

Documentação do Drone

Gerado por Doxygen 1.8.16

1 Índice da hierarquia	1
1.1 Hierarquia de classes	1
2 Índice dos componentes	3
2.1 Lista de componentes	3
3 Índice dos ficheiros	5
3.1 Lista de ficheiros	5
4 Documentação da classe	7
4.1 ControleSerial	7
4.1.1 Descrição detalhada	7
4.1.2 Documentação dos Construtores & Destrutor	8
4.1.2.1 ControleSerial()	8
4.1.3 Documentação dos métodos	8
4.1.3.1 iniciar()	8
4.1.3.2 print()	8
4.1.3.3 println() [1/2]	9
4.1.3.4 println() [2/2]	9
4.1.3.5 setSerial()	9
4.1.4 Documentação dos dados membro	10
4.1.4.1 serial2	10
4.1.4.2 usando	10
4.2 Drone	10
4.2.1 Descrição detalhada	11
4.2.2 Documentação dos Construtores & Destrutor	11
4.2.2.1 Drone()	11
4.2.3 Documentação dos métodos	11
4.2.3.1 atualiza()	12
4.2.3.2 atualizaEntradas()	12
4.2.3.3 atualizaMotor()	12
4.2.3.4 atualizaPID()	12
4.2.3.5 desligar()	12
4.2.3.6 irPara()	12
4.2.3.7 leitura()	13
4.2.3.8 ligar()	13
4.2.3.9 pousar()	13
4.2.3.10 setMotor()	13
4.2.4 Documentação dos dados membro	14
4.2.4.1 controle	14
4.2.4.2 inercial	14
4.2.4.3 motor	14
4.2.4.4 pidPitch	14

4.2.4.5 registro	14
4.2.4.6 varPIDPitch	14
4.2.4.7 voltmetro	15
4.3 Inercial	15
4.3.1 Descrição detalhada	16
4.3.2 Documentação dos Construtores & Destrutor	16
4.3.2.1 Inercial()	16
4.3.3 Documentação dos métodos	16
4.3.3.1 getPitch()	16
4.3.3.2 getRoll()	17
4.3.3.3 getYaw()	17
4.3.3.4 leitura()	17
4.3.4 Documentação das classes amigas e funções relacionadas	17
4.3.4.1 Registro	17
4.3.5 Documentação dos dados membro	17
4.3.5.1 filtroPitch	18
4.3.5.2 filtroRoll	18
4.3.5.3 filtroYaw	18
4.3.5.4 pitch	18
4.3.5.5 roll	18
4.3.5.6 yaw	18
4.4 Motor	19
4.4.1 Descrição detalhada	19
4.4.2 Documentação dos Construtores & Destrutor	19
4.4.2.1 Motor()	20
4.4.3 Documentação dos métodos	20
4.4.3.1 desligar()	20
4.4.3.2 girar() [1/2]	20
4.4.3.3 girar() [2/2]	20
4.4.3.4 ligar()	20
4.4.3.5 setMinimo()	21
4.4.4 Documentação das classes amigas e funções relacionadas	21
4.4.4.1 Registro	21
4.4.5 Documentação dos dados membro	21
4.4.5.1 minimo	21
4.4.5.2 velocidade	21
4.5 PassaBaixa	22
4.5.1 Descrição detalhada	22
4.5.2 Documentação dos Construtores & Destrutor	22
4.5.2.1 PassaBaixa()	23
4.5.3 Documentação dos métodos	23
4.5.3.1 calculaConstante()	23

4.5.3.2 entrar()	23
4.5.3.3 sair()	23
4.5.3.4 setAmostragem()	24
4.5.3.5 setFrequencia()	24
4.5.4 Documentação dos dados membro	24
4.5.4.1 amostragem	24
4.5.4.2 constante	24
4.5.4.3 freqCorte	25
4.5.4.4 saida	25
4.5.4.5 saidaAnt	25
4.6 Registro	25
4.6.1 Descrição detalhada	26
4.6.2 Documentação dos Construtores & Destrutor	26
4.6.2.1 Registro()	26
4.6.3 Documentação dos métodos	27
4.6.3.1 print()	27
4.6.3.2 printAngulo()	27
4.6.3.3 printPID()	27
4.6.3.4 printTensao()	27
4.6.3.5 printVelocidade()	27
4.6.3.6 setAngulo()	27
4.6.3.7 setTensao()	28
4.6.3.8 setVelocidade()	28
4.6.4 Documentação dos dados membro	28
4.6.4.1 controle	29
4.6.4.2 imprimeAngulo	29
4.6.4.3 imprimePID	29
4.6.4.4 imprimeVelocidade	29
4.6.4.5 pitch	29
4.6.4.6 roll	29
4.6.4.7 tensao	30
4.6.4.8 vel0	30
4.6.4.9 vel1	30
4.6.4.10 vel2	30
4.6.4.11 vel3	30
4.6.4.12 yaw	30
4.7 VariaveisPID	31
4.7.1 Descrição detalhada	31
4.7.2 Documentação dos dados membro	31
4.7.2.1 entrada	31
4.7.2.2 kd	31
4.7.2.3 ki	32

