CONTROLE DE TEMPERATURA PID

PID TEMPERATURE CONTROL

SOARES, Derik [1]
DEBRUEM, Renato [2]

RESUMO

Este documento apresenta o controle de temperatura de um sistema de malha fechada, onde um cooler será manipulado através de um PWM, fornecido pelo cálculo de um PID de acordo com a temperatura atual e o setpoint desejado. A temperatura será medida através de um sensor de temperatura (LM35) que estará em contato com um resistor aquecido de 100Ω. Para o controle foi utilizado um MCU ATMEL AT89S2051 com um código em linguagem C embarcado.

Palavras-chave: PID; Controle de Temperatura; AT89S2051.

ABSTRACT

This document presents the temperature control of a closed loop system, where a cooler will be handled through a PWM, provided by the calculation of a PID according to the current temperature and the desired setpoint. The temperature will be measured through a temperature sensor (LM35) that will be in contact with a heated resistor of 100. For the control, an ATMEL AT89S2051 MCU with an embedded C-code was used.

Keywords: PID; Temperature Control; AT89S2051.

[1]Graduando do Curso de Engenharia de Computação da Universidade Potiguar (UNP) de Natal-RN, wsb_walt_d@hotmail.com;
[21]Graduando do Curso de Engenharia de Computação da Universidade Potiguar (UNP) de Natal-RN, rdebruem@gmail.com;

INTRODUÇÃO

O controle de processo surgiu com a necessidade de se obter melhores desempenhos de equipamentos e sistemas industriais. Consiste na técnica de manter as variáveis de um processo em valores pré-determinados, conhecidos como set points, a partir de algoritmos relacionados às variáveis que são fornecidas pelos sensores do processo em um controle de malha fechada.

Este projeto tem como objetivo realizar um controle da temperatura em um resistor, utilizando um controle proporcional, integral, derivativo (PID) implementado em um controlador AT89S2051 ATMEL.

MATERIAIS E MÉTODOS

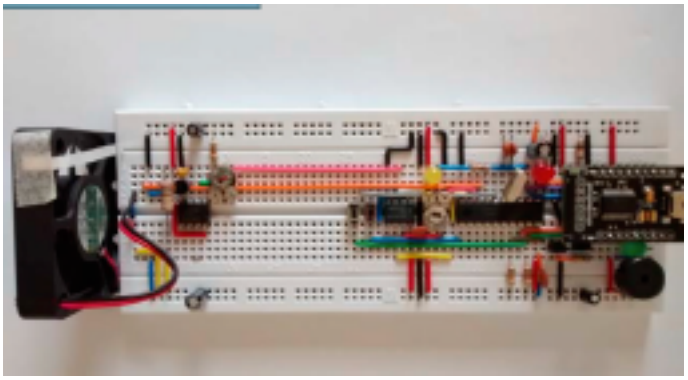

Fig. 1. Protótipo montado em um Protoboard.

Componentes Principais

- Protoboard
- AT89S2051 ATMEL
- Amp Op LM358
- LM35
- Mini Cooler de 5V

Outros Componentes

- Transistor BC337 NPN 45V 0,8A 100MHz
- Diodo 1N914 de chaveamento e baixo sinal
- Trimpot 5K mini
- Capacitor cerâmico de 100nF
- Capacitor eletrolítico de 1uf
- Capacitor eletrolítico de 10uF
- Resistor 1k 0,25W
- Resistor 100 0,25W (p/ 5Vcc)
- Pasta térmica

Características dos Componentes

- LM35 (SENSOR)
    - Faixa de temperatura -40ºC a 110ºC
    - SINAL DE SAIDA
        - Linear 10mV/o C, Offset 0o C = 0mV
        - Impedância de Saída = 0,1 Ohm a 1mA

- PRECISÃO GARANTIDA
  - 0,25ºC de 0 a 100ºC, 0,50ºC no restante.

Fig. 2. LM35.

- AMP OP LM358 (AMPLIFICADOR)
  - Amplificar o sinal para que:
    - Maior valor do sinal = maior valor analógico aceito (100o C => 255)
    - Ganho do amplificador = 5V/1V = 5.
  - CI com duplo Amp Op
    - Invólucro de 8 pinos
    - Semelhante ao LM324 (quadruplo)
    - Funciona com Vcc de 5 a 32V
  - Excursão da saída de 0 a aproximadamente Vcc – 1,6V

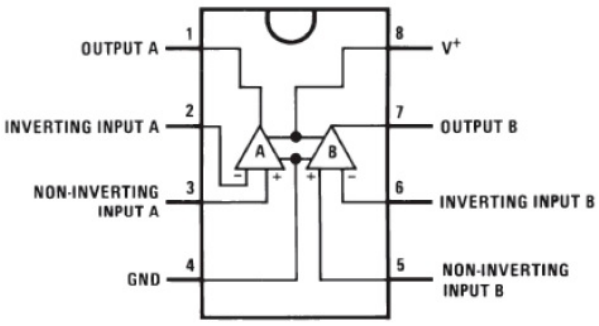

Fig. 3. Amplificador Operacional.

