_descricao.Text = evento[1];
                lbl_sugestao.Text = "Caso desejado, abrir Porta Serial";
            }
            catch
            {
                MessageBox.Show("Não foi possível fechar a porta selecionada"); //Msg de erro quando não for possivel
fechar porta
                string[] evento = new string[3] { DateTime.Now.ToString("HH:mm:ss"), "Falha na Tentativa de Fechamento
da Porta Serial", "Alerta" };
                dataGridView1.Rows.Add(evento); //Avisando sobre o evento na lista de eventos
                label_Hora_Alarme.Text = DateTime.Now.ToString("HH:mm:ss");
                label_razao_alarme.Text = evento[2]; //preenchendo os textos no monitor de alarmes
                label_90.Visible = true;
                lbl_descricao.Text = evento[1];
                lbl_sugestao.Text = "Aguarde alguns instantes e tente novamente";
            }
        }
    }

    private void but_pasta_Click(object sender, EventArgs e)  //Abrir pasta dos arquivos gerados no botão
    {
        Process.Start("Explorer", "C:\\Arquivos Supervisório TCC\\" + DateTime.Now.ToString("dd-MM-yyyy"));
    }

    private void trackBar_tempdes_Scroll(object sender, EventArgs e)  //Comando de temperatura desejada
    {
        tempdes = 100 + trackBar_tempdes.Value; //valor da barra de rolagem
        label_tempdes.Text = tempdes.ToString("000"); //3 digitos
        segundos = 0; //zerando timer
        label_comando.Text = "            ";
        panel_comando.BackColor = Color.DarkGray;
    }

    private void timer1_Tick(object sender, EventArgs e) //Metodo para não enviar instantaneamente comandos
    {                                       //de temperatura desejada
        segundos++;
```

```
        if (segundos==2) serialPort1.WriteLine("t" + tempdes.ToString("000") + "&%$");
        if (segundos>=2 && segundos <=3) //Enviar comando depois de 2 segundos e apagar mensagens depois de 3 s
        {
            label_comando.Text = "Comando enviado!";
            label_valor_comando.Text = "(" + tempdes.ToString("00") + " °C)";
            panel_comando.BackColor = Color.Green;
            if (segundos >= 2 && segundos < 3)
            {
                string[] evento = new string[3] { DateTime.Now.ToString("HH:mm:ss"), "Comando de Alteração de
Temperatura Desejada " + label_valor_comando.Text, "Normal" };
                dataGridView1.Rows.Add(evento); //Avisando sobre o evento na lista de eventos
                arquivo.WriteLine(DateTime.Now + "                                                    " + evento[1] + "/" +
evento[2]);
            }  //Escrevendo evento no txt
        }
        else
        {
            label_comando.Text = "            "; //Apagando labels ao enviar comando
            label_valor_comando.Text = "            ";
            panel_comando.BackColor = Color.DarkGray;
        }
    }

    private void timer_relogio_Tick(object sender, EventArgs e) //Relogio
    {
        lbl_Data.Text = DateTime.Now.ToString("dd/MM/yyyy");
        lbl_Horas.Text = DateTime.Now.ToString("HH:mm");
    }

    private void but_vent_Click(object sender, EventArgs e)  //Comando para acionar sistema de arrefecimento
    {
        string message = "Você deseja " + but_vent.Text + "?";
        string caption = "Confirme esta ação"; //mensagens de aviso antes de confirmar o comando

        MessageBoxButtons buttons = MessageBoxButtons.YesNo;
        DialogResult result;