4.7.2.4 kp	32
4.7.2.5 saida	32
4.7.2.6 setPoint	32
4.8 Voltmetro	32
4.8.1 Descrição detalhada	33
4.8.2 Documentação dos Construtores & Destrutor	33
4.8.2.1 Voltmetro()	33
4.8.3 Documentação dos métodos	33
4.8.3.1 leitura()	33
4.8.3.2 setPorta()	33
4.8.4 Documentação das classes amigas e funções relacionadas	34
4.8.4.1 Registro	34
4.8.5 Documentação dos dados membro	34
4.8.5.1 porta	34
4.8.5.2 tensao	34
5 Documentação do ficheiro	35
5.1 Referência ao ficheiro 1Classe.ino	35
5.2 Referência ao ficheiro 2Variaveis.ino	35
5.3 Referência ao ficheiro 3Funcoes.ino	35
5.4 Referência ao ficheiro 4SetupLoop.ino	35
5.4.1 Documentação das funções	36
5.4.1.1 loop()	36
5.4.1.2 setup()	36
5.5 Referência ao ficheiro NovoDrone.ino	36
5.5.1 Documentação das macros	36
5.5.1.1 imprimeSensor	37
5.5.1.2 KDpitch	37
5.5.1.3 Klpitch	37
5.5.1.4 KPpitch	37
5.5.1.5 LIGA	37
5.5.1.6 outputLIMIT	37
5.5.1.7 QUANT_MOTOR	38
5.5.1.8 RODA	38
5.5.1.9 serial	38
5.5.1.10 VOO	38
Índice	39

Capítulo 1

Índice da hierarquia

1.1 Hierarquia de classes

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

ControleSerial	7
Drone	10
MPU6050	
Inercial	15
PassaBaixa	22
Registro	25
Servo	
Motor	19
VariaveisPID	31
Voltmetro	32

Capítulo 2

Índice dos componentes

2.1 Lista de componentes

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

ControleSerial	Realiza o envio e recebimento de dados da porta serial, que pode ser definida	7
Drone	Implementa o controle de um drone	10
Inercial	Classe para controle de uma IMU do modelo "MPU6050"	15
Motor	Realiza o controle de um motor	19
PassaBaixa	Implementa um filtro de frequência do tipo "passa baixa"	22
Registro	Implementa um registro/log para dados do drone	25
VariaveisPID	Estrutura para armazenar dados de um controlador PID	31
Voltmetro	Implementa um voltímetro conectado a uma porta analógica do Arduino	32

Capítulo 3

Índice dos ficheiros

3.1 Lista de ficheiros

Lista de todos os ficheiros com uma breve descrição:

1Classe.ino	35
2Variaveis.ino	35
3Funcoes.ino	35
4SetupLoop.ino	35
NovoDrone.ino	36

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

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

<i>uso</i>	Porta desejada: 0 = Padrão do Arduino; 1 = Porta definida pelo usuário
------------	--

4.1.3.2 print()

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

Imprime um valor dado.

Elton, 14/08/2019.

Parâmetros de template

<i>tipo</i>	Tipo do valor.
-------------	----------------

Parâmetros

<i>texto</i>	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

<i>tipo</i>	Tipo do valor.
-------------	----------------

Parâmetros

<i>texto</i>	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

<i>serial2</i>	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 definida 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 mantém 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`
- `Voltmetro voltmetro`
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

<i>angulo</i>	O ângulo desejado em [°].
---------------	---------------------------

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

<i>indice</i>	Índice do motor a ser definido.
<i>porta</i>	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 voltmetro

`Voltmetro` `voltmetro` `[protected]`

Objeto para controle da comunicação do drone.

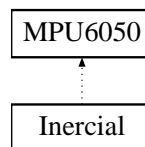
A documentação para esta classe foi gerada a partir do seguinte ficheiro:

- [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

<i>w</i>	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/s²].

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/s²].

