Familiarização com o Equipamento

PTC 3471 – Práticas de Projeto de Sistemas de Controle
2º semestre de 2019
Bruno Angélico / Fuad Kassab Jr. / Diego Colón

Laboratório de Automação e Controle
Departamento de Engenharia de Telecomunicações e Controle
Escola Politécnica da Universidade de São Paulo

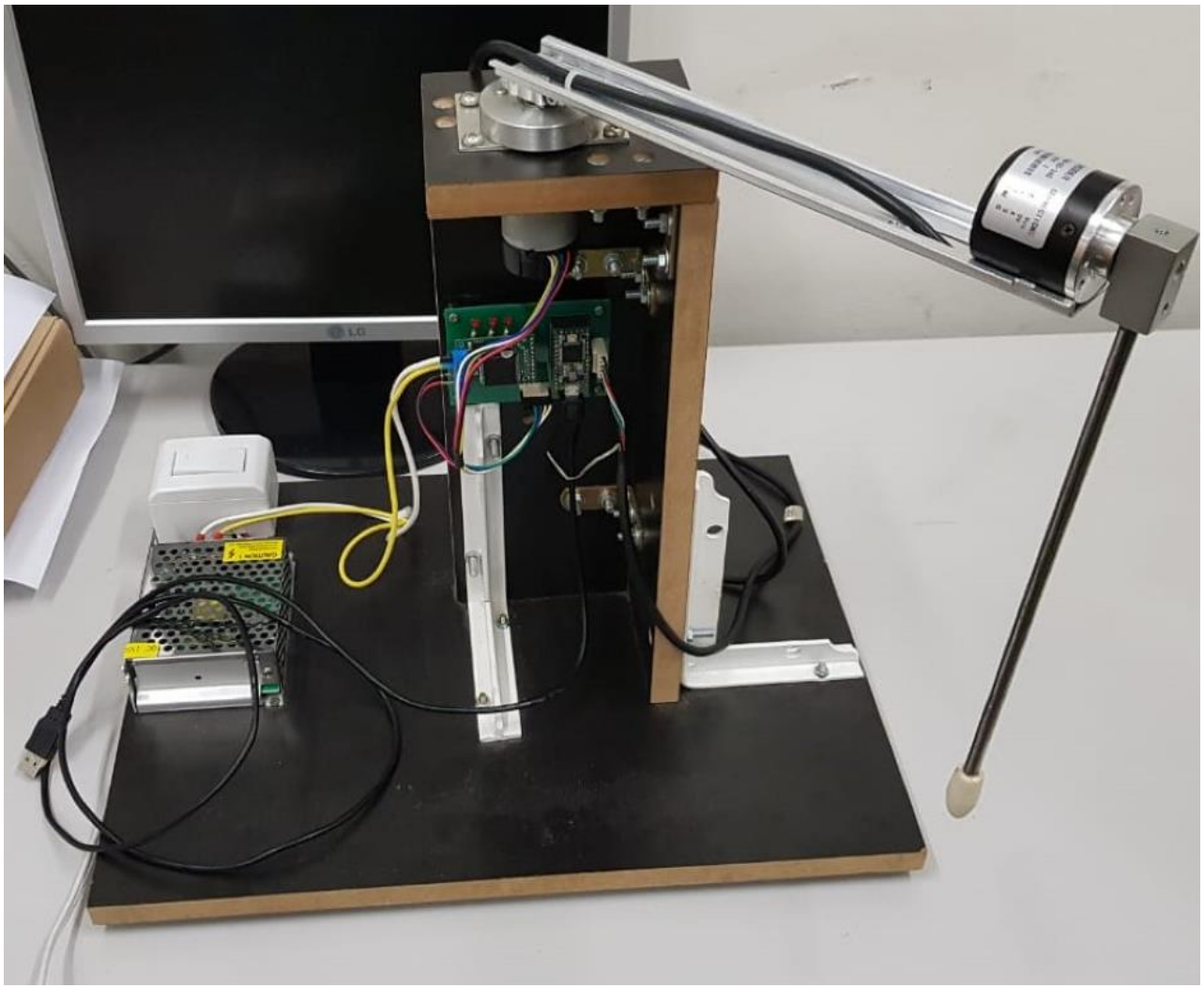
Introdução

- A disciplina PTC3471 - Práticas de Projeto de Sistemas de Controle foi criada para a Nova Estutura Curricular da Poli (EC3) aprovada em 25 de abril de 2013.
- É disciplina obrigatória da Engenharia Elétrica – Automação e Controle.
- Curso busca abordar aspectos de controle avançados, como modelagem de sistemas mecânicos, controle digital, controle ótimo e controle não linear, com enfoque prático.

