FACULDADE DE TALENTOS HUMANOS – FACTHUS

RAFAEL FARIA FELIX

CONSTRUÇÃO DE UM PÊNDULO INVERTIDO A PARTIR DE RESÍDUOS DE EQUIPAMENTOS ELETRÔNICOS

UBERABA – MG

2017

FACULDADE DE TALENTOS HUMANOS – FACTHUS

RAFAEL FARIA FELIX

CONSTRUÇÃO DE UM PÊNDULO INVERTIDO A PARTIR DE RESÍDUOS DE EQUIPAMENTOS ELETRÔNICOS

Trabalho de Conclusão de Curso apresentado à Faculdade de Talentos Humanos – FACTHUS, como requisito parcial de obtenção de título de bacharel em Engenharia Mecânica.
Orientador: Esp. Cleiton Silvano Goulart

UBERABA – MG

2017

Dados Internacionais de Catalogação na Publicação (CIP)
Joilsa Fonseca de Oliveira – CRB6 / 2639

F31c
2017

Felix, Rafael Faria, 1991-
Construção de um pêndulo invertido a partir de resíduos de equipamentos eletrônicos / Rafael Faria Félix. -- Uberaba, 2017.
11 f. : il.

Orientador: Esp. Cleiton Silvano Goulart.
Trabalho de conclusão de curso (Graduação em Engenharia Mecânica). Faculdade de Talentos Humanos (FACTHUS).
Inclui bibliografia.

1. Arduino. 2. Controle proporcional integral derivativo. 3. Pêndulo invertido. 4. PID. 5. Reutilização. I. Goulart, Cleiton Silvano, 1988. II. Título.

CDU: 621

RAFAEL FARIA FELIX

CONSTRUÇÃO DE UM PÊNDULO INVERTIDO A PARTIR DE RESÍDUOS DE EQUIPAMENTOS ELETRÔNICOS

Trabalho de Conclusão de Curso apresentado à Faculdade de Talentos Humanos – FACTHUS, como requisito parcial de obtenção de título de bacharel em Engenharia Mecânica.
Orientador: Esp. Cleiton Silvano Goulart

ÁREA DE CONCENTRAÇÃO: Controle de Sistemas Mecânicos

Uberaba, 8 de dezembro de 2017.

BANCA EXAMINADORA:

_____

Prof. Esp. Cleiton Silvano Goulart – FACTHUS

_____

Prof. Esp. Antônio Carlos Lemos Junior - FACTHUS

UBERABA – MG
2017

CONSTRUÇÃO DE UM PÊNDULO INVERTIDO A PARTIR DE RESÍDUOS DE EQUIPAMENTOS ELETRÔNICOS

Rafael Faria Felix[1]; Cleiton Silvano Goulart[2]

[1] Faculdade de Talentos Humanos-FACTHUS, Uberaba (MG), Brasil, e-mail: rafaelfariafelix@gmail.com
[2] Faculdade de Talentos Humanos-FACTHUS, Uberaba (MG), Brasil, e-mail: cleiton.goulart@facthus.edu.br

RESUMO: Este artigo apresenta as etapas de construção de um pêndulo invertido a partir de peças adaptadas de uma impressora HP Deskjet 1000. Foi utilizada uma placa Arduino Due e aplicado um controlador proporcional integral derivativo (Controlador PID) para o controle e estabilização do sistema. É demonstrado também um método para a definição dos parâmetros do controlador e a resposta do conjunto aos parâmetros definidos. Concluiu-se que é possível a construção de um pêndulo invertido de baixo custo através da reutilização de componentes eletrônicos descartados e que a placa Arduino Due atende satisfatoriamente aos requisitos propostos. O controlador PID apresentou bons resultados frente a pequenas perturbações, porém ao se deparar com variações maiores este não foi o suficiente para efetuar o controle do sistema.

PALAVRAS CHAVE: Arduino; Controle proporcional integral derivativo; Pêndulo invertido; PID; Reutilização.

CONSTRUCTION OF AN INVESTED PENDULUM FROM WASTE OF ELECTRONIC EQUIPMENT

ABSTRACT: This article presents the steps of constructing an inverted pendulum from pieces adapted from an HP Deskjet 1000 printer. An Arduino Due board was used and an integral proportional derivative controller (PID controller) was used to control and stabilize the system. It is also demonstrated a method for defining the parameters of the controller and the response of the set to the defined parameters. It was concluded that it is possible to construct a low cost inverted pendulum by reusing discarded electronic components and that the Arduino Due board satisfies the proposed requirements. The PID controller presented good results against small disturbances, but when faced with larger variations this was not enough to control the system.

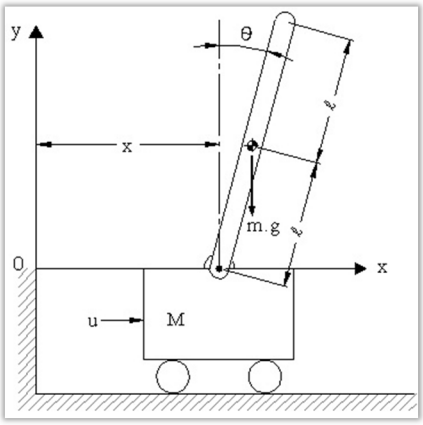
KEYWORDS: Arduino; Derivative proportional integral control; Inverted pendulum; PID; Reuse.

