ASPECTOS DE IMPLEMENTAÇÃO DE CONTROLADOR PI EM AMBIENTE SIMULADO

VARASCHIM, F. H.[1]; ALMEIDA, J. P. L. S. de.[2]

[1] Discente do curso de Engenharia de Controle e Automação, IFPR, Jacarezinho, Paraná, e-mail: filipe_varaschim@hotmail.com
[2] Doutor em Engenharia Elétrica e Informática Industrial, Instituto Federal do Paraná (IFPR), Jacarezinho, Paraná, e-mail: joao.almeida@ifpr.edu.br

RESUMO

Sistemas de controle e de automação estão cada vez mais presentes nas demandas industriais e, por este motivo, os conceitos que envolvem tais temas são importantes elementos de estudo e pesquisa em cursos de Engenharia de Controle e Automação, tanto quando considerados em experimentos em plantas reais, em escalas laboratoriais, quanto em simuladores. Neste contexto, este trabalho tem os objetivos de apresentar os principais aspectos de implementação de um controlador Proporcional-Integral no ambiente de simulação *Tinkercad*, por meio do *hardware* de controle Arduino, e mostrar as principais observações que podem ser efetuadas por um estudante/pesquisador. Os componentes utilizados são padrões do simulador. Para fins de validação, o controlador foi utilizado para o controle de um processo, representado neste trabalho por um sistema de primeira ordem. Os resultados mostram que a simulação é uma opção válida para o estudo de implementação de controladores PI.

PALAVRAS-CHAVE: Sistemas de Controle, Controle PI, Simulação, *Tinkercad*.

INTRODUÇÃO

Com a industrialização cada vez mais exigente e avançada, produzindo mercadorias com maior qualidade, agilidade e eficácia, se faz necessário estabelecer maneiras eficientes de controlar os processos produtivos e, neste aspecto, a escolha de um controlador adequado pode ser crucial para que os requisitos operacionais sejam atingidos.

Um dos controladores mais conhecidos e usados em ambientes industriais é o do tipo Proporcional-Integral-Derivativo (PID), incluindo suas variações, P, PI e PD. A característica básica destes controladores, que os tornam muito populares, é o fato de quando adequadamente ajustados (manualmente ou automaticamente), eles geralmente conduzem o sistema a um desempenho satisfatório, em termos de redução de sobressinais, de erros em regime e tempo de acomodação (OGATA, 2003).

Neste trabalho, a implementação de um controlador do tipo PI será apresentada sob a perspectiva do uso de simuladores para tal finalidade. Neste contexto, considerou-se a necessidade de controlar a saída de um sistema de primeira ordem, que pode ser a representação de diversos tipos de processos encontrados em demandas industriais, tais como os que envolvem o controle de nível de fluido, de temperatura, de vazão e de pressão, dadas as devidas simplificações matemáticas acerca de suas características dinâmicas. A implementação do controlador e do sistema de primeira ordem foi realizada por meio do simulador *on-line Tinkercad (disponível em: https://www.tinkercad.com/)*. Uma das principais contribuições deste trabalho é a de mostrar aspectos de implementação, ainda que em simulação, do controlador PI em um *hardware* de baixo custo e de fácil acesso por pesquisadores, que se trata da plataforma Arduino, além de afirmar que os ambientes de simulação podem ser uma opção válida para os estudos de sistemas de controle.