Introdução

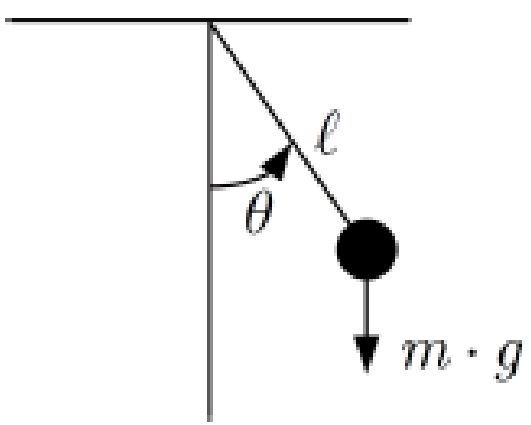
- Foram desenvolvidos kits didáticos do sistema pêndulo invertido rotacional.
- O Laboratório de Controle Aplicado (LCA) foi responsável pelo projeto e construção do pêndulo.

Introdução



O Pêndulo Simples e o Pêndulo Invertido

- Pêndulo simples:



- Período de oscilação do pêndulo depende apenas do comprimento da haste e não da massa, e também independe de θ, para valores pequenos desse ângulo.
- A equação diferencial do pêndulo simples é dada por:

$$\frac{d^2\theta}{dt^2} + \frac{g}{\ell}\sin(\theta) = 0$$

O Pêndulo Simples e o Pêndulo Invertido

- Para pequenos ângulos ($\sin(\theta) \sim \theta$), tem-se:

$$\frac{d^2\theta}{dt^2} + \omega^2\theta = 0\,, \qquad \omega = \sqrt{g/\ell}.$$

- Trata-se do oscilador harmônico simples, cuja solução geral é dada por:

$$\theta(t) = A\cos(\omega t) + B\sin(\omega t)$$

onde A e B são constantes determinadas pelas cond. Iniciais.

- O período de oscilação é dado por:

$$T = \frac{2\pi}{\omega} = 2\pi\sqrt{\frac{\ell}{g}}.$$

O Pêndulo Simples e o Pêndulo Invertido

- Para pequenos ângulos ($\sin(\theta) \sim \theta$), tem-se:

$$\frac{d^2\theta}{dt^2} + \omega^2\theta = 0\,, \qquad \omega = \sqrt{g/\ell}.$$

- Trata-se do oscilador harmônico simples, cuja solução geral é dada por:

$$\theta(t) = A\cos(\omega t) + B\sin(\omega t)$$

onde A e B são constantes determinadas pelas cond. Iniciais.

- O período de oscilação é dado por:

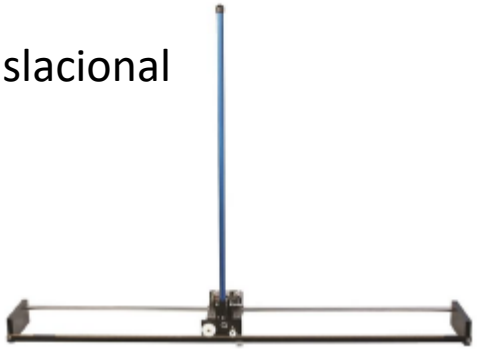
$$T = \frac{2\pi}{\omega} = 2\pi\sqrt{\frac{\ell}{g}}.$$

O Pêndulo Simples e o Pêndulo Invertido

- O pêndulo invertido é um dos problemas e controle mais utilizados com exemplo em textos didáticos de Engenharia de Controle.
- Trata-se de um sistema naturalmente instável.
- Há algumas variações de equipamentos didáticos do pêndulo invertido, como o pêndulo translacional, o pêndulo rotacional (ou de Furuta), pêndulo com roda de reação e o pêndulo com barra.

