IMPLEMENTAÇÃO DE SISTEMAS DE CONTROLE COM OBJETIVOS DIDÁTICOS

Ricardo Bortoluzzi Morim, Hilton A. Grundling, Humberto Pinheiro
Grupo de Eletrônica de Potência e Controle - GEPOC, Universidade Federal de Santa Maria - UFSM, Brazil
eletroinf@gmail.com, ghilton03@gmail.com, humberto.ctlab.ufsm.br@gmail.com

***Resumo*—Com o intuito de criar opções didáticas de introdução à prática de sistemas de controle, neste artigo é apresentada a implementação de dois sistemas de controle aplicados a um processo térmico. São mostrados os passos de modelagem do circuito proposto, projeto dos controladores (PI e Adaptativo), simulação e implementação digital utilizando uma placa Arduino Uno. Conquanto o sistema proposto seja muito simples, verificou-se que é uma alternativa versátil, segura e de baixo custo para a iniciação prática, podendo ser utilizado para outros tipos de controladores, além dos aqui apresentados. Pode-se concluir que a sua utilização possui grande potencial no aprendizado de sistemas de controle.**

***Keywords* – Controle Adaptativo, Arduino, RMRAC, PID, Modelagem.**

I. INTRODUÇÃO

A disciplina de controle de sistemas faz parte da grade de conteúdos de diversos cursos de engenharia. O aprendizado e domínio deste conteúdo passa pelo estudo das teorias clássicas de controle, englobando noções de estabilidade e técnicas de controle básicas, como os controladores com ações proporcional, integral e derivativo (PID) [1]. A realização de experiências práticas geralmente não é uma prioridade, restando o conteúdo bastante teórico.

Em [2] é proposto o questionamento se o professor ensina ou pesquisa. A partir do questionamento, é construída a ideia de que o professor, ao pretender ensinar, precisa dominar não só a técnica, mas o processo pelo qual se constrói o conhecimento. Para isso é necessário descobrir o que o aluno pensa e como pensa, visando entender a sua capacidade cognitiva naquele momento. Com este conhecimento, é possível de se propor desafios coerentes ao seu estágio de aprendizado.

Todavia, a realização de práticas tende a aprofundar e a fixar melhor o conhecimento. O estudo das disciplinas de controle emprega frequentemente *softwares* de simulação, o que é muito importante e pode ser complementado. Porém, há opções simples e de baixo custo para introduzir o aluno a um primeiro contato com as realizações de bancada. Nesse contexto, é interessante que as primeiras realizações utilizem meios seguros e sem riscos, portanto tensões e potências baixas são desejáveis. Outrossim, circuitos simples representam uma alternativa natural de introduzir o estudante à prática, porquanto este trabalho explora a utilização de um processo térmico como uma alternativa para o início da prática de controladores.

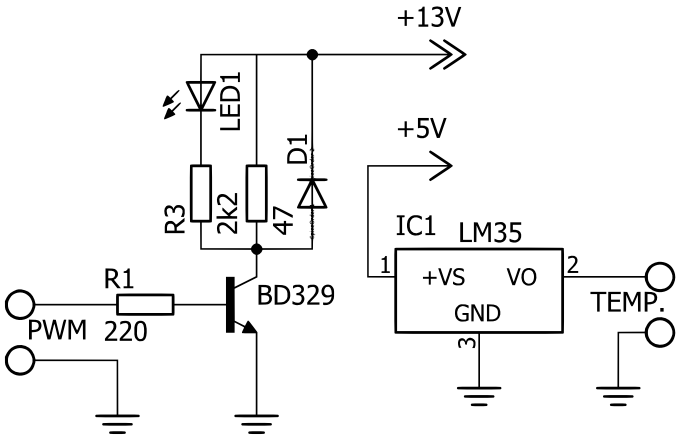

Figura 1. Circuito proposto.

Este artigo está organizado da seguinte maneira. Na seção II é apresentado o sistema proposto. Na seção III é mostrado o processo de modelagem da planta. Na seção IV é realizado o projeto e simulação dos controladores enquanto a seção V contém os detalhes da implementação e resultados obtidos. Na seção VI são discutidos os resultados e feitas algumas sugestões. Finalmente, o trabalho é encerrado com a conclusão.