- COOLER (ATUADOR)
  - VCC: 5V
  - Corrente: DC

DESENVOLVIMENTO

Foi desenvolvido um kit em uma protoboard com o objetivo de controlar a temperatura de um resistor (100 ohms) através de sensor (LM35) em contato com o resistor. Os valores lidos serão enviados para o AT89S2051 ATMEL onde o controle PID está implementado. O PID controla o PWM de um cooler de forma a estabilizar a temperatura do resistor conforme o set point preestabelecido pelo usuário. O set point pode ser estabelecido através de dois botões físicos presentes na protoboard. Os valores podem ser vistos através de uma tela LCD. A tensão de alimentação do kit é de 5V a uma temperatura ambiente de 23 a 30 graus Celsius.

A. CONTROLADOR

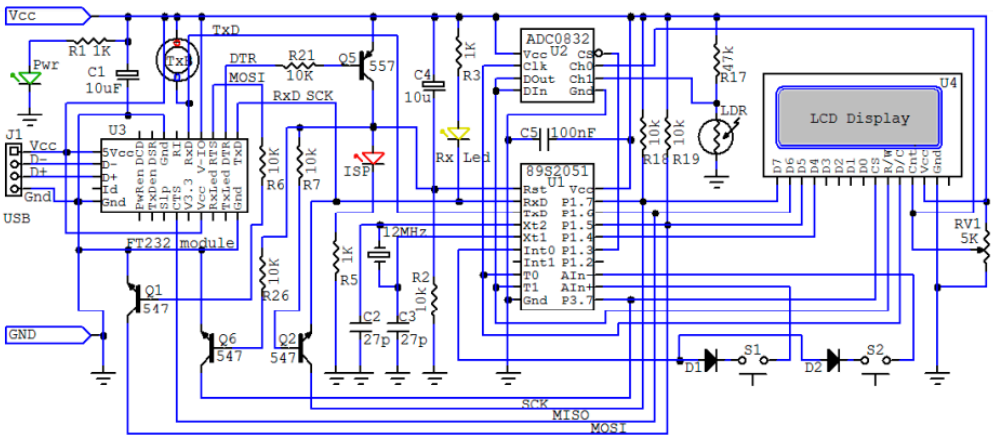

Fig. 4. Diagrama de Blocos do Controlador.

B. CONTROLE DE ARREFECIMENTO

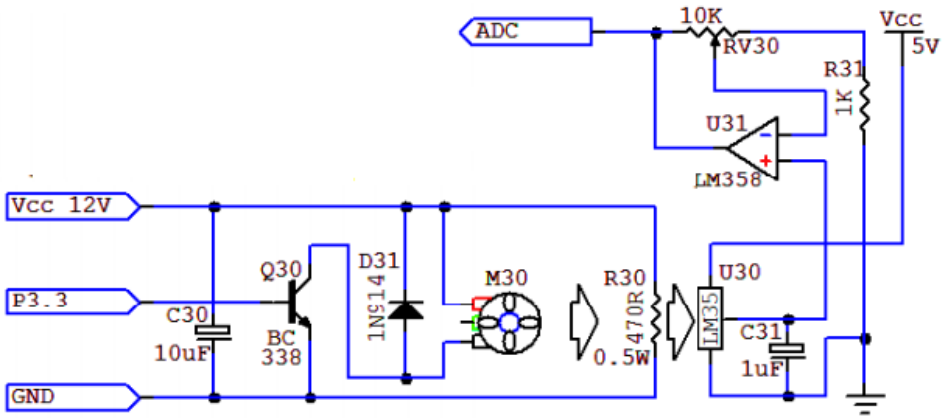

Fig. 5. Diagrama do Controle de Arrefecimento.