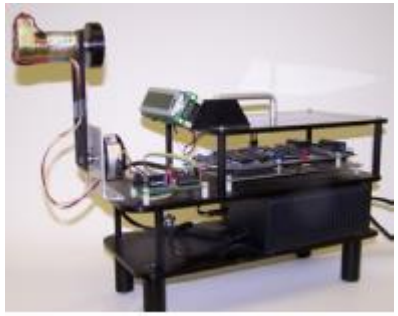
O Pêndulo Simples e o Pêndulo Invertido

Translacional



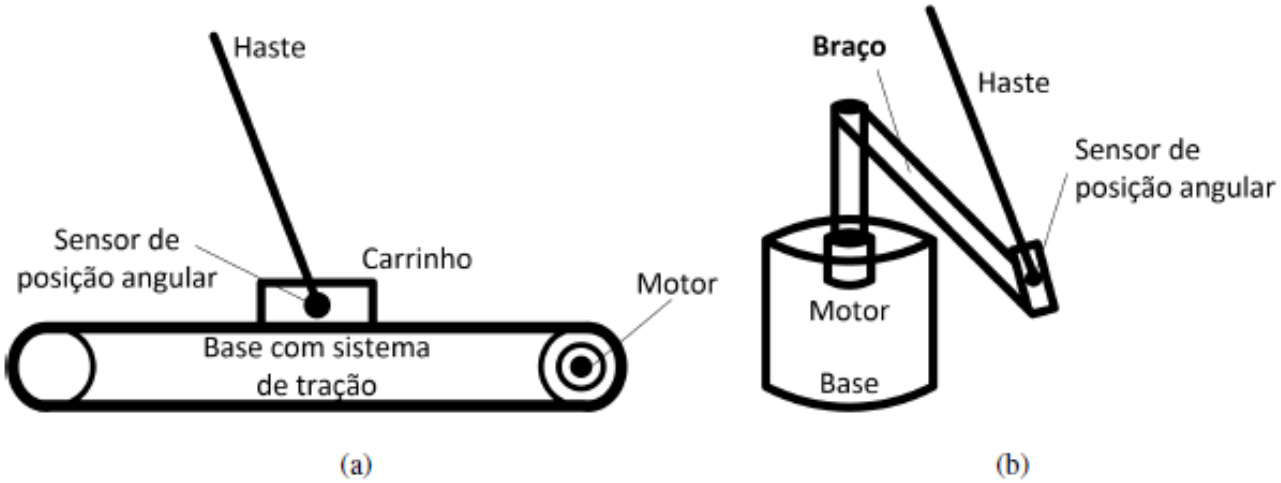
Rotacional

Com roda de reação



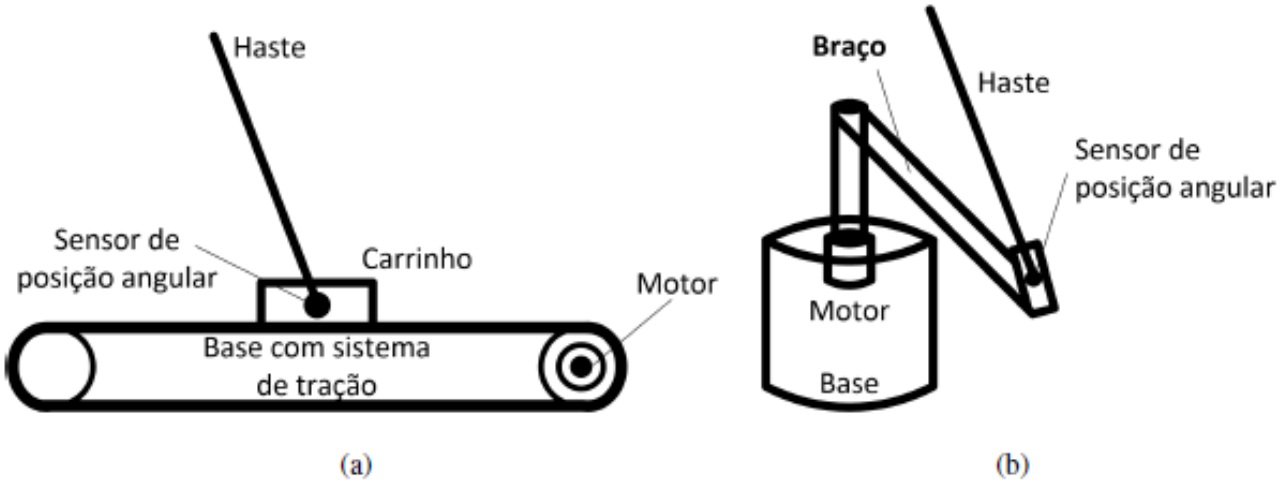
Com barra

O Pêndulo Simples e o Pêndulo Invertido



Diagramas esquemáticos: (a) pêndulo linear; (b) pêndulo rotacional.

O Pêndulo Simples e o Pêndulo Invertido



Diagramas esquemáticos: (a) pêndulo linear; (b) pêndulo rotacional.

Hardware e Software

O hardware consiste basicamente das seguintes partes:

- fonte de alimentação DC de 12V;
- chave liga/desliga;
- motor DC com redução e encoder;
- encoder;
- driver ponte-H;
- microcontrolador.

