Ball and Plate

1.0

Generated by Doxygen 1.8.18

Chapter 1

Class Index

1.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 2

Class Documentation

2.1 Motor Class Reference

Implementa funções para controle dos servomotores.

`#include <Motor.h>`

Public Member Functions

- Motor (int controlPin, int infLimit, int supLimit)
  Constrói um objeto Motor.
- Motor ()
  Constrói um objeto Motor padrão.
- void setupMotor ()
  Realiza configurações iniciais do motor.
- void setPos (int pos)
  Envia o motor para uma dada posição.
- void setZero (int zeroPos)
  Define o zero do motor (igual a 90 por padrão).
- void goZero ()
  Envia o motor para a posição 0.
- void invertMotor ()
  Inverte a direção do giro dos motores quando chamada.

2.1.1 Detailed Description

Implementa funções para controle dos servomotores.

Definition at line 8 of file Motor.h.

2.1.2 Constructor & Destructor Documentation

2.1.2.1 Motor()

```
Motor::Motor (
            int controlPin,
            int infLimit,
            int supLimit )
```

Constrói um objeto Motor.

Parameters

controlPin	Controle PWM do motor.
infLim	Limite inferior para a posição do motor (-90 - 0).
supLim	Limite superior para a posição do motor (0 - 90).

Definition at line 3 of file Motor.cpp.

```
4 {
5     if(infLimit < -90) infLimit = -90;
6     if(infLimit > 0) infLimit = 0;
7     if(supLimit < 0) supLimit = 0;
8     if(supLimit > 90) supLimit = 90;
9
10     this->controlPin = controlPin;
11     this->infLimit = infLimit;
12     this->supLimit = supLimit;
13     zeroPos = 90;
14     direction = 1;
15 }
```

2.1.3 Member Function Documentation

2.1.3.1 setPos()

```
void Motor::setPos (
            int pos )
```

Envia o motor para uma dada posição.

Parameters

pos	Posição para a qual é enviada o motor.

Definition at line 27 of file Motor.cpp.

```
28 {
29     pos *= direction;
30     if(pos < infLimit) pos = infLimit;
31     if(pos > supLimit) pos = supLimit;
32     servo.write(pos + zeroPos);
33 }
```

2.1.3.2 setZero()

```
void Motor::setZero (
```

```
            int zeroPos )
```

Define o zero do motor (igual a 90 por padrão).

Parameters

zeroPos	Posição considerada como zero do motor (0 - 180).

Definition at line 35 of file Motor.cpp.

```
36 {
37     this->zeroPos = zeroPos;
38 }
```

The documentation for this class was generated from the following files:

- /home/iasbeck/Área de Trabalho/ballPlate/include/Motor.h
- /home/iasbeck/Área de Trabalho/ballPlate/src/Motor.cpp

2.2 MovingAverage Class Reference

Implementa um filtro de médias móveis.

```
#include <MovingAverage.h>
```

Public Member Functions

- MovingAverage (int filterSize)
  Constrói um objeto MovingAverage.
- float compute (float input)
  Retorna o dado filtrado (saída do filtro)

2.2.1 Detailed Description

Implementa um filtro de médias móveis.

Definition at line 9 of file MovingAverage.h.

2.2.2 Constructor & Destructor Documentation

2.2.2.1 MovingAverage()

```
MovingAverage::MovingAverage (
            int filterSize )
```

Constrói um objeto MovingAverage.

Parameters

filterSize	Número de dados considerado na computação da média.

Definition at line 3 of file MovingAverage.cpp.

```
4 {
5     // Inicializando o vetor que armazena os dados passados
6     for(int i = 0; i < 20; i++)
7     {
8         lastInputs[i] = 0;
9     }
10
11     // Verificando a dimensão do filtro
12     if (filterSize > 20)
13     {
14         Serial.println("filterSize deve ser menor ou igual a 20!");
15         filterSize = 20;
16     }
17     this->filterSize = filterSize;
18 }
```

2.2.3 Member Function Documentation

2.2.3.1 compute()

```
float MovingAverage::compute (
            float input )
```

Retorna o dado filtrado (saída do filtro)

Parameters

input	último valor bruto lido.