C. DIAGRAMA GERAL DO CIRCUITO

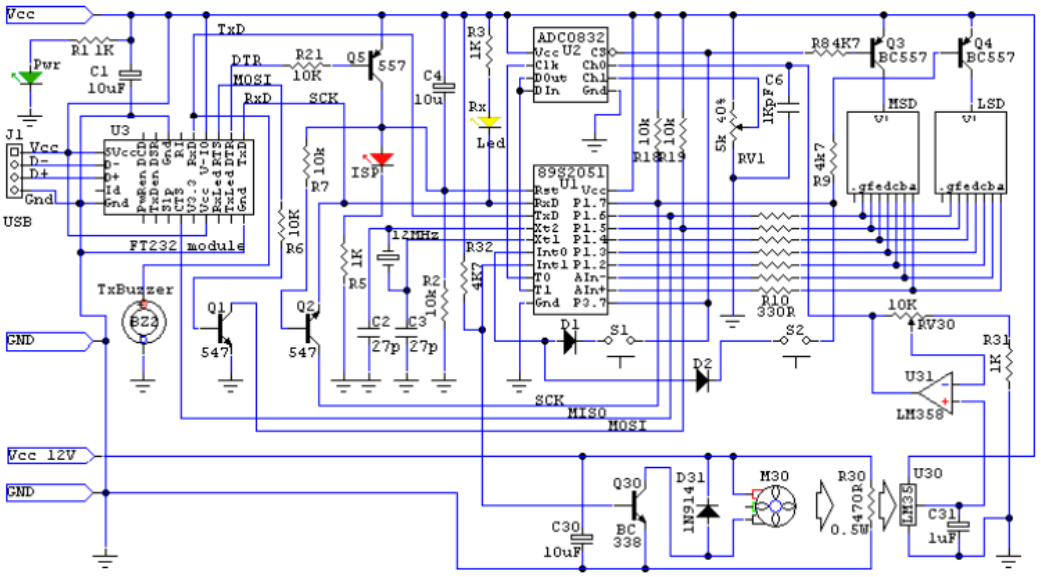

Fig. 6. Diagrama Geral do Circuito.

D. PROGRAMAÇÃO DO CONTROLADOR

```
#include <uRTOS_2.21.h>
#include <convAD08.h>
#include <serialInt_1.2.h>
char __code Tit1[13]= "p i d ";
char __code Tit2[13]= "Setp Temp";
unsigned char pid[3]=0,0,0;
unsigned char pwm=0;
void RTproc_1(){
    if(cnt200==pwm) P3_3=1; }
void RTproc_2(){
    P3_3=0; }
void ajustPID(){
    unsigned char Kx=0;
    __bit ajust=1;
    pwm=0;
    while(ajust){
        if(S1==3){
            S1=0;
            pid[Kx]++;
        }
        if(S2==3){
            S2=0;
            pid[Kx]-;
        }
        if(Kx==0) char2LCD(0x81, pid[0], 2);
        if(Kx==1) char2LCD(0x86, pid[1], 2);
        if(Kx==2) char2LCD(0x8B, pid[2], 2);
        umSeg=0; cnt100=255;
        while(S1==2);
        if(umSeg){S1=0;
            umSeg=0; Kx++;
            if(Kx==3) Kx=0;
        }
        umSeg=0; cnt100=255;
        while(S2==2);
        if(umSeg){S2=0;
            umSeg=0; ajust=0;
        }
    }
}
void main (){
    unsigned char setup=70;
    unsigned char tempC=0;
    signed char errAt=0;
    signed char erAnt=0;
    signed int ctrlAt=0;
    signed int ctlAnt=0;
    signed int P=0, I=0, D=0;
    unsigned int tempo=0;
    unsigned char i=0;
    inic();
    P3_3=0;
    Ch01=0;
    wrLCD4(comand,0x80);
    for(i = 0; i < 12 ; i ++){
        wrLCD4(letra, Tit1[i]);
    }
    wrLCD4(comand,0xC0);
    for(i=0; i<12; i++){
        wrLCD4(letra, Tit2[i]);
    }
    ajustPID();
    while(1){
        if(S1==3){
            S1=0; setup++;
        }
        if(S2==3){
            S2=0; setup-;
        }
        if((S1==2)&&(S2==2)){
            S1=0; S2=0; tempo=0;
            ajustPID();
        }
        char2LCD(0xC4, setup, 3);
        char2LCD(0xCC, tempC, 3);
        char2Ser(0,errAt,3);
        int2Ser(2,ctrlAt,5);
        char2Ser(2,pwm,3);
```

```
79          int2Ser(2, tempo,5);
80          while(!TxFlag); TxFlag=0;
81          SBUF=13;
82          Ch01=0;
83          tempC=convAD08();
84          erAnt=errAt;
85          errAt=tempC-setup;
86          P=errAt; I=erAnt; D=errAt-erAnt;
87          P=(P*pid[0])/10; I=(I*pid[1])/10; D=(D*pid[2])/10;
88          ctlAnt=ctrlAt;
89          ctrlAt=ctlAnt+P+I+D;
90          if(ctrlAt>255) ctrlAt=255;
91          if(ctrlAt<0) ctrlAt=0;
92          pwm=ctrlAt*199/255;
93          while(!umSeg); umSeg=0; tempo++;
94      }
95  }
96
```

