CONTROLE DIGITAL APLICADO AO PÊNDULO INVERTIDO COM GIROSCÓPIO

RAPHAEL TOMÉ SANTANA[1]; JOSÉ CLÁUDIO DE SOUZA JR[2]; MARCELO ESPOSITO[3]

[1]*Universidade Federal de Pelotas – raphaelts3 @gmail.com*
[2]*Universidade Federal de Pelotas – jclaudiodsj@gmail.com*
[3]*Universidade Federal de Pelotas – marcelo.esposito@ufpel.edu.br*

1. INTRODUÇÃO

Um dos sensores largamente utilizados no mercado para detectar o movimento angular, inclusive nos *smartphones*, é o giroscópio. Com ele é possível medir a velocidade angular de um movimento. A partir disto, é possível criar uma vasta quantidade de aplicações, um exemplo de aplicação do giroscópio é o controle do vídeo game Wii (NINTENDO, 2013) ou o aplicativo Labyrinth (GOOGLE, 2012) para o sistema Android. Outra interessante aplicação para os giroscópios são os sistemas de equilíbrio baseados no modelo de um pêndulo invertido. Na Figura 1 (a) é apresentado o modelo físico de um pêndulo invertido.

(a) (b)

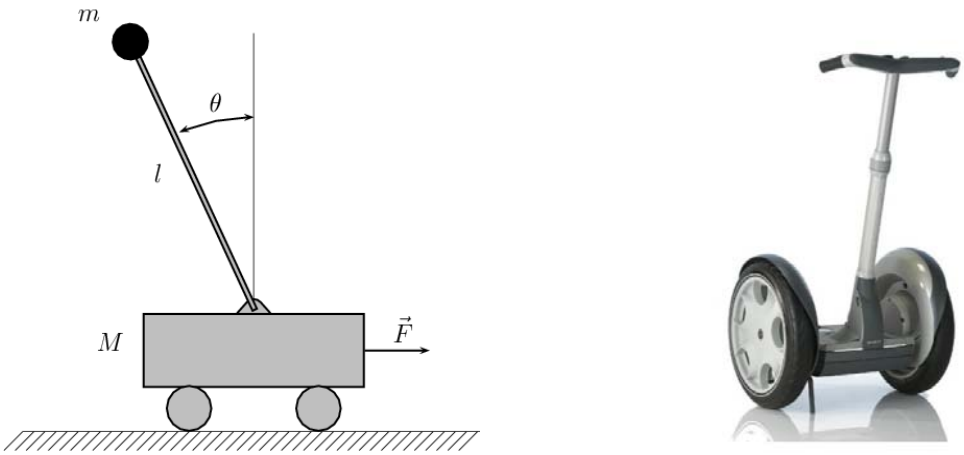

Figura 1 - (a) Modelo físico do pêndulo invertido. (b) *Segway Personal Transporter*

A estrutura do pêndulo invertido é composta por uma base móvel e uma massa conectada a base móvel através de uma haste rígida. A ideia é que a massa fique equilibrada no alto através da movimentação da base. O produto mais famoso que implementa um pêndulo invertido é o *Segway Personal Transporter*, apresentado na Figura 1 (b). Idealizado na década de 90, o *Segway* é utilizado para a mobilidade urbana.

Para o controle do pêndulo invertido, o *Segway* utiliza uma unidade de medição inercial constituída por giroscópios e sensores de nível que detectam a inclinação da plataforma onde fica posicionado o passageiro. Para se movimentar com o *Segway*, basta que o passageiro incline-se para a direção desejada. A partir do módulo de controle e dos sensores, os motores são acionados para promover o deslocamento do ocupante ou para manter o equilíbrio.

Este trabalho tem como objetivo aplicar os conceitos de controle digital no desenvolvimento de um módulo de controle para o equilíbrio de um pêndulo invertido.

2. METODOLOGIA

Neste trabalho foram utilizados dois giroscópios, o L3G4200D da ST (ST, 2010) e o MPU6050 da Invensense (INVENSENSE, 2011), além de um kit LEGO NXT que proveu a unidade de controle programável, a estrutura física e os atuadores. A comunicação entre os equipamentos foi realizada utilizando protocolo I2C, que utiliza duas vias de comunicação, o sinal de *clock* (SCL) e o sinal de dados (SDA).

O pêndulo invertido possui uma estrutura física especifica que deve ser suficientemente rígida para não alterar o centro de massa do sistema, o que dificultaria o controle na posição vertical. O pêndulo montado neste trabalho pode ser visto na Figura 2, onde a base móvel do pêndulo invertido foi reproduzida com o uso de dois servomotores do kit LEGO NXT 2.0. A leitura dos *encoders,* também foi utilizada no controle em malha fechada. Os *encoders* fornecem o deslocamento resultante da ação dos atuadores (servomotores).

