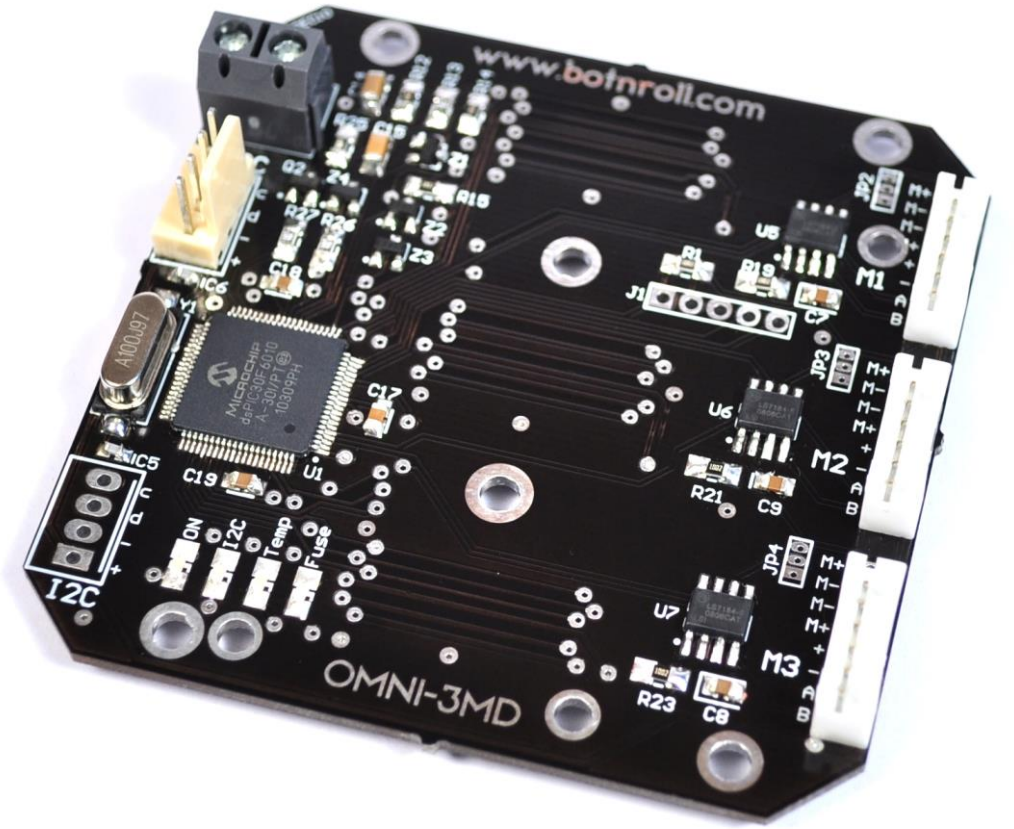
OMNI-3MD

Placa Controladora de 3 Motores

Disponível Biblioteca para Arduino



Manual de Software

Versão de firmware 1.73.00 de Agosto de 2013

1 Índice

2 Visão Geral da OMNI-3MD

A OMNI-3MD é um dispositivo I2C SLAVE capaz de movimentar 3 motores DC 7V a 24V e correntes até 4A por motor. Através de *encoders*, efetua a movimentação dos motores com controlo em malha fechada PID. Um processador dsPIC de 16bits a 40MHz permite vários modos de movimentação dos motores, nomeadamente:

- Movimentação omnidirecional de 3 motores (concêntricos, com a mesma distância ao centro e desfasados 120º) com controlo P ID.
- Movimentação diferencial de 2 motores com controlo PID usando unidades do sistema internacional (SI).
- Movimentação linear de 1, 2 ou 3 motores com/sem controlo PID.
- Movimentação posicional de 1, 2 ou 3 motores com controlo PID.

Para mais detalhes de relacionados com as funcionalidades de *hardware* da OMNI-3MD consulte o manual de hardware disponível em http://www.botnroll.com/omni3md/

3 Comunicação com a OMNI-3MD

A comunicação com a OMNI-3MD efetua-se via barramento I2C. A placa controladora possui por defeito o endereço I2C de 7 bits 0x18 e recebe comandos enviados de 100KHz a 400KHz. Responde também ao endereço de *broadcast* 0x00.

A comunicação com a OMNI-3MD efetua-se por tramas enviadas para o barramento I2C e podem ser de escrita ou de leitura. Uma trama é constituída por vários bytes, que de uma forma geral, respeita o seguinte formato:

ADDRESS	W/R	COMMAND	BYTE1	BYTE2	BYTE3	BYTE4	BYTE5	…	BYTEn
0x18	W=0		**Data Bytes**						

ADDRESS: É o campo do endereço I2C. Este primeiro byte contém os 7 bits do endereço I2C e o oitavo bit, o menos significativo, define se a trama é de escrita ou de leitura. Se o bit menos significativo do campo endereço for 0 (zero), a trama é de escrita e o valor do byte é 0x30. Se o bit for 1 a trama é de leitura e o valor do byte é 0x31. Em resumo:

- Endereço I2C de 7 bits: 0x18
- Endereço I2C de 8 bits: 0x30 – trama de escrita
- Endereço I2C de 8 bits: 0x31 – trama de leitura

COMMAND: É o campo do comando I2C. O comando especifica a acção que a OMNI-3MD terá que efetuar.

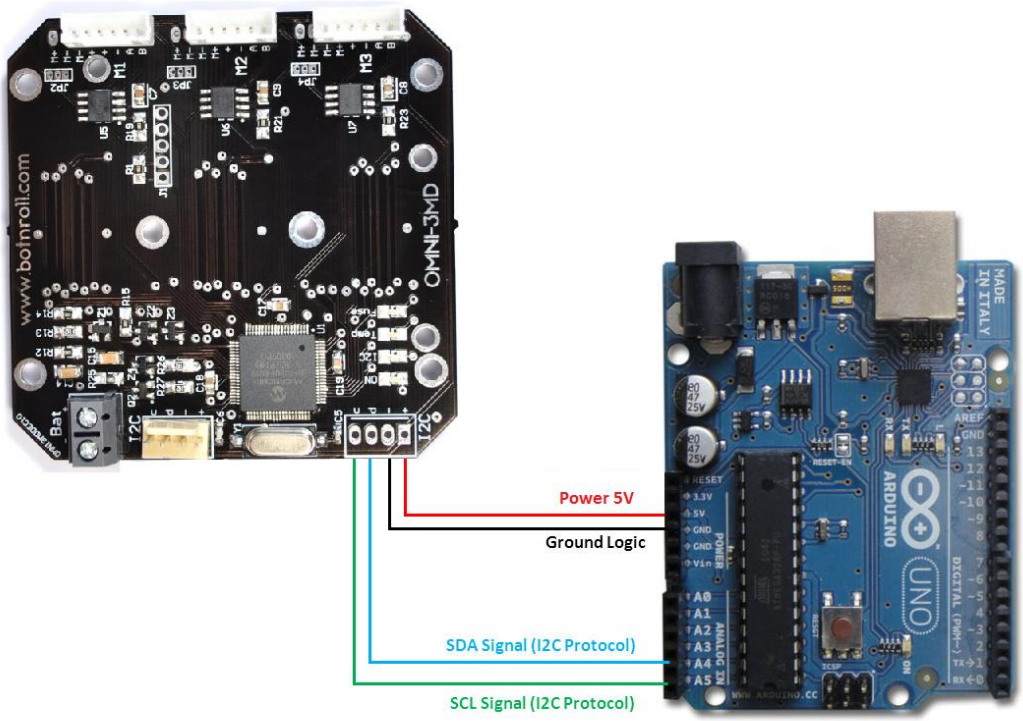
Data Bytes: São os bytes de dados associados a cada comando I2C. Alguns comandos estão associados a duas "**chaves**", bytes 0xAA e 0x55. Estas chaves têm a finalidade de aumentar a fiabilidade da comunicação.