E. SOFTWARE DE GERENCIAMENTO

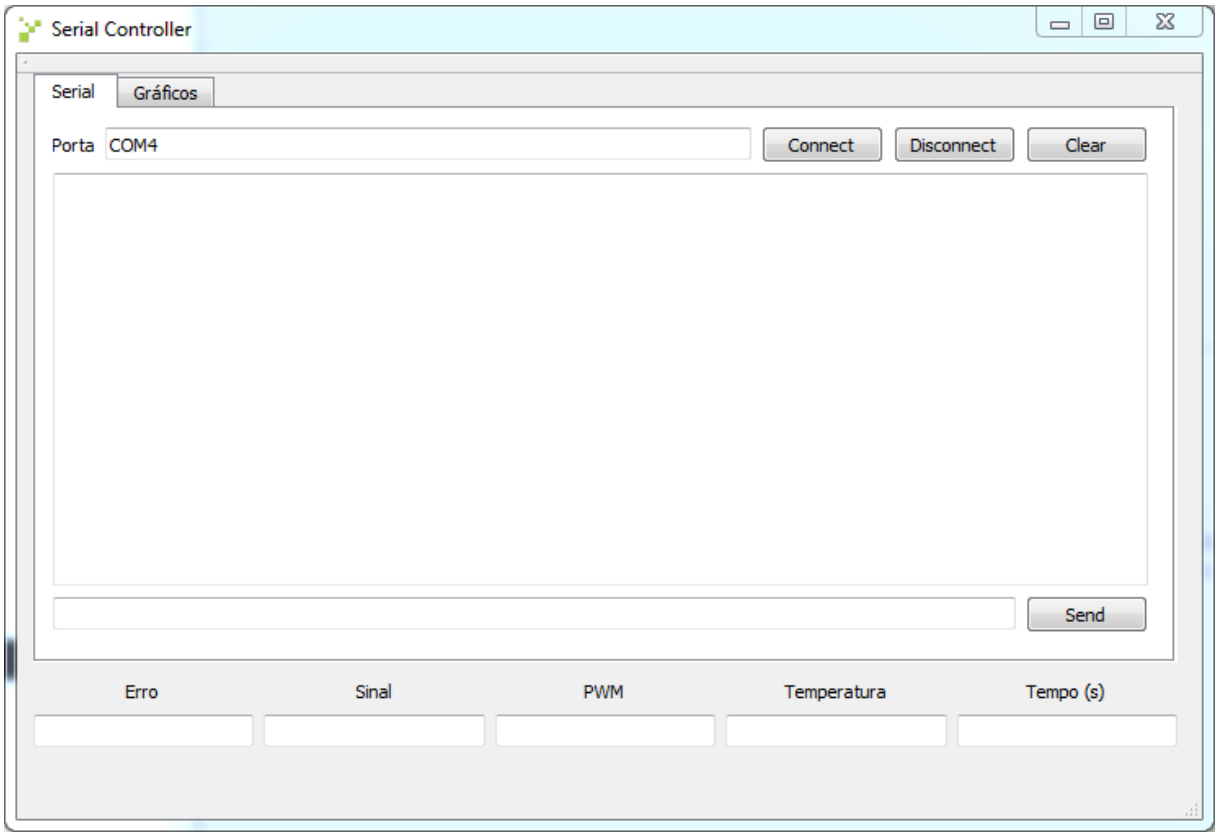

Fig. 7. Interface de comunicação com o sistema.

