# Documentação do Drone

Gerado por Doxygen 1.8.16

# Capítulo 1

# Índice da hierarquia

## 1.1 Hierarquia de classes

Esta lista de heranças está organizada, dentro do possível, por ordem alfabética:

# Capítulo 2

# Índice dos componentes

## 2.1 Lista de componentes

Lista de classes, estruturas, uniões e interfaces com uma breve descrição:

# Capítulo 3

# Índice dos ficheiros

## 3.1 Lista de ficheiros

Lista de todos os ficheiros com uma breve descrição:

# Capítulo 4

# Documentação da classe

## 4.1 ControleSerial

Realiza o envio e recebimento de dados da porta serial, que pode ser definida

### Membros públicos

- ControleSerial ()

  *Construtor da classe*
- void iniciar (int uso)

  *Inicia a comunicação serial na porta desejada.*
- template<class tipo >
  void print (tipo texto)

  *Imprime um valor dado.*
- void println ()

  *Imprime uma quebra de linha*
- template<class tipo >
  void println (tipo texto)

  *Imprime um valor dado, com uma quebra de linha.*
- void setSerial (SoftwareSerial *serial2)

  *Define uma outra porta serial.*

### Atributos Protegidos

- SoftwareSerial * serial2

  *Ponteiro para o objeto serial definido pelo usuário.*
- int usando

### 4.1.1 Descrição detalhada

Realiza o envio e recebimento de dados da porta serial, que pode ser definida

Elton, 14/08/2019

### 4.1.2 Documentação dos Construtores & Destrutor

#### 4.1.2.1 ControleSerial()

```
ControleSerial ( )  [inline]
```

Construtor da classe

Elton, 14/08/2019.

### 4.1.3 Documentação dos métodos

#### 4.1.3.1 iniciar()

```
void iniciar (
            int uso )  [inline]
```

Inicia a comunicação serial na porta desejada.

Elton, 14/08/2019.

**Parâmetros**

| | |
|---|---|
| *uso* | Porta desejada:<br>0 = Padrão do Arduino;<br>1 = Porta definada pelo usuário |

#### 4.1.3.2 print()

```
void print (
            tipo texto )  [inline]
```

Imprime um valor dado.

Elton, 14/08/2019.

**Parâmetros de template**

| | |
|---|---|
| *tipo* | Tipo do valor. |

**Parâmetros**

| | |
|---|---|
| *texto* | O valor a ser impresso. |

#### 4.1.3.3 println() [1/2]

```
void println ( )  [inline]
```

Imprime uma quebra de linha

Elton, 14/08/2019.

#### 4.1.3.4 println() [2/2]

```
void println (
            tipo texto )  [inline]
```

Imprime um valor dado, com uma quebra de linha.

Elton, 14/08/2019.

**Parâmetros de template**

| | |
|---|---|
| *tipo* | Tipo do valor. |

**Parâmetros**

| | |
|---|---|
| *texto* | O valor a ser impresso. |

#### 4.1.3.5 setSerial()

```
void setSerial (
            SoftwareSerial * serial2 )  [inline]
```

Define uma outra porta serial.

Elton, 14/08/2019.

**Parâmetros**

| | |
|---|---|
| *serial2* | Ponteiro para o objeto "SoftwareSerial" da nova porta. |

### 4.1.4 Documentação dos dados membro

#### 4.1.4.1 serial2

```
SoftwareSerial* serial2  [protected]
```

Ponteiro para o objeto serial definido pelo usuário.

#### 4.1.4.2 usando

```
int usando  [protected]
```

Contém o índice da porta sendo usada.

0 = Padrão do Arduino

1 = Porta definada pelo usuário

A documentação para esta classe foi gerada a partir do seguinte ficheiro:

- 1Classe.ino

## 4.2 Drone

Implementa o controle de um drone.

