Desenvolvimento de Algoritmos de Controlo para Locomoção de um Robot Humanóide

Manual do Utilizador

Milton Ruas da Silva Nº21824

Orientação:

Prof. Dr. Filipe Silva (DETI-IEETA)
Prof. Dr. Vítor Santos (DEM-TEMA)

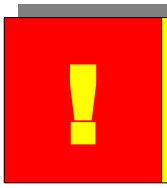
Universidade de Aveiro
Departamento de Electrónica, Telecomunicações e Informática, IEETA
Licenciatura em Engenharia Electrónica e Telecomunicações

Agosto de 2006

Índice

1. Apresentação

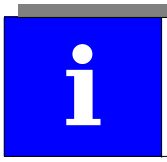
Obrigado pela escolha deste Projecto. Desde já informamos que o sistema desenvolvido corresponde a uma versão inicial e, embora tenha sido submetido a exaustivos testes e a processos de *debugging*, não asseguramos o correcto funcionamento de todos os componentes, pelo que não nos responsabilizamos por eventuais problemas e/ou danos que podem resultar. Agradecemos por isso a sua paciência, e para minimizar estes efeitos, aconselhamos vivamente a leitura deste manual antes da utilização do sistema e a atenção merecida aos tópicos com os seguintes símbolos associados:



Perigo: Nota muito importante a pôr em prática. Ignorar este símbolo pode infligir danos ao sistema ou ao próprio utilizador!

Problema ainda por solucionar. Agradece-se a atenção para a sua resolução.

Recomendação a tomar em conta por parte do utilizador. Ignorá-lo, apenas pode levar a uma degradação da performance do sistema, mas sem levantar qualquer problema a nível de segurança.



Informações úteis a transmitir.

É a sua tarefa, como *quase* Engenheiro, o melhoramento e a resolução, se possível, de todos os problemas evidenciados no relatório 2005/06 na secção "*Resolução de Anomalias*" mais os que entretanto forem surgindo ao longo da fase de desenvolvimento.

De salientar, que este manual pressupõe o conhecimento da arquitectura e das funcionalidades do protótipo humanóide. Para tal, é recomendável a leitura do relatório de projecto 2005/06.

Antes de prosseguir, verifique se possui os seguintes elementos:

- CD de projecto 2005/06 (referenciado como CD_PROJ);
- CD com o software utilizado no projecto (referenciado como CD_SW);
- Software MatLab, versão 7.0.1 (R14);
- *Mini-toolbox* cport, versão 1.3 (para MatLab) – presente no CD do software.

2. Setup Experimental

2.1. Introdução

Neste capítulo apresentamos o sistema desenvolvido e utilizado para a execução de testes. Dois tipos de *setup* foram utilizados:

1. Um só actuador (Fig. 1):
   Este *setup* foi utilizado tendo em mente o teste da resposta de um só servomotor. Prendendo-o num torno de fixação, garante-se um funcionamento seguro independentemente da gama de posições e velocidades testadas, bem como da carga aplicada no braço de ligação ao veio do motor.

Fig. 1: Setup com um só Servomotor.

2. Uma/duas perna(s) (Fig. 2):
   Este setup, utilizado posteriormente, já pretende testar o funcionamento dos servomotores incluídos nas pernas do robot humanóide.

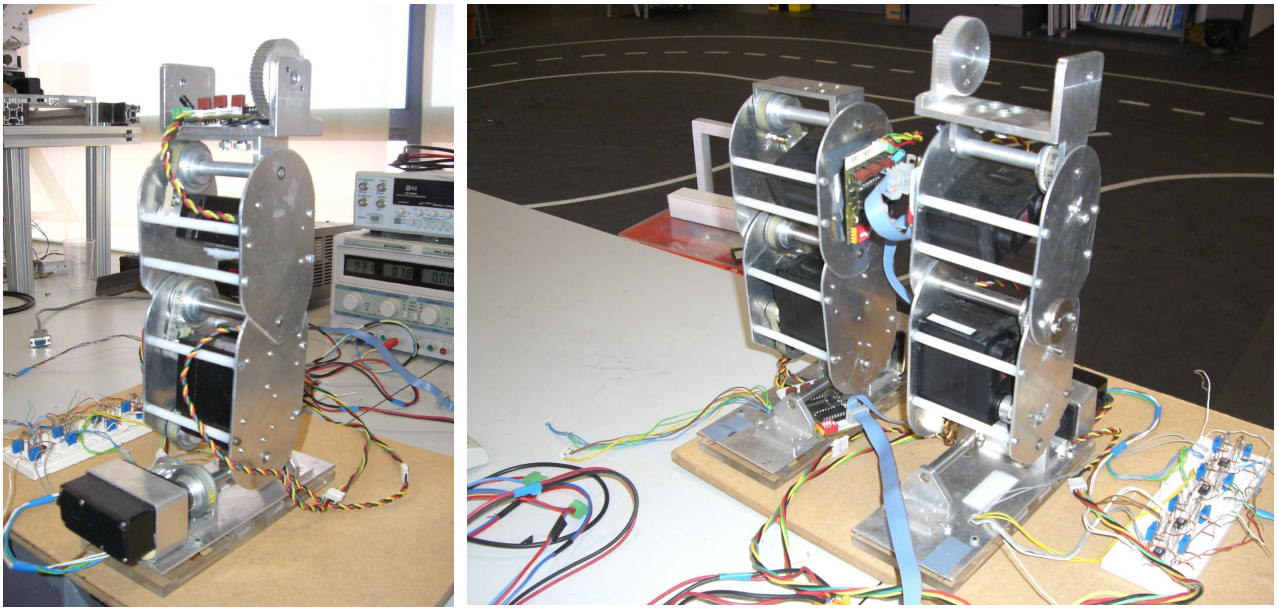


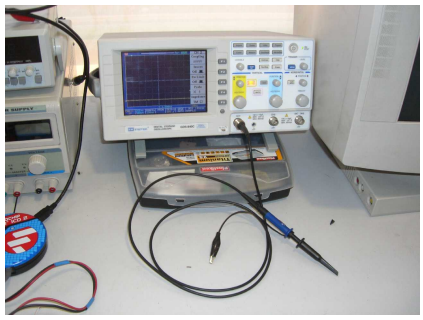
Fig. 2: Setup com a(s) perna(s) do robot humanóide.

O único aspecto que difere nestes *setups* corresponde à localização dos actuadores, pelo que no restante tudo é semelhante.

2.2. Componentes

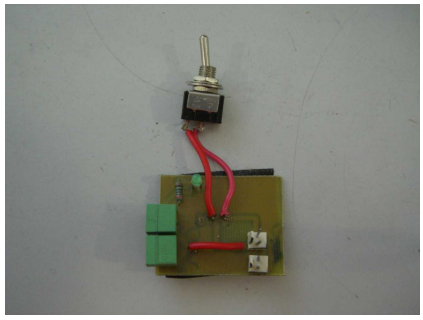

(1) Computador com porta RS-232

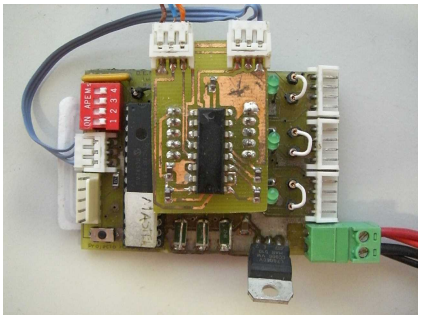
(2) Programador ICD2.

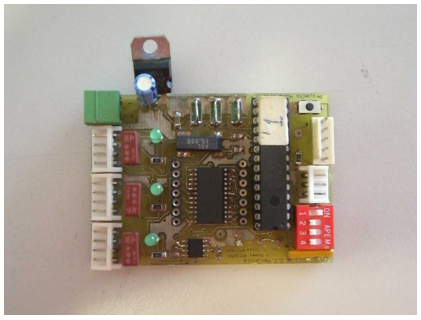

(3) Osciloscópio

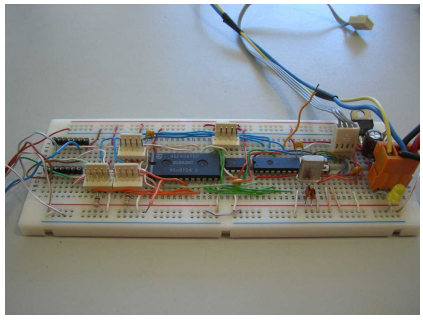

(4) Fonte de Alimentação


(5) Baterias de Lítio + Carregador


(6) Placa de interface às baterias


(7) Unidade Master

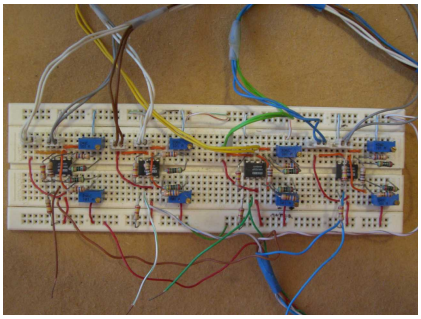

(8) Unidade Slave convencional


(9) Unidade Master principal


(10) Servomotores


(11) Torno de Fixação

(12) Pesos


(13) Interface aos extensómetros


(14) Plataforma do pé

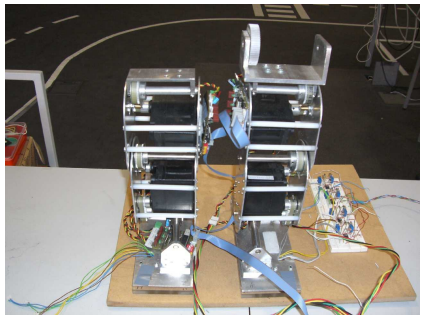

(15) Pernas + Plano de apoio

Fig. 3: Componentes do Setup Experimental.

Os elementos comuns a todas as experiências são (Fig. 3):

- Computador com porta RS-232 incluída ou com adaptador (1);
- Programador ICD2 (2), que em conjunto com o software MPLAB, programa os microprocessadores PIC.
- Osciloscópio para análise dos sinais de saída (3);
- Fonte de alimentação de 8V e de elevada amperagem (4), ou baterias de Lítio (5) com a respectiva placa de interface para o restante hardware (6);
- Unidade Master (7) e Slave (pelo menos uma) podendo-se utilizar a placa em PCB (8) ou a experimental (9).

R	Foi verificado que as várias unidades Slave em PCB apresentavam defeitos nas ligações entre os diversos componentes. Muito embora, muito se fez para os resolver, é possível que algumas deles ainda apresentem problemas no seu funcionamento. Por isso recomenda-se primeiramente o uso do Slave em placa experimental, dada a sua elevada qualidade de funcionamento. Devido a este motivo atribuímos a designação de Slave principal a esta placa.

Para a locomoção de servomotores é necessário adicionalmente:

- Um servomotor (10);
- O torno de fixação, caso pretenda fixar os actuadores (11);
- Pesos de massas diferentes para o teste com diferentes binários de carga (12).

Se em vez de se utilizar um servo, utilizar a estrutura humanóide, são precisos, pelo menos, os membros inferiores – as pernas (15).
No caso da medição das forças de reacção, adicionalmente aos componentes iniciais, é necessário:

- Plataforma(s) do(s) pé(s) com os extensómetros colocados (14) – no caso do uso das pernas, estas já estão incorporadas;
- Circuito de acondicionamento de sinal dos extensómetros (13).

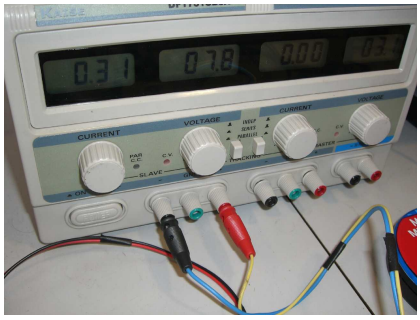
R	Na estrutura de cada pé, verifique se os extensómetros estão presentes e se estão correctamente colados às placas de acrílico.

A Fig. 4 apresenta os meios de conexão entre os vários elementos apresentados na Fig. 3. Para alimentação utilizou-se um cabo comum a todas as unidades Slave (18), que por sua vez um dos slaves (neste caso o slave principal) se liga à fonte de alimentação através do cabo da imagem 17. Caso se usem baterias, este cabo é dispensado.

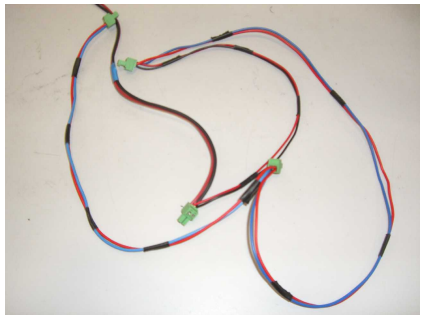
!	É altamente recomendável, no uso de baterias, a remoção do cabo de alimentação da fonte de alimentação ao slave principal (17), para evitar o curto-circuito acidental entre a fonte e as baterias!

Para as comunicações entre nós, são usados os cabos RS-232 entre o PC e o Master (19), e o barramento CAN (20) entre todos os nós.

(16) Cabo USB (PC) + Cabo de programação (Slave principal)



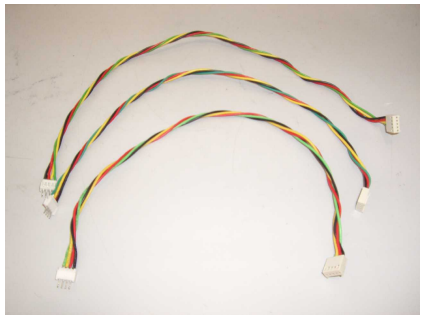
(17) Cabo de Alimentação do Slave principal.



(18) Cabo de alimentação comum a todos os Slaves

(19) Cabo RS-232 para ligação série entre o PC e o Master

(20) Barramento CAN de ligação a todas as unidades controladoras



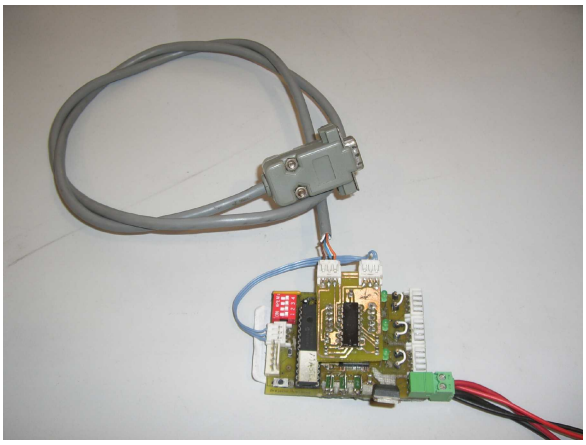
(21) Cabos de ligação dos servos ao Slave principal.

Fig. 4: Conexão entre os vários componentes do setup.

2.3. Montagem do Sistema

A montagem do sistema base, desde o PC até cada servo, segue os seguintes passos:

1. Ligação do PC (1) à unidade Master (7) através do cabo RS-232 (19) (Fig. 5);



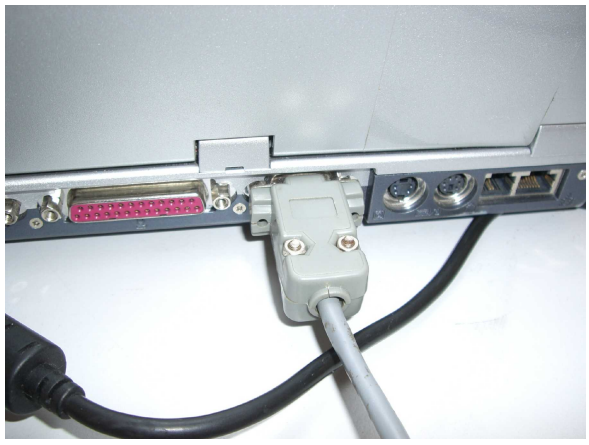


Fig. 5: Ligação RS-232 entre o PC e o Master.