A unidade de processamento utilizada foi a *NXT Intelligent Brick*, juntamente com o ambiente de desenvolvimento de programação *Bricx Command Center (BricxCC) Build 3.3.8.10* (BRICXCC, 2014) e com o *driver* específico para Windows 7/8 64 Bits. A linguagem de programação utilizada foi a NXC (*Not eXactly C*).

Figura 2 - Estrutura física do pêndulo invertido.

A primeira etapa do projeto foi a definição e teste da comunicação do giroscópio L3G4200D com o *Brick*. Efetuou-se a adequação da tensão de alimentação de 4,6V, do *Brick*, para aproximadamente 3,3V do sensor, utilizando dois diodos em série. Dois resistores de 10kΩ foram utilizados como resistores de *pull-up,* para a comunicação. A partir do trabalho desenvolvido por GROTKASTEN (2012), foi obtido o projeto do controlador. O giroscópio utilizado pelo autor foi o MPU6050 e considerações foram feitas com relação a tensão de alimentação e resistores de *pull-up*. Destaca-se que os endereços dos registradores do MPU6050 são diferentes dos endereços do L3G4200D.

3. RESULTADOS E DISCUSSÃO

A partir da reprodução dos resultados obtidos por GROTKASTEN (2012), com o equilíbrio do pêndulo invertido, foi possível identificar um modelo empírico para o caso estudado.

Com o objetivo de verificar a diferença de leitura obtida entre os sensores (L3G4200D e MPU6050), elaborou-se um experimento baseado no movimento de um pêndulo convencional. O robô foi posicionado na posição horizontal e então liberado para atingir o equilíbrio na posição vertical. O *Brick* foi programado para salvar, em formato binário, todas as amostras lidas com o sensor MPU6050, com a taxa de 2000 dps (graus por segundo). Posteriormente o mesmo experimento foi realizado com o L3G4200D utilizando uma taxa de 250 dps, 500 dps, 1000 dps e 2000 dps.

A fim de visualizar e identificar os endereços para a comunicação I2C foi utilizado um analisador lógico. Além disso, programas específicos, em linguagem NXC, foram criados para a configuração dos sensores nas diferentes taxas de amostragem. Para a correta interpretação das informações gravadas no arquivo binário, foi desenvolvido um programa, em linguagem C, para converter as amostras para o formato texto. Estes experimentos foram fundamentais, uma vez que permitiram identificar a perda de informações, sobre o movimento do pêndulo, com amostragens abaixo de 2000 dps.

As Figuras 3 e 4 apresentam partes do controlador digital implementado em linguagem NXC utilizando o giroscópio MPU6050 e o L3G4200D, respectivamente. Inicialmente são escritos os endereçamentos I2C para os diferentes sensores (em negrito) e abaixo a inicialização das variáveis, que neste caso tem relação direta com os ganhos do controlador proporcional e derivativo (PD) utilizado. Embora o *hardware* Lego NXT seja praticamente o mesmo nos dois casos, as diferenças nos valores das variáveis do controlador PD (em vermelho) deve-se ao uso de giroscópios diferentes e com isso, a necessidade de se considerar a especificidade da sintonia de cada controlador.

```
#define PORT S2
#define Adresse (0x68<<1)
byte init1[] = {Adresse, 0x6B, 0x80};
byte init2[] = {Adresse, 0x6B, 0x03};
byte init3[] = {Adresse, 0x1A, 0x00};
byte init4[] = {Adresse, 0x1B, 0x18};
byte read_gyro[] = {Adresse, 0x43};
int angle_speed0= -18;
int angle_speed=0;
int angle_absolute=0;
float angle_P_factor=1.2;
float angle_D_factor=10;
int last_angle_error=0;
int route=0;
float route_P_factor=0.1;
float route_D_factor=25;
int last_route_error=0;
...
angle_speed = (((data[0]<<8) | data[1])) - angle_speed0;
angle_speed/=50;
...
        angle_absolute  += angle_speed;
        int error_angle =  0 - angle_absolute;
        int angle_PTerm = error_angle * angle_P_factor;
        int angle_DTerm = angle_speed * angle_D_factor;
        route = (MotorRotationCount(OUT_B));
        int error_route = 0 - route;
        int route_PTerm = error_route * route_P_factor;
        int route_DTerm  = (error_route - last_route_error) * route_D_factor;
        last_route_error = error_route;
        int PD = angle_PTerm - angle_DTerm - route_PTerm - route_DTerm;
        if (PD >100) {PD = 100;}
        if (PD < -100) {PD = -100;}
        OnFwd(OUT_BC, PD);
```