        // Apresenta a MessageBox.
        result = MessageBox.Show(message, caption, buttons);
        if (result == System.Windows.Forms.DialogResult.Yes) //Se a resposta for sim
        {
            try
            {
                if (but_vent.Text == "Acionar Ventilador")
                {
                    serialPort1.WriteLine("ca" + msg.Substring(2, 1) + "cv1&%$");  //Enviando string
                    string[] evento = new string[3] { DateTime.Now.ToString("HH:mm:ss"), "Comando de Acionamento de
Ventilador Realizado", "Normal" };
                    dataGridView1.Rows.Add(evento); //Avisando sobre o evento na lista de eventos
                    arquivo.WriteLine(DateTime.Now + "                                                    " + evento[1] + "/" +
evento[2]);
                }  //Escrevendo evento no txt
                else
                {
                    serialPort1.WriteLine("ca" + msg.Substring(2, 1) + "cv0&%$"); //Enviando string
                    string[] evento = new string[3] { DateTime.Now.ToString("HH:mm:ss"), "Comando de Desligamento de
Ventilador Realizado", "Normal" };
                    dataGridView1.Rows.Add(evento); //Avisando sobre o evento na lista de eventos
                    arquivo.WriteLine(DateTime.Now + "                                                    " + evento[1] + "/" +
evento[2]);
                } //Escrevendo evento no txt
                label_comandoBut.Text = "            "; //Apagando label do comando
                panel_comandoBut.BackColor = Color.DarkGray;
            }
            catch
            {
                label_comandoBut.Text = "ERRO"; //Erro no envio dos comandos
```

```
                panel_comandoBut.BackColor = Color.Red;
                string[] evento = new string[3] { DateTime.Now.ToString("HH:mm:ss"), "Erro no Envio do Comando para o
Ventilador", "Alerta" };
                dataGridView1.Rows.Add(evento); //Avisando sobre o evento na lista de eventos
                label_Hora_Alarme.Text = DateTime.Now.ToString("HH:mm:ss");
                label_razao_alarme.Text = evento[2]; //Preenchendo os textos no monitor de alarmes
                label_90.Visible = true;
                lbl_descricao.Text = evento[1];
                lbl_sugestao.Text = "Aguarde alguns instantes e tente novamente ou acione manualmente";
                lbl_emerg_pendentes.Text = qtd_emerg_pendentes.ToString();
                arquivo.WriteLine(DateTime.Now + "                                                    " + evento[1] + "/" +
evento[2]);
            }    //Escrevendo evento no txt
        }
    }

    private void but_aque_Click(object sender, EventArgs e)   //Comando para iniciar processo térmico
    {
        string message = "Você deseja " + but_aque.Text + "?";
        string caption = "Confirme esta ação"; //mensagens de aviso antes de confirmar o comando

        MessageBoxButtons buttons = MessageBoxButtons.YesNo;
        DialogResult result;

        // Apresenta a MessageBox.
        result = MessageBox.Show(message, caption, buttons);
        if (result == System.Windows.Forms.DialogResult.Yes) //Se a resposta for sim
        {
            try
        {

                if (but_aque.Text == "Iniciar Processo")
                {
                    serialPort1.WriteLine("ca1cv" + msg.Substring(5, 1) + "&%$"); //Enviando string
                    string[] evento = new string[3] { DateTime.Now.ToString("HH:mm:ss"), "Comando de Iniciar Processo
Realizado", "Normal" };
                    dataGridView1.Rows.Add(evento); //Avisando sobre o evento na lista de eventos
                    arquivo.WriteLine(DateTime.Now + "                                                    " + evento[1] + "/" +
evento[2]);
                }   //Escrevendo evento no txt
                else
                {
                    serialPort1.WriteLine("ca0cv" + msg.Substring(5, 1) + "&%$"); //Enviando string
                    string[] evento = new string[3] { DateTime.Now.ToString("HH:mm:ss"), "Comando de Finalizar Processo
Realizado", "Normal" };
                    dataGridView1.Rows.Add(evento); //Avisando sobre o evento na lista de eventos
                    arquivo.WriteLine(DateTime.Now + "                                                    " + evento[1] + "/" +
evento[2]);
                }   //Escrevendo evento no txt
                label_comandoBut.Text = "            ";  //Apagando label do comando
                panel_comandoBut.BackColor = Color.DarkGray;
            }
        catch
        {
            label_comandoBut.Text = "ERRO"; //Erro no envio dos comandos
            panel_comandoBut.BackColor = Color.Red;
            string[] evento = new string[3] { DateTime.Now.ToString("HH:mm:ss"), "Erro no Envio do Comando para a
Inicialização/Finalização do Processo Térmico", "Alerta" };
            dataGridView1.Rows.Add(evento); //Avisando sobre o evento na lista de eventos
            label_Hora_Alarme.Text = DateTime.Now.ToString("HH:mm:ss");
            label_razao_alarme.Text = evento[2]; //Preenchendo os textos no monitor de alarmes
            label_90.Visible = true;
            lbl_descricao.Text = evento[1];
            lbl_sugestao.Text = "Aguarde alguns instantes e tente novamente";
            lbl_emerg_pendentes.Text = qtd_emerg_pendentes.ToString();
            arquivo.WriteLine(DateTime.Now + "                                                    " + evento[1] + "/" +
evento[2]);
```