Hardware e Software

Motor:

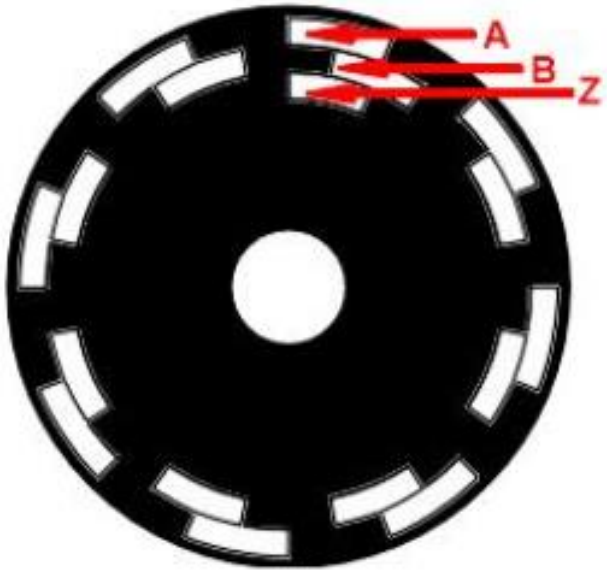
- 12 Vdc, fabricante Pololu.
- Possui uma caixa de redução com fator 18,75:1;
- Possui encoder com resolução de 16 ppr por canal.
- Considerando a redução, a resolução por canal é 300ppr.

Hardware e Software

Encoder:

- são sensores digitais para obtenção de posição angular;
- O sistema de leitura é geralmente óptico (pode ser por efeito Hall também), baseado em um disco formado por janelas radiais.
- Pode ser incremental ou absoluto.
- Os incrementais fornecem dois pulsos em quadratura, denominados canal A B. Alguns ainda possuem um canal Z.

Hardware e Software



https://www.automotsys.com.au/encodersmc.html

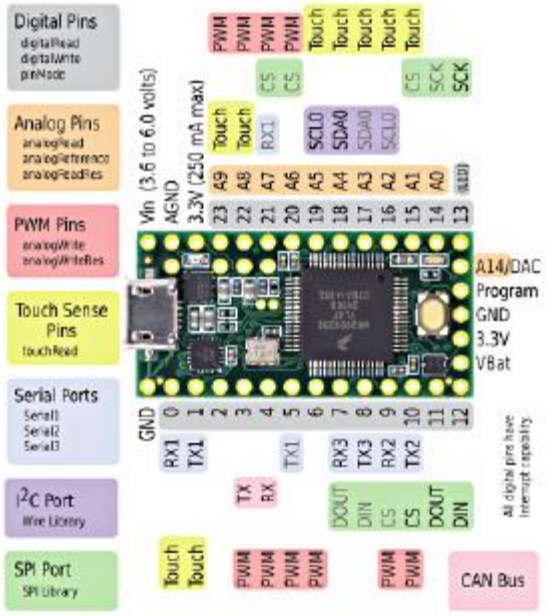
Hardware e Software

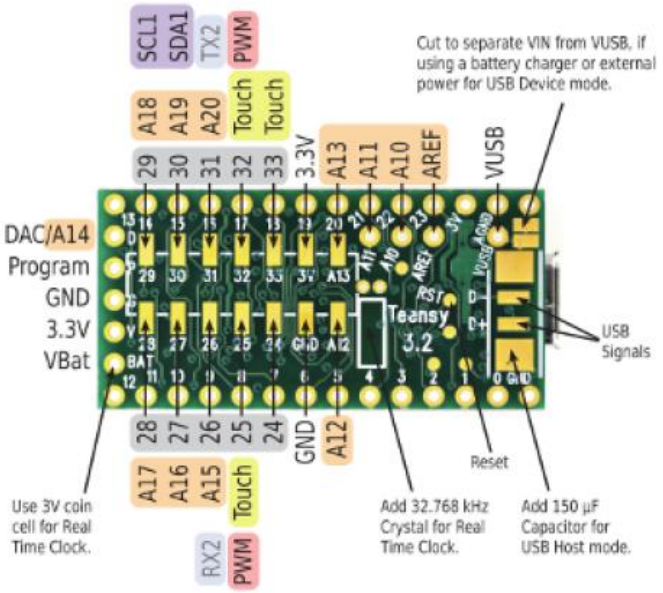
- O encoder do pêndulo já possui mancal de sustentação e tem resolução de 600 ppr por canal.

