GAT125 - LABORATÓRIO INTEGRADOR

CONTROLE PID DE TEMPERATURA ATRAVÉS DA AÇÃO DE UM COOLER

GABRIEL HENRIQUE RIBEIRO DOS SANTOS
LUIS GUSTAVO DE SOUZA OLIVEIRA

Lavras - MG
28 de Novembro de 2021

Sumário

1. Introdução **3**
1.1 Objetivos 3
1.2 Fluxograma geral 4
1.3 Sobre hardware e software 4

2. Modelagem **5**
2.1 Sobre o experimento para aquisição de dados 6
2.2 Modelos obtidos usando o Ident Toolbox do MATLAB® 8
2.3 Simulações em malha aberta 10

3. Controle **11**
3.1 Metodologia para a escolha de parâmetros PID 11
3.2 Projeto de controle no espaço de estados 12
3.3 Comparação da malha fechada em simulação MATLAB® e planta física 14
3.3.1 Controlador PID 15
3.3.2 Controlador no espaço de estados 18
3.3.3 Comparação dos controladores ajustados 21

4. Conclusão **22**
4.1 Síntese do trabalho 22
4.2 Dificuldades 22

5. Referências bibliográficas **23**

6. Anexos **25**
6.1 Anexo A 25
6.2 Anexo B 27

1. Introdução

Seja na indústria ou na agropecuária, em vários processos de produção se faz necessário o controle de temperatura. Por exemplo na produção animal, o controle da temperatura de um confinamento é uma das variáveis a serem consideradas. O desempenho dos animais está associado não só a fatores genéticos, como também ao manejo, à nutrição e sanidade, que, por sua vez, estão correlacionados a fatores climáticos, especialmente temperatura ambiente e umidade relativa do ar.(CARVALHO et al., 2004) Na agricultura de cultivo protegido, isto é em estufas, a variação da temperatura está altamente correlacionada com o desempenho fisiológico das plantas. A ventilação de estufa é fundamental por três razões básicas: assegura uma taxa mínima de CO2 para a vegetação, evita o excesso de umidade bem como o calor excessivo durante o dia.(DO BRASIL, 2004) Porém nem sempre é possível contar com uma ventilação natural para o controle de temperatura, fazendo com que as instalações, de confinamento de animais ou estufas de plantas, utilizem de recursos mecânicos para uma ventilação forçada. Segundo Perdomo (1999), na produção animal em confinamento a adequação do ambiente deve ser constante, visando o bem-estar dos animais, sendo indispensável o uso de atuadores na climatização, como ventiladores e exaustores. Ou seja, o controle refinado da temperatura, com o objetivo de anular as variações decorrentes de distúrbios, leva a maximização da produção.

Além do controle da temperatura estar relacionada com o ganho de produtividade nos casos já mencionados, existe também uma outra boa razão para um controle variável, que é uma maior eficiência energética. Controles baseados em On/Off, podem consumir muita energia elétrica e provocam uma deterioração precoce dos componentes elétricos e mecânicos. A maioria das estufas disponíveis no mercado utiliza controle eletromecânico de temperatura, com um par bimetálico (termostato), constituído de duas lâminas justapostas, de diferentes materiais.(MAXGG et al., 2004) A utilização de técnicas de controles mais simples se justificam muitas vezes pelo baixo custo de implantação, porém nem sempre correspondem com as expectativas dos produtores, justamente por não se tratar de um controle variável.

Grande parte dos sistemas de controle e automação utilizados atualmente nas indústrias se baseiam na estrutura clássica de reguladores, como os do tipo proporcional, integral e derivativo (PID). (CARVALHO, 2010) Com o advento dos CLPs e sua popularização a partir da década de 70, os antigos controladores analógicos perderam espaço e a inserção dos métodos de controle digital nos processos industriais tornou-se uma realidade. Os métodos convencionais de controle analógico sofrem vários problemas, incluindo não linearidade no transdutor analógico de velocidade e dificuldade em transmitir com precisão o sinal analógico após ter sido obtido do transdutor. (MASON, 1964)

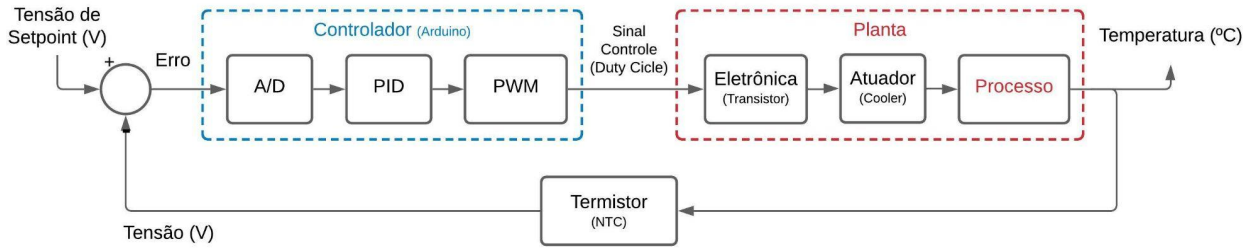
1.1. Objetivo

O objetivo deste trabalho é controlar a temperatura de um ambiente fechado, que tem um aquecimento interno promovido por uma resistência elétrica, através da ação de uma ventilação forçada gerada por um *cooler*. A lógica de controle será a de um controlador digital obtido por meio de técnicas de modelagem caixa preta, que por vez estará embarcado numa placa de desenvolvimento Arduíno UNO R3. O sistema será composto também por um termistor NTC para monitorar a temperatura, realimentando o sinal de leitura para fazer o cálculo do erro.

1.2. Fluxograma geral

A princípio, o sistema de controle de temperatura pode ser resumido a partir da figura 1.1, onde o Arduino UNO é encarregado de avaliar o sinal de erro e gerar o PWM (variável manipulada) correspondente ao sinal de controle. Serão apresentadas neste projeto duas metodologias para desenvolver os ganhos do controlador, um por meio do método PID (conforme a figura) e outro a partir da alocação de pólos no espaço de estados. O sinal de controle chaveia o transistor conectado em série ao *cooler* e por fim o processo é excitado para modificar a temperatura (variável controlada). Em malha fechada, o termistor apresenta um ganho de 0,05, onde o pino analógico do Arduino recebe o sinal em tensão (Volts) para processar e gerar o sinal de erro (em ºC).

Figura 1.1 - Fluxograma geral do projeto em Malha Fechada



Fonte: Autores do projeto (2021)

1.3. Sobre hardware e software

Tanto o sensor de temperatura quanto o circuito de acionamento do *cooler* serão conectados à placa de prototipagem Arduino UNO R3 com o objetivo de implementar uma malha fechada a partir da leitura e transmissão de dados do sensor de temperatura, algoritmo de controle no microprocessador da placa Arduino e o acionamento do circuito de potência do *cooler* com um sinais de saída PWM (*Pulse With Modulation*) para modificar a velocidade do atuador.

O Arduino é uma plataforma de computação física de fonte aberta, com base em uma placa simples de entrada/saída. (BANZI, 2014) Pode receber e enviar informações para a maioria dos dispositivos e até mesmo através do Internet para comandar outros dispositivos. (BADAMASI, 2014) A placa Arduíno utilizada neste projeto é da arquitetura UNO R3, que é baseada no microcontrolador ATmega328P.

A planta de estudo é composta por um motor CC de ímã permanente integrado a uma ventoinha (*Cooling Fan*) do modelo *FM802512M*. O componente apresenta dois terminais para alimentação e normalmente é encontrado em gabinetes de computadores antigos para arrefecimento, dificilmente é vendido separado no mercado pois o mesmo está fora de fabricação. Entretanto, um modelo semelhante *TianXuan TX8025L12S* apresenta características semelhantes, principalmente em relação a alimentação, número de aletas e dimensões, conforme as especificações da tabela 1.

Tabela 1.1 - Especificações do modelo *TX8025L12S*

Tamanho	80 x 80 x 25 mm
Tensão	12 V

Potência	1.68 W
Corrente	0.14 A
Volume de ar	30 cfm
Intensidade de ruídos	25 dba
Nº aletas	7
Velocidade	2400 rpm

O sensor de temperatura utilizado é um termistor do tipo NTC (*Negative Temperature Coefficient*). O termistor é um componente cuja resistência elétrica entre seus terminais tem relação com a temperatura e com a sua estrutura, ou seja, é um resistor cuja resistência elétrica varia de acordo com a temperatura do mesmo. (SILVA, 2011) Operando em faixas de temperatura que vão de valores negativos até aproximadamente 125ºC, esses dispositivos são utilizados como sensores em uma grande quantidade de aplicações, dada a facilidade com que podemos trabalhar com eles e inclusive seu baixo custo. (WENDLING, 2010) Como o próprio nome já sugere, um termistor do tipo NTC é um termistor cuja a sua resistência diminui enquanto a temperatura aumenta.

Tabela 1.2 - Especificações do sensor NTC modelo MF52

Tamanho	2.5 x 4 x 3 mm
Tensão aplicada	0 à 5 V
Resistência	10k à 0 Ω
Faixa detecção	-55 à 125°C
Tolerância	±1%
Potência	50 mW

2. Modelagem

Para que seja desenvolvido um controlador de forma corresponder com as nossas necessidades, primeiramente é necessário a modelagem da planta. São raras as situações em que se tem informações relativas ao sistema eletromecânico de motores, sendo a posse deste tipo de informação muito restrito a quem fabrica estes equipamentos. (GUSTAVSEN e SEMLYEN, 1998; SILVA, 2014) Devido a isso, a modelagem matemática do equipamento nem sempre pode corresponder a realidade já que não é de conhecimento do usuário todos os parâmetros do sistema físico. No caso de motores, a modelagem matemática consiste em representar o sistema como um circuito RL equivalente. A denominação “caixa branca” é utilizada para designar os métodos de modelagem em que se conhecem a fundo as relações matemáticas que descrevem o comportamento dinâmico do sistema que se pretende representar (AGUIRRE, 2007). Diante deste problema, se justifica o uso da modelagem caixa preta que consiste em realizar medições dos sinais de entrada e saída de um sistema. Em teoria de sistemas, a denominação “caixa preta” se refere aos métodos de modelagem em que a única fonte de informação a partir da qual um modelo é construído é a relação causa-efeito apresentada pelos sinais de entrada e saída do sistema modelado (AGUIRRE, 2007).

