INSTITUTO FEDERAL DE GOIÁS
Engenharia de Controle e Automação

FAMILIARIZAÇÃO COM MICROCONTROLADOR - CONTROLE DE TEMPERATURA

Alexandre Alves Trindade

22 de novembro de 2018

RESUMO

O presente trabalho consiste em projetar controlador PID que obedeça a determinada lei 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. 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 obedecem ás especificações para este projeto.

CONTEÚDO

Pág.

LISTA DE FIGURAS

Pág.

LISTA DE TABELAS

Pág.

CAPÍTULO 1

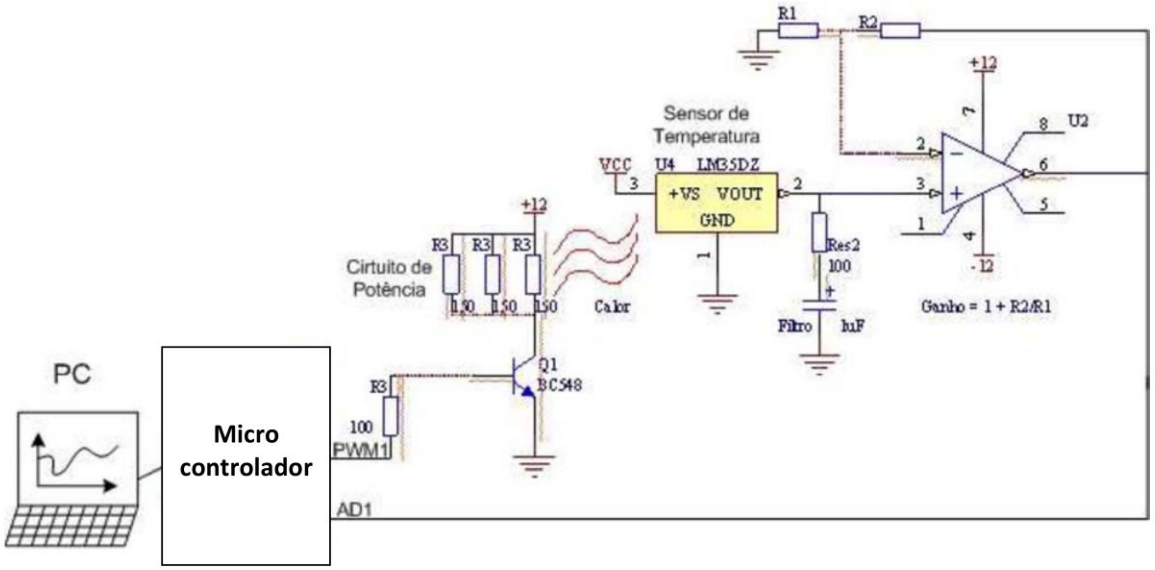
Introdução

O presente trabalho consiste em projetar controlador PID 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 circuito consiste de três resistores de 150Ω em paralelo alimentadas com 12V, é o circuito de potência. O transistor BC548 tem função de limitar a corrente elétrica que irá aquecer os resistores; capacitor de 100 μF é o filtro do sinal de saída do sensor de temperatura LM35; e amplificador operacional. 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.

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

Especificações do projeto:

a) Período de amostragem = 1s;

b) Referência é uma onda quadrada entre 40°C e 50°C, com período de 800s;

c) Resolução do valor de temperatura <0,1°C, faixa dinâmica = 0°C a 110°C;

d) Tempo de acomodação (2%) $\leq$ 400s;

e) Sobrepasso percentual $\leq$ 20%;

f) Tempo de pico $\leq$ 40s.

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;

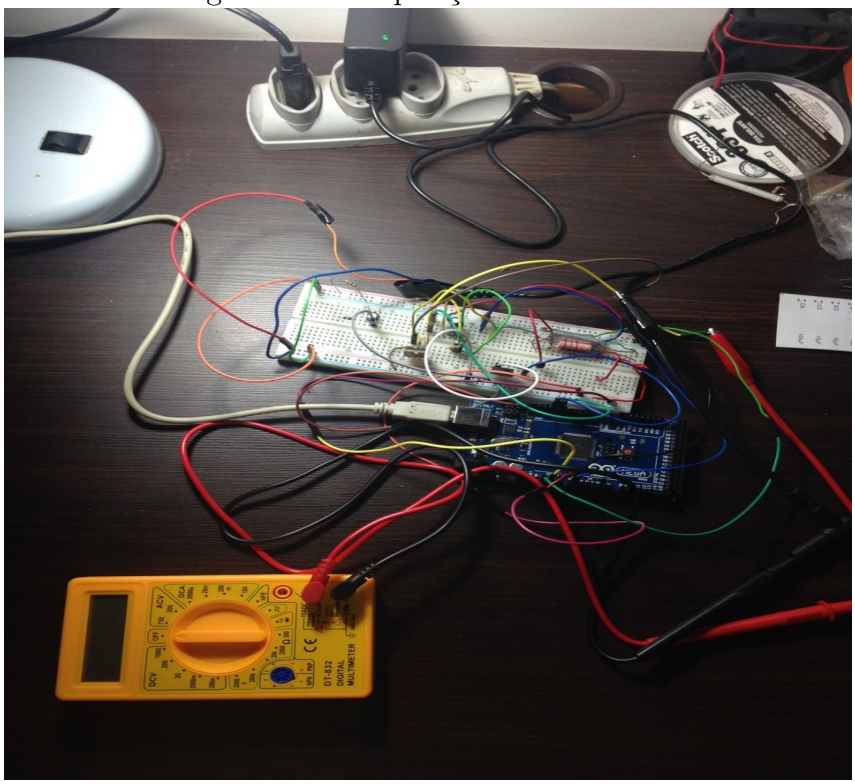
7) Etapa de testes e análise dos resultados.

2.1 MONTAGEM DO EXPERIMENTO

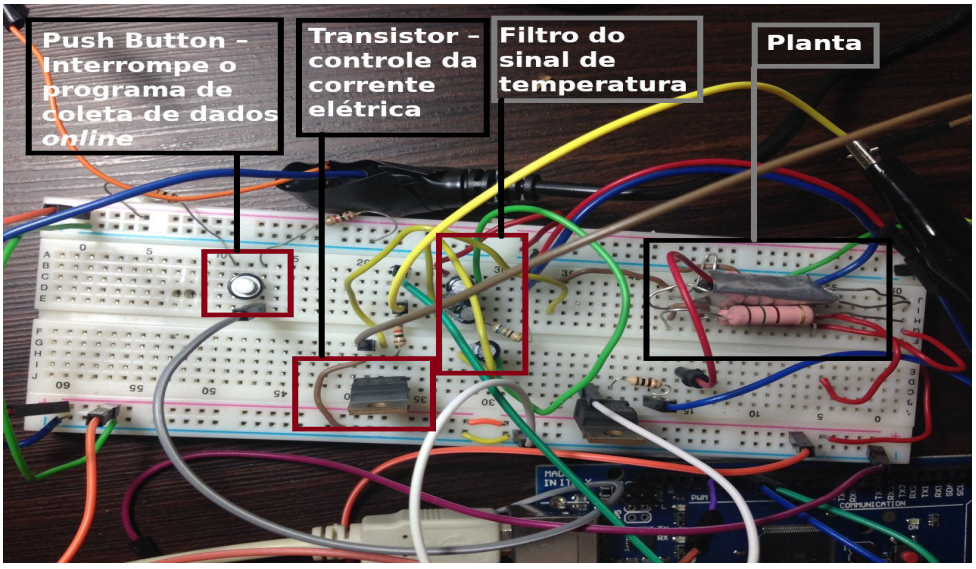
Os materiais utilizados no experimento são: 1 fonte 12VCC (corrente 1A), 1 multímetro digital, 3 resistores 1kΩ, 1 resistor 68Ω, 4 resistores 150Ω (potência 5W), 1 transistor TIP31C, 3 capacitores de lítio 10μF, 1 *protoboard*, 1 sensor de temperatura LM35, 1 *push button*, 1 microcontrolador Arduino Mega 2560, 1 placa de alumínio (para aumentar o tempo de aquecimento), cabos e *jumpers*.

