PID temperature control with ESP32 and FreeRTOS

Diego Luiz de Oliveira Batista
Depart. de Eng. Elétrica
Instituto Federal do Paraná - Paranavaí
Paranavaí, PR, Brasil
diego.luiz236@gmail.com

Edno Gentilho Júnior
Depart. de Eng. Elétrica
Instituto Federal do Paraná - Paranavaí
Paranavaí, PR, Brasil
edno.junior@ifpr.edu.br

***Resumo*—Control theory applied with embedded systems allowed a great advance in industrial processes. With the application of sensors and microcontrollers, a physical quantity can be measured and controlled. This work presents the development of a PID controller applied to a heating plant. The development of the electronic circuit, the controller design and the experimental results are presented and discussed.**

***Index Terms*—FreeRTOS, ESP32, PID controller, Temperature control.**

I. Introdução

A temperatura é uma das grandezas físicas mais importantes no controle de diversos processos industriais [1]. A sua medição e controle está presente em equipamentos que vão desde simples eletrodomésticos até à sofisticados sistemas na industria petroquímica.

A correta aferição e controle da temperatura são muito importantes, principalmente em processos químicos sensíveis que possuem alta dependência da temperatura, como na industria farmacêutica [2].

Há diferentes sensores para a medição de temperatura, cada um com suas particularidades físicas e faixas de atuação. Os termopares possuem a maior faixa de atuação, podendo ser de $-270°C$ à $2.600°C$ com precisão em torno de $\pm 1°C$ [1]. Os RTDs (*Resistance Temperature Detectors*) trabalham em uma faixa de temperatura de $-200°C$ à $600°C$, com precisão em torno de $\pm 0,2°C$ [1].

Outra classe de sensores de temperatura são os sensores de temperatura de circuito integrado [3]. Por serem constituídos de materiais semicondutores, sua faixa de temperatura é pequena [3]. Sua faixa de trabalho é normalmente de $-40°C$ à $150°C$, com precisão em volta de $\pm 1°C$ [1]. Estes sensores eletrônicos já possuem internamente circuitos dedicados ao tratamento e conversão dos sinais, com alguns já entregando em sua saída sinais digitais que podem ser conectados diretamente as entradas digitais de diferentes microcontroladores [1].

O objetivo deste trabalho é o de apresentar o desenvolvimento de um controlador PID (Proporcional Integral e Derivativo) aplicado no microcontrolador ESP32 para o controle de aquecimento de um recipiente com água.

O desenvolvimento do circuito eletrônico, bem como os resultados experimentais obtidos, são apresentados e discutidos neste trabalho.

II. Circuito eletrônico de potência e controle

O sistema de aquecimento é composto por um recipiente com, aproximadamente, 2 litros de água a temperatura ambiente e um resistor com potência de $1.250W$ e tensão de $127V_{rms}$.

O controle da temperatura é feito por meio do número de ciclos da senoide da rede elétrica que são entregues ao resistor de aquecimento em um período de tempo. Há a possibilidade de definir qualquer período no qual se tem um número fixo de ciclos. Neste trabalho, o período corresponde a um segundo, que possui um número de 60 ciclos. Com o número de ciclos igual à zero o resistor está totalmente desligado e, com 60 ciclos, o resistor se encontra totalmente ligado em um período de um segundo.

O dispositivo que se utilizou como chave para controlar o número de ciclos que são enviados ao resistor, é o TRIAC BTA12-600B com corrente nominal de operação de $12A_{rms}$ [4].

Para uma operação segura, e isolação do microcontrolador, o circuito que se desenvolveu é composto por duas partes, sendo a primeira um circuito de potência responsável por atuar diretamente no resistor de aquecimento junto a rede elétrica, e a segunda sendo o controlador formado pelo microcontrolador ESP32 e um circuito de baixa potência.

O circuito de potência é constituído pelo TRIAC BTA12-600B, o resistor de aquecimento e um resistor que limita a corrente de entrada na porta de acionamento do TRIAC.

Os sinais de disparo são provenientes do ESP32 e não são conectados diretamente ao TRIAC BTA12-600B, com o TRIAC optoisalador MOC3021 fazendo a ponte de conexão entre os estágios de potência e de controle. Para evitar disparos acidentais do optoisolador, se utilizou de um circuito RC *snubber*.