Hardware e Software

Microcontrolador:

- Teensy 3.2: processador Cortex-M4, 32 bits, com 72 Mhz de *clock*, memória flash de 256 kbytes, memória RAM de 64 kbytes;

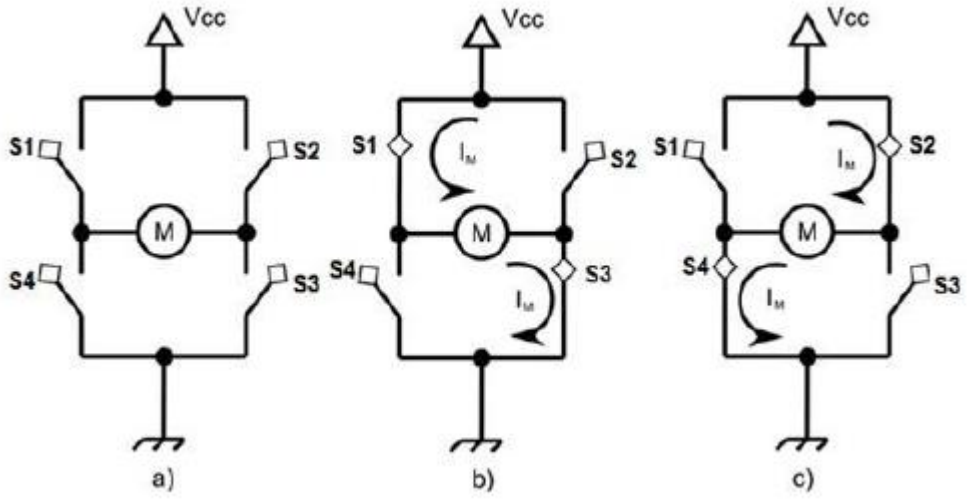




Hardware e Software

Ponte-H:

- Usada para acionamento de componentes que precisam de inversão do sentido da passagem de corrente durante seu funcionamento, como motores elétricos.



Hardware e Software

- O uso coordenado de ponte-H com sinais PWM permite não só a inversão de sentido, mas também uma atuação proporcional na velocidade, como por exemplo, variando o *duty-cycle* do sinal PWM de acionamento.
- O modelo de ponte-H utilizado no projeto foi o VNH5019, também da Pololu.

Hardware e Software

O template:

- Deve-se criar uma conta no Mbed e baixar o template do programa disponível em:

https://os.mbed.com/users/lcaepusp/code/Template_PTC3471_Geral/

- Basta entrar na conta do Mbed, acessar o link e selecionar ``import into compiler''.

Hardware e Software

- O programa inicia com a inclusão de bibliotecas-base:

```
1 #include "mbed.h"
2 #include "QEI.h"
3 #include "USBSerial.h"
4 #include "PTC3471.h"
5
6 #define Ts 0.01
7 #define pi 3.14159
```

- Em seguida há as definições das variáveis, objetos e funções a serem usados no programa:

Hardware e Software

```
10 /*****************  Definição de Variaveis, Objetos e Funções ********************/
11 /*******************************************************************************/
12
13 USBSerial  pc;                                                    // Objeto de comunicação serial com o TeraTerm
14
15 Ticker Control_Interrupt;                                         // Interrupção de Tempo para acionamento do algoritmo de controle
16
17 //QEI Encoder_Motor (PTD0,PTB17,NC, 300, QEI::X4_ENCODING);         // Objeto de leitura do encoder do motor
18 QEI Encoder_Motor (PTB17,PTD0,NC, 300, QEI::X4_ENCODING);         // Objeto de leitura do encoder do motor
19 QEI Encoder_Pendulo (PTA12,PTA13,NC, 600, QEI::X4_ENCODING);     // Objeto de leitura do encoder do pêndulo
20
21 DigitalOut Horario(PTC1);                                        // DigitalOut que sinaliza se deve virar o motor no sentido horário
22 DigitalOut AntiHorario(PTD5); //                                 // DigitalOut que sinaliza se deve virar o motor no sentido anti-horário
23 PwmOut Motor(PTD6); //                                           // AnalogOut (PWM) que indica de 0 a 1 qual o módulo da tensão sobre o motor
24
25 bool Flag_Controle = false;
26 int PlotCount = 0;
27
28 double phi0 = 0;                                                 // phi0 -> Ângulo lido pelo Encoder_Braco
29 double phi1 = 0;                                                 // phi1 -> Ângulo lido pelo Encoder_Pendulo
30
31 double th0 = 0;                                                  // th0 -> Ângulo do braço
32 double th1 = 0;                                                  // th1 -> Ângulo do pêndulo
33 double dth0 = 0;                                                 // dth0 -> Velocidade do braço
34 double dth1 = 0;                                                 // dth1 -> Velocidade do pêndulo
35
36 double th0_f = 0;                                                // th0 -> Ângulo do braço
37 double th1_f = 0;                                                // th1 -> Ângulo do pêndulo
38 double dth0_f = 0;                                               // dth0 -> Velocidade do braço
39 double dth1_f = 0;                                               // dth1 -> Velocidade do pêndulo
40
41 double tau = 5e-2;
42
43 double th0_a = 0;                                                // Valor de th0 um período de amostragem anterior
44 double th1_a = 0;                                                // Valor de th1 um período de amostragem anterior
45
46 float  u=0;
47
48 void Init(void);                                                 // Função de Inicialização
49 void Control_Function(void);                                     // Função de flag do controle, a ser chamada pela interrupção
50 void Sensor_Read(void);                                          // Função de leitura dos sensores
51 void Controle_Algoritmo(void);                                   // Função que implementa o algoritmo de controle escolhido
```