A Figura 2.1 (a) apresenta a bancada em que os experimentos foram feitos, a Figura 2.1 (b) indica partes específicas dos componentes. O botão, ao ser pressionado, interrompe a visualização da ação de controle em tempo real. Ao comparar o esquema elétrico Figura 2.2 com o da Figura 1.1 é notado as modificações realizadas.

Figura 2.1 - Disposição do *hardware*.



(a) Bancada onde experimentos foram realizados.

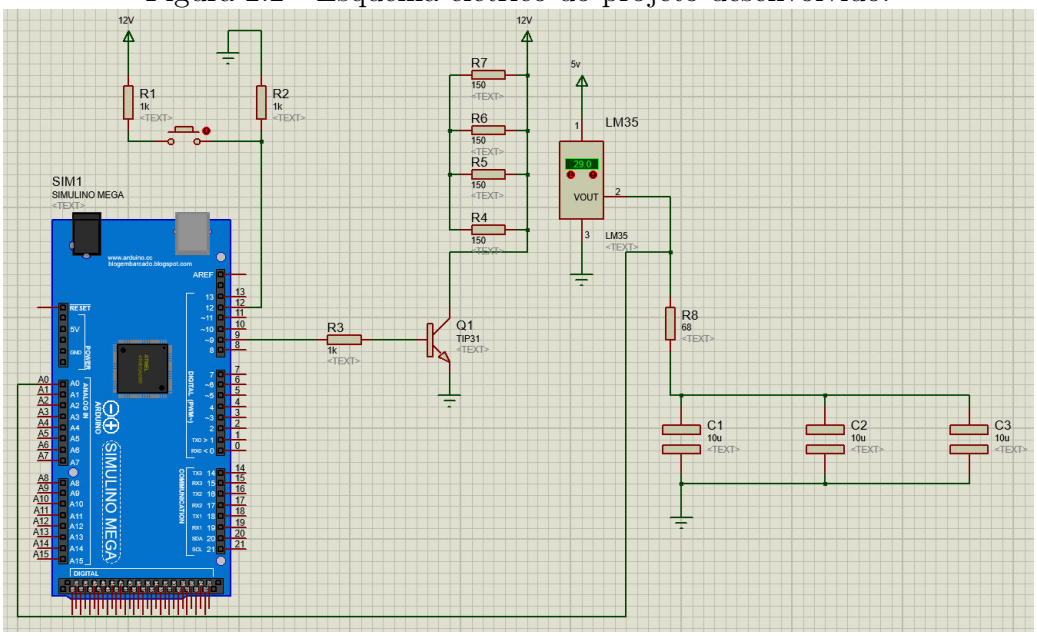


(b) Vista superior da *protoboard*.

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.

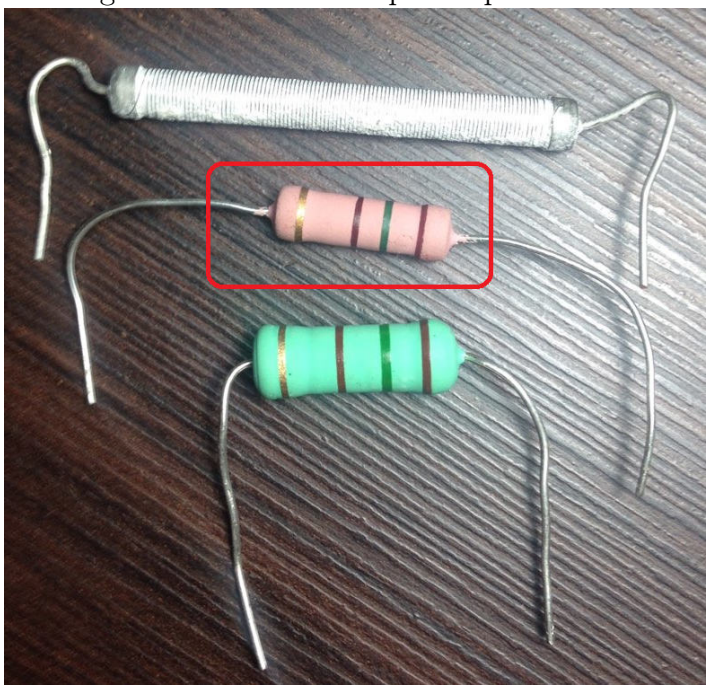
O desafio de obter o tempo de pico menor que 40s, resultou na modificação do transistor BC548 para o TIP31C, como pode ser verificado no Anexo A deste trabalho, os valores de corrente do primeiro transistor são de 100mA ou 200mA, enquanto do segundo são de 1A a 5A. Três tipos de resistores foram testados, Figura 2.3, e variadas combinações dos mesmos. Todos são de 150Ω, de cima para baixo 10W, 5W e 3W.

Figura 2.2 - Esquema elétrico do projeto desenvolvido.



O resistor destacado na Figura 2.3 foi identificado com o que tem sua temperatura elevada em menor tempo, portanto foi utilizado em paralelo com resistência equivalente de 37,5Ω. A placa de alumínio foi colocada em contato com os resistores e o sensor LM35, o alumínio tem baixo custo e é um dos metais que aquece mais rápido (calor específico 0,22 cal/g°C), ver Anexo B.

Figura 2.3 - Resistores para aquecimento.



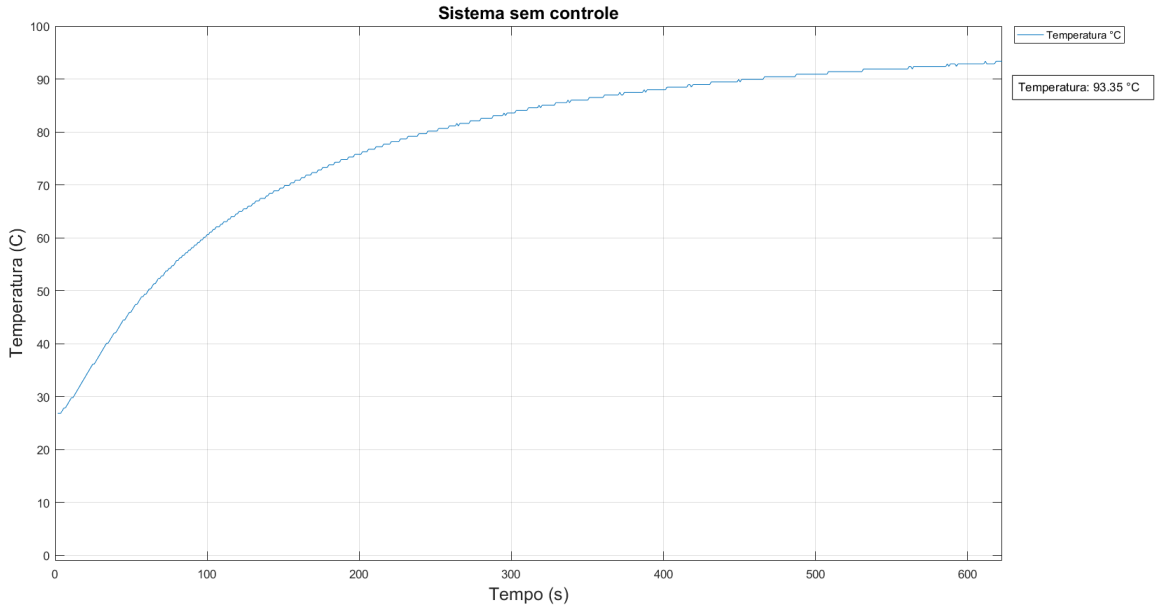
2.2 FUNÇÃO DE TRANSFERÊNCIA DA PLANTA

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 2.4, 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 2.4 o período do experimento foi de 600s e foi interrompido, tendo em vista temperatura alcançada de 93,35°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 2.4 - Leitura de temperatura aquecimento contínuo.



$$data = iddata(temperatura\_saida, A, 1)$$
$$sys = tfest(data, 1) \tag{2.1}$$

Figura 2.5 - FT da planta.

```
sys =

  From input "u1" to output "y1":
      0.637
  ------------
  s + 0.006853

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: 95.94% (simulation focus)
FPE: 0.4629, MSE: 0.4585
```

FT PLANTA

A Figura 2.5 apresenta a FT da planta, resultado da sequência descrita nesta seção. A adequação aos dados com 1 pólo e nenhum zero foi de 95,94%.