### Membros públicos

- Drone ()
- void atualiza ()

  *Atualiza os dados do drone e o matém na posição desejada.*
- void desligar ()

  *Desliga os motores.*
- void irPara (float angulo)

  *Define o ângulo para o eixo pitch.*
- void ligar ()

  *Liga os motores do drone.*
- void pousar ()

  *Pousa o drone.*
- void setMotor (int indice, int porta)

  *Define a porta do motor.*

## Membros protegidos

- void atualizaEntradas ()

  *Objeto para leitura da tensão da bateria*
- void atualizaMotor ()

  *Define as novas velocidades dos motores.*
- void atualizaPID ()

  *Atualiza os PIDs.*
- void leitura ()

  *Efetua as leituras dos sensores.*

## Atributos Protegidos

- ControleSerial controle

  *Objeto para log do drone.*
- Inercial inercial

  *Variáveis do PID para controle do eixo pitch*
- Motor motor [4]

  *PID para controle da inclinação no eixo pitch.*
- PID pidPitch

  *Objeto para controle da IMU.*
- Registro registro

  *Objetos para controle dos motores.*
- VariaveisPID varPIDPitch
- Voltimetro voltimetro

  *Objeto para controle da comunicação do drone.*

### 4.2.1 Descrição detalhada

Implementa o controle de um drone.

Elton, 14/08/2019.

### 4.2.2 Documentação dos Construtores & Destrutor

#### 4.2.2.1 Drone()

```
Drone ( )  [inline]
```

### 4.2.3 Documentação dos métodos

#### 4.2.3.1 atualiza()

```
void atualiza ( )  [inline]
```

Atualiza os dados do drone e o matém na posição desejada.

Elton, 14/08/2019.

#### 4.2.3.2 atualizaEntradas()

```
void atualizaEntradas ( )  [inline], [protected]
```

Objeto para leitura da tensão da bateria

Atualiza entradas dos PIDs.

Elton, 14/08/2019.

#### 4.2.3.3 atualizaMotor()

```
void atualizaMotor ( )  [inline], [protected]
```

Define as novas velocidades dos motores.

Elton, 14/08/2019.

#### 4.2.3.4 atualizaPID()

```
void atualizaPID ( )  [inline], [protected]
```

Atualiza os PIDs.

Elton, 14/08/2019.

#### 4.2.3.5 desligar()

```
void desligar ( )  [inline]
```

Desliga os motores.

Elton, 14/08/2019.

#### 4.2.3.6 irPara()

```
void irPara (
            float angulo )  [inline]
```

Define o ângulo para o eixo pitch.

Elton, 14/08/2019.

**Parâmetros**

| | |
|---|---|
| *angulo* | O ângulo desejado em [º]. |

#### 4.2.3.7 leitura()

```
void leitura ( )  [inline], [protected]
```

Efetua as leituras dos sensores.

Elton, 14/08/2019.

#### 4.2.3.8 ligar()

```
void ligar ( )  [inline]
```

Liga os motores do drone.

Elton, 14/08/2019.

#### 4.2.3.9 pousar()

```
void pousar ( )  [inline]
```

Pousa o drone.

NÃO IMPLEMENTADO.

Elton, 14/08/2019.

#### 4.2.3.10 setMotor()

```
void setMotor (
            int indice,
            int porta )  [inline]
```

Define a porta do motor.

Elton, 14/08/2019.

**Parâmetros**

| | |
|---|---|
| *indice* | Índice do motor a ser definido. |
| *porta* | Porta desejada. |

### 4.2.4 Documentação dos dados membro

#### 4.2.4.1 controle

```
ControleSerial controle  [protected]
```

Objeto para log do drone.

#### 4.2.4.2 inercial

```
Inercial inercial  [protected]
```

Variáveis do PID para controle do eixo pitch

#### 4.2.4.3 motor

```
Motor motor[4]  [protected]
```

PID para controle da inclinação no eixo pitch.

#### 4.2.4.4 pidPitch

```
PID pidPitch  [protected]
```

Objeto para controle da IMU.

#### 4.2.4.5 registro

```
Registro registro  [protected]
```

Objetos para controle dos motores.

#### 4.2.4.6 varPIDPitch

```
VariaveisPID varPIDPitch  [protected]
```

#### 4.2.4.7 voltimetro

`Voltimetro voltimetro [protected]`

Objeto para controle da comunicação do drone.