II. O SISTEMA PROPOSTO

Com o objetivo de propor um sistema simples, de fácil utilização e seguro, algumas restrições foram consideradas. As peças e partes deveriam ser de baixo custo; os ambientes de desenvolvimento deveriam ser disponíveis gratuitamente e com documentação minimamente adequada. Com estas restrições, a plataforma Arduino [3] mostra-se adequada, uma vez que é *open source* e amplamente documentada. Para a planta, foi definido um processo térmico, com a utilização de um resistor de aquecimento de 47 Ω (capacidade de dissipação de 10 W), acionado por meio de um transistor de junção bipolar (TJB). Para sensor de temperatura, foi escolhido o LM35, que é amplamente documentado e conhecido. Para a alimentação elétrica do circuito foi definida uma fonte de Corrente Contínua (CC) na faixa de 12 a 15 V (neste artigo foi utilizado 13 V) com capacidade de fornecer corrente de até 1 A. O esquema elétrico proposto é mostrado na Figura 1.

Para o acionamento de potência da planta, a escolha pelo TJB é justificada pelo baixo custo e pela possibilidade do TJB poder ser controlado diretamente pelo microcontrolador

da placa Arduino Uno. No caso da utilização de um *Metal-Oxide-Semiconductor Field Effect Transistor* (MOSFET), por exemplo, seria necessário um circuito para adequar o sinal de saída do microcontrolador de forma a compatibilizá-lo com o nível de tensão exigido para o acionamento do Gate. No caso do TJB, basta apenas um resistor limitador de corrente para esta função.

Foi utilizado também um LED indicador, em paralelo com o resistor de potência, para inspeção visual durante o funcionamento. O diodo D1 tem a finalidade de proteger o TJB de sobretensões que podem surgir devido ao comportamento indutivo do resistor de carga. A corrente i_b é definida de acordo com a escolha do TJB, consultando no *datasheet* o seu ganho e considerando a corrente que o mesmo irá acionar. Deve ser levado em conta um fator de segurança, isto é, não escolher i_b próximo do mínimo necessário.

III. Modelagem da planta

O primeiro passo a ser executado para o projeto de um controlador é a modelagem da planta. Dentre as formas possíveis de se obter o modelo, uma delas consiste em aplicar um degrau e gravar os dados da resposta. No caso deste trabalho, o degrau consiste em aplicar uma tensão de valor conhecido sobre o resistor e gravar o histórico da variação de temperatura. Através do uso da plataforma Arduino, esta etapa pode ser realizada facilmente, como descrito a seguir.

A. Configuração do Arduino

Foi utilizada uma placa Arduino Uno [3], que contém um microcontrolador de 8 bit e o suporte básico de *hardware*, como circuitos de *clock*, reguladores de tensão e circuitos de comunicação serial. Apesar do microcontrolador do Arduino não possuir suporte a operações matemáticas utilizando ponto flutuante em *hardware*, o compilador é capaz de prover, de forma emulada, estas operações. O custo disso é a exigência de um tempo maior e também a utilização de mais memória do que seria necessário em um microcontrolador com suporte a ponto flutuante. Este fato limita a utilização do Arduino em relação ao quesito desempenho, mas ainda assim ele é um sistema viável para diversas aplicações.

Para realizar o degrau de tensão e medir a resposta, em temperatura, é necessário que se configure o Conversor Analógico-Digital (ADC, do inglês *Analog Digital Converter*) para a leitura de temperatura e uma saída como modulação por largura de pulso (PWM, do inglês, *Pulse Width Modulation*) para o acionamento da planta. Além disso, foi configurada a porta serial para a transmissão dos dados para o computador.

O ADC do Arduino possui uma resolução de 10 bit, isto é, enquanto o sinal na entrada analógica varia entre zero e a tensão de referência (*Vref*) o valor numérico resultante da conversão varia de zero a $2^{10} - 1$. Para esta aplicação, foi utilizada a tensão de referência interna do microcontrolador, que é de 1,1 V. Isto limita a faixa de tensão admitida na entrada do ADC, porém está dentro dos valores necessários para esta implementação.