Com o intuito de facilitar a interação com a OMNI-3MD foi desenvolvida uma biblioteca para programação em C pensada para o Arduino IDE. A **biblioteca Omni3MD** é open-source e pode ser descarregada na página de suporte da placa controladora Omni-3MD:

Download da biblioteca Omni-3MD: http://www.botnroll.com/omni3md/

Esta biblioteca permite ao utilizador uma utilização de mais alto nível da Omni-3MD que possibilita a programadores menos experientes obter uma interação com a OMNI-3MD de forma fácil e rápida. Os exemplos da biblioteca abordam todos os comandos existentes. A placa controladora Arduino e a Omni-3MD deverão ser ligadas por I2C.

3.1 Exemplo de ligação I2C entre OMNI-3MD e Arduino



4 A biblioteca Omni3MD

São aqui descritas todas as funções existentes na biblioteca Omni3MD. A cada função está associado um comando e respectivos parâmetros representados pelos *bytes* de dados da trama I2C anteriormente especificada. Existem funções de configuração, de leitura e de movimentação. Todas estas funções e os respectivos comandos estão descritos no ficheiro Omni3MD.h e codificados no ficheiro Omni3MD.cpp da biblioteca Omni3MD.

4.1 Funções de configuração

4.1.1 void i2c_connect(byte omniAddress);

Esta função deverá ser executada uma única vez, na rotina setup() do Arduino e antes de todas as outras funções da biblioteca Omni3MD: configura o barramento I2C do Arduino para comunicar com a OMNI-3MD recorrendo à biblioteca wire.h.

O parâmetro omniAddress é o endereço I2C de 8 bits para escrita, por defeito 0x30.

Exemplo:

```
#include <Wire.h>          //required by Omni3MD.cpp
#include <Omni3MD.h>
#define OMNI3MD_ADDRESS 0x30  //default factory address
Omni3MD omni;              //declaration of object variable to control the Omni3MD

void setup()
{
    omni.i2c_connect(OMNI3MD_ADDRESS);  //set i2c connection
}
```

Sendo esta função do tipo void nenhum valor é devolvido pela rotina no final da sua execução.

4.1.2 void set_i2c_address(byte newAddress);

Esta rotina permite que se altere o endereço I2C da Omni-3MD sempre que necessário.

O parâmetro newAddress define o novo endereço da Omni-3MD. As especificações do protocolo I2C definem que são válidos os endereços de 7bits entre 0x08 e 0x77, ou seja, deverão ser enviados para a Omni-3MD endereços pares de 8 bits entre 0x10 (16) e 0xEE (238).

Exemplo:

```
#include <Omni3MD.h>
Omni3MD omni;            //declaration of object variable to control the Omni3MD

void setup()
{
        ...
        omni.set_i2c_address(0x2E);
        delay(10);          // 10ms pause required for Omni3MD EEPROM writing
        ...
}
```

O endereço I2C recebido é armazenado em memória EEPROM e não se perde a informação ao desligar a Omni-3MD. A escrita na EEPROM deste parâmetro demora cerca de 5ms e durante este tempo a Omni-3MD não responde a comandos. O delay de 10 ms presente no exemplo contempla a escrita na EEPROM e evita que o Arduino envie comandos durante esse tempo.

Sendo esta função do tipo void nenhum valor é devolvido pela rotina no final da sua execução.

4.1.3 void set_i2c_timeout(byte timeout);

O timeout I2C é um mecanismo de proteção contra falhas de comunicação e consequentemente uma medida importante de segurança. Se a Omni-3MD não receber dados I2C durante um certo tempo (*timeout*), pára os motores evitando assim possíveis danos e acidentes que possam ocorrer no sistema onde a Omni-3MD é aplicada. O parâmetro timeout será internamente multiplicado por 10 milissegundos e define o tempo de *timeout* I2C. Se desejarmos que o tempo de *timeout* seja de 500ms, por exemplo, deveremos enviar o valor 50 como parâmetro na chamada a esta função.

Este mecanismo pode ser desactivado enviando o valor 0 como parâmetro.

Exemplo:

```
#include <Omni3MD.h>
Omni3MD omni;            //declaration of object variable to control the Omni3MD

void setup()
{
        ...
        omni.set_i2c_timeout(50);   //500ms timeout
        delay(10);          // 10ms pause required for Omni3MD EEPROM writing
        ...
}
```

Sendo timeout um byte, podemos enviar valores entre 0 e 255. O tempo máximo de timeout é de 2550ms, ou seja, cerca de dois segundos e meio. O valor de timeout recebido é armazenado em memória EEPROM e não se perde a informação ao desligar a Omni-3MD. A escrita na EEPROM deste parâmetro demora cerca de 5ms e durante este tempo a Omni-3MD não responde a comandos.

Sendo esta função do tipo void nenhum valor é devolvido pela rotina no final da sua execução.

4.1.4 void calibrate(boolean way1, boolean way2, boolean way3);

A calibração só é necessária quando se usam motores com *encoders*. Esta rotina coloca os motores em movimento e analisa a contagem dos *encoders* com a variação da velocidade. É registada a contagem máxima dos *encoders* e definida automaticamente a taxa de controlo PID, que dependendo dos *encoders* poderá ser de 10, 20 ou 40 vezes por segundo.

Os *encoders* em quadratura fornecem informação sobre o sentido de rotação. A Omni-3MD regista o sentido de rotação da calibração como referência para o sentido positivo do movimento. A rotina de calibração permite definir qual o sentido que desejamos como sendo o positivo, para qualquer motor. Desta forma evitam-se alterações no *hardware*, nomeadamente, trocar os fios de alimentação dos motores. Ao assemblar o seu sistema, a rotação que os motores terão por defeito irá sempre depender da caixa redutora e da ligação elétrica dos motores.

Os parâmetros way1, way2 e way3 servem para definir o sentido de movimento da calibração para cada motor e o valor de cada parâmetro poderá ser 0 ou 1.

Sendo esta função do tipo void nenhum valor é devolvido pela rotina no final da sua execução.

Exemplo:

```
#include <Omni3MD.h>
Omni3MD omni;            //declaration of object variable to control the Omni3MD

void setup()
{
        ...
        omni.calibrate(1,0,1);
        delay(95000);          // wait 95s for calibration to end
}
```

A calibração deverá ser efetuada na preparação inicial do sistema ou sempre que houver alguma alteração significativa no *hardware*, nomeadamente motores ou baterias, caso as suas tensões (Volts) sejam diferentes. A calibração pode ser efetuada com os motores com carga ou sem carga (ex.: rodas no ar ou no chão), sendo esta uma decisão a ser tomada pelo utilizador de acordo com o seu sistema. Se o robô foi anteriormente calibrado no chão e se houver entretanto uma alteração significativa no peso do robô, é recomendada uma nova calibração. Os valores obtidos na calibração são armazenados em memória EEPROM e não se perdem ao desligar a Omni-3MD. O processo de calibração tem uma duração aproximada de 95 segundos na qual os motores devem rodar sem interferências externas ao sistema. Durante a calibração o LED amarelo "I2C" permanece aceso e a Omni-3MD não responde a comandos I2C.