INTRODUÇÃO

Diversas aplicações reais podem ser simplificadas na forma do conceito de pêndulo invertido. Aplicações diretas deste princípio podem ser encontradas no mercado, na construção do Double[1], robô de teleconferência da Double Robotics, no veículo de transporte humano Segway[2], e outros veículos de eixo único, como o Elektor OSPV1[3]. Este sistema também possui aplicações mais complexas, como o controle de direção de um foguete durante o lançamento, sistemas de proteção de edifícios em caso de terremoto, ou o equilíbrio de robôs bípedes. (DA SILVA, 2012; DO PRADO, 2011; VENDRAMINI, 2010)

O pêndulo invertido é um sistema mecânico constituído de uma barra rígida que oscila sobre uma base móvel, fixada por um eixo em sua parte inferior. Essa base, por sua vez, se movimenta sobre um trilho com o auxílio de um pequeno motor. Esta construção possui dois graus de liberdade: o ângulo entre a barra e o plano vertical (θ); e a posição do carrinho em relação ao plano horizontal (x): conforme ilustrado na figura 1. Na posição vertical, a ação da gravidade faz com que a barra sofra um deslocamento angular em direção ao solo, caracterizando este como um sistema naturalmente instável. A solução do problema proposto pelo pêndulo invertido é utilizar um sistema de controle em malha fechada para movimentar o carrinho a fim de contrabalancear o movimento de queda da hasta, a fim de manter essa equilibrada na posição vertical. (DO PRADO, 2011)

Figura 1. Sistema pêndulo invertido



Fonte: autor, 2017

[1] Mais informações disponíveis em: <https://www.doublerobotics.com/>
[2] Mais informações disponíveis em: <http://www.segway.com/ >
[3] Mais informações disponíveis em: <https://www.elektormagazine.com/magazine/elektor-201106/19599>

No projeto de sistemas mecânicos móveis a autonomia é um aspecto muito importante. Estes devem ser independentes de uma fonte de energia fixa, como uma tomada ou gerador, e devem ter um processador dedicado que atenda aos requisitos de processamento. Segundo Chase (2007), desde as fases iniciais do projeto deve haver especial atenção ao tamanho, peso e consumo energético de cada componente, fatores que devem ser os menores possível para garantir a mobilidade do conjunto. Para atender a estes requisitos, uma possível solução é a utilização das placas Arduino que, por possuírem hardware de código livre (open source), são de fácil acesso e baixo custo. O Arduino Due é um microcontrolador dotado de um processador ARM de 32 bits e possui 512 KB de Memória Flash. Comparado com seus predecessores, esta versão possui um poder de processamento e armazenamento de instruções muito maior, além de outros aspectos que o tornam ideal para sistemas complexos e que necessitam de operações em tempo real. (ARDUINO, 2017).

Diante disso, este artigo apresenta uma análise da aplicabilidade do Arduino Due como alternativa de baixo custo para efetuar o processamento e controle em sistemas mecânicos que utilizem o princípio do pêndulo invertido. Para tal, foi construído um pêndulo invertido sobre trilhos para se avaliar a eficácia do Arduino Due nesta situação.

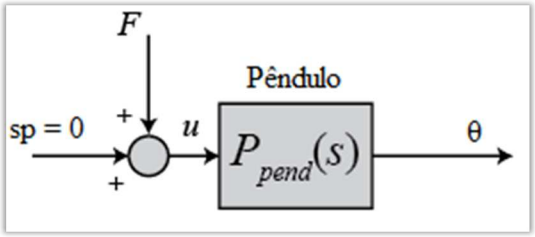
MATERIAL E MÉTODOS

Segundo Ogata (2010, p. 60), a variação do ângulo de um pêndulo invertido em relação às forças externas aplicadas a este pode ser definida de acordo com a equação 1, sendo C_1 e C_2 constantes determinadas de acordo com as propriedades físicas da construção. O diagrama de blocos deste sistema é representado na figura 2, em que $P_{pend}(s)$ é a função de transferência da equação 1.

$$\frac{\theta_{(s)}}{U_{(s)}} = \frac{1}{C_1(s + C_2)(s - C_2)} \tag{1}$$

$$\text{Polos} = +C_2;\ -C_2 \tag{2}$$

Figura 2 - Diagrama de blocos de um pêndulo invertido em malha aberta



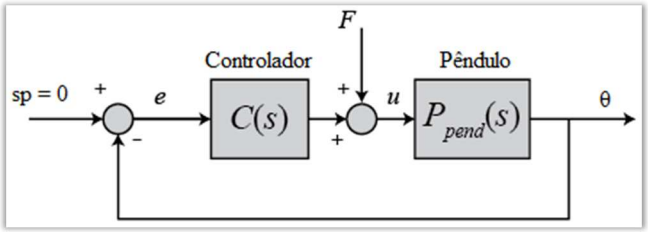
Fonte: autor, 2017

A partir da equação 1 observa-se que este sistema possui um polo no semieixo positivo do eixo real (evidenciado na equação 2), e isto demonstra que o sistema é instável em malha aberta. (OGATA, 2010)

Para a estabilização deste sistema faz-se necessário então a aplicação de alguma técnica de controle. Neste artigo foi escolhido a aplicação de um controlador Proporcional Integral Derivativo (Controlador PID) em malha fechada. Este controlador define a saída *u(t)* a partir de um erro *e(t)* através da equação 3, e as constantes K_p, K_i e K_d podem ser alteradas de forma a regular a resposta do sistema. O diagrama de blocos com o controlador aplicado é representado na figura 3.

$$u(t) = K_p e(t) + K_i \int_0^t e(t)dt + K_d \frac{de(t)}{dt} \tag{3}$$

Figura 3 - Diagrama de blocos de um pêndulo invertido e controlador em malha fechada

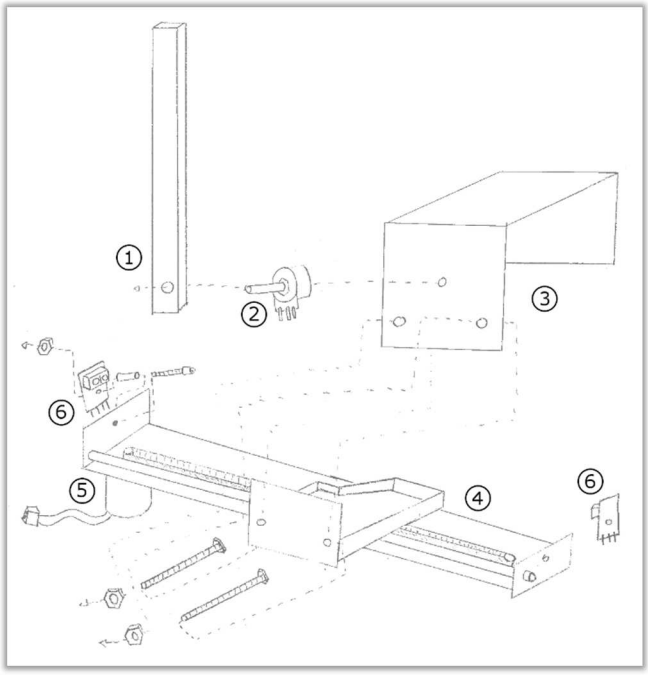


Fonte: autor, 2017

A construção do pêndulo invertido foi feita com base neste modelo matemático, conforme demonstrado na figura 4. Os materiais utilizados foram:

- Arduino Due
- Botão de pulso
- Chapa galvanizada
- Impressora HP Deskjet 1000
- Placa de papelão
- Potenciômetro 10kΩ
- Resistor 10kΩ
- Sensor ótico reflexivo TCRT5000

Figura 4 - Vista explodida da montagem mecânica do sistema. 1. Haste; 2. Potenciômetro; 3. Cobertura; 4. Trilho; 5. Motor DC; 6. Sensor ótico reflexivo TCRT5000.