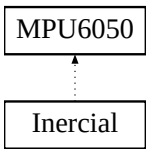
A documentação para esta classe foi gerada a partir do seguinte ficheiro:

- 1Classe.ino

## 4.3 Inercial

Classe para controle de uma IMU do modelo "MPU6050"

Diagrama de heranças da classe Inercial




### Membros públicos

- Inercial (TwoWire &w)
  *Construtor da classe*
- float getPitch ()
  *Retorna o valor filtrado no eixo Pitch.*
- float getRoll ()
  *Retorna o valor filtrado no eixo Roll.*
- float getYaw ()
  *Retorna o valor filtrado no eixo Yaw.*
- void leitura ()
  *Efetua uma leitura filtrando os valores lidos*

### Atributos Protegidos

- PassaBaixa filtroPitch
  *Filtro passa baixa para o eixo Roll.*
- PassaBaixa filtroRoll
  *Último valor lido e filtrado no eixo roll.*
- PassaBaixa filtroYaw
  *Filtro passa baixa para o eixo Pitch.*
- float pitch
  *Último valor lido e filtrado no eixo roll.*
- float roll
- float yaw
  *Último valor lido e filtrado no eixo roll.*

## Amigos

- class Registro

### 4.3.1 Descrição detalhada

Classe para controle de uma IMU do modelo "MPU6050"

Elton, 14/08/2019.

### 4.3.2 Documentação dos Construtores & Destrutor

#### 4.3.2.1 Inercial()

```
Inercial (
            TwoWire & w )  [inline]
```

Construtor da classe

Elton, 14/08/2019.

**Parâmetros**

| | |
|---|---|
| *w* | Objeto "Wire" |

### 4.3.3 Documentação dos métodos

#### 4.3.3.1 getPitch()

```
float getPitch ( )  [inline]
```

Retorna o valor filtrado no eixo Pitch.

Elton, 14/08/2019.

**Retorna**

Valor filtrado no eixo Pitch em [m/s^2].

#### 4.3.3.2 getRoll()

```
float getRoll ( )  [inline]
```

Retorna o valor filtrado no eixo Roll.

Elton, 14/08/2019.

**Retorna**

Valor filtrado no eixo Roll em [m/s^2].

#### 4.3.3.3 getYaw()

```
float getYaw ( )  [inline]
```

Retorna o valor filtrado no eixo Yaw.

Elton, 14/08/2019.

**Retorna**

Valor filtrado no eixo Yaw em [m/s^2].

#### 4.3.3.4 leitura()

```
void leitura ( )  [inline]
```

Efetua uma leitura filtrando os valores lidos

Elton, 14/08/2019.

### 4.3.4 Documentação das classes amigas e funções relacionadas

#### 4.3.4.1 Registro

```
friend class Registro  [friend]
```

### 4.3.5 Documentação dos dados membro

#### 4.3.5.1 filtroPitch

```
PassaBaixa filtroPitch  [protected]
```

Filtro passa baixa para o eixo Roll.

#### 4.3.5.2 filtroRoll

```
PassaBaixa filtroRoll  [protected]
```

Último valor lido e filtrado no eixo roll.

#### 4.3.5.3 filtroYaw

```
PassaBaixa filtroYaw  [protected]
```

Filtro passa baixa para o eixo Pitch.

#### 4.3.5.4 pitch

```
float pitch  [protected]
```

Último valor lido e filtrado no eixo roll.

#### 4.3.5.5 roll

```
float roll  [protected]
```

#### 4.3.5.6 yaw

```
float yaw  [protected]
```

Último valor lido e filtrado no eixo roll.

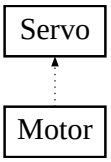
A documentação para esta classe foi gerada a partir do seguinte ficheiro:

- 1Classe.ino

## 4.4 Motor

Realiza o controle de um motor

Diagrama de heranças da classe Motor




### Membros públicos

- Motor ()

  *Construtor da Classe*
- void desligar ()

  *Desliga o motor, encerrando a rotação*
- void girar ()

  *Inicia a rotação do motor, imprimindo o mínimo para girar*
- void girar (int diferenca)

  *Define a velocidade do motor*
- void ligar ()

  *Liga o motor*
- void setMinimo (int minimo)

  *Define o valor mínimo de velocidade para o motor girar*

### Atributos Protegidos

- int minimo
- int velocidade

  *A velocidade mínima para rotação do motor*

### Amigos

- class Registro

### 4.4.1 Descrição detalhada

Realiza o controle de um motor

Elton, 14/08/2019.

### 4.4.2 Documentação dos Construtores & Destrutor

#### 4.4.2.1 Motor()

```
Motor ( )  [inline]
```

Construtor da Classe

Elton, 14/08/2019.

### 4.4.3 Documentação dos métodos

#### 4.4.3.1 desligar()

```
void desligar ( )  [inline]
```

Desliga o motor, encerrando a rotação

Elton, 14/08/2019.