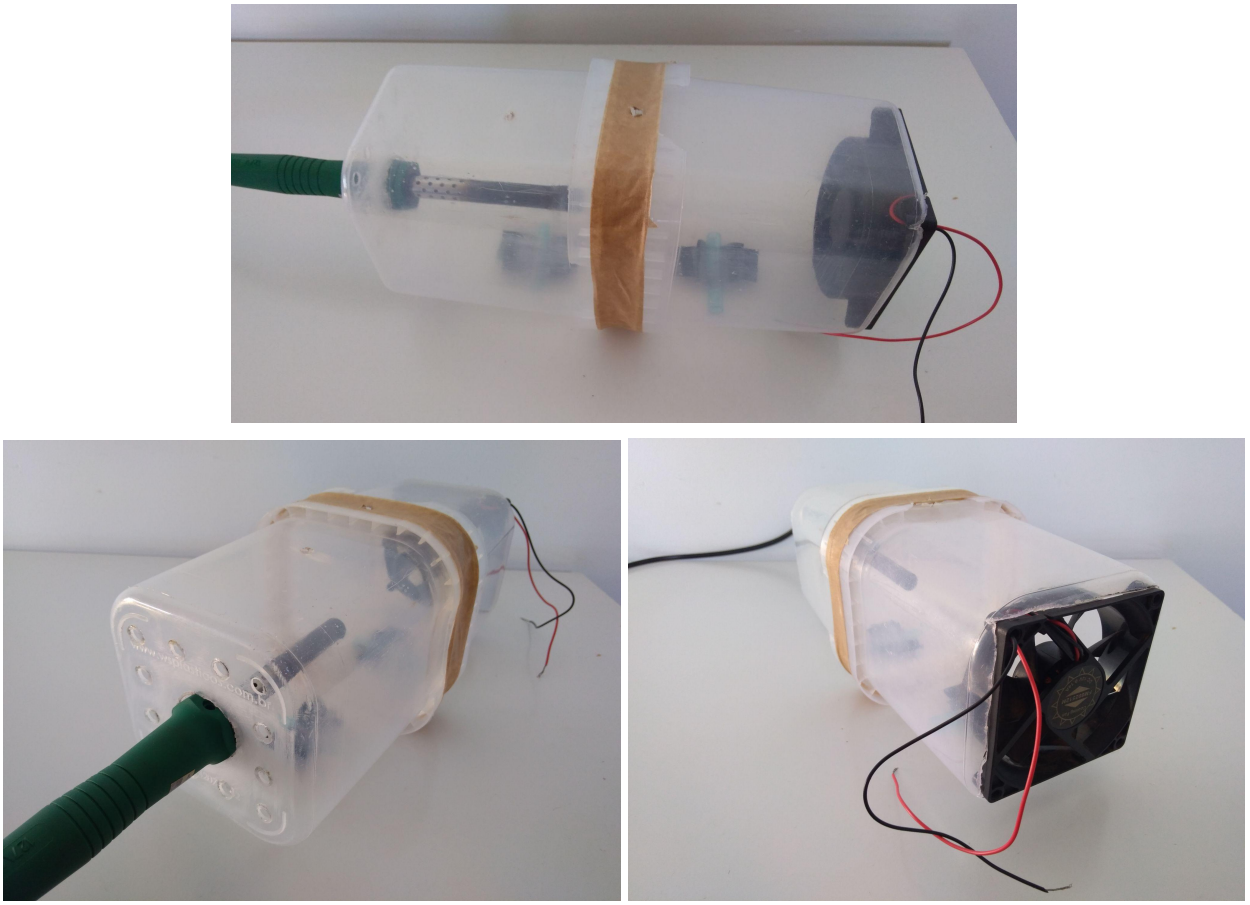
2.1. Sobre o experimento para aquisição de dados

O protótipo que foi implementado para o controle de temperatura pela ação do cooler baseou-se na ideia das várias aplicações reais que existem em campo, como por exemplo em estufas, confinamentos fechados, aviários “*dark house*”, entre outros, porém com o objetivo de desenvolver uma estrutura que possa garantir um sistema de aquecimento e arrefecimento eficientes e com boa performance em termos de flexibilidade para a coleta dos dados e portabilidade (LEVARDA, 2010).

Sendo assim, este tópico foca em demonstrar a estrutura do protótipo, bem como todos os seus componentes mecânicos e eletrônicos, além também dos resultados obtidos para a metodologia empregada na coleta dos dados e seu tratamento. Análises do comportamento do sistema também serão introduzidas para posterior estimação do sistema em um modelo “caixa-preta” e implementação de uma estratégia de controle.

A estrutura física é composta por um recipiente plástico que suporta temperaturas de até 150ºC, no qual em uma de suas extremidades foi colocado uma resistência de 60w com temperatura fixa para manter o sistema aquecido, e na outra extremidade o *cooler* disposto como exaustor. Após vários testes, observa-se que a forma como o *cooler* era introduzido no sistema (ventilador ou exaustor), influenciava no comportamento da temperatura, sendo que como exaustor a taxa de variação de temperatura era maior. Alguns furos para entrada de ar foram feitos a fim de evitar a geração de vácuo dentro do recipiente.

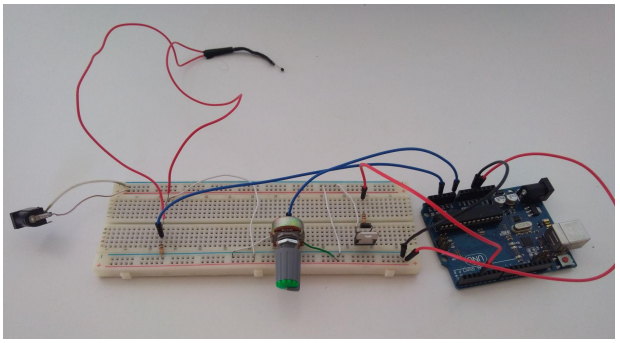
Figura 2.1 - Protótipo usado para a coleta dos dados (lateral e extremidades)



Fonte: Autores do projeto (2021)

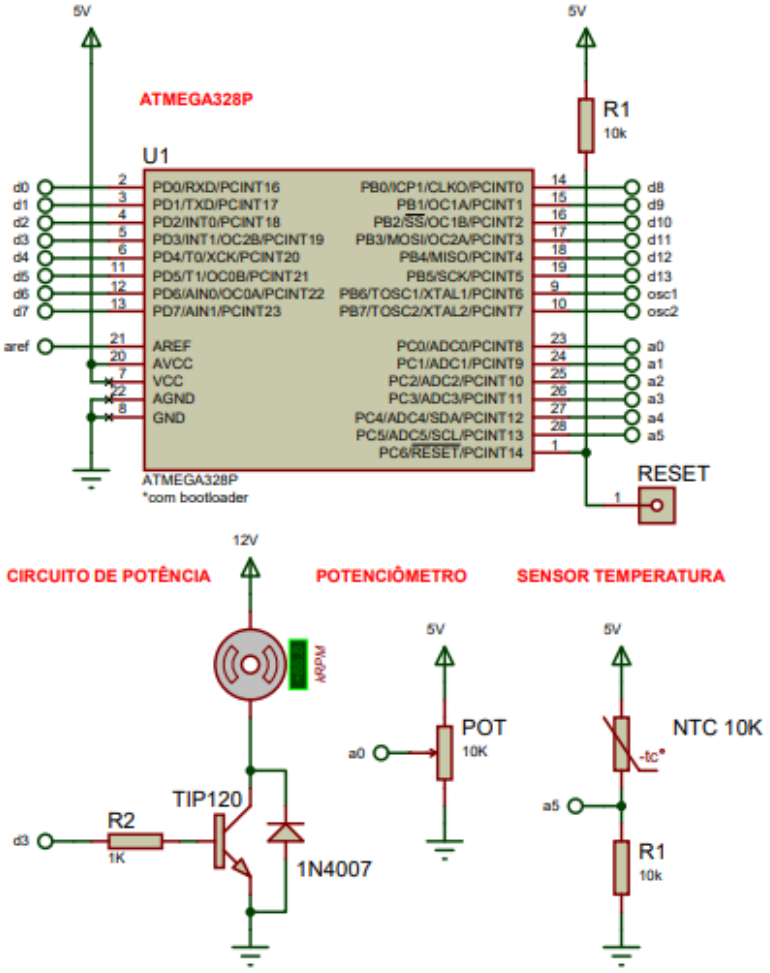
Em relação ao circuito eletrônico, foi implementado um pequeno circuito para o sensor de temperatura, além de um potenciômetro para variar o ciclo de trabalho do PWM que energiza a base do transistor TIP120. O PWM é gerado pelo Arduíno UNO em intervalos de 0 a 255 bits ou de 0% a 100%, indicando respectivamente o motor DC do *cooler* parado e em sua velocidade máxima. Inicialmente a proposta do potenciômetro é gerar os sinais de entrada para a planta do sistema e obter na saída a temperatura no seu regime transitório e permanente em malha aberta.

Figura 2.2 - Montagem do circuito eletrônico do protótipo



Fonte: Autores do projeto (2021)

Figura 2.3 - Esquemático do circuito eletrônico



Fonte: Proteus (2021)

Conforme proposto por Levarda e Budaciu (2010), a aquisição dos dados foi realizada partindo da resistência elétrica alimentada externamente por 127V e já aquecida (ou seja, com condição inicial não nula) entre 120ºC e 140ºC com um intervalo de amostragem de 0.05 segundos (20Hz). O microcontrolador do Arduino UNO envia um sinal PWM para o TIP120 comandando a velocidade de giro do motor DC do *cooler*, assim buscando ajustar a temperatura do ar no recipiente em um set point requerido, no qual será posteriormente refinado por um controlador em malha fechada. Por meio do Arduino UNO as informações de momento da coleta, ciclo de trabalho do PWM de 0 a 255 e temperatura em ºC são coletados e enviados diretamente a uma planilha do Excel para registro em banco de dados. Posteriormente o MATLAB® foi usado para tratamento dos mesmos.

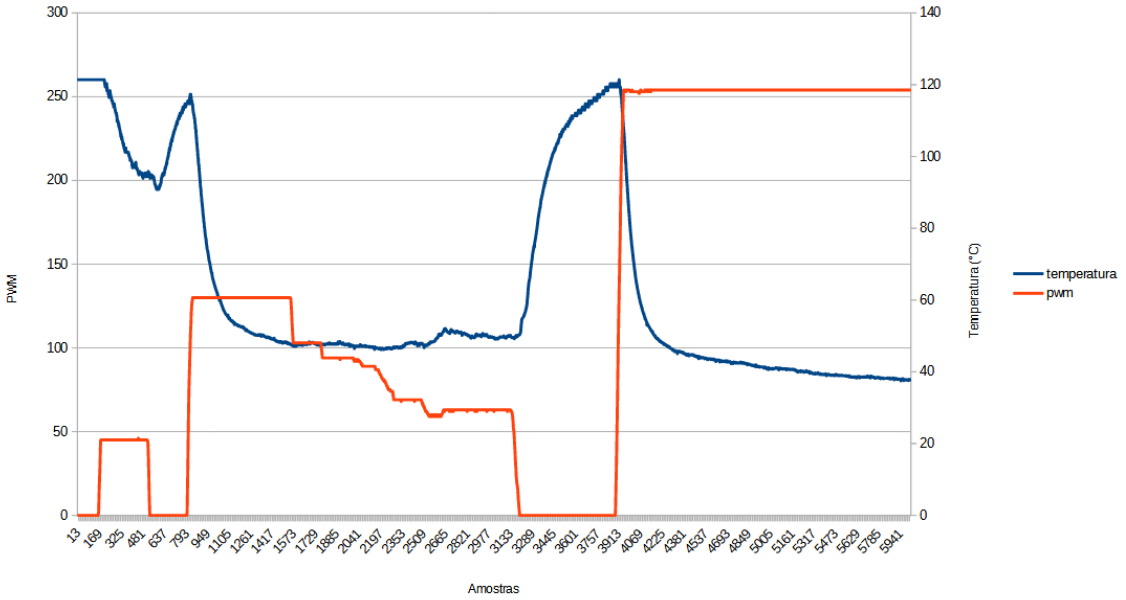
2.2. Modelos obtidos usando o Ident Toolbox do MATLAB®

Para identificação da função de transferência que relaciona a temperatura com o valor de PWM como entrada-saída da planta, foi utilizado o System Identification Toolbox do MATLAB®. Dentro desta ferramenta é possível:

- Analisar os dados de entrada e saída do experimento;
- Escolher uma estrutura para a modelagem (funções de transferência, espaço de estados, ARX, redes não lineares);
- Estimar um modelo a partir dos dados coletados, selecionando uma estrutura que melhor encaixa na situação (*fit*);
- Validar o modelo matemático com os dados experimentais.