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 2.6 constitui a parte gráfica necessária para este método, a tangente traçada permite encontrar o valor de L e T. O Anexo C 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.

Figura 2.6 - Método de Ziegler-Nichols.

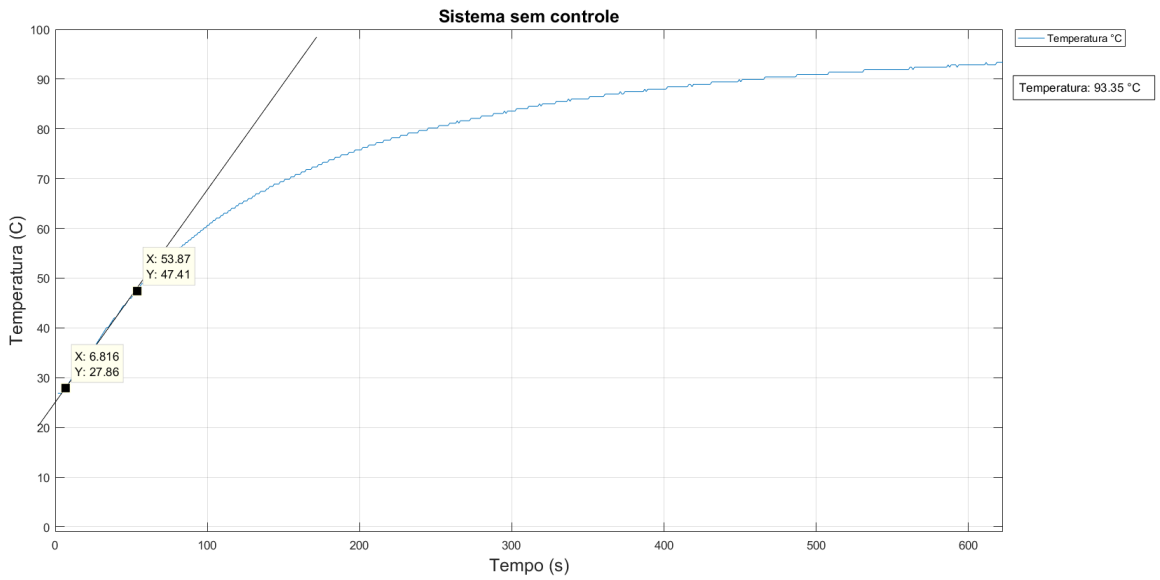


Tabela 2.1 - Parâmetros do controlador.

T=47,05	L=6,82		
	Kp	Ti	Td
P	6,90	∞	0
PI	6,21	22,72	0
PID	8,28	13,63	3,41

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

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

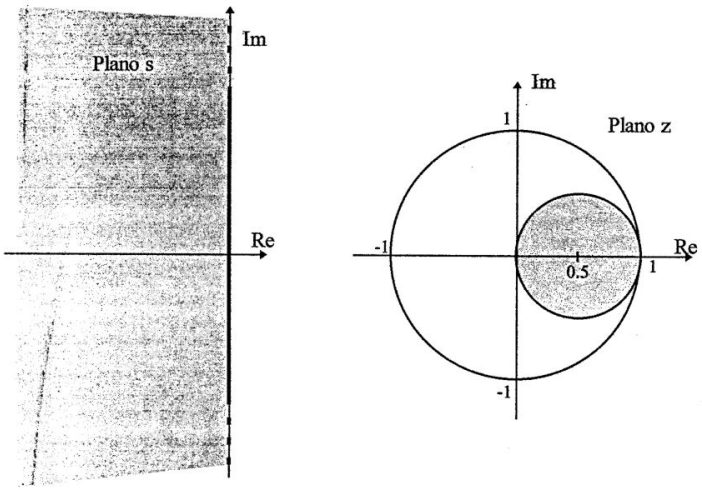
A Equação 2.4 resulta da aplicação do método descrito nesta seção, a função de transferência no domínio do tempo contínuo do controlador PID.

$$
G_c(s) = \frac{28,23s^2 + 8,284s + 0,6077}{s} \tag{2.4}
$$

2.4 MÉTODO DE EULER

Neste método a equação que relaciona **s** e **z** foi apresentada no capítulo 1. A Figura 2.7 apresenta o mapeamento dos pólos do tempo contínuo para o domínio discreto.

Figura 2.7 - 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.

$$\text{FT CONTROLADOR}$$
$$G_c(z) = \frac{37,12z^2 - 64,74z + 28,23}{z^2 - z} \tag{2.5}$$
$$\text{FT PLANTA}$$
$$G_p(z) = \frac{0,6327z}{z - 0,9932}$$

2.5 LUGAR DAS RAÍZES

A presente seção apresenta o desenho do LGR, foi utilizado a ferramenta *sisotool*. A Figura 2.8 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 2.9 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 40s, tempo de acomodação menor ou igual 400s e percentual de sobressinal até 20%, Figura 2.10, o resultado desta modificação feita no controlador. A Equação 2.6 é a FT do controlador, resultado da adequação indicada nesta seção.

Figura 2.8 - Configuração da arquitetura do sistema.

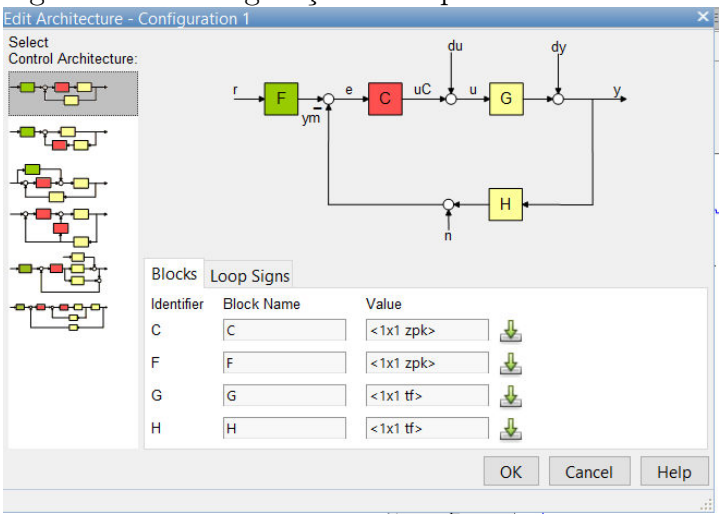


Figura 2.9 - Resposta ao degrau, e LGR.