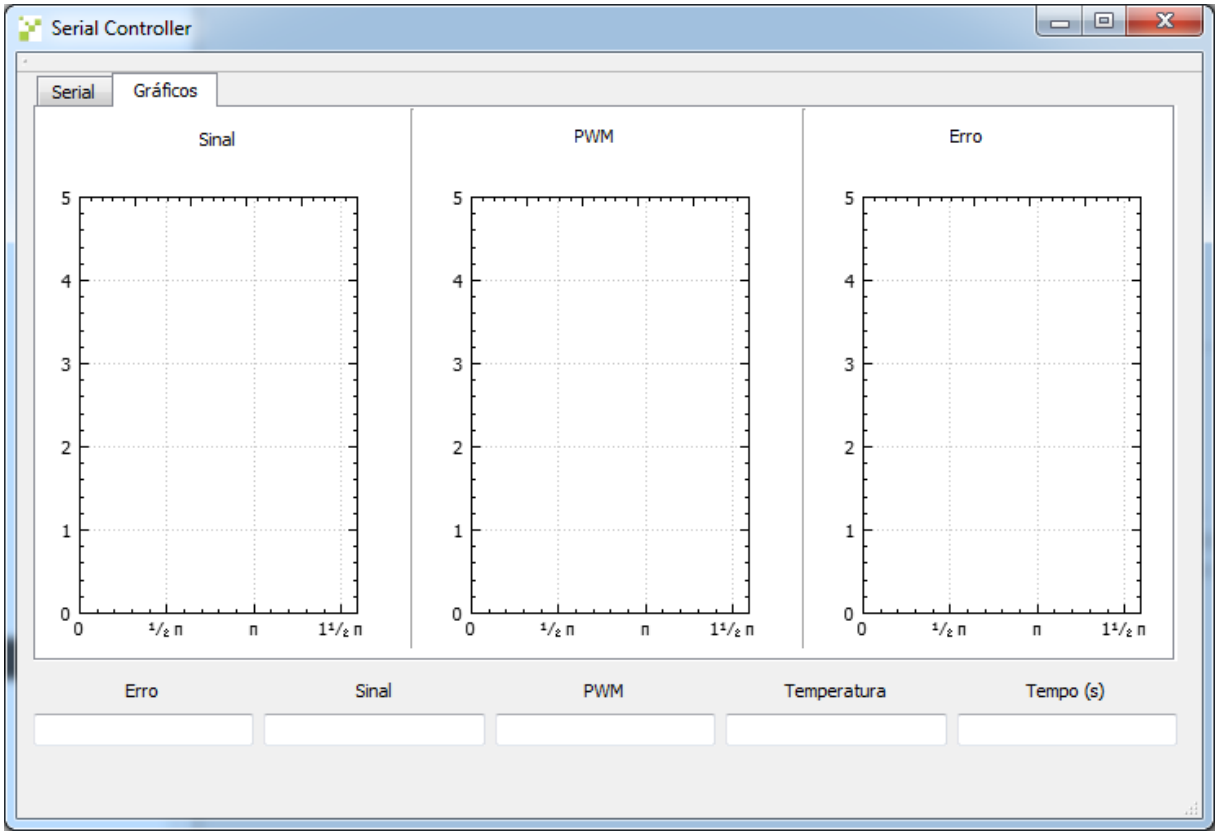

Fig. 8. Interface de Saída do Sistema com gráficos do Sinal, PWM e Erro.

FUNCIONALIDADE

Ao ligar o sistema, é mostrado no LCD a mensagem para selecionarmos os valores de Kp, Ki e Kd. Para selecionar, basta pressionar os botões presentes na protoboard (S1 ou S2) para selecionar os valores para mais ou para menos, respectivamente.

Inicialmente, o primeiro valor solicitado é do Kp. Após selecionar o primeiro valor, pressiona-se o botão S2 e o segura por 2 segundos para mudar de opção e poder selecionar o segundo valor, no caso Ki. Após selecionar o segundo valor, pressionar novamente o botão S2 por 2 segundos para conseguir selecionar o valor de Kd.

Quando os valores tiverem sido ajustados, deve-se pressionar o botão S1 e o segurar por 2 segundos para então dar início ao controle e habilitar a opção de ajustar o setpoint para o valor desejado.

Nesta demonstração, selecionamos os seguintes valores para Kp = 0.8, Ki = 3.0 e Kd = 1.5, como mostrado na figura 9.