METODOLOGIA

De uma forma geral, a simulação realizada objetiva implementar um controlador PI para o controle da saída de um sistema de primeira ordem de representação genérica, como apresentado pela Equação (1), em que: K é o ganho do processo; τ é a constante de tempo; $Y(s)$ é a saída; e $U(s)$ é a entrada. Assume-se, como variável manipulada deste sistema, a velocidade de rotação de um motor DC com *encoder*, que pode representar, indiretamente, a velocidade de uma bomba centrífuga para o deslocamento

de fluido, o atuador de um sistema de refrigeração, etc. A saída do sistema (variável controlada) é uma medida que dependerá do processo a ser representado pela Equação (1) e, portanto, pode ser a medida de nível em um reservatório, a temperatura de um determinado objeto, a vazão de fluido em uma tubulação, entre outras possibilidades.

$$\frac{Y(s)}{U(s)} = \frac{K}{\tau s + 1} \tag{1}$$

Inicialmente, para que o sistema da Equação (1) pudesse ser implementado no *hardware* digital de simulação, aplicou-se uma discretização na função de transferência considerando: $K = 1, \tau = 1$ *e* $dt = 0{,}1$s, com o método retentor de primeira ordem. O resultado da discretização é mostrado na Equação (2).

$$\frac{Y(z)}{U(z)} = \frac{0{,}04837z + 0{,}04679}{z - 0{,}9048} \tag{2}$$

Inicialmente, foi montado no simulador *Tinkercad* um circuito baseado em (PAVANI, 2021), mostrado na Figura 1, em que podem ser observados os seguintes componentes: uma placa Arduino UNO, uma matriz de contatos (*protoboard*), um motor DC com *encoder*, um circuito para o condicionamento do sinal do *encoder* (transistor TIP120, resistor de 10 kΩ e resistor de 5 Ω), uma fonte de tensão variável e conectores gerais.

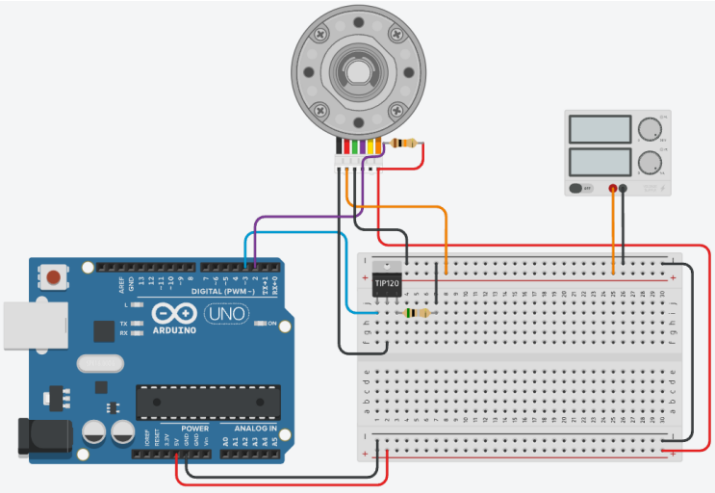

Figura 7 – Circuito eletrônico no simulador Tinkercad (adaptado de Pavani (2021).

Em seguida, foi estruturado o algoritmo do controlador PI, o qual é mostrado nas Figuras 2 e 3. O algoritmo foi estabelecido de tal forma que o usuário estabeleça um valor para *setpoint* e para os ganhos K_p e K_I do controlador. Além disso, pode-se observar na linha 53 (Figura 3), que o sinal PWM é aplicado ao motor e, na linha 37 (Figura 3), que é realizado o cálculo da velocidade do motor, variável que é aplicada como entrada do sistema de primeira ordem discretizado, como mostrado na linha 39 (Figura 3). É importante ressaltar que, apesar da representação da planta a ser controlada (Equação (2)) ser incluída no algoritmo executado pelo próprio *hardware* Arduino, o que não é a forma mais adequada de simulação, demais parâmetros de implementação podem ser observados, tais como a utilização da linguagem difundida para plataformas Arduino, a geração de sinais PWM compatíveis com o atuador (motor DC), a utilização da instrução para a leitura do *encoder* (medida de velocidade do motor DC), entre outras.

```
// Mapeamento Hardware
#define PWM_OUT       3
#define ENCODER_A 2

int output;           // define a saida de pwm
int output_last; // sinal de controle anterior
int speed;            // velocidade do motor
int speed_last; // velocidade do motor anterior
int Y = 0;            // sinal de saida da FT
int Y_last = 0;      // sinal de saida anterior da FT
int error = 0;        // erro atual
int lastError = 0;    // declara o último erro
int setpoint = 300;         // declara o setpoint
int I_error = 0;      // ganho integral
float KP = 1;       // constante proporcional
float KI = 0.1;    // constante integrativa

void setup() {
  Serial.begin(9600); // inicia a comunicação serial

  pinMode(ENCODER_A, INPUT); // define o encoder para enviar dados
  pinMode(PWM_OUT, OUTPUT);
  // define a porta 3 para envio de sinal PWM para controle de velocidade

  output = 10;              // define um valor inicial para pwm_valor
  output_last = output;
  analogWrite(PWM_OUT, output); // controla o valor pwm através da saida porta 3

  // realiza a leitura do encoder e transforma o valor dos pulsos em velocidade
  speed = 19.1*((60*1000*10) / pulseIn(ENCODER_A, HIGH));
  speed_last = speed;
  }
```