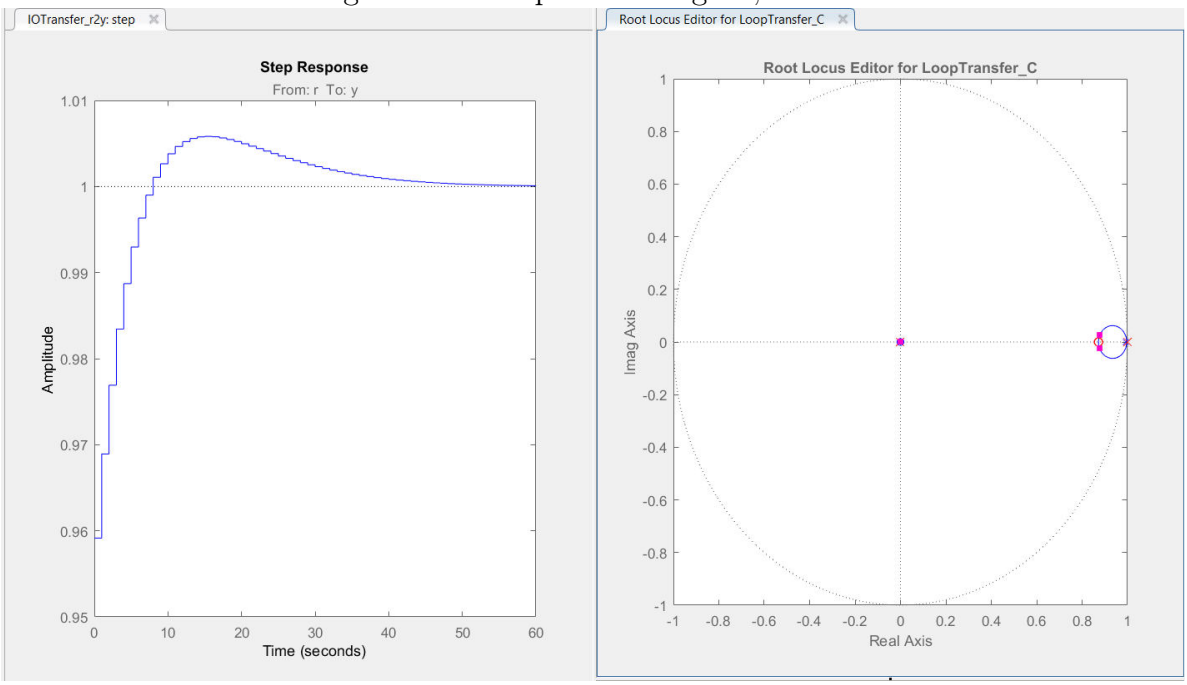
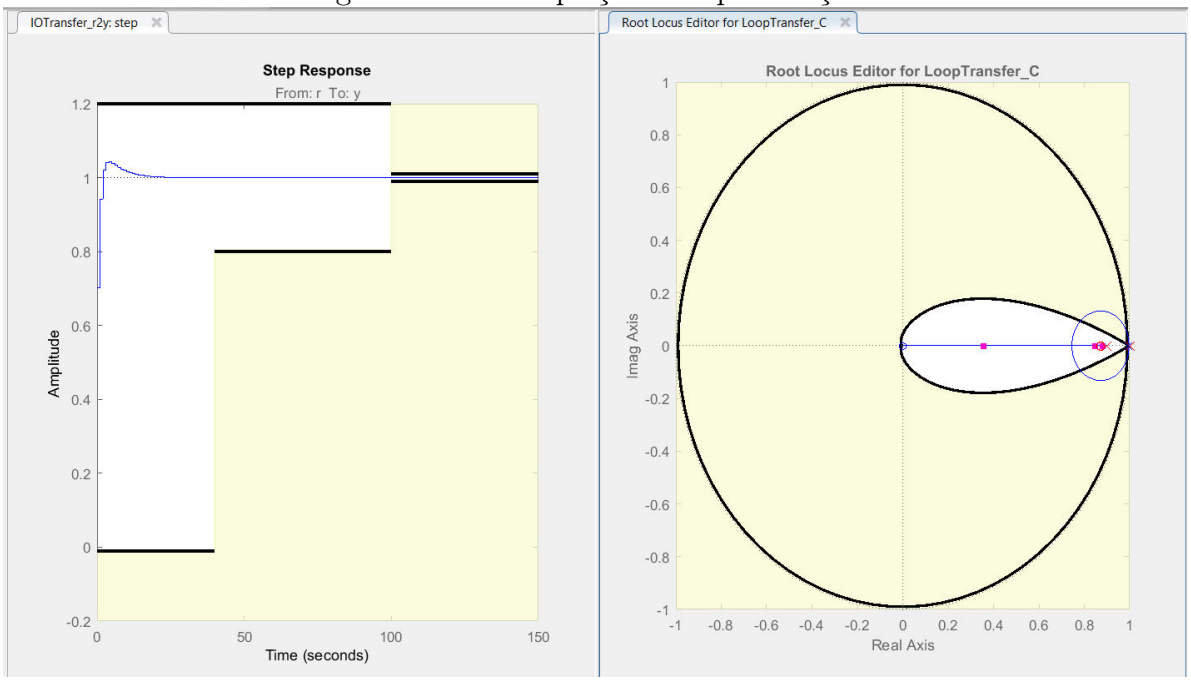


Figura 2.10 - Adequação às especificações.

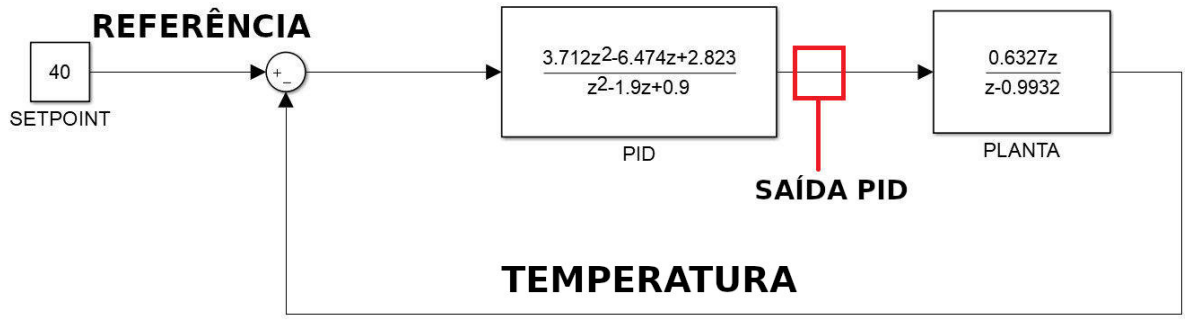


$$G_c(z) = \frac{3,712z^2 - 6,474z + 2,823}{(z - 0,9)(z - 1)} \tag{2.6}$$

2.6 EQUAÇÃO A DIFERENÇAS

A equação a diferenças do controlador foi obtida pelo método de divisão longa, Equação 2.7, foi considerado para implementar no microcontrolador dois tempos anteriores (kT - 2). A Figura 2.11 apresenta o diagrama de blocos do sistema em malha fechada. 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 indicação da Figura 2.11 é a saída do controlador, valor que é utilizado para modificar o *duty cycle* da tensão PWM que aquece as resistências.

Figura 2.11 - Sistema malha fechada.



$$\begin{aligned} erro &= Setpoint - Temperatura; \\ G_c(kT) &= 3,712 \cdot erro[kT] + 0,579 \cdot erro[kT-1] + 0,582 \cdot erro[kT-2] \end{aligned} \tag{2.7}$$

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 Tabelas 3.1 - 3.3 mostram os valores da saída do controlador para diferentes valores de temperatura. O intuito é familiarizar ao comportamento numérico do controlador, e entender seu funcionamento para determinada faixa de temperatura, seja distante, próximo ou superior ao valor do *setpoint*.