```
            //Escrevendo evento no txt
        }
    }
}

private void but_expandir_Click(object sender, EventArgs e)  //Botão do monitor de alarme
{
    if (but_expandir.Text == ">")
    {  //Botão para expandir tela
        but_expandir.Text = "<";
        panel_alarme.Width = 800;
        but_expandir.Left = 775;
        label_90.Visible = false;
        lbl_descricao.Location = new Point(120, 40);
        lbl_sugestao.Location = new Point(120, 71);
        lbl_sugestao.Visible = true;
        lbl_descricao.Visible = true;
    }
    else
    {  //O mesmo botão para contrair a tela
        but_expandir.Text = ">";
        but_expandir.Left = 135;
        panel_alarme.Width = 160;
        lbl_sugestao.Visible = false;
        lbl_descricao.Visible = false;
        if (label_razao_alarme.Text =="Alerta" || label_razao_alarme.Text == "Emergência")
        {
            label_90.Visible = true;
        }
    }
}


private void label_Hora_Alarme_TextChanged(object sender, EventArgs e)
{
    timer_luz_alarme.Enabled = true;  //Ativar timer de piscar painel do monitor de alarme
}



private void timer_luz_alarme_Tick(object sender, EventArgs e)
{
    if (panel_alarme.BackColor == Color.Black) //Função para piscar a tela do monitor de alarme
    {
        if (label_razao_alarme.Text == "Emergência") panel_alarme.BackColor = Color.Red;
        if (label_razao_alarme.Text == "Alerta") panel_alarme.BackColor = Color.Yellow;

    }
    else panel_alarme.BackColor = Color.Black;

    segundosLuzAlarme++;

    if (segundosLuzAlarme >= 6) //Parar de piscar depois de alguns segundos
    {
        panel_alarme.BackColor = Color.Black;
        segundosLuzAlarme = 0;
        timer_luz_alarme.Enabled = false;
        if (label_razao_alarme.Text == "Alerta") but_ignorarAlerta.Visible = true;
    }
}



private void but_ignorarAlerta_Click(object sender, EventArgs e) //Botão de ignorar alerta (não é possivel apagar
emergencias!)
{
    label_Hora_Alarme.Text = "     ";  //Apagando tudo do monitor de Alarme
```

```
            label_razao_alarme.Text = "     "; label_90.Visible = false;
            lbl_descricao.Text = "     ";
            lbl_sugestao.Text = "    ";

