INSTITUTO FEDERAL DE GOIÁS
Engenharia de Controle e Automação

CONTROLE PID ANTI-WINDUP COM REJEIÇÃO DE PERTURBAÇÕES

Alexandre Alves Trindade

29 de novembro de 2018

RESUMO

O presente trabalho consiste em projetar controlador PID, com técnica antiwindup e presença de perturbação adicionado ao sistema de controle. Para testar o controlador projetado, a parte experimental de montagem e visualização dos dados em tempo real devem ser realizadas. O projeto elétrico foi modificado conforme a disponibilidade de componentes, e principalmente para ter aquecimento rápido, foi usado resistências adequadas. O método utilizado para obter parâmetros do PID foi Ziegler-Nichols, para discretizar as funções transferência o método de Euler (emulação). A análise gráfica dos resultados obtidos durante o período de meia hora de experimento atestam que o sinal de perturbação não foi capaz de modificar tanto o sinal de controle, devido ás características da planta.

CONTEÚDO

Pág.

LISTA DE FIGURAS

Pág.

CAPÍTULO 1

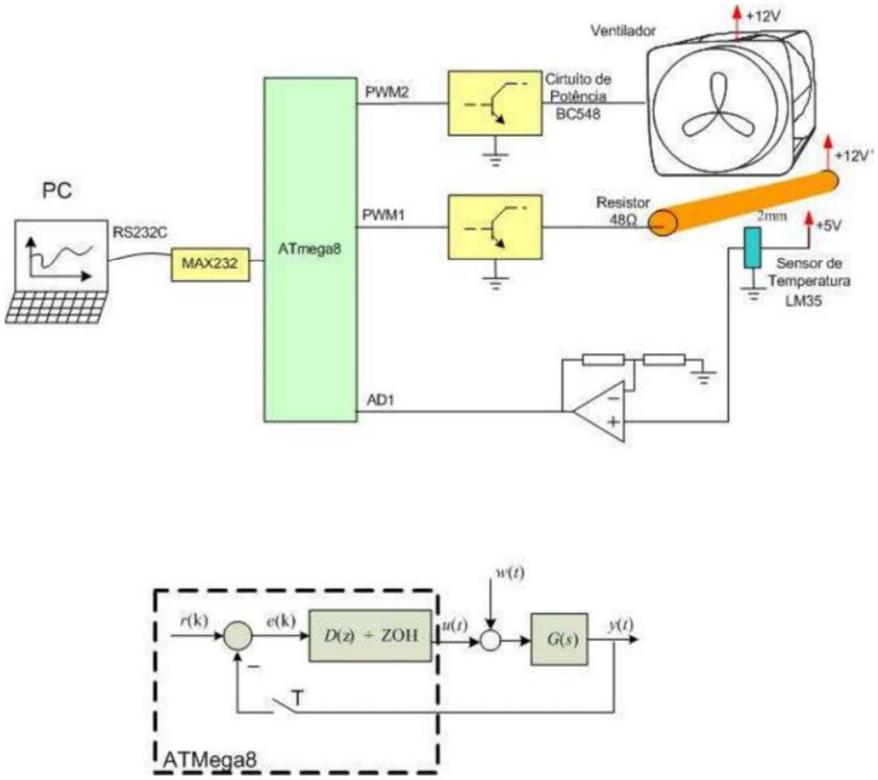
Introdução

O presente trabalho consiste em projetar controlador PID anti-windup discreto que atenda determinadas especificações. A variável controlada é temperatura, é utilizado microcontrolador para gerar tensão por largura de banda - PWM - e aquecer as resistências elétricas.

A Figura 1.1 é o esquema elétrico indicado para execução do trabalho. O sistema inclui transistores para limitar a corrente elétrica que aumenta a temperatura, e para gerar onda senoidal de funcionamento do cooler; o qual é utilizado como fonte de perturbação; capacitores para filtrar o sinal da temperatura; sensor de temperatura. A conversão da leitura da porta analógica A0 do microcontrolador para °C é apresentado na Equação 1.1.

$$\begin{aligned} A0\_value &\rightarrow \{0 - 1023\} \\ 1°C &\rightarrow 10mV \\ Temperatura &= \frac{[A0\_value] \cdot [5V/1023]}{10mV} \end{aligned} \tag{1.1}$$

Figura 1.1 - Esquema elétrico do projeto.



Para projetar o controlador PID, e converter do tempo contínuo para discreto foi utilizado o método de Euler (emulação), especificamente *backward difference*. Equação 1.2, relação matemática entre s e z, T é o período de amostragem. A equação a diferenças, obtida deste método, deve ser implementada no microcontrolador. O procedimento experimental tem duração de 30 minutos, período de 400s do setpoint (onda quadrada), ação anti-windup; a partir de 1200s acionado ventilador como perturbação senoidal.

$$s \approx \frac{z-1}{Tz} \tag{1.2}$$

Especificações do projeto:

a) Sobrepasso projetado em malha fechada de $\leq 40\%$;

b) Tempo de pico $\leq$ 20s;

c) Rejeição de perturbações constantes;

d) Período de amostragem 1s;

e) Resolução da temperatura $\leq$ 0,1°C.

CAPÍTULO 2

Metodologia

O capítulo de Metodologia trata de como foi desenvolvido o projeto do controlador, os *softwares* utilizados e a montagem do experimento. O desenvolvimento do trabalho é dividido da seguinte forma:

1) Montagem do *hardware*, foram feitas adaptações em relação ao projeto da Figura 1.1 para atender às especificações;

2) Obter a função de transferência da planta, com base em dados experimentais que foram coletados com o sistema em malha aberta;

3) Aplicar o método de Ziegler-Nichols para encontrar as constantes proporcional (Kp), integral (Ki) e derivativo (Kd) do PID;

4) Com o método de Euler, encontrar a transformada Z da função de transferência do controlador e da planta;

5) Com a ferramenta **sisotool** visualizar o LGR - Lugar das Raízes - e modificar os pólos, caso necessário, para atender as especificações do projeto;

6) Escrever no microcontrolador a equação a diferenças;

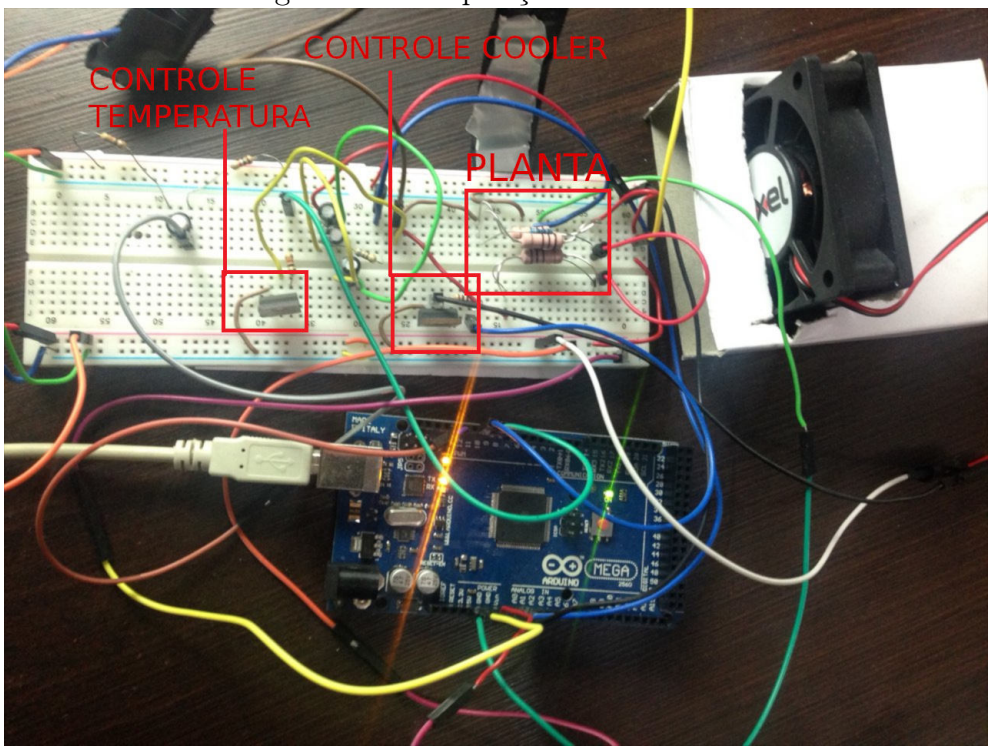
7) Etapa de testes e análise dos resultados.