Nota importante: Nunca desligar nem interferir com o sistema durante a calibração!

Calibração e Movimento Omnidirecional

Ao usar a Omni-3MD para movimentação omnidirecional, os parâmetros da rotina de calibração way1, way2 e way3 deverão ser configurados de forma que todos os motores rodem no mesmo sentido e que a rotação dos motores coloque o robô a rodar no sentido contrário aos ponteiros do relógio (CCW-*Counterclockwise*).

Calibração e Movimento Diferencial em Unidades do Sistema SI

Se usar a Omni-3MD para movimentação diferencial tendo como referência as unidades do sistema internacional (SI), utilize as saídas dos motores 1 e 3. Configure os parâmetros da rotina de calibração way1 e way3 de forma a que o robô se mova para a frente.

4.1.5 void set_PID(int Kp, int Ki, int Kd);

Esta rotina permite ajustar o controlo PID da Omni-3MD através do envio dos parâmetros Kp, Ki e Kd que ajustam na Omni-3MD os ganhos proporcional, integral e diferencial respetivamente.

Os ganhos do controlo PID deverão ser ajustados de forma a ser obtido o movimento desejado e pequenas variações dos ganhos podem fazer uma grande diferença no movimento. Os ganhos PID tomam habitualmente valores entre 0 e 1, mas dependendo do sistema, podem ser superiores a 1. O utilizador deve definir os ganhos PID que melhor servem o seu sistema.

Os valores de Kp, Ki e Kd deverão ser positivos e são internamente divididos por 1000 pela Omni-3MD. Para se definir um ganho proporcional de 0.65 na Omni-3MD, por exemplo, deverá ser enviado como parâmetro Kp o valor 650.

Exemplo:

```
#include <Omni3MD.h>
Omni3MD omni;            //declaration of object variable to control the Omni3MD

void setup()
{
        ...
        omni.set_PID(650,450,250);
        delay(15);          // 15ms pause required for Omni3MD EEPROM writing
        ...
}
```

Os valores de Kp, Ki e Kd recebidos são armazenados em memória EEPROM e não se perde a informação ao desligar a Omni-3MD. A escrita na EEPROM destes 3 parâmetros demora cerca de 15ms e durante este tempo a Omni-3MD não responde a comandos.

Sendo esta função do tipo void nenhum valor é devolvido pela rotina no final da sua execução.

Nota: Verifique a influência do parâmetro **Kl** da rotina void set_ramp(int time, int slope, int **Kl**) de forma a minimizar *overshoots* no caso especial de arranque com velocidade inicial 0.

4.1.6 void set_ramp(int time, int slope, int KI);

Esta função configura a rampa de aceleração através dos parâmetros time e slope. Uma rampa de aceleração bem ajustada permite um arranque eficaz, suave e com deslizamento mínimo. A rampa de aceleração implementada na Omni-3MD não é uma reta mas sim uma aproximação a uma função logarítmica em que a aceleração inicial é reduzida e aumenta substancialmente na parte final.

O parâmetro time define o tempo da rampa de aceleração em milissegundos, isto é, o tempo que demora a acelerar da velocidade inicial até à velocidade desejada. Os valores enviados deverão ser positivos. Por exemplo, para uma rampa de um segundo deverá ser enviado o valor 1000 como parâmetro time.
Para desativar a rampa de aceleração envia-se o valor 0 como parâmetro time.
Foi definido que a rampa de aceleração terá um máximo de 40 iterações do ciclo de controlo PID. Assim, o tempo máximo permitido para a rampa de aceleração é o tempo de 40 iterações da taxa de controlo PID, ou seja:

- 4 segundos para uma taxa de controlo PID de 10 vezes por segundo (40 x 100ms);
- 2 segundos para uma taxa de controlo PID de 20 vezes por segundo (40 x 50ms);
- 1 segundo para uma taxa de controlo PID de 40 vezes por segundo (40 x 25ms);

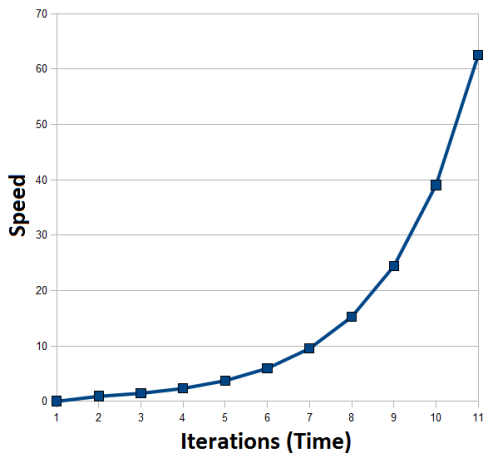
Assim, pode também inferir-se que:

- Uma rampa de um segundo terá 10 iterações se a taxa de controlo for de 10 vezes por segundo.
- Uma rampa de um segundo terá 40 iterações se a taxa de controlo for de 40 vezes por segundo.

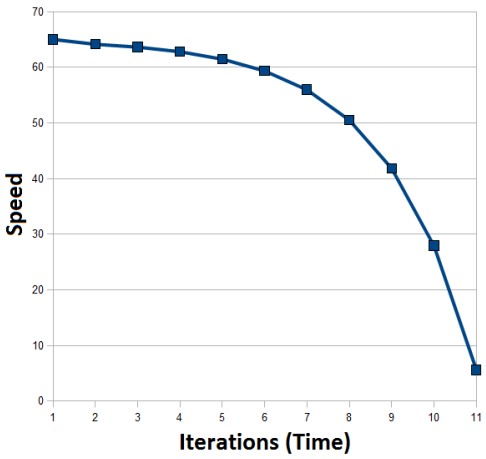
O parâmetro slope define a inclinação da rampa de aceleração e o valor enviado será dividido por 1000 pela Omni-3MD. Um valor de slope de 1500, por exemplo, corresponde a um fator de inclinação de 1.5 na Omni-3MD.
Nota importante: os valores de slope deverão ser sempre superiores a 1000.

Valores de slope mais baixos provocarão uma aceleração mais brusca na fase inicial da rampa de aceleração. Valores de slope mais elevados provocarão uma aceleração mais brusca na fase final da rampa de aceleração.

Apresenta-se abaixo um exemplo de uma rampa de aceleração e de desaceleração para a visualização gráfica dos processos:



Aceleração

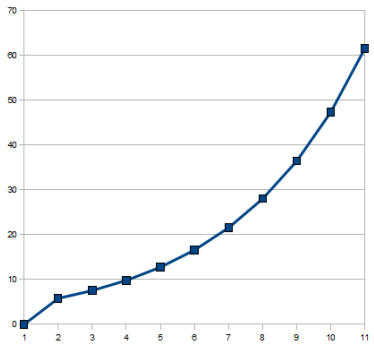


Desaceleração

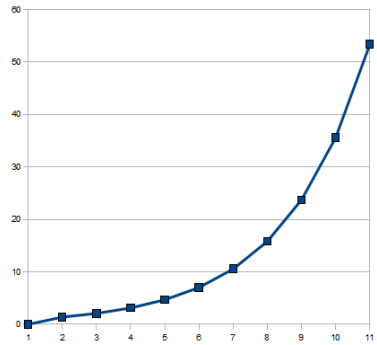
O valor de slope deverá ser também ajustado de acordo com o número de iterações (tempo) da rampa de aceleração.