2. Ligação do Master (7) a cada uma das unidades Slave (8 ou 9) através do barramento CAN (20) (Fig. 6);

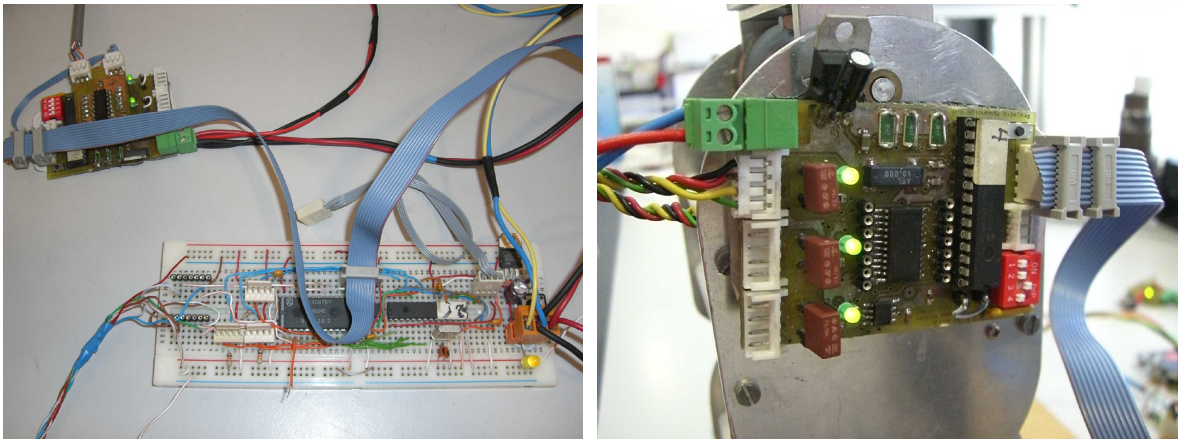


Fig. 6: Ligação CAN entre todas as unidades (cabo *flat* azul).

3. Ligação de cada um dos Slaves (8 ou 9) aos seus (três) servomotores (10). No caso do uso do slave principal (9) dever-se-á utilizar os cabos 21 (Fig. 7).

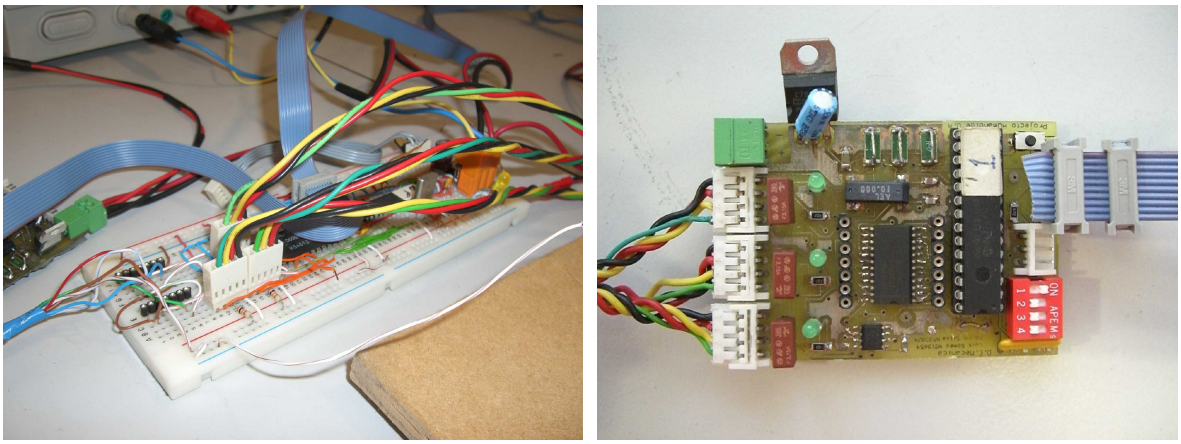


Fig. 7: Ligação dos servomotores (cabos coloridos) a cada slave.

As figuras seguintes apresentam um diagrama demonstrativo das ligações a efectuar, para o caso de um só actuador (Fig. 8), três actuadores (Fig. 9), e para quatro unidades Slave (Fig. 10). Para várias unidades verifique que cada uma possui um endereço único, através do *swicth* de quatro terminais (componente a vermelho) e diferente de 2 (*0b*0010) – este pertence ao slave principal.

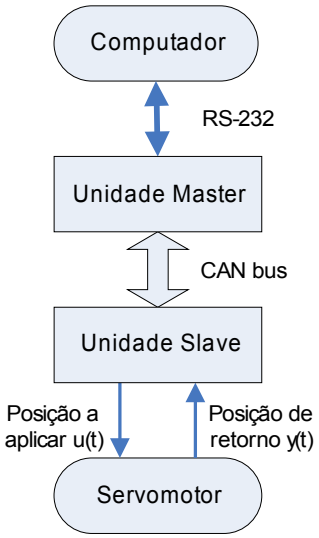


Fig. 8: Diagrama de montagem de um só servo.

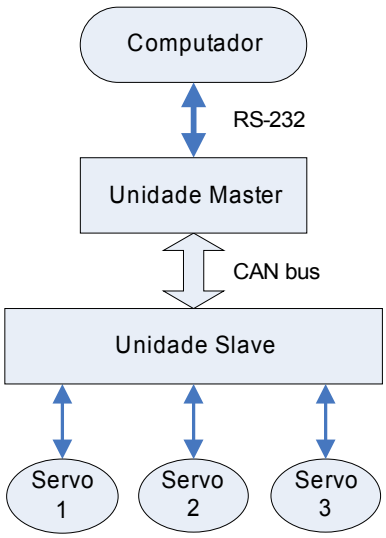


Fig. 9: Diagrama da montagem de três servos.

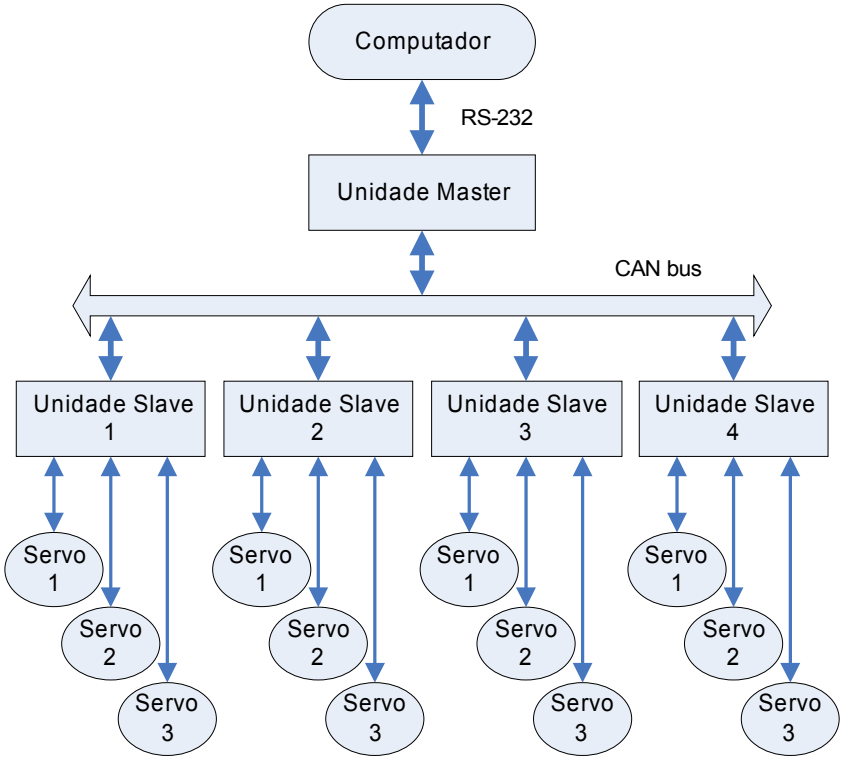


Fig. 10: Diagrama de montagem de várias unidades Slave.

A Fig. 11 visualiza o conjunto completo montado e pronto para funcionamento. Pode-se observar os servomotores da perna ligados à unidade slave principal (fios coloridos), e este à placa Master pelo barramento CAN. Por sua vez, o Master liga ao PC através do cabo RS-232.

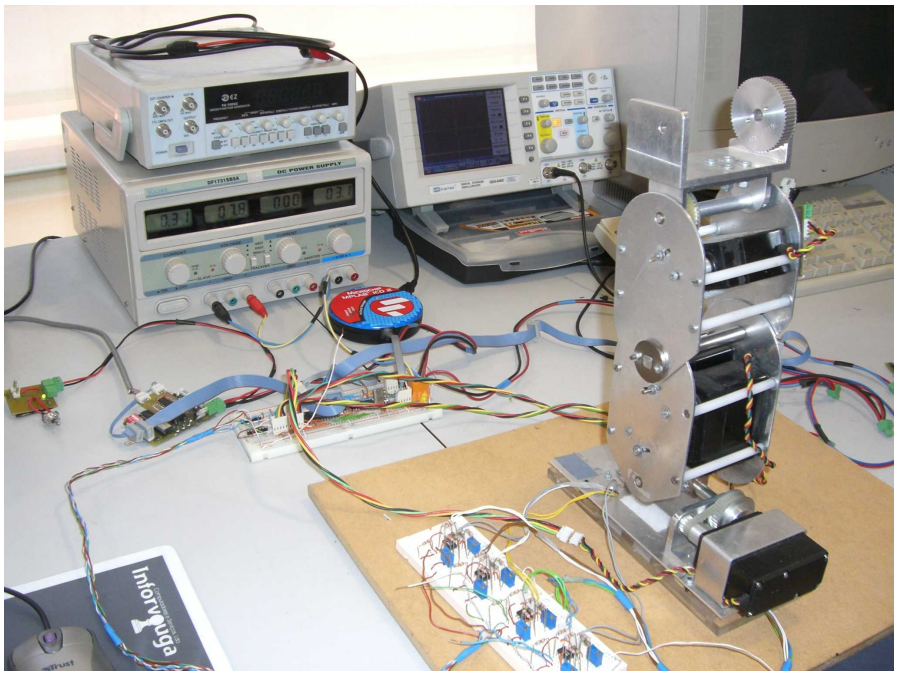


Fig. 11: Imagem final com o sistema completo.

Finalmente, apenas resta a alimentação. No caso de uso das baterias (5) dever-se-á utilizar a placa de interface (6) que efectua a ligação entre as baterias e o cabo de alimentação comum a todos os slaves (18). Verifique com o multímetro se têm carga suficiente (devem ter no mínimo 7.5V).
Se a opção cair na fonte de alimentação (4), regule a tensão de saída entre **7.0V e 8.0V**, e utilize o cabo 17 para ligação à placa slave principal (9).

A fonte de alimentação apenas pode fornecer um máximo de 5A!
Caso a quantidade de servomotores impliquem um consumo maior, dever-se-á utilizar baterias como alternativa – de preferência com várias em paralelo.

Atenção: Nunca trocar as polaridades da alimentação!
Não existe nenhuma protecção contra inversão, pelo que verifique sempre com atenção redobrada as polaridades sempre que faz a ligação dos cabos de alimentação.

2.4. Verificações Finais

Antes da activação dos sistema é fundamental fazer algumas verificações finais para evitar qualquer fenómeno inesperado. Tais verificações desenrolam-se ao nível dos sensores, pois todos os controladores baseiam-se nestes sinais; e como estes actuam directamente sobre os servomotores, é recomendável a verificação da validade dos sinais de saída dos sensores.

a) Sensores dos Servomotores

Os servomotores fornecem para *feedback* os sinais de posição dos seus veios, de modo a que o controlador de posição externo corrija eventuais desvios da posição solicitada.

1. Verifique a correcta ligação do sinal de posição a cada ficha (fio verde) e da ficha à unidade *slave* correspondente;
2. Com o multímetro (modo díodo) confirme conexão de cada sinal à entrada correspondente do multiplexer (do slave).

Servomotor	Entrada do MUX	Código de selecção	Pino do C.I.
1	Y_2	*0b*0010	7
2	Y_1	*0b*0001	8
3	Y_0	*0b*0000	9

Tabela 1: Ligação dos sinais de posição no multiplexer HEF4067BP.

Caso esteja a utilizar as pernas, verifique, também, se em toda a excursão de movimento dos actuadores, a estrutura mecânica bloqueia de alguma forma o seu movimento. Para evitar tal, defina a posição neutra dos servomotores, de modo a corresponder à posição central da excursão de cada junta.

b) Alimentação

Com o sistema desconectado à alimentação confirma as tensões de alimentação:
- No uso da fonte de alimentação: entre 7.0 e 8.0V;
- Com baterias (de 7.4V): mínimo de 7.5V.

Desligue a fonte (se tal é o caso), e conecte a alimentação ao sistema verificando a polaridade:
- Positivo (+): Vermelho;
- Ground (GND): Preto.

... e confirma as ligações pelo cabo de alimentação comum às unidades.

3. Programação do PIC

Para programação dos microprocessadores PIC, dois procedimentos podem ser efectuados:

1. Programação directa pelo *kit* ICD2;
2. Programação através de um bootloader presente na memória do PIC.

O primeiro método é o mais simples e directo, uma vez que utiliza o mesmo software para a edição dos programas, compilação e *downloading* para o dispositivo. Além disso, a ligação deste dispositivo directamente a alguns terminais do PIC faz deste método o mais simplificado.

Quanto ao segundo método, apresenta a desvantagem de exigir software diferenciado para cada tarefa, no que toca à compilação e ao *download* para o PIC. Além disso, exige, também, uma interface série RS-232 para transferência do programa, mas permite dispensar o uso do dispendioso kit ICD2.

3.1. Instalação e Configuração do Software

Instale o seguinte software, na ordem prescrita:

1. MPLab 7.30: este é o ambiente de programação que lhe permite editar o programa, compilá-lo e descarregá-lo no PIC. Durante a instalação, não esquecer de incluir o componente ICD2 e de instalar os *drivers* associados à conexão por USB ao PC (siga as instruções indicadas) – caso não o faça terá de utilizar comunicação RS-232 conjuntamente com alimentação independente.
2. MPLab C18 C compiler 3.00: *Toolsuite* para a compilação/linkagem do programa em linguagem C.

A não instalação dos drivers USB implicam a transferência de informação para o kit ICD2 por RS-232, precisando este elemento de alimentação independente.

Caso utilize a versão 3.00 (ou superior) do compilador, o ambiente MPLab deve possuir a versão mínima 7.30. Em caso contrário a compilação não funcionará.

Recomenda-se altamente a utilização da versão 3.00 (ou superior) do compilador, dado que só a partir desta versão a funcionalidade de salvaguarda do contexto na chamada de funções dentro das rotinas de serviço à interrupção é oferecida. Para versões inferiores, poderá ocorrer corrupção de dados nestas situações!