O módulo PWM do Arduino possui uma resolução de 8 bit (2^8) e frequência pré-programada para aproximadamente 1 kHz. Através da definição do ciclo ativo do PWM se estabelece a tensão média na saída. Para ajustar corretamente o ciclo ativo do módulo PWM é preciso conhecer a relação do peso de cada *step* do ajuste para com a tensão de alimentação da planta. Como foi utilizada uma fonte de 13 V, com o ciclo ativo em 100 % se obtém 13 V sobre a planta, enquanto que com o ciclo ativo de 0 % a tensão é de 0 V. Assim, como o número de *steps* é conhecido, cada variação unitária no ajuste do ciclo ativo resulta em:

$$\text{Res}_{PWM} = \frac{13 - 0(V)}{2^8 - 1} \approx 0,0051V \tag{1}$$

que representa a resolução máxima do módulo PWM.

Para o arquivamento dos dados coletados nos experimentos utilizando a placa Arduino, uma boa alternativa é a utilização da porta serial. Através desta, os valores podem ser transferidos para o computador de forma *online*, isto é, concomitante a realização experimental. As configurações necessárias são simples, resumindo-se a escolha da taxa de transferência, que foi ajustada para 115200 bits por segundo (bps).

B. Resposta ao degrau

Para obtenção do modelo, o procedimento utilizado será a realização de um degrau de tensão na planta e a medição da resposta em temperatura, gerando um gráfico em função do tempo [1]. Em outras palavras, o procedimento será o seguinte:

- Estabelecer uma tensão constante sobre o resistor através da definição de um valor fixo para a razão cíclica do PWM.
- Realizar as medidas de temperatura (amostragem) em intervalos de tempo (T_s) constantes.

Estas informações devem ser salvas em forma de tabela para a próxima etapa, que é gerar um gráfico com a curva de resposta de temperatura em função da tensão aplicada a planta. Este gráfico será o meio através do qual serão extraídos os dados da planta, nomeadamente o ganho k, a constante de tempo τ e o atraso de transporte, uma vez que foi considerada como sendo de primeira ordem.

Após a realização do experimento, com um tempo de amostragem $T_s = 1s$, a resposta de temperatura para um degrau de tensão com PWM de razão cíclica $D = 150/255$ é apresentada na Figura 2.

C. Determinação do modelo da planta

Para a determinação do modelo da planta, analisando a Figura 2 se tem as seguintes informações:

- Tempo de atraso: 5 s (a planta foi acionada em 1 s e a temperatura começou a variar em 6 s)
- Constante de tempo: 189 s, tempo em que a resposta do sistema atinge 63 % do valor em regime
- Ganho da planta (k): Dado pela variação de temperatura ($^\circ$C) em função da tensão (em V) aplicada. Neste caso, a temperatura variou de 25,9 $^\circ$C para 72,3 $^\circ$C enquanto

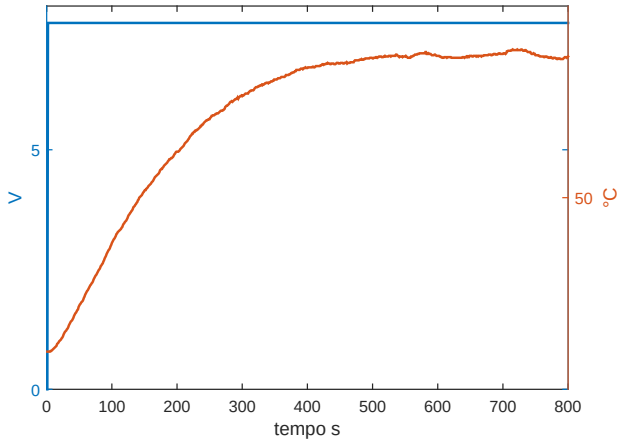

Figura 2. Resposta ao degrau da planta.

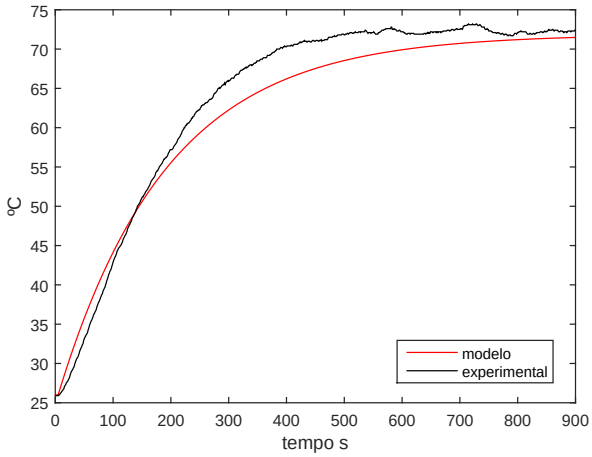

Figura 3. Resposta experimental e do modelo obtido.

a tensão foi modificada de 0 V para $(150/255)13 = 7,65V$.