De seguida apresentam-se alguns exemplos em que se acelera da velocidade inicial 0 para a velocidade final de 80 numa escala de 0 a 100. Nos exemplos apresentam-se os valores mínimo e máximo de *slope* considerados como limites. Ao centro apresenta-se a situação aconselhada para a maioria das aplicações.

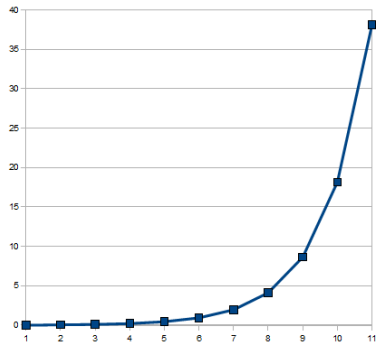
Influência do slope numa rampa de aceleração de 10 iterações:



slope = 1300

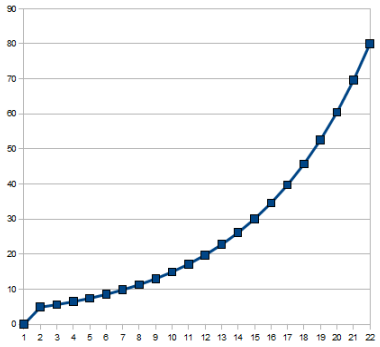


slope = 1500

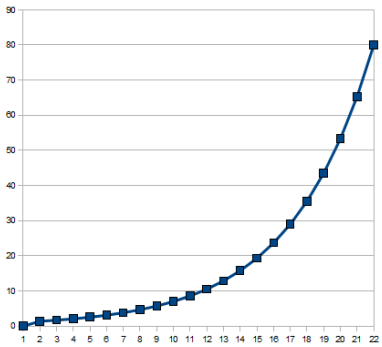


slope = 2100

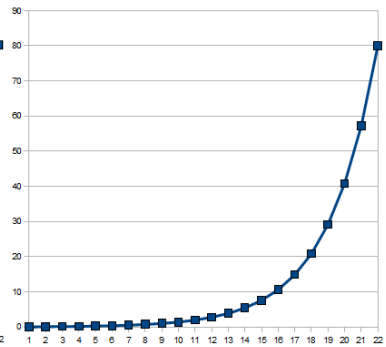
Influência do slope numa rampa de aceleração de 20 iterações:



slope = 1150

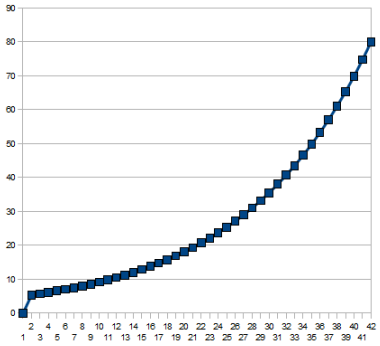


slope = 1225

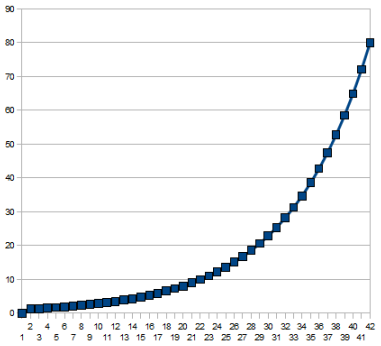


slope = 1400

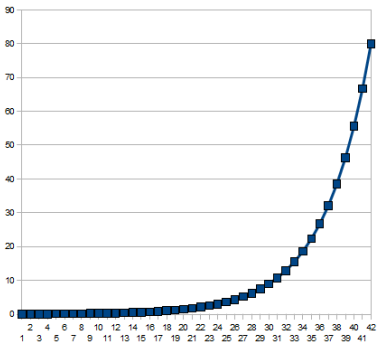
Influência do slope numa rampa de aceleração de 40 iterações:



slope = 1070



slope = 1110



slope = 1200

Notar que rampas mais curtas (com menos iterações) deverão usar valores de slope mais altos que rampas mais longas. Apresenta-se abaixo uma tabela com valores recomendados:

Iterações	1	2	3	4	5	6	7	8	9	10
slope	-	-	-	1720	1680	1642	1606	1572	1540	1500

Iterações	11	12	13	14	15	16	17	18	19	20
slope	1462	1426	1392	1360	1330	1304	1280	1260	1242	1225

Iterações	21	22	23	24	25	26	27	28	29	30
slope	1216	1207	1198	1190	1182	1174	1167	1160	1153	1147

Iterações	31	32	33	34	35	36	37	38	39	40
slope	1141	1136	1131	1127	1123	1119	1116	1113	1111	1110

O parâmetro Kl é o "fator limiar", um fator para o erro integral aplicado na primeira iteração do controlo PID no caso especial do arranque com velocidade inicial 0. Este mecanismo surgiu com o intuito de combater a não linearidade dos motores DC. Na calibração é detectada qual a potência necessária a colocar nos motores para provocar movimento. Esse valor "limiar" do movimento, detectado na calibração, será multiplicado pelo "ganho limiar" (Kl/1000) e colocado no erro integral do PID somente na primeira iteração do controlo e quando o sistema parte de velocidade inicial 0. Com isto reduz-se o tempo de arranque dos motores e o desempenho do controlo PID melhora substancialmente, reduzindo *overshoots*. Ao ajustar o controlo PID deve-se também contar com este parâmetro para otimizar a situação do arranque com velocidade inicial 0.

Os valores de Kl deverão ser positivos e normalmente variam entre 0 e 1000 que ao serem divididos por 1000 pela Omi-3MD correspondem ao ganho limiar que varia entre 0 e 1. O valor 0 de Kl desabilita este mecanismo.

Exemplo:

```
#include <Omni3MD.h>
Omni3MD omni;              //declaration of object variable to control the Omni3MD

void setup()
{
        ...
        omni.set_ramp(500,1350,800);
        delay(15);              // 15ms pause required for Omni3MD EEPROM writing
        ...
}
```

Os valores de time, slope e Kl recebidos são armazenados em memória EEPROM e não se perde a informação ao desligar a Omni-3MD. A escrita na EEPROM destes 3 parâmetros demora cerca de 15ms e durante este tempo a Omni-3MD não responde a comandos.

Sendo esta função do tipo void nenhum valor é devolvido pela rotina no final da sua execução.

4.1.7 void set_prescaler(byte encoder, byte value);

Com esta função o utilizador define o prescaler associado a cada encoder. O prescaler é um factor que define como a contagem real dos *encoders* se transforma na contagem incremental. A cada ciclo de controlo PID a contagem real dos *encoders* é transformada na contagem incremental tendo em conta o prescaler selecionado e a direção do movimento.

- Se a movimentação de um determinado motor se efetuar no sentido positivo, o valor real do encoder será somado à contagem incremental tendo em conta o prescaler definido.
- Se a movimentação de um determinado motor se efetuar no sentido negativo, o valor real do encoder será subtraído à contagem incremental tendo em conta o prescaler definido.