2.1 MONTAGEM DO EXPERIMENTO

Os materiais utilizados no experimento são: 1 fonte 12VCC (corrente 1A), 1 multímetro digital, 3 resistores 100Ω (2W), 2 resistores 100Ω (1W), 2 transistores TIP31C, 3 capacitores de lítio 10μF, 1 *protoboard*, 1 sensor de temperatura LM35, 1 *push button*, 1 microcontrolador Arduino Mega 2560, cooler 12VCC cabos e *jumpers*.

A Figura 2.1 apresenta a bancada em que os experimentos foram feitos, e indica partes específicas dos componentes. O botão, ao ser pressionado, interrompe a visualização da ação de controle em tempo real.

Figura 2.1 - Disposição do *hardware*.



O amplificador operacional não foi utilizado, principalmente em função da falta da fonte simétrica ±12VCC necessário para seu funcionamento. Para tratar o sinal de saída do sensor de temperatura, foram utilizados três capacitores em paralelo, capacitância equivalente de 30μF. Essa substituição foi resultado de diferentes montagens que foram testadas, a atual foi que se obteve menor ruído na leitura da temperatura durante o período de meia hora do experimento.

O desafio de obter o tempo de pico menor que 20s, resultou na modificação do transistor BC548 para o TIP31C, como pode ser verificado no Anexo A deste trabalho, os valores de corrente do primeiro transistor são de 100mA ou 200mA, enquanto do segundo são de 1A a 5A. Os resistores usados de 1 e 2W tem aquecimento mais rápido do que outras combinações testadas de 3, 5, e 10W de potência.

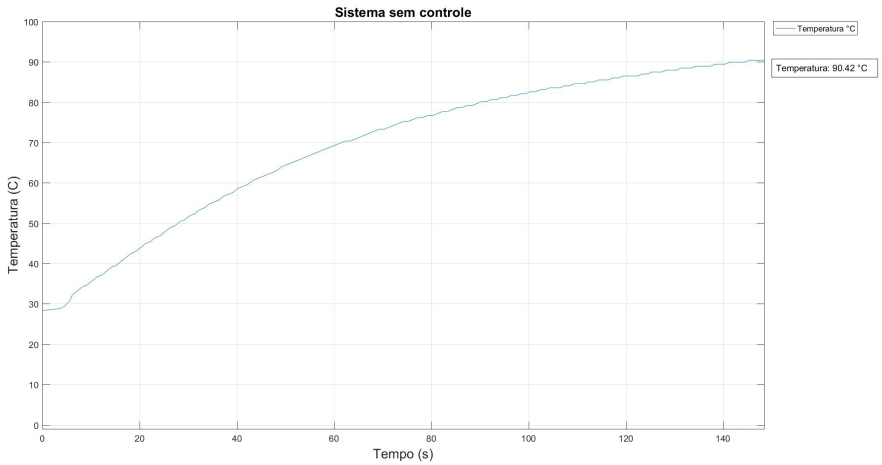
2.2 FUNÇÃO DE TRANSFERÊNCIA DA PLANTA

O método para obter a função de transferência da planta, consiste em utilizar a saída PWM do Arduino em 100% de *Duty Cycle* por período de tempo superior a 20 minutos. Os dados desse aquecimento constante são armazenados no *workspace* do Matlab, e ferramentas desse programa são usadas para obter a FTMA. O resultado gráfico é apresentado na Figura 2.2, o Apêndice A mostra o código fonte do microcontrolador que gerou este aquecimento, o Apêndice B é o código fonte do Matlab para plotar o gráfico e armazenar os dados.

O comando do Arduino *analogWrite(pin,value)* indica o pino a ser escrito o *duty cycle* do PWM, varia de 0 a 255 (100% *duty cycle*) (ARDUINO.CC). Na Figura 2.2 o período do experimento foi de 140s e foi interrompido, tendo em vista temperatura alcançada de 90,42°C, há riscos de causar danos a *protoboard* se mantida por período longo, a temperatura máxima suportada por maior parte das *protoboard* é de 80°C (FILIPEFLOP.COM).

Na Equação 2.1, a primeira linha cria um objeto no tempo contínuo, contém o sinal de saída (temperatura_saida) e o sinal de entrada (A), ambos vetores de mesmo tamanho, o tempo de amostragem de 1s. Todos elementos do vetor 'A' são 1, para analisar a resposta ao degrau do sistema. A segunda linha o comando *tfest - transfer function estimation* - estima a função de transferência do objeto *data*, considera o número de pólos igual a 1.

Figura 2.2 - Leitura de temperatura aquecimento contínuo.



$$data = iddata(temperatura\_saida, A, 1)$$
$$sys = tfest(data, 1)$$

(2.1)

Figura 2.3 - FT da planta.

```
sys =

  From input "u1" to output "y1":
      1.452
  -----------
  s + 0.01464

Continuous-time identified transfer function.

Parameterization:
   Number of poles: 1   Number of zeros: 0
   Number of free coefficients: 2
   Use "tfdata", "getpvec", "getcov" for parameters and their uncertainties.

Status:
Estimated using TFEST on time domain data "data".
Fit to estimation data: 96.94% (simulation focus)
FPE: 0.3301, MSE: 0.3171
```

A Figura 2.3 apresenta a FT da planta, resultado da sequência descrita nesta seção. A adequação aos dados com 1 pólo e nenhum zero foi de 96.94%.

2.3 MÉTODO DE ZIEGLER-NICHOLS

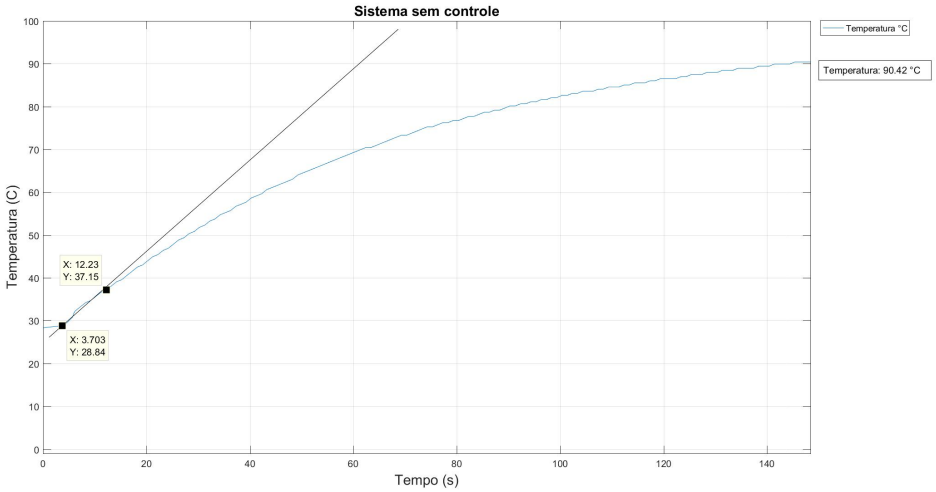
Esta seção é dedicada à descrição de como foi aplicado o método citado para obter os parâmetros Kp, Ki, Kd do controlador PID. A Figura 2.4 constitui a parte gráfica necessária para este método, a tangente traçada permite encontrar o valor de L e T. O Anexo B contém a tabela modelo em que os cálculos da Tabela 2.1 são baseados.