Disto, se tem que:

$$k = \frac{72,3 - 25,9(°C)}{(150/255)13(V)} \approx 6°C/V. \quad (2)$$

Com os dados obtidos, considerando o modelo simplificado de primeira ordem, se tem a seguinte função de transferência, que relaciona a temperatura y(s) com a tensão de entrada u(s):

$$G(s) = \frac{y(s)}{u(s)} = \frac{6}{189s + 1}e^{-5s}. \quad (3)$$

Pela equação pode ser inferido que se trata de um sistema de primeira ordem, estável, com um atraso de transporte. O comparativo do modelo obtido com o resultado experimental da aplicação de um degrau de tensão é apresentado na Figura 3. Através desta Figura pode ser observado que o modelo representa a planta de forma aproximada. De fato, a simplificação assumida na representação da planta através de um modelo de primeira ordem não captura todas as dinâmicas presentes no sistema. Todavia esta simplificação apresenta uma equivalência aceitável para o que é proposto neste trabalho.

Em virtude do modelo da planta estar definido, a etapa que segue apresenta o projeto e simulação dos controladores propostos para a mesma.

IV. Projeto e Simulação dos Controladores

Uma vez definido o modelo da planta, o procedimento subsequente é o projeto do controlador e a sua simulação. Após esta etapa de validação, procede-se a sua implementação. Na primeira subseção será descrita a utilização de um controlador PI e na segunda um controlador RMRAC. Em se tratando da simulação, algumas considerações são necessárias. Em primeiro lugar, os limites de atuação do sistema de potência são estabelecidos pela tensão de alimentação, de forma que pode ocorrer saturação da saída da lei de controle. Este fenômeno pode ser observado em simulação impondo limites para esta variável e verificando as consequências. Outros

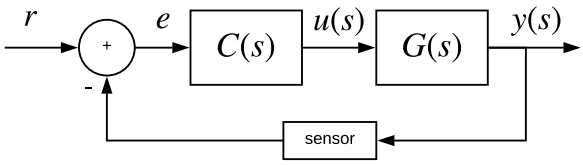

Figura 4. Sistema em MF.

fatores que apresentam diferenças quando da implementação são as simplificações assumidas em termos de modelagem. Na prática, o sensor de temperatura possui uma dinâmica, pois a sua resposta não é imediata frente a variação de temperatura. Além disso, as leituras serão corrompidas por ruído. O diagrama em blocos da aplicação com controlador e o sistema em MF é apresentado na Figura 4.

A. Controlador PI

Para o projeto do controlador PI foi estabelecido o desempenho esperado para o sistema quando em Malha Fechada (MF) através de uma Função de Transferência (FT) de 2º ordem, do tipo:

$$G_m(s) = \frac{2\zeta\omega_n s + \omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2}. \quad (4)$$

Através desta função se estabelece o desempenho em MF desejado pela seleção de seus parâmetros. A próxima etapa consiste em igualar esta função com a função de transferência em MF do sistema mostrado na Figura 4. Sabendo que a função de transferência do controlador PI é dada por:

$$C(s) = \frac{u(s)}{e(s)} = k_p\left(1 + \frac{k_i}{s}\right). \quad (5)$$

A realização em MF do sistema da Figura 4 com o controlador PI e a planta identificada, dada pela equação (3), ignorando o atraso de transporte, passa a ser:

$$G_{mf}(s) = \frac{y(s)}{r} = \frac{k_p\left(1 + \frac{k_i}{s}\right)\left(\frac{6}{189s+1}\right)}{1 + k_p\left(1 + \frac{k_i}{s}\right)\left(\frac{6}{189s+1}\right)}. \quad (6)$$