Verifique se o MPLab está devidamente configurado par o tipo de dispositivos a usar:

1. *Configure → Select Device*
   a) "Device: PIC18F258" (PIC a programar)
   b) "Device family: All"
   c) "Microchip programmer tool support": MPLAB ICD2 deve estar activado
   d) "Microchip debugger tool support": MPLAB ICD2 deve estar activado

2. *Configure → Configuration Bits*
   a) "Oscillator: HS-PLL enabled"
   b) "Osc. switch enable: disabled"
   c) "Power up timer: enabled"
   d) "Brown out detect: enabled"
   e) "Brown out voltage: 2.7V"
   f) "Watchdog timer: disabled-controlled by SWDTEN bit"
   g) "Watchdog postscaler: 1:128"
   h) "Todas as restantes opções desactivadas (disabled)"
3. *Configure → ID Memory*
   a) "User ID: FFFFFFFF"
   b) "Use unprotected checksum: disabled"
4. *Configure → Settings*
   a) Seleccione a *tab* Projects:
      - "Save project before build: enabled"
      - "Save files before build: yes"
      - "Halt build on first failure: enabled"
   b) *Tab* Program Loading:
      - "Clear program memory upon loading a program: enabled"

Caso já possua um projecto preparado (*workspace*), abra o *workspace* e verifique as configurações da *toolsuite* de compilação (em caso negativo salte estes passos):

1. Abra o *workspace* em *File → Open Workspace*
2. Clique *Project → Select language toolSuite*:
   a) "Active toolsuite: Microchip C18 toolsuite"
   b) "Toolsuite contents: MPASM assembler (mpasmwin.exe)":
      - Location: "C:\<MPLAB_DIR>\Microchip\MPASM Suite\mpasmwin.exe"
   c) "Toolsuite contents: MPLINK object linker (mplink.exe)":
      - Location: "C:\<C18_DIR>\MCC18\bin\mplink.exe"
   d) "Toolsuite contents: MPLAB C18 C compiler (mcc18.exe)":
      - "Location: "C:\<C18_DIR>\MCC18\bin\mcc18.exe"
   e) "Toolsuite contents: MPLIB librarian (mplib.exe)":
      - Location: "C:\<C18_DIR>\MCC18\bin\mplib.exe"

As referências <C18_DIR> e <MPLAB_DIR> dizem respeito aos directórios onde o compilador C18 e o MPLAB, respectivamente, foram instalados. Estas informações foram fornecidas durante a instalação destes dois programas.

Caso não possua nenhum projecto:
1. Crie um novo projecto clicando em *Project → Project Wizard*
2. Janela de boas vindas: clique Next/Seguinte
3. Step 1 – Selecção do dispositivo:
   a) "Device: PIC18F258"
   b) Next/Seguinte
4. Step 2 – Selecção da *toolsuite*:
   a) "Active toolsuite: Microchip C18 toolsuite"
   b) "Toolsuite contents: MPASM assembler (mpasmwin.exe)":
      - Location: "C:\<MPLAB_DIR>\Microchip\MPASM Suite\mpasmwin.exe"
   c) "Toolsuite contents: MPLINK object linker (mplink.exe)":
      - Location: "C:\<C18_DIR>\MCC18\bin\mplink.exe"

   d) "Toolsuite contents: MPLAB C18 C compiler (mcc18.exe)":
      - Location: "C:\<C18_DIR>\MCC18\bin\mcc18.exe"
   e) "Toolsuite contents: MPLIB librarian (mplib.exe)":
      - Location: "C:\<C18_DIR>\MCC18\bin\mplib.exe"
   f) Next/Seguinte
3. Step 3 – Nome e localização do projecto:
   a) Indique um nome e a localização do seu novo projecto
   b) Next/Seguinte
4. Step 4 – Adição dos ficheiros de que faz parte o projecto:
   a) Seleccione os ficheiros e clique em "Add"
   b) Next/Seguinte
5. Sumário:
   a) Confirme os dados que introduziu
   b) Clique em Finish/Concluir

Uma janela rectangular surgirá no canto superior esquerdo indicando todos os ficheiros de que faz parte o projecto. Verifique que nos "*Linker Scripts*" está adicionado o ficheiro "18f258.lkr" para o PIC18F258. Caso não esteja, clique com o botão direito neste campo e adicione o respectivo ficheiro em "C:\<C18_DIR>\MCC18\lkr\18f258.lkr". Verifique, também, que nos seus programas faz a inclusão da header file "p18f258.h" para manipulação das macros do PIC:

```
#include <p18f258.h>
```

Finalmente, só falta definir as opção de compilação (as opções a ***bold*** correspondem às que implicam modificação – as restantes devem coincidir com as indicadas):

1. Clique em *Project* → *Build Options* → *Project*
2. ***Tab* General – Localização dos ficheiros:**
   a) Include path, $(INCDIR): "C:\<C18_DIR>\MCC18\h"
   b) Library path, $(LIBDIR): "C:\<C18_DIR>\MCC18\lib"
   c) Linker-script path, $(LKRDIR): "C:\<C18_DIR>\MCC18\lkr"
3. *Tab* MPASM/C17/C18 suite – opções gerais:
   a) "Categories: All options"
   b) "Generate command line":
      - "Build normal target (invoke MPLINK)"
4. *Tab* MPASM assembler – opções do assembler:
   a) "Categories: General"
   b) "Generate command line":
      - "Default Radix: Hexadecimal"
5. *Tab* MPLINK linker – opções do linker:
   a) "Categories: All options"
   b) "Generate command line":
      - "HEX-file format: INHX32"
      - "Generate map file: enabled"
      - **"Supress COD-file generation: enabled"**
6. *Tab* MPLAB C18 – opções do compilador:
   a) "Categories: General"
   b) "Generate command line":
      - "Diagnostics level: errors and warnings"
      - "Default storage class: Auto"

A supressão da geração do ficheiro COD permite evitar o erro de comprimento excessivo do caminho (*path*) do projecto. Além disso, a sua supressão é segura, uma vez que apenas era utilizado em versões antigas do compilador.

Agora estamos em condições de executar compilações. Para tal abra um *workspace* já existente (File → Open Workspace), ou se necessário crie um, e clique em *Project→Build All* (CTRL+F10). A Fig. 12 apresenta a informação de saída resultante de uma compilação.

```
Clean: Done.
Executing: "C:\Programas\UA\MCC18\bin\mcc18.exe" -p=18F258 /i"C:\Programas\UA\MCC18\h" "slave.c" -fo="slave.o" -Ou- -Ot- -Ob- -Op- -Or- -Od- -Opa-
MPLAB C18 v3.00
Copyright 1999-2005 Microchip Technology Inc.
Cracked by LZO [Trace]
Executing: "C:\Programas\UA\MCC18\bin\mcc18.exe" -p=18F258 /i"C:\Programas\UA\MCC18\h" "can.c" -fo="can.o" -Ou- -Ot- -Ob- -Op- -Or- -Od- -Opa-
MPLAB C18 v3.00
Copyright 1999-2005 Microchip Technology Inc.
Cracked by LZO [Trace]
C:\Milton\UA\Humanoid\Lab\Fase3_Integration\Last Version\Slave_Stable_2.00\can.c:150:Warning [2060] shift expression has no effect
C:\Milton\UA\Humanoid\Lab\Fase3_Integration\Last Version\Slave_Stable_2.00\can.c:222:Warning [2060] shift expression has no effect
C:\Milton\UA\Humanoid\Lab\Fase3_Integration\Last Version\Slave_Stable_2.00\can.c:360:Warning [2060] shift expression has no effect
Executing: "C:\Programas\UA\MCC18\bin\mcc18.exe" -p=18F258 /i"C:\Programas\UA\MCC18\h" "CanDrivers.c" -fo="CanDrivers.o" -Ou- -Ot- -Ob- -Op- -Or- -Od- -Opa-
MPLAB C18 v3.00
Copyright 1999-2005 Microchip Technology Inc.
Cracked by LZO [Trace]
Executing: "C:\Programas\UA\MCC18\bin\mcc18.exe" -p=18F258 /i"C:\Programas\UA\MCC18\h" "global.c" -fo="global.o" -Ou- -Ot- -Ob- -Op- -Or- -Od- -Opa-
MPLAB C18 v3.00
Copyright 1999-2005 Microchip Technology Inc.
Cracked by LZO [Trace]
Executing: "C:\Programas\UA\MCC18\bin\mcc18.exe" -p=18F258 /i"C:\Programas\UA\MCC18\h" "pic2.c" -fo="pic2.o" -Ou- -Ot- -Ob- -Op- -Or- -Od- -Opa-
MPLAB C18 v3.00
Copyright 1999-2005 Microchip Technology Inc.
Cracked by LZO [Trace]
Executing: "C:\Programas\UA\MCC18\bin\mcc18.exe" -p=18F258 /i"C:\Programas\UA\MCC18\h" "servo.c" -fo="servo.o" -Ou- -Ot- -Ob- -Op- -Or- -Od- -Opa-
MPLAB C18 v3.00
Copyright 1999-2005 Microchip Technology Inc.
Cracked by LZO [Trace]
Executing: "C:\Programas\UA\MCC18\bin\mcc18.exe" -p=18F258 /i"C:\Programas\UA\MCC18\h" "trajectory.c" -fo="trajectory.o" -Ou- -Ot- -Ob- -Op- -Or- -Od- -Opa-
MPLAB C18 v3.00
Copyright 1999-2005 Microchip Technology Inc.
Cracked by LZO [Trace]
Executing: "C:\Programas\UA\MCC18\bin\mcc18.exe" -p=18F258 /i"C:\Programas\UA\MCC18\h" "myMaths.c" -fo="myMaths.o" -Ou- -Ot- -Ob- -Op- -Or- -Od- -Opa-
MPLAB C18 v3.00
Copyright 1999-2005 Microchip Technology Inc.
Cracked by LZO [Trace]
Executing: "C:\Programas\UA\MCC18\bin\mcc18.exe" -p=18F258 /i"C:\Programas\UA\MCC18\h" "data.c" -fo="data.o" -Ou- -Ot- -Ob- -Op- -Or- -Od- -Opa-
MPLAB C18 v3.00
Copyright 1999-2005 Microchip Technology Inc.
Cracked by LZO [Trace]
Executing: "C:\Programas\UA\MCC18\bin\mplink.exe" /l"C:\Programas\UA\MCC18\lib" /k"C:\Programas\UA\MCC18\lkr" "C:\Programas\UA\MCC18\lkr\18f258.lkr" "C:\Milton\UA\Hur
MPLINK 4.00, Linker
Copyright (c) 2005 Microchip Technology Inc.
Errors    : 0

MP2HEX 4.00, COFF to HEX File Converter
Copyright (c) 2005 Microchip Technology Inc.
Errors    : 0

Loaded C:\Milton\UA\Humanoid\Lab\Fase3_Integration\Last Version\Slave_Stable_2.00\slave.cof.
BUILD SUCCEEDED: Fri Sep 01 17:32:44 2006
```

Fig. 12: Dados imprimidos pelo MPLAB após uma compilação bem sucedida.

Note a ausência de erros no processo de linkagem e no de geração do ficheiro HEX numa compilação bem sucedida. A presença de *warnings* nem sempre afecta o funcionamento do programa, como é o caso de *shiftings* de zero bits indicadas na compilação da Fig. 12 (pode ocorrer dado que os deslocamentos a efectuar são indicados pelo programador através de constantes *#define*). No entanto, tal depende de caso para caso, pelo que se recomenda a análise cuidada dos *warnings*.

O próximo passo é agora a transferência do ficheiro HEX gerado pela compilação para o respectivo PIC. Embora o programa seja praticamente igual entre todas as unidades slave, há algumas diferenças em termos de hardware entre as diversas placas o que implica pequenas modificações no software. Tais diferenças verificam-se ao nível da ligação dos sinais de PWM do PIC para os motores (Tabela 2). As placas slave secundárias utilizam pinos do PIC que coincidem com os utilizados pelo kit ICD2, pelo que a placa slave principal, que contempla a ligação a este kit, evita o uso destes pinos para evitar a entrada de sinais estranhos nos motores.

Tipo de Slave	Pinos de saída do PWM

Slave principal	`// Pinos de interface do PIC`
	`#define SERVO1_PIN PORTBbits.RB1 // Servomotor 1`
	`#define SERVO2_PIN PORTBbits.RB4 // Servomotor 2`
	`#define SERVO3_PIN PORTBbits.RB5 // Servomotor 3`

Slave secundário	`// Pinos de interface do PIC`
	`#define SERVO1_PIN PORTBbits.RB5 // Servomotor 1`
	`#define SERVO2_PIN PORTBbits.RB6 // Servomotor 2`
	`#define SERVO3_PIN PORTBbits.RB7 // Servomotor 3`

Tabela 2: Pinos de saída de PWM do PIC para diferentes tipos de Slave.

Por isso, verifique sempre se a associação dos pinos do PIC à saída de PWM está de acordo com o slave a programar. Os dados da Tabela 2 encontram-se na *header file* **pic2.h**!

!	Na unidade slave principal, NUNCA redefinir as ligações de saída de PWM para os pinos RB7, RB6 e RB5, de modo a coincidir com os dos slaves secundários! Estes pinos coincidem com os de programação do kit ICD2, pelo que estes sinais seriam introduzidos nas entradas dos servomotores, produzindo efeitos imprevisíveis!

3.2. Utilização do kit ICD2

Fig. 13: Kit ICD2.

Antes da conexão do kit ao PC confirme a sua desactivação no ambiente MPLAB: para tal a opção *Programmer → Select Programmer → None* deve estar seleccionada.

1. Conecte o kit ao PIC utilizando para isso a placa slave principal, através do cabo de ficha *molex* de 5 pinos.
2. Ligue o cabo USB entre o kit e o PC. Caso utilize a ligação RS-232, conecte o cabo RS-232 fornecido e só depois o transformador para alimentação. O led *power* deverá estar a verde.
3. Ligue a alimentação (fonte ou baterias) da placa slave que está ligada ao kit;
4. No ambiente MPLAB active o kit fazendo *Programmer → Select Programmer → MPLAB ICD2*.

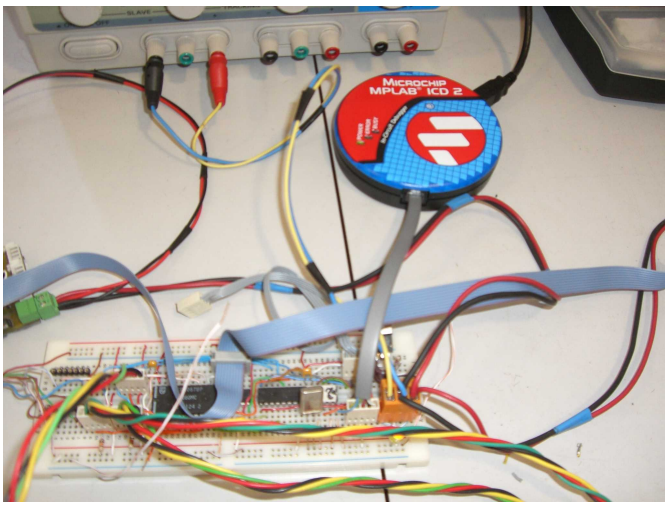


Fig. 14: Ligação do kit ICD2 à placa slave principal.

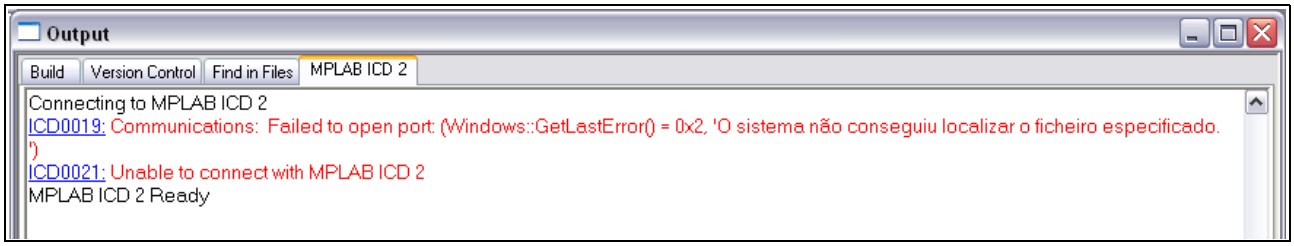
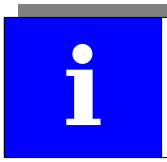


Fig. 15: Ligação mal sucedida ao kit ICD2.