Primeiramente foi feito o carregamento da base de dados adquirida com as medições realizadas no protótipo. Foi considerado como *input* do sistema da planta os valores de PWM, enquanto os *output* foram os valores de temperatura em °C. As amostras foram coletadas com um período de amostragem de 0,05 segundos. A figura 2.4 apresenta o resultado de um dos experimentos realizados.

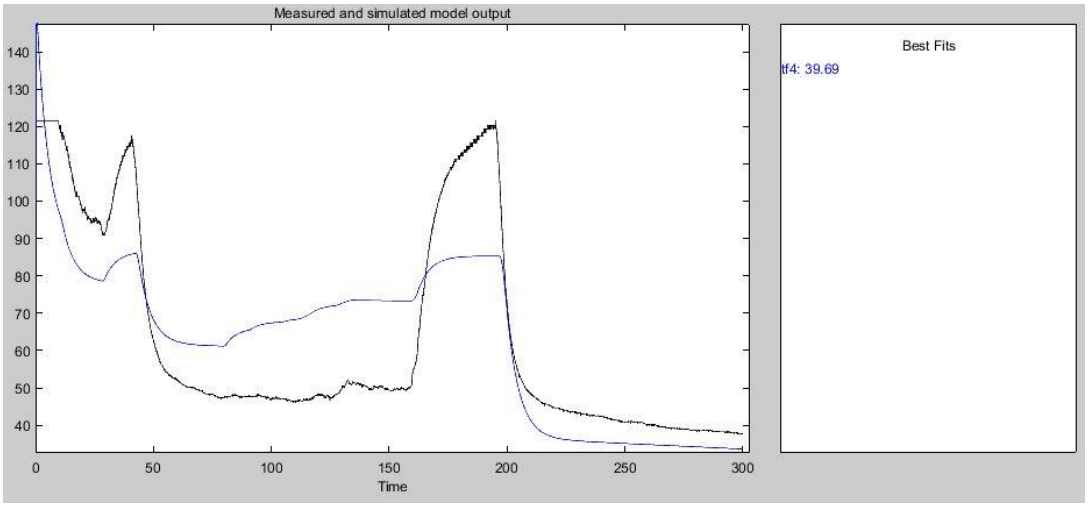
Figura 2.4 - Dados coletados a partir de um experimento



Fonte: Toolbox Ident MATLAB® (2021)

O tipo de modelo utilizado na identificação foi uma função de transferência e no espaço contínuo. Foram testadas várias combinações de quantidade de zeros e pólos, de forma a achar uma função de transferência que tivesse a maior percentagem de *fit* com os dados de entrada-saída, isto é, que melhor se ajustasse aos dados. A função de transferência que teve o maior índice de *fit* foi a função de transferência parametrizada com 3 polos e 1 zero. O ajuste foi de 39,69%, percentagem relativamente baixa.

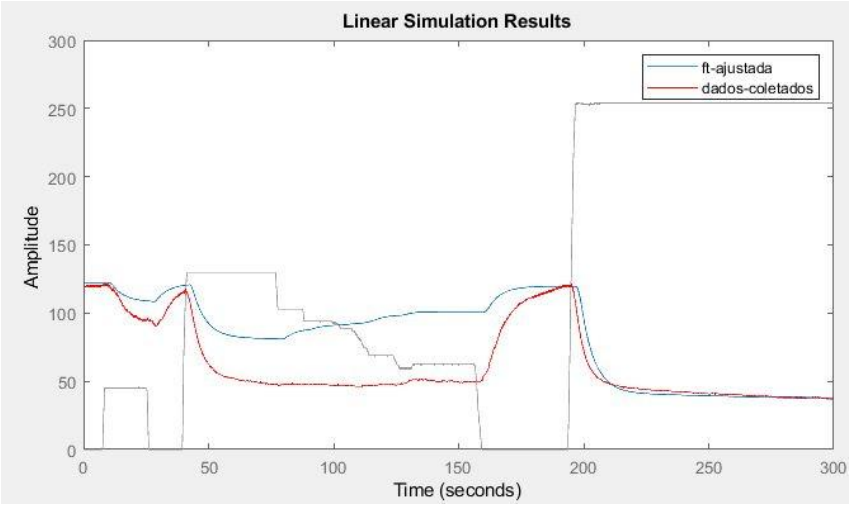
Figura 2.5 - Melhor ajuste encontrado para a relação entrada-saída



Fonte: Toolbox Ident MATLAB® (2021)

Graficamente é possível observar que de fato a função de transferência identificada não se ajusta corretamente aos dados, principalmente em relação a amplitude das respostas e ao deslocamento vertical, porém os transitórios foram mapeados corretamente. Um outro motivo que justifica o baixo ajuste é que as amostras coletadas possuem alguns comportamentos que teoricamente não seriam esperados, como por exemplo o fato da temperatura se manter praticamente constante quando os valores de PWM estão diminuindo. Para amenizar a falta de ajuste, a função de transferência obtida foi modificada empiricamente com multiplicação de ganho e conversão do sistema para espaço de estados com uma matriz de estados iniciais manualmente ajustada. O modelo final foi um sistema que se assemelha bastante aos dados experimentais, com exceção dos comportamentos inesperados já mencionados.

Figura 2.6 - Resposta dos modelos obtidos



Fonte: Toolbox Ident MATLAB® (2021)

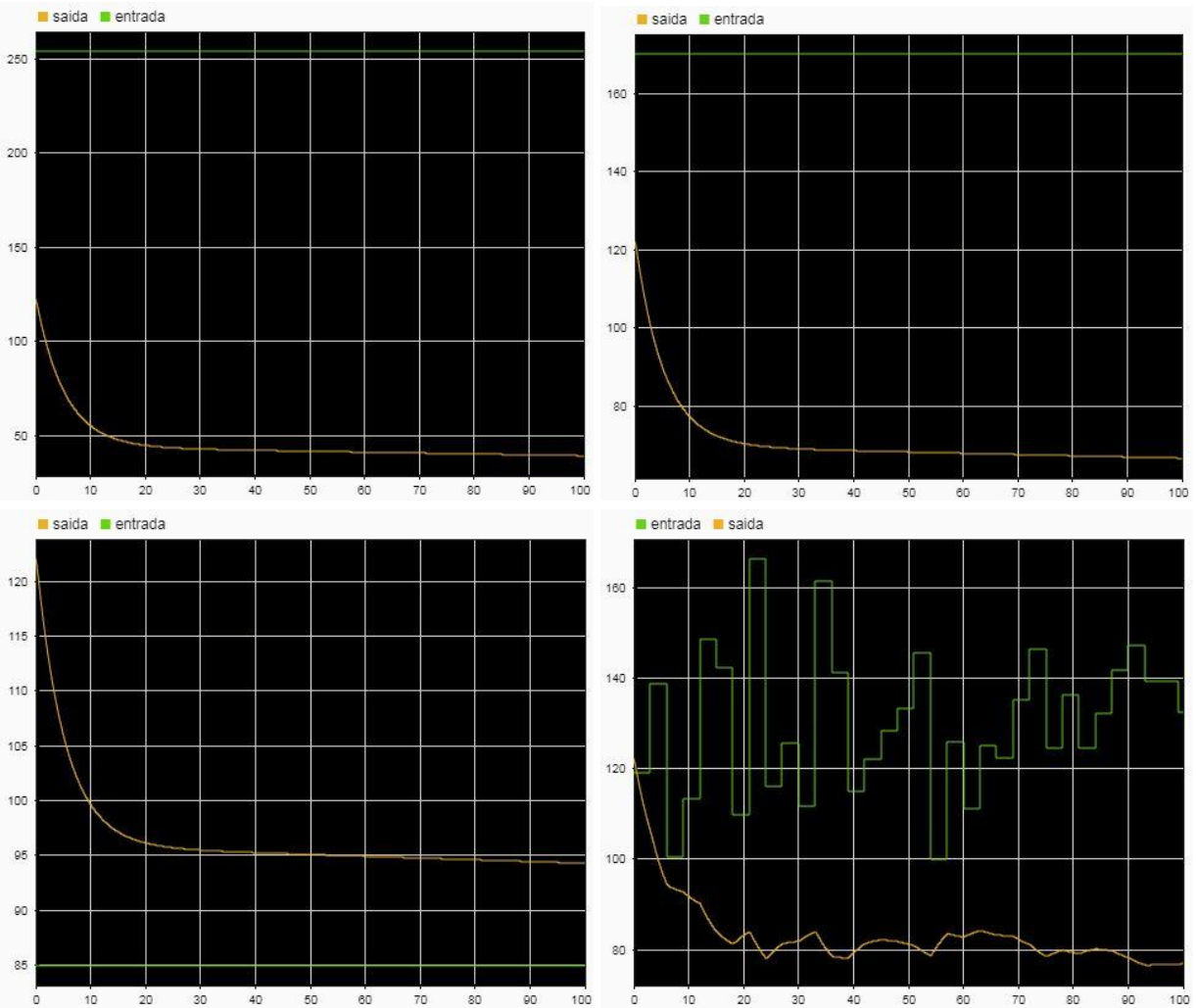
A função de transferência a seguir representa o modelo final:

$$H(S) = e^{-2,9 \cdot s} \frac{-0,4369 \cdot s - 0,0002663}{s^3 + 7,586 \cdot s^2 + 1,421 \cdot s + 3.757 \cdot 10^{-10}}$$

2.3. Simulações em malha aberta

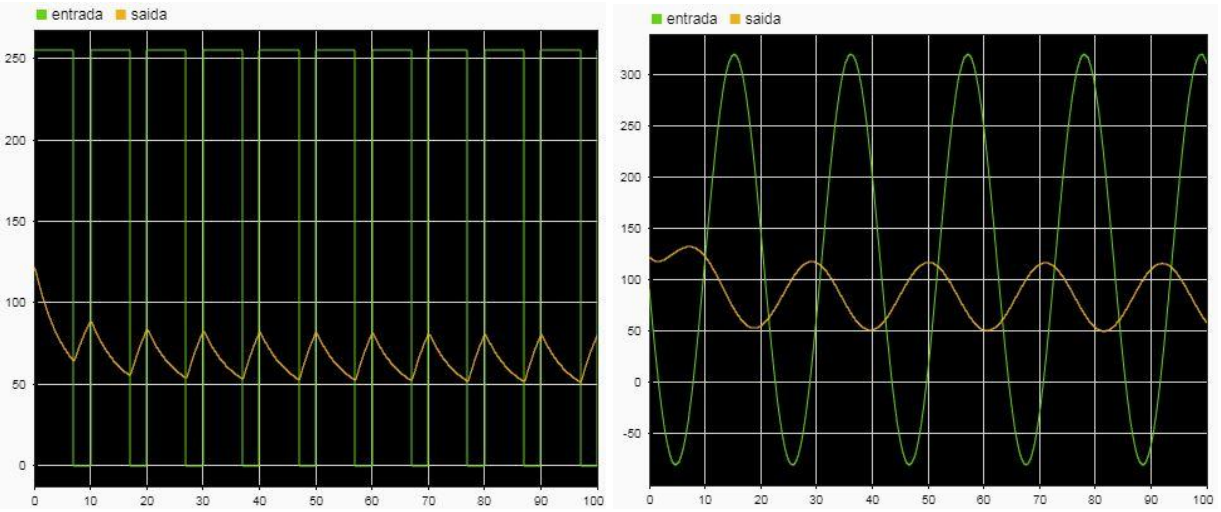
Com o modelo encontrado foram feitas algumas simulações em malha aberta a fim de avaliar a dinâmica da resposta do sistema. Foram considerados alguns tipos de sinal de entrada como resposta ao degrau, com três valores diferentes, resposta ao sinal pulsado (onda quadrada) e resposta ao sinal aleatório.