A Equação 2.2 é a função de transferência do controlador PID em termos do ganho proporcional - Kp - tempo integral - Ti - e tempo derivativo - Td. A Equação 2.3 é a mesma função de transferência em termo dos ganhos proporcional, integral e derivativo.

Tabela 2.1 - Parâmetros do controlador.

T=8.53	L=3.70		
	Kp	Ti	Td
P	2.30	∞	0
PI	2.07	12.34	0
PID	2.76	7.41	1.85

Figura 2.4 - Método de Ziegler-Nichols.



$$
\begin{aligned}
G_c(s) = Kp(1 + \frac{1}{T_i s} + T_d s) \\
Ki = \frac{Kp}{T_i} \\
Kd = Kp \cdot T_d
\end{aligned}
\tag{2.2}
$$

$$
G_c(s) = Kp + \frac{Ki}{s} + Kd \cdot s \tag{2.3}
$$

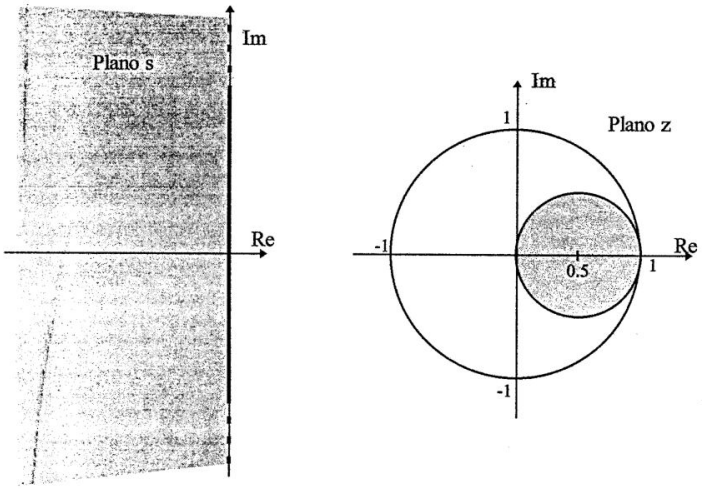
A Equação 2.4 resulta da aplicação do método descrito nesta seção, a função de transferência no domínio do tempo contínuo do controlador PID.

$$
G_c(s) = \frac{5.12s^2 + 2.76s + 0.37}{s} \tag{2.4}
$$

2.4 MÉTODO DE EULER

Neste método a equação que relaciona **s** e **z** foi apresentada no capítulo 1. A Figura 2.5 apresenta o mapeamento dos pólos do tempo contínuo para o domínio discreto.

Figura 2.5 - Mapeamento entre os planos **s** e **z**.



Fonte: (SOARES, 1996).

A Equação 2.5 mostra a função de transferência no domínio **z** do controlador e da planta, após ser utilizado o método de Euler.

$$\text{FT CONTROLADOR}$$
$$G_c(z) = \frac{8.252z^2 - 12.99z + 5.116}{z^2 - z} \tag{2.5}$$
$$\text{FT PLANTA}$$
$$G_p(z) = \frac{1.431z}{z - 0.9856}$$

2.5 LUGAR DAS RAÍZES

A presente seção apresenta o desenho do LGR, foi utilizado a ferramenta *sisotool*. A Figura 2.6 a configuração do diagrama de blocos do sistema em malha fechada, é a configuração da arquitetura do sistema. O elemento **C** recebe a FT do controlador e o elemento **G** recebe a FT da planta, ambas no domínio **z**.

A Figura 2.7 mostra a resposta ao degrau e o LGR que resulta da configuração feita no programa. As especificações, ver capítulo 1, foram feitas através da opção *Design Requirements*, são essas: tempo de subida menor ou igual 20s, percentual de sobressinal até 40%, Figura 2.8, o resultado desta modificação feita no controlador. A Equação 2.6 é a FT do controlador, resultado da adequação indicada nesta seção.

Figura 2.6 - Configuração da arquitetura do sistema.

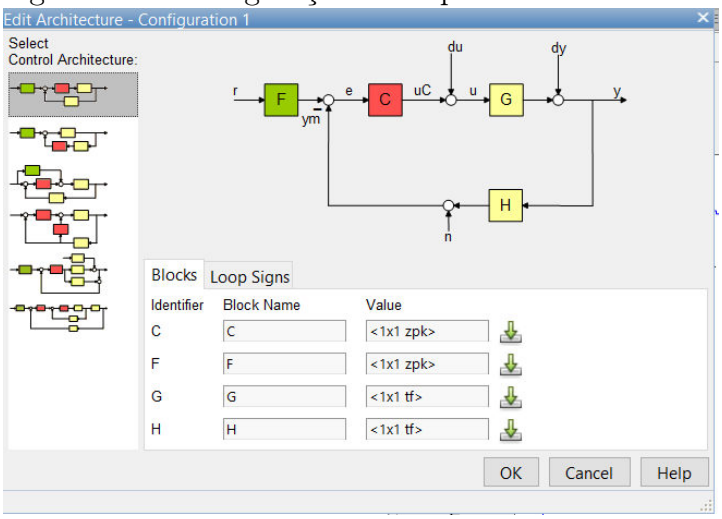
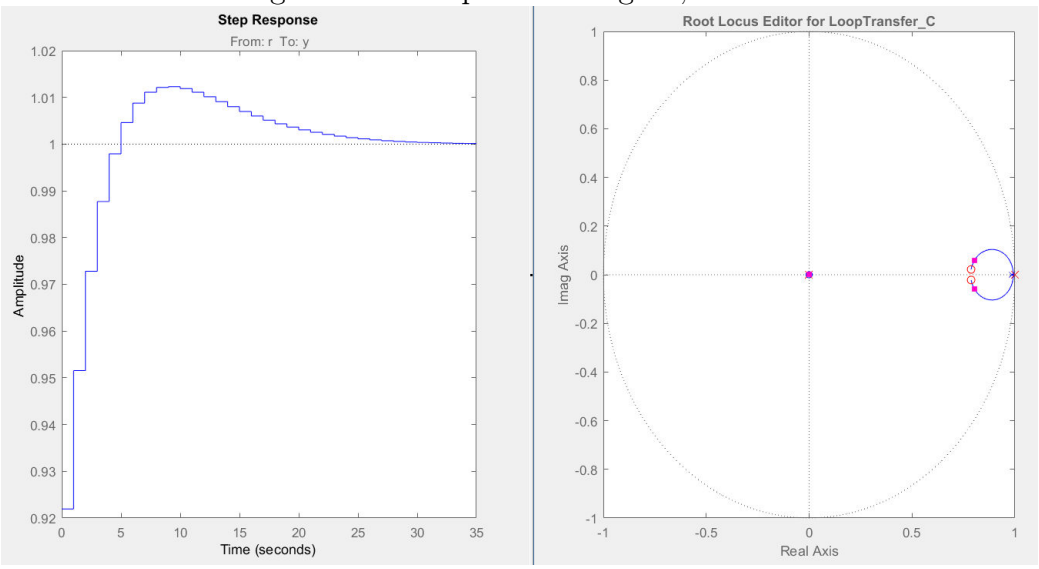
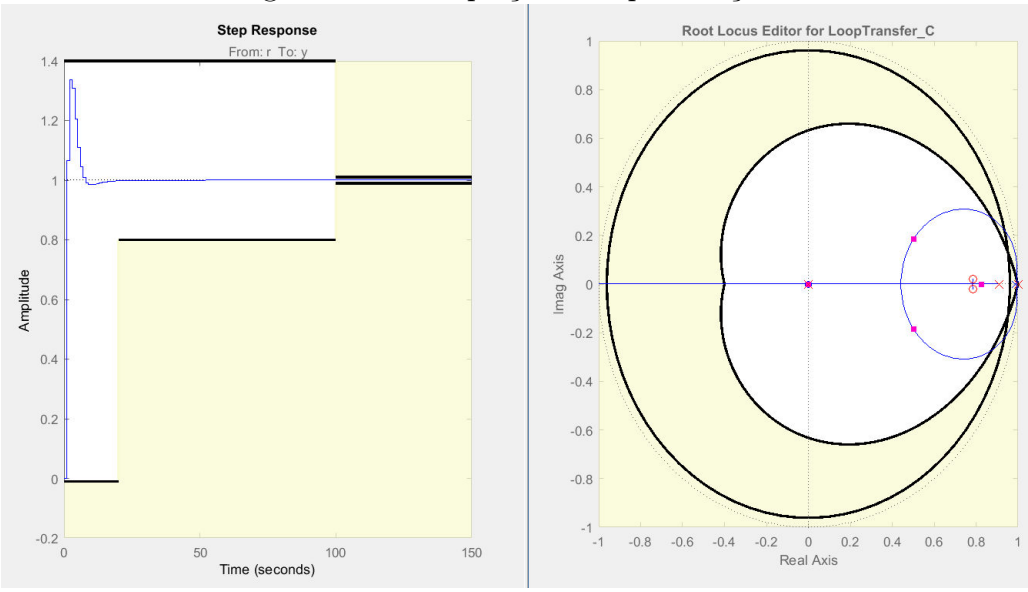


