Desenvolvimento de Ajudas Computacionais para a Operação e Controlo de um Robot Humanóide

Relatório Final da Bolsa B.I.I.C.

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

Julho de 2007

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABELAS

I. INTRODUÇÃO

Resumo:

Neste trabalho pretende-se continuar o desenvolvimento de estratégias e algoritmos de controlo usando os sensores de força presentes nos pés, bem como uma revisão de toda a arquitectura de comunicações de modo a permitir a troca eficiente de informação entre todos os nós.

Este robot compreende 22 graus de liberdade com o peso e as dimensões adequadas à participação no concurso RoboCup mundial, modalidade humanóide. Espera-se que os frutos deste trabalho contribuam para tal ambição.

1. INTRODUÇÃO

A concepção de um robot Humanóide constitui um dos maiores desafios na área da robótica: construir um ser artificial antropomórfico semelhante ao homem é um sonho inato do nosso engenho, e não é para menos, pois o ser humano é a forma de vida mais complexa existente à face da Terra. O século XX encheu a nossa imaginação com livros e filmes que demonstram esse sonho do ser artificial capaz de, além de ajudar a desempenhar tarefas, aprender coisas por si mesmo, interagir connosco, expressar emoções, possuir uma consciência própria... tudo características que por enquanto ainda consideramos como puramente humanas. Marcas como a Honda, a Sony ou a Fujitsu já deram os primeiros passos no desenvolvimento de máquinas que imitam os comportamentos físicos dos seres humanos… caminhar, dançar ou pegar objectos. Outros passos também já foram dados no que respeita ao processamento de visão, de som, com o objectivo de realizar tarefas ou simplesmente de interagir com o ser humano.

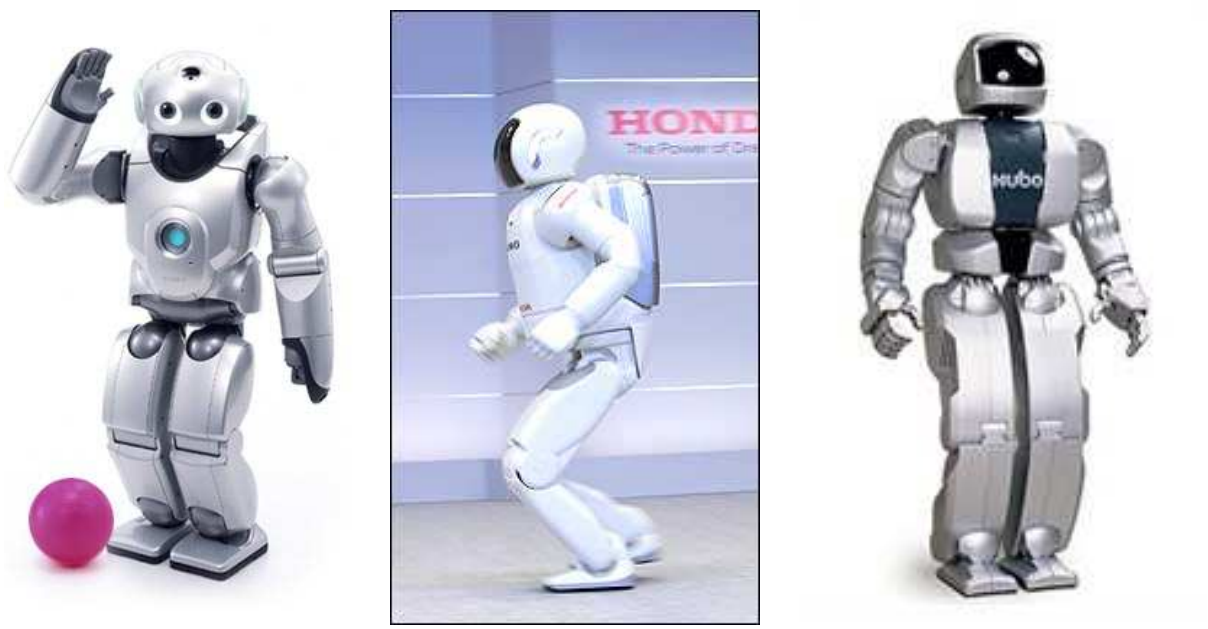


Fig. 1: QRIO (Sony), Asimo (Honda) e KHR 3 (Hubo Lab).