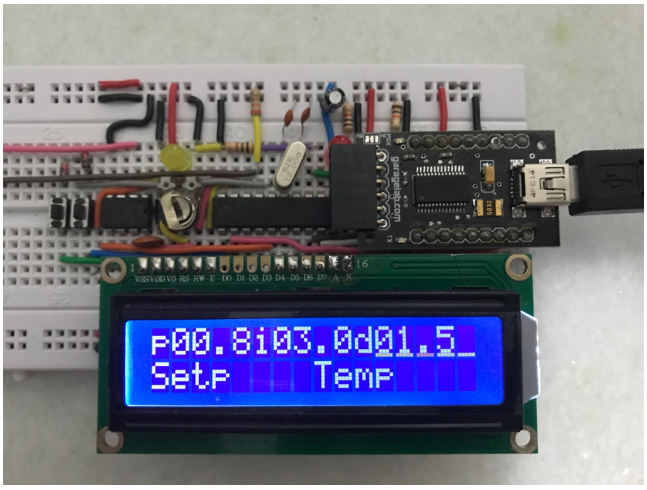

Fig. 9. Display LCD com os valores de Kp, Ki e Kd ajustados.

Após o ajuste dos valores, pressionamos o botão S1 por 2 segundos e damos início ao Controle. No display LCD agora será mostrado um valor predefinido para o SETPOINT e ao lado direito o valor atual da TEMPERATURA medida pelo sensor LM35.

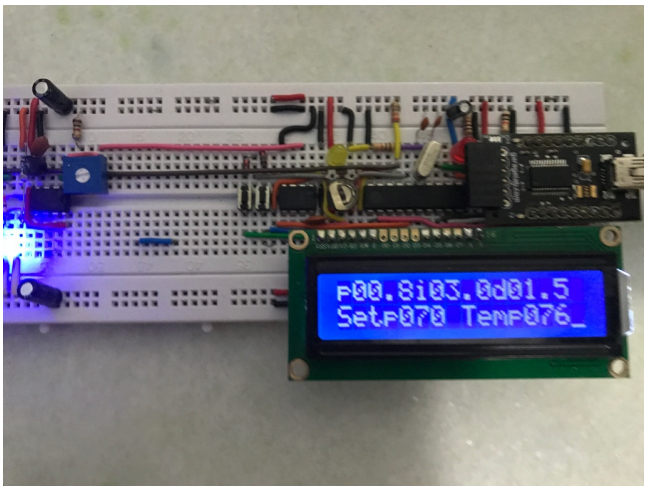

Fig. 10. Display LCD com os valores do setpoint e temperatura sendo mostrados

Nesse momento, o controlador já está efetuando o cálculo do PID e enviando pela Comunicação SERIAL as informações de Temperatura atual, PWM fornecido, Erro calculado, Sinal de Saída do PID e Tempo decorrido, onde são recebidos pelo Software de Gerenciamento e gera os gráficos da saída.

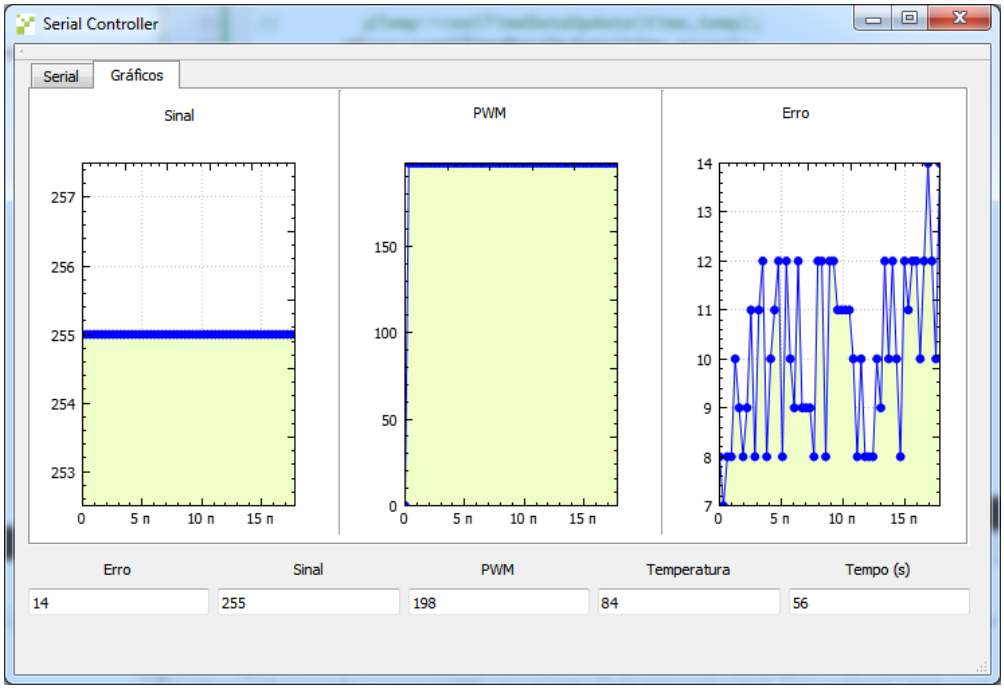

Fig. 11. Gráfico de saída do sistema em T=56s

Com o decorrer do tempo, o gráfico do erro vai se estabilizando com os valores próximos a 0, onde significa que a temperatura estabilizou no SETPOINT desejado.