Hardware e Software

- Função main():

```
65 int main() {
66
67     /*****************************************************************************/
68     /** Inicialização do algoritmo de proteção. NUNCA DEVE SER RETIRADO DO PROGRAMA **/
69     /**/                                wait(5);                                  /**/
70     /**/         Protecao_Init(&Encoder_Motor, &Control_Interrupt, pi);         /**/
71     /** Inicialização do algoritmo de proteção. NUNCA DEVE SER RETIRADO DO PROGRAMA **/
72     /*****************************************************************************/
73
74     Init();
75     while(1) {
76
77         if(Flag_Controle){
78
79             Sensor_Read();                      // Executa a leitura dos sensores
80             Controle_Algoritmo();               // Execução do seu algoritmo de controle
81
82             PlotCount++;
83             if(PlotCount>=100){            // Controla para que o printf ocorra apenas uma vez a cada 10 iterações
84
85                 PlotCount = 0;
86                 pc.printf("Theta_0: %f, dTheta_0: %f\n\r", th0*180/pi, dth0*180/pi);
87                 pc.printf("Theta_1: %f, dTheta_1: %f\n\r", th1*180/pi, dth1*180/pi);
88
89             }
90
91             Flag_Controle = false;      // Sinaliza que deve-se esperar o próximo sinal da
92                                         //interrupção de tempo para executar o próximo passo de controle
93         }
94     }
95 }
```

Hardware e Software

- Função de interrupção para temporização do loop de controle:

```
177 /**************** Função de flag do algoritmo de controle *****************/
178 // Esta função avisa a main quando executar o próximo passo do algoritmo de
179 // controle. O uso de uma interrupção para o acionamento da flag garante que
180 // haja exatamente Ts segundos entre execuções.
181 void Control_Function(void){
182
183     Flag_Controle = true;
184
185 }
```

- Função de inicialização:

```
139 /************************ Função de Inicialização ****************************/
140 // Esta função concentra todas as inicializações do sistema
141 void Init(void){
142
143     Motor.period(0.0001);
144     Horario = 0;
145     AntiHorario = 0;
146     Motor = 0.0;
147     Control_Interrupt.attach(&Control_Function, Ts);
148
149 }
```