2. ENQUADRAMENTO DO PROJECTO

Muitos outros grupos de investigação iniciaram a construção de robôs de baixo custo no sentido de realizarem investigação em áreas tão diversas como o controlo, a percepção, a navegação, o comportamento ou a cooperação. Este foi, também, o móbil principal que levou um grupo, resultante da cooperação do D.E.M. com o D.E.T.I. da Universidade de Aveiro, a encetar a tarefa de construção de uma tal plataforma. O estado actual de desenvolvimento perspectiva a abordagem de algoritmos eficientes ao nível do controlo, planeamento e percepção.

A motivação para os projectos propostos na área da robótica humanóide é encontrada em diversas vertentes, das quais se destacam as seguintes:

- A aposta nos robôs humanóides como a via mais promissora para chegar a sistemas de elevada mobilidade, versatilidade de operação e facilidade de interacção com os humanos;
- A criação de uma plataforma de investigação de grande valor pedagógico face aos enormes desafios científicos e técnicos, à diversidade de problemas, ferramentas e níveis de integração;
- A promoção do envolvimento de um grupo de estudantes da UA em competições robóticas internacionais. Por exemplo, o ROBOCUP e o FIRA são duas organizações internacionais que realizam anualmente competições na classe dos humanóides.

3. OBJECTIVOS E MOTIVAÇÕES DO TRABALHO

Este trabalho está inserido numa bolsa de iniciação à investigação científica (BIIC) iniciada em Março e finalizada em Julho de 2007, com a finalidade de realização de trabalho de investigação no âmbito do Projecto Humanóide da Universidade de Aveiro (PhUA), e visa a concepção e o desenvolvimento de ajudas computacionais para a operação e o controlo de um robot humanóide, mais concretamente na parte da utilização dos sensores de força presentes nos pés para a detecção de eventos importantes e de realização de movimentos de locomoção.
De referir que tal constitui uma continuação do trabalho desenvolvido no ano lectivo 2005/06 que se centrou sobretudo no desenvolvimento de algoritmos de controlo para a locomoção usando servomotores como actuadores, e os respectivos potenciómetros medidores da sua posição angular como sinal de retorno. Na parte final desse trabalho iniciou-se o estudo dos sensores de força, presentes na base dos pés, utilizando-os unicamente para o equilíbrio estático da plataforma humanóide. Com este trabalho, pretende-se ir um passo mais além e efectuar equilíbrio dinâmico, ou seja, realizando movimento usando desta vez os sensores de força como sinal de retorno, de forma a assegurar, ao mesmo tempo, o equilíbrio da estrutura. Adicionalmente também se pretende trabalhar sobre a arquitectura de comunicações entre as unidades de controlo de baixo nível e a unidade principal, de modo a adaptar o protocolo de comunicações às novas exigências requeridas pelo controlo baseado em sensores de força, bem como resolver alguns problemas deixados em aberto no último ano lectivo relativos à robustez e fiabilidade das comunicações. Pretende-se assim uma gestão eficiente da largura de banda da rede utilizada, tal como um funcionamento robusto que permite a operação continuada duranate períodos de tempo indefinidos.