Fonte: autor, 2017

Para a haste foi utilizado uma placa de papelão, dobrada em três camadas. Este material foi escolhido por ter um peso muito pequeno, apresentar uma boa resistência mecânica e ser de fácil manuseio. Na extremidade da haste foi fixado um pequeno peso de metal, para que o sistema tenha seu centro de gravidade concentrado na ponta superior, conforme proposto pelo modelo matemático utilizado. O carro foi adaptado a partir de partes de uma impressora HP Deskjet 1000[4]. Desta impressora foi utilizado o carro de impressão como base para a construção do carro do pêndulo invertido, bem como seu trilho, o motor de corrente contínua que movimenta este carro de impressão, a correia dentada que transmite o movimento, a fita codificada e o encoder ótico linear que faz a leitura da fita.

Figura 5 - Carro de impressão retirado da impressora

Fonte: autor, 2017

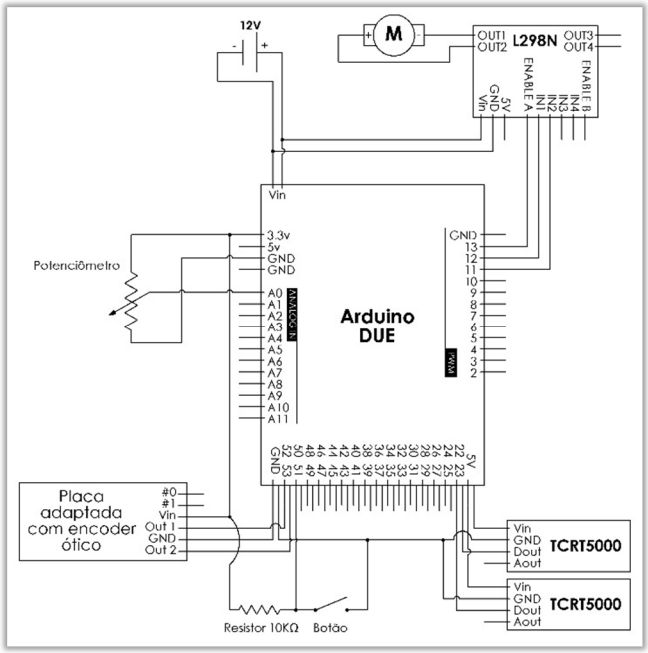
Para a leitura do ângulo da barra foi utilizado um potenciômetro rotativo devido à sua alta resolução e facilidade de utilização. A fixação deste componente na posição desejada foi obtida utilizando-se uma cobertura no carro de impressão, feita com chapa galvanizada dobrada em um formato cúbico, conforme apêndice A.

Em cada extremidade do trilho foi fixado um módulo sensor ótico TCRT5000 a fim de detectar quando o carro se aproxima do fim do curso, e assim evitar a colisão deste com as laterais da montagem.

O controle deste sistema foi implementado em uma placa Arduino Due, com o controle do motor efetuado com o intermédio de um driver ponte H LN298N, conforme diagrama elétrico demonstrado na figura 6.

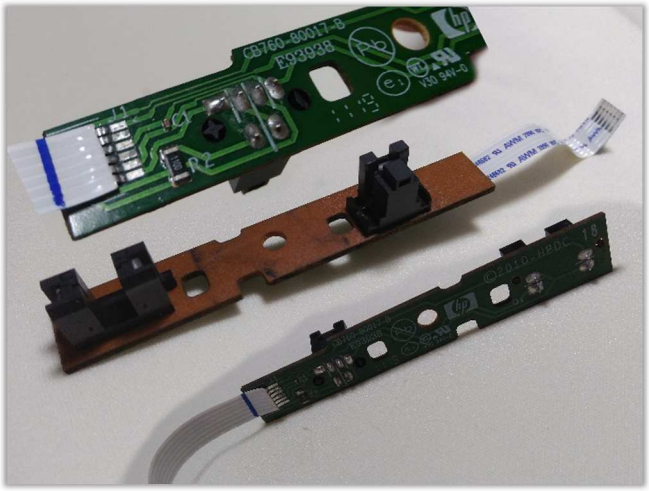
A fim de monitorar a posição do carro, a fita codificada foi mantida e um encoder ótico foi adaptado de outra parte da impressora. Este encoder estava acoplado ao rolete de alimentação do papel, fazendo a leitura de um disco codificado. Este componente foi escolhido ao invés daquele já existente nesta posição por estar montado em um circuito mais simples e visível, portanto sua utilização se mostrou mais fácil, conforme observado na figura 7. Nesta mesma placa estava acoplado outro sensor e, para a fixação da placa no local desejado, foi cortada a placa ao meio e o cabo flat dobrado, conforme visto nas figuras 8 e 9.