Para a correta determinação do número de ciclos que são entregues ao resistor de aquecimento a cada segundo, é necessário sincronizar o microcontrolador com a rede elétrica. Para esta sincronização se projetou um detector de passagem por zero que gera uma interrupção externa no microcontrolador a todo o momento em que a senoide da rede estiver em $0V$.

O circuito de sincronização é constituído por um transformador que entrega $9V_{rms}$ na sua saída, um optoacoplador 4N25 e um resistor de *pull-up* de $10k\Omega$. Além disto, se utilizou duas portas lógicas *schmitt trigger* do circuito integrado

ISSN 2525-9458

CD4093BE para ajustar o sinal de sincronização. O circuito completo de potência e controle é ilustrado na Figura 1.

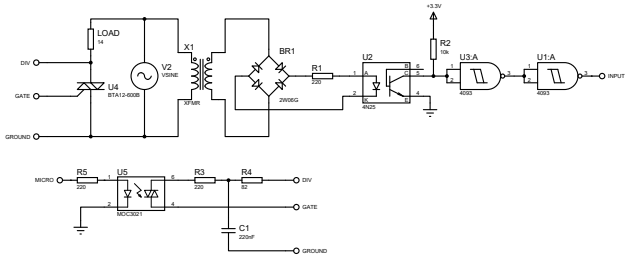

Figura 1. Esquemático dos estágios de potência e controle

Além destes componentes eletrônicos, o sistema conta com dois *push buttons* para a entrada no microcontrolador da temperatura de *setpoint*, e um display LCD 16×2 para a visualização da temperatura dada pelo sensor e a temperatura de *setpoint*.

Para a medição da temperatura se empregou o sensor digital DS18B20, que possui resolução variável de 9 bits à 12 bits [5], sendo que neste trabalho o sensor operou com resolução de 12 bits. Nesta resolução, o tempo máximo de conversão do sensor é de $750ms$ [5]. A faixa de atuação do DS18B20 é de $-55°C$ a $+125°C$, com erro de $\pm 2°C$ [5].

III. Identificação da Planta

O ESP32 foi programado na linguagem do Arduíno com a implementação do FreeRTOS (*Real Time Operational System*). Muitos microcontroladores são compostos por um processador que é capaz de executar uma tarefa por vez. O FreeRTOS se baseia na ideia de *multitasking* (multitarefa), no qual o microcontrolador alterna rapidamente entre as tarefas dando a impressão de que todas estão sendo executadas ao mesmo tempo [6]. Portanto, cada tarefa possui seu período de execução, o que possibilita gerenciar o uso do *hardware* por parte de cada tarefa.

Para a identificação da planta se desenvolveu um programa composto por três tarefas, conforme listado na Tabela I. A primeira tarefa é responsável por acionar o TRIAC BTA12-600B e é a tarefa com maior prioridade dentre as três. A segunda tarefa tem por objetivo medir a temperatura e envia-lá para a porta serial para registro no software MATLAB. Já a terceira, e última tarefa, mostra a temperatura no display LCD. A Tabela I também lista o núcleo de execução de cada tarefa, já que o ESP32 conta com dois núcleos.

Tabela I
Tarefas para a identificação da planta

Tarefa	Atividade	Prioridade	Núcleo
1	Acionamento do BTA12-600B	3	0
2	Temperatura e serial	1	1
3	Display	2	1

A identificação da planta é feita com base na sua resposta à um sinal *step* de entrada. Neste caso, inicialmente o resistor se encontrava totalmente desligado. Após 20 segundos o resistor é totalmente ligado e a temperatura é amostrada a cada um segundo e enviada via serial para o MATLAB. O sinal *step* de entrada considerado varia de 0 a 1, ou seja, 0% de ciclos e 100% de ciclos em um segundo. O tempo total de observação da planta foi de 1.570 segundos. A Figura 2 ilustra o gráfico de temperatura que se obteve.