Se uma mensagem de erro do tipo da aparecer, é porque o kit não está devidamente configurado. Os tópicos a ***bold*** são fundamentais para a correcta configuração:

1. Clique em *Programmer* → *Settings*
2. *Tab* Status:
   a) "Automatically connect at startup: enabled"
   b) "Automatically download firmware if needed: enabled"
3. *Tab* Communication:
   a) No caso de comunicação USB, definir "Com Port: USB"
   b) Em caso de comunicação RS-232, definir "Com Port: COM?" e Baudrate: 57600"
4. *Tab* Power:
   a) "Power target circuit from MPLAB ICD2 (5V V_{DD}): disabled" (alimentação do PIC independente)
5. *Tab* Program:
   a) "Allow ICD2 to select memories and ranges: enabled"
   b) "Automatically":
   - "Program after successful build: enabled"
   - "Run after successful program: enabled"



O ponto 5.b), embora em nada se relaciona com o funcionamento do kit, permite a automatização do processo de *downloading*, efectuando a programação e arranque automáticos do PIC logo após uma compilação bem sucedida.

Também poderá configurá-lo através da ferramenta *Programmer* → *MPLAB ICD2 Setup Wizard* introduzindo as informações descritas atrás.
Neste ponto, active o ICD2 fazendo *Programmer* → *Connect* (*Programmer* → *Select Programmer* → *MPLAB ICD2* deve estar seleccionado).
Se mesmo assim, o kit ICD2 continua sem responder proceda do seguinte modo:

1. Desseleccione o programador fazendo *Programmer* → *Select Programmer* → *None*;
2. Remova o cabo USB do PC;
3. Espere uns segundos e volte a conectar o cabo;
4. Volte a tentar a activação fazendo *Programmer* → *Select Programmer* → *MPLAB ICD2*.

Se mesmo assim, nada se resolver, reinicie o PC e repita os passos descritos no início desta secção.

É normal que o kit ICD2 deixe de responder após suspensão ou hibernação do PC, pelo que a desactivação e consecutiva activação, deverá resolver este problema.

É normal que após a mudança do modelo do PIC, o MPLAB precise de descarregar um novo sistema operativo para redefinição do *software* de interface.
Este procedimento também pode ser utilizado em caso de disfunção do kit – *Programmer → Download ICD2 Operating System*. No entanto apenas o deve fazer em último recurso.

Com o kit a funcionar, a programação e o arranque do PIC devem ser executados automaticamente após uma compilação bem sucedida. É esta a grande vantagem do uso do kit ICD2: utilizando somente o ambiente MPLAB é possível editar o programa e compilá-lo, com o *downloading* e *startup* automáticos!

Para a execução isolada destas tarefas:
1. *Downloading* do programa para o PIC: *Programmer → Program*
2. *Startup* do PIC: *Programmer → Release from Reset*

3.3. Utilização de um *BootLoader*

Caso não se disponha de um kit ICD2, apenas nos resta a alternativa de usar um *bootloader* para carregar o programa. Este método baseia-se na instalação de um pequeno programa no ínicio da memória do PIC que é executado sempre que é inicializado. Quando em execução realiza o seguinte procedimento:
1. Verifica durante um determinado período de tempo, se na porta série RS-232, alguém está a tentar enviar um programa;
   a) Em caso afirmativo, o *bootloader* efectua a transferência para a zona de memória que se segue à localização do mesmo. No fim da transferência, o PIC bloqueia numa *dummy task* esperando pelo RESET do dispositivo (volta ao passo 1);
   b) Se nenhum programa estiver presente na entrada RS-232, após o timeout, o bootloader executa o último programa carregado na zona de memória que se segue à sua localização.

Resumindo, se não se estiver a enviar nenhum programa pela linha série, o PIC executa o último programa carregado sempre que for inicializado. Só na tentativa de transferência, é que o *bootloader* o carrega em memória. Note que cada PIC precisa de uma interface RS-232 para efectuar a transferência dos programas.

Para transferir o programa para o PIC é utilizado o software PicLoader (ver nota informativa abaixo). Apenas terá de configurar a porta série utilizada para comunicar (*Com. Port:* COM?) e o *baudrate* (*tab* Comm) à velocidade indicada pelo *bootloader* adoptado.

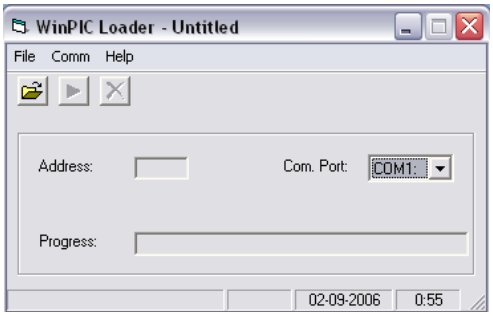
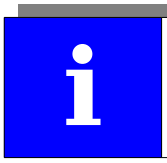


Fig. 16: Software PicLoader para transferir programas para o PIC.

Para transferir o programa gerado pelo compilador MPLAB C18 compiler:

1. **Com um editor de texto, abra o ficheiro HEX gerado pelo compilador e remova a primeira linha;**
2. Agora com o PicLoader, carregue o ficheiro HEX através de *File* → *Open*;
3. Verificar a ligação do cabo RS-232 entre o controlador e o PC;
4. Carregar em *Play*;
5. Ligar o PIC ou pressionar RESET.



A remoção da primeira linha do código HEX apenas é indispensável para os PIC's da série 18F.

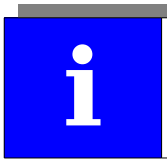
O programa deverá iniciar a transferência. No final, volte a premir RESET, e o mesmo inicia a sua execução!

A parte mais complicada, é a programação do *bootloader* para cada PIC. Tal exige *hardware* dedicado, pois desta vez não temos a ajuda de nenhum *software* para a transferência deste bocado de código para a memória. As opções mais simples, mas também mais dispendiosas, consistem no uso do kit ICD2 ou num programador dedicado fornecido pela *Microchip*. Uma solução mais barata consiste na utilização do denominado "programador universal", cujo circuito pode ser encontrado facilmente na internet, com o respectivo *software* de interface.

Programação do *bootloader* usando o kit ICD2:

1. Abrir o ambiente MPLAB;
2. Conectar-se ao kit ICD2: *Programmer* → *Select Programmer* → *MPLAB ICD2*;
3. Importar o ficheiro HEX com o bootloader a varregar: *File* → *Import* ...
4. Carregar o bootloader para o PIC: *Programmer* → *Program*.

Caso utilize o programador da *Microchip* ou o "programador universal" leia a documentação fornecida para manipular estes dispositivos.
Tome atenção na escolha do *bootloader*, pois cada um é específico para cada modelo de PIC e para cada frequência de cristal e de oscilação. Adicionalmente, tome nota do *baudrate* de transferência dos programas para configuração do PicLoader.



Tanto o PicLoader como os vários *bootloaders* para diferentes PIC's e diferentes frequências de cristal estão disponíveis na página http://www.ieeta.pt/~jla (fornecidos pelo docente José Luís Azevedo).

Caso o software PicLoader não funcione, na situação do uso do "programador universal", recomenda-se a substituição deste pelo *TruTrack Boot Loader* (disponível gratuitamente na internet).

4. Operações Básicas

Após ter passado todas as verificações enunciadas na secção 2.4 (Verificações Finais) e de ter transferido o programa correcto para cada PIC (ver fim da secção 3.1) estamos em condições de colocar o sistema em funcionamento.
Todos os procedimentos descritos consideram a utilização do software, versão 2.00 desenvolvido no ano 2005/06.

4.1. Comunicação a partir do PC

Para comunicar com o sistema de placas controladoras, será utilizado o ambiente MatLab conjuntamente com os device drivers cport para comunicação RS-232. As ordens são enviadas para a unidade Master de acordo com o protocolo enunciado no relatório de projecto (capítulo 2), esta unidade trata de redireccionar os dados para os slaves, caso seja preciso, e responde ao PC com os dados pedidos ou uma mensagem de confirmação.

Instalação do ambiente de comunicações:

1. Instale o software MatLab 7.0.1[1]: Este software será utilizado como o ambiente de comunicação com o sistema;

2. No CD de software, descomprima a mini-toolbox cport 1.3[2], e copie o directório para "C:\<MATLAB_DIR>\MatLab701\toolbox\".

3. Abra o MatLab e em *File→Set Path...* adicione o directório cport para onde copiou-o (*Add Folder*) e faça *Save* (termine com *Close*). Feche o MatLab e volte a abri-lo. Após este procedimento, as rotinas do cport são sempre reconhecidas.

Algumas rotinas em MatLab foram construídas, para execução de todas as operações de actuação e leitura sensorial, fazendo uso dos device-drivers do cport e implementando o protocolo definido. Estas rotinas encontram no CD do projecto e visam as seguintes operações:

Ficheiro	***Descrição***
initcom.m	Criação de uma nova linha de comunicações entre o PC e a unidade Master
killcom.m	Término da linha de comunicações
calibcom.m	Pedido ao Master do retorno de uma sequência de teste: FA F9 F8 F7 F6 F5 (hex)
readcanstat.m	Consulta do estado do barramento CAN (erros de comunicação master/slaves)
readjoint.m	Leitura sensorial das posições dos servomotores de um slave
readspecial.m	Leitura dos sensores especiais (sensores de pressão, giroscópio ou inclinómetro)
applyjoint.m	Actuação nos servomotores de um determinado slave
applycontrol.m	Actualização dos parâmetros de compensação de uma determinado slave

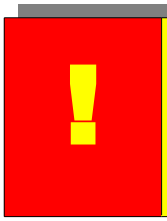
Tabela 3: Lista de device drivers da unidade principal.

1 O software MatLab necessita de uma licença para o obter/utilizar. Fale com o (co)orientador para a obter.
2 <CD_SW>\RS-232\CPORT\cport13.rar

Mais informações sobre estas rotinas podem ser encontradas no manual do programador ou no capítulo 2 do relatório de projecto.
Para sua utilização copie o directório LAB do CD de projecto[3] para o disco e defina-as também como globais repetindo o mesmo procedimento indicado atrás (passo 3) para este directório.

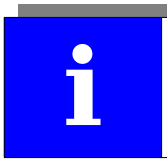
4.2. Activação do Sistema

Antes de ligar o sistema, desligue a ligação do kit ICD2 ao slave principal para evitar quaisquer interferências, e volte a verificar se a alimentação está conectada com as polaridades correctas e as tensões adequadas (secção 2.4.b)
Como medida de segurança, não comece por ligar aos servomotores presentes nas pernas, mas sim primeiramente sem nenhum, ou com servos aparte. Só depois de confirmada a validade dos sinais de controlo, é seguro a ligação aos actuadores da estrutura humanóide.



A estratégia de ligar primeiramente os servomotores aparte, e não os da estrutura humanóide, é uma medida de segurança para prevenir comportamentos imprevistos que podem tornar a própria estrutura perigosa para o ser humano. Como exemplo podemos referir o perigo que as pernas representam em situações anómalas – podem comportar-se como "tesouras" colocando em risco o ser humano que opera!

Agora sim, pode ligar a fonte de alimentação/baterias! Os *leds* verdes das várias unidades de controlo devem ficar luminosos, indicando que um sinal de controlo está pronto a ser aplicado a todos os motores.



Cada led está associado a cada um dos motores. Caso algum não se ligue, é porque o fusível de corte da alimentação do respectivo motor queimou. Substitua-o inserindo outro de pelo menos 3.15A (ligação por encaixe).

Devido à presença de picos de corrente no momento da activação, é normal verificar pequenos deslocamentos bruscos por parte dos servomotores, mas logo após algumas centenas de milisegundos os servos devem encontrar-se em repouso. Verifique com o osciloscópio a ausência de qualquer sinal de PWM e que os dispositivos se deslocam livremente.

Pode acontecer, se estiver a usar várias unidades slave, que algumas unidades não deixem em repouso os respectivos motores, aplicando imediatamente os sinais de PWM. Tal ainda é inexplicável, uma vez que o programa é igual entre todos os slaves, e tal só acontece com o funcionamento de várias unidades.

Se devido aos picos de corrente iniciais, algum fusível se queimou, verifique a amperagem do mesmo – é provável que apenas seja de 2A. Substitua-o por um de 3.15A!

3 <CD_PROJ>\LAB\

4.3. Criação de uma Linha de Comunicações

Verifique se o cabo RS-232 está ligado entre a unidade Master e o PC. Tome nota da porta COM que está a ser utilizada e do baudrate de transferência do Master (115200 bps para a versão 2.00 do software). Por questões de simplicidade consideraremos a utilização da porta COM1.

Parâmetros	***Valores***
Porta	COM1
Baurate	115200 bps

Tabela 4: Definições da comunicação série.

No ambiente MatLab, utilize a rotina *initcom* para a criação de uma linha de comunicações:

```
[handler,state]=initcom(porta,baudrate)
```

... em que o *handler* é um identificador da linha de comunicações – todos os acessos seguintes devem fazer referência a este *handler* – e *state* indica as configurações da linha.

Para criação de uma linha, utilizando a porta COM1 a 115200bps, é suficiente fazer:

```
>> H=initcom(1,115200)
```

A variável H possuirá o identificador handler da nova linha. Se este valor for não nulo, a linha foi criada com sucesso!
Para encerrar a linha de comunicações utilize a rotina *killcom*:

```
stat=killcom(handler)        →        >> killcom(H)
```

... retornando 1 (*stat*) em caso de sucesso.

Se durante a abertura, algum destes erros ocorre:

- Porta COM já aberta;
- Handler nulo;
- Bloqueio durante a execução.

... termine a ligação com um *killcom*, desligue o sistema, e volte-o a ligar tentando de novo a abertura.

Para fazer o *debugging* destes procedimentos, um terminal de comunicações é muito útil. Embora o *Hyperterminal* fornecido pelo Windows possa ser utilizado para trocar mensagens de texto, tal mostra a sua fraqueza quando queremos trocar caracteres não imprimíveis. Um terminal muito popular, o R.E.Simith – I/O Commander[4], está disponível gratuitamente na internet[5], e permite a troca de caracteres de qualquer código ASCII nos dois sentidos.

4 <CD_SW>\RS-232\Rescomdr-20050625-V4p3.zip
5 http://www.rs485.com

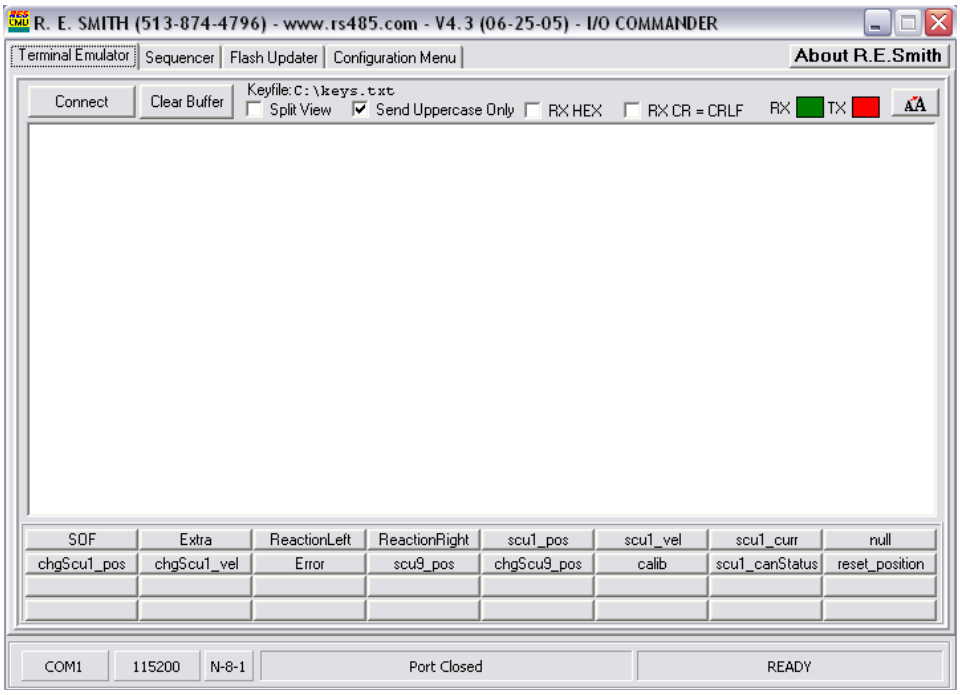


Fig. 17: Terminal R.E.Smith - I/O Commander.