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.

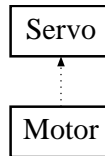
A documentação para esta classe foi gerada a partir do seguinte ficheiro:

- [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

<i>diferenca</i>	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

<i>minimo</i>	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

<i>medida</i>	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

<i>amostragem</i>	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

<i>freqCorte</i>	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 angulares 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

<i>controle</i>	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 angulares do robô.

Elton, 14/08/2019.

Parâmetros

<i>roll</i>	Ponteiro para a variável que contém a leitura do eixo "roll".
<i>pitch</i>	Ponteiro para a variável que contém a leitura do eixo "pitch".
<i>yaw</i>	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

<i>tensao</i>	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

<i>vel0</i>	Ponteiro para a variável que contém a leitura do motor 0.
<i>vel1</i>	Ponteiro para a variável que contém a leitura do motor 1.
<i>vel2</i>	Ponteiro para a variável que contém a leitura do motor 2.
<i>vel3</i>	Ponteiro para a variável que contém a leitura do motor 3.

4.6.4 Documentação dos dados membro

4.6.4.1 controle

```
ControleSerial* 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 Voltmetro

Implementa um voltímetro conectado a uma porta analógica do Arduino

Membros públicos

- [Voltmetro](#) ()
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 Voltmetro()

```
Voltmetro ( ) [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

<i>porta</i>	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 [Voltmetro](#)
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 proporcional 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 proporcional 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

- 1Classe.ino, [35](#)
- 2Variaveis.ino, [35](#)
- 3Funcoes.ino, [35](#)
- 4SetupLoop.ino, [35](#)
 - loop, [36](#)
 - setup, [36](#)
- amostragem
 - PassaBaixa, [24](#)
- atualiza
 - Drone, [11](#)
- atualizaEntradas
 - Drone, [12](#)
- atualizaMotor
 - Drone, [12](#)
- atualizaPID
 - Drone, [12](#)
- calculaConstante
 - PassaBaixa, [23](#)
- constante
 - PassaBaixa, [24](#)
- controle
 - Drone, [14](#)
 - Registro, [28](#)
- ControleSerial, [7](#)
 - ControleSerial, [8](#)
 - iniciar, [8](#)
 - print, [8](#)
 - println, [9](#)
 - serial2, [10](#)
 - setSerial, [9](#)
 - usando, [10](#)
- desligar
 - Drone, [12](#)
 - Motor, [20](#)
- Drone, [10](#)
 - atualiza, [11](#)
 - atualizaEntradas, [12](#)
 - atualizaMotor, [12](#)
 - atualizaPID, [12](#)
 - controle, [14](#)
 - desligar, [12](#)
 - Drone, [11](#)
 - inercial, [14](#)
 - irPara, [12](#)
 - leitura, [13](#)
 - ligar, [13](#)
 - motor, [14](#)
 - pidPitch, [14](#)
 - pousar, [13](#)
 - registro, [14](#)
 - setMotor, [13](#)
 - varPIDPitch, [14](#)
 - voltmetro, [14](#)
- entrada
 - VariaveisPID, [31](#)
- entrar
 - PassaBaixa, [23](#)
- filtroPitch
 - Inercial, [17](#)
- filtroRoll
 - Inercial, [18](#)
- filtroYaw
 - Inercial, [18](#)
- freqCorte
 - PassaBaixa, [24](#)
- getPitch
 - Inercial, [16](#)
- getRoll
 - Inercial, [16](#)
- getYaw
 - Inercial, [17](#)
- girar
 - Motor, [20](#)
- imprimeAngulo
 - Registro, [29](#)
- imprimePID
 - Registro, [29](#)
- imprimeSensor
 - NovoDrone.ino, [36](#)
- imprimeVelocidade
 - Registro, [29](#)
- Inercial, [15](#)
 - filtroPitch, [17](#)
 - filtroRoll, [18](#)
 - filtroYaw, [18](#)
 - getPitch, [16](#)
 - getRoll, [16](#)
 - getYaw, [17](#)
 - Inercial, [16](#)
 - leitura, [17](#)
 - pitch, [18](#)
 - Registro, [17](#)
 - roll, [18](#)