O trabalho proposto pode ser dividido nas seguintes etapas:

1. Modificações nos módulos de comunicação RS-232 e CAN tendo em vista a adaptação do protocolo aos controladores de força/equilíbrio, e a melhoria do desempenho e da robustez:
   - Alterações na arquitectura de comunicações CAN para lidar com novas exigências do ponto de vista de mensagens de controlo e leituras sensoriais;
   - Alteração do protocolo de comunicação CAN de modo a conferir-lhe elevada robustez e maximização da largura de banda disponível;

2. Estudo e implementação de novos algoritmos de controlo baseados em realimentação sensorial: sensores de força nos pés e estimativa da corrente consumida pelos actuadores;
   - Análise dos sensores de força para a detecção de impactos sobre o solo, de forma a ser possível a percepção dos pés no toque ao solo;
   - Utilização dos algoritmos de controlo baseados nos sensores de força, mais concretamente no baseado na matriz Jacobiano do Centro de Massa, para a realização de trajectórias (pelas pernas) por variação do ponto de referência do Centro de Pressão sobre cada pé;

3. Avaliação do desempenho da arquitectura distribuída com destaque para a funcionalidade das comunicações CAN e as capacidades do controlo local.

Resumindo os principais objectivos a atingir, aponta-se o melhoramento do *firmware* presente nos microcontroladores, tendo em vista a integração dos diversos controladores e a elevada fiabilidade de funcionamento, tal como a identificação dos problemas envolvidos na utilização dos sensores de força nos algoritmos de detecção (de eventos) e controlo (de locomoção), e respectivas soluções. Estes dados serão importantes para o desenvolvimento de um segundo protótipo que se pretende mais robusto e fiável.

O cumprimento do primeiro objectivo possibilitará uma eficiente comunicação entre o computador central e as unidades de controlo local, com a troca de todas variáveis necessárias para a correcta gestão dos movimentos a executar e dos conteúdos a percepcionar pelos diversos sensores espalhados pela estrutura humanóide.
Relativamente ao segundo objectivo – controlo das juntas com base nos sensores de força – será possível a realização de trajectórias de locomoção pelas pernas, seja a execução de um passo, a manobra de rotação ou a subida de um degrau, através de um algoritmo dinâmico que apenas precisa de um conjunto reduzido de pontos de referência para executar a trajectória pretendida, garantindo simultaneamente o equilíbrio da estrutura. De outra forma, sem a utilização destes controladores, seria necessário especificar uma trajectória para cada junta de uma forma estática, com base em cálculos analíticos feitos *à priori* para a plataforma em causa. A grande desvantagem desta última estratégia é a grande incapacidade de adaptação a mudanças na geometria e peso do sistema, o que poria em causa a sua estabilidade, dado que não existiria qualquer sinal de retorno indicando esta mudança.
Usando os controladores de força, é possível realizar uma manobra só com base em dois pontos de referência, que podem ser cartesianos, como por exemplo as coordenadas intermédias e finais do pé que executa o movimento, que, pela variação do Centro de Pressão referência aplicado ao controlador, a perna em questão realizaria a manobra, compensando, através dos sensores de força, quaisquer desvios à manutenção do equilíbrio, quer tenha havido mudanças estruturais na plataforma, quer alguma perturbação externa tenha sido aplicada, como por exemplo um toque ou um empurrão. Adicionalmente, usando igualmente os sensores de força, é possível a detecção da chegada do pé ao solo, o que permitirá a possibilidade de executar vários passos de locomoção de uma forma dinâmica, independentemente das características do solo (variação de declive e irregularidades): cada passo apenas terminaria, dando início a outro, após o pé encontrar o solo, e não através de parâmetros temporais estáticos.
Com estas funcionalidades, poderemos ter um robot bastante adaptável ao ambiente que o rodeia, com capacidades de reacção e de compensação da sua postura em resposta a estímulos externos, quer sejam provenientes do solo, como da acção directa com um ser humano.

Com base nos objectivos traçados, organizou-se este relatório em quatro capítulos:

- Capítulo I: **Introdução**
  Apresentação do enquadramento e objectivos do trabalho.

- Capítulo II: **Arquitectura das Comunicações**
  Descrição da arquitectura física e do protocolo aplicado nas redes de comunicação presentes no robot, de forma a tirar partido de todas as suas capacidades.

- Capítulo III: **Locomoção baseada em sensores de força**
  Apresentação das técnicas de percepção e de controlo de forma a fazer uso dos sensores de força para as tarefas de detecção de eventos e realização de movimentos de locomoção.