Instale este terminal, e com a linha de comunicações desactivada (*killcom*) proceda do seguinte modo:

1. Abra o terminal R.E.Smith;
2. Seleccione a tab *Configuration Menu*;
   a) Crie a KEY LABEL "*test frame*" e associe a este label o valor ~250 (*0x*FA) correspondente ao pedido de transmissão de uma sequência de teste.
3. Seleccione a tab *Terminal Emulator*;
   a) Especifique as definições da ligação no canto inferior esquerdo: COM1 (porta), 115200 (baudrate), N-8-1 (sem bit de paridade, 8 bits de dados e 1 stop bit).
   b) Desactive a opção *Send Uppercase Only* e active *RX HEX* para visualizar todos os caracteres em código hexadecimal.
   c) Estabeleça as comunicações premindo *Connect*!

Para testar o correcto funcionamento, envie o caracter *0x*FA, premindo na label "*test frame*". O master deverá responder com a sequência de teste FA F9 F8 F7 F6 F5 (aparecerá escrito a verde). Caso tal aconteça as comunicações estão operacionais! Termine a ligação com um *unconnect*, feche o terminal, e volte para o MatLab criando a linha de comunicações com o *initcom*.

Caso ainda não tenha conseguido, verifique com o multímetro as ligações desde a unidade Master até ao PC (o cabo pode ter um defeito).
Se mesmo assim tudo parece estar bem, verifique no software da unidade master, se a constante BAUD do ficheiro usart.c está definida a 115200. Como última tentativa, edite a função *usartInit*() atribuindo o valor 21 ao registo SPBRG. Este valor apenas válido para baudrates de 115200bps com osciladores a 40 MHz – PLL activa com cristal a 10 MHz; para outros parâmetros siga a seguinte fórmula, com a função *round* representativa do arredondamento para o valor inteiro mais próximo:

$$SPBRG = round\left(\frac{f_{osc}}{16 \times baudrate} - 1\right) \qquad f_{osc}\text{: frequência do oscilador}$$

4.4. Activações dos sinais de PWM nos Motores

Para actuação sobre as juntas, a rotina *applyjoint* é utilizada:

```
[rx,error,errorstr,tries]=applyjoint(H,scu_id,param,servos)
```

Tipo de parâmetros	*Variável*	*Descrição*
Entrada	`H`	Handler da linha de comunicações
	`scu_id`	Identificador da unidade controladora
	`param`	Parâmetro a controlar
	`servos`	Valores de actuação
Saída	`rx`	Sequência de resposta proveniente do master.
	`error`	Código de erro
	`errorstr`	Mensagem de erro
	`tries`	Número de tentativas para o contacto

Tabela 5: Parâmetros de entrada/saída da rotina *applyjoint*.

O endereço identificativo de cada unidade (*scu_id*) é definido por hardware através do *switch* de quatro terminais presente em cada um. Na utilização da estrutura humanóide cada um tem a localização indicada na Tabela 6, de acordo com a perspectiva do próprio robot humanóide (vista de trás).

Identificador do Slave	*Endereço a definir no switch*	*Localização*
1	*0b*0001	Perna direita
2	*0b*0010	Perna esquerda (slave principal)
3	*0b*0011	Anca direita (*)
4	*0b*0100	Anca esquerda (*)
5	*0b*0101	Tronco (ainda não testado)

Tabela 6: Localização de cada unidade slave.

(*) Ainda apenas foi testado um servomotor.

O parâmetro *param* pode controlar vários aspectos tal como os indicados na Tabela 7:

Parâmetro	*Valor*	*Descrição*
PARAM_POSITION	0	Posição das juntas
PARAM_VELOCITY	1	Duração do movimento até à posição desejada
PARAM_SPECIAL	3	(Des)Activação de PWM/filtros

Tabela 7: Tipo de parâmetros a controlar numa ordem de actuação *applyjoint*.

Finalmente o campo *servos* indica o conjunto de valores a aplicar aos três servos (vector linha de três elementos) respeitante ao parâmetro indicado em *param*.
O vector linha servos para o parâmetro PARAM_SPECIAL segue a seguinte estrutura:

```
servos=[PWM on/off, filter on/off, (sem significado)]
```

O primeiro elemento é um valor booleano que diz respeito à activação/desactivação do sinal de PWM a aplicar aos motores (1 activa-os). O segundo valor booleano está relacionado com a filtragem dos dados sensoriais provenientes de cada servomotor.
Para activação do sinal de PWM e da filtragem nos três motores do slave principal (endereço 2), basta fazer:

```
>> applyjoint(H,2,3,[1 1 0])
```

Deverá confirmar a activação do PWM, dois segundos depois, pelo deslocamento do(s) servo(s) até à posição original à máxima velocidade, e pelo seu comportamento rígido na posição final. Estes dois segundos devem-se à execução de rotinas de calibração dos sensores.
Observando os sinais de controlo presentes à saída pode-se observar um sinal de onda quadrada de duty-cycle entre 5 a 10% do período (1 e 2 ms para um período de 20 ms) (Fig. 18).

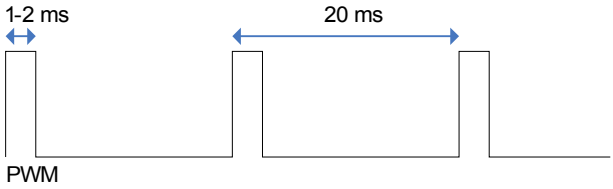


Fig. 18: Sinal de PWM aplicado no servomotor.

R Para minimizar os picos de corrente na activação dos sinais de PWM, recomenda-se o deslocamento manual dos servos para a posição original. Desta forma, o deslocamento dos servos, bem como o consumo de corrente neste instante, é mínimo.

		***Posição Original* (º)**		
Identificador do Slave	***Localização***	Junta 1	Junta 2	Junta 3
1	Perna direita	0	0	–70
2	Perna esquerda (slave principal)	0	0	+70
3	Anca direita	0	0	0
4	Anca esquerda	0	0	0
5	Tronco (ainda não testado)	0	0	0

Tabela 8: Posições originais dos servos de cada unidade.

A Tabela 8 indica as posições originais dos servos para cada unidade controladora, de acordo com o ficheiro *global.h* do programa Master. Na aplicação nas pernas, deverá corresponder à posição vertical destes membros.

? Na utilização de várias unidades slave, com a alimentação baseada numa fonte, é comum o bloqueio da comunicação CAN em algumas unidades. A substituição da fonte por baterias completamente carregadas costuma resolver o problema, sugerindo que a insuficiência no fornecimento de corrente pode ter alguma responsabilidade neste efeito.
Mesmo assim, por vezes, o mesmo acontece com baterias, mesmo aparentemente carregadas. Suspeita-se que mesmo com uma elevada tensão de saída, a corrente máxima de saída esteja em baixa. Experimente desligar o sistema durante alguns minutos para deixar as baterias recuperar.
Note que os programas são todos iguais entre slaves, pelo que a possibilidade de *bug* no *software* não faz muito sentido, pois tal só ocorre em algumas unidades.

4.5. Shut-Down do Sistema

Para desactivação do sistema é necessário seguir o seguinte conjunto de passos:

1. Desactivação do sinal de PWM (opcional);

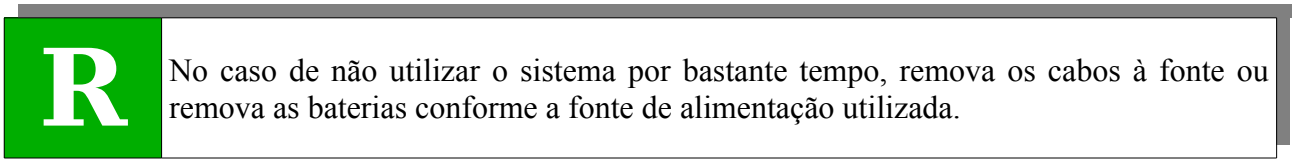
```
>> applyjoint(H,2,3,[0 0 0])
```

2. Término da linha de comunicações (opcional);

```
>> killcom(H)
```

3. Desactivação da alimentação: desligue a fonte ou as baterias;
4. Reposicionamento das juntas na posição original (ver última recomendação).

Embora a maioria dos passos sejam desnecessários, são recomendados para que a reactivação se proceda sem problemas de maior.



5. Locomoção das Juntas

Este capítulo descreve os métodos a seguir para efectuar acções de actuação sobre os servomotores, bem como para amostrar os sensores existentes nos mesmos.

5.1. Procedimentos Iniciais

Siga os passos descritos no capítulo 4 para inicialização do sistema:

1. Inicialize a posição das juntas das pernas (se as estiver a utilizar) na posição mais próxima possível da posição original;
2. Ligue a alimentação;
3. Crie a linha de comunicações utilizando o MatLab

```
>> H=initcom(1,115200);
```

4. Active os sinais de PWM das unidades slave de interesse

```
>> applyjoint(H,scu_id,3,[1 1 0]);
```

Tome nota, que o problema evidenciado sobre o bloqueio da comunicação CAN em algumas unidades Slave, também poderá ocorrer ao fim de algum tempo de funcionamento. Caso tal aconteça, desligue o sistema e volte a ligar ao fim de alguns minutos. Se mesmo assim, nada se resolver, verifique se a alimentação se encontra capaz de fornecer bons níveis de corrente.

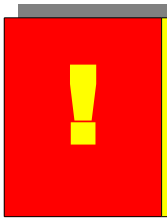
5.2. Deslocamento dos Servomotores

Para efectuar deslocamentos nas juntas, utilize a mesma rotina applyjoint, mas com o parâmetro APPLY_POSITION (Tabela 7):

```
[rx,error,errorstr,tries]=applyjoint(H,scu_id,0,[<pos1> <pos2> <pos3>])
```

A lista [<pos_1> <pos_2> <pos_3>] representa as posições desejadas que os três servomotores da unidade *scu_id* devem atingir. Como por exemplo, para deslocar os três servos da unidade 2 para as posições [0 0 50] graus, considerando que a posição actual corresponde à original ([0,0,70]), dever-se-á fazer:

```
>> applyjoint(H,2,0,[0 0 50])
```



Tome atenção que o controlo de velocidade ainda está desligado, pelo que todos os deslocamentos serão feitos à velocidade máxima. Tenha em atenção a definição das posições finais, para que não se distanciem demasiado das iniciais, para não correr o risco dos movimentos bruscos provocarem saltos nas correias de transmissão, ou mesmo, folgas na estrutura.

5.3. Definição de Velocidade

Para evitar deslocamentos à velocidade máxima, pode-se activar o controlo de velocidade utilizando para isso o parâmetros PARAM_VELOCITY (Tabela 7) na chamada à rotina *applyjoint*.

```
[rx,error,errorstr,tries]=applyjoint(H,scu_id,1,[<T1> <T2> <T3>])
```

A lista [<T_1> <T_2> <T_3>] representa a lista das durações dos movimentos a executar até à posição desejada. Note que, embora de fale em velocidade, o que realmente se controla é a duração do movimento:

- Para T=0, o controlo está desligado, e os movimentos realizam-se à velocidade máxima;
- Para T>0, o controlo está activo, executando todas as trajectórias no tempo especificado.

As trajectórias são realizadas seguindo o comportamento de uma curva polinomial de terceiro grau (Fig. 19), evitando, desta forma, a presença de descontinuidades na curva de velocidade, e de deltas de Dirac na curva de aceleração.

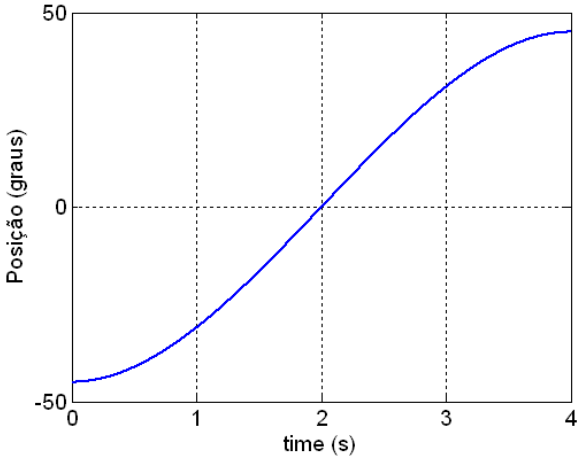
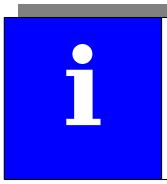


Fig. 19: Trajectória polinomial de terceira ordem.

T é especificado em períodos de PWM. Como cada período de PWM tem a duração de 20 ms (50 Hz), os valores são fornecidos em ciclos de 20 ms. Como estes valores são armazenados no formato de 8 bits (um byte), as durações limite são:

- T_{min}=1 (20ms);
- T_{max}=255 (5.1s);

$$Duração = T \times 0.02 \quad (s)$$



Caso pretenda aumentar mais a duração máxima dos movimentos, aumente a constante PER_FACTOR no ficheiro *trajectory.c* (programa slave). Esta constante representa o multiplicador do período de cada movimento: para 2, o período máximo é de 10.2s (por favor evite o uso de *floats*).

A título de exemplo, se se pretender executar movimentos suaves de duração de 2s (100 ciclos) para todos os três servomotores da unidade 2, devará fazer o seguinte:

```
>> applyjoint(H,2,1,[100 100 100])
```

E agora realize deslocamentos à vontade, tal como indicado na secção 5.2 – Deslocamento dos Servomotores.

5.4. Amostragem dos Sensores dos Servomotores

Para efectuar amostragem dos sensores presentes nos servomotores, precisará de rotinas que efectuem leituras sensoriais. A rotina *readjoint* pode ser utilizada para a leitura dos sensores presentes nos servomotores:

```
[servos,state,rx,error,errorstr,tries]=readjoint(H,scu_id,param)
```

Tipo de parâmetros	*Variável*	*Descrição*
Entrada	`H`	Handler da linha de comunicações
	`scu_id`	Identificador da unidade controladora
	`param`	Parâmetro a ler
Saída	`servos`	Valores sensoriais
	`state`	Estado de funcionamento dos servos
	`rx`	Sequência de resposta proveniente do master.
	`error`	Código de erro
	`errorstr`	Mensagem de erro
	`tries`	Número de tentativas para o contacto

Tabela 9: Parâmetros de entrada/saída da rotina *readjoint*.

Parâmetro	*Valor*	*Descrição*
PARAM_POSITION	0	Posição das juntas
PARAM_VELOCITY	1	Velocidade média
PARAM_CURRENT	2	Corrente consumida (normalizada)

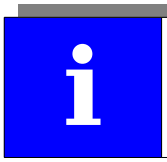
Tabela 10: Tipo de parâmetros a ler na chamada da rotina *readjoint*.

Como entradas é solicitado um parâmetro dos indicados na Tabela 10 (*param*) da unidade slave *scu_id*. No retorno, são devolvidos os mesmos parâmetros que na função *applyjoint*, adicionalmente dos dados pedidos em *servos* e do estado de funcionamento destes dispositivos (*state*).

```
servos=[<data1>, <data2>, <data3>]
```

O vector servos segue a mesma estrutura que na rotina de actuação, com cada elemento associado a cada servo:

- Caso o parâmetro pedido é de posição (PARAM_POSITION) a posição dos servos é devolvida em graus (entre -90 e +90º) com uma resolução de 1º;
- Se *param*=PARAM_VELOCITY, a velocidade média nos últimos 100ms (5 ciclos de PWM) é devolvida. Este valor é fornecido em graus/100ms.
- Se pretender medir a corrente (param=PARAM_CURRENT), é devolvido um valor normalizado entre 0 e 100 representativo da percentagem de ocupação do impulso de corrente no período de PWM. Uma média deslizante, ao longo dos últimos 100ms, é calculada tendo em vista a filtragem destes valores.



Caso pretenda aumentar a janela de tempo de medição da velocidade/filtragem de corrente, aumente o número de ciclos de PWM (ciclos de 20ms) a considerar na constante N_POS do ficheiro pic2.c (programa slave).