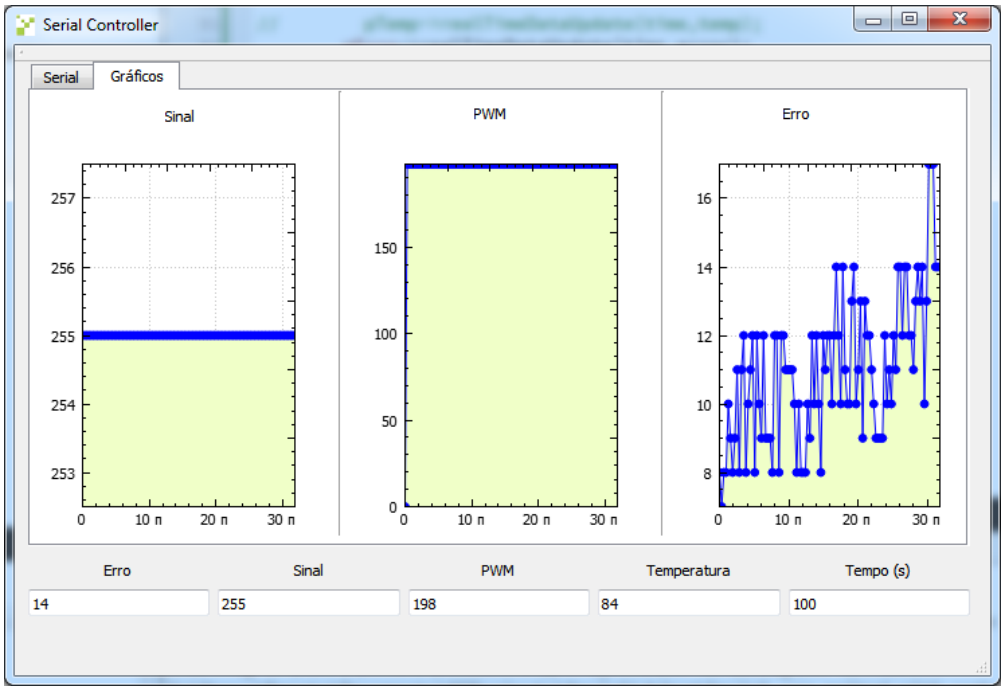

Fig. 12. Gráfico de saída do sistema em T=100s.

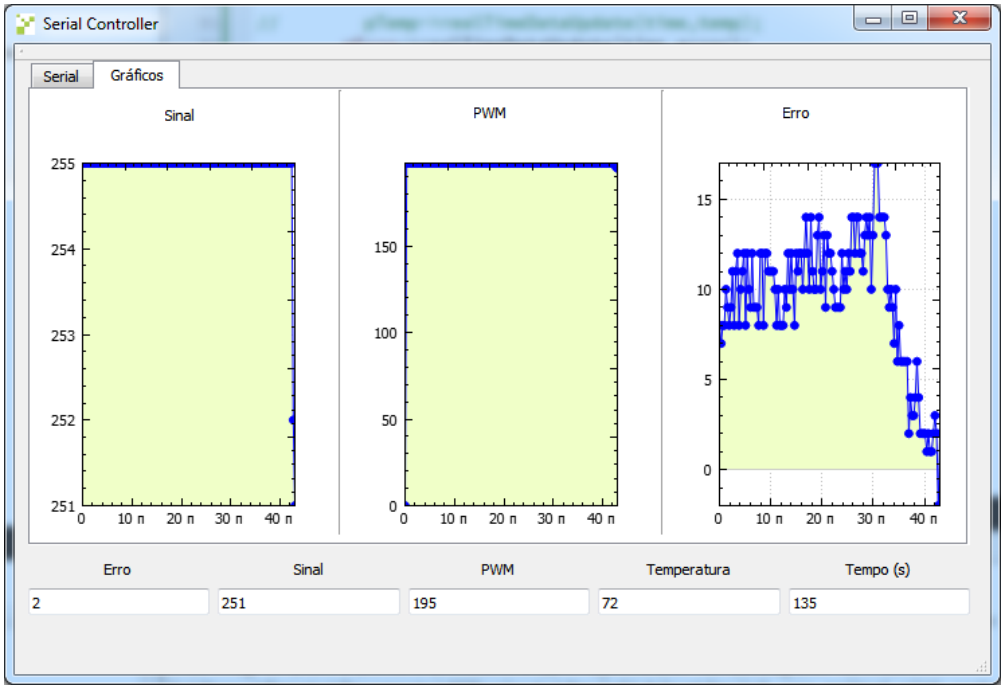

Fig. 13. Gráfico de saída do sistema em T=135s.