Tabela 3.1 - Valores de saída do controlador.

	kT-2	kT-1	kT
Setpoint	40	40	40
Temperatura	20	20.5	21
Output 93,46			

Tabela 3.2 - Valores de saída do controlador.

	kT-2	kT-1	kT
Setpoint	40	40	40
Temperatura	30	31	32
Output 40,72			

Tabela 3.3 - Valores de saída do controlador.

	kT-2	kT-1	kT
Setpoint	40	40	40
Temperatura	40	40.5	41
Output -4			

Os valores do *output* são similares para 40 e 50°C, para mesma diferença de temperatura em relação à temperatura atual. Os valores do *output* são equiparados a faixa de 0 a 100% *duty cycle* da tensão de controle PWM. Foi testado diferentes valores máximos e mínimos que limitam a saída do PID, com o objetivo de observar quais atendem às especificações para esse projeto, e apontar para qual faixa de trabalho da saída do PID a ação de controle é mais estável e rápida.

A Tabela 3.4 constitui como primeira configuração dos valores da saída do PID. O código fonte do microcontrolador para este teste específico, como base dos demais testes apresentados neste capítulo é o Apêndice C, e o código fonte do Matlab que gera o gráfico *online* o Apêndice D.

O início da ação de controle, tem a temperatura inicial menor que 30°C e alcança o *setpoint* em tempo inferior a 40s, Figura 3.1. A Figura 3.2 tempo entre 450 e 500 segundos, estabilização próxima de 50°C. O período completo do experimento é registrado na Figura 3.3.

Tabela 3.4 - Faixa valor de saída PID.

Output	*Duty Cycle*(%)	Valor Arduino	erro(°C)
20	100	255	5
-3	0	0	1

Figura 3.1 - Início do controle de temperatura.

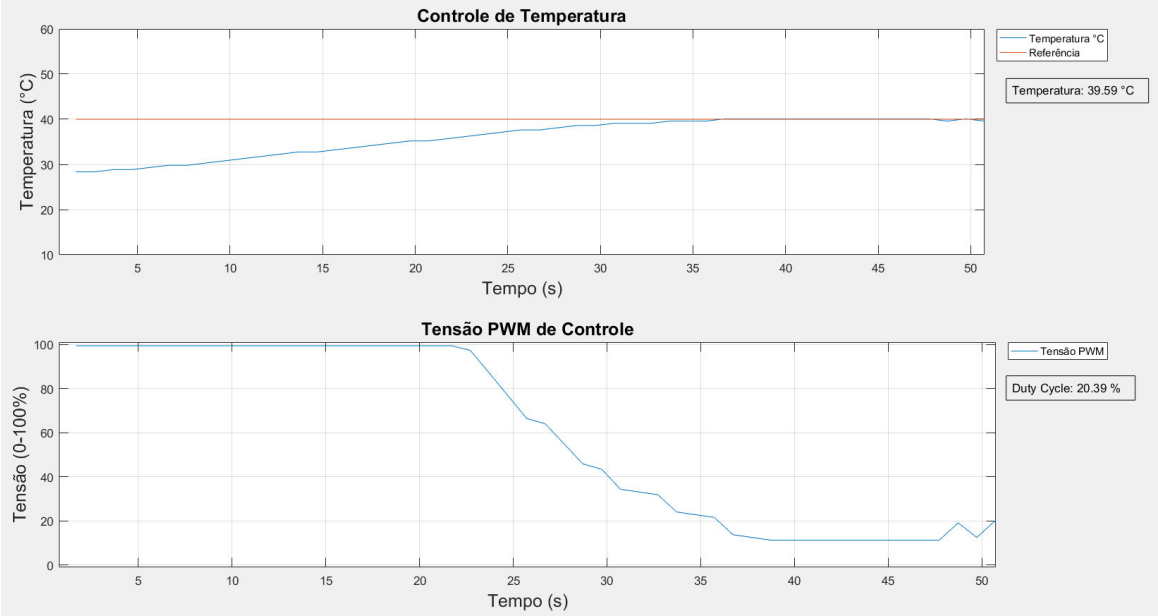


Figura 3.2 - Setpoint = 50°C.

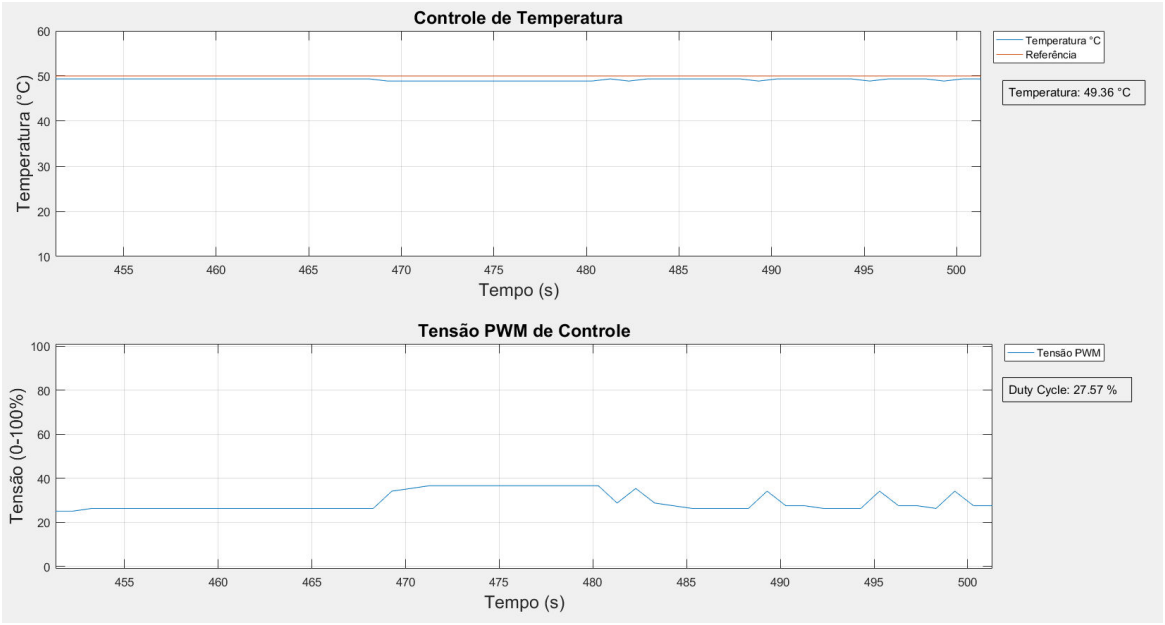
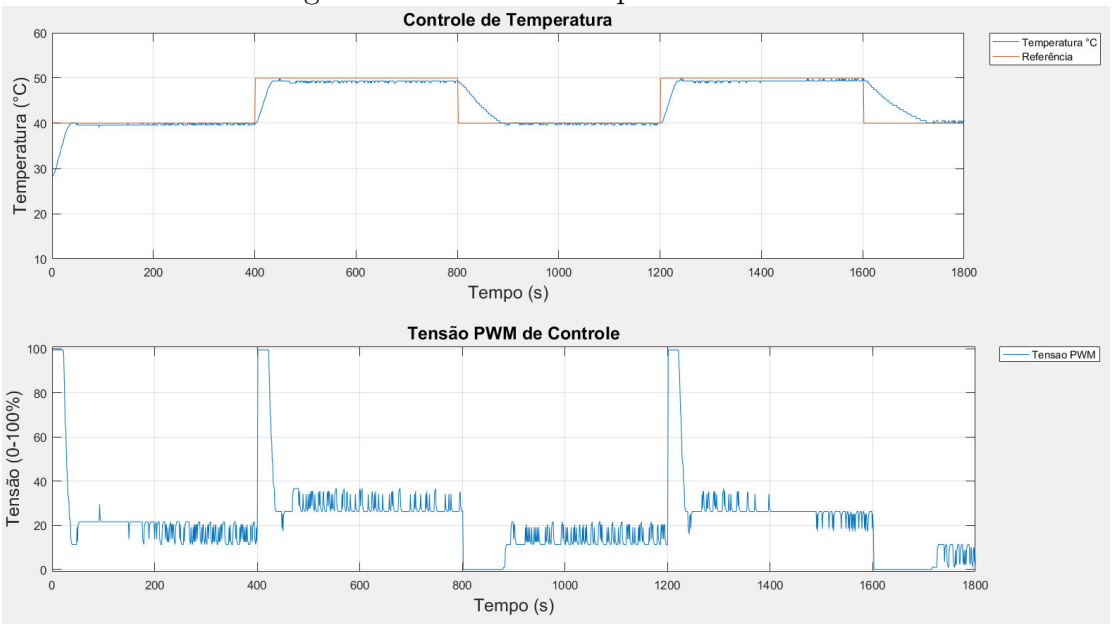


Figura 3.3 - Período completo de análise.