#### 4.4.3.2 girar() [1/2]

```
void girar ( )  [inline]
```

Inicia a rotação do motor, imprimindo o mínimo para girar

Elton, 14/08/2019.

#### 4.4.3.3 girar() [2/2]

```
void girar (
            int diferenca )  [inline]
```

Define a velocidade do motor

Elton, 14/08/2019.

**Parâmetros**

| | |
|---|---|
| *diferenca* | Quanto acima do mínimo para girar o motor deverá girar |

#### 4.4.3.4 ligar()

```
void ligar ( )  [inline]
```

Liga o motor

Elton, 14/08/2019.

#### 4.4.3.5 setMinimo()

```
void setMinimo (
            int minimo )  [inline]
```

Define o valor mínimo de velocidade para o motor girar

Elton, 14/08/2019.

**Parâmetros**

| | |
|---|---|
| *minimo* | Valor mínimo definido |

### 4.4.4 Documentação das classes amigas e funções relacionadas

#### 4.4.4.1 Registro

```
friend class Registro  [friend]
```

### 4.4.5 Documentação dos dados membro

#### 4.4.5.1 minimo

```
int minimo  [protected]
```

#### 4.4.5.2 velocidade

```
int velocidade  [protected]
```

A velocidade mínima para rotação do motor

A documentação para esta classe foi gerada a partir do seguinte ficheiro:

- 1Classe.ino

## 4.5 PassaBaixa

Implementa um filtro de frequência do tipo "passa baixa"

### Membros públicos

- PassaBaixa ()

  *Construtor da classe*
- void entrar (float medida)

  *Insere uma nova medida no*
- float sair ()

  *Retorna o último valor calculado pelo filtro*
- void setAmostragem (float amostragem)

  *Define a amostragem do filtro.*
- void setFrequencia (float freqCorte)

  *Define a frequência de corte do filtro.*

### Membros protegidos

- void calculaConstante ()

  *Amostragem do sinal*

### Atributos Protegidos

- float amostragem

  *Frequência de corte do filtro em [rad/s]*
- float constante

  *Última saída do filtro*
- float freqCorte

  *Saída atual do filtro*
- float saida

  *Constante do filtro*
- float saidaAnt

### 4.5.1 Descrição detalhada

Implementa um filtro de frequência do tipo "passa baixa"

Elton, 14/08/2019.

### 4.5.2 Documentação dos Construtores & Destrutor

#### 4.5.2.1 PassaBaixa()

```
PassaBaixa ( )  [inline]
```

Construtor da classe

Elton, 14/08/2019.

### 4.5.3 Documentação dos métodos

#### 4.5.3.1 calculaConstante()

```
void calculaConstante ( )  [inline], [protected]
```

Amostragem do sinal

Calcula a constante utilizada no cálculo do filtro.

Elton, 14/08/2019.

#### 4.5.3.2 entrar()

```
void entrar (
             float medida )  [inline]
```

Insere uma nova medida no

Elton, 14/08/2019.

**Parâmetros**

| | |
|---|---|
| *medida* | A nova medida |

#### 4.5.3.3 sair()

```
float sair ( )  [inline]
```

Retorna o último valor calculado pelo filtro

Elton, 14/08/2019.

**Retorna**

O valor filtrado

#### 4.5.3.4 setAmostragem()

```
void setAmostragem (
            float amostragem )  [inline]
```

Define a amostragem do filtro.

Elton, 14/08/2019.

**Parâmetros**

| | |
|---|---|
| *amostragem* | A amostragem definida. |

#### 4.5.3.5 setFrequencia()

```
void setFrequencia (
            float freqCorte )  [inline]
```

Define a frequência de corte do filtro.

Elton, 14/08/2019.

**Parâmetros**

| | |
|---|---|
| *freqCorte* | A frequência definida. |

### 4.5.4 Documentação dos dados membro

#### 4.5.4.1 amostragem

```
float amostragem  [protected]
```

Frequência de corte do filtro em [rad/s]

#### 4.5.4.2 constante

```
float constante  [protected]
```

Última saída do filtro

#### 4.5.4.3 freqCorte

```
float freqCorte  [protected]
```

Saída atual do filtro

#### 4.5.4.4 saida

```
float saida  [protected]
```

Constante do filtro

#### 4.5.4.5 saidaAnt

```
float saidaAnt  [protected]
```

A documentação para esta classe foi gerada a partir do seguinte ficheiro:

- 1Classe.ino

## 4.6 Registro

Implementa um registro/log para dados do drone.