- Capítulo IV: **Notas finais**
  Conclusões finais e outras informações úteis para a continuação deste trabalho.

O primeiro capítulo fará um breve resumo das características da plataforma humanóide, tendo como objectivo essencial a familiarização do leitor para com o sistema em estudo. Os capítulos com o desenvolvimento do trabalho, nomeadamente o II e o III, apresentam inicialmente um breve resumo com os objectivos a atingir, e no final, uma secção conclusiva com a análise de resultados e conclusões tiradas. O último capítulo descreverá de uma forma mais global as conclusões obtidas nos capítulos anteriores, e aponta as novas oportunidades que este projecto abriu em prol do futuro.

4. DESCRIÇÃO DA PLATAFORMA HUMANÓIDE

A plataforma humanóide possui um conjunto de 22 graus de liberdade, distribuídos da seguinte forma:

- 2 em cada pé (2x2);
- 1 em cada joelho (1x2);
- 3 em cada anca (3x2);
- 2 no tronco (2x1);
- 3 em cada braço (3x2);
- 2 no suporte da câmara (cabeça) (2x1).

A estrutura é constituída essencialmente por alumínio e aço nos eixos e outros pequenos componentes, pesando um total de 6 Kg com as baterias incluídas, e medindo cerca de 60 cm de altura. Estes valores foram estabelecidos de acordo com as regras impostas pelo RoboCup, baseando-se no pressuposto de que para valores superiores estes, o uso de servomotores de baixo custo poderia tornar-se inviável dada a impossibilidade de conciliar binários de motores e pesos dos equipamentos e acessórios como as baterias.
Por razões de estética e de acomodação de componentes, foi adoptada uma estrutura em forma de exoesqueleto (carapaça) dotando assim o sistema de módulos ocos onde são alojados os motores, sensores, cablagens, placas de controlo, etc (Fig. 2).

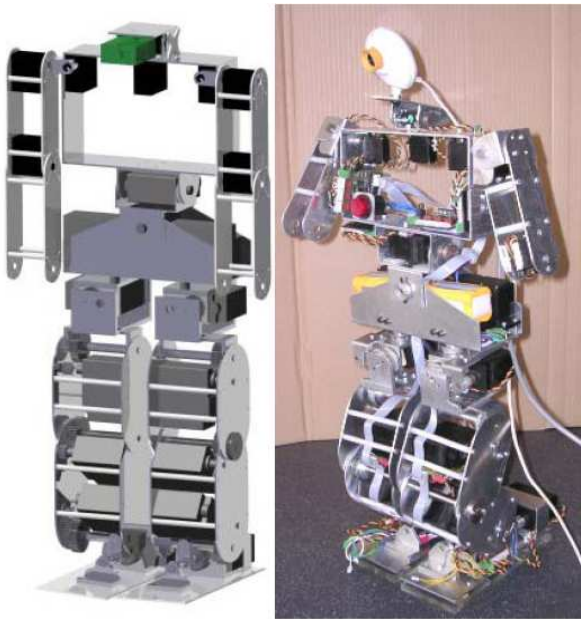


Fig. 2: Modelo 3D do robot e implementação actual.

Preocupações com a autonomia energética do sistema, exigiram a escolha de baterias de elevada capacidade de corrente, pois será necessário alimentar 22 motores de binário apreciável (embora uns mais do que outros). Tendo em conta esses aspectos utilizaram-se duas baterias de 7.4V ligadas em paralelo com uma capacidade total máxima de 9600mAh.

No que respeita ao controlo, foram assumidas, desde logo, as vantagens de uma arquitectura distribuída e modular baseada num barramento CAN responsável por permitir a troca de informação entre as diversas unidades de controlo a partir de uma unidade mestre que faz a gestão da rede e que está directamente ligada a uma unidade primária de decisão que, neste momento, é um computador comum. Posteriormente substituir-se-á o computador por uma *embedded motherboard* do tipo PC-104 ou nano ITX com as mesmas funcionalidades que um PC, no que respeita às tarefas a executar, mas com as vantagens de dimensões reduzidas e de baixo custo.