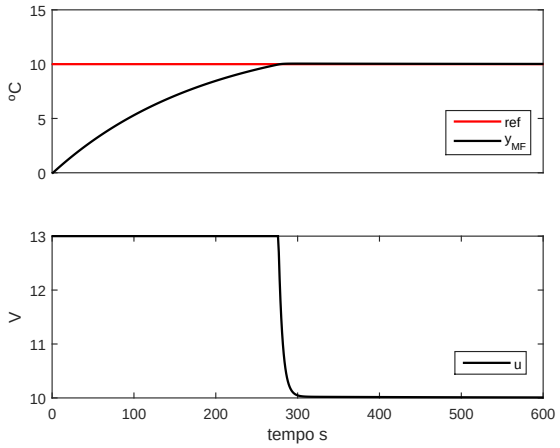

Figura 5. Simulação com controlador PI.

Como a frequência do polo da planta em MA é conhecida através da equação (3), foi estabelecido um modelo com desempenho mais rápido para o sistema em MF baseado na localização do polo em MA. Isto é, através da equação (4) a localização dos polos em MF pode ser fixada em frequências mais altas, e, por consequência, o sistema terá resposta em menor tempo. Assim, para este projeto foi definido $\omega_n = 0,02$. O parâmetro ζ define o comportamento do sistema em relação ao amortecimento. Se o sistema for subamortecido, resultará em *overshoot* na resposta de MF. Logo, o valor escolhido foi $\zeta = 3,5$, objetivando um comportamento sobreamortecido em MF. Com estas definições a equação (4) se torna:

$$G_m(s) = \frac{0,14s + 0,02^2}{s^2 + 0,14s + 0,02^2}. \tag{7}$$

Para encontrar os valores de k_p e k_i do controlador C(s) as equações (7) e (6) são igualadas e os valores calculados. Dessa forma, os valores obtidos são $k_p = 26$ e $k_i = 0,003$. Portanto, o controlador PI é dado por:

$$C(s) = \frac{u(s)}{e(s)} = 26\left(1 + \frac{0,003}{s}\right). \tag{8}$$

A simulação do sistema em MF com este controlador é apresentada na Figura 5. Pode ser verificado por meio desta Figura o comportamento suave da resposta da planta e a ausência de *overshoot*, como esperado.

B. Controlador RMRAC

Para este exemplo, foi escolhido o controlador proposto por [4], que utiliza uma lei de controle do tipo:

$$u = \theta^T \omega + c_0 ref. \tag{9}$$

Onde θ é o vetor de parâmetros, ω é o vetor regressor e ref é a referência. Para a adaptação do vetor de parâmetros foi utilizado um adaptador Gradiente, dado por:

$$\dot{\theta} = -\frac{\Gamma \varepsilon_1 \zeta}{m^2} - \Gamma \sigma \theta \tag{10}$$

onde Γ é a matriz de ganho do adaptador e σ é a função de chaveamento (definida em [4]), cuja principal função é modificar o ganho no caso de um aumento de $\| \theta \|$ além do limite dado por $M0$. O Erro aumentado é dado por:

$$\varepsilon_1 = y - y_m + \theta^T \zeta - v \tag{11}$$

na qual y é a saída da planta e $y_m = Wm(s)[ref]$ é o modelo de referência. São usados ainda os seguintes sinais auxiliares:

$$\zeta = \mathrm{Wm}(s)[I\omega], \quad v = \mathrm{Wm}(s)[\theta^T \omega]. \tag{12}$$

O normalizador m é calculado através de:

$$\dot{m} = -\delta_0 m + \delta_1(|u| + |y| + 1). \tag{13}$$

Procedendo a simulação do controlador RMRAC, os parâmetros foram definidos como:

$$\begin{aligned} &\mathrm{Wm}(s) = \tfrac{0,015}{s+0,015}, \quad \Gamma = 5,0, \quad \delta_0 = 0,5, \\ &\delta_1 = 0,9, \quad M0 = 2,0, \quad \sigma_0 = 0,1, \qquad c_0 = 1. \end{aligned} \tag{14}$$