Figura 3 – Algoritmo de GROTKASTEN (2012) para o sensor MPU6050.

```
#define PORT S2
#define Address (0x69<<1)
byte init1[] = {Address, 0x20, 0x09};
byte init2[] = {Address, 0x23, 0x30};
byte read_gyro_OUT_X_L [] = {Address, 0x28};
byte read_gyro_OUT_X_H [] = {Address, 0x29};
int read_gyro=0;
int angle_speed0= -1;
int angle_speed=0;
int angle_absolute=0;
float angle_P_factor=1.164;
float angle_D_factor=11.8;
int last_angle_error=0;
int route=0;
float route_P_factor=0.217;
float route_D_factor=22;
int last_route_error=0;
...
angle_speed = (((data[1]<<8) | data[0])) - angle_speed0;
angle_speed /= -50;
...
        angle_absolute  += angle_speed;
        int error_angle =  0 - angle_absolute;
        int angle_PTerm = error_angle * angle_P_factor;
        int angle_DTerm = angle_speed * angle_D_factor;
        route = (MotorRotationCount(OUT_B));
        int error_route = 0 - route;
        int route_PTerm = error_route * route_P_factor;
        int route_DTerm  = (error_route - last_route_error) * route_D_factor;
        last_route_error = error_route;
        int PD = angle_PTerm - angle_DTerm - route_PTerm - route_DTerm;
        if (PD > 100) {PD = 100;}
        if (PD < -100) {PD = -100;}
        OnFwd(OUT_C, PD*0.98);
        OnFwd(OUT_B, PD);
```

Figura 4 – Algoritmo para o sensor L3G4200D.

As partes de cor verde, nas Figuras 3 e 4, referem-se à operação com os dados medidos com o giroscópio. O ganho proporcional é multiplicado pela diferença de posição com relação ao estado inicial desejado para o robô (posição vertical e velocidade angular igual à zero). O ganho derivativo é multiplicado pela velocidade angular medida. As partes na cor azul referem-se ao controle de posição obtido com o uso de servomotores. Os servomotores são, neste caso, motores de corrente contínua (CC) que possuem um *encoder* óptico acoplado ao eixo, para o monitoramento da velocidade e direção do motor CC. O ganho proporcional é multiplicado pelo erro obtido entre o número de rotações inicial e o valor atual medido. Como o objetivo é que o pêndulo fique em equilíbrio na vertical e sempre retorne a condição inicial, após a ocorrência de perturbações, o ganho derivativo é multiplicado pela diferença entre os erros no número de rotações atual e o anterior.

As análises de estabilidade do compensador digital em cascata foram realizadas de forma empírica, uma vez que não fez parte do escopo do presente trabalho a determinação de um modelo matemático do processo a controlar. O valor no instante *i* da variável PD representa a potência do motor aplicada (atuador) naquele instante, ou seja, o valor amostrado presente da saída do compensador. Questões de saturação foram levadas em consideração porque os valores de potência podem variar de -100% a 100% dependendo do sentido de rotação. Outro ponto chave é que ambos os motores recebem o mesmo comando, exceto pelo ajuste (de 0,98) aplicado a apenas um dos motores. Isso foi necessário para compensar outra não linearidade do sistema, a folga de engrenagens.

4. CONCLUSÕES

Este trabalho apresentou uma solução para o pêndulo invertido utilizando um controlador digital e um giroscópio. Com a implementação do controlador proporcional e derivativo e aplicando adequadamente o teorema da amostragem, foi possível construir um pêndulo invertido com sistema automático de equilíbrio.

5. REFERÊNCIAS BIBLIOGRÁFICAS

BRICXCC. **Bricx Command Center 3.3**. Acessado em maio 2014. Online. Disponível em: http://bricxcc.sourceforge.net/

ST. **L3G4200D.** 2010. Acessado em maio 2014. Online. Disponível em: http://www.st.com/st-web-ui/static/active/en/resource/technical/document/datasheet/CD00265057.pdf

INVENSENSE. **MPU-6000/6050.** 2011. Acessado em maio 2014. Online. Disponível em: http://www.invensense.com/mems/gyro/mpu6050.html

GROTKASTEN, R. **LEGO NXT Standalone (MPU6050, NXC)** 2012. Acessado em maio 2014. Online. Disponível em: http://robin-grotkasten.jimdo.com/elektronik-projekte/lego-nxt-standalone-mpu6050-nxc/

GOOGLE ILLUSION LABS **Labyrinth.** 2012. Acessado em maio 2014. Online. Disponível em: https://play.google.com/store/apps/details?id=se.illusionlabs.labyrinth.full&hl=en

NINTENDO. **WiiU**. 2013. Acessado em maio 2014. Online. Disponível em: http://www.nintendo.com/wiiu