Para a movimentação de um determinado motor no sentido positivo:
- Um prescaler de 1 define que o valor real do encoder é somado directamente à contagem incremental, a cada ciclo de controlo PID.
- Um prescaler de 10 define que por cada 10 pulsos de contagem real do encoder, é somada 1 unidade à contagem incremental, a cada ciclo de controlo PID.
- Um prescaler de 1000 define que por cada 1000 pulsos de contagem real do encoder, é adicionada 1 unidade à contagem incremental, a cada ciclo de controlo PID.

O parâmetro encoder define o encoder ao qual se vai alterar o prescaler. Os valores válidos para este parâmetro são 1, 2 e 3 que correspondem aos motores 1, 2 e 3 respectivamente.

O parâmetro value define o prescaler a associar ao encoder. Valores válidos para este parâmetro são:

- 0: define um prescaler de 1
- 1: define um prescaler de 10
- 2: define um prescaler de 100
- 3: define um prescaler de 1000
- 4: define um prescaler de 10000

Exemplo:

```
#include <Omni3MD.h>
Omni3MD omni;            //declaration of object variable to control the Omni3MD
#define M1  1        //Motor1
#define M2  2        //Motor2
#define M3  3        //Motor3

void setup()
{
  omni.set_prescaler(M1, 0);      //sets Motor1 prescaler to 1
  delay(10);                      // 10ms pause required for Omni3MD EEPROM writing
  omni.set_prescaler(M2, 2);      //sets Motor2 prescaler to 100
  delay(10);                      // 10ms pause required for Omni3MD EEPROM writing
  omni.set_prescaler(M3, 3);      //sets Motor3 prescaler to 1000;
  delay(10);                      // 10ms pause required for Omni3MD EEPROM writing
}
```

O valor de value recebido é armazenado em memória EEPROM e não se perde a informação ao desligar a Omni-3MD. A escrita na EEPROM deste parâmetro demora cerca de 5ms e durante este tempo a Omni-3MD não responde a comandos.

Sendo esta função do tipo void nenhum valor é devolvido pela rotina no final da sua execução.

4.1.8 void set_differential(double axis_radius, double whell_radius, double gearbox_factor, double encoder_cpr);

Esta configuração é necessária quando se pretende movimento diferencial usando unidades do sistema internacional (SI). Neste tipo de movimento são usadas as unidades de velocidade linear em metros por segundo (m/s) e velocidade angular (rotação) em radianos por segundo (rad/s).

Para que o movimento se efetue corretamente, é necessário configurar alguns parâmetros que correspondem a características físicas do robô:

axis_radius : Raio do eixo de rotação do robô, ou seja, metade da distância entre as rodas motrizes. O valor deverá ser introduzido em milímetros.

whell_radius : Raio das rodas motrizes, em milímetros.

gearbox_factor : Fator de redução da caixa redutora do motor DC. Se a caixa redutora tem uma redução de 50:1, ou seja, a cada 50 voltas do motor corresponde uma volta da roda, deverá ser introduzido o valor 50.

encoder_cpr : É o número de pulsos gerado pelo encoder em quadratura para uma volta do motor DC (não da roda).

Exemplo:

```
#include <Omni3MD.h>
Omni3MD omni;            //declaration of object variable to control the Omni3MD

void setup()
{
  ...
    omni.set_differential(97.5,37.5,29,60);
    delay(20);           // 20ms pause required for Omni3MD EEPROM writing
  ...
}
```

Os valores de axis_radius, whell_radius, gearbox_factor e encoder_cpr são armazenados em memória EEPROM e não se perde a informação ao desligar a Omni-3MD. A escrita na EEPROM destes 4 parâmetros demora cerca de 20ms e durante este tempo a Omni-3MD não responde a comandos.

Sendo esta função do tipo void nenhum valor é devolvido pela rotina no final da sua execução.

4.2 Funções de leitura

4.2.1 float read_temperature();

Leitura da temperatura da Omni-3MD. É devolvida a temperatura em graus Celsius que deverá ser armazenada numa variável do tipo float.

Exemplo:

```
#include <Omni3MD.h>
Omni3MD omni;              //declaration of object variable to control the Omni3MD
float temperature=0.0;  // temperature reading

void loop()
{
    temperature=omni.read_temperature();  // read temperature value
}
```

4.2.2 float read_battery();

Leitura da tensão de alimentação dos motores da Omni-3MD, que é normalmente fornecida por uma bateria. É devolvida a tensão em Volt (V) que deverá ser armazenada numa variável do tipo float.

Exemplo:

```
#include <Omni3MD.h>
Omni3MD omni;     //declaration of object variable to control the Omni3MD
float battery=0.0;      // battery reading

void loop()
{
    battery=omni.read_battery();       // read battery value
}
```

4.2.3 void read_firmware(byte*,byte*,byte*);

Leitura da versão do software da Omni-3MD. A versão do software possui 3 campos e à data da escrita deste manual, a versão 1.73.00 era a mais atualizada. Os parâmetros desta função recebem a leitura dos valores por meio de apontadores e são recebidos 3 bytes.

Exemplo:

```
#include <Omni3MD.h>
Omni3MD omni;     //declaration of object variable to control the Omni3MD
byte firm_rel=0;     // the firmware release field
byte firm_int=0;     // the firmware intermediary field
byte firm_dev=0;     // the firmware development field

void loop()
{
    omni.read_firmware(&firm_rel,&firm_int,&firm_dev); // read firmware version value
}
```

4.2.4 float read_firmware();

Esta rotina foi mantida na biblioteca Omni-3MD por questões de compatibilidade com versões de firmware anteriores ao ano de 2012 em que são usados somente 2 campos para a identificação da versão do software. A versão do software é devolvida numa variável do tipo float.

Exemplo:

```
#include <Omni3MD.h>
Omni3MD omni;      //declaration of object variable to control the Omni3MD
float firmware=0.0;     //

void loop()
{
    firmware=omni.read_firmware(); // read firmware version
}
```

4.2.5 byte read_control_rate();

Leitura da taxa de controlo definida pela rotina de calibração da Omni-3MD. Esta rotina devolve um byte.

Exemplo:

```
#include <Omni3MD.h>
Omni3MD omni;      //declaration of object variable to control the Omni3MD
byte ctrl_rate=0;       // the control rate for your motors defined at calibration

void loop()
{
    ctrl_rate=omni.read_control_rate();   // read the control rate value
}
```

4.2.6 int read_enc1_max();

Leitura do valor máximo de pulsos do encoder1 detetado durante a calibração, para a taxa de controlo definida. Esta rotina devolve uma variável do tipo inteiro de 16bits.

Exemplo:

```
#include <Omni3MD.h>
Omni3MD omni;      //declaration of object variable to control the Omni3MD
int enc1_max;      // maximum count for encoder1 at calibration, for the defined control rate

void loop()
{
    enc1_max=omni.read_enc1_max();      // read encoder1 maximum value at calibration
}
```

4.2.7 int read_enc2_max();

Leitura do valor máximo de pulsos do encoder2 detetado durante a calibração, para a taxa de controlo definida. Esta rotina devolve uma variável do tipo inteiro de 16bits.

