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	Pino de 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/MEGASync/Mestrado/2020-1/ballPlate/include/Motor.h
- /home/iasbeck/MEGASync/Mestrado/2020-1/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/MEGASync/Mestrado/2020-1/ballPlate/include/MovingAverage.h
- /home/iasbeck/MEGASync/Mestrado/2020-1/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     errPrev = 0;
12 }
```

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 19 of file Pid.cpp.

```
20 {
21     float control;
22
23     err = ref - out;
24     de = (err - errPrev)/T;
25     ie = ie + err*T;
26
27     // Zero o integrador para amenizar o overshoot
28     // if(err*errPrev < 0) ie = 0; // DEBUG
29
30     control = kp*err + ki*ie + kd*de;
31
32     // Serial.print(String(kp*err) + " ");
33     // Serial.print(String(ki*ie) + " ");
34     // Serial.print(String(kd*de) + " ");
35     // Serial.print("\n");
36
37     if(control < infLim) control = infLim;
38     if(control > supLim) control = supLim;
39
40     errPrev = err;
41
42     return control;
43 }
```

2.3.3.2 getErr()

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

Retorna o valor do erro.

Returns

Erro computado na última iteração.

Definition at line 56 of file Pid.cpp.

```
57 {
58     return err;
59 }
```

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 45 of file Pid.cpp.

```
46 {
47     this->infLim = infLim;
48     this->supLim = supLim;
49 }
```

2.3.3.4 setRef()

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

Define a referência.

Parameters

ref	Referência (valor desejado para a saída).

Definition at line 51 of file Pid.cpp.

```
52 {
53     this->ref = ref;
54 }
```

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

- /home/iasbeck/MEGASync/Mestrado/2020-1/ballPlate/include/Pid.h
- /home/iasbeck/MEGASync/Mestrado/2020-1/ballPlate/src/Pid.cpp

2.4 PidDigital Class Reference

Public Member Functions

- PidDigital (float A, float B, float C, float D, float E, float T)

  Constrói um objeto PID.
- PidDigital ()

  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.

Public Attributes

- int **debug**

2.4.1 Detailed Description

Definition at line 6 of file PidDigital.h.

2.4.2 Constructor & Destructor Documentation

2.4.2.1 PidDigital()

```
PidDigital::PidDigital (
            float A,
            float B,
            float C,
            float D,
            float E,
            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 PidDigital.cpp.

```
4 {
5     this->A = A;
6     this->B = B;
7     this->C = C;
8     this->D = D;
9     this->E = E;
10     this->T = T;
11
12     u2 = u1 = u = e2 = e1 = e = 0;
13 }
```

2.4.3 Member Function Documentation

2.4.3.1 compute()

```
float PidDigital::compute (
            float out )
```

Computa a ação de controle.

Parameters

out	Saída do sistema (valor advindo do sensor).

Returns

Ação de controle.

Definition at line 20 of file PidDigital.cpp.

```
21 {
22     e = ref - out;
23
24     u = A*u1 + B*u2 + C*e + D*e1 + E*e2;
25
26     if(u < infLim) u = infLim;
27     if(u > supLim) u = supLim;
28
29     u2 = u1;
30     u1 = u;
31     e2 = e1;
32     e1 = e;
33
34     return u;
35 }
```

2.4.3.2 getErr()

```
float PidDigital::getErr ( )
```

Retorna o valor do erro.

Returns

Erro computado na última iteração.

Definition at line 48 of file PidDigital.cpp.

```
49 {
50     return e;
51 }
```

2.4.3.3 setLimits()

```
void PidDigital::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 37 of file PidDigital.cpp.

```
38 {
39     this->infLim = infLim;
40     this->supLim = supLim;
41 }
```

2.4.3.4 setRef()

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

Define a referência.

Parameters

ref	Referência (valor desejado para a saída).

Definition at line 43 of file PidDigital.cpp.

```
44 {
45     this->ref = ref;
46 }
```

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

- /home/iasbeck/MEGASync/Mestrado/2020-1/ballPlate/include/PidDigital.h
- /home/iasbeck/MEGASync/Mestrado/2020-1/ballPlate/src/PidDigital.cpp

2.5 SerialComm Class Reference

Implementa funções para comunicação serial.

`#include <SerialComm.h>`

Public Member Functions

- SerialComm ()

  Constrói um novo objeto SerialComm.
- void setupComm ()

  Inicializa a comunicação serial.
- void setPrecision (int precision)

  Determina o número de casas após a vírgula que será adotado no envio das mensagens.
- void sendData (String data)

  Realiza o envio de mensagens do tipo string.
- void sendData (float data)

  Realiza o envio de mensagens do tipo float.
- void sendData (float data[ ], int sizeData)

  Realiza o envio de mensagens compostas por vários elementos do tipo float.
- float rcvData ()

  Realiza o recebimento de dados do tipo float.
- void rcvData (float ∗rcvFloat, int sizeData)

  Realiza o recebimento de dados compostos por vários elementos do tipo float.
- bool dataReceived ()

  Retorna uma flag que indica se a chamada da função resultou na leitura de algum dado, ou seja, se algum dado foi de fato recebido. Esta função deve ser chamada logo após a execução da função rcvData().

2.5.1 Detailed Description

Implementa funções para comunicação serial.

Definition at line 10 of file SerialComm.h.

2.5.2 Constructor & Destructor Documentation

2.5.2.1 SerialComm()

```
SerialComm::SerialComm ( )
```

Constrói um novo objeto SerialComm.

Definition at line 3 of file SerialComm.cpp.