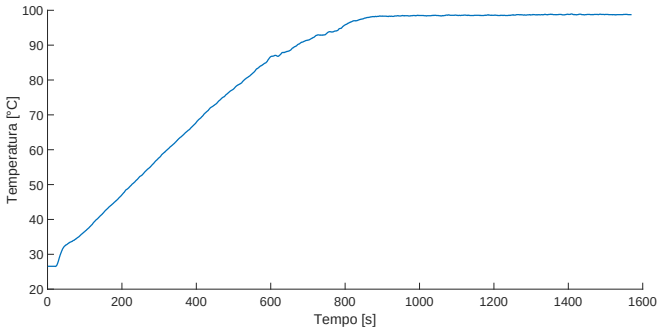

Figura 2. Resposta da planta ao sinal *step* de entrada

Com os dados da resposta da planta, e com a ferramenta pidTuner do MATLAB, se chegou a função de transferência descrita em (1), com ajuste de $92,34\%$.

$$G(s) = \frac{5,0691}{651,52s + 1} \tag{1}$$

Na Figura 3 se têm a ilustração dos gráficos da resposta da planta e da função de transferência definida em (1).

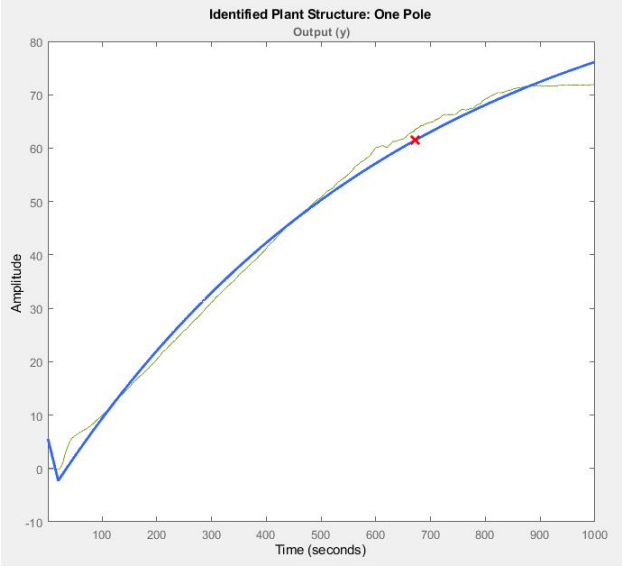

Figura 3. Gráfico da resposta da planta em verde e da função de transferência estimada em azul

IV. Controlador PID

Para a determinação dos parâmetros do controlador PID também se utilizou a ferramenta pidTuner. No pidTuner é possível ajustar a resposta da planta e ver o seu resultado em tempo real.

Os parâmetros K_p, K_i e K_d foram determinados de modo a se ter o menor *overshoot* possível. Os valores obtidos para o controlador PID são $K_p = 0,4126$, $K_i = 1/578,6$ e $K_d = 0,03043$. Na Tabela II se têm as características do controlador PID projetado.

Tabela II
CARACTERÍSTICAS DO CONTROLADOR PID

Settling Time	Rise Time	Overshoot
$983s$	$618s$	$0,642\%$

Da mesma forma que para a identificação da planta, para a aplicação do controlador PID também se utilizou o FreeRTOS. Neste caso, o programa é dividido em cinco tarefas, conforme listado na Tabela III.

Tabela III
TAREFAS PARA A APLICAÇÃO DO CONTROLADOR PID

Tarefa	Atividade	Prioridade	Núcleo
1	Acionamento do BTA12-600B	5	0
2	Leitura dos *push buttons*	4	1
3	Leitura da temperatura	1	1
4	Controlador PID e envio de dados	3	1
5	Display	2	1

A terceira tarefa, que faz a leitura da temperatura, é executada a cada $980ms$. Já a quarta tarefa, que calcula a ação de saída do controlador PID, é executada a cada $1s$. Como o controlador PID depende da leitura da temperatura para calcular o erro, há um tempo de atraso de $20ms$ entre a leitura e a ação do controlador PID. Por se tratar de uma planta de resposta lenta, este tempo de atraso não compromete a aplicação do controlador.

Como mencionado anteriormente, o sinal *step* de entrada foi considerado entre 0 e 1. Portanto, a saída do controlador PID está compreendida entre 0 e 1, que corresponde ao percentual do número de ciclos que deve ser entregue ao resistor de aquecimento.