### Membros públicos

- Registro (ControleSerial ∗controle)
  *Construtor da Classe*
- void print ()
  *Imprime os valores desejados na porta serial.*
- void setAngulo (float ∗roll, float ∗pitch, float ∗yaw)
  *Define os endereços para as posições ângulares do robô.*
- void setTensao (float ∗tensao)
  *Define o endereço para o valor de tensão.*
- void setVelocidade (float ∗vel0, float ∗vel1, float ∗vel2, float ∗vel3)
  *Define os endereços para as velocidades dos motores do robô.*

## Membros protegidos

- void printAngulo ()

  *True = Deve imprimir os dados dos controladores PID.*
- void printPID ()

  *Imprime as inclinações.*
- void printTensao ()

  *Imprime a tensão.*
- void printVelocidade ()

  *Imprime as velocidades dos motores.*

## Atributos Protegidos

- ControleSerial * controle
- bool imprimeAngulo

  *Ponteiro para o endereço da variável com a velocidade do motor 3.*
- bool imprimePID

  *True = Deve imprimir as velocidades.*
- bool imprimeVelocidade

  *True = Deve imprimir as inclinações.*
- float * pitch

  *Ponteiro para o endereço da variável com o valor de inclinação no eixo roll.*
- float * roll

  *Ponteiro para o endereço da variável com o valor de tensão.*
- float * tensao

  *Ponteiro para o objeto "ControleSerial" para o envio de dados.*
- float * vel0

  *Ponteiro para o endereço da variável com o valor de inclinação no eixo yaw.*
- float * vel1

  *Ponteiro para o endereço da variável com a velocidade do motor 0.*
- float * vel2

  *Ponteiro para o endereço da variável com a velocidade do motor 1.*
- float * vel3

  *Ponteiro para o endereço da variável com a velocidade do motor 2.*
- float * yaw

  *Ponteiro para o endereço da variável com o valor de inclinação no eixo pitch.*

### 4.6.1 Descrição detalhada

Implementa um registro/log para dados do drone.

Elton, 14/08/2019.

### 4.6.2 Documentação dos Construtores & Destrutor

#### 4.6.2.1 Registro()

```
Registro (
            ControleSerial * controle )  [inline]
```

Construtor da Classe

Elton, 14/08/2019.

**Parâmetros**

| | |
|---|---|
| *controle* | Ponteiro para um objeto "ControleSerial" |

### 4.6.3 Documentação dos métodos

#### 4.6.3.1 print()

```
void print ( )  [inline]
```

Imprime os valores desejados na porta serial.

Elton, 14/08/2019.

#### 4.6.3.2 printAngulo()

```
void printAngulo ( )  [inline], [protected]
```

True = Deve imprimir os dados dos controladores PID.

Imprime as inclinações.

Elton, 14/08/2019.

#### 4.6.3.3 printPID()

```
void printPID ( )  [inline], [protected]
```

Imprime as inclinações.

Elton, 14/08/2019.

#### 4.6.3.4 printTensao()

```
void printTensao ( )  [inline], [protected]
```

Imprime a tensão.

Elton, 14/08/2019.

#### 4.6.3.5 printVelocidade()

```
void printVelocidade ( )  [inline], [protected]
```

Imprime as velocidades dos motores.

Elton, 14/08/2019.

#### 4.6.3.6 setAngulo()

```
void setAngulo (
            float * roll,
            float * pitch,
            float * yaw )  [inline]
```

Define os endereços para as posições ângulares do robô.

Elton, 14/08/2019.

**Parâmetros**

| | |
|---|---|
| *roll* | Ponteiro para a variável que contém a leitura do eixo "roll". |
| *pitch* | Ponteiro para a variável que contém a leitura do eixo "pitch". |
| *yaw* | Ponteiro para a variável que contém a leitura do eixo "yaw". |

#### 4.6.3.7 setTensao()

```
void setTensao (
            float * tensao )  [inline]
```

Define o endereço para o valor de tensão.

Elton, 14/08/2019.

**Parâmetros**

| | |
|---|---|
| *tensao* | Ponteiro para a variável que contém a tensão lida. |

#### 4.6.3.8 setVelocidade()

```
void setVelocidade (
            float * vel0,
            float * vel1,
            float * vel2,
            float * vel3 )  [inline]
```

Define os endereços para as velocidades dos motores do robô.