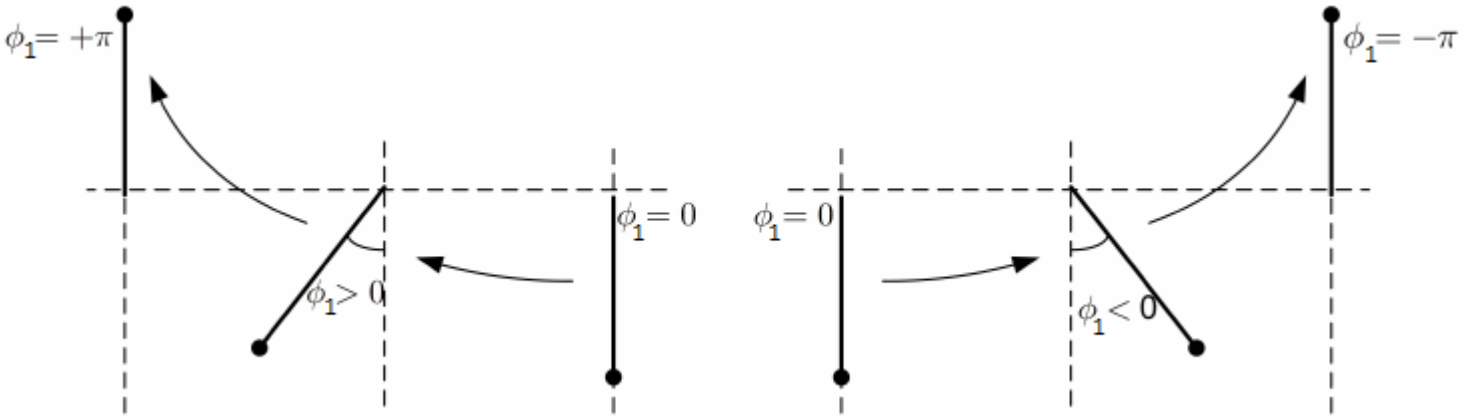
Hardware e Software

- Função de leitura dos sensores e tratamento de dados:

```
/********************* Função de leitura dos sensores ************************/
// Cada vez que esta função é chamada deve-se calcular os ângulos e velocidades
// angulares por algum método conhecido
void Sensor_Read(void){

    th0_a=th0;
    th1_a=th1;

    /** Leituras cruas dos ângulos do encoder **/
    phi0 = pi*Encoder_Motor.getPulses()/600.0;     // (eventos_lidos/eventos_por_volta)*2*pi = angulo_em_radianos
    phi1 = pi*Encoder_Pendulo.getPulses()/1200.0;  // (eventos_lidos/eventos_por_volta)*360  = angulo_em_graus

    th0 = phi0;
    /** Tratamento do ângulo lido para ser zero na vertical para cima **/
    // Como o encoder é incremental quando inicializamos o programa com o pêndulo na posição
    if(phi1>0)                // vertical para baixo esta passa a ser lida como 0°. Porém, para o algoritmo de controle
        th1 = phi1-pi;        // funcionar corretamente 0° deve ser o pêndulo na posição vertical para cima. Para
                              // garantir que isso aconteça subido o pêndulo no sentido horário ou anti-horário fazemos
    else if(phi1<=0)          // th1 = th1-sgn(th1)*pi, onde sgn(x) é o sinal de x.
        th1 = phi1+pi;        // Para ficar mais claro o funcionamento destes "if else" plote o sinal de th1 no tera term

    // Filtro (1/tau*s +1) nos angulos
    th0_f = (tau/(Ts+tau))*th0_f + (Ts/(Ts+tau))*th0;
    th1_f = (tau/(Ts+tau))*th1_f + (Ts/(Ts+tau))*th1;

}
```

Hardware e Software

- Ao inicializar o sistema, o pêndulo deve estar em equilíbrio na posição vertical para baixo.
- Quando o encoder é inicializado com a haste na vertical para baixo, tem-se o ângulo ϕ_1 = 0 rad.
- Ao mover o pêndulo para a posição invertida, o ângulo passa a ser $\phi_1 \pm \pi$ rad, dependendo se a haste foi movida no sentido horário ou anti-horário.



Hardware e Software

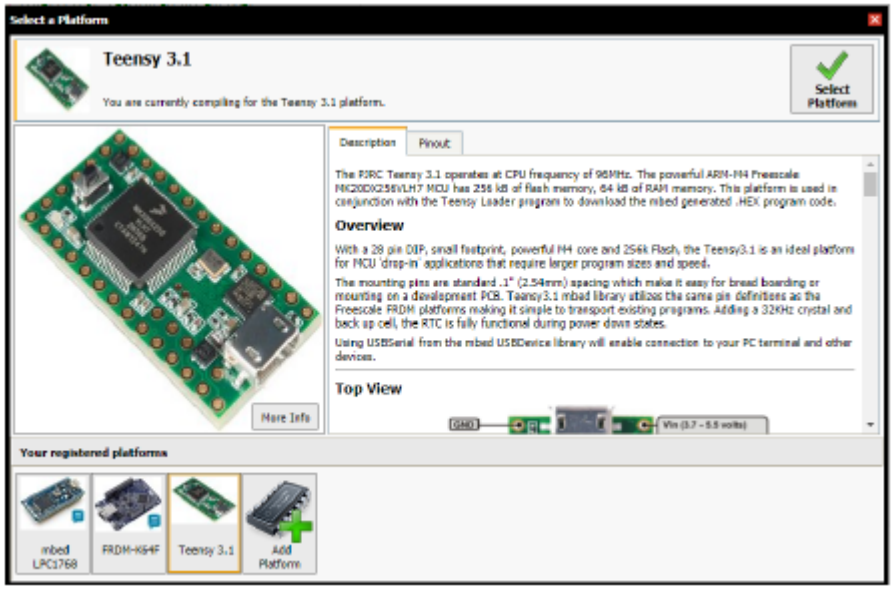
- Porém, na posição invertida (posição em que o pêndulo será equilibrado), pelo modelo a ser apresentado, deve-se ter um ângulo $\theta_1 = 0$.