Figura 6 - Diagrama elétrico



Fonte: autor, 2017

Figura 7 - Placa retirada da impressora com encoder ótico e sensor de tampa fechada



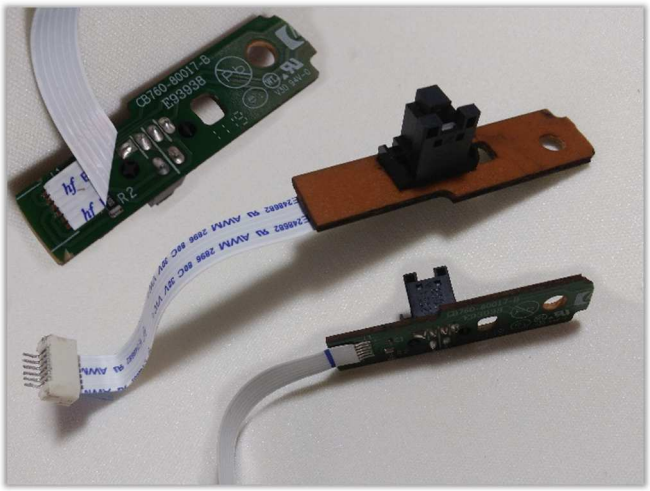
Fonte: autor, 2017

O sensor rotativo utilizado tem um ângulo de giro de 270°, portanto foi necessário implantar um limite físico no movimento da barra para evitar danos a este. Para tal, as garras do carro de impressão foram mantidas, e a posição de montagem do potenciômetro foi escolhida de forma a

[4] Esta impressora seria descartada por apresentar defeito, e foi gentilmente doada por uma empresa local de reparos de equipamentos eletrônicos.

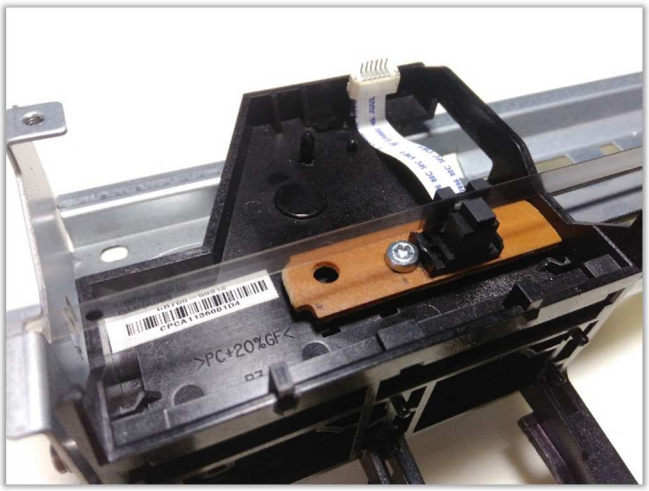
formar um ângulo de 270° com estas garras, conforme figura 10.

Figura 8 - Placa cortada para isolar o encoder ótico



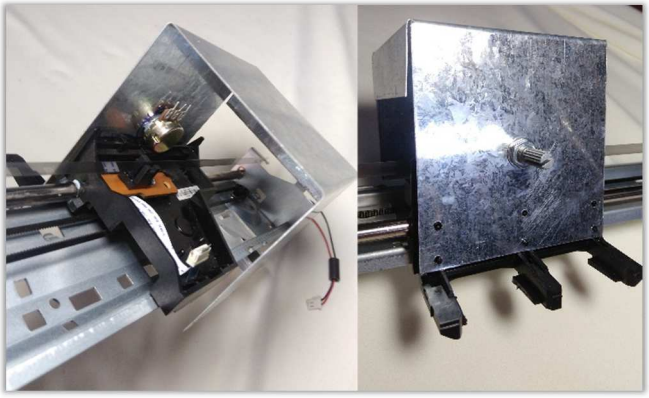
Fonte: autor, 2017

Figura 9 - Encoder adaptado montado no carro



Fonte: autor, 2017

Figura 10 - Cobertura de chapa galvanizada e potenciômetro



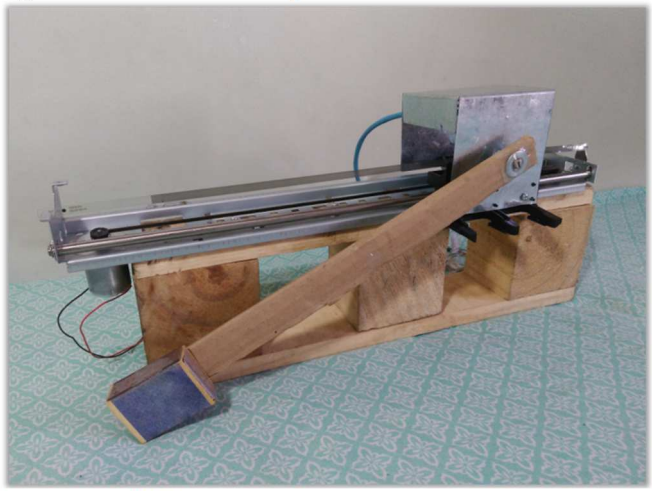
Fonte: autor, 2017

A barra foi construída de forma a minimizar seu peso e maximizar sua resistência. Em uma de suas extremidades foi fixada uma caixa de fósforo, onde colocou-se em seu interior pequenos parafusos para deslocar o centro de massa e regular o peso total da barra, conforme visto nas figuras 11 e 12.

Figura 11 - Peso próprio da barra: 12 gramas

Fonte: autor, 2017

Figura 12 - Barra montada no pêndulo invertido



Fonte: autor, 2017

A programação em C++ do Arduino pode ser conferida no apêndice B, e foi desenvolvida para seguir o seguinte procedimento de operação:

1. Ao ser ligado pela primeira vez, efetua-se o processo de centralização;
2. No processo de centralização o carro desloca-se até o fim de curso esquerdo, percorre todo o trilho contando os pulsos do encoder para encontrar o comprimento total do sistema, e então desloca-se para metade desta distância;
3. Após a centralização, uma luz pisca para indicar que o pêndulo está pronto para iniciar a operação;
4. O operador deve colocar manualmente a barra na posição vertical e apertar um botão para definir o ponto de equilíbrio a ser mantido pelo programa;
5. Durante a operação do sistema, a leitura do encoder é constantemente comparada com a leitura feita no momento que o botão de início foi pressionado, e a diferença é o "erro" a ser eliminado pelo controlador;

6. Um controlador PID[5] calcula a intensidade do sinal a ser enviado para os motores a partir do erro encontrado;
7. Caso o erro seja negativo, a barra está se movimentando no sentido anti-horário, e o carro deve se mover para a esquerda;
8. Caso o erro seja positivo, a barra está se movimentando no sentido horário, e o carro deve se mover para a direita;
9. Caso o erro seja nulo, a barra se encontra em equilíbrio, e o movimento do carro deve ser interrompido;
10. Se algum dos dois sensores de fim de curso for acionado, o movimento deve ser interrompido, e o processo de centralização efetuado novamente.