Returns

Dado filtrado

Definition at line 25 of file MovingAverage.cpp.

```
26 {
27     float output = 0;
28
29     // Atualizando o vetor que armazena os dados passados
30     for (int i = MAX_SIZE - 1; i > 0; i--)
31     {
32         lastInputs[i] = lastInputs[i-1];
33     }
34     lastInputs[0] = input;
35
36     // Calculando o valor de saída do filtro
37     for (int i = 0; i < filterSize; i++)
38     {
39         output += lastInputs[i];
40     }
41
42     output /= filterSize;
43
44     return output;
45 }
```

The documentation for this class was generated from the following files:

- /home/iasbeck/Área de Trabalho/ballPlate/include/MovingAverage.h
- /home/iasbeck/Área de Trabalho/ballPlate/src/MovingAverage.cpp

2.3 Pid Class Reference

Classe que representa o controlador PID.

`#include <Pid.h>`

Public Member Functions

- Pid (float kp, float ki, float kd, float T)
  Constrói um objeto PID.
- Pid ()
  Constrói um objeto PID padrão.
- float compute (float out)
  Computa a ação de controle.
- void setLimits (float infLim, float supLim)
  Define os limites da ação de controle.
- void setRef (float ref)
  Define a referência.
- float getErr ()
  Retorna o valor do erro.

2.3.1 Detailed Description

Classe que representa o controlador PID.

Definition at line 7 of file Pid.h.

2.3.2 Constructor & Destructor Documentation

2.3.2.1 Pid()

```
Pid::Pid (
            float kp,
            float ki,
            float kd,
            float T )
```

Constrói um objeto PID.

Parameters

kp	Constante proporcional.
ki	Constante integral.
kd	Constante derivativa.
T	Tempo de amostragem (em milissegundos).

Definition at line 3 of file Pid.cpp.

```
4 {
5     this->kp = kp;
6     this->ki = ki;
7     this->kd = kd;
8     this->T = T;
9
10     ie = 0;
11 }
```

2.3.3 Member Function Documentation

2.3.3.1 compute()

```
float Pid::compute (
            float out )
```

Computa a ação de controle.

Parameters

out	Saída do sistema (valor advindo do sensor).

Returns

Ação de controle.

Definition at line 18 of file Pid.cpp.

```
19 {
20     float control;
21
22     err = ref - out;
23     de = (err - errPrev);
24     ie = ie + err;
25
26     // Zero o integrador para amenizar o overshoot
27     if(err*errPrev < 0) ie = 0;
28
29     control = kp*err + ki*ie + kd*de;
30
31     Serial.println("output = " + String(out));
32     Serial.println("err = " + String(err));
33     Serial.println("de = " + String(de));
34     Serial.println("ie = " + String(ie));
35     Serial.println("kp*err = " + String(kp*err));
36     Serial.println("ki*ie = " + String(ki*ie));
37     Serial.println("kd*de = " + String(kd*de));
38     Serial.println("control = " + String(control));
39
40     if(control < infLim) control = infLim;
41     if(control > supLim) control = supLim;
42
```

```
43     Serial.println("control = " + String(control));
44     Serial.println("-----------------------------------------");
45
46     errPrev = err;
47
48     return control;
49 }
```

2.3.3.2 getErr()

```
float Pid::getErr ( )
```

Retorna o valor do erro.

Returns

Erro computado na última iteração.

Definition at line 62 of file Pid.cpp.

```
63 {
64     return err;
65 }
```

2.3.3.3 setLimits()

```
void Pid::setLimits (
             float infLim,
             float supLim )
```

Define os limites da ação de controle.

Parameters

infLim	Limite inferior da ação de controle.
supLim	Limite superior da ação de controle.

Definition at line 51 of file Pid.cpp.

```
52 {
53     this->infLim = infLim;
54     this->supLim = supLim;
55 }
```

2.3.3.4 setRef()

```
void Pid::setRef (
             float ref )
```

Define a referência.

Parameters

ref	Referência (valor desejado para a saída).

Definition at line 57 of file Pid.cpp.

```
58 {
59     this->ref = ref;
60 }
```

The documentation for this class was generated from the following files:

- /home/iasbeck/Área de Trabalho/ballPlate/include/Pid.h
- /home/iasbeck/Área de Trabalho/ballPlate/src/Pid.cpp

2.4 Touch Class Reference

Realiza a leitura da touchscreen.