O vector *state* contém os seguintes dados em formato binário:

bit 7	*6*	*5*	*4*	*3*	*2*	*1*	*bit 0*
PWM flag	Filter flag	0	0	Deadline error	Motion finished		

***Status* dos servomotores**

Elemento	***Bit***	***Label***	***Descrição***
1	7	PWM flag	Sinais de PWM activos
2	6	Filter flag	Filtros sensoriais activos
3,4	5,4	0	Sem significado
5	3	Deadline error	Erro de deadline do período de PWM
6	2	Motion finished 3	Servo 3 terminou o movimento
7	1	Motion finished 2	Servo 2 terminou o movimento
8	0	Motion finished 1	Servo 1 terminou o movimento

Tabela 11: Elementos do vector state devolvido pela rotina *readjoint*.

Os bits *motion finished* podem ser utilizados em rotinas de mais alto nível para a execução de sequências de movimentos. Estes bits servem, portanto, como bloqueios antes de passar ao movimento seguinte.

Para efectuar uma leitura sensorial terá de seguir a seguinte sintax:

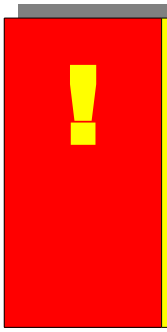
```
[servos,state]=readjoint(H,scu_id,param)
```

No caso da leitura de posição à unidade 2, apenas terá de fazer:

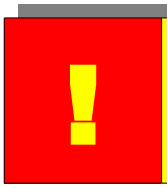
```
[servos,state]=readjoint(H,2,0)
```

5.5. O Controlador de Posição

Cada unidade Slave possui integrada, para cada servomotor, um controlador PID para compensação de desvios decorrentes da presença de cargas nos veios. Para ajuste deste controlador é possível controlar os parâmetros K_P, K_I e K_D, correspondentes às componentes integral, proporcional e derivativa respectivamente, a partir do PC usando o barramento CAN para transmitir esses dados à unidade de interesse.



Antes de prosseguir, verifique que efectuou com sucesso os procedimentos da secção 5.4 (Amostragem dos Sensores dos Servomotores), pois a posição de cada servo é usado como sinal de *feedback* para o controlador PID. É essencial que cada unidade consiga medir com sucesso estes sensores, para que o compensador possa ser utilizado!
Como medida de segurança adicional, efectue estes procedimentos primeiramente com servomotores aparte, e só depois de tudo confirmado, pode passar aos membros humanóides.



O compensador PID, tal como pode melhorar a performance dos actuadores, também pode provocar instabilização e danos à estrutura e ao próprio actuador, se mal ajustado. Siga, por isso, todas as instruções para manipulação destes compensadores.

a) Definição dos Parâmetros de Controlo

A função *applycontrol* é a responsável pelo controlo dos compensadores PID das unidades slave, podendo activar/desactivar o controlador e ajustar os seus parâmetros K_P, K_I e K_D.

```
[rx,error,errorstr,tries]=applycontrol(H,scu_id,param,servos)
```

Tipo de parâmetros	*Variável*	*Descrição*
Entrada	`H`	Handler da linha de comunicações
	`scu_id`	Identificador da unidade controladora
	`param`	Parâmetro de controlo a ajustar
	`servos`	Valores de actuação
Saída	`rx`	Sequência de resposta proveniente do master.
	`error`	Código de erro
	`errorstr`	Mensagem de erro
	`tries`	Número de tentativas para o contacto

Tabela 12: Valores de entrada e saída da função *applyjoint*.

Esta função é muito semelhante à de actuação *applyjoint*, possuindo os mesmos parâmetros de entrada e saída, com a diferença do seu significado (Tabela 12). O valor *param* identifica o parâmetro a controlar que pode ser uma das componentes do compensador PID, ou do tipo de compensador em funcionamento. A Tabela 13 apresenta a lista de parâmetros possíveis de controlar.

Param	*Valor*
Ajuste do K_I	0
Ajuste do K_P	1
Ajuste do K_D	2
Tipo de Controlo	3

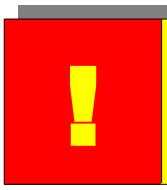
Tabela 13: Tipo de parâmetros a controlar pela rotina *applycontrol*.

Quanto ao vector *servos*, é representado na mesma forma que na função *applyjoint*: é um vector linha de três elementos com os valores correspondentes ao parâmetro *param* dos três servos.

$$servos = [servo_1, servo_2, servo_3]$$

Como por exemplo, para definir o parâmetro de compensação K_I dos três servos da unidade slave 2 como sendo *servos*=[3, 3, 3], teremos de fazer o seguinte:

```
>> applycontrol(H,2,0,[3 3 3])
```



Note o reduzido valor que atribuímos inicialmente ao parâmetro K_I. Esta é a estratégia a efectuar no ajuste: começar com valores baixos e ir aumentando à medida que a performance melhora. A partir do momento em que já não é possível melhorar mais, ou começa a deteriorar-se, chegámos aos melhores parâmetros.

b) Activação do Controlador de Posição

Note que o ajuste que fizemos, ainda não se reflecte nos servos. Ainda é necessário activar o controlador! Para tal é utilizado também a função applycontrol, mas com o parâmetro "tipo de controlo". A Tabela 14 apresenta o conjunto de controladores a activar para cada junta (conteúdo de *servos*).

Tipo de Controlo	*Valor*	*Descrição*
NO_CONTROL	0	Controlo em malha aberta
LOCOMOTION_CONTROL	1	Controlo de posição
REACTION_CONTROL	2	Controlo das forças de reacção
BALANCE_CONTROL	3	Controlo de equilíbrio no tronco

Tabela 14: Tipo de controladores a activar.

De acordo com a Tabela 14, podemos indicar se pretendemos o funcionamento em malha aberta (*NO_CONTROL*), ou em malha fechada utilizando o compensador PID referido (*LOCOMOTION_CONTROL*). Outros tipos de controladores possíveis serão discutidos no capítulo sobre equilíbrio.

Por isso, para activar o controlador de posição em todos os servos da unidade 2, basta fazer:

```
>> applycontrol(H,2,3,[1 1 1])
```

Para desactivar o controlador em todos os servos é só fazer:

```
>> applycontrol(H,2,3,[0 0 0])
```

5.6. Locomoção das Juntas na Presença de Cargas

Depois de todas as operações estarem em bom funcionamento, é altura de testar o controlador com cargas aplicadas nas juntas/servomotores.

a) Setup usando apenas um Servomotor

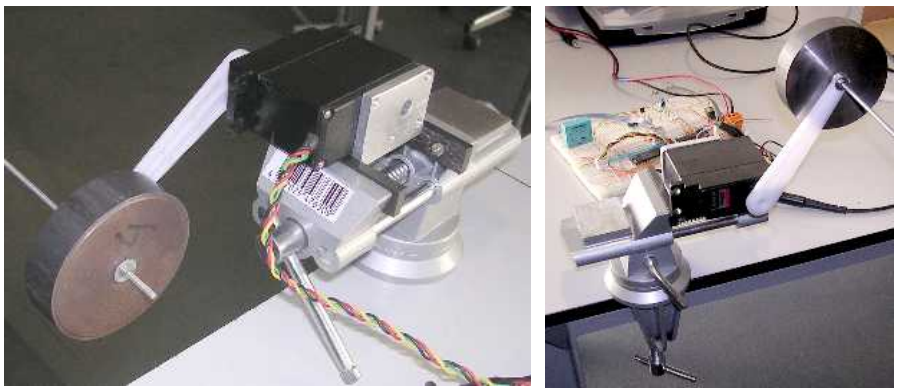


Fig. 20: Setup com um só Servomotor.

1. Monte um servomotor num torno de fixação preso à mesa tal como nos mostra a Fig. 20, e assegure-se que se encontra bem fixo;

2. Conecte o alongamento plástico ao veio do motor, de modo a que os limites -90º e +90º do servo correspondam às posições inferior e superior respectivamente;

> Quando não é possível identificar os extremos do servo (acontece quando as patilhas de segurança internas foram removidas), ligue o servomotor à placa de controlo, e posicione-o na posição 0º. Com base nessa posição, estime os extremos -90 e +90º.

3. Coloque uma massa circular no extremo do alongamento utilizando para isso o veio metálico e os parafusos de fixação.

4. Conecte o servo à primeira saída do slave (servo 1);

5. O servo está pronto para ser testado!

Fig. 21: Conjunto de massas a utilizar como carga no servo.

b) Setup da(s) Perna(s)

Caso pretenda utilizar a estrutura humanóide, pode começar pelos membros inferiores, as pernas (Fig. 22).

Fig. 22: Utilização das pernas, com carga incluída, para teste dos servomotores.

As instruções de (des)montagem das pernas, bem como de outros partes, estão incluídos no relatório de projecto 2004/05. Consulte o CD desse ano para consulta desses documentos.

Para simular o tronco, pode utilizar o peso da Fig. 23, que pesa aproximadamente 2Kg. Insira-o nas cavidades do topo de cada perna, e execute movimentos, procurando sempre garantir a horizontalidade da carga.

Fig. 23: Carga utilizada para simular o tronco.

5.7. Rotinas de Alto-Nível para Controlo da Locomoção

Embora as funções apresentadas tenham a capacidade de controlar o movimento de cada trio de juntas, por vezes pretendemos fazer mais do que isso, como por exemplo, a coordenação de movimentos entre várias unidades de controlo, podendo abranger por vezes 8 ou mais juntas.

Outra necessidade, prende-se com a visualização e análise de dados. Pretende-se também analisar as trajectórias efectuadas por todas as juntas envolvidas num determinado movimento, sendo necessário, por isso, desenvolver uma forma de amostrar a posição e outros dados, em tempo real, para visualização e comparação de dados.

a) Amostragem de Trajectórias de uma Unidade de Controlo

Rotinas especiais foram escritas[6] para amostragem de trajectórias completas, ou seja, deslocamentos desde uma posição inicial até à desejada, podendo incluir um período de tempo extra para análise da resposta em regime estacionário. A Tabela 15 apresenta esse conjunto de rotinas, que correspondem às primeiras desenvolvidas, e apenas permitem executar deslocamentos individuais apenas para os servomotores de uma só unidade de controlo.

Todas estas rotinas, geram uma matriz com toda a informação amostrada e a prevista. Para visualização dos resultados é necessário efectuar *plots* manuais das colunas de interesse destas matrizes. Todos os dados amostrados durante este ano, bem como alguns gráficos presentes no relatório, estão disponíveis no CD de projecto[7].

b) Execução de Movimentos Coordenados entre várias Unidades de Controlo

Para execução de movimentos que envolvam várias unidades de controlo, torna-se necessário comunicar e ler os dados sensoriais de todas as unidades, garantindo sempre sincronismo temporal, para que todas as juntas executem os deslocamentos pedidos no mesmo intervalo de tempo e sem desfasamentos temporais.

Três tipos de movimentos foram contemplados, envolvendo apenas as pernas:

- Flexão da(s) perna(s): envolve o deslocamento da junta do joelho e das juntas dianteiras do pé e da anca (3×2);
- Deslocamento lateral da(s) perna(s): junta lateral do pé (1×2);
- Dobra da(s) perna(s): junta do joelho e dianteira da anca (2×2).

Com o objectivo de uniformizar o software na execução dos vários tipos de movimentos, desenvolveram-se várias rotinas para o cálculo dos parâmetros necessários para a execução de cada um (Tabela 16), e uma rotina universal que utiliza esses parâmetros para sua execução e amostragem sensorial – *exe_traj*.

6 <CD_PROJ>:\Lab\Fase3_Integration\PC\Control\Motion\
7 <CD_PROJ>:\Lab\Data\

Ficheiro M	***Chamada da função***
predict_trajectory.m	`[posT,velT,accT]=predict_trajectory(pos0,posF,n_samples,ratio)` Previsão da execução de uma trajectória ideal. Com o fornecimento da posição inicial (*pos0*) e final (*posF*), mais o número total de amostras (*n_samples*) e a fracção correspondente à trajectória (*ratio*), é devolvida uma trajectória polinomial de terceiro grau em posição (*posT*), velocidade (*velT*) e aceleração (*acctT*).

sample_traj.m	`array=sample_traj(H,T,pos0,posF,k)`
	
	Amostragem da trajectória de um servo de uma unidade slave.
	A partir das informações da duração do movimento *T*, da posição inicial (*pos0*) e final (*posF*), e dos parâmetros k=[K_I,K_P,K_D], o primeiro servo da unidade slave de teste (endereço fornecido no interior da rotina – *scu*), um array (*array*) é produzido com a informação de tempo, posição, velocidade, corrente e as trajectórias previstas de posição e velocidade, dispostas em vectores coluna.
	
	T: Duração do Movimento
	pos0: Posição inicial
	posF: Posição final
	$k = [K_I, K_P, K_D]$: Parâmetros de controlo PID)
	
	`array=[tempo posição velocidade corrente posT velT]`
	Todos os elementos são vectores coluna.

sample_traj3.m	`array=sample_traj3(H,T,pos0,posF,k)`
	
	Equivalente à rotina *sample_traj*, mas com a diferença de efectuar amostragem de todos os três servomotores de uma determinada unidade slave (introduzida no conteúdo – *scu*). *Pos0* e *posF* são vectores linha de três elementos, e *k* é uma matriz de três colunas para os três tipos de parâmetros K_I, K_P e K_D, por três linhas, correspondentes aos três servomotores.
	
	T: Duração do Movimento
	$pos0 \Leftrightarrow posF = [servo_1, servo_2, servo_3]$ $\quad k = \begin{bmatrix} K_{1I} & K_{1P} & K_{1D} \\ K_{2I} & K_{2P} & K_{2D} \\ K_{3I} & K_{3P} & K_{3D} \end{bmatrix} \begin{matrix} (servo_1) \\ (servo_2) \\ (servo_3) \end{matrix}$
	
	`array=[tempo` **`posição posT velocidade velT corrente`**`]`
	Todos os elementos à excepção do tempo (vector coluna), são matrizes com três colunas correspondentes aos três servomotores.

Tabela 15: Lista de rotinas para amostragem de trajectórias de uma unidade de controlo.

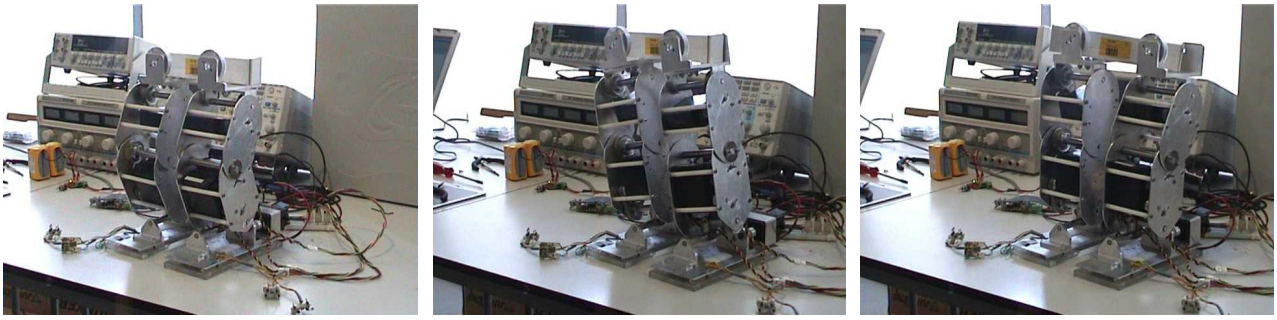


Fig. 24: Tipos de movimentos das pernas.