Neste trabalho o grau relativo da parte modelada da planta foi estabelecido como sendo de primeira ordem. Portanto, os sinais e filtros utilizados no controlador são de primeira ordem, com valores escalares, uma vez que o controlador possui estrutura com grau relativo $2n - 1$ [4]. Para a escolha do modelo de referência deve-se ter em mente que a sua dinâmica tem consequências diretas em relação a resposta dinâmica do controlador. Assim, quanto maior a frequência do seu polo, mais rápidas serão as reações do controlador, o que pode provocar *overshoot* excessivo e até mesmo dificuldade na convergência dos parâmetros. Na Figura 6 é apresentado o resultado de simulação do RMRAC. A atuação da lei de controle foi limitada de acordo com os limites da fonte de tensão, no caso entre 0 e 13 V. Como pode ser visto, os parâmetros convergem para valores estáveis e o erro de rastreio tende a zero ao longo do tempo. Portanto, o controlador projetado está pronto para ser implementado.

V. Resultados Experimentais

Para a realização experimental, os controladores foram programados em linguagem C utilizando a IDE do Arduino. As variáveis dos controladores foram declaradas como tipo *float* para garantir a precisão numérica necessária.

Uma vez que os controladores foram projetados no domínio da frequência, foi necessário realizar a sua discretização. Foi empregado o método de Euler para esta finalidade, por motivo de simplicidade, considerando a finalidade didática deste trabalho. A referência [5] apresenta detalhes a respeito da discretização de modelos e de controladores.

A. Implementação do PI

A partir dos parâmetros do controlador PI obtidos na equação (8) através da função de desempenho desejado dada pela equação (7), foi realizada a discretização para a implementação digital. A discretização foi realizada da seguinte forma:

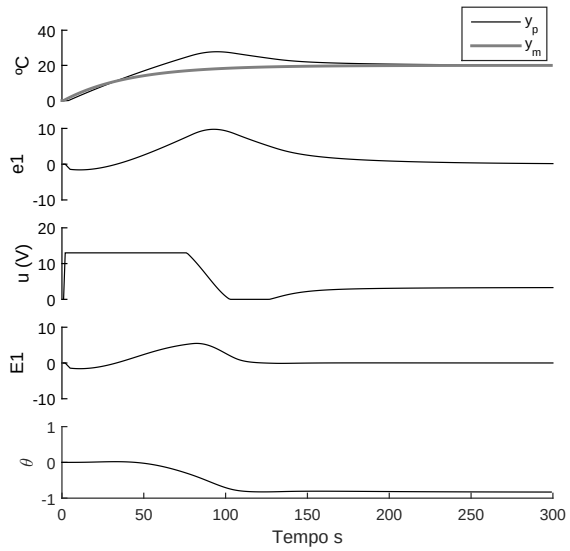

Figura 6. Simulação com RMRAC.

$$u_i(k) = u_i(k-1) + T_s k_p k_i e(k)$$
$$u(k) = k_p e(k) + u_i(k). \quad (15)$$

A implementação em código C deste controlador é simples, porém devem ser tomados alguns cuidados, como por exemplo a inserção de um limite para o integrador e também para o valor do ciclo ativo do PWM. Basta que sejam definidos os limites, de acordo com os valores de saturação da saída da lei de controle. O tempo entre as amostragens T_s utilizado foi de $1s$, logo o código do controlador deve ser executado a cada intervalo T_s. Em código C a implementação da equação 15 se torna:

```
while(1) {
lm35 = analogRead(LM35_PIN); // conv. ADC LM35
t_lm35 = ((float)lm35*1.1/1023); // ADC -> V
t_lm35 = (t_lm35 + 0.02) * 100; // V-> Graus
e = r - t_lm35; // erro
ui = ui + kp*ki*Ts*e; // Integral
// limites da integral
if(ui > 255)  ui = 255;
if(ui < 0)    ui = 0;
u = ui + kp*e;
up = (int) (u/k*(255/13));
if(up < 0)    up = 0;
if(up > 255)  up = 255;
analogWrite(PwrOut, up); // ajusto PWM
delay(1000); // Ts = 1 s
}
```

Neste código, a variável r é a referência desejada, t_{lm35} armazena a temperatura lida do sensor, em graus Celsius, u é o valor calculado da lei de controle e k é o ganho de Bode da planta. O valor de saída da lei de controle precisa ser escalonado de acordo com a faixa de variação do PWM (0 a 255) para poder ser escrito no registrador de ciclo ativo do PWM. As variáveis são do tipo *float*, exceto *up*, que armazena o valor do ciclo ativo, que é um inteiro. O ADC foi configurado para a utilização da tensão de referência interna, de 1,1 V. O