RESULTADOS E DISCUSSÃO

A sintonização do controlador PID foi feita conforme o método Ziegler-Nichols (ZIEGLER; NICHOLS, 1942). Este método foi escolhido por dispensar que se conheça a equação do sistema, sendo um método de calibragem empírico. Pra tal, aplicou-se apenas o controle proporcional, aumentando o valor deste até se obter uma oscilação estável, representada na figura 13. Os testes foram feitos aplicando-se uma força instantânea na barra e a resposta coletada através do software próprio do Arduino.

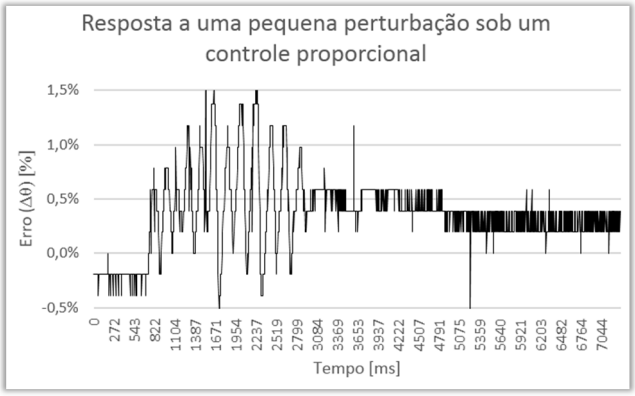
Figura 13 - Resposta do sistema ao ser aplicado um controle estritamente proporcional de intensidade 10 (parâmetros: $K_p = 10$, $K_i = 0$, $K_d = 0$)

Fonte: autor, 2017

Pela análise do gráfico, encontrou-se o período P_u de aproximadamente 0,07 segundos, quando a constante proporcional atinge seu valor crítico $S_u = 10$. De posse destes dois valores, testou-se a resposta do sistema a um controle estritamente proporcional, ao sintonizar o controlador de acordo com a equação 4. Os resultados obtidos são apresentados nas figuras 14 e 15.

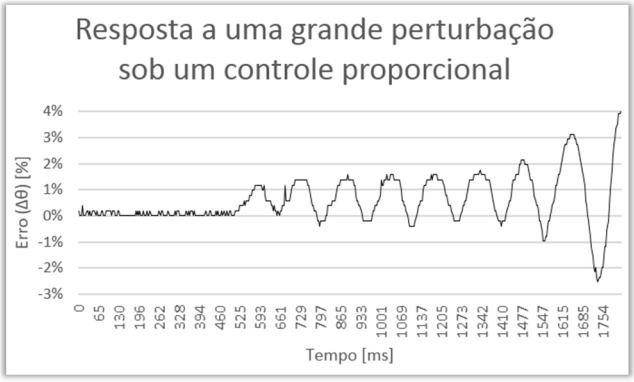
$$K_P = 0{,}5\, S_u \tag{4}$$

Figura 14 - Resposta do sistema a uma pequena perturbação sob um controle estritamente proporcional de intensidade 5 (parâmetros: $K_p = 5$; $K_i = 0$; $K_d = 0$)



Fonte: autor, 2017

Figura 15 - Resposta do sistema a uma grande perturbação sob um controle estritamente proporcional de intensidade 5 (parâmetros: $K_p = 5$; $K_i = 0$; $K_d = 0$)



Fonte: autor, 2017

Na figura 14 pode ser observado que ao se aplicar uma pequena perturbação o sistema estabiliza-se, porém em uma posição um pouco deslocada em relação à posição inicial. Enquanto na figura 15 foi aplicada uma perturbação maior, e a amplitude do movimento aumentou até que o sistema se desestabilizasse.

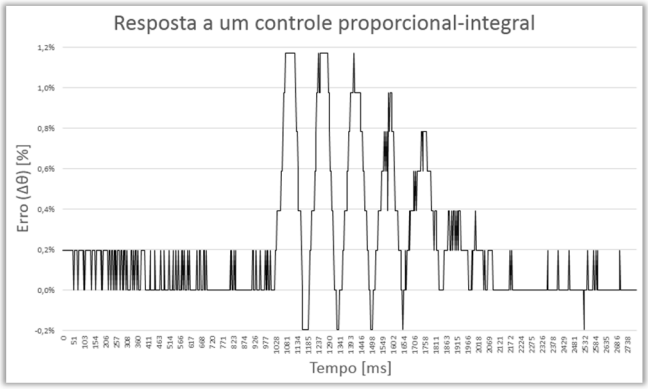
Para corrigir a variação observada na figura 14, utiliza-se a parte integral do controlador. Portanto, em seguida testou-se a resposta ao se aplicar um controlador proporcional e integral, apresentada na figura 16. Para a sintonização das constantes utilizou-se as equações 5 e 6.

$$K_P = 0{,}45\, S_u \tag{5}$$

$$K_i = \frac{P_u}{1{,}2} \tag{6}$$

[5] PID: Proporcional, Integral, Derivativo. Controlador onde a saída *u(t)* é definida a partir de um erro *e(t)* através da equação $u(t) = K_p e(t) + K_i \int_0^t e(t)dt + K_d \frac{de(t)}{dt}$, e as constantes K_p, K_i e K_d podem ser definidas para regular a resposta do sistema.

Figura 16 - Resposta do sistema a uma pequena perturbação sob um controle proporcional-integral de intensidade 4,5 e 0,058 respectivamente (parâmetros: K_p = 4,5; K_i = 0,058; K_d = 0)



Fonte: autor, 2017

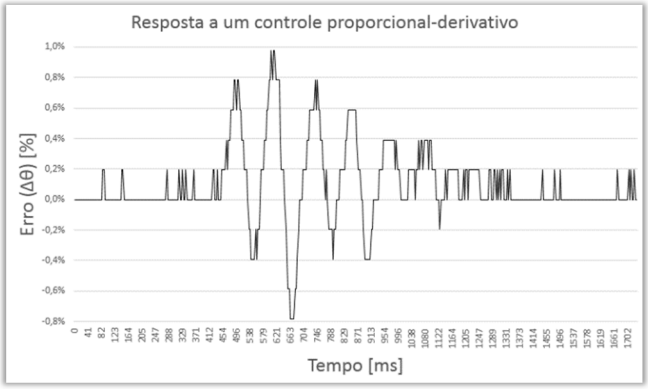
Ao se analisar o gráfico da resposta observa-se que o deslocamento do ponto de estabilização foi eliminado com êxito ao se aplicar a parte integral do controlador PID. Observa-se também que com estes parâmetros foi possível estabilizar uma perturbação de maior intensidade, que no controlador estritamente proporcional desestabilizou o sistema.