Exemplo:

```
#include <Omni3MD.h>
Omni3MD omni;      //declaration of object variable to control the Omni3MD
int enc2_max;      // maximum count for encoder2 at calibration, for the defined control rate

void loop()
{
    enc2_max=omni.read_enc2_max();      // read encoder2 maximum value at calibration
}
```

4.2.8 int read_enc3_max();

Leitura do valor máximo de pulsos do encoder3 detetado durante a calibração, para a taxa de controlo definida. Esta rotina devolve uma variável do tipo inteiro de 16bits.

Exemplo:

```
#include <Omni3MD.h>
Omni3MD omni;      //declaration of object variable to control the Omni3MD
int enc3_max;      // maximum count for encoder3 at calibration, for the defined control rate

void loop()
{
    enc3_max=omni.read_enc3_max();      // read encoder3 maximum value at calibration
}
```

4.2.9 int read_enc1();

Leitura do valor da contagem incremental do encoder1. A cada ciclo de controlo PID, a contagem incremental é atualizada tendo em conta a contagem real do encoder, o *prescaler* e a direção do movimento. Esta contagem pode, por exemplo, representar um deslocamento no espaço e o utilizador pode usar esta informação para controlar o movimento do seu robô/sistema.

Esta rotina devolve uma variável do tipo inteiro de 16bits que varia entre -32768 e 32767.

Exemplo:

```
#include <Omni3MD.h>
Omni3MD omni;      //declaration of object variable to control the Omni3MD
int enc1=0;        // encoder1 reading, this is the encoder incremental count for the defined prescaler

void loop()
{
    enc1=omni.read_enc1();          // read encoder1 incremental value
}
```

4.2.10 int read_enc2();

Leitura do valor da contagem incremental do encoder2. A cada ciclo de controlo PID, a contagem incremental é atualizada tendo em conta a contagem real do *encoder*, o *prescaler* e a direção do movimento. Esta contagem pode, por exemplo, representar um deslocamento no espaço e o utilizador pode usar esta informação para controlar o movimento do seu robô/sistema.

Esta rotina devolve uma variável do tipo inteiro de 16bits que varia entre -32768 e 32767.

Exemplo:

```
#include <Omni3MD.h>
Omni3MD omni;      //declaration of object variable to control the Omni3MD
int enc2=0;        // encoder2 reading, this is the encoder incremental count for the defined prescaler

void loop()
{
    enc2=omni.read_enc2();          // read encoder2 incremental value
}
```

4.2.11 int read_enc3();

Leitura do valor da contagem incremental do encoder3. A cada ciclo de controlo PID, a contagem incremental é atualizada tendo em conta a contagem real do *encoder*, o *prescaler* e a direção do movimento. Esta contagem pode, por exemplo, representar um deslocamento no

espaço e o utilizador pode usar esta informação para controlar o movimento do seu robô/sistema.

Esta rotina devolve uma variável do tipo inteiro de 16bits que varia entre -32768 e 32767.

Exemplo:

```
#include <Omni3MD.h>
Omni3MD omni;      //declaration of object variable to control the Omni3MD
int enc3=0;        // encoder3 reading, this is the encoder incremental count for the defined prescaler

void loop()
{
    enc3=omni.read_enc3();          // read encoder3 incremental value
}
```

4.2.12 void read_encoders(int*,int*,int*);

Leitura do valor da contagem incremental dos *encoders* dos 3 motores. Esta função otimiza a comunicação I2C e numa só transmissão I2C recebem-se os valores de contagem incremental dos 3 *encoders*. Os parâmetros desta função recebem a leitura dos valores por meio de apontadores e são recebidos 3 inteiros de 16bits.

Exemplo:

```
#include <Omni3MD.h>
Omni3MD omni;      //declaration of object variable to control the Omni3MD
int enc1=0;        // encoder1 reading, this is the encoder incremental count for the defined prescaler
int enc2=0;        // encoder2 reading, this is the encoder incremental count for the defined prescaler
int enc3=0;        // encoder3 reading, this is the encoder incremental count for the defined prescaler

void loop()
{
    omni.read_encoders(&enc1,&enc2,&enc3); // read incremental value for 3 encoders at once
}
```

4.2.13 void read_mov_data(int*,int*,int*,float*,float*);

De uma só vez, efetua a leitura dos parâmetros que estão diretamente relacionados com a movimentação. Numa só transmissão I2C recebe a leitura do valor incremental dos 3 *encoders*, o valor da tensão de alimentação dos motores e a temperatura da Omni-3MD. Todos estes valores são colocados nas variáveis (parâmetros) por intermédio de apontadores. Esta rotina reduz significativamente o tempo de comunicação comparativamente à leitura individual destes mesmos valores, otimizando o uso do barramento I2C.

Exemplo:

```
#include <Omni3MD.h>
Omni3MD omni;      //declaration of object variable to control the Omni3MD
int enc1=0;        // encoder1 reading, this is the encoder incremental count for the defined prescaler
int enc2=0;        // encoder2 reading, this is the encoder incremental count for the defined prescaler
int enc3=0;        // encoder3 reading, this is the encoder incremental count for the defined prescaler
```

```
float battery=0.0;      // battery reading
float temperature=0.0;   // temperature reading

void loop()
{
   omni.read_mov_data(&enc1,&enc2,&enc3,&battery,&temperature);
}
```

4.2.14 void read_all_data(int*,int*,int*,float*,float*,byte*,byte*,byte*,byte*,int*,int*,int*);

Leitura de todos os parâmetros possíveis de serem lidos da Omni-3MD. Numa só transmissão I2C efetua a leitura do valor incremental dos três *encoders*, o valor da tensão de alimentação dos motores, a temperatura da Omni-3MD, o três campos do firmware, a taxa de controlo e os valores máximos de pulsos dos *encoders* detetados durante a calibração. Esta rotina reduz significativamente o tempo de comunicação comparativamente à leitura individual destes mesmos valores, otimizando o uso do barramento I2C.

Exemplo:

```
#include <Omni3MD.h>
Omni3MD omni;      //declaration of object variable to control the Omni3MD
int enc1=0;          // encoder1 reading, this is the encoder incremental count for the defined prescaler
int enc2=0;          // encoder2 reading, this is the encoder incremental count for the defined prescaler
int enc3=0;          // encoder3 reading, this is the encoder incremental count for the defined prescaler
float battery=0.0;      // battery reading
float temperature=0.0;   // temperature reading
byte firm_rel=0;      // the firmware release field
byte firm_int=0;      // the firmware intermediary field
byte firm_dev=0;      // the firmware development field
byte ctrl_rate=0;       // the control rate for your motors defined at calibration (in times per second)
int enc1_max;         // maximum count for encoder 1 at calibration, for the defined control rate
int enc2_max;         // maximum count for encoder 2 at calibration, for the defined control rate
int enc3_max;         // maximum count for encoder 3 at calibration, for the defined control rate

void loop()
{
   omni.read_all_data(&enc1,&enc2,&enc3,&battery,&temperature,&firm_rel,&firm_int,&firm_dev,&ctrl_rate,&enc1_m
ax,&enc2_max,&enc3_max);
}
```

4.3 Funções de movimentação

4.3.1 void mov_omni(byte linear_speed, int rotational_speed, int direction);

Esta rotina envia uma ordem de movimentação omnidirecional para a Omni-3MD. Este tipo de movimento só é possível num sistema com *encoders* e está sujeito a controlo em malha fechada do tipo PID.