Figura 2.7 - Resposta ao degrau, e LGR.



$$G_c(z) = \frac{0.74z^2 - 1.17z + 0.46}{(z)(z-1)(z-0.9098)} \tag{2.6}$$

Figura 2.8 - Adequação às especificações.



2.6 EQUAÇÃO A DIFERENÇAS

A equação a diferenças do controlador foi obtida conforme Equação 2.7 e Equação 2.8, foi considerado para implementar no microcontrolador 3 tempos anteriores (kT - 3). A Figura 2.10 apresenta o diagrama de blocos do sistema em malha fechada. A Figura 2.9 é o diagrama de blocos que foi considerado para evitar a saturação do elemento integrador do PID (Ki).

A Equação 2.9, onde o sinal u(t) é o sinal não saturado e v(t) o sinal saturado, K_t constante que determina quanto é subtraído da parcela a ser integrada (NETO, 2005). Esta relação foi implementada no microcontrolador como técnica antiwindup back calculation quando a saída do controlador ultrapassa o valor máximo.

A referência é o valor constante, 40 ou 50°C, a realimentação do sistema recebe a temperatura medida pelo sensor a cada período de amostragem. A saída do controlador é utilizada para modificar o *duty cycle* da tensão PWM que aquece as resistências.

$$\frac{OUTPUT}{INPUT} = \frac{0.74z^2 - 1.17z + 0.46}{(z)(z-1)(z-0.9098)} \tag{2.7}$$

$$
\begin{aligned}
erro &= Setpoint - Temperatura; \\
output(kT) &= 1.91 \cdot output[kT-1] - 0.91 \cdot output[kT-2] \\
&+0.74 \cdot erro[kT-1] - 1.17 \cdot erro[kT-2] + 0.46 \cdot erro[kT-3]
\end{aligned} \tag{2.8}
$$

Figura 2.9 - Sistema malha fechada.

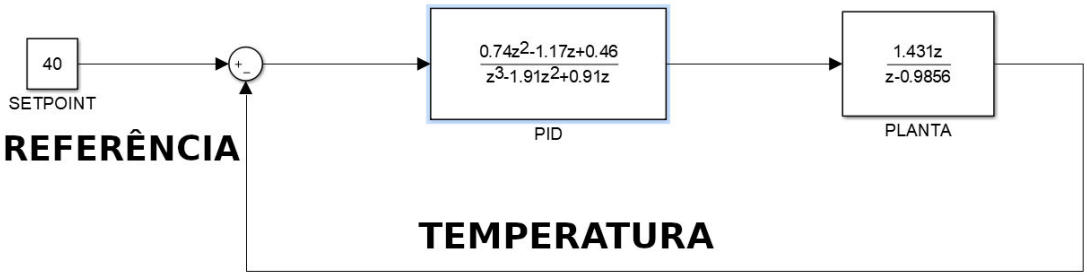
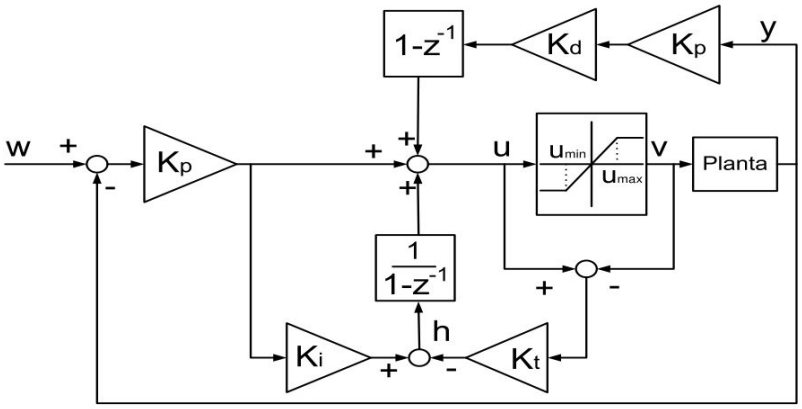


Figura 2.10 - Estrutura da realimentação linear da técnica Back Calculation.



Fonte: (NETO, 2005).

$$
h(t) = K_t(u(t) - v(t)) \tag{2.9}
$$

CAPÍTULO 3

Resultados e Discussão

Este capítulo apresenta o resultado da ação de controle de temperatura, conforme detalhado nos capítulos anteriores. O simulador gráfico mostra na tela intervalo de 50s, garantindo assim visualização adequada dos dados de temperatura e tensão PWM durante o intervalo de 30 minutos do experimento. Ao final do intervalo de tempo definido, é gerado uma figura que contém os dados do tempo 0 ao 1800s.

As Figuras 3.1 e 3.2 apresentam o gráfico gerado do período completo do experimento. Na parte superior temperatura, e na inferior a tensão PWM do cooler (vermelho) e do controle de temperatura (azul). Apesar de similares, o segundo gráfico foi modificado em relação à constante da ação antiwindup.

As Figura 3.3 e 3.4 apresentam o tempo de pico de cada configuração relatada neste capítulo. O primeiro, tempo de pico aproximado de 30s, alcança o setpoint em 19.1s. O segundo, tempo de pico de 34.1s, alcança o setpoint em 26.1s (tempo = 200s). A Figura 3.5 apresenta o resultado do sinal de controle sob efeito da ventilação como onda senoidal, o tempo de pico passa para 40s e tempo que alcança o setpoint 31s, sistema do primeiro experimento (tempo = 1400s).

Figura 3.1 - Procedimento experimental.