Figura 2.7 - Resposta do modelo estimado a entradas diversas (a) degrau em 255, (b) degrau em 150, (c) degrau em 85, (d) sinal aleatório



Fonte: Toolbox Simulink MATLAB® (2021)

Figura 2.7 - Resposta do modelo estimado a entradas diversas (e) sinal pulsado, (f) sinal senoidal



Fonte: Toolbox Simulink MATLAB® (2021)

3. Controle

3.1. Metodologia para a escolha de parâmetros PID

O primeiro compensador projetado foi baseado na técnica de controle proporcional com ação integral para corrigir o erro em regime permanente e derivativa para melhorar a resposta transitória em questão da ultrapassagem percentual e tempo de assentamento. Seu dimensionamento parte da ideia do lugar geométrico das raízes, sendo necessário ajustar 3 parâmetros para alcançar a dinâmica desejada: o ganho proporcional "K_1", o ganho integrativo "K_2" e o ganho derivativo "K_3".

Existem algumas abordagens para determinar os ganhos, sendo que para o projeto foram realizadas simulações por tentativa e erro, visto a dificuldade de encontrar os parâmetros a partir de formulações como de Ziegler e Nichols para a obtenção dos ganhos iniciais de cada etapa do compensador (OGATA, 1987). Portanto, o método de tentativa e erro foi suficiente para obter um ajuste fino e atingir os requisitos desejados.

Os seguintes passos foram seguidos para dimensionar o controlador (NISE, 2017):

1) Avaliar a performance do sistema sem compensação, para identificar os requisitos do sistema;
2) Dimensionar o controlador PD para atender as especificações da resposta transiente, incluindo o zero e o ganho em malha fechada;
3) Dimensionar o controlador PI para atender aos requisitos de erro em regime permanente;
4) Determinar os ganhos K_1 ou K_P , K_2 ou K_i e K_3 ou K_D e a função de transferência para o controlador PID;
5) Simulação do sistema com o controlador para verificar os requisitos desejados;

6) Durante as etapas 2 e 3, caso os requisitos não forem individualmente atingidos, o controlador PI ou PD será redimensionado.

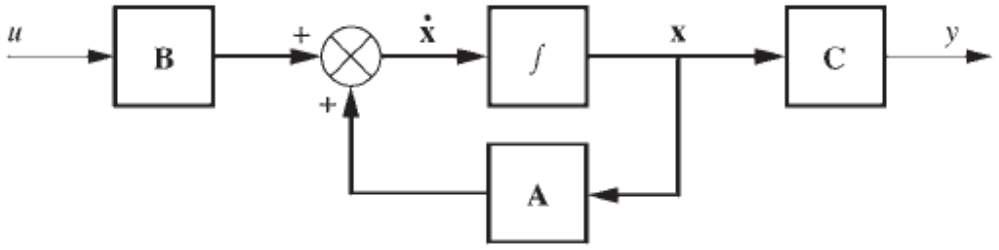
O objetivo era obter uma resposta que tivesse um tempo de assentamento (Ts) de aproximadamente 15 segundos e com máximo de 1% de ultrapassagem percentual (%UP). Depois de algumas tentativas e com auxílio da interface gráfica do PID Tuner do MATLAB®, foram encontrados os seguintes valores de ganhos:

$$K_P = -15$$
$$K_I = -1$$
$$K_D = -0.1$$

3.2. Projeto de controle no espaço de estados

Para ser feito o dimensionamento do controlador pelo método de Ackermann primeiramente é necessário achar as matrizes que representam o sistema no espaço de estados. Esta tarefa é facilmente realizada através do comando *ss()* do MATLAB®. O diagrama de blocos correspondente ao sistema representado no espaço de estados é mostrado a seguir.

Figura 3.1 - Diagrama de blocos geral de modelo em espaço de estados



Fonte: NISE (2017)

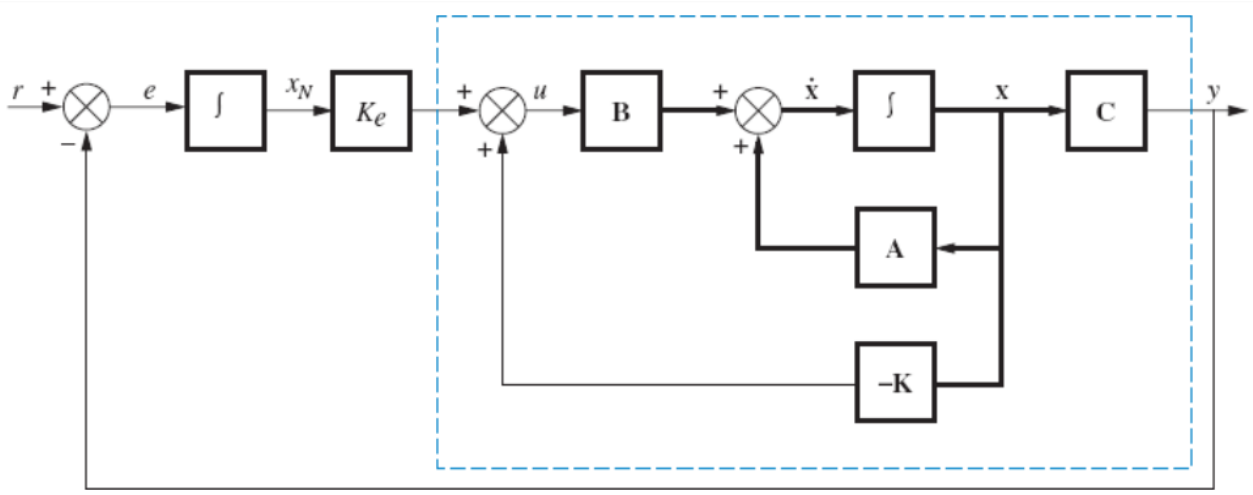
A premissa para que o método de Ackermann funcione é garantir que o sinal de controle pode controlar o comportamento de cada variável de estado, isto é, que a realimentação de estado seja efetiva na estabilização das variáveis. Caso contrário, será impossível de alocar os pólos nos lugares desejados. Uma maneira de caracterizar se o sistema é controlável ou não, é através da matriz de controlabilidade e de seu posto. Através do comando *ctrb()* e *rank()* é possível obter estas informações.

```
Cm =

    0.5000   -3.7930   28.0628
         0    0.5000   -3.7930
         0         0    0.5000

Rank =

     3
```

Como observado no console do MATLAB® o posto da matriz de controlabilidade é 3, igual a ordem do sistema. Com isso conclui-se que o sistema permite a alocação dos polos como forma de controlar as variáveis de estado.

Antes de fazer o controlador foram estabelecidos os requisitos de projeto a serem atendidos. Para ser feito uma análise da efetividade da alocação de pólos utilizando o método de Ackermann, foi definido que os requisitos seriam: tempo de assentamento (Ts) de 5 segundos e ultrapassagem percentual (%UP) de 1% (*overshoot*). A motivação por trás da escolha de um tempo de assentamento tão pequeno é que já se espera uma resposta mais lenta do que a projetada, o objetivo é que o tempo de assentamento não seja tão maior que 15 segundos, de forma a ter uma resposta equiparável ao do controlador PID projetado anteriormente. Além disso, deseja-se que o erro em regime permanente tenda a zero, ou seja, um controlador no espaço de estados com ação integral. O integrador aumenta a ordem do sistema, fazendo com que tenha um pólo adicional para ser alocado.

Figura 3.2 - Diagrama de blocos geral para sistemas em espaço de estados em malha fechada com ação integral



Fonte: NISE, 2017

Para atingir os valores de Ts e %UP desejados, primeiramente é feito o cálculo de quais são os polos dominantes de uma aproximação de segunda ordem que geram estes comportamentos requisitados. A localização dos polos está em função do fator de amortecimento e da frequência natural, como mostrado na seguinte equação:

$$p = -\zeta \cdot \omega_n \pm \omega_n \sqrt{1-\zeta^2}$$

O próximo passo é definir os valores do fator de amortecimento e da frequência natural, que para uma aproximação de segunda ordem são calculados da seguinte forma:

$$\%UP = e^{\frac{-\zeta \cdot \pi}{\sqrt{1-\zeta^2}}} \cdot 100\% \qquad \omega_n = \frac{4}{\zeta \cdot T_s}$$

Como o sistema a ser estudado é de quarta ordem (o integrador aumenta a ordem do sistema), isso significa que é necessário garantir que uma aproximação de segunda ordem seja a melhor possível.

> *"Justifique sua hipótese de segunda ordem determinando a posição de todos os polos de ordem superior e avaliando o fato de que eles estão muito mais afastados do eixo jω do que o par de segunda ordem dominante. Como regra prática, este livro considera um fator de cinco vezes mais afastado."*
>
> *(NISE, 2017)*

Desta forma, foi definido que o terceiro polo do sistema será alocado numa posição 5 vezes mais longe que a parcela real do par de pólos dominantes, garantindo assim que a aproximação de segunda ordem seja eficiente.

Segundo Nise (2017), é necessário levar em consideração os efeitos causados pelos zeros na resposta transitória do sistema, de forma que estes não interfiram no projeto. Uma hipótese, que deve ser verificada, é que os zeros do sistema em malha fechada são os mesmos do sistema da planta em malha aberta. Através da função *zero()* foi possível obter a localização do zero do sistema, que está sobre o eixo real em $-6{,}0965 \cdot 10^{-4}$. Para que os possíveis efeitos causados por esse zero sejam anulados, o quarto pólo deve ser alocado na mesma posição dele.