O próximo controlador testado foi o controlador proporcional-derivativo, o qual foi sintonizado de acordo com as equações 7 e 8. A parte derivativa do controlador PID analisa a taxa de variação do erro a fim de prever o comportamento deste e se adiantar a estas variações, sendo que na figura 17 podemos observar os resultados desta análise. Após a perturbação, o sistema oscilou um pouco e logo em seguida buscou o equilíbrio.

$$K_P = 0{,}8\, S_u \tag{7}$$

$$K_d = \frac{P_u}{8} \tag{8}$$

Figura 17 - Resposta do sistema a uma pequena perturbação sob um controle proporcional-derivativo de intensidade 8 e 0,00875 respectivamente (parâmetros: K_p = 8; K_i = 0; K_d = 0,00875)



Fonte: autor, 2017

Observa-se então que este controlador cumpre satisfatoriamente a função de estabilizar o sistema. Comparado com o controlador proporcional-integral, o controlador proporcional-derivativo se mostrou ser mais rápido, estabilizando o pêndulo invertido em aproximadamente 600 milissegundos, contra aproximadamente 1000 milissegundos do anterior.

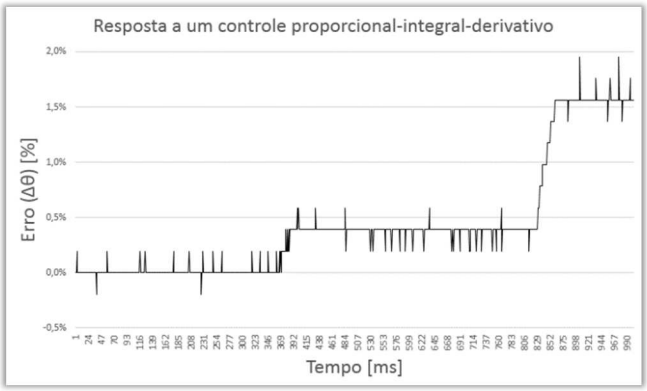
Testou-se também a resposta a uma controlador proporcional-integral-derivativo completo, de acordo com as equações 9, 10 e 11. Conforme demonstrado na figura 18, este controlador não foi capaz de estabilizar o sistema. Nota-se então que a utilização de maior quantidade de parâmetros não é garantia de maior desempenho, para cada aplicação é necessário analisar qual conjunto de parâmetros é mais efetivo no controle do sistema.

$$K_P = 0{,}6\, S_u \tag{9}$$

$$K_i = \frac{P_u}{2} \tag{10}$$

$$K_d = \frac{P_u}{8} \tag{11}$$

Figura 18 - Resposta do sistema sob um controle proporcional-integral-derivativo de intensidade 6, 0,035 e 0,00875 respectivamente (parâmetros: K_p = 6; K_i = 0.035; K_d = 0.00875)



Fonte: autor, 2017

Para a aplicação de um controlador PID é necessário que o sistema seja definido por um modelo linear. Como o sistema do pêndulo invertido depende do ângulo da barra, isso faz com que seu modelo não seja linear. Pode-se, porém, linearizar este modelo aproximando sen $\theta \approx \theta$ e cos $\theta \approx 1$ para valores de θ muito pequenos (OGATA, 2010, p. 60). Isso se mostra verdadeiro ao se analisar os resultados obtidos, pois o sistema se comportou conforme o previsto para pequenas variações da posição da barra. Porém, ao deparar-se com maiores perturbações, os parâmetros definidos não atenderam aos requisitos.

O motor utilizado adquire força suficiente para mover o conjunto apenas quando alimentado com uma tensão superior a 3,2V. Com as configurações padrões do Arduino, esta tensão é atingida com um valor de PWM[6]

[6] *Pulse-Width Modulation* (modulação por largura de pulso)

igual a 140, sendo o valor máximo para este parâmetro igual a 255. Para contornar esta limitação do motor foi definido um valor mínimo para o PWM. Porém, este limite deixa pouco espaço para o controle de velocidade do carro, principalmente quando a constante proporcional do controlador é definida com valores altos. Para leitura de entradas analógicas e escrita de saídas em PWM, a placa Arduino Due utiliza uma resolução de 10bits e 8bits, respectivamente, para manter compatibilidade com códigos escritos para outras placas. Mas esta placa possui duas funções, *analogReadResolution()* e *analogWriteResolution()*, que possibilitam mudar estas resoluções para 12bits, aumentando o valor máximo de leitura de 1023 para 4095, e o de escrita de 255 para 4095. Esta mudança de resolução aumenta a precisão dos dados, melhorando significativamente o desempenho geral do sistema.

Diferente de situações simuladas, aplicações reais de um controlador apresentam muito ruído, assim como visto nas figuras 13 a 17. Isso interfere na atuação deste controlador e dificulta a estabilização do sistema.

CONCLUSÃO

A aplicação do controlador PID apresenta bons resultados quando a variação do ângulo da barra é mínimo, porém não é a melhor opção quando necessário controlar variações maiores. Entretanto, a aplicação deste controlador foi suficiente para o objetivo proposto de testar a aplicabilidade da placa Arduino Due para a estabilização do pêndulo invertido. Esta placa atende aos requisitos por apresentar alta frequência de processamento e ser de fácil utilização. Tanto sua conexão física entre o hardware da placa e os componentes do sistema, quanto a sua programação são simples e possuem extensa documentação de fácil acesso.

É necessário salientar que o método de sintonização Ziegler–Nichols é um método heurístico. Ou seja, este método faz algumas suposições e ignora alguns dados a fim de prover uma resposta rápida e prática. Portanto, este método é indicado para se obter um valor aceitável dos parâmetros quando não se conhece a equação do sistema, e melhores resultados são obtidos com o subsequente ajuste destes parâmetros. Neste artigo foram utilizados os parâmetros obtidos através destas equações sem posterior ajuste a fim de apresentar uma análise da eficiência deste método.