Figura 2 – Algoritmo do controlador PI parte 1.

```
void loop() {

  // realiza a leitura do encoder e transforma o valor dos pulsos em velocidade
  speed = 19.1*((60*1000*10) / pulseIn(ENCODER_A, HIGH));

  Y = 0.04837*speed + 0.04679*speed_last + 0.9048*Y_last;

  error = setpoint - Y; // calcula o valor do erro

  I_error += error; // somatorio do erro

  // Implementação do Algoritimo PI

  output = KP*error + KI*I_error;

  // limita a saida do controle para os valores válidos de pwm (10-255)
  output = constrain(output, 10, 255);

  // envia o sinal para o "motor"
  analogWrite(PWM_OUT, output);

  // Imprime Speed, Y e Setpoint
  Serial.print(speed);
  Serial.print(",");
  Serial.print(Y);
  Serial.print(",");
  Serial.println(setpoint);

  speed_last = speed;
  Y_last = Y;

  delay(100);
}
```

Figura 3 – Algoritmo do controlador PI parte 2.

RESULTADOS E DISCUSSÕES

Dada a utilização do algoritmo apresentado nas Figuras 2 e 3, os resultados de duas simulações são apresentados nas Figuras 4 – 7. Especificamente, nas Figuras 4 e 5 são apresentadas, respectivamente, a resposta do sistema considerando um controlador PI ($K_p = 1$ e $K_I = 0{,}1$) e a evolução da ação de controle ao longo do tempo. A fim de mostrar uma segunda possibilidade de simulação, para o mesmo algoritmo, o termo K_I foi anulado do controlador, com o objetivo de estabelecer um controlador P, cujos resultados de resposta do sistema e ação de controle são apresentados nas Figuras 6 e 7, respectivamente. No segundo experimento, nota-se que apenas a ação proporcional não foi suficiente para levar a saída do sistema ao valor de *setpoint*, o que representa uma situação de erro em regime.

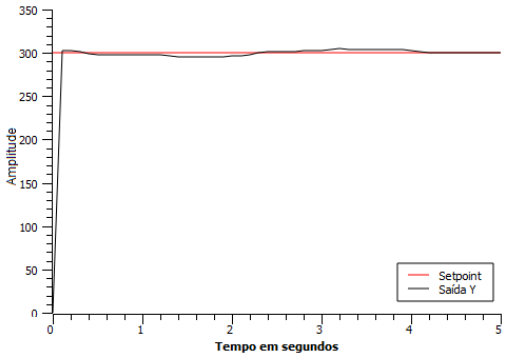

Figura 4 – Resposta do sistema ($K_p = 1$ e $K_I = 0{,}1$).