Os parâmetros enviados para a Omni-3MD são:

linear_speed : o valor da velocidade linear do movimento, em percentagem da velocidade máxima, que varia entre 0 e 100;

rotational_speed : o valor da velocidade angular do movimento, em percentagem da velocidade máxima, que varia entre -100 e 100. O sinal indica o sentido da rotação e o valor 0 significa que não é aplicada rotação ao movimento;

direction : o valor da direcção do movimento em graus que varia entre 0 e 360;

Exemplo:

```
#include <Omni3MD.h>
Omni3MD omni;      //declaration of object variable to control the Omni3MD
 int lin_speed=50;
 int rot_speed=0;
 int dir=180;

void loop()
{
   omni.mov_omni(lin_speed,rot_speed,dir);   //move motors
   delay(100);               // The time for the PID control rate
}
```

O *delay* imediatamente a seguir à ordem de movimentação não é necessário, no entanto, a última ordem de movimentação recebida pela OMNI-3MD só será executada no ciclo de controlo PID seguinte. Dependendo da taxa de controlo, o tempo para a execução da ordem de movimentação é de 25ms, 50ms ou 100ms.

Sendo esta função do tipo void nenhum valor é devolvido pela rotina no final da sua execução.

4.3.2 void mov_dif_si(double linear_speed, double rotational_speed);

Esta rotina envia uma ordem de movimentação diferencial em unidades do sistema internacional (SI) para a Omni-3MD. Este tipo de movimento só é possível num sistema com *encoders* e está sujeito a controlo em malha fechada do tipo PID.

Os parâmetros enviados para a Omni-3MD são:

linear_speed : valor da velocidade linear em metros por segundo (m/s);

rotational_speed : valor da velocidade angular em radianos por segundo (rad/s);

Para um determinado motor, o valor de velocidade 0 corresponde à paragem do motor. Valores de velocidade negativos fazem o motor rodar no sentido negativo (definido na calibração) e valores de velocidade positivos fazem o motor rodar no sentido positivo.

Exemplo:

```
#include <Omni3MD.h>
Omni3MD omni;      //declaration of object variable to control the Omni3MD
double lin_speed_si=0.5;    //Linear speed in m/s
double rot_speed_si=0.0;    //Rotational speed in rad/s
void loop()
{
   omni.mov_dif_si(lin_speed_si,rot_speed_si);   //Move forward
   delay(100);
}
```

O *delay* imediatamente a seguir à ordem de movimentação não é necessário, no entanto, a última ordem de movimentação recebida pela OMNI-3MD só será executada no ciclo de controlo PID seguinte. Dependendo da taxa de controlo, o tempo para a execução da ordem de movimentação é de 25ms, 50ms ou 100ms.

Sendo esta função do tipo void nenhum valor é devolvido pela rotina no final da sua execução.

4.3.3 void mov_lin3m_pid(int speed1, int speed2, int speed3);

Esta rotina envia para a Omni-3MD uma ordem para a movimentação independente de três motores. Este tipo de movimento só é possível num sistema com *encoders* e está sujeito a controlo em malha fechada do tipo PID.

Os parâmetros da rotina enviados para a Omni-3MD são:

speed1 : valor da velocidade do motor 1, em percentagem da velocidade máxima, que varia entre -100 e 100;

speed2 : valor da velocidade do motor 2, em percentagem da velocidade máxima, que varia entre -100 e 100;

speed3 : valor da velocidade do motor 3, em percentagem da velocidade máxima, que varia entre -100 e 100;

Para um determinado motor, o valor de velocidade 0 corresponde à paragem do motor. Valores de velocidade negativos fazem o motor rodar no sentido negativo (definido na calibração) e a velocidade máxima é -100. Valores de velocidade positivos fazem o motor rodar no sentido positivo (definido na calibração) e a velocidade máxima é 100.

Exemplo:

```
#include <Omni3MD.h>
Omni3MD omni;      //declaration of object variable to control the Omni3MD
int speed1=20;
int speed2=40;
int speed3=60;
void loop()
{
   omni.mov_lin3m_pid(speed1,speed2,speed3);   // move motors with PID control
   delay(100);
}
```

O *delay* imediatamente a seguir à ordem de movimentação não é necessário, no entanto, a última ordem de movimentação recebida pela OMNI-3MD só será executada no ciclo de controlo PID seguinte. Dependendo da taxa de controlo, o tempo para a execução da ordem de movimentação é de 25ms, 50ms ou 100ms.

Sendo esta função do tipo void nenhum valor é devolvido pela rotina no final da sua execução.

4.3.4 void mov_lin1m_pid(byte motor, int speed);

Esta rotina envia para a Omni-3MD uma ordem para a movimentação independente de um motor. Este tipo de movimento só é possível num sistema com *encoders* e está sujeito a controlo em malha fechada do tipo PID.

Os parâmetros da rotina enviados para a Omni-3MD são:

motor : O motor que se deseja movimentar: valores entre 1 e 3;

speed : valor da velocidade do motor, em percentagem da velocidade máxima, que varia entre -100 e 100;

Para um determinado motor, o valor de velocidade 0 corresponde à paragem do motor. Valores de velocidade negativos fazem o motor rodar no sentido negativo (definido na calibração) e a velocidade máxima é -100. Valores de velocidade positivos fazem o motor rodar no sentido positivo (definido na calibração) e a velocidade máxima é 100.

Exemplo:

```
#include <Omni3MD.h>
#define M2  2          //Motor2
Omni3MD omni;      //declaration of object variable to control the Omni3MD
int speed=20;

void loop()
{
   omni.mov_lin1m_pid(M2,speed);    // move motor 2 at the desired speed
   delay(100);
}
```

O *delay* imediatamente a seguir à ordem de movimentação não é necessário, no entanto, a última ordem de movimentação recebida pela OMNI-3MD só será executada no ciclo de controlo PID seguinte. Dependendo da taxa de controlo, o tempo para a execução da ordem de movimentação é de 25ms, 50ms ou 100ms.

Sendo esta função do tipo void nenhum valor é devolvido pela rotina no final da sua execução.

4.3.5 void mov_lin3m_nopid(int speed1, int speed2, int speed3);

Esta rotina envia para a Omni-3MD uma ordem para a movimentação independente de três motores. Este tipo de movimento não está sujeito a controlo em malha fechada do tipo PID.

Os parâmetros da rotina enviados para a Omni-3MD são:

speed1 : valor da velocidade do motor 1, em percentagem da velocidade máxima, que varia entre -100 e 100;

speed2 : valor da velocidade do motor 2, em percentagem da velocidade máxima, que varia entre -100 e 100;

speed3 : valor da velocidade do motor 3, em percentagem da velocidade máxima, que varia entre -100 e 100;

Para um determinado motor, o valor de velocidade 0 corresponde à paragem do motor. Valores de velocidade negativos fazem o motor rodar no sentido negativo (definido na calibração) e a velocidade máxima é -100. Valores de velocidade positivos fazem o motor rodar no sentido positivo (definido na calibração) e a velocidade máxima é 100.

Exemplo:

```
#include <Omni3MD.h>
Omni3MD omni;      //declaration of object variable to control the Omni3MD
int speed1=20;
int speed2=40;
int speed3=60;
void loop()
{
   omni.mov_lin3m_nopid(speed1,speed2,speed3);   // move motors with no PID control
   delay(100);
}
```

O *delay* imediatamente a seguir à ordem de movimentação não é necessário, no entanto, as ordens de movimentação recebidas pela OMNI-3MD são executadas a cada 100ms, se não estiver definida a taxa de controlo PID.