Além da temperatura da planta ser registrada no MATLAB através de comunicação serial, se configurou uma rede Modbus TCP/IP com a finalidade de enviar os valores de temperatura por rede ao *software* LabVIEW. Devido a limitações da biblioteca de rede utilizada na programação, os valores de temperatura enviados por rede ao LabVIEW são do tipo inteiro, ao contrário da comunicação serial com o MATLAB que permite o envio de valores do tipo *float*. No entanto, para aplicações que não requerem os valores decimais de temperatura, está limitação não é um problema.

O envio dos dados de temperatura por serial e rede são feitos pela quarta tarefa. A escolha pela quarta tarefa se deve ao seu período de execução, o que garante que a temperatura amostrada seja registrada a cada um segundo pelo MATLAB e LabVIEW.

V. RESULTADOS E DISCUSSÃO

Após a programação do microcontrolador, e a inserção dos ganhos do controlador PID, se realizou um experimento no qual se definiu três temperaturas de *setpoint*, sendo a primeira de $50°C$, a segunda de $60°C$ e a terceira de $80°C$. Como maneira de validar o controlador projetado, se aplicou um distúrbio nos *setpoints* de $60°C$ e $80°C$.

Como já mencionado, o recipiente é preenchido com 2 litros de água a temperatura ambiente, e para a geração dos distúrbios se adicionou $200ml$ de água congelada.

O tempo total de observação da resposta da planta foi de 4.015 segundos. A Figura 4 ilustra a resposta da planta registrada no MATLAB ao longo de todo o período de observação.

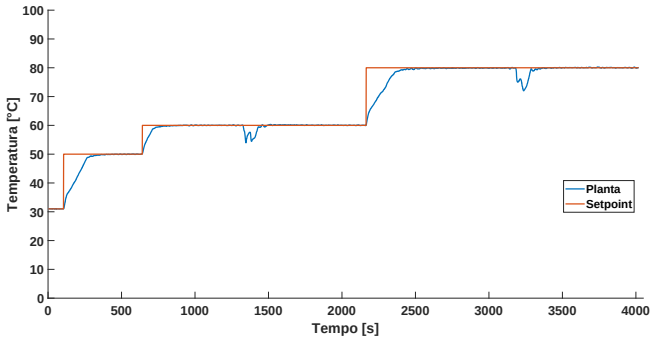

Figura 4. Resposta da planta registrada pelo MATLAB através de comunicação serial

Na Figura (5) se tem ilustrado o gráfico da resposta completa da planta que se obteve pelo LabVIEW.

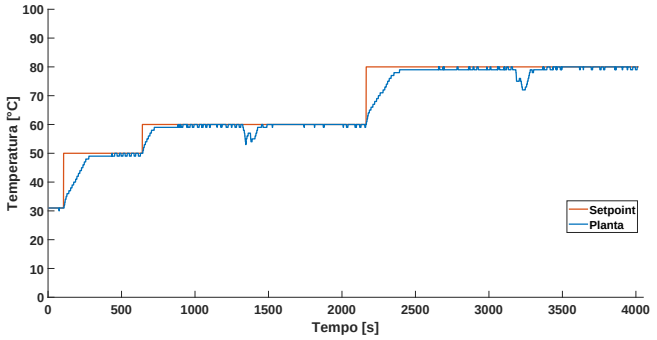

Figura 5. Resposta da planta registrada pelo LabVIEW por rede Modbus TCP/IP

Para o primeiro *setpoint* de $50°C$, se tem na Figura 6 os gráficos da resposta da planta obtida no MATLAB e LabVIEW. Devido a limitação de envio de dados do tipo inteiro pela rede Modbus TCP/IP comentada anteriormente, os valores de *overshoot* são obtidos com base nos dados do MATLAB.

Para o *setpoint* de $50°C$ o maior valor de temperatura registrado foi de $50,06°C$, que corresponde a um *overshoot* de $0,12\%$.

A Figura 7 ilustra o gráfico da resposta da planta para o *setpoint* de $60°C$, bem como o seu comportamento em relação ao distúrbio aplicado. Após a aplicação do distúrbio, a planta levou em torno de 164 segundos para se estabilizar novamente em torno do valor de referência. O *overshoot* registrado foi de $0,5167\%$ decorrente da aplicação do distúrbio.