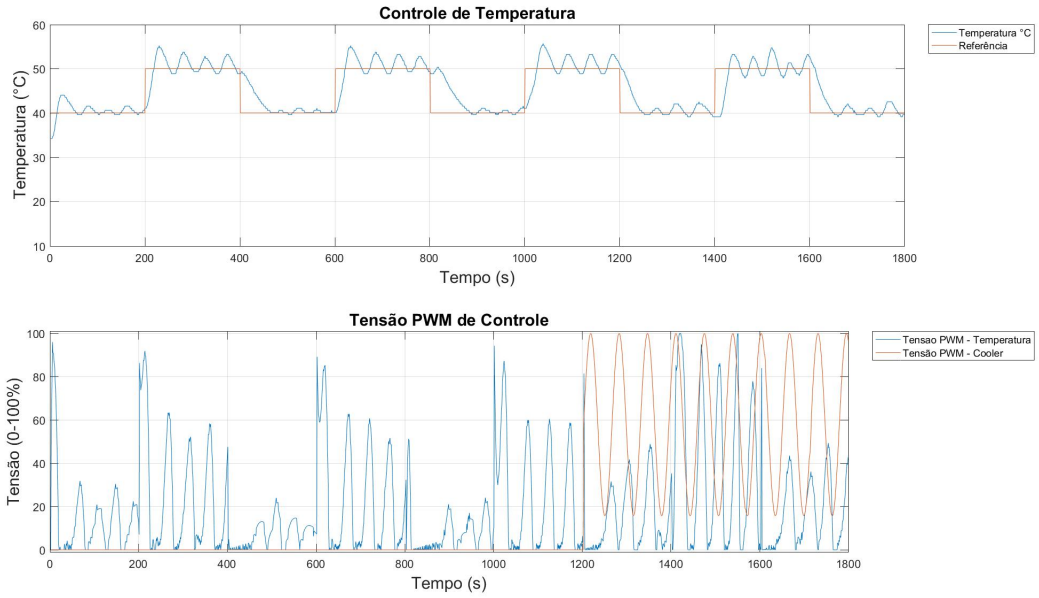


Figura 3.2 - Procedimento experimental, modificado.

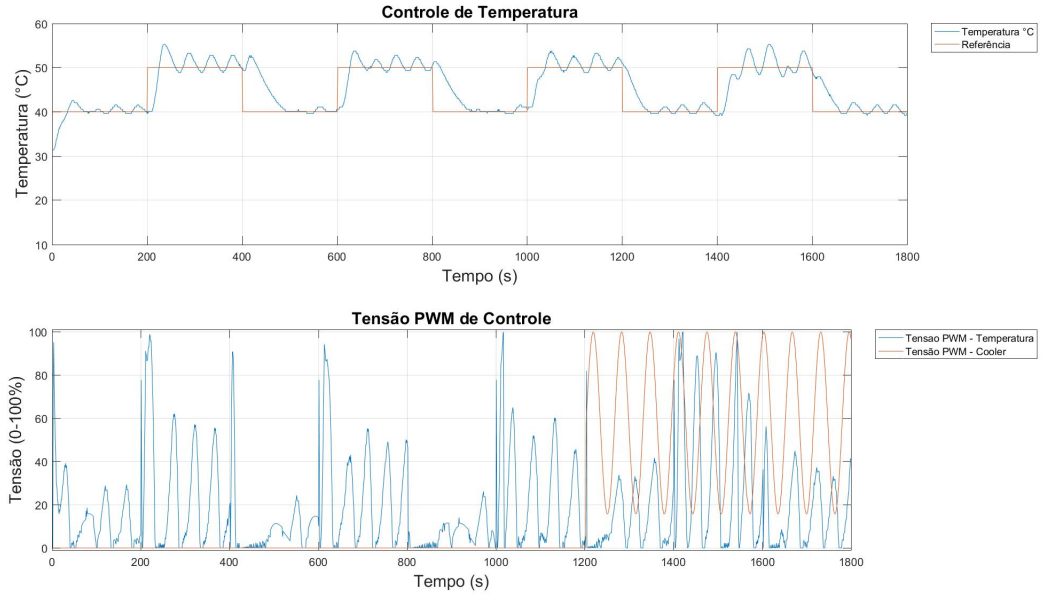


Figura 3.3 - Experimento 1, tempo de pico.

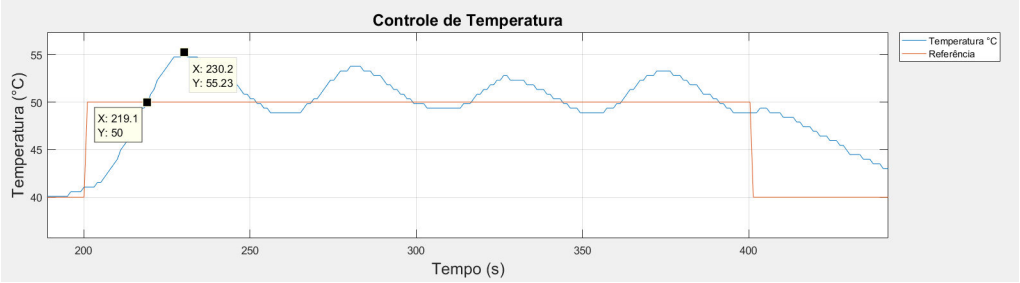


Figura 3.4 - Experimento 2, tempo de pico.

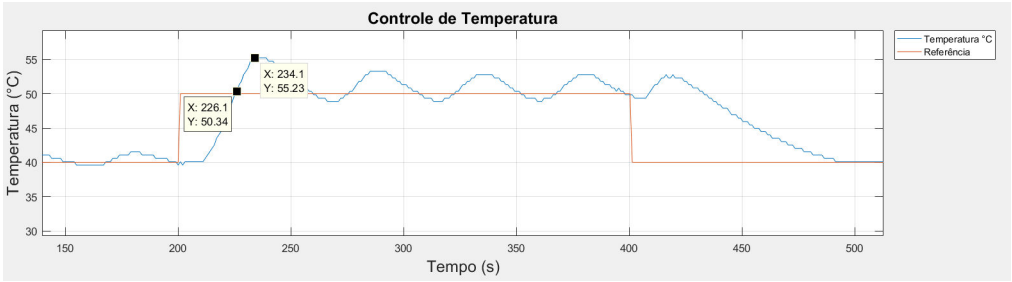
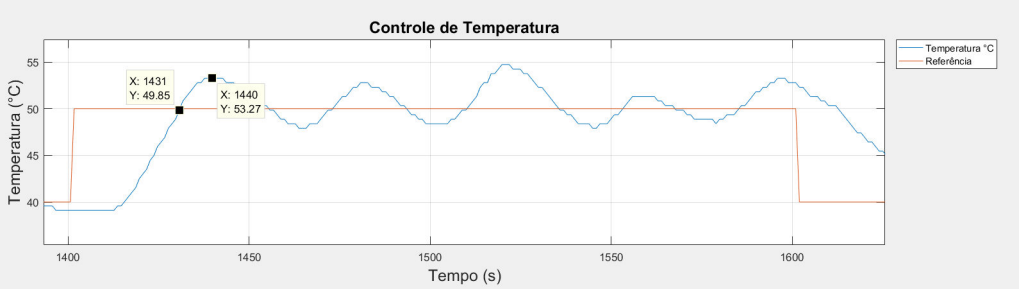


Figura 3.5 - Experimento 1, tempo de pico, ação de perturbação do cooler.



CAPÍTULO 4

Conclusão

A montagem dos componentes na *protoboard* no início tem por objetivo adaptar o projeto elétrico de modo que não tenha perda na funcionalidade, e tenha capacidade de atingir às especificações quanto ao tempo de pico, estabilização e percentual de sobressinal.

A utilização de programas foi aplicado na coleta de dados, visualização em tempo real, e no projeto do controlador. Foi possível dessa maneira obter a função de transferência da planta, com o método Ziegler-Nichols os parâmetros do controlador PID, discretizá-las com o método de Euler, e com recursos computacionais adaptar a FT do controlador para obedecer a ação de controle requerida.