Com o objetivo de ter reposta mais rápida à mudança do valor do *setpoint* o valor da saída do PID tem o intervalo modificado para -4 a 5, Tabela 3.5. Na prática a ação de controle irá manter 100% do *duty cycle* até atingir um valor que tende ao *setpoint*, tendo sobressinal maior do que a análise feita com o intervalo -3 a 20 de saída do PID.

Tabela 3.5 - Faixa valor de saída PID.

Output	*Duty Cycle*(%)	Valor Arduino	erro(°C)
5	100	255	1
-4	0	0	1

Figura 3.4 - Início do experimento.

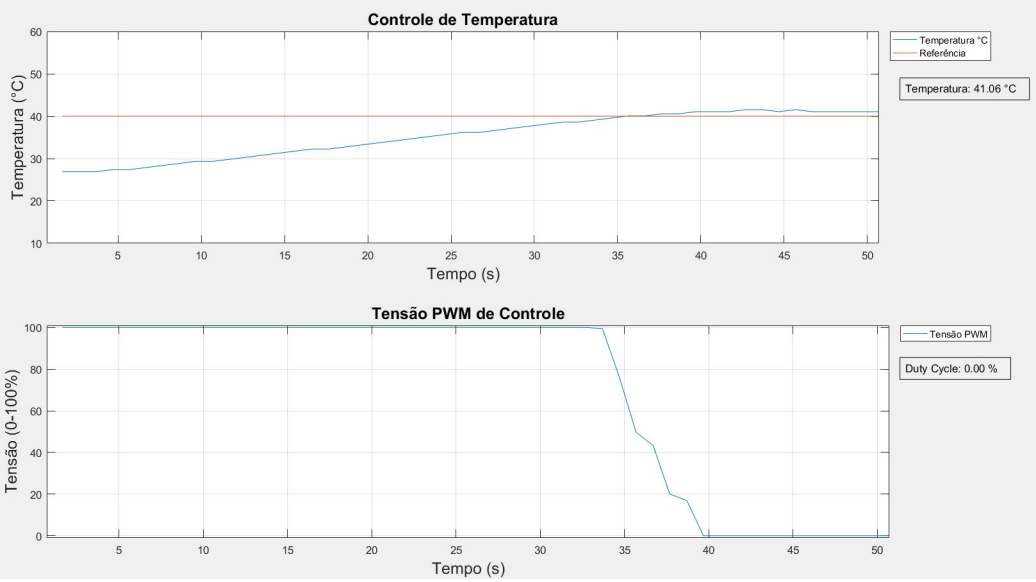
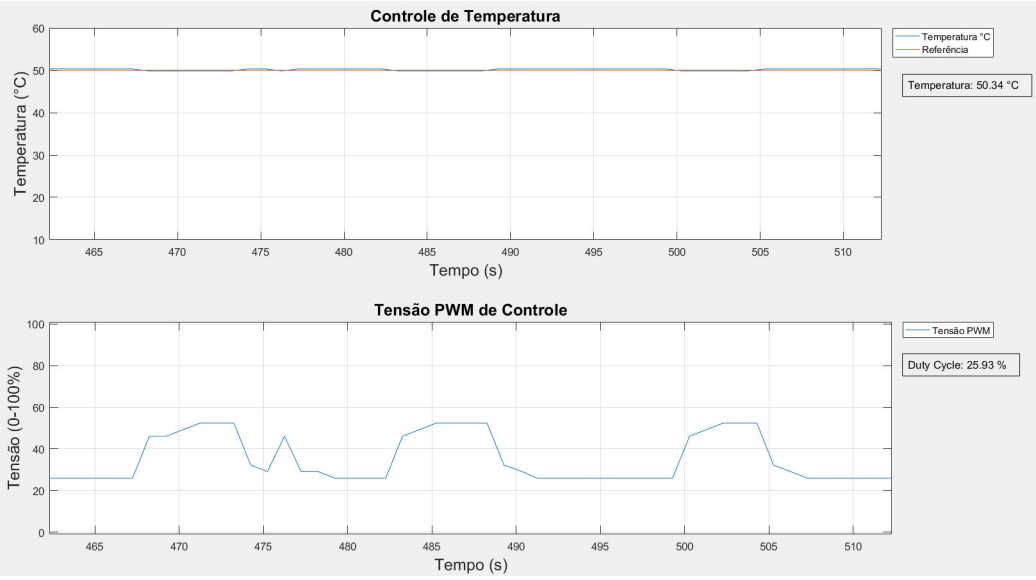


Figura 3.5 - Setpoint=50°C.



As Figuras 3.4 - 3.6 correspondem ao início, período entre 460 e 510 segundos, e período completo do experimento. De modo geral, neste segundo teste, é notado as seguintes mudanças: tensão PWM oscila com valor maior de *duty cycle* durante o período de estabilização; a estabilização parece estar mais próximo do *setpoint*, ainda que o erro seja desprezível em relação ao primeiro teste; maior tempo de tensão máxima (subida) e mínima (descida).

Figura 3.6 - Período completo.