            if (label_Hora_Alarme.Text == "      ") but_ignorarAlerta.Visible = false;
        }
    }
}
```

Apêndice A. Código do sistema supervisório em C#.

Apêndice B – Código do simulador de processo térmico

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.IO;
using System.IO.Ports;

namespace Simulador_de_Processo_Termico
{
   public partial class Form1 : Form
   {
      public delegate void Fdelegate(string a);

      #region Declarar variáveis
      string msg = string.Empty;    //string do pic
      string recebe = string.Empty; //p criar o objeto de recepção serial
      string nome = string.Empty;
      bool teste = true; //variavel acionada para abertura de porta, coleta as informações do mod. RF
      double temp = 0;
      double tempaux = 0;
      double temporizador = 000;
      double tempdes = 200;
      double acaoresf = 0;
      double tempres = 0;
      int botao_vent_acion = 0;
      int botao_aque_acion = 0;
      bool flagprocandamento = false;

      #endregion

      public Form1()
      {
         InitializeComponent(); //Inicializando Form
         serialPort1.DataReceived += new SerialDataReceivedEventHandler(serialPort1_DataReceived);
      }

      private void Form1_Load(object sender, EventArgs e)
      {
         Leitura_porta();  //chama a fç abaixo
         Padrao();       //Configurações padrão do modulo RF
      }

      public void Leitura_porta()
      {
         #region Configurar Portas
         String[] valoresPort = SerialPort.GetPortNames();//Comando p pegar as entradas
         for (int i = 0; i < valoresPort.Length; i++)     // do pc
         {
            combo_port.Items.Add(valoresPort[i]);
         }
         #endregion

         #region Configurar Baud Rate
         int[] valoresBaud = { 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200 };
```

```
    for (int i = 0; i < valoresBaud.Length; i++)
    {
        combo_baud.Items.Add(valoresBaud[i].ToString());
    }
    #endregion

    #region Configurar Data Bits
    combo_data.Items.Add("7");
    combo_data.Items.Add("8");
    #endregion

    #region Configurar Stop Bits
    combo_stop.Items.Add("None");
    combo_stop.Items.Add("One");
    combo_stop.Items.Add("two");
    #endregion

    #region Configurar Parity
    combo_parity.Items.Add("NONE");
    combo_parity.Items.Add("EVEN");
    combo_parity.Items.Add("ODD");
    combo_parity.Items.Add("MARK");
    combo_parity.Items.Add("SPACE");
    #endregion

}

public void Padrao()
{
    combo_port.Text = "COM1"; //Configurações padrão da comunicação serial
    combo_baud.Text = "9600";
    combo_data.Text = "8";
    combo_stop.Text = "One";
    combo_parity.Text = "NONE";
}

private void serialPort1_DataReceived(object sender, SerialDataReceivedEventArgs args)
{
    try
    {
        recebe = serialPort1.ReadLine();  //Leitura do que é recebido antes de quebrar linha
    }
    catch (IOException)
    {
    }
    this.BeginInvoke(new Fdelegate(Recepcao_serial), new object[] { recebe }); //chama a função de recepção serial
}

public void Recepcao_serial(string a)
{
    msg = a;

    if (msg.Length == 7 && msg.Substring(4, 3).Equals("&%$") && msg.Substring(0, 1).Equals("t"))
    {
        tratamento_strings(msg); //string adequada (dentro do padrão de segurança) vai para esta função
    }
    else if (msg.Length == 9 && msg.Substring(0, 2).Equals("ca") && msg.Substring(6, 3).Equals("&%$")
        && msg.Substring(3, 2).Equals("cv"))
    {
        comando_strings(msg); //String adequada para comandar o processo
    }
}

public void tratamento_strings(string msg)
{
    tempdes = double.Parse(msg.Substring(1, 3)); //Coletando os dados das strings recebidas
```

```
            lbl_tempdes.Text = msg.Substring(1,3);
        }

        public void comando_strings(string msg)
        {
            if (msg.Substring(2, 1) == "1") but_mot_Click(but_mot, null);
            else but_desmot_Click(but_desmot, null); //Acionando ou desligando processo a partir dos comandos do supervisório

            if (msg.Substring(5,1) == "1") but_acventoinha_Click(but_acventoinha,null);
            else but_desventoinha_Click(but_desventoinha,null); //Acionando ou desligando ventilador a partir dos comandos do
supervisório
        }
        private void but_padrao_Click(object sender, EventArgs e)