A equação a diferenças foi implementada no microcontrolador Arduino Mega 2560, foi comparado faixas de valores da saída do PID e o resultado do período completo de meia hora de análise foi estudado. A técnica antiwindup back calculation foi utilizada para solucionar o problema de saturação do termo integrador da saída do controlador. A perturbação do cooler não modificou significativamente os tempos do sinal de controle, a configuração da planta (potência dos resistores utilizados) tem capacidade em malha aberta de aquecimento mais de um grau Celsius por segundo. O método para validar os resultados foi análise gráfica.

REFERÊNCIAS

ARDUINO.CC. **analogWrite();**. [S.l.]: <https://www.arduino.cc/reference/en/language/functions/analog-io/analogwrite/>. Acesso: 17 de novembro de 2018. 5

FILIPEFLOP.COM. **Protoboard 830 Pontos**. [S.l.]: <https://www.filipeflop.com/produto/protoboard-830-pontos/#tab-description>. Acesso: 17 de novembro de 2018. 5

NETO, A. H. **Técnicas Anti-Windup em Estruturas de Controle PID, RST e GPC**. [S.l.]: <https://repositorio.ufsc.br/bitstream/handle/123456789/102669/223414.pdf?sequence=1>. Acesso: 29 de novembro de 2018, 2005. 10, 11

SOARES, P. M. O. dos R. **Discretização de Controladores Contínuos**. [S.l.]: DEEC FEUP.<https://repositorio-aberto.up.pt/bitstream/10216/11227/2/Texto%20integral.pdf>. Acesso: 17 de novembro de 2018, 1996. 8

APÊNDICE A

Código Fonte Aquecimento Constante

```
int value;
long previousMillis2=0; //PRINT VALORES
long interval2=1000; //PRINT VALORES
const int sensorValue = A0;
int button = digitalRead(12); // botao de parada
float temperatura;

void setup()
{
  Serial.begin(9600);
}
void loop()
{
  //aquecimento das resistencias
  analogWrite(9,255);
// CONDICAO PARADA MATLAB
  button = digitalRead(12);
  temperatura = (float(analogRead(sensorValue))*5/(1023))/0.01;
//INTERVALO QUE ATUALIZA VALORES COMUNICACAO SERIAL
 unsigned long currentMillis2=millis();
  if (currentMillis2-previousMillis2>interval2){
    previousMillis2=currentMillis2;
//print na tela valores de temperatura e o tempo em segundos
    Serial.print(temperatura);
    Serial.print("\n");
    Serial.print("tempo:␣"); Serial.print(currentMillis2/1000);
    Serial.print("\n");
   }
}
```

APÊNDICE B

Código Fonte do Matlab para Obter a FT da Planta

```
clear
clc
%% inicio COMUNICACAO SERIAL
comecar_serial=1;
terminar_serial=0;
if terminar_serial==1
    fclose(s)
    delete(s)
    delete(instrfind)
end

if comecar_serial==1
    s = serial('COM3','BAUD',9600);
    fopen(s)
end

% DEFINE O EIXO Y
yMax = 100; %Máximo valor de y
yMin = 0;   %Mínimo valor de y
plotGrid = 'on';
min = -1; % y mínimo - no gráfico
max = 100; % y maximo no gráfico
delay = 0;

%% DECLARACAO DE VARIAVEIS
time = 3000;
periodo=5; %segundos
BOTAO=0;
tensao_saida_lm35 = 0;
temperatura_saida=0;
setpoint=1; %SETPOINT
output_saida_pid=0;
%%
count = 0; % CONTADOR

%% CONFIGURAÇO DO GRÁFICO
plotGraph_temperatura_saida = plot(time, temperatura_saida, 'DisplayName', 'Temperatura C');
hold on
tic
while ishandle(plotGraph_temperatura_saida) && BOTAO==0 %LOOP PARA PLOTAR EM TEMPO REAL
%RECEBE AS VARIÁVEIS DO ARDUINO
 if terminar_serial==0
    for i=1:5;
    VARIAVEIS =fscanf(s,'%f');
    RESP(i)=VARIAVEIS(1);
 end
 TEMPERATURA=RESP(1);
 BOTAO=RESP(4);
 SETPOINT=RESP(3);
 OUTPUT=RESP(2);
 TENSAO=RESP(5);
 end
%% VARIAVEIS RECEBEM DADOS SERIAL DO ARDUINO
 tensao_lm35=TEMPERATURA/100;
 temperatura=TEMPERATURA;
 tensao_pwm=TENSAO;
 setpoint=SETPOINT;
%% time RECEBE TEMPO REAL
 count = count + 1;
 time(count) = toc;
%% VARIAVEIS A SEREM PLOTADAS
 tensao_saida_lm35(count) = tensao_lm35(1); % tensao_saida_lm35 tensao saida lm35
 temperatura_saida(count)=temperatura(1); % temperatura
 setpoint_saida(count)=setpoint(1);
 output_saida_pid(count)=tensao_pwm(1);
%CONFIGURA NOMES DOS EIXOS, TÍTULO E LEGENDA DO PLOT
 title('Sistema sem controle','FontSize',15);
 xlabel(xLabel,'FontSize',15);
 ylabel(yLabel,'FontSize',15);
 set(plotGraph_temperatura_saida,'XData',time,'YData',temperatura_saida); % temperatura
 legend('Location','northeastoutside')
 delete(findall(gcf,'type','annotation'))
 dim = [.83 .55 .3 .3];
 str = sprintf('Temperatura: %0.2f C',temperatura_saida(count));
 annotation('textbox',dim,'String',str,'FitBoxToText','on');
```

```
76    axis([yMin yMax min max]);
77    grid(plotGrid);
78    axis([0 time(count) min max]);
79    %ATUALIZA O GRÁFICO
80    pause(delay);
81    if (time(count)>1800)
82        BOTAO=1;
83    end
84  end
85  disp('Plot Closed and arduino object has been deleted');
86  %FECHA A COMUNICAÇO SERIAL COM O ARDUINO
87  fclose(s)
88  delete(s)
89  delete(instrfind)
```