Sendo esta função do tipo void nenhum valor é devolvido pela rotina no final da sua execução.

4.3.6 void mov_lin1m_nopid(byte motor, int speed);

Esta rotina envia para a Omni-3MD uma ordem para a movimentação independente de um motor. Este tipo de movimento não está sujeito a controlo em malha fechada do tipo PID.

Os parâmetros da rotina enviados para a Omni-3MD são:

motor : O motor que se deseja movimentar: valores entre 1 e 3;

speed : valor da velocidade do motor, em percentagem da velocidade máxima, que varia entre -100 e 100;

Para um determinado motor, o valor de velocidade 0 corresponde à paragem do motor. Valores de velocidade negativos fazem o motor rodar no sentido negativo (definido na calibração) e a velocidade máxima é -100. Valores de velocidade positivos fazem o motor rodar no sentido positivo (definido na calibração) e a velocidade máxima é 100.

Exemplo:

```
#include <Omni3MD.h>
#define M2  2          //Motor2
Omni3MD omni;      //declaration of object variable to control the Omni3MD
int speed=20;

void loop()
{
   omni.mov_lin1m_nopid(M2,speed);    // move motor 2 at the desired speed
   delay(100);
}
```

O *delay* imediatamente a seguir à ordem de movimentação não é necessário, no entanto, as ordens de movimentação recebidas pela OMNI-3MD são executadas a cada 100ms, se não estiver definida a taxa de controlo PID.

Sendo esta função do tipo void nenhum valor é devolvido pela rotina no final da sua execução.

4.3.7 void set_enc_value(byte encoder, int encValue);

Esta rotina permite colocar a contagem incremental dos *encoders* num valor pré-definido pelo utilizador.

O parâmetro encoder define qual o encoder que estará sujeito à alteração da contagem incremental. Os valores válidos para este parâmetro são 1, 2 e 3 que correspondem aos motores 1, 2 e 3 respectivamente.

O parâmetro encValue define o valor a colocar na contagem incremental do encoder. Sendo esta uma variável do tipo int de 16bits, podem ser colocados valores entre -32768 e 32767.

Exemplo:

```
#include <Omni3MD.h>
#define M1  1          //Motor1
#define M2  2          //Motor2
#define M3  3          //Motor3
Omni3MD omni;              //declaration of object variable to control the Omni3MD
...
void loop()
{
   omni.set_enc_value(M1,0);     // resets to zero the encoder value [byte encoder, word encValue]
   delay(1);                  // waits 1ms for Omni3MD to process information
   omni.set_enc_value(M2,-27000);
   delay(1);                  // waits 1ms for Omni3MD to process information
   omni.set_enc_value(M3,25000);
   delay(1);                  // waits 1ms for Omni3MD to process information
...
}
```

Sendo esta função do tipo void nenhum valor é devolvido pela rotina no final da sua execução.

4.3.8 void mov_pos(byte motor, int speed, int encPosition, boolean stoptorque);

Esta rotina envia para a Omni-3MD uma ordem para a movimentação posicional de um motor. Faz com que um determinado motor se movimente de uma posição inicial (valor do encoder) para uma posição final (valor do encoder) definida pelo utilizador. Este tipo de movimento só é possível num sistema com *encoders* e está sujeito a controlo em malha fechada do tipo PID.

As rotinas void **set_enc_value**(byte encoder, int encValue) e void **set_prescaler**(byte encoder, byte value) estão diretamente relacionadas com este tipo de movimentação pelo que a compreensão do seu funcionamento é essencial.

O parâmetro motor define qual o motor que se deseja movimentar podendo assumir valores entre 1 e 3.

O parâmetro speed define a velocidade do motor, em percentagem da velocidade máxima, e são válidos valores entre -100 e 100.

Para um determinado motor, o valor de velocidade 0 corresponde à paragem do motor. Valores de velocidade negativos fazem o motor rodar no sentido negativo (definido na calibração) e a velocidade máxima é -100. Valores de velocidade positivos fazem o motor rodar no sentido positivo (definido na calibração) e a velocidade máxima é 100.

encPosition define a posição final (contagem incremental) que se deseja atingir. São válidos valores entre -32768 e 32767, pois a contagem incremental é armazenada numa variável do tipo int de 16 bits.

O parâmetro stoptorque define como se efetua a paragem na posição desejada e os valores possíveis são 0 ou 1. A paragem pode ser efetuada com ou sem binário de retenção (torque).

- O valor 0 efetua uma paragem sem binário de retenção.
- O valor 1 efetua uma paragem com binário de retenção.

Exemplo:

```
#include <Omni3MD.h>
#define M1  1          //Motor1
Omni3MD omni;      //declaration of object variable to control the Omni3MD
//Variables for motion control
int speed1=90;
int preset1=-3200;
int pos1=3200;

void setup()
{
   ...
   omni.set_prescaler(M1, 1);        //sets the prescaler to 10
   delay(10);                        // 10ms pause required for Omni3MD EEPROM writing
   ...
}

void loop()
{
   //Encoder preset
   omni.set_enc_value(M1,preset1);     // resets to zero the encoder value [byte encoder, word encValue]

//Send movement instructions
   omni.mov_pos(M1,speed1,pos1,1);  //
   delay(1);                  // waits 1ms for Omni3MD to process information
   while(enc1<pos1)
   {
    omni.read_encoders(&enc1,&enc2,&enc3); // read all encoders at once (in a single I2C request)
    Serial.print("PositionM1:");  Serial.println(enc1); //Print encoder position
    delay(100);
   }
}
```

O *delay* imediatamente a seguir à ordem de movimentação não é necessário, no entanto, a última ordem de movimentação recebida pela OMNI-3MD só será executada no ciclo de controlo PID seguinte. Dependendo da taxa de controlo, o tempo para a execução da ordem de movimentação é de 25ms, 50ms ou 100ms.

Sendo esta função do tipo void nenhum valor é devolvido pela rotina no final da sua execução.

4.3.9 void save_position();

Guarda o valor incremental dos *encoders* em memória EEPROM para que não se perca a informação quando se desliga o sistema. No final de uma movimentação posicional pode-se guardar a posição chamando esta rotina e fica salvaguardada a posição dos motores para o caso do sistema ser desligado voluntariamente ou devido a uma falha de energia.

Nota importante: Não deverá ser usada esta rotina durante a movimentação dos motores pois a escrita na EEPROM desativa temporariamente o controlo PID e a contagem dos *encoders*, comprometendo o posicionamento.

Exemplo:

```
#include <Omni3MD.h>
Omni3MD omni;              //declaration of object variable to control the Omni3MD
...
void loop()
{
  omni.save_position();  //save encoders positional data to EEPROM
  delay(15);           // 15ms pause required for Omni3MD EEPROM writing
}
```

Sendo esta função do tipo void nenhum valor é devolvido pela rotina no final da sua execução.

4.3.10 void stop_motors();

Esta rotina envia para a Omni-3MD uma ordem para a paragem de todos os motores. Os motores param sem binário de retenção, ou seja, rodam livremente.

Exemplo:

```
#include <Omni3MD.h>
Omni3MD omni;              //declaration of object variable to control the Omni3MD
...
void loop()
{
  omni.stop_motors();  //stop all motors
}
```

Sendo esta função do tipo void nenhum valor é devolvido pela rotina no final da sua execução.