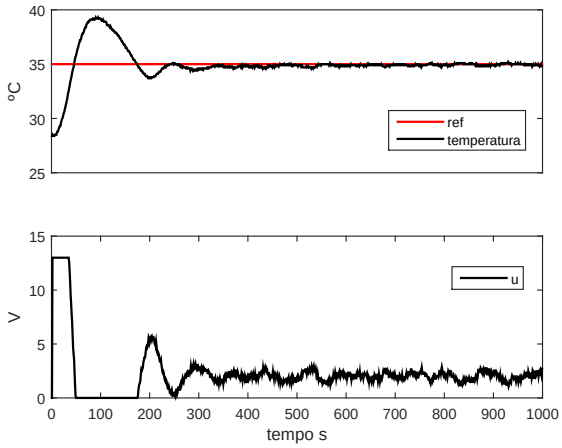

Figura 7. Temperatura e ação de controle com controlador PI.

código de inicialização das variáveis, dos pinos e a função de envio de dados pela serial foi omitido por limitação de espaço, porém são de fácil implementação.

Na Figura 7 é apresentado o resultado experimental com controlador PI e referência de 32 °C. Pela resposta da planta é notório o *overshoot* de temperatura, após o qual o controlador é capaz de fazer com que a saída da planta rastreie a referência. A dúvida que surge é o porque da diferença em relação a simulação. Para obter a resposta, basta que se verifique o projeto do controlador, no qual foram assumidas simplificações, inclusive o atraso de transporte foi ignorado.

B. *Implementação do RMRAC*

Para realizar a implementação do controlador adaptativo, as funções com dinâmica foram discretizadas e programadas em linguagem C. O código programado para a implementação é apresentado a seguir. O cabeçalho, contendo as definições:

```
#include <math.h> // fabs()
#define Am      0.015
#define Bm      0.015
#define c0      1.0
#define delta0  0.5
#define delta1  0.9
#define M0      2.0
#define M02     (2.0*M0)
#define SIGMA0  0.1
#define GAMA    5.0
#define Ts      1.0
```

E a função contendo o código do controlador RMRAC:

```
float RMRAC(float yp, float ref) {
static float th = 0.3;
static float Sigma = 0;
static float v = -23.5;
static float zeta = 25;
static float xm = 25.0;
static float m = (delta1/delta0)*1.5;
float E1, ym, t, u, m2;

u = th*yp + c0*ref;
ym = xm;
E1 = yp - ym + th*zeta - v;
// Wm[r]
```

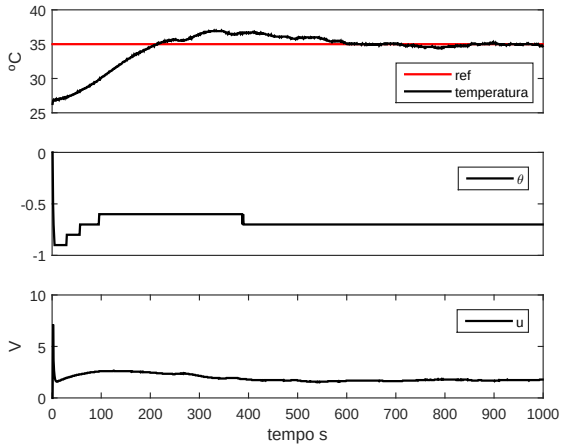

Figura 8. Resultados experimentais com o controlador RMRAC.

```
t = Bm*Ts*ref;
xm = (1 - Am*Ts)*xm + t; // xm(k+1)
// zeta(k+1) = Wm[w]
t = Bm*Ts*yp;
zeta = (1 - Am*Ts)*zeta + t;
// v(k+) = Wm[th*zeta]
t = Bm*th*zeta;
v = (1 - Am*Ts)*v + t*Ts;
t = delta1 + fabs(u) + fabs(yp);
m = (1 - delta0*Ts)*m + t*Ts;
m2 = m*m;
t = fabs(th);
if(t < M0) {
        Sigma = 0;
}else if(t < M02) {
        Sigma = SIGMA0*(t/M0-1);
}else { Sigma = SIGMA0; }
// th(k+1)
th = (1 - GAMA*Sigma*Ts)*th - Ts*GAMA*E1*zeta/m2;
return(u);
}
```

A título de organização, todo o código do RMRAC foi concentrado em uma função que retorna o valor da lei de controle. Na implementação, o código do controlador normalmente é inserido em uma interrupção do microcontrolador. Dessa forma, a lei de controle é calculada logo no início do código e o valor é atualizado no *hardware*, que neste caso corresponde a saída de PWM.