`#include <Touch.h>`

Public Member Functions

- Touch (int touchPin1, int touchPin2, int touchPin3, int touchPin4)

  Constrói um objeto Touch.
- Touch ()

  Constrói um objeto Touch padrão.
- int getRawX ()

  Retorna o valor bruto (0-1023) lido no eixo X.
- int getRawY ()

  Retorna o valor bruto (0-1023) lido no eixo Y.
- float getCmX ()

  Retorna a posição do toque em cm no eixo X.
- float getCmY ()

  Retorna a posição do toque em cm no eixo Y.
- boolean **isTouching** ()

2.4.1 Detailed Description

Realiza a leitura da touchscreen.

Definition at line 9 of file Touch.h.

2.4.2 Constructor & Destructor Documentation

2.4.2.1 Touch()

```
Touch::Touch (
            int touchPin1,
            int touchPin2,
            int touchPin3,
            int touchPin4 )
```

Constrói um objeto Touch.

Parameters

touchPin1	Pino 1 da touchscreen.
touchPin2	Pino 2 da touchscreen.
touchPin3	Pino 3 da touchscreen.
touchPin4	Pino 4 da touchscreen.

Definition at line 3 of file Touch.cpp.

```
4 {
5     this->touchPin1 = touchPin1;
6     this->touchPin2 = touchPin2;
7     this->touchPin3 = touchPin3;
8     this->touchPin4 = touchPin4;
9
10     xRaw = yRaw = xCm = yCm = 0;
11     touching = false;
12 }
```

2.4.3 Member Function Documentation

2.4.3.1 getCmX()

```
float Touch::getCmX ( )
```

Retorna a posição do toque em cm no eixo X.

Returns

Posição no eixo X.

Definition at line 47 of file Touch.cpp.

```
48 {
49     xRaw = getRawX();
50     if(xRaw > 10)
51     {
52         touching = true;
53         xCm = 0.0409*xRaw - 19.642;
54     }
55     else
56     {
57         touching = false;
58     }
59
60     return xCm;
61 }
```

2.4.3.2 getCmY()

```
float Touch::getCmY ( )
```

Retorna a posição do toque em cm no eixo Y.

Returns

Posição no eixo Y.

Definition at line 63 of file Touch.cpp.

```
64 {
65     yRaw = getRawY();
66     if(yRaw > 10)
67     {
68         touching = true;
69         yCm = 0.0354*yRaw - 16.612;
70     }
71     else
72     {
73         touching = false;
74     }
75
76     return yCm;
77 }
```

2.4.3.3 getRawX()

```
int Touch::getRawX ( )
```

Retorna o valor bruto (0-1023) lido no eixo X.

Returns

Valor bruto lido no eixo X (0-1023)

Definition at line 19 of file Touch.cpp.

```
20 {
21     pinMode(touchPin3, INPUT);
22     pinMode(touchPin1, INPUT);
23     digitalWrite(touchPin1, LOW);
24     pinMode(touchPin2, OUTPUT);
25     digitalWrite(touchPin2, LOW);
26     pinMode(touchPin4, OUTPUT);
27     digitalWrite(touchPin4, HIGH);
28     delay(3);
29     xRaw = analogRead(touchPin3);
30     return xRaw;
31 }
```

2.4.3.4 getRawY()

```
int Touch::getRawY ( )
```

Retorna o valor bruto (0-1023) lido no eixo Y.

Returns

Valor bruto lido no eixo YS (0-1023)

Definition at line 33 of file Touch.cpp.

```
34 {
35     pinMode(touchPin3, OUTPUT);
36     digitalWrite(touchPin3, HIGH);
37     pinMode(touchPin1, OUTPUT);
38     digitalWrite(touchPin1, LOW);
39     pinMode(touchPin2, INPUT);
40     pinMode(touchPin4, INPUT);
41     digitalWrite(touchPin4, LOW);
42     delay(3);
43     yRaw = analogRead(touchPin2);
44     return yRaw;
45 }
```

The documentation for this class was generated from the following files:

- /home/iasbeck/Área de Trabalho/ballPlate/include/Touch.h
- /home/iasbeck/Área de Trabalho/ballPlate/src/Touch.cpp

Index