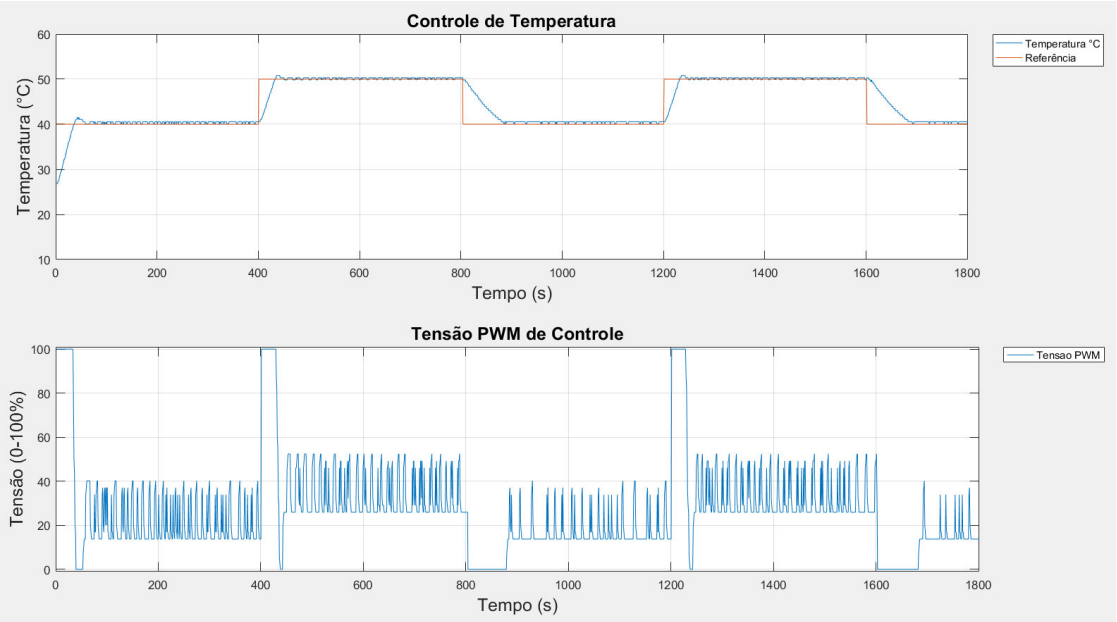
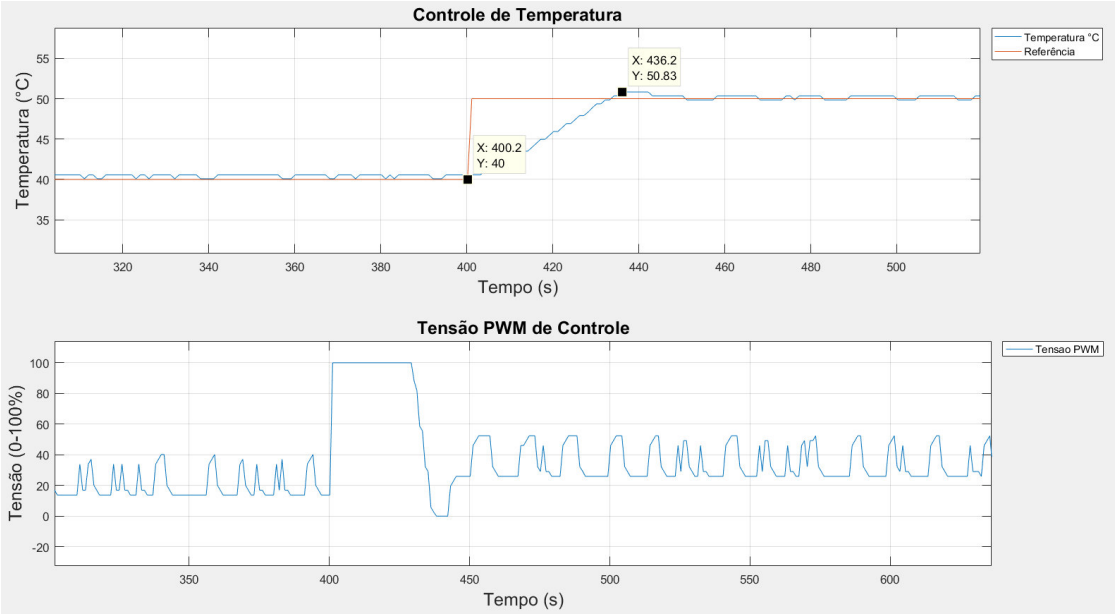
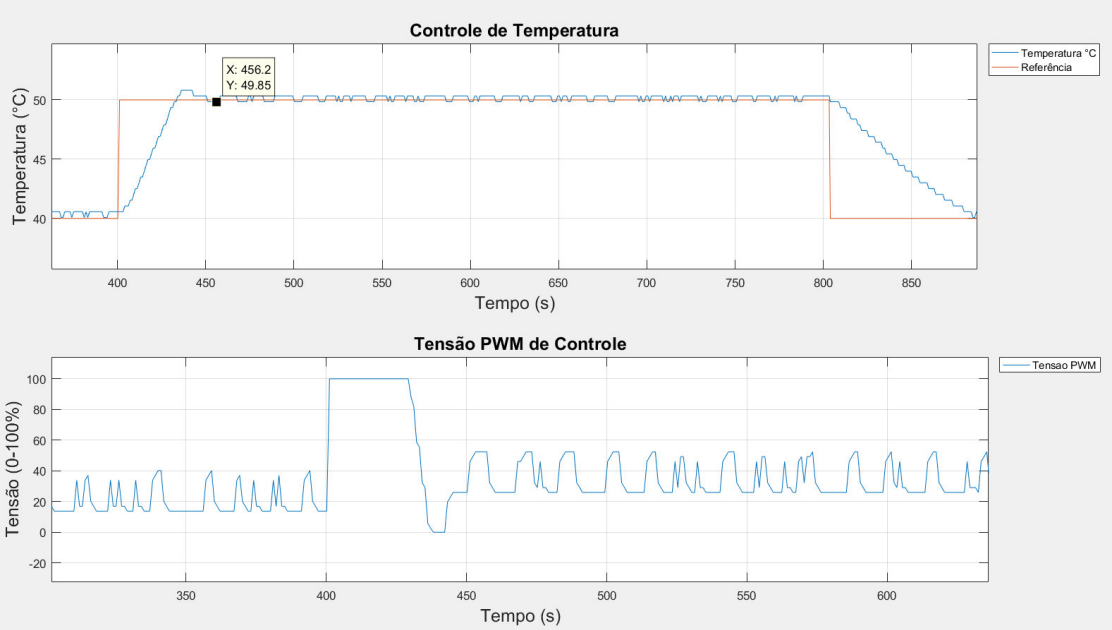


Figura 3.7 - Tempo de pico.



A Figura 3.7 mostra o tempo de pico menor que 40s, a Figura 3.8 a estabilização do valor de temperatura ocorre em aproximadamente 55s, o valor então oscila entre 49,85 e 50,34°C.

Figura 3.8 - Estabilização.



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. Foi registrado neste trabalho duas faixas de valores (-3,20) e (-4,5) ambas acompanham o valor do *setpoint* e obedecem às especificações, porém, ao diminuir o erro (setpoint - temperatura) e limitar, consequentemente, o valor da saída do PID, é obtido resposta mais rápida no aquecimento e resfriamento (mudança de *setpoint*). 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. 7

FILIPEFLOP.COM. **Protoboard 830 Pontos**. [S.l.]: <https://www.filipeflop.com/produto/protoboard-830-pontos/#tab-description>. Acesso: 17 de novembro de 2018. 7

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. 10

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  //DEFINIÇO DE VARIÁVEIS
2  float vetor_temp[3];
3  float temp_atual=0;
4  float input_atual=0;
5  float output_atual=0;
6  float setp_atual=0;
7  float output[3];
8  float input[3];
9  float setp[3];
10 int value;
11 int count=0;
12 float tensao=0;
13 float Setpoint=40;
14 long previousMillis_temp=0;
15 long previousMillis_euler=0;
16 long previousMillis1=0; //SETPOINT
17 long previousMillis2=0; //PRINT VALORES
18 long interval_temp=1000;
19 long interval_euler=1000;
20 long interval1=400000; //SETPOINT
21 long interval2=1000; //PRINT VALORES
22
23 const int sensorValue = A0; //LEITURA DO SENSOR LM35
24 int button = digitalRead(12); // BOTAO DE PARADA DO MATLAB
25 float temperatura;
26
27 void setup()
28 {
29   Serial.begin(9600);
30 }
31
32 void loop()
33 {
34   temperatura = (float(analogRead(sensorValue))*5/(1023))/0.01; // CONVERSAO PARA TEMPERATURA (SINAL ANALOGICO)
35
36 // CONDICAO PARADA MATLAB
37   button = digitalRead(12);
38
39   //A CADA 1s MUDA O VALOR DO Setpoint
40  unsigned long currentMillis1=millis();
41   if (currentMillis1-previousMillis1>interval1){
42     previousMillis1=currentMillis1;
43     if (Setpoint==40){
44       Setpoint=50;
45     } else Setpoint=40;
46   }
47
48  // CRIA LISTA DE VALORES DA TEMPERATURA
49  unsigned long currentMillis_temp=millis();
50   if (currentMillis_temp-previousMillis_temp>interval_temp){
51     previousMillis_temp=currentMillis_temp;
52       if (count<=2){
53         vetor_temp[count]=temperatura;
54         setp[count]=Setpoint;
55         temp_atual=vetor_temp[count];
56         input[count]=Setpoint-temperatura;
57         input_atual=input[count];
58         setp_atual=setp[count];
59        }
60      if (count>2){
61       vetor_temp[0]=vetor_temp[1];
62       input[0]=input[1];
63       setp[0]=setp[1];
64       vetor_temp[1]=vetor_temp[2];
65       input[1]=input[2];
66       setp[1]=setp[2];
67       vetor_temp[2]=temperatura;
68       input[2]=Setpoint-vetor_temp[2];
69       setp[2]=Setpoint;
70       temp_atual=vetor_temp[2];
71       input_atual=input[2];
72       setp_atual=setp[2];
73       }
74
75  // PID EQUACAO A DIFERENCAS
```