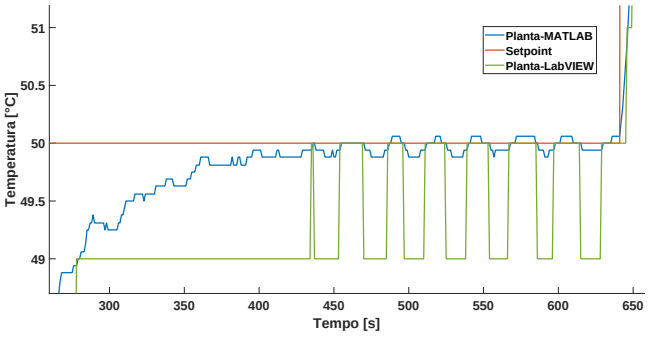

Figura 6. Reposta da planta para o *setpoint* de $50°C$

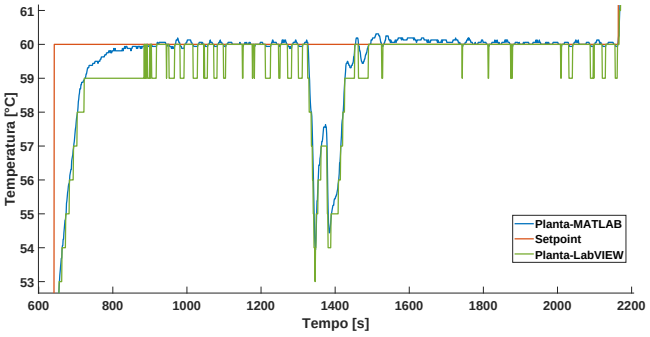

Figura 7. Reposta da planta para o *setpoint* de $60°C$

Já na Figura 8 se tem o gráfico da resposta da planta em relação ao *setpoint* de $80°C$. O tempo entre a aplicação do distúrbio e a estabilização da planta em torno do *setpoint* foi de 165 segundos, com um *overshoot* registrado de $0,3125\%$ após a aplicação do distúrbio.

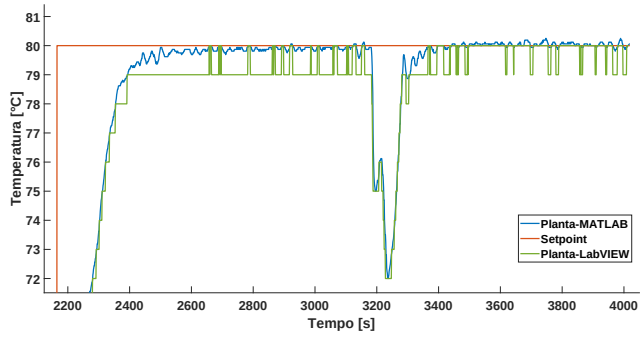

Figura 8. Reposta da planta para o *setpoint* de $80°C$

VI. Conclusão

Neste trabalho foi apresentado e discutido o projeto de um controlador PID aplicado à uma planta de aquecimento de água.

Os resultados apresentados demonstraram a efetividade do sistema desenvolvido e aplicado, com os valores de *overshoot* se mantendo abaixo do valor estimado durante o projeto do controlador.

O sistema de controle se mostrou capaz de retornar a normalidade após a aplicação de distúrbios na planta.

O FreeRTOS se mostrou uma excelente ferramente, já que permite separar a aplicação em diversas tarefas com períodos específicos de execução, o que torna o sistema mais eficiente ao possibilitar o gerenciamento de uso de *hardware*.

Referências

[1] D. Ibrahim, *Microcontroller-based temperature monitoring and control*. Elsevier, 2002.

[2] W. C. Dunn, *Introduction to instrumentation, sensors, and process control*. ARTECH HOUSE, 2005.

[3] T. A. Hughes, *Measurement and control basics*. ISA, 2015.

[4] *BTA/BTB12 and T12 Series*, STMicroelectronics, 2002. [Online]. Available: https://storage.googleapis.com/baudaeletronicadatasheet/BTA12.pdf

[5] *DS18B20 Programmable Resolution 1-Wire Digital Thermometer*, Dallas Semiconductor. [Online]. Available: https://cdn.sparkfun.com/datasheets/Sensors/Temp/DS18B20.pdf

[6] M.-Y. Zhu, "Understanding freertos: A requirement analysis," *CoreTek Syst., Inc., Beijing, China, Tech. Rep*, 2016.