Posições de alocação dos polos
-0,8 + 0,5458j
-0,8 - 0,5458j
-4
-0,00060965

Com a controlabilidade do sistema verificada e com os pólos desejados já calculados, foi utilizado a função *acker()*. Nesta função foi passado como argumento as matrizes A e B do sistema em espaço de estados, e um vetor contendo os 4 pólos que satisfazem os requisitos de projeto. A função retorna o vetor de ganhos K para a realimentação de cada variável de estado.

$$K = [-3{,}9706 \quad 11{,}8414 \quad 0{,}0090 \quad -8.5872]$$

Onde os três primeiros valores do vetor correspondem aos ganhos das variáveis de estado do sistema e o último valor refere-se ao ganho de estado do integrador.

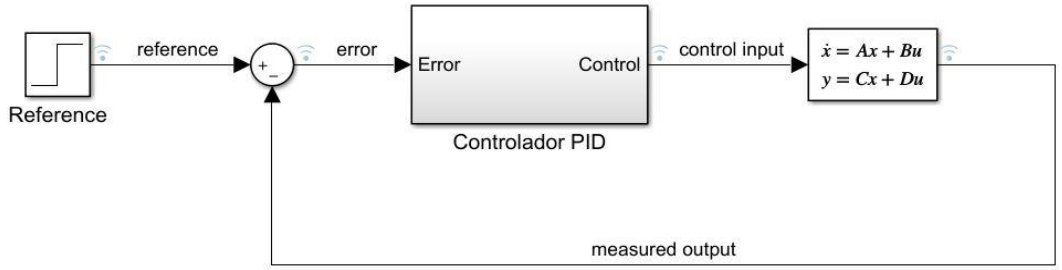
3.3. Comparação da malha fechada em simulação MATLAB® e planta física

Após ter encontrado os parâmetros de projeto para os dois tipos de controladores, foram realizadas simulações no Simulink com condições iniciais não nulas. A simulação teve como objetivo realizar a validação dos projetos, ou seja, se atenderam aos critérios de desempenho (**tempo de assentamento aproximadamente 15 segundos e ultrapassagem percentual menor que 1%**). Além disso, os resultados obtidos na simulação serviram para comparar com os resultados da implementação de cada controlador no sistema físico.

3.3.1. Controlador PID

A topologia utilizada na simulação do controlador PID foi a seguinte:

Figura 3.3 - Topologia implementada com o controlador PID

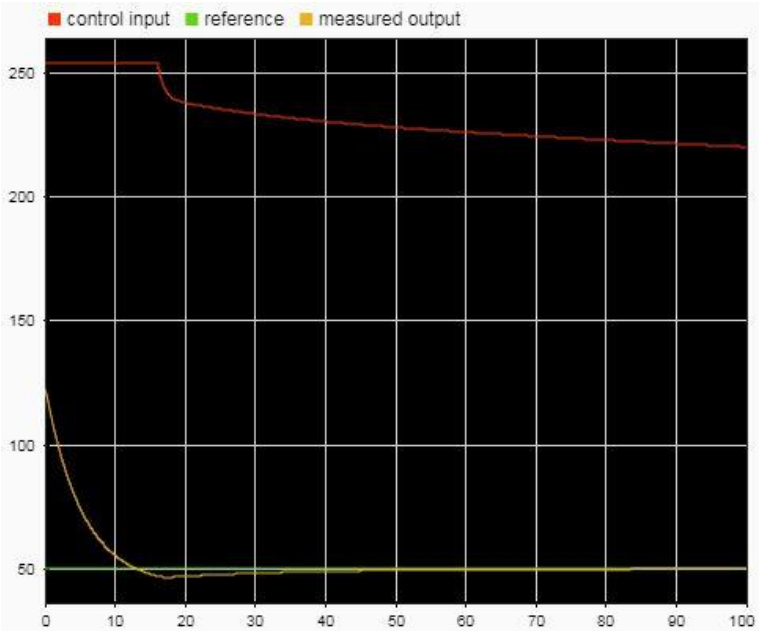


Fonte: Toolbox Simulink MATLAB® (2021)

Onde dentro do bloco do controlador está inserido um bloco PID padrão do Simulink, e no bloco do sistema em espaço de estados estão as matrizes do sistema que representa a planta. O bloco PID do Simulink fornece muita facilidade para a implementação do controle PID, através de uma interface gráfica intuitiva e com configurações muito úteis, como é o caso dos limites de valores máximos e mínimos que o controlador pode gerar. Como a entrada da planta foi modelada para receber valores de PWM, os limites para o sinal do controlador foram configurados para operar de 0 a 255. Além disso, a outra configuração necessária foi a dos ganhos proporcional, integrativo e derivativo.

Foi realizada uma simulação com um valor de *setpoint* de 50ºC e a resposta do sistema foi satisfatória, ocorrendo a diminuição da temperatura de 120ºC até ao valor de referência em torno de 15 segundos, com um leve *undershoot*. Após o transitório, a temperatura atingida no regime permanente foi igual a temperatura de *setpoint*, ou seja, o erro tendeu a zero.

Figura 3.4 - Simulação em software do sistema com controlador PID projetado

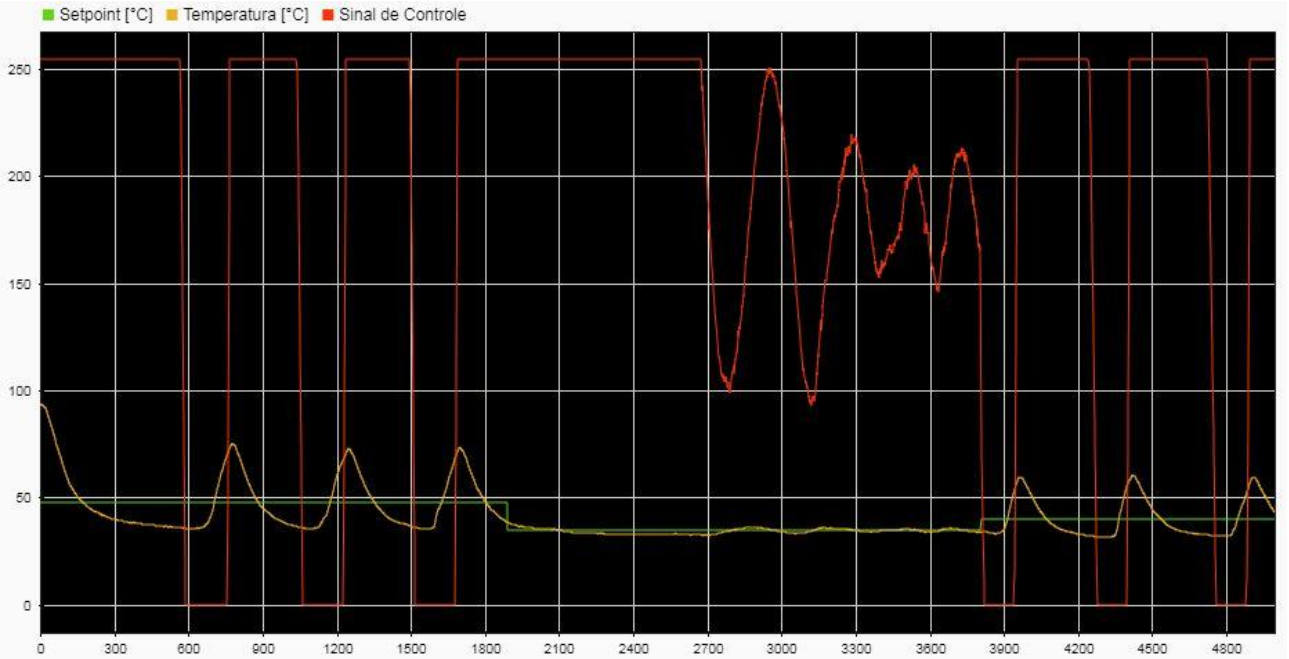


Fonte: Toolbox Simulink MATLAB® (2021)

Com o ótimo resultado obtido na simulação, foi implementado no Arduino a programação responsável por fazer o controle de temperatura do sistema. O código do controlador no espaço de estados está disponível no Anexo B deste relatório.

Apesar do resultado obtido na simulação estar condizente com os requisitos do projeto, quando o mesmo controlador foi implementado no sistema físico os resultados não ficaram próximos do que foi simulado ou esperado. Inicialmente foi ajustado um valor de *setpoint* de 50ºC, que foi o mesmo valor utilizado na simulação, porém foi observada uma grande dificuldade do controlador em manter a temperatura do sistema no valor desejado. A resposta gerada foi oscilatória e muito mais lenta do que a simulada. Foi verificado também que o sinal de controle gerado pela equação de controle PID estava saturando o PWM, significando que o *cooler* atuou com tudo ou nada (liga-desliga).

Figura 3.5 - Simulação do sistema físico com controlador PID projetado
Tempo de amostragem = 0,05 segundos, eixo horizontal em número de amostras



Fonte: Toolbox Simulink MATLAB® (2021)

Quando submetido a um valor de *setpoint* um pouco menor do que 50ºC, o sistema respondeu bem melhor, fazendo com que a temperatura ficasse bem próxima da referência. E nessa ocasião foi observado como a amplitude do sinal de controle variou dentro da faixa de valores do PWM (0-255), sem que este ficasse saturado. Após um leve aumento do valor de *setpoint*, a temperatura medida voltou a oscilar como no início do teste, e o sinal de controle voltou a saturar.

O fato do sinal de controle ter se apresentado com amplitudes tão abruptas fez com que fosse levantada a hipótese de que o controle estava agressivo demais. Portanto, a proposta foi inserir uma constante de "Agressividade" *C*, basicamente esta constante foi um fator de redimensionamento para fazer com que os ganhos do controlador PID fossem modificados de forma proporcional. Observando os valores encontrados anteriormente para Kp, Ki e Kd, um intervalo de 10 a 20 foi definido para a constante *C* sendo que o melhor resultado durante as simulações (para condições iniciais não nulas) foi o valor 15. Sendo assim:

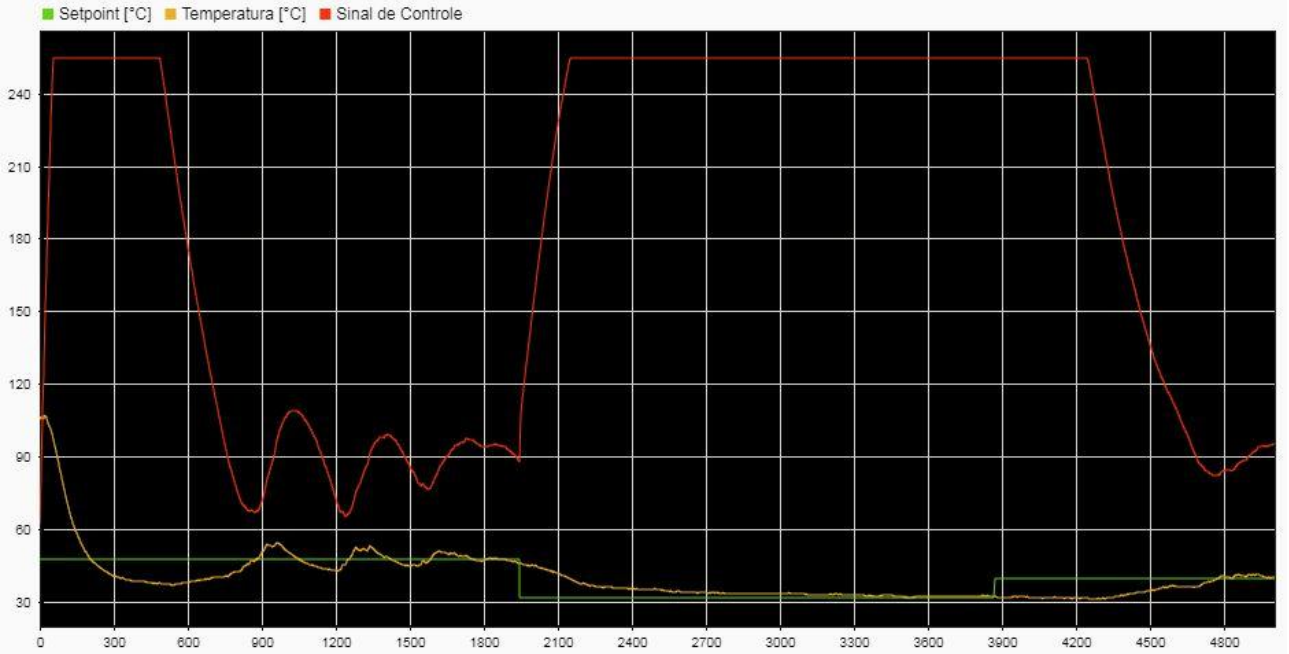
$$kn = \frac{kn}{C}, \quad C = 15$$

Tabela 3.1 Ganhos ajustados

Ganhos	Calculados (*Kn*)	Ajustado (*Kn / C*)
Proporcional (*Kp*)	-15	-1
Integral (*Ki*)	-1	-0.067
Derivativo (*Kd*)	-0.1	-0.0067

Observa-se nas figuras 3.5 e 3.6 o resultado do ajuste dos ganhos. Na figura 3.6 é notável que o tempo de assentamento do sinal aumentou, quando este passa do setpoint de 50ºC para 35ºC sendo aproximadamente 52,5 segundos (1050 x 0.05 - número de amostras vezes o tempo de amostragem) e também de 35ºC para 40ºC, aproximadamente 51,5 segundos (1030 x 0.05 - número de amostras vezes o tempo de amostragem). Além disso, o sinal apresentou poucas oscilações, no qual as existentes são provenientes de distúrbios do ambiente externo e ruídos do sensor, e estes são corrigidos pelo controlador. Ainda na figura, durante o setpoint de 35ºC o sinal de controle ainda satura o PWM, e isso foi de certa forma esperado pois durante os testes na planta física e pela temperatura ambiente (em torno de 30ºC), a temperatura dentro do recipiente tinha tendência de se estabilizar em 32ºC, portanto o controlador saturava o PWM e a integral do erro era sempre somada até que o setpoint fosse alcançado.

Figura 3.6 - Simulação do sistema físico com controlador PID ajustado
Tempo de amostragem = 0,05 segundos, eixo horizontal em número de amostras

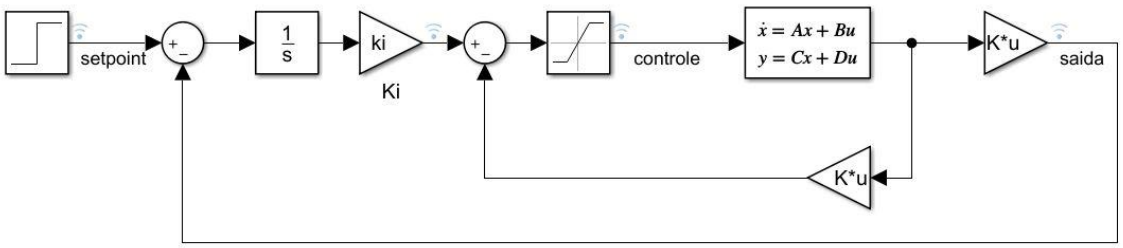


Fonte: Toolbox Simulink MATLAB® (2021)

3.3.2. Controlador no espaço de estados

Para a simulação do controlador no espaço de estados foi utilizada a seguinte topologia no Simulink:

Figura 3.7 - Topologia implementada com o controlador por alocação de pólos

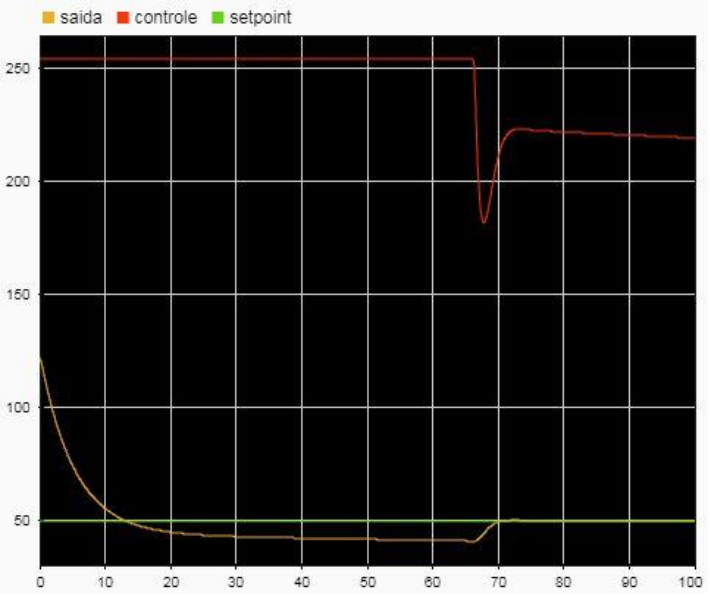


Fonte: Toolbox Simulink MATLAB® (2021)

Neste caso também foi necessário implementar uma forma de limitar os valores do sinal de controle recebidos pelo bloco de espaço de estados. Foi utilizado um bloco de saturação, configurado para limitar o sinal de controle entre 0 e 255, da mesma forma como é feito automaticamente pelo bloco PID mencionado na simulação anterior.

Assim como realizado na simulação do controlador PID, o valor de *setpoint* utilizado foi de 50ºC. A resposta do sistema não atendeu perfeitamente aos critérios de desempenho projetados, mas de certa forma o ajuste da temperatura se deu de forma rápida, sendo que a temperatura cruzou o valor de referência em menos de 15 segundos. O problema apresentado foi que o *undershoot* se manteve por um longo período de tempo até que a ação integral do sistema conseguisse zerar o erro em regime permanente. Além disso, foi observado que o sinal de controle gerado pelo controlador nos instantes iniciais atua no sistema de forma total, isto é, com valor de PWM no limite máximo.

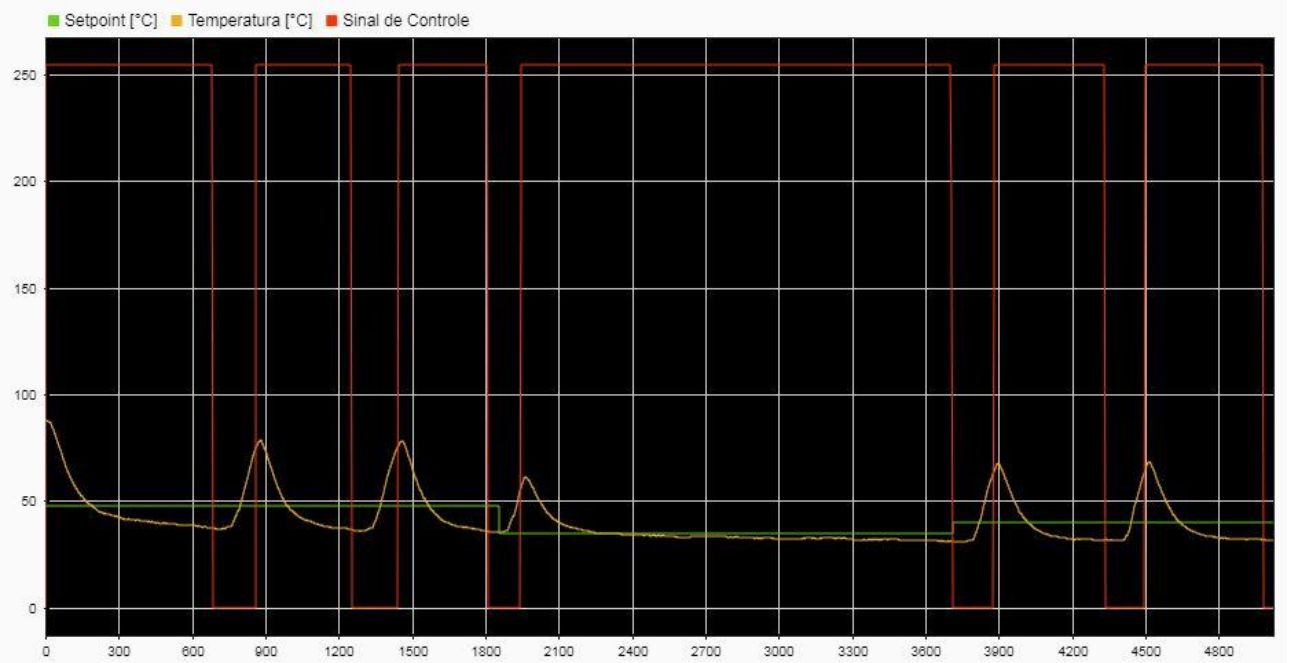
Figura 3.8 - Simulação em software do sistema com controlador por alocação de pólos projetado



Fonte: Toolbox Simulink MATLAB® (2021)

O código implementado no Arduino se encontra no Anexo A. Apesar da resposta aceitável obtida pela simulação, novamente durante a implementação do controlador no sistema físico os resultados não foram parecidos com os obtidos pela simulação. Assim como aconteceu durante a implementação do controlador PID, houve uma grande oscilação da temperatura e com o sinal de controle se apresentando durante todo o tempo de forma saturada.

Figura 3.9 - Implementação do sistema físico com controlador por alocação de pólos projetado
Tempo de amostragem = 0,05 segundos, eixo horizontal em número de amostras



Fonte: Toolbox Simulink MATLAB® (2021)

Foi realizado o mesmo procedimento utilizado na implementação do PID, em que foi feita uma variação do valor de *setpoint*. E novamente para uma valor um pouco abaixo de 50ºC, o sistema conseguiu manter a temperatura controlada com um erro relativamente pequeno. Porém assim que este valor de referência foi aumentado o sistema voltou a oscilar como no início.