- Uma forma solucionar isso consiste em fazer:

$$\theta_1 = \phi_1 - \text{sign}(\phi_1)\pi$$

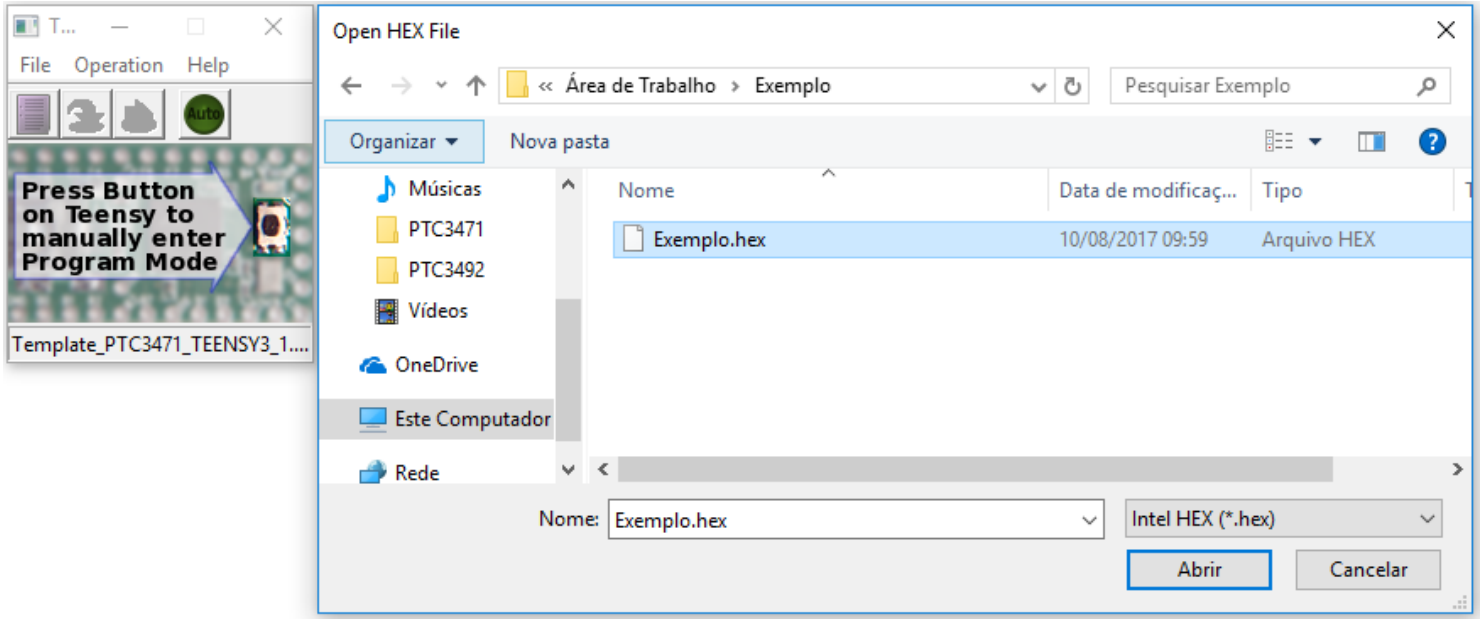
Hardware e Software

- Depois de feita a programação desejada, é necessário carregar um formato de arquivo específico (.hex) na Teensy.
- Primeiramente, é necessário garantir que o Mbed esteja utilizando o microcontrolador correto.
- Há o modelo Teensy 3.1, compatível com a versão 3.2.



Hardware e Software

- Para o correto download do arquivo .hex na Teensy é necessário utilizar o Software *Teensy Loader.*



- Para que não seja necessário selecionar o arquivo .hex toda vez que o programa for alterado, é recomendado que substitua o arquivo anterior por um novo com mesmo nome e selecionar a opção AUTO.

Hardware e Software

- Assim, toda vez que o usuário apertar o botão físico no microcontrolador, este entrará em modo de programação do arquivo .hex.

Instruções para uso do equipamento

Todos os membros do grupo devem se atentar à região completa de operação do sistema. Para inicialização do equipamento, os seguintes passos devem ser seguidos:

- garantir que o interruptor esteja na posição desligada;
- manter o pêndulo na posição de repouso (deve parar completamente de oscilar);
- selecionar o arquivo .hex no *Teensy Loader* e utilizar o software no modo AUTO.
- apertar o botão da Teensy para download do arquivo .hex;
- após três segundos, mover o pêndulo para a posição invertida (modo sem *swing-up*);
- mudar o interruptor para posição ligada.

Instruções para uso do equipamento

Se o grupo realizar sequencialmente os passos anteriores, cinco segundos após pressionar o botão da *Teensy*, o programa começará a atuar. Após a realização do experimento, mudar o interruptor para a posição desligada.