APÊNDICE C

Código Fonte Controle PID no Microcontrolador

```
1
2  //Declaracao das variaveis
3  float vetor_temp[4];
4  float temp_atual=0;
5  float input_atual=0;
6  float output_atual=0, temp_pwm=0, limite_inf=0;
7  float setp_atual=0;
8  float output[4];
9  float input[4];
10 float setp[4];
11 int value, unico=0;
12 int count=0;
13 float count_cooler=0, cooler_pwm=0, angulo=0, pi=3.1415, periodo=64; //variaveis acao do cooler
14 float tensao=0;
15 float Setpoint=40;
16 long previousMillis_temp=0;
17 long previousMillis_euler=0;
18 long previousMillis1=0; //SETPOINT
19 long previousMillis2=0; //PRINT VALORES
20 long interval_temp=1000;
21 long interval_euler=1000;
22 long interval1=200000; //SETPOINT
23 long interval2=1000; //PRINT VALORES
24 //
25 float ki=0.373; //ELEMENTO INTEGRADO DO PID
26 //anti windup h(t)= constante*(sinal_nao_saturado-sinal_saturado)
27 float limite=9;//sinal_nao_saturado
28 float k=-5; //constante anti windup
29 const int sensorValue = A0;
30 int button = digitalRead(12);
31 float temperatura;
32
33 void setup()
34 {
35   Serial.begin(9600);
36 }
37
38 void loop()
39 {
40   temperatura = (float(analogRead(sensorValue))*5/(1023))/0.01;
41
42 // CONDICAO PARADA MATLAB
43   button = digitalRead(12);
44
45   //A CADA 1s MUDA O VALOR DO Setpoint
46  unsigned long currentMillis1=millis();
47   if (currentMillis1-previousMillis1>interval1){
48     previousMillis1=currentMillis1;
49     if (Setpoint==40){
50       Setpoint=50;
51     } else Setpoint=40;
52   }
53
54  // CRIA LISTA DE VALORES DA TEMPERATURA
55  unsigned long currentMillis_temp=millis();
56   if (currentMillis_temp-previousMillis_temp>interval_temp){
57     previousMillis_temp=currentMillis_temp;
58       if (count<=3){
59         vetor_temp[count]=temperatura;
60         setp[count]=Setpoint;
61         temp_atual=vetor_temp[count];
62         input[count]=Setpoint-temperatura;
63         input_atual=input[count];
64         setp_atual=setp[count];
65        }
66      if (count>3){
67       vetor_temp[0]=vetor_temp[1];
68       input[0]=input[1];
69       setp[0]=setp[1];
70       //
71       vetor_temp[1]=vetor_temp[2];
72       input[1]=input[2];
73       setp[1]=setp[2];
74       //
75       vetor_temp[2]=vetor_temp[3];
```

```
76          input[2]=input[3];
77          setp[2]=setp[3];
78          //
79          vetor_temp[3]=temperatura;
80          input[3]=Setpoint-vetor_temp[3];
81          setp[3]=Setpoint;
82          temp_atual=vetor_temp[3];
83          input_atual=input[3];
84          setp_atual=setp[3];
85          }
86    //}
87
88   // PID EQUACAO A DIFERENCAS
89
90   //MODIFICA A CONSTANTE DA TECNICA ANTIWINDUP K_t
91       if (Setpoint==40){
92       if (input_atual>=6){k=-1;}
93       if (input_atual<6){k=-10;}
94       }
95           else {
96       if (input_atual>=6){k=-0.5;}
97       if (input_atual<6){k=-20;}
98          }
99
100         //INICIO DA EQ A DIFERENCA
101      if (count==0){
102     output[count]=0;
103     output_atual=output[count];
104         }
105      if (count==1){
106     output[count]=1.91*output[count-1]+ 0.74*input[count-1];
107     if (output[count]>limite){output[count]=output[count-1]+(ki-(limite-output[count])*(k));} //BACK CALC
108     if(output[count]<0){output[count]=0;}
109     output_atual=output[count];
110         }
111         if (count==2) {
112 output[count]=-0.91*output[count-2]-1.17*input[count-2] + 1.91*output[count-1]+ 0.74*input[count-1];
113 if (output[count]>limite){output[count]=output[count-1]+(ki-(limite-output[count])*(k));} //BACK CALC
114 if(output[count]<0){output[count]=0;}
115           output_atual=output[count];
116           }
117         if (count==3) {
118 output[count]=0.46*input[count-3] -0.91*output[count-2]-1.17*input[count-2] + 1.91*output[count-1]+ 0.74*input[count-1];
119 if (output[count]>limite){output[count]=output[count-1]+(ki-(limite-output[count])*(k));} //BACK CALC
120 if(output[count]<0){output[count]=0;}
121           output_atual=output[count];
122           }
123
124           if (count>3){
125             output[0]=output[1];
126             output[1]=output[2];
127             output[2]=output[3];
128             output[3]=0.46*input[0] -0.91*output[1]-1.17*input[1] + 1.91*output[2]+ 0.74*input[2];
129             if (Setpoint==50 && unico==0){output[3]=7; unico=1;} //REINICIA O OUTPUT PARA MANTER A RELACAO DA
                    ↪ DIFERENCA DE TEMPERATURA
130             if(Setpoint==40){unico=0;}
131             if (output[3]>limite){output[3]=output[2]+(ki-(limite-output[3])*(k));} //BACK CALC
132             if(output[3]<0){output[3]=0;} //REJEITA VALORES NEGATIVOS
133             output_atual=output[3];
134             }
135
136 // PWM OUTPUT
137    //TEMPERATURA
138 temp_pwm=output_atual*255/limite;
139 if(temp_pwm>255){temp_pwm=255;}
140 analogWrite(9,temp_pwm);
141
142 if (count_cooler>=1200){
143  //ACAO DO COOLER
144 cooler_pwm = 40 + (sin(angulo)+1)*215/2;
145 analogWrite(10,cooler_pwm);
146 cooler_pwm=cooler_pwm*100/255;
147 angulo=angulo+5.625*pi/180;
148 if(angulo>=6.2832){angulo=0+5.625*pi/180;}
149 } count_cooler++;
150  //FIM ACAO COOLER
151
152   temp_pwm=temp_pwm*100/255; // VIZUALIZAR PWM EM TERMO DE PORCENTAGEM
153
154
```

```
155       if (count<=3){
156       count=count+1;}
157      }
158      //INTERVALO QUE ATUALIZA VALORES COMUNICACAO SERIAL
159     unsigned long currentMillis2=millis();
160      if (currentMillis2-previousMillis2>interval2){
161        previousMillis2=currentMillis2;
162
163          //INICIO DO PRINT MATLAB
164        Serial.print(temp_atual);
165        Serial.print("\n");
166
167        Serial.print(output_atual);
168        Serial.print("\n");
169
170        Serial.print(Setpoint);
171        Serial.print("\n");
172
173        Serial.print(button);
174        Serial.print("\n");
175
176        Serial.print(temp_pwm);
177        Serial.print("\n");
178
179        Serial.print(cooler_pwm);
180        Serial.print("\n");
181        //FIM PRINT MATLAB
182       }
183   }
```