Conclui-se que é possível a construção de um pêndulo invertido com o mínimo de custo utilizando-se resíduos de equipamentos eletrônicos. Esta construção apresenta um desempenho satisfatório, e pode ser controlada por uma placa Arduino.

Para futuros trabalhos recomenda-se que seja utilizada a função *analogWriteResolution()* para adequar o controle PWM a um nível de melhor capacidade de atuação, e a função *analogReadResolution()* para aumentar a sensibilidade do potenciômetro. Sugere-se que sejam feitas simulações computacionais utilizando para o controlador PID os mesmos parâmetros sintonizados neste artigo a fim de comparar os dados obtidos. Propõe-se também que seja aplicada alguma técnica de tratamento de sinal com o intuído de minimizar o ruído na leitura do potenciômetro.

REFERÊNCIAS

ARDUINO. Disponível em: <https://www.arduino.cc/>. Acesso em: 04 set. 2017.

CHASE, Otavio; ALMEIDA, F. J. **Sistemas embarcados. Mídia Eletrônica. Página na internet: <www.sbajovem.org/chase>**, v. 10, n. 11, 2007.

DA SILVA, Ricardo Teixeira. **Melhorias no kit educacional pêndulo invertido montado com REEE**. Recife: Escola Politécnica de Pernambuco, 2012.

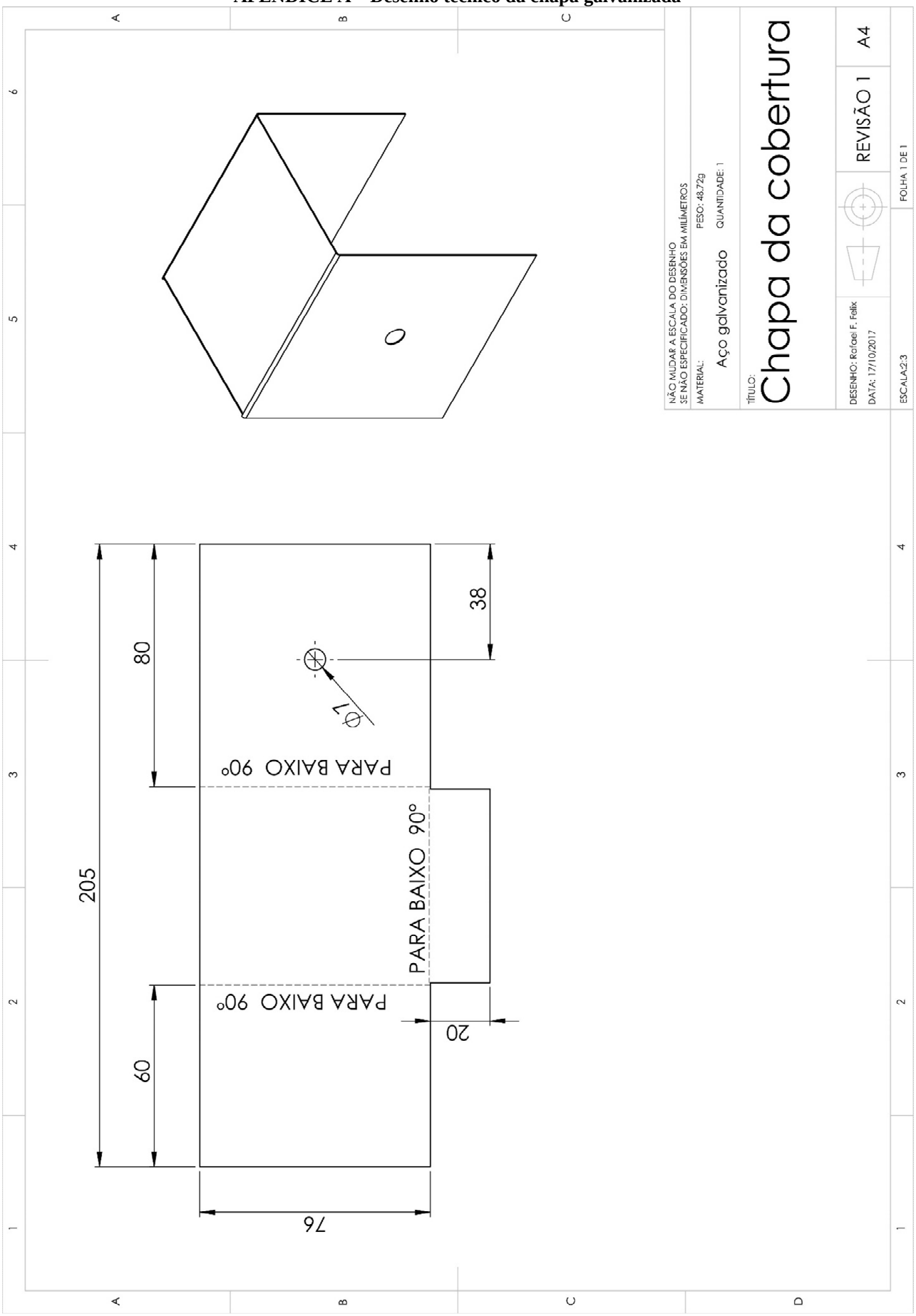
DO PRADO, Igor Ferreira. **Construção e controle do sistema pêndulo invertido**. Vitória da Conquista: Instituto Federal da Bahia, 2011.

OGATA, Katsuhiko. **Engenharia de controle moderno**. São Paulo: Pearson Prentice Hall, 5. ed., 2010.

VENDRAMINI, Gabriel; DA SILVA, Paulo Sérgio. **Controle de um pêndulo invertido sobre uma plataforma móvel utilizando PID e MFAC (Model-Free Adaptive Control)**. Serra Negra, SP: 9th Brazilian Conference on Dynamics Control and their Applications, 2010.

ZIEGLER, John G.; NICHOLS, Nathaniel B.. **Optimum settings for automatic controllers**. trans. ASME 64.11 (1942).