```
76
77      if (count==0){
78
79     output[count]=3.712*setp[count]-3.712*vetor_temp[count];
80     output_atual=output[count];
81
82
83          }
84       if (count==1){
85
86 output[count]=3.712*setp[count]-3.712*vetor_temp[count] + 0.579*setp[count-1] - 0.579*vetor_temp[count-1];
87     output_atual=output[count];
88
89          }
90          if (count==2) {
91
92 output[count]=3.712*setp[count]-3.712*vetor_temp[count]+0.579*setp[count-1]-0.579*vetor_temp[count-1]+0.582*setp[count
       ↪ -2]-0.582*vetor_temp[count-2];
93            output_atual=output[count];
94
95            }
96            if (count>2){
97              output[0]=output[1];
98              output[1]=output[2];
99              output[2]=3.712*setp[2]-3.712*vetor_temp[2]+0.579*setp[1]-0.579*vetor_temp[1]+0.582*setp[0]-0.582*vetor_temp[0];
100             output_atual=output[2];
101
102             }
103
104 // PWM OUTPUT
105
106       if (output_atual>=0){
107   if (output_atual>20){output_atual=20;}
108   tensao=(2+output_atual+1)*11.02;
109   }
110 if (output_atual<0){
111   if (output_atual<=-3){tensao=0;} else {
112   tensao=(2-abs(output_atual)+1)*11.02;}
113   }
114
115 analogWrite(9,tensao); // ESCREVE A TENSAO CALCULADA PARA O VALOR PWM DO ARDUINO
116
117   output_atual=tensao*100/255; // VALOR QUE SERA PLOTADO FAIXA 0 - 100%
118
119
120    if (count<=2){
121    count=count+1;}
122   }
123   //INTERVALO QUE ATUALIZA VALORES COMUNICACAO SERIAL
124  unsigned long currentMillis2=millis();
125   if (currentMillis2-previousMillis2>interval2){
126     previousMillis2=currentMillis2;
127
128     Serial.print(temp_atual); // TEMPERATURA
129     Serial.print("\n");
130
131     Serial.print(output_atual); // OUTPUT
132     Serial.print("\n");
133
134     Serial.print(Setpoint);
135    Serial.print("\n");
136
137    Serial.print(button);
138        Serial.print("\n");
139
140
141    Serial.print(output_atual); //DUTY CYCLE
142    Serial.print("\n");
143    }
144 }
```

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; %Máximo valor y
yMin = 0; %Minimo valor y
plotGrid = 'on';
min = 10; % valor y-min
max = 60; % valor y-max
delay = 0; % tempo de atualizacao do gráfico

%% DECLARACAO DE VARIAVEIS
time = 3000;
BOTAO=0;
tensao_saida_lm35 = 0;
temperatura_saida=0;
setpoint=1; %SETPOINT
output_saida_pid=0;
 count = 0;
%CONFIGURACAO DO PLOT
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' );
tic
%LOOP PLOTAR GRAFICO ONLINE
while ishandle(plotGraph_temperatura_saida) && BOTAO==0
   %RECEBE VALOR DAS VARIAVEIS 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);

          subplot(2,1,1);
          title('Controle de Temperatura','FontSize',15);
          xlabel(xLabel,'FontSize',15);
```

```
76            ylabel(yLabel,'FontSize',15);
77            set(plotGraph_temperatura_saida,'XData',time,'YData',temperatura_saida); % temperatura
78
79            set(plotGraph_setpoint,'XData',time,'YData',setpoint_saida); %temperatura
80            legend('Location','northeastoutside')
81            %CRIA ANOTACAO DE TEMPERATURA NO GRAFICO
82    delete(findall(gcf,'type','annotation'))
83    dim = [.83 .55 .3 .3];
84    str = sprintf('Temperatura: %0.2f C',temperatura_saida(count));
85    annotation('textbox',dim,'String',str,'FitBoxToText','on');
86
87            axis([yMin yMax min max]);
88            grid(plotGrid);
89            axis([time(count)-50 time(count) min max]);
90            subplot(2,1,2);
91            title('Tenso PWM de Controle','FontSize',15);
92            xlabel(xLabel,'FontSize',15);
93            ylabel('Tenso (0-100%)','FontSize',15);
94            set(plotGraph_output_saida_pid,'XData',time,'YData',output_saida_pid);
95            legend('Location','northeastoutside')
96    %ANOTACAO DO DUTY CYCLE NO GRAFICO
97    dim1 = [.83 .10 .3 .3];
98    str1 = sprintf('Duty Cycle: %0.2f %%',output_saida_pid(count));
99    annotation('textbox',dim1,'String',str1,'FitBoxToText','on');
100   grid(plotGrid);
101   axis([time(count)-50 time(count) -1 101]);
102
103             %ATUALIZA O GRAFICO
104             pause(delay);
105       if (time(count)>1800)
106           BOTAO=1;
107           end
108   end
109   %CRIA FIGURA COM TODOS OS DADOS 0-1800s
110   figure('units','normalized','outerposition',[0 0 1 1]);
111   subplot(2,1,1);
112   x=time;
113   y=temperatura_saida;
114   y1=setpoint_saida;
115   plot(x,y, x,y1);
116   title('Controle de Temperatura','FontSize',15);
117   xlabel(xLabel,'FontSize',15);
118   ylabel(yLabel,'FontSize',15);
119   legend({'Temperatura C', 'Referência'},'Location','northeastoutside')
120   axis([yMin yMax min max]);
121   grid(plotGrid);
122   axis([0 time(count) min max]);
123
124   subplot(2,1,2);
125   x1=time;
126   y2=output_saida_pid;
127   plot(x1,y2);
128   title('Tenso PWM de Controle','FontSize',15);
129   xlabel(xLabel,'FontSize',15);
130   ylabel('Tenso (0-100%)','FontSize',15);
131   legend({'Tensao PWM'},'Location','northeastoutside');
132   grid(plotGrid);
133   axis([0 time(count) -1 101]);
134   %FECHA A COMUNICACAO SERIAL
135   disp('Plot Closed and arduino object has been deleted');
136   fclose(s)
137   delete(s)
138   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

Calor Específico

A Figura B.1 mostra o calor específico de diversos tipos de materiais, o alumínio fica em terceiro lugar da lista.

Figura B.1 - Calor específico - comparação de materiais.

substância	calor específico (cal/g°C)
água	1,0
álcool	0,6
alumínio	0,22
ar	0,24
carbono	0,12
chumbo	0,031
cobre	0,091
ferro	0,11
gelo	0,5
hélio	1,25
hidrogênio	3,4
latão	0,092
madeira	0,42
mercúrio	0,033
nitrogênio	0,25
ouro	0,032
oxigênio	0,22
prata	0,056
rochas	0,21
vidro	0,16

Fonte: <http://fep.if.usp.br/~profis/experimentando/diurno/downloads/Tabela%20de%20Calor%20Especifico%20de%20Varias%20Substancias.pdf>
Acesso: 16 de novembro de 2018.

C

Ziegler-Nichols Malha Aberta

A Figura C.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 C.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