APÊNDICE D

Código Fonte Plotar Ação de Controle *Online*

```
clear
clc

%% inicio COMUNICACAO SERIAL
comecar_serial=1;
terminar_serial=0;

if terminar_serial==1
    fclose(s)
    delete(s)
    delete(instrfind)
end

if comecar_serial==1
    s = serial('COM3','BAUD',9600);
    fopen(s)
end



yMax = 100; %valor maximo de y
yMin = 0; % valor minimo de y
plotGrid = 'on';
min = 10; % eixo y minimo
max = 60; % eixo y maximo
delay = 0;

%% DECLARACAO DE VARIAVEIS
time = 3000;
BOTAO=0;
tensao_saida_lm35 = 0;
temperatura_saida=0;
setpoint=1; %SETPOINT
output_saida_pid=0;
cooler_pwm_saida=0;
%%

 count = 0;
%CRIA AMBIENTE DO GRAFICO
figure('units','normalized','outerposition',[0 0 1 1]);
subplot(2,1,1);

plotGraph_temperatura_saida = plot(time, temperatura_saida, 'DisplayName', 'Temperatura C');
hold on
plotGraph_setpoint = plot(time, setpoint,'DisplayName', 'Referência' );

subplot(2,1,2);
plotGraph_output_saida_pid = plot(time, output_saida_pid,'DisplayName','Tenso PWM - Temperatura' );
hold on
plotGraph_cooler_pwm_saida = plot(time, cooler_pwm_saida,'DisplayName','Tenso PWM - Cooler' );

tic

while ishandle(plotGraph_temperatura_saida) && BOTAO==0 %LOOP QUE EXECUTA DURANTE TEMPO DO EXPERIMENTO
    % RECEBE VARIAVEIS DO ARDUINO
if terminar_serial==0
    for i=1:6;

        VARIAVEIS =fscanf(s,'%f');
    RESP(i)=VARIAVEIS(1);

    end
        TEMPERATURA=RESP(1);
        BOTAO=RESP(4);
        SETPOINT=RESP(3);
        OUTPUT=RESP(2);
        TENSAO=RESP(5);
        COOLER=RESP(6);
end

%% VARIAVEIS RECEBEM DADOS SERIAL DO ARDUINO
   tensao_lm35=TEMPERATURA/100;
    temperatura=TEMPERATURA;
          tensao_pwm=TENSAO;
          setpoint=SETPOINT;
```

```
76          cooler=COOLER;
77
78      %% time RECEBE TEMPO REAL
79          count = count + 1;
80          time(count) = toc;
81      %% VARIAVEIS A SEREM PLOTADAS
82          tensao_saida_lm35(count) = tensao_lm35(1); % tensao_saida_lm35 tensao saida lm35
83          temperatura_saida(count)=temperatura(1); % temperatura
84          setpoint_saida(count)=setpoint(1);
85          output_saida_pid(count)=tensao_pwm(1);
86              cooler_pwm_saida(count)=cooler(1);
87
88              %plot SUPERIOR DA TEMPERATURA E REFERENCIA
89          subplot(2,1,1);
90          title('Controle de Temperatura','FontSize',15);
91          xlabel(xLabel,'FontSize',15);
92          ylabel(yLabel,'FontSize',15);
93          set(plotGraph_temperatura_saida,'XData',time,'YData',temperatura_saida); % temperatura
94
95          set(plotGraph_setpoint,'XData',time,'YData',setpoint_saida); %temperatura
96          legend('Location','northeastoutside')
97  delete(findall(gcf,'type','annotation'))
98  dim = [.83 .55 .3 .3];
99  str = sprintf('Temperatura: %0.2f C',temperatura_saida(count));
100 annotation('textbox',dim,'String',str,'FitBoxToText','on');
101
102         axis([yMin yMax min max]);
103         grid(plotGrid);
104         axis([time(count)-50 time(count) min max]);
105
106         %PLOT INFERIOR NA TELA, TENSAO PWM DO COOLER E DO AQUECIMENTO DAS RESISTENCIAS
107         subplot(2,1,2);
108         title('Tenso PWM de Controle','FontSize',15);
109         xlabel(xLabel,'FontSize',15);
110         ylabel('Tenso (0-100%)','FontSize',15);
111         set(plotGraph_output_saida_pid,'XData',time,'YData',output_saida_pid);
112         set(plotGraph_cooler_pwm_saida,'XData',time,'YData',cooler_pwm_saida);
113
114         legend('Location','northeastoutside')
115 dim1 = [.83 .10 .3 .3];
116 dim2 = [.83 .06 .3 .3];
117 str1 = sprintf('Duty Cycle - Temperatura: %0.2f %%',output_saida_pid(count));
118 str2 = sprintf('Duty Cycle - Cooler: %0.2f %%',cooler_pwm_saida(count));
119 annotation('textbox',dim1,'String',str1,'FitBoxToText','on');
120 annotation('textbox',dim2,'String',str2,'FitBoxToText','on');
121 grid(plotGrid);
122 axis([time(count)-50 time(count) -1 101]);
123
124          %ATUALIZA O GRAFICO
125          pause(delay);
126     if (time(count)>1800)
127         BOTAO=1;
128         end
129 end
130 % CRIA FIGURA APOS TERMINO DO LOOP COM TODO O PERIODO DO EXPERIMENTO
131 figure('units','normalized','outerposition',[0 0 1 1]);
132 subplot(2,1,1);
133 x=time;
134 y=temperatura_saida;
135 y1=setpoint_saida;
136 plot(x,y, x,y1);
137 title('Controle de Temperatura','FontSize',15);
138 xlabel(xLabel,'FontSize',15);
139 ylabel(yLabel,'FontSize',15);
140 legend({'Temperatura C', 'Referência'},'Location','northeastoutside')
141 axis([yMin yMax min max]);
142 grid(plotGrid);
143 axis([0 time(count) min max]);
144
145 subplot(2,1,2);
146 x1=time;
147 y2=output_saida_pid;
148 y3=cooler_pwm_saida;
149 plot(x1,y2, x1, y3);
150 title('Tenso PWM de Controle','FontSize',15);
151 xlabel(xLabel,'FontSize',15);
152 ylabel('Tenso (0-100%)','FontSize',15);
153 legend({'Tensao PWM - Temperatura', 'Tenso PWM - Cooler'},'Location','northeastoutside');
154 grid(plotGrid);
155 axis([0 time(count) -1 101]);
```

```
156
157 % TERMINA A COMUNICACAO SERIAL
158 disp('Plot Closed and arduino object has been deleted');
159 fclose(s)
160 delete(s)
161 delete(instrfind)
```

A

Comparação dos Transistores

A Figura A.1 apresenta valores máximos, a temperatura de 25°C, do transistor BC548.
A Figura A.2 apresenta os valores máximos do transistor TIP31C, temperatura de 25°C.

Figura A.1 - Valores máximos BC548.

Maximum ratings (T_A = 25°C) **Grenzwerte (T_A = 25°C)**

			BC 546	BC 547	BC 548/549
Collector-Emitter-voltage	B open	V_{CE0}	65 V	45 V	30 V
Collector-Emitter-voltage	B shorted	V_{CES}	85 V	50 V	30 V
Collector-Base-voltage	E open	V_{CB0}	80 V	50 V	30 V
Emitter-Base-voltage	C open	V_{EB0}	6 V	6 V	5 V
Power dissipation – Verlustleistung		P_{tot}	500 mW [1])		
Collector current – Kollektorstrom (DC)		I_C	100 mA		
Peak Coll. current – Kollektor-Spitzenstrom		I_{CM}	200 mA		
Peak Base current – Basis-Spitzenstrom		I_{BM}	200 mA		
Peak Emitter current – Emitter-Spitzenstrom		$-I_{EM}$	200 mA		
Junction temp. – Sperrschichttemperatur		T_j	150°C		
Storage temperature – Lagerungstemperatur		T_S	-65...+ 150°C		

Fonte: <http://pdf.datasheetcatalog.com/datasheets/150/128380_DS.pdf>
Acesso: 16 de novembro de 2018.

Figura A.2 - Valores máximos TIP31C.

Absolute Maximum Ratings

Stresses exceeding the absolute maximum ratings may damage the device. The device may not function or be operable above the recommended operating conditions and stressing the parts to these levels is not recommended. In addition, extended exposure to stresses above the recommended operating conditions may affect device reliability. The absolute maximum ratings are stress ratings only. Values are at T_C = 25°C unless otherwise noted.

Symbol	Parameter		Value	Unit
V_{CBO}	Collector-Base Voltage	TIP31A	60	V
		TIP31C	100	
V_{CEO}	Collector-Emitter Voltage	TIP31A	60	V
		TIP31C	100	
V_{EBO}	Emitter-Base Voltage		5	V
I_C	Collector Current (DC)		3	A
I_{CP}	Collector Current (Pulse)		5	A
I_B	Base Current		1	A
T_J	Junction Temperature		150	°C
T_{STG}	Storage Temperature Range		-65 to 150	°C

Fonte: <http://www.mouser.com/ds/2/149/fairchild%20semiconductor_tip31a-549394.pdf>
Acesso: 16 de novembro de 2018.

B

Ziegler-Nichols Malha Aberta

A Figura B.1 apresenta a relação matemática entre L e T - encontrados analisando o gráfico do sistema em malha aberta - que resulta nos parâmetros Kc, Ti e Td do controlador PID.

Figura B.1 - Tabela Ziegler-Nichols.

Tipo de controlador	Kc	Ti	Td
P	T/L	∞	0
PI	0,9T/L	L/0,3	0
PID	1,2T/L	2L	0,5L