Na Figura 8 são apresentados os resultados obtidos da implementação com o RMRAC. Os principais parâmetros do controlador são apresentados, restando evidente a sua convergência. A resposta é relativamente lenta, porém estável e de baixa variação na saída da lei de controle, que aciona a planta de forma suave e com baixo ruído.

Finalmente, os componentes de *hardware* utilizados são apresentados na Figura 9.

VI. Discussão e sugestões

A realização dos experimentos propostos traz a tona algumas questões interessantes pertinentes ao aprendizado de sistemas de controle. Primeiramente, relativo a modelagem, fica evidente que simplificações em termos de modelo geralmente

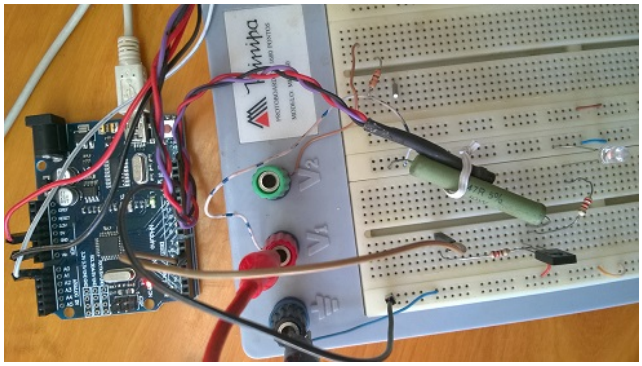

Figura 9. Material experimental.

trazem consequências quando da implementação, devendo o projetista estar atento a isso. Outros fatores interessantes de se considerar são a resolução do ADC e a influência do ruído presente nos sinais analógicos.

Como sugestões de ampliação, com finalidade didática, pode-se citar: Projeto de controladores PI considerando o atraso de transporte e também a influência do amostrador (ADC); implementação de controladores preditivos; implementação de observadores de estado para melhorar as medidas; utilização do *oversampling* [6] para melhorar a resolução do ADC; utilizar outros métodos de discretização.

VII. Conclusão

Este trabalho mostrou a implementação de dois controladores, um do tipo PI e outro RMRAC, aplicados a uma planta de primeira ordem, com o objetivo didático de mostrar os passos necessários para a realização de um sistema de controle. Foi demonstrada a modelagem da planta (processo térmico) através da sua resposta a um acionamento do tipo degrau, obtendo assim a sua função de transferência. De posse do modelo, assumindo algumas simplificações, foram realizados os projetos dos controladores PI e Adaptativo, simulação e implementação.

A análise dos resultados e comparação dos valores de simulação com aqueles obtidos da realização experimental evidenciaram algumas diferenças peculiares de quando são assumidas simplificações nos projetos. A realização proposta, que pode ser reproduzida de forma simples e segura, mostrou-se relevante para estudos da prática de sistemas de controle.

Referências

[1] Norman S. Nise. Engenharia de Sistemas de Controle, 3º ed, 2002. Livros Técnicos e Científicos Editora Ltda.

[2] Tania B. I. Marques. Professor ou pesquisador?. In: Becker, Fernando; Marques, Tania (Org.). Ser professor é ser pesquisador. Porto Alegre: Mediação, 2010.

[3] Arduino. Disponível em <https://www.arduino.cc>. Acessado em 15/07/2018.

[4] P. Ioannou and K. Tsakalis. "A robust direct adaptive controller,"in IEEE Transactions on Automatic Control, vol. 31, no. 11, pp. 1033-1043, Nov 1986.

[5] R. J. Vaccaro. Digital Control A State-Space Approach. 1995, McGraw-Hill, Inc.

[6] R. Lyons. Understanding Digital Signal Processing, 3rd Edition, 2011. Prentice Hall.