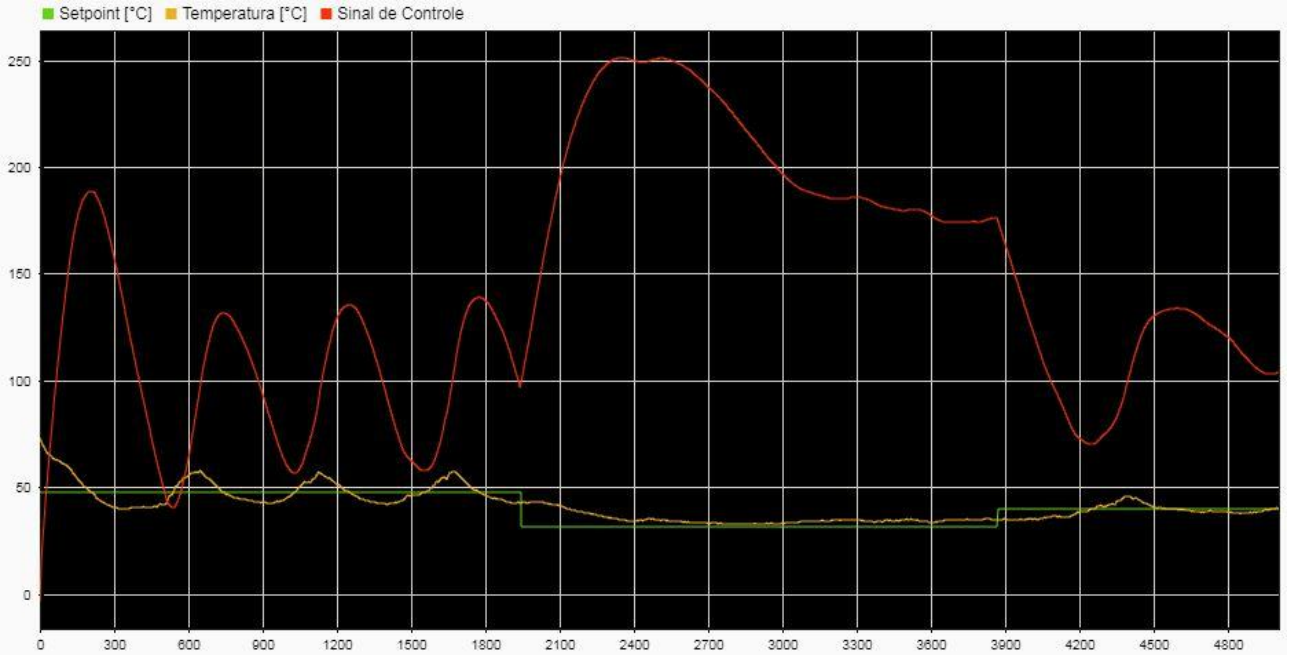
Como a resposta obtida pela implantação do controlador no espaço de estados foi muito similar a resposta obtida pelo controlador PID, novamente a hipótese de que o controle tenha sido agressivo foi levantada. E novamente considerando esta hipótese como verdadeira, duas alternativas foram utilizadas para o ajuste dos ganhos calculados aos pólos:

1. Modificar de forma proporcional a partir de uma constante de “Agressividade” *C*, da mesma forma como foi realizado no controlador PID
2. Implementar uma constante de “Ponderamento” *P*, ou seja, várias coletas de dados foram realizadas para as mesmas condições iniciais, evitando ao máximo a influência de distúrbios. Para cada oscilação gerada, coletou-se os pontos mínimos e máximos locais das oscilações no sinal de controle bruto, sem os limites de saturação (vales e cristas - *Mi* e *Ma*) e realizou-se a seguinte relação, encontrando um valor de *P* igual a *0.009*:

$$P = \frac{255}{Mi} \; ou \; P = \frac{255}{Ma}, \qquad P = 0.009$$

O maior valor encontrado para *Mi* ou *Ma* foi de *28333,33*, concluindo que o sinal de controle está bem agressivo. A constante *P* foi inserida no algoritmo do controlador multiplicando o sinal de controle, cujo resultado está indicado na figura 3.10. Apesar da tentativa de ponderamento não ser a mais correta em termos de referências na literatura, os resultados para o sistema de controle foram satisfatórios em questão de saturação. Entretanto, observa-se que o sinal apresenta um tempo de assentamento maior do que o requerido, onde na passagem do setpoint de 50ºC para 32ºC foram 18,5 segundos (370 x 0.05 - número de amostras vezes o tempo de amostragem) e da passagem de 35ºC para 40 foram 30 segundos (600 x 0.05 - número de amostras vezes o tempo de amostragem)

Figura 3.10 - Implementação do sistema físico com controlador por alocação de pólos ajustado
Tempo de amostragem = 0,05 segundos, eixo horizontal em número de amostras

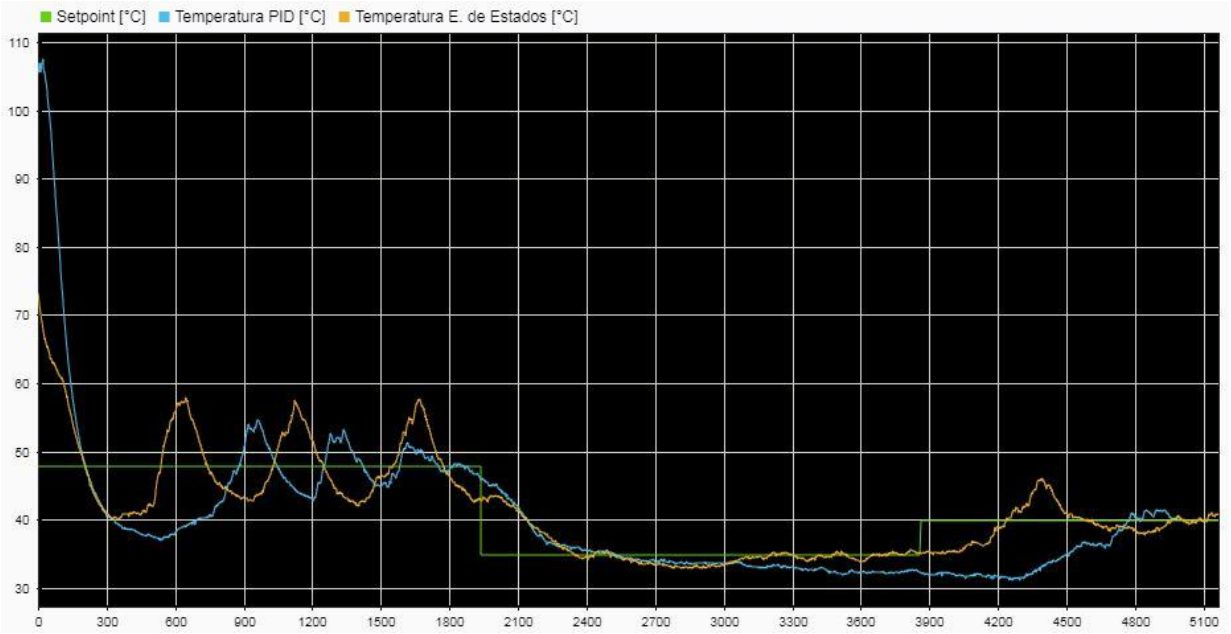


Fonte: Toolbox Simulink MATLAB® (2021)

3.3.3. Comparação dos controladores ajustados

A figura 3.11 apresenta o comportamento da temperatura no sistema com controlador PID e com controlador por alocação de pólos a partir de variações no setpoint: a) iniciando em 48°C, b) de 48ºC para 35ºC e c) de 35°C para 40°C. As condições iniciais não são nulas, porém ambas as simulações foram feitas com temperaturas iniciais diferentes. Observa-se como que, após a inserção dos ganhos e com os devidos ajustes, o sistema tem tendência de convergir para o valor do setpoint. A escolha de qual controlador utilizar é um balanço entre o comportamento das oscilações e do tempo de assentamento, sendo que ocorreram dificuldades para determinar o *overshoot* ou o *undershoot* da resposta do sistema e o tempo de assentamento foi calculado a partir da média de todas as coletas e testes realizados, conforme a tabela 3.1.

Figura 3.11 - Comparação entre os controladores ajustados
Tempo de amostragem = 0,05 segundos, eixo horizontal em número de amostras



Fonte: Toolbox Simulink MATLAB® (2021)

Tabela 3.2 Resultados dos sistemas controlados com ajuste

	Requisitos	com PID	com Alocador
Tempo de assentamento	15 segundos	52 segundos	25 segundos
Overshoot	1%	x	x
Ajuste	-	15	0.009
Ganhos pré-ajuste	-	*Kp* = -15 *Ki* = -1 *Kd* = -0,1	*K1* = -3,9706 *K2* = 11,8414 *K3* = 0,0090 *Ki* = -8,5872
Ganhos pós-ajuste	-	*Kp* = -1 *Ki* = -0,067 *Kd* = -0,0067	*u* = 0.009**sinal bruto*

4. Conclusão

4.1. Síntese do trabalho

Neste trabalho foram apresentados algumas técnicas fornecidas pelo Ident Toolbox do MATLAB® para identificação de uma função de transferência a partir de dados obtidos experimentalmente a partir da planta, no qual o foco foi estimar o sistema justificando teoricamente pela modelagem em caixa preta. Foram projetados dois controladores, sendo um deles PID e outro no espaço de estados com ação integral utilizando alocação dos pólos. Ambos foram simulados em malha fechada e comparados com o sistema físico.

Os dois controladores projetados não tiveram resultados satisfatórios quando implementados no sistema físico, sendo o controlador projetado via alocação de pólos aquele que obteve os piores resultados. Ambos os controladores provocaram oscilações de grande amplitude na temperatura. Foi concluído que esse tipo de comportamento estava relacionado ao fato de que os dois controladores geraram sinais de controle agressivos demais, causando saturação do PWM durante quase todo o tempo.

Diante do tal problema, foram realizados ajustes nos valores de ganhos de ambos os controladores. A priori, o objetivo foi definir um constante de “agressividade” que ajustasse os ganhos do controlador PID de forma proporcional e que não variasse muito em relação aos ganhos calculados teoricamente, e para o controlador por alocação de pólos a única alternativa foi determinar uma constante de “ponderamento” no qual multiplica o sinal de controle para entrar nos limites de saturação (entre 0 e 255). Entretanto, foi visto que isso implica no regime transitório em um tempo de assentamento maior e oscilatório, o que é aceitável em prototipagens porém indesejável em processos industriais.

Em seguida, observa-se a importância de se validar a estimação com os dados experimentais para uma melhor representatividade do modelo, sendo crucial para a escolha de qual resultado adotar.

4.2. Dificuldades

Durante a realização deste trabalho aconteceram vários imprevistos, a começar pela dificuldade de identificar um modelo que se ajusta-se bem à base de dados coletada. O sistema físico se comportou de maneira um pouco estranha em algumas ocasiões, como por exemplo, quando houve uma diminuição do PWM. Teoricamente esperava-se que a temperatura medida começasse a aumentar, porém as amostras coletadas mostraram que a temperatura se manteve constante. Este obstáculo foi superado parcialmente, já que a função de transferência encontrada não teve um valor de *fit* tão alto.

Após o problema da identificação, outra dificuldade encontrada foi durante a definição dos parâmetros do controlador PID. Apesar da interface do PID Tuner do MATLAB® ser bastante intuitiva, foram necessárias várias tentativas até que fossem encontrados os valores de ganhos que satisfaziam os critérios de desempenho desejados, e como se trata de um processo caro computacionalmente, foi gasto muito tempo de trabalho atrás destes parâmetros.