Elton, 14/08/2019.

**Parâmetros**

| | |
|---|---|
| *vel0* | Ponteiro para a variável que contém a leitura do motor 0. |
| *vel1* | Ponteiro para a variável que contém a leitura do motor 1. |
| *vel2* | Ponteiro para a variável que contém a leitura do motor 2. |
| *vel3* | Ponteiro para a variável que contém a leitura do motor 3. |

### 4.6.4 Documentação dos dados membro

#### 4.6.4.1 controle

```
ControroleSerial* controle  [protected]
```

#### 4.6.4.2 imprimeAngulo

```
bool imprimeAngulo  [protected]
```

Ponteiro para o endereço da variável com a velocidade do motor 3.

#### 4.6.4.3 imprimePID

```
bool imprimePID  [protected]
```

True = Deve imprimir as velocidades.

#### 4.6.4.4 imprimeVelocidade

```
bool imprimeVelocidade  [protected]
```

True = Deve imprimir as inclinações.

#### 4.6.4.5 pitch

```
float * pitch  [protected]
```

Ponteiro para o endereço da variável com o valor de inclinação no eixo roll.

#### 4.6.4.6 roll

```
float * roll  [protected]
```

Ponteiro para o endereço da variável com o valor de tensão.

#### 4.6.4.7 tensao

```
float* tensao  [protected]
```

Ponteiro para o objeto "ControleSerial" para o envio de dados.

#### 4.6.4.8 vel0

```
float * vel0  [protected]
```

Ponteiro para o endereço da variável com o valor de inclinação no eixo yaw.

#### 4.6.4.9 vel1

```
float * vel1  [protected]
```

Ponteiro para o endereço da variável com a velocidade do motor 0.

#### 4.6.4.10 vel2

```
float * vel2  [protected]
```

Ponteiro para o endereço da variável com a velocidade do motor 1.

#### 4.6.4.11 vel3

```
float * vel3  [protected]
```

Ponteiro para o endereço da variável com a velocidade do motor 2.

#### 4.6.4.12 yaw

```
float * yaw  [protected]
```

Ponteiro para o endereço da variável com o valor de inclinação no eixo pitch.

A documentação para esta classe foi gerada a partir do seguinte ficheiro:

- 1Classe.ino

## 4.7 VariaveisPID

Estrutura para armazenar dados de um controlador PID.

### Atributos Públicos

- double entrada
- double kd

  *Coeficiente integrativo do PID*
- double ki

  *Coeficiente proporcional do PID*
- double kp

  *SetPoint do PID.*
- double saida

  *Última entrada do PID.*
- double setPoint

  *Última saída do PID.*

### 4.7.1 Descrição detalhada

Estrutura para armazenar dados de um controlador PID.

Elton, 14/08/2019.

### 4.7.2 Documentação dos dados membro

#### 4.7.2.1 entrada

```
double entrada
```

#### 4.7.2.2 kd

```
double kd
```

Coeficiente integrativo do PID

#### 4.7.2.3 ki

```
double ki
```

Coeficiente proporcional do PID

#### 4.7.2.4 kp

```
double kp
```

SetPoint do PID.

#### 4.7.2.5 saida

```
double saida
```

Última entrada do PID.

#### 4.7.2.6 setPoint

```
double setPoint
```

Última saída do PID.

A documentação para esta estrutura foi gerada a partir do seguinte ficheiro:

- 1Classe.ino

## 4.8 Voltimetro

Implementa um voltímetro conectado a uma porta analógica do Arduino

### Membros públicos

- Voltimetro ()
  *Construtor da classe*
- int leitura ()
  *Efetua a leitura analógica e a converte para a tensão em [V]*
- void setPorta (int porta)
  *Define a porta de leitura*

## Atributos Protegidos

- int porta
- float tensao

  *Porta de leitura.*

## Amigos

- class Registro

### 4.8.1 Descrição detalhada

Implementa um voltímetro conectado a uma porta analógica do Arduino

Elton, 14/08/2019.

### 4.8.2 Documentação dos Construtores & Destrutor

#### 4.8.2.1 Voltimetro()

```
Voltimetro ( )  [inline]
```

Construtor da classe

Elton, 14/08/2019.