APÊNDICE A – Desenho técnico da chapa galvanizada



APÊNDICE B – Código em C++ do Arduino

```
//*****************************************************************//
// Código Arduino em C++ para controle de um pêndulo invertido     //
// Autor: Rafael Faria Felix                                       //
// Faculdade de Talentos Humanos – Uberaba, MG – Engenharia Mecânica //
//*****************************************************************//

//entradas
#define POT A0 //leitura do potenciômetro
#define BTN 52 //leitura do botão
#define ENC 53 //leitura do encoder linear
#define FCE 22 //leitura do fim de curso da esquerda
#define FCD 23 //leitura do fim de curso da direita

//saídas
#define MD1 11 //gira o motor no sentido 1
#define MD2 12 //gira o motor no sentido 2
#define MSP 13 //define a velocidade de giro do motor (PWM)
#define LUZ LED_BUILTIN //LED indicador de estado

//ajuste
#define PID_P 1
#define PID_I 0
#define PID_D 0

//limites
#define VCAL 255 //velocidade de calibração
#define VMIN 140 //velocidade mínima
#define VMAX 255 //velocidade máxima
#define TPOS 2 //tolerância do contador ao encontrar o fim do trilho
#define TANG 1 //tolerância do potenciômetro antes de tentar equilibrar a barra
#define TLD 500 //intervalo de tempo em que o LED pisca

#include <PID_v1.h>

double in, out, sp, input, output, setpoint; //variáveis do PID
char dir = 'p'; //direção do movimento: p-parado; e-esquerda; d-direita
volatile int pos = 0; //posição do carro
volatile int fim_trilho = 0; //tamanho do trilho (em pulsos do encoder)
PID pendulo(&in,&out,&sp,PID_P,PID_I,PID_D,DIRECT);

void setup() {
      Serial.begin(115200);

      pinMode(BTN, INPUT);
      pinMode(ENC, INPUT);
      pinMode(FCE, INPUT);
      pinMode(FCD, INPUT);
      pinMode(MD1, OUTPUT);
      pinMode(MD2, OUTPUT);
      pinMode(MSP, OUTPUT);
      pinMode(LUZ, OUTPUT);

      sp = 0;
      pendulo.SetSampleTime(20);
      pendulo.SetMode(AUTOMATIC);
      pendulo.SetOutputLimits(-255,255);

      attachInterrupt(digitalPinToInterrupt(ENC), encoder, FALLING);
      centraliza();
}
```

```
61
62  void encoder() {
63        if(dir == 'e') {
64              pos -= 1; //diminui o contador caso o movimento seja à esquerda
65        } else {
66              pos += 1; //aumenta o contador caso o movimento seja à direita
67        }
68        if(fim_trilho != 0 && (pos < -TPOS || pos > (fim_trilho + TPOS))) {
69              //recalibrar caso haja divergência nas informações de posição
70              fim_trilho = 0;
71              centraliza();
72        }
73  }
74
75  void mover(int velocidade) {
76        if(dir == 'e') { //mover à esquerda
77              digitalWrite(MD1,LOW);
78              digitalWrite(MD2,HIGH);
79        } else if(dir == 'd') { //mover à direita
80              digitalWrite(MD1,HIGH);
81              digitalWrite(MD2,LOW);
82        } else { //parar
83              digitalWrite(MD1,LOW);
84              digitalWrite(MD2,LOW);
85        }
86        analogWrite(MSP,velocidade);
87  }
88
89  void centraliza() {
90
91        //limpa interruptores dos sensores de fim de curso caso haja algum
92        detachInterrupt(digitalPinToInterrupt(FCE));
93        detachInterrupt(digitalPinToInterrupt(FCD));
94
95        if(fim_trilho == 0) { //caso não tenha sido calculado o fim do trilho ainda
96
97              //se não estiver à esquerda, mover à esquerda
98              int fce = digitalRead(FCE);
99              if(fce == 0) {
100                   dir = 'e'; //direção: esquerda
101                   mover(VCAL);
102                   while(fce == 0) {
103                         fce = digitalRead(FCE);
104                   }
105             }
106
107             //mover à direita e contar os pulsos do encoder
108             pos = 0;
109             dir = 'd';
110             mover(VCAL);
111             int fcd = digitalRead(FCD);
112             while(fcd == 0) {
113                   fcd = digitalRead(FCD);
114             }
115
116             fim_trilho = pos; //determinar o final do trilho
117       }
118
119       //se à esquerda do trilho, mover à direita e vice-versa
120       if(pos <= int(fim_trilho / 2)) {
121             dir = 'd';
```

```
122         } else {
123               dir = 'e';
124         }
125
126         mover(VCAL); //mover até o meio do trilho
127         while(pos != int(fim_trilho / 2)); //espera
128
129         //aguarda até que o botão seja pressionado e pisca o led
130         bool aguarda = true;
131         int tempoAnterior = millis();
132         int botao = digitalRead(BTN);
133         while(aguarda) {
134               delay(20); //debounce
135               if(digitalRead(BTN) && botao) {
136                     aguarda = false;
137               } else {
138                     botao = digitalRead(BTN);
139               }
140               if(millis() – tempoAnterior >= TLD) {
141                     digitalWrite(LUZ, !digitalRead(LUZ));
142                     tempoAnterior = millis();
143               }
144         }
145
146         setpoint = analogRead(POT);
147
148         //volta ao centro caso chegue ao final do trilho
149         attachInterrupt(digitalPinToInterrupt(FCE), centraliza, RISING);
150         attachInterrupt(digitalPinToInterrupt(FCD), centraliza, RISING);
151   }
152
153   void loop() {
154
155         input = analogRead(POT);
156         int val = setpoint - input;
157         in = val;
158         pendulo.Compute();
159         if(abs(val) > TANG) {
160               if(out > 0) {
161                     dir = 'e'; //move à esquerda
162               } else {
163                     dir = 'd'; //move à direita
164               }
165               output=abs(out) + VMIN;
166               if(output > VMAX) {
167                     output = VMAX;
168               }
169               mover(output);
170         } else {
171               dir = 'p'; //parar
172               mover(0);
173         }
174
175         int tempo = millis();
176         Serial.print(tempo); Serial.print(";"); //tempo decorrido, em milissegundos
177         Serial.print(setpoint); Serial.print(";"); //setpoint do PID
178         Serial.print(in); Serial.print(";"); //entrada do PID
179         Serial.print(out); Serial.print(";"); //saída do PID
180         Serial.print(input); Serial.print(";"); //leitura do sensor
181         Serial.print(output); Serial.print(";"); //saída para o motor
182   }
```