        {
            Padrao();   //Botão "padrão" da interface (configurações da comunicação serial)
        }

        private void but_atualizar_Click(object sender, EventArgs e)
        { //botão atualizar (apaga tudo e depois põe padrão)
            combo_port.Items.Clear();
            combo_baud.Items.Clear();
            combo_data.Items.Clear();
            combo_parity.Items.Clear();
            combo_stop.Items.Clear();
            Leitura_porta();
            Padrao();
        }

        private void but_porta_Click(object sender, EventArgs e)
        {
            //botão abrir porta serial
            if (teste)
            {
                if (serialPort1.IsOpen == true) serialPort1.Close();
                else
                {
                    //fç do VS que recebe as informações do módulo RF
                    serialPort1.PortName = combo_port.Text;
                    serialPort1.BaudRate = int.Parse(combo_baud.Text);
                    serialPort1.DataBits = int.Parse(combo_data.Text);
                    serialPort1.StopBits = (StopBits)(combo_stop.SelectedIndex);
                    serialPort1.Parity = (Parity)(combo_parity.SelectedIndex);
                }
                try
                {

                    teste = false;  //não permitir os botões ficarem "clicaveis" ao abrir a porta
                    serialPort1.Open();
                    but_porta.Text = "Fechar Porta";
                    but_padrao.Enabled = false;
                    combo_port.Enabled = false;
                    combo_baud.Enabled = false;
                    combo_data.Enabled = false;
                    combo_parity.Enabled = false;
                    combo_stop.Enabled = false;
                    but_atualizar.Enabled = false;
                    timer5.Enabled = true;

                }
                catch
                {
                    MessageBox.Show("Não foi possível abrir a porta selecionada"); //Mensagem de erro ao tentar abrir porta serial
                }
            }
            else
            {
                try
```

```
            {
                teste = true; //deixar os botões "clicaveis" quando o programa estiver de porta fechada
                serialPort1.Close();
                but_porta.Text = "Abrir Porta";
                but_padrao.Enabled = true;
                combo_port.Enabled = true;
                combo_baud.Enabled = true;
                combo_data.Enabled = true;
                combo_parity.Enabled = true;
                combo_stop.Enabled = true;
                but_atualizar.Enabled = true;
                timer5.Enabled = false;
            }
            catch
            {
                MessageBox.Show("Não foi possível fechar a porta selecionada"); //Mensagem de erro ao tentar fechar porta
serial
            }
        }
    }

    private void but_mot_Click(object sender, EventArgs e)
    {  //Função do aquecimento do processo térmico ao apertar botão
        timer1.Enabled = true;
        timer2.Enabled = false;
        but_desmot.Enabled = true;
        but_mot.Enabled = false;
        temporizador = 0;
        if (flagprocandamento == false) tempaux = temp + 27; //Se processo não for iniciado, cair nesse caso
        else tempaux = temp;  //Variavel auxiliar de memoria para o caso do processo ja ter sido iniciado
        flagprocandamento = true;
        lbl_ONOFF1.Text = "ON"; //mudando labels e imagens
        pictureBox2.Image = Simulador_de_Processo_Termico.Properties.Resources.simbolo_resistencias_quente;
    }

    private void timer1_Tick(object sender, EventArgs e)
    {  //aquecimento da resistencia
        if (tempres == 0)
        {
            temporizador += 1;
        }  //Aquecimento dado por uma função exponencial
        temp = tempaux + 800 * (1 - (Math.Exp(-temporizador / 2500))) - tempres;
        if (temp >= 800)
        {
            temp = 800;  //Caso temperatura superior a 800C, mostrar mensagem
            lbl_temp.Text = "Equipamentos com danos permanentes!";
        }
        else lbl_temp.Text = temp.ToString("000");
    }

    private void but_desmot_Click(object sender, EventArgs e)
    {
        timer1.Enabled = false;  //Caso processo seja desligado
        timer2.Enabled = true;
        but_desmot.Enabled = false;