```
4 {
5     precision = 5;
6 }
```

2.5.3 Member Function Documentation

2.5.3.1 dataReceived()

```
bool SerialComm::dataReceived ( )
```

Retorna uma flag que indica se a chamada da função resultou na leitura de algum dado, ou seja, se algum dado foi de fato recebido. Esta função deve ser chamada logo após a execução da função rcvData().

Returns

true - Houve recebimento de dados.

false - Não havia dados a serem lidos.

Definition at line 100 of file SerialComm.cpp.

```
101 {
102     return received;
103 }
```

2.5.3.2 rcvData() [1/2]

```
float SerialComm::rcvData ( )
```

Realiza o recebimento de dados do tipo float.

Returns

Dado (float) recebido.

Definition at line 36 of file SerialComm.cpp.

```
37 {
38     String rcvString;
39     float rcvFloat;
40     char rcvChar;
41 
42     received = false;
43     if (Serial.available())
44     {
45         rcvString = "";
46         while (true)
47         {
48             rcvChar = Serial.read();
49             if (rcvChar == '\n') break;
50             rcvString += rcvChar;
51         }
52         rcvFloat = rcvString.toFloat();
53         serialClear();
54         received = true;
55     }
56     return rcvFloat;
57 }
```

2.5.3.3 rcvData() [2/2]

```
void SerialComm::rcvData (
            float * rcvFloat,
            int sizeData )
```

Realiza o recebimento de dados compostos por vários elementos do tipo float.

Parameters

rcvFloat	Endereço da primeira posição do vetor que armazenará, após a chamada da função, os dados (floats) recebidos.
sizeData	Número de dados a serem recebidos.

Definition at line 59 of file SerialComm.cpp.

```
60 {
61     String rcvStrings[sizeData];
62     String rcvString = "";
63     int rcvStringIndex = 0;
64     char rcvChar;
65 
66     received = false;
67     if (Serial.available())
68     {
69         rcvString = "";
70         while (true)
71         {
72             rcvChar = Serial.read();
73             if((rcvChar == '\n') || (rcvChar == MSG_BREAK))
74             {
75                 rcvStrings[rcvStringIndex] = rcvString;
76                 rcvStringIndex++;
77                 rcvString = "";
78 
79                 if (rcvChar == '\n')
80                 {
81                     break;
82                 }
83             }
84             else
85             {
86                 rcvString += rcvChar;
87             }
88         }
89 
90         for(int i = 0; i < sizeData; i++)
91         {
92             rcvFloat[i] = rcvStrings[i].toFloat();
93         }
94 
95         serialClear();
96         received = true;
97     }
98 }
```

2.5.3.4 sendData() [1/3]

```
void SerialComm::sendData (
            float data )
```

Realiza o envio de mensagens do tipo float.

Parameters

data	Mensagem (float) a ser enviada.

Definition at line 19 of file SerialComm.cpp.

```
20 {
21     Serial.println(data, precision);
22 }
```

2.5.3.5 sendData() [2/3]

```
void SerialComm::sendData (
            float data[],
            int sizeData )
```

Realiza o envio de mensagens compostas por vários elementos do tipo float.

Parameters

data	Vetor contendo os dados (float) a serem enviados.
sizeData	Número de dados a serem enviados.

Definition at line 24 of file SerialComm.cpp.

```
25 {
26     String msg;
27 
28     msg = String(data[0], precision);
29     for(int i = 1; i < sizeData; i++)
30     {
31         msg += MSG_BREAK + String(data[i], precision);
32     }
33     sendData(msg);
34 }
```

2.5.3.6 sendData() [3/3]

```
void SerialComm::sendData (
            String data )
```

Realiza o envio de mensagens do tipo string.

Parameters

data	Mensagem (string) a ser enviada.

Definition at line 14 of file SerialComm.cpp.

```
15 {
16     Serial.println(data);
17 }
```

2.5.3.7 setPrecision()

```
void SerialComm::setPrecision (
            int precision )
```

Determina o número de casas após a vírgula que será adotado no envio das mensagens.

Parameters

precision	Número de casas após a vírgula.

Definition at line 105 of file SerialComm.cpp.

```
106 {
107     this->precision = precision;
108 }
```

2.5.3.8 setupComm()

```
void SerialComm::setupComm ( )
```

Inicializa a comunicação serial.

Definition at line 8 of file SerialComm.cpp.

```
9 {
10     Serial.begin(BAUD_RATE);
11     while (!Serial);
12 }
```

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

- /home/iasbeck/MEGASync/Mestrado/2020-1/ballPlate/include/SerialComm.h
- /home/iasbeck/MEGASync/Mestrado/2020-1/ballPlate/src/SerialComm.cpp

2.6 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** ()

2.6.1 Detailed Description

Realiza a leitura da touchscreen.

Definition at line 9 of file Touch.h.

2.6.2 Constructor & Destructor Documentation

2.6.2.1 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 }
```

2.6.3 Member Function Documentation

2.6.3.1 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 }
```

2.6.3.2 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 }
```

2.6.3.3 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 }
```

2.6.3.4 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/MEGASync/Mestrado/2020-1/ballPlate/include/Touch.h
- /home/iasbeck/MEGASync/Mestrado/2020-1/ballPlate/src/Touch.cpp

Index