Ficheiro M	***Descrição***
flexao.m	`[pos_apply,k,t]=flexao(T,motion)` Com base no período de deslocamento *T* e o ângulo máximo de dobra do tornozelo *motion*, são calculados os parâmetros para execução do movimento de flexão das duas pernas.
inclinar.m	`[pos_apply,k,t]=inclinar(T,motion)` Cálculo dos parâmetros associados ao movimento lateral das pernas. Além do período de deslocamento *T* é fornecido o ângulo máximo de variação da junta lateral do pé *motion*.
levantar_perna.m	`[pos_apply,k,t]=levantar_perna(T,motion,perna)` Levantamento da perna até um ângulo máximo do joelho *motion* da perna indicada na variável *perna*. Se *perna* possuir um valor inválido (diferente de 1 e de 2), as duas efectuarão o movimento.

Tabela 16: Funções de cálculo das trajectórias para os diferentes tipos de movimentos.

Os *setpoints* da trajectória calculada por cada uma destas funções são armazenadas em três variáveis:

- *pos_apply* : Setpoints de posição (final);
- *k* : parâmetros de compensação a adoptar;
- *t* : Periodo de deslocamento.

As características destas variáveis estão descritas na Tabela 17 onde...

- N_SCU é o número total de unidades slave: 8;
- N_SERVOS é o número de servos para cada unidade: 3
- N_MOTIONS é a quantidade de movimentos: normalmente 1.

Variável	***Dimensão***	***Descrição***
pos_apply	N_SCU×N_SERVOS×N_MOTIONS (3 dimensões)	Posições finais para cada unidade, servo e movimento.
k	N_SCU×N_SERVOS×N_MOTIONS×4 (4 dimensões)	Parâmetros de compensação (4) para cada unidade, servo e movimento.
t	N_MOTIONS (1 dimensão)	Período de deslocamento para cada movimento.

Tabela 17: Descrição da estrutura das variáveis retornadas pelas funções de cálculo de trajectórias.

Embora as funções da Tabela 16 não aproveitem este recurso, é possível armazenar vários movimentos diferentes nas variáveis de setpoints da Tabela 17 (N_MOTIONS). As rotinas que sejam desenvolvidas podem aproveitar este recurso para a execução de sequências de movimentos utilizando os bits de *Motion Finished* devolvidos no array *state* (secção 5.4) para saber quando iniciar cada um dos movimentos.

Como funções para a execução efectiva dos deslocamentos temos os indicados na Tabela 18.

Ficheiro M	***Descrição***
exe_traj.m	`[tt,position,velocity,current]=exe_traj(H,t,pos_apply,Textra)` Rotina de execução dos *setpoints* calculados pelas rotinas da Tabela 16.
recycle.m	`recycle(H,speed,correct_flag)` "Reciclagem" das juntas das pernas, reposicionado-as na posição vertical. *Speed* indica a velocidade a que deve ser feita a reciclagem (em ciclos de 20 ms) e *correct_flag* está relacionada com a correcção das posições standard, de modo a acertar a postura vertical das pernas.

Tabela 18: Rotinas de execução de movimentos.

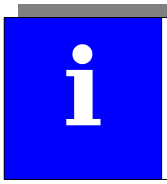
Outras funções adicionais utilizadas pelas rotinas da Tabela 17 e da Tabela 18 são apresentadas na Tabela 19. Embora não sejam de uso directo pelo utilizador, a sua edição pode revelar-se bastante útil para a correcção de pequenos desvios que podem verificar-se durante a execução das trajectórias.

!	Ao contrário das rotinas da Tabela 15, os parâmetros *K* não são passados directamente na lista de argumentos das funções de execução de trajectórias, mas são passados através da função *controller*. Verifique sempre o conteúdo desta função, na execução destas rotinas, para evitar acidentes posteriores.

R	Se na execução de trajectórias, ou na reciclagem, verificar desvios nas posições desejadas, como por exemplo a postura não vertical no procedimento de reciclagem (correct_flag deve estar a 1), edite as funções da Tabela 19 que permitem ajustar estes pequenos desvios. Estas funções permitem, deste modo, poupar algum trabalho, evitando assim o ajuste da estrutura física. Apenas quando os desvios começam a ser elevados é que convém passar à estrutura para os corrigir.

Ficheiro M	*Descrição*
controller.m	Definição e activação dos controladores PID a todas as unidades de controlo.
controller_off.m	Desactivação dos controladores PID em todas as unidades de controlo.
adaptate.m	`pos_final=adaptate(scuid,position)` Considerando que as pernas na postura vertical apresentam todas as suas juntas com a posição nula, esta função adapta os ângulos *position* para que correspondam correctamente aos servomotores da unidade *scuid* a que vão ser aplicados. Com base nas matrizes signal e offset – sinal de variação e desvio respectivamente, a posição final é calculada do seguinte modo: $position_{Final} = position \times signal + offset$ Verifique se estas matrizes possuem os valores correctos.
correct.m	`position_final=correct(scuid,position)` Correcção das posições *position* solicitadas para a unidade *scuid*, devido a desvios devido à imperfeita ligação dos servos à estrutura. Pequenos desvios da correia de transmissão ou do próprio veio do servomotor pode provocar deslocamentos, para uma determinada posição, ao longo do tempo. Actualize periodicamente a matriz *calib* de modo a reflectir esses desvios.
limit.m	`position_final=limit(scuid,position)` Limitação da posição *position* solicitada à unidade *scuid*. Além da limitação de excursão dos servos (-90 a +90º) a própria estrutura mecânica do robot aumenta ainda mais esta limitação. Confirme os dados da matriz *lim_min* e *lim_max*.

Tabela 19: Funções adicionais utilizadas pelas rotinas de cálculo e execução de trajectórias.



Embora os limites de posicionamento dos servomotores sejam de ±90º, limitou-se para ±70º devido à diferença do duty-cycle de funcionamento entre os vários actuadores. Dando uma margem de 20º garante-se que todos os servos utilizam toda a gama considerada (140º).

Exemplo da execução de um movimento de flexão das pernas, com um deslocamento de 60º durante 2s (100 ciclos de PWM):

```
>> [pos_apply,k,t]=flexao(100,60);

>> exe_traj(H,t,pos_apply,0);
```

c) Amostragem de Trajectórias em Modo de Deslocamento Coordenado

A função *exe_traj*, além de executar movimentos, também pode amostrar os sensores de todas as unidades pretendidas. Edite a função e introduza no vector scu2test as unidades a amostrar:

scu2test=[1, 2, 3, 4] → Amostragem das unidades slave de endereços 1, 2, 3 e 4

Depois execute a mesma função e guarde as matrizes de retorno:

```
[time,position,velocity,current]=exe_traj(H,t,pos_apply,Textra)
```

Variável de retorno	***Dimensão***	***Descrição***
time	Vector coluna (1 dimensão)	Vector com o intervalo de tempo do movimento.
position	N_TEST×N_SERVOS×TIME (3 dimensões)	Posições medidas para cada unidade e cada servo ao longo do tempo.
velocity	N_TEST×N_SERVOS×TIME (3 dimensões)	Velocidade média medida para cada unidade e cada servo ao longo do tempo.
current	N_TEST×N_SERVOS×TIME (3 dimensões)	Corrente média medida para cada unidade e cada servo ao longo do tempo.

Tabela 20: Parâmetros de retorno da função *exe_traj*.

Os arrays retornados não dizem respeito a todas as unidades slave, uma vez que somente as indicadas em *scu2test* são amostradas. Por isso, cada parâmetro de saída apresenta os dados sensoriais apenas para os servos amostrados (N_TEST) e na ordem designada por *scu2test*.

Para visualização dos resultados na forma de gráficos terá de extrair um conjunto de dados das matrizes, dado que a versão actual do MatLab não permite a manipulação de matrizes com mais de 2 dimensões na utilização do comando *plot*. No caso da visualização da posição terá de fazer o seguinte:

```
L=length(position(<scu_id>,<servo>,:));

data=[];
for i=1:L
    data=[data; position(<scu_id>,<servo>,i);
end

plot(time,data);
```

6. Controlo de Equilíbrio

O controlador de equilíbrio tem como objectivo, a manutenção da estrutura humanóide numa postura vertical e cujo centro de pressão se projecte sobre o centro do pé de apoio. Desta forma, é possível conferir à estrutura o estado de equilíbrio, como se de um pêndulo invertido se tratasse.

Este capítulo pressupõe que todo o sistema base, com os servomotores, já foi montado e verificado (ver secções 2.3 e 2.4) pelo que o controlador de equilíbrio é um componente adicional que, usando os sensores de força como *feedback*, actua nos actuadores segundo uma determinada lei de controlo.

6.1. Os Sensores de Força

De modo a poder avaliar a localização do centro de pressão sobre cada pé, extensómetros localizados nas quatro extremidades dos pés, são utilizados para medir a distribuição das forças de reacção ao longo da sua estrutura. Pela distribuição, é possível avaliar a posição do centro de pressão.

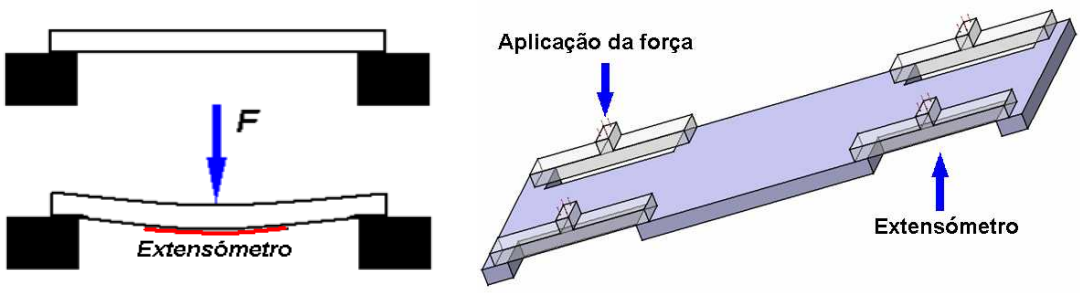


Fig. 25: Colocação dos sensores na estrutura do pé.

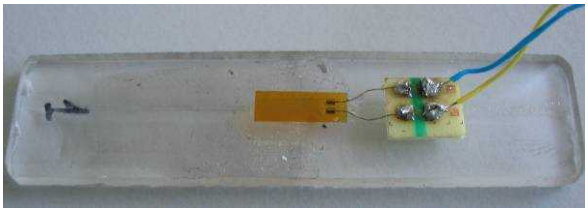


Fig. 26: Peça de acrílico contendo o extensómetro para medição da sua deformação.

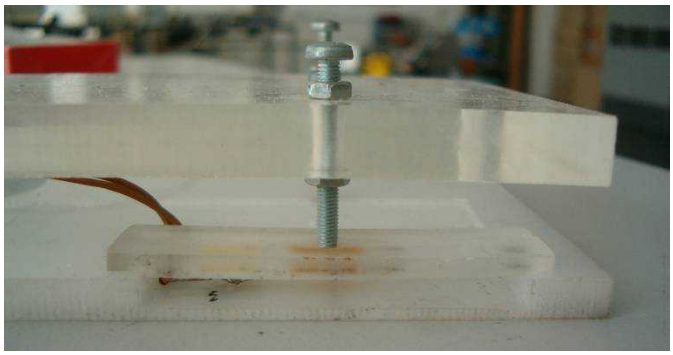


Fig. 27: Pontos de contacto entre as 2 plataformas do pé.

6.2. Montagem dos Componentes

Verifique que os pés da estrutura humanóide apresentam os sensores bem colados às peças de acrílico e que estas se encontram bem encaixadas nas suas ranhuras. Os pés devem apresentar uma configuração bastante semelhante à da Fig. 28. Verifique também que a altura dos parafusos de apoio sobre as peças de acrílico é aproximadamente igual entre os quatro, de modo a evitar desigualdades na força aplicada sobre cada um deles, no estado de equilíbrio.

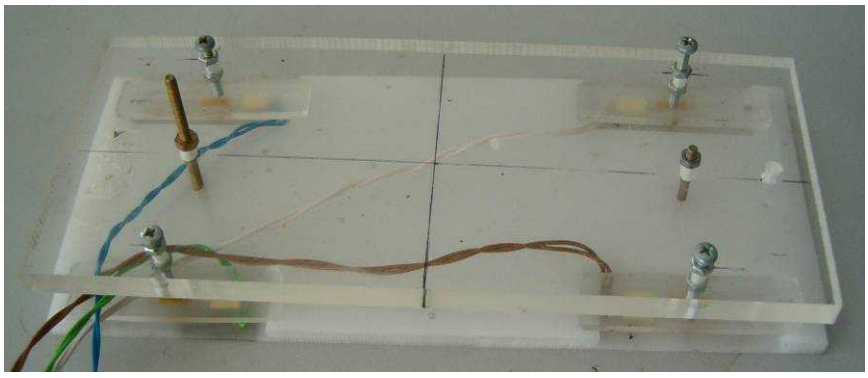


Fig. 28: Visão completa do pé.

Verifique também a presença dos extensómetros "espelho" que serão utilizados para efeitos de simetria nas pontes de *Wheatstone* da placa de acondicionamento. Esta placa (Fig. 29) além de medir a variação de resistência dos extensómetros de um dos pés, também amplifica estes pequenos sinais, de modo a poder serem utilizáveis pela unidade de controlo local. Conecte os cabos oriundos dos extensómetros (de medição das forças e "espelhos") a esta placa.

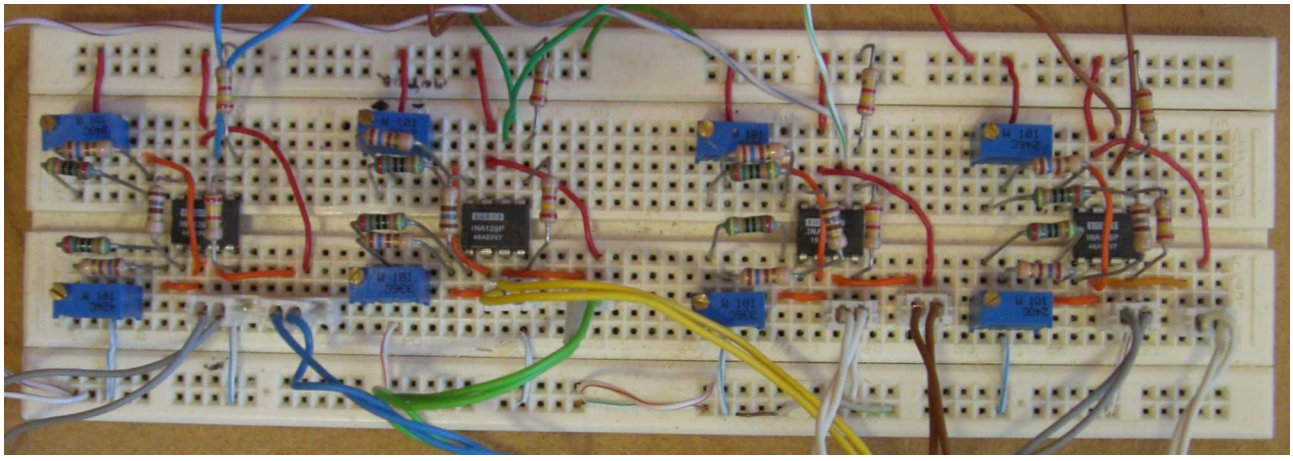


Fig. 29: Placa de acondicionamento de sinal.

As quatro saídas deste circuito, correspondentes aos sensores de um pé, são direccionados para as quatro entradas analógicas presentes em cada slave (Fig. 30).