- yaw, 18
- inercial
 - Drone, 14
- iniciar
 - ControleSerial, 8
- irPara
 - Drone, 12
- kd
 - VariaveisPID, 31
- KDpitch
 - NovoDrone.ino, 37
- ki
 - VariaveisPID, 31
- KIpitch
 - NovoDrone.ino, 37
- kp
 - VariaveisPID, 32
- KPpitch
 - NovoDrone.ino, 37
- leitura
 - Drone, 13
 - Inercial, 17
 - Voltmetro, 33
- LIGA
 - NovoDrone.ino, 37
- ligar
 - Drone, 13
 - Motor, 20
- loop
 - 4SetupLoop.ino, 36
- minimo
 - Motor, 21
- Motor, 19
 - desligar, 20
 - girar, 20
 - ligar, 20
 - minimo, 21
 - Motor, 19
 - Registro, 21
 - setMinimo, 21
 - velocidade, 21
- motor
 - Drone, 14
- NovoDrone.ino, 36
 - imprimeSensor, 36
 - KDpitch, 37
 - KIpitch, 37
 - KPpitch, 37
 - LIGA, 37
 - outputLIMIT, 37
 - QUANT_MOTOR, 37
 - RODA, 38
 - serial, 38
 - VOO, 38
- outputLIMIT
 - NovoDrone.ino, 37
- PassaBaixa, 22
 - amostragem, 24
 - calculaConstante, 23
 - constante, 24
 - entrar, 23
 - freqCorte, 24
 - PassaBaixa, 22
 - saida, 25
 - saidaAnt, 25
 - sair, 23
 - setAmostragem, 23
 - setFrequencia, 24
- pidPitch
 - Drone, 14
- pitch
 - Inercial, 18
 - Registro, 29
- porta
 - Voltmetro, 34
- pousar
 - Drone, 13
- print
 - ControleSerial, 8
 - Registro, 27
- printAngulo
 - Registro, 27
- println
 - ControleSerial, 9
- printPID
 - Registro, 27
- printTensao
 - Registro, 27
- printVelocidade
 - Registro, 27
- QUANT_MOTOR
 - NovoDrone.ino, 37
- Registro, 25
 - controle, 28
 - imprimeAngulo, 29
 - imprimePID, 29
 - imprimeVelocidade, 29
 - Inercial, 17
 - Motor, 21
 - pitch, 29
 - print, 27
 - printAngulo, 27
 - printPID, 27
 - printTensao, 27
 - printVelocidade, 27
 - Registro, 26
 - roll, 29
 - setAngulo, 27
 - setTensao, 28
 - setVelocidade, 28
 - tensao, 29

- vel0, [30](#)
- vel1, [30](#)
- vel2, [30](#)
- vel3, [30](#)
- Voltmetro, [34](#)
- yaw, [30](#)
- registro
 - Drone, [14](#)
- RODA
 - NovoDrone.ino, [38](#)
- roll
 - Inercial, [18](#)
 - Registro, [29](#)
- saida
 - PassaBaixa, [25](#)
 - VariaveisPID, [32](#)
- saidaAnt
 - PassaBaixa, [25](#)
- sair
 - PassaBaixa, [23](#)
- serial
 - NovoDrone.ino, [38](#)
- serial2
 - ControleSerial, [10](#)
- setAmostragem
 - PassaBaixa, [23](#)
- setAngulo
 - Registro, [27](#)
- setFrequencia
 - PassaBaixa, [24](#)
- setMinimo
 - Motor, [21](#)
- setMotor
 - Drone, [13](#)
- setPoint
 - VariaveisPID, [32](#)
- setPorta
 - Voltmetro, [33](#)
- setSerial
 - ControleSerial, [9](#)
- setTensao
 - Registro, [28](#)
- setup
 - 4SetupLoop.ino, [36](#)
- setVelocidade
 - Registro, [28](#)
- tensao
 - Registro, [29](#)
 - Voltmetro, [34](#)
- usando
 - ControleSerial, [10](#)
- VariaveisPID, [31](#)
 - entrada, [31](#)
 - kd, [31](#)
 - ki, [31](#)
 - kp, [32](#)
 - saida, [32](#)
 - setPoint, [32](#)
- varPIDPitch
 - Drone, [14](#)
- vel0
 - Registro, [30](#)
- vel1
 - Registro, [30](#)
- vel2
 - Registro, [30](#)
- vel3
 - Registro, [30](#)
- velocidade
 - Motor, [21](#)
- Voltmetro, [32](#)
 - leitura, [33](#)
 - porta, [34](#)
 - Registro, [34](#)
 - setPorta, [33](#)
 - tensao, [34](#)
 - Voltmetro, [33](#)
- voltmetro
 - Drone, [14](#)
- VOO
 - NovoDrone.ino, [38](#)
- yaw
 - Inercial, [18](#)
 - Registro, [30](#)