Também foi enfrentado problema semelhante durante o projeto de controlador no espaço de estados pelo método de alocação de pólos. Os valores de ganhos obtidos pela função *acker()* não satisfaziam os requisitos de projetos, com isso foi feito um ajuste manual de cada valor de ganho até que os critérios de desempenho fossem atingidos.

Por fim, a implementação dos controladores projetados no sistema físico não corresponderam com os resultados obtidos a partir das simulações das malhas fechadas no

Simulink. Os principais motivos que não garantiram a relação foram os ajustes manuais dos ganhos durante a simulação na prática e o *fit* não satisfatório para o modelo estimado, visto que a temperatura se comportava de maneira inesperada em relação ao sinal de PWM que era enviado ao *cooler* durante a etapa de modelagem, o que dificultou a estimação do modelo para determinar com exatidão os ganhos para o controlador PID e para a alocação dos pólos no espaço de estados. Assim, na tentativa de corrigir o sinal de controle, os ajustes foram feitos, porém como consequência a saída é mais lenta e oscilatória.

5. Referências bibliográficas

AGUIRRE, L. A. "Introdução à Identificação de Sistemas: técnicas lineares e não-lineares aplicadas a sistemas reais". 3. ed rev. e ampl. Belo Horizonte: 2007.

BADAMASI, Yusuf Abdullahi. The working principle of an Arduino. In: 2014 11th international conference on electronics, computer and computation (ICECCO). IEEE, 2014. p. 1-4.

BANZI, Massimo; SHILOH, Michael. Getting started with Arduino: the open source electronics prototyping platform. Maker Media, Inc., 2014.

CARVALHO, S. N. N. Modellistica e Controllo di un Motore in Corrente Continua, Monografia – E265 – Politécnico de Turim, 2010.

CARVALHO, Luiz Euquério de; OLIVEIRA, Sônia Maria Pinheiro; TURCO, Sílvia Helena Nogueira. Utilização da nebulização e ventilação forçada sobre o desempenho e a temperatura da pele de suínos na fase de terminação. Revista Brasileira de Zootecnia, v. 33, p. 1486-1491, 2004.

DO BRASIL, René Porfirio Camponez. Utilização de exaustores eólicos no controle da temperatura e da ventilação em ambiente protegido. Biblioteca digital de teses e dissertações da USP. Disponível em: https://www.teses.usp.br/teses/disponiveis/11/11143/tde-25042005-170200/publico/rene.pdf Acesso em, v. 2, 2004.

SILVA, Edison Luiz Salgado et al. OBTENÇÃO DE SENSORES DE TEMPERATURA DO TIPO NTC A PARTIR DE MISTURA MECÂNICA DE ÓXIDOS DE NÍQUEL E COBALTO. 2011.

GUSTAVSEN, B.; SEMLYEN, A., “Application of vector fitting to state equation representation of transformers for simulation of eletromagnétic transients”, Power Delivery, IEEE Transactions on, v. 13, n. 3, p. 834-842, July 1998.

LEVARDA,B.; BUDACIU, C. The design of temperature control system using PIC18F4620. Technical University Gheorghe Asachi Iasi, 2010.

NERI, MAXGG et al. Sistema de Controle de Temperatura para Estufa. In: XV Congresso Brasileiro de Automdtica. 2004.

NISE, Norman S. Engenharia de Sistemas de Controle, 7ª edição. Grupo GEN, 2017. 9788521634379. Disponível em: https://integrada.minhabiblioteca.com.br/#/books/9788521634379/. Acesso em: 26 Sep 2021

MASON, J. "Guide to measuring rotary speed with tachometergenerators," Machine Design, page 157, October 8, 1964.

OGATA, K. Discrete-Time Control Systems. Prentice Hall,1987, New Jersey, USA.

PERDOMO, L.C. Ambiência e produção. Suinocultura Industrial, v.21, n.136, p.12, 1999. Anuário 99.

SILVA, J. C. S., “Modelagem caixa preta de transformadores de potência em amplo espectro de frequências”, Dissertação de Mestrado, Programa de Pós-graduação em Engenharia Elétrica (PPGEE), UFMG – Universidade Federal de Minas Gerais, 2014.

WENDLING, Marcelo. Sensores. Universidade Estadual Paulista. São Paulo, v. 2010, p. 20, 2010.

6. Anexos

6.1. Anexo A

Código usado para aquisição de dados com controlador por alocação de pólos implementado.

```
// Declaração das variáveis para a base do transistor e inclusão da
biblioteca do NTC
#define base_transistor 3
#include <Thermistor.h>

Thermistor temp(5);       // Variável do tipo Thermistor, indicando o
pino no qual está conectado o sensor

// Declaração das variáveis para controle da malha
float y = 0.0;           // Temperatura atual
float y0 = 0.0;          // Temperatura anterior
float y00 = 0.0;         // Temperatura antes da anterior
float setpoint = 0.0;    // Valor do setpoint
float u = 0.0;           // Sinal de controle em PWM
float e = 0.0;           // Sinal de erro
float xn = 0.0;          // Estado referente a ação integral
float x1 = 0.0;          // Estado 1
float x2 = 0.0;          // Estado 2
float x3 = 0.0;          // Estado 3
float P = 0.009;           // Ganho de ajuste do sinal de controle para
ponderamento
float raw = 0.0;

// Ganhos calculados a partir do modelo estimado para os estados
(método Acker)
float K[3] = {-3.9706, 11.8414, 0.0090};        // Ganho dos pólos da
equação característica
float Ke = {-8.5872};                          // Ganho do integrador

void setup()
{
  Serial.begin(9600);                 // Configuração de transmissão de
bits do Arduino UNO

  // Os dados são enviados diretamente para uma planilha do Excel, logo
esta linha é para limpar as colunas no software
  // A linha seguinte é para criar o cabeçalho da planilha
  Serial.println("CLEARSHEET");
  Serial.println("LABEL,Time,Timer,SetPoint [C], Temperatura [C], Sinal
de Controle");

   pinMode(base_transistor,OUTPUT);      //Define pino como saida para
chavear o transistor
}

void loop()
{
```

```
   // Printa na planilha a data atual, o horário e o instante de amostragem
  Serial.print((String)"DATA,TIME,TIMER,");

  setpoint = 48.0;
  if (millis()>100000 && millis()<=200000){
    setpoint = 35.0;           // Setpoint definido para 40°C
  }else if(millis()>200000){
    setpoint = 40.0;           // Setpoint definido para 70°C
  }

   y = temp.getTemp();              // Temperatura atual resgatada e armazenada em y
  e = setpoint-y;              // Sinal de erro
   xn = e+xn;                        // Valor do estado relacionado ao integrador, no qual é a integral do erro

   y00 = y0;                        // y00 é a variável que armazena a temperatura da segunda leitura anterior a atual
  y0 = x1;                       // y0 armazena a leitura de temperatura anterior a atual
  x1 = y;                       // x1 é o estado que representa e recebe a leitura da temperatura atual

   x2 = y-y0;                       // x2 é o estado que representa a variação das duas últimas leituras de temperatura realizadas
   x3 = (y-y0)-(y0-y00);           // x3 é o estado que representa a variação entre as duas últimas leituras com as duas leituras anteriores a atual ("variação da variação")

  u = P*(-((K[0])*x1 + (K[1])*x2 + (K[2])*x3) + (Ke)*xn);   // Sinal de controle é gerado a partir da lei de controle
  raw = u;

   // A ideia do IF é justamente não saturar o atuador, sendo que os limites são 0 e 255 para gerar o ciclo de trabalho do sinal que chaveia o transistor em série com o cooler (motor DC)
  if(u>255.0){
    u = 255.0;
  }else if(u<0.0){
    u = 0.0;
  }

   analogWrite(base_transistor,u);    // o pino PWM do Arduíno gera o sinal para chavear o transistor

  // Print dos valores do setpoint, temperatura e sinal de controle na planilha do excel
     Serial.println((String)""  +  setpoint  +  ","  +  y  +  ","+  u +",AUTOSCROLL_20");
  delay(10);
}
```

6.2. Anexo B

Código usado para aquisição de dados com controlador PID implementado

---

```
// Declaração das variáveis para a base do transistor e inclusão da
biblioteca do NTC
#define base_transistor 3
#include <Thermistor.h>

Thermistor temp(5);     // Variável do tipo Thermistor, indicando o
pino no qual está conectado o sensor

// Declaração das variáveis para controle da malha
float y = 0.0;          // Temperatura atual
float setpoint = 0.0;   // Valor do setpoint
float u = 0.0;          // Sinal de controle em PWM
float e = 0.0;          // Sinal de erro
float e0 = 0.0;         // Erro calculado no momento anterior
float Sume = 0.0;       // Integral do erro
float Vare = 0.0;       // Derivada do erro
float C = 15.0;         // Constante de agressividade
float raw = 0.0;

// Ganhos calculados a partir do método adotado no projeto
float K[3] = {-15.0/C, -1.0/C, -0.1/C};         // Respectivamente,
ganhos Kp, Ki e Kd

void setup()
{
  Serial.begin(9600);                 // Configuração de transmissão de
bits do Arduino UNO

  // Os dados são enviados diretamente para uma planilha do Excel, logo
esta linha é para limpar as colunas no software
  // A linha seguinte é para criar o cabeçalho da planilha
  Serial.println("CLEARSHEET");
  Serial.println("LABEL,Time,Timer,SetPoint [C], Temperatura [C], Sinal
de Controle");

  pinMode(base_transistor,OUTPUT);    //Define pino como saida para
chavear o transistor
}

void loop()
{
  // Printa na planilha a data atual, o horário e o instante de
amostragem
  Serial.print((String)"DATA,TIME,TIMER,");
```

```
  setpoint = 48.0;
  if (millis()>100000 && millis()<=200000){
    setpoint = 35.0;           // Setpoint definido para 40°C
  }else if(millis()>200000){
    setpoint = 40.0;           // Setpoint definido para 70°C
  }

  y = temp.getTemp();          // Temperatura atual resgatada e
armazenada em y
  e = setpoint-y;              // Sinal de erro
  Sume = e+Sume;               // Resultado da integral do sinal de
erro
  Vare = e-e0;                 // Resultado da derivada do sinal de
erro

  e0 = e;                                      // Registra o valor do
erro para se usado na medição posterior
  u = ((K[0])*e + (K[1])*Sume + (K[2])*Vare);   // Sinal de controle é
gerado a partir da lei de controle
  raw = u;

  // A ideia do IF é justamente não saturar o atuador, sendo que os
limites são 0 e 255 para gerar o ciclo de trabalho do sinal que chaveia
o transistor em série com o cooler (motor DC)
  if(u>255.0){
    u = 255.0;
  }else if(u<0.0){
    u = 0.0;
  }

  analogWrite(base_transistor,u);   // o pino PWM do Arduíno gera o
sinal para chavear o transistor

  // Print dos valores do setpoint, temperatura e sinal de controle na
planilha do excel
  Serial.println((String)"" + setpoint + "," + y + ","+ u
+",AUTOSCROLL_20");
  delay(10);
}
```