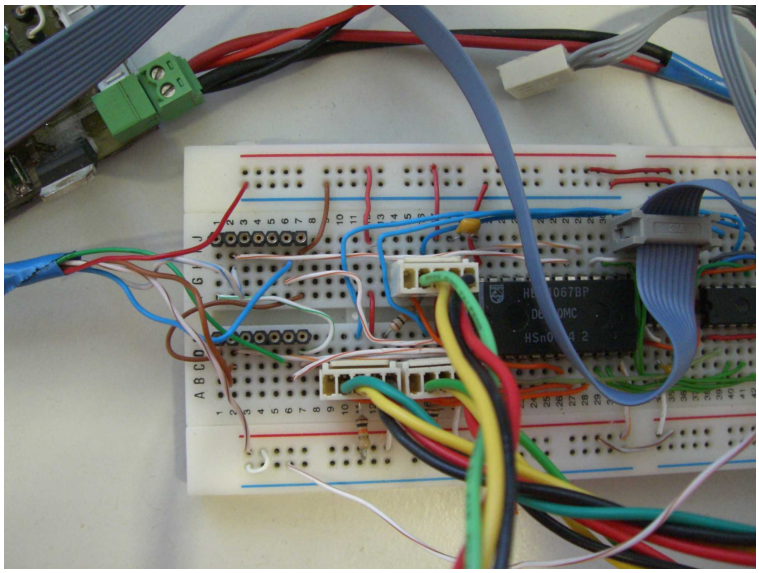


Fig. 30: Ligação das quatro entradas analógicas ao slave principal (esquerda da placa).

Para ligação das saídas analógicas às placas slave em PCB, é necessário inserir os fios nas réguas de terminais destinadas às placas *piggy-back* (terminais que fazem ligação ao *multiplexer*). Posteriormente poder-se-á construir o circuito de acondicionamento em PCB de modo a poder encaixar nas réguas de terminais.

A Fig. 31 apresenta um diagrama do esquema de ligações a fazer desde os extensómetros até à unidade de controlo local.

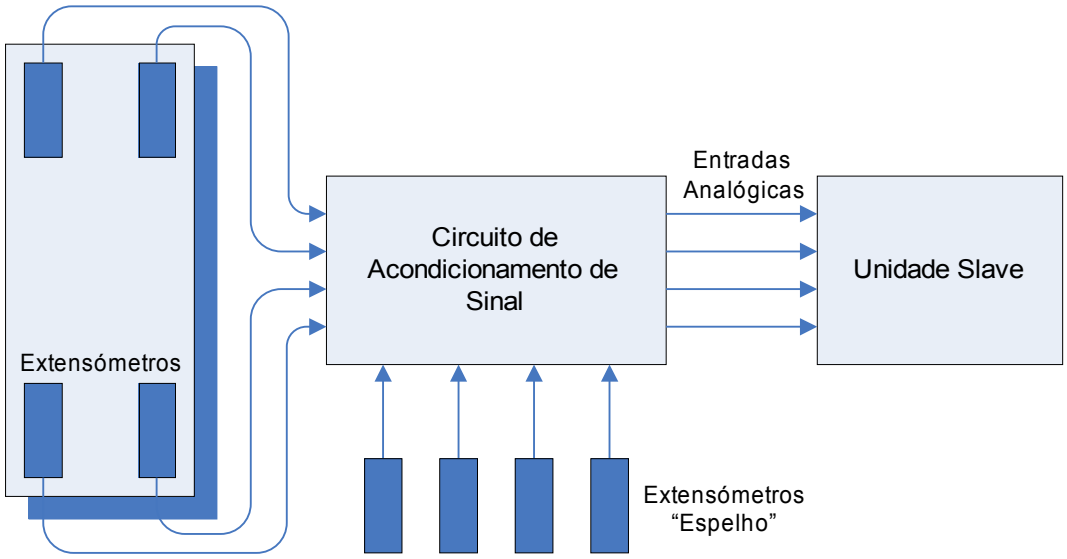


Fig. 31: Ligações para medição dos sensores de pressão de um pé.

6.3. Verificação das Ligações

Relativamente aos extensómetros:

1. Certifique-se que as placas de acrílico onde se alojam os extensómetros estão em boas condições;

2. Verifique a colocação dos extensómetros nas placas de acrílico (os sensores devem estar bem colados);

3. Confirme com o multímetro a ligação dos sensores à placa de acondicionamento de sinal (não esquecer os extensómetros "espelho").

4. Verifique a resistência dos potenciómetros superiores da placa de acondicionamento de sinal (Fig. 29). Devem possuir metade do seu valor máximo: 50Ω.

5. Alimente apenas o circuito de acondicionamento com 5V (verifique as polaridades), com os sensores em vazio, ajuste os potenciómetros de calibração de modo a que a saída do amplificador corresponda a 1V. Os potenciómetros de calibração localizam-se na parte inferior da placa (Fig. 29). Caso a saída nunca varie, é porque existe algum mau contacto na placa – verifique novamente as ligações (passo 3).

R O circuito de acondicionamento de sinal foi implementado numa placa experimental de baixa qualidade, pelo que será de esperar alguns problemas nas ligações.
Para evitar qualquer problema, recomenda-se a reimplementação cuidada numa placa PCB. Se possível use fichas RJ45 de quatro terminais (2 para cada extensómetro + 2 para o respectivo "espelho").

6. Depois de calibrado, pressione propositadamente nos sensores de pressão, e deverá observar variação no sinal de saída do amplificador entre 1 e 4V. Se verificar bastantes oscilações no sinal de saída, volte ao passo 3 com a verificação das ligações.

i Dada a margem de 1V na saída dos amplificadores INA129P, apenas obtemos variações na gama de 1 a 4V, quando alimentado entre 0 e +5V.

6. Desligue a alimentação, e verifique a ligação entre a saída dos amplificadores e as entradas analógicas da unidade slave, bem como destas às correspondentes entradas no multiplexer.

Extensómetro	Entrada do MUX	Código de selecção	Pino do C.I.
Superior esquerdo	Y_6	*0b*0110	3
Superior direito	Y_7	*0b*0111	2
Inferior esquerdo	Y_{14}	*0b*1110	17
Inferior direito	Y_{15}	*0b*1111	16

Tabela 21: Ligação das saídas de amplificação (perspectiva do robot) ao multiplexer.

6.4. Activação do Sistema

Depois da validação de toda parte física e eléctrica, estamos preparados para colocar o sistema em funcionamento e em operação:

1. Ligue a fonte de alimentação/baterias;
2. Coloque a perna na postura vertical;
3. Calibre as pontes de Wheatstone, através dos potenciómetros inferiores (Fig. 29), de modo a que os amplificadores (saídas do circuito) apresentem uma tensão de 1V;
4. Teste o funcionamento dos sensores premindo-os;
5. Crie uma linha de comunicações e active os sinais de PWM, através dos procedimentos indicados no capítulo 4.

Quando o sistema é ligado, ele coloca-se no modo *stand-by* à espera que o utilizador posicione a perna na postura vertical e calibre as pontes de *Wheatstone*, de modo a que todas as saídas apresentem o mesmo valor.
Quando o utilizador activa os sinais de PWM, o sistema pressupõe que a estrutura está em equilíbrio, e logo a saída de todos os sensores deve ser igual. Com esse pressuposto, efectua uma segunda calibração, por *software*, de forma a garantir que todos os sensores apresentam as mesmas saídas. Este procedimento dura cerca de dois segundos, daí o atraso no arranque dos servomotores.

6.5. Leitura Sensorial dos Sensores de Pressão

Para a leitura sensorial dos sensores de pressão é utilizada a função *readspecial*. Esta função foi construída tendo em mente a leitura das quatro entradas analógicas adicionais (Tabela 21) que também poderão ser utilizadas no futuro para a leitura de inclinómetros/giroscópios.

```
[special,rx,error,errorstr,tries]=readspecial(H,scu_id)
```

Tipo de parâmetros	*Variável*	*Descrição*
Entrada	`H`	Handler da linha de comunicações
	`scu_id`	Identificador da unidade controladora
Saída	`special`	Valores sensoriais medidos a partir dos quatros sensores.
	`rx`	Sequência de resposta proveniente do master.
	`error`	Código de erro
	`errorstr`	Mensagem de erro
	`tries`	Número de tentativas para o contacto

Tabela 22: Parâmetros de entrada/saída da rotina *readspecial*.

$$special = [sensor_1, sensor_2, sensor_3, sensor_4\ ]$$

Esta função retorna os valores pré-processados (calibrados por *software*) dos quatro sensores especiais ligados à unidade controladora *scu_id*.

Exemplo – leitura dos sensores de pressão da unidade de controlo 2:

```
>> special=readspecial(H,2)
```

Três rotinas de mais alto nível foram desenvolvidas[8] tendo em vista o cálculo, armazenamento e representação gráfica do centro de pressão CoP.

Ficheiro M	***Descrição***
calccop.m	`CoP=calccop(react)` Com base nos dados sensoriais dos sensores de pressão (vector *react*) o centro de pressão *CoP* é calculado e retornado na forma de coordenadas rectangulares (*x*,*y*).
cop_realtime.m	(Rotina em formato script) Apresenta a localização do centro de pressão na base do pé, em tempo real, através de uma interface gráfica representativa de um pé.
cop_acquire.m	(Rotina em formato script) O armazenamento dos sensores de pressão (*reaction*) e do respectivo centro de pressão (*CoP*) é realizado durante um determinado período de tempo especificado pelo utilizador em *maxtime* (edit o *script*). Também a posição (*servo*) e a corrente consumida (*current*) da unidade slave local é capturada e armazenada. Entrada: *maxtime*: Tempo de captura Saídas: *reaction*: valores dos sensores de pressão (dimensão 4×TIME); *CoP*: localização (*x*,*y*) do centro de pressão (dimensão 2×TIME); *servo*: trajectória de posição realizada pelos 3 servos (dimensão 3×TIME) *current*: corrente consumida para os três servos (dimensão 3×TIME).
cop_graphs.m	(Rotina em formato script) Visualização gráfica de um ficheiro workspace com as variáveis produzidas pela rotina *cop_acquire*. Após produzir estas variáveis salvaguarde-as num ficheiro, e edite esta rotina para ler este ficheiro sempre que precisar.

Tabela 23: Rotinas de alto-nível para medição das forças de reacção.

Para salvaguarda dos arrays de saída da rotina cop_acquire pode utilizar o comando *save*:

```
>> save filename reaction CoP servo current
```

Para posterior leitura, é só fazer:

```
>> array=load('filename');
>> reaction=array.reaction;
>> CoP=array.CoP;
>> servo=array.servo;
>> current=array.current;
```

Quanto à rotina *cop_realtime*, o centro de pressão é representado graficamente num pé. A Fig. 32 mostra-nos um exemplo: a perspectiva considerada é a do robot – a parte superior corresponde à frontal – e apresenta-nos um caso em que o centro de pressão coincide com o centro do pé.
A rotina *cop_graphs* pode ser utilizada para a visualização gráfica dos resultados produzidos pelo *cop_acquire* após salvaguarda em ficheiro. Edite esta rotina de modo a fazer o *loading* do ficheiro salvaguardado e utilizar os seus dados.

8 <CD_PROJ>:\Lab\Fase3_Integration\PC\Control\CoP_realtime

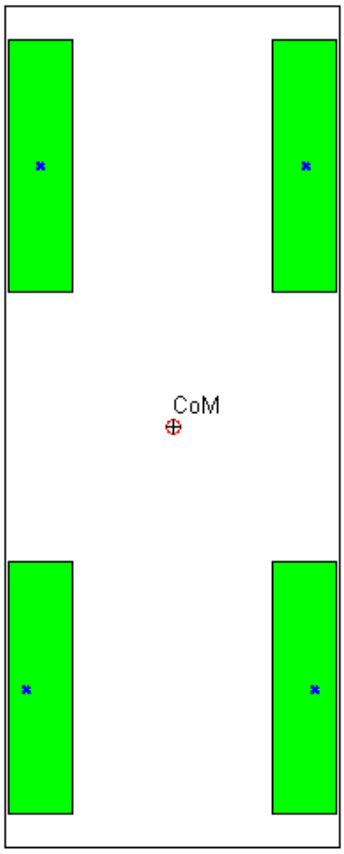


Fig. 32: Representação gráfica do CoP.

6.6. Algoritmos de Controlo

Quando os sinais de PWM são ligados, o sistema é inicializado em modo de malha aberta, ou seja, nenhum controlador se encontra activo. Para activação do controlador de equilíbrio é necessária a função *applycontrol* usando o parâmetro *param* com o valor 3 – definição do tipo de controlador – tal como nos indica a Tabela 13.

```
[rx,error,errorstr,tries]=applycontrol(H,scu_id,3,[servo1 servo2 servo3])
```

Relativamente aos tipos de controlador a que podemos atribuir a cada servo, através do array *servo*, a Tabela 14 apresenta-nos dois algoritmos que utilizam as forças de reacção como sinais de *feedback*:

- REACTION_CONTROL (2): Controlo segundo a lei proporcional;
- BALANCE_CONTROL (3): Controlo utilizando a matriz Jacobiana.

Ambos os controladores possuem o mesmo objectivo: colocar o centro de pressão no centro do pé de apoio, o que é o mesmo que dizer, manter a perna na vertical, quando apenas uma é utilizada. Isto permite fazer experiências, como a variação do plano de suporte do pé, para teste dos compensadores (Fig. 33).

Repare que a unidade de controlo local a que se ligam os sensores de força, também é capaz de controlar as três juntas da perna – as duas do pé mais o joelho – pelo que apenas uma unidade slave é necessária para cumprir o objectivo proposto.

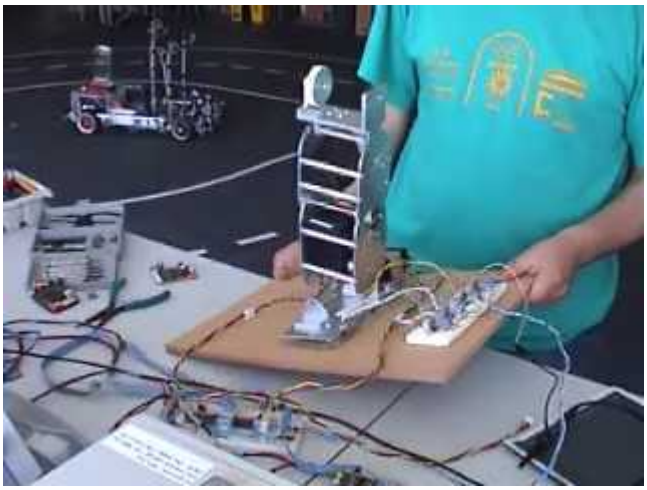


Fig. 33: Equilíbrio de uma perna com a variação do declive do plano.

Apenas active o controlador de equilíbrio para as unidades que possuem entradas especiais válidas, como é o caso das unidades relativas às pernas (endereços 1 e 2) que utilizam os sensores de força dos pés.
A activação por parte de outras unidades, leva a comportamentos imprevistos, uma vez que não estão ligados a nenhum sensor e, por isso, medem valores aleatórios.

Para o nosso caso em concreto, apenas faz sentido manipular as unidades de endereços 1 e 2. Utilizando a placa slave principal (slave 2), podemos activar cada um dos dois tipos de compensador do seguinte modo:

- Compensação proporcional: `>> applycontrol(H,2,3,[2 2 2])`
- Compensação utilizando o Jacobiano: `>> applycontrol(H,2,3,[3 3 3])`

Com a versão actual do software, ainda não é possível controlar os parâmetros de compensação via PC. Por enquanto, estes parâmetros só podem ser definidos pelas constantes *KP_FORCEX* e *KP_FORCEY*, ou através de *jacobKp*, dependendo do tipo de controlador em uso, incluídas no ficheiro *servo.c* (programa slave).
Num futuro próximo poder-se-á aproveitar os parâmetros K_P utilizados para a compensação de posição, para uso deste controlador. Desta forma, o destino dos parâmetros de compensação dependeria do controlador activo.

Para testar novos algoritmos de controlo, sem ter que os implementar nas unidades de controlo, pode utilizar o simulador cinemático disponível em...

<CD_PROJ>:\Lab\Simulation\Simulator

... que imita o funcionamento de uma perna do robot humanóide. Todos os aspectos físicos presentes na perna, incluindo os dos servomotores, foram considerados para que a simulação seja o mais realista possível.

Edite e corra o script *simleg.m* para considerar as suas condições de funcionamento.

Índice de Figuras

Índice de Tabelas