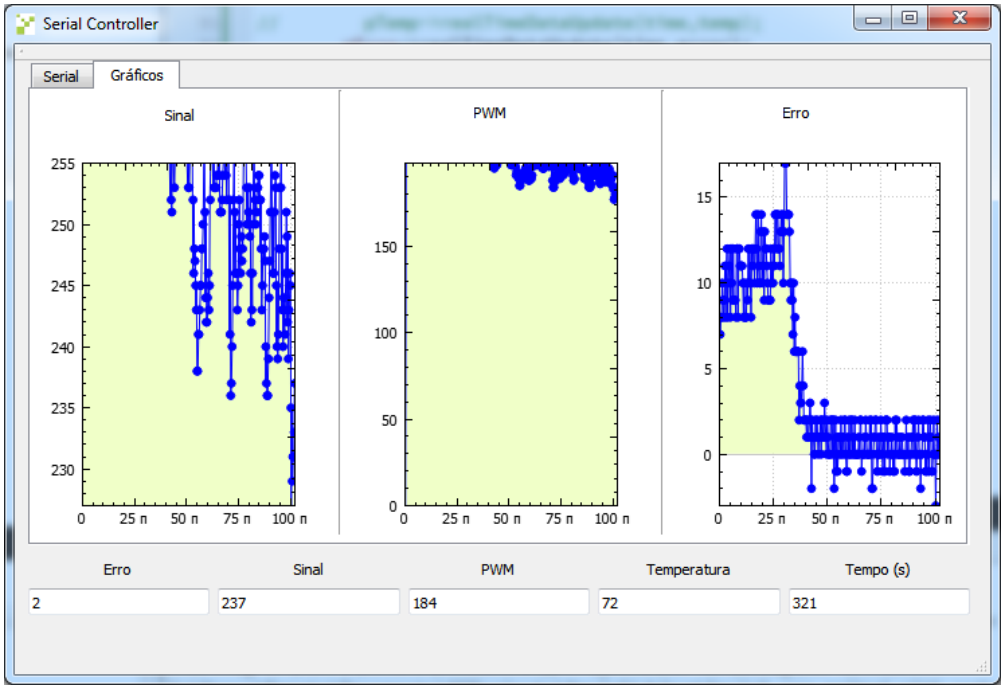

Fig. 14. Gráfico de saída do sistema em T=321s.

Na figura 14, pode-se observar que o Erro tende ao valor de 0, tentando estabilizar a temperatura no valor do SETPOINT desejado.

CONCLUSÃO

Este trabalho foi realizado com o objetivo de reunir os conhecimentos adquiridos durante o curso de Engenharia de Computação para desenvolver um controlador PID, implementado digitalmente em um microcontrolador da família ATMEL, para controlar a temperatura em um resistor ohmico através de um Cooler.

Foi possível constatar que a modelagem do sistema é importante para a calibração dos parâmetros do controlador e que os controladores PID apesar de simples, são largamente utilizados na indústria.

Os resultados experimentais comprovaram que o controlador cumpriu com o objetivo para o qual fora projetado, estabilizando a temperatura no SETPOINT desejado.

REFERÊNCIAS

[1] Åström, K. J.; Hägglund, T., PID Controllers: Theory, Design, and Tuning. 2a Ed. Carolina do Norte: Instrument Society of America, 1995.

[2] Martins, N. A., Sistemas Microcontrolados. 1a Ed. Novatec, 2005.

[3] Stallings, W., Arquitetura e Organização de Computadores. 5a Ed. Pearson Addison Wesley, 2005.

[4] Ogata, K., Engenharia de Controle Moderno. 4a Ed. São Paulo: Prentice Hall, 2003.

[5] Omron Electronics LLC, Temperature and Process Control Instrumentation. Catálogo H301-E3-1, 2004, pp. E-1 – E-23.

[6] Datasheet LM35. Disponível em <http://www.national.com>. Acesso em 10 de Maio de 2017.

[7] Datasheet AT89S2051. Disponível em <http://www.atmel.com>. Acesso em 10 de Maio de 2017.