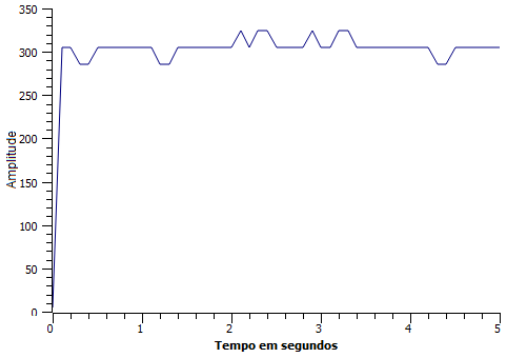

Figura 5 – Ação de controle ($K_p = 1$ e $K_I = 0{,}1$).

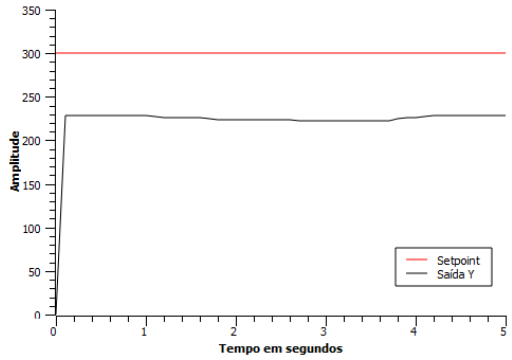

Figura 6 – Resposta do sistema ($K_p = 1$).

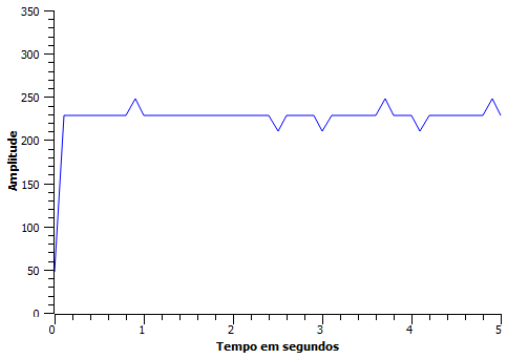

Figura 7 – Ação de controle ($K_p = 1$).

CONSIDERAÇÕES FINAIS

O presente resumo expandido apresentou os principais aspectos de implementação, via simulador *Tinkercad*, de um controlador PI em uma malha de controle cujo processo a ser controlado foi representado por um sistema de primeira ordem. Nas simulações, foi considerado um conjunto de *hardware* de baixo custo e de fácil acesso por pesquisadores, que se trata da plataforma Arduino e de componentes comuns compatíveis.

Para fins de validação foi apresentada a resposta do sistema de primeira ordem, sob atuação do controlador PI, para duas condições de ganhos de K_p e K_I. Os resultados mostraram uma resposta já esperada considerando este conjunto de aplicações, em relação ao erro em regime permanente, tempo de subida, entre outros conceitos abordado em teorias de controle.

Vale ressaltar que, como o foco deste trabalho foi de mostrar aspectos de implementação do controlador PI na plataforma Arduino, ou seja, uma etapa que vem após a aplicação de metodologias de projeto (sintonia e ganhos) deste tipo de controlador, as análises do ponto de vista de controle não foram consideradas, tais como a aplicação de métodos de sintonia, a experimentação com outros elementos dinâmicos no sistema (perturbação, ruído, características não-lineares, etc). Além disso, ainda que a representação da planta a ser controlada seja incluída no algoritmo executado pelo *hardware* de controle, o que não é um método convencional de simulação, pôde-se observar os principais aspectos de implementação do código em um *hardware* comum e de tecnologia aberta. Portanto, estas abordagens ausentes podem ser consideradas em trabalhos futuros.

REFERÊNCIAS

OGATA, Katsuhiko; **ENGENHARIA DE CONTROLE MODERNO.** Tradução Paulo Alvoro Maya; Revisão técnica Fabricio Leonardi, São Paulo, Prentice Hall, 2003.

PAVANI, V. **Controle PID Utilizando Arduino.** UFRJ Nautilus. Disponível em: https://pt.ufrjnautilus.com/post/controle-pid-utilizando-arduino. Acesso em: 20/10/2021.