### 4.8.3 Documentação dos métodos

#### 4.8.3.1 leitura()

```
int leitura ( )  [inline]
```

Efetua a leitura analógica e a converte para a tensão em [V]

Elton, 14/08/2019.

**Retorna**

Tensão lida

#### 4.8.3.2 setPorta()

```
void setPorta (
            int porta )  [inline]
```

Define a porta de leitura

Elton, 14/08/2019.

**Parâmetros**

| | |
|---|---|
| *porta* | Porta da leitura |

### 4.8.4 Documentação das classes amigas e funções relacionadas

#### 4.8.4.1 Registro

```
friend class Registro  [friend]
```

### 4.8.5 Documentação dos dados membro

#### 4.8.5.1 porta

```
int porta  [protected]
```

#### 4.8.5.2 tensao

```
float tensao  [protected]
```

Porta de leitura.

A documentação para esta classe foi gerada a partir do seguinte ficheiro:

- 1Classe.ino

# Capítulo 5

# Documentação do ficheiro

## 5.1 Referência ao ficheiro 1Classe.ino

### Componentes

- class ControleSerial

  *Realiza o envio e recebimento de dados da porta serial, que pode ser definida*
- class Drone

  *Implementa o controle de um drone.*
- class Inercial

  *Classe para controle de uma IMU do modelo "MPU6050"*
- class Motor

  *Realiza o controle de um motor*
- class PassaBaixa

  *Implementa um filtro de frequência do tipo "passa baixa"*
- class Registro

  *Implementa um registro/log para dados do drone.*
- struct VariaveisPID

  *Estrutura para armazenar dados de um controlador PID.*
- class Voltimetro

  *Implementa um voltímetro conectado a uma porta analógica do Arduino*

## 5.2 Referência ao ficheiro 2Variaveis.ino

## 5.3 Referência ao ficheiro 3Funcoes.ino

## 5.4 Referência ao ficheiro 4SetupLoop.ino

### Funções

- void loop ()
- void setup ()

### 5.4.1 Documentação das funções

#### 5.4.1.1 loop()

```
void loop ( )
```

#### 5.4.1.2 setup()

```
void setup ( )
```

## 5.5 Referência ao ficheiro NovoDrone.ino

```
#include <Servo.h>
#include <MPU6050_tockn.h>
#include <Wire.h>
#include <PID_v1.h>
#include <SoftwareSerial.h>
```

### Macros

- #define imprimeSensor 1
- #define KDpitch 0

  *Constante derivativa do PIDPitch*
- #define KIpitch 0

  *Constante integrativa do PIDPitch*
- #define KPpitch 1

  *Constante proporcial do PIDPitch*
- #define LIGA 17

  *Velocidade mínima necessária para ligar os motores*
- #define outputLIMIT 15

  *Limite de velocidade dos motores*
- #define QUANT_MOTOR 4

  *Quantidade de motores do drone*
- #define RODA 31

  *Velocidade mínima necessária para os motores começarem a rodar*
- #define serial 1
- #define VOO 80

  *Velocidade mínima necessária para o drone levantar voo*

### 5.5.1 Documentação das macros

#### 5.5.1.1 imprimeSensor

```
#define imprimeSensor 1
```

Constante de controle do registro

0 = não registrar dados dos sensores, 1 = registrar

#### 5.5.1.2 KDpitch

```
#define KDpitch 0
```

Constante derivativa do PIDPitch

#### 5.5.1.3 KIpitch

```
#define KIpitch 0
```

Constante integrativa do PIDPitch

#### 5.5.1.4 KPpitch

```
#define KPpitch 1
```

Constante proporcial do PIDPitch

#### 5.5.1.5 LIGA

```
#define LIGA 17
```

Velocidade mínima necessária para ligar os motores

#### 5.5.1.6 outputLIMIT

```
#define outputLIMIT 15
```

Limite de velocidade dos motores

#### 5.5.1.7 QUANT_MOTOR

```
#define QUANT_MOTOR 4
```

Quantidade de motores do drone

#### 5.5.1.8 RODA

```
#define RODA 31
```

Velocidade mínima necessária para os motores começarem a rodar

#### 5.5.1.9 serial

```
#define serial 1
```

Constante de controle da porta de comunicação utilizada

1 = bluetooth, 0 = serial

#### 5.5.1.10 VOO

```
#define VOO 80
```

Velocidade mínima necessária para o drone levantar voo

# Índice