        but_mot.Enabled = true;
        temporizador = temp;
        tempaux = temp;
        lbl_ONOFF1.Text = "OFF";

        tempres = 0;   //para entrar na conidção de aquecimento
        temporizador = 0; //esta linha e a de baixo para armazenar na memoria o valor de temperatura
        tempaux = temp;   //e reiniciar aquecimento

    }

    private void timer2_Tick(object sender, EventArgs e)
```

```
{   //esfriamento da resistencia
    temp = tempaux + temporizador - tempres;
    temporizador -= 0.2; //Função de refriamento (sem sistema de arrefecimento)

    if (temp <= 27)
    {
        temp = 0;
        temporizador = 000;
        flagprocandamento = false;
        lbl_temp.Text = "027";  //Voltar com labels e imagens caso temperatura seja a ambiente
        pictureBox2.Image = Simulador_de_Processo_Termico.Properties.Resources.simbolo_resistencias;
    }
    else lbl_temp.Text = temp.ToString("000");
}

private void but_acventoinha_Click(object sender, EventArgs e)
{ //Botão do sistema de arrefecimento
    timer3.Enabled = true;
    timer4.Enabled = false;
    but_acventoinha.Enabled = false;
    but_desventoinha.Enabled = true;
}

private void timer3_Tick(object sender, EventArgs e)
{  //acionamento da ventoinha
    //if ((timer1.Enabled == true) && (acaoresf < 3800) && (tempdes<temp))
    if ((acaoresf < 3800) && (tempdes < temp))
    {  //Calculos de refriamento pelo sistema de arrefecimento
        acaoresf = acaoresf + 23;
        pictureBox3.Visible = true;
        lbl_ONOFF2.Text = "ON";
        timer6.Enabled = true;
    }
    else
    {

        if (acaoresf >= 3800)
        {
            acaoresf = 3800; //Travando velocidade do ventilador
            lbl_ONOFF2.Text = "ON";
            timer6.Enabled = true;
        }

        else if (acaoresf <= 0)
        {
            acaoresf = 0;
            lbl_ONOFF2.Text = "OFF";
            timer6.Enabled = false;
        }
        else
        {
            timer3.Enabled = false;
            timer4.Enabled = true;
        }
    }

    lbl_rot.Text = acaoresf.ToString("0000");
}

private void but_desventoinha_Click(object sender, EventArgs e)
{  //Função para desabilitar sistema de arrefecimento
    timer3.Enabled = false;
    timer4.Enabled = true;
    but_acventoinha.Enabled = true;
    but_desventoinha.Enabled = false;
    timer6.Enabled = false;
    lbl_ONOFF2.Text = "OFF";
```

```
            tempres = 0;   //para entrar na conidção de aquecimento
            temporizador = 0; //esta linha e a de baixo para armazenar na memoria o valor de temperatura
            tempaux = temp;   //e reiniciar aquecimento
        }

        private void timer4_Tick(object sender, EventArgs e)
        {   //desligamento da ventoinha
            acaoresf = acaoresf - 9;
            if (acaoresf <= 0)
            {
                acaoresf = 0;
                pictureBox3.Visible = false;
            }
            else
            {
                pictureBox3.Visible = true;
            }
            lbl_rot.Text = acaoresf.ToString("0000");
        }

        private void timer5_Tick(object sender, EventArgs e)
        {    //Envio das informações para o supervisorio
            if (but_desventoinha.Enabled == true) botao_vent_acion = 1;
            else botao_vent_acion = 0;
            if (but_desmot.Enabled == true) botao_aque_acion = 1;
            else botao_aque_acion = 0;
            serialPort1.WriteLine("ba" + botao_aque_acion +"bv" + botao_vent_acion + "t" + lbl_temp.Text + "r" + lbl_rot.Text
+ "&%$");
        }

        private void timer6_Tick(object sender, EventArgs e)
        {    //ação de resfriamento da temperatura
            if (tempdes < temp)
            {
                tempres++;  //valor que subtrai do aquecimento

                timer3.Enabled = true;
                timer4.Enabled = false;
            }
            else
            {
                tempres = 0;   //para entrar na conidção de aquecimento
                temporizador = 0; //esta linha e a de baixo para armazenar na memoria o valor de temperatura
                tempaux = temp;   //e reiniciar aquecimento

                timer3.Enabled = false;
                timer4.Enabled = true;
            }

        }

    }
}
```

Apêndice B. Código do simulador de processo térmico em C#.