Neste momento, a plataforma é constituída por 9 unidades de controlo, 8 de controlo local dos actuadores e sensores, e um de controlo de tráfego na rede CAN. As unidades de controlo local estão distribuídas de

forma a agrupar conjuntos de três actuadores relativos a um determinado membro, como é o caso das pernas ou dos braços (Fig. 5). Com uma arquitectura deste género, pretende-se que o hardware que constitui estes módulos seja idêntico com um software muito similar entre eles. Implementando esta estratégia consegue-se um grau de fiabilidade superior, uma vez que os módulos são independentes permitindo que as anomalias sejam mais facilmente detectadas e corrigidas. Os módulos podem ser trocados facilmente, pois bastará programar o microprocessador com o *firmware* padrão que ele mesmo se adaptará à tarefa necessária, tanto através do endereço atribuído como pela gestão directa da unidade central de processamento.

Em resumo, apresentam-se as vantagens da arquitectura distribuída:

- Sistemas fiáveis (operação independente)
- Sistemas de controlo mais simples
- Mais fácil detecção de anomalias
- Actualização fácil de firmware

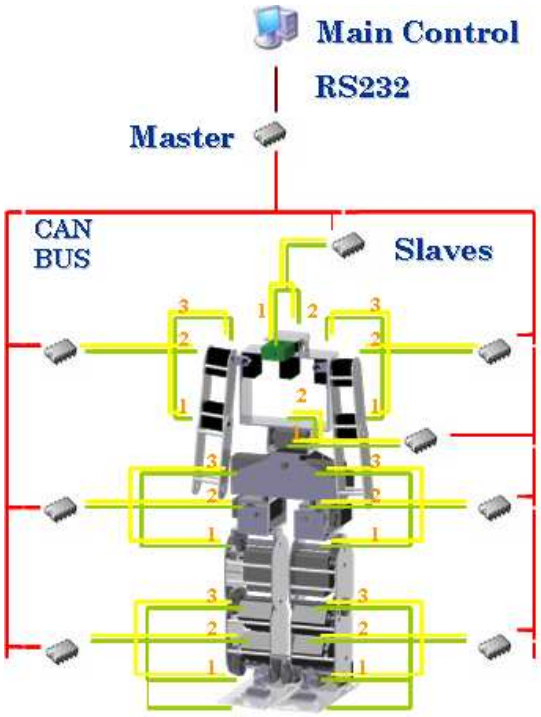


Fig. 3: Arquitectura distribuída da plataforma.

Na percepção destacam-se os sensores proprioceptivos e alguns inerciais:

1. Potenciómetros de medição da posição das juntas;
2. Sensores de força para medição das forças de reacção dos pés;
3. Inclinómetros e giroscópios para medição da aceleração gravítica e da velocidade angular respectivamente serão incorporados num futuro próximo.

Os sensores do tipo 2 e 3 serão designados por sensores especiais dada a sua natureza específica e podem ser ligados um conjunto máximo de quatro sensores a cada unidade de controlo local. Tal compromisso deve-se a questões de organização da informação trocada entre as diversas unidades de controlo.

II. ARQUITECTURA DAS COMUNICAÇÕES

Resumo:
Este capítulo descreve a organização distribuída da plataforma humanóide e como é realizada a comunicação entre os diversos nós de modo a poder trocar informação sensorial e de actuação sem o risco de colisões num barramento de natureza partilhada.

1. ARQUITECTURA DO SISTEMA

O sistema de controlo implementado é baseado numa configuração *master*/*slave*, e é constituído por três tipos de unidades ligadas em rede:

- A **unidade central** de controlo é responsável pela gestão global dos procedimentos, efectuando o cálculo das configurações que as juntas tem de adoptar em função dos valores dos sensores.
- A **unidade Master (mestre)** tem como principal tarefa distribuir os comandos provenientes da unidade principal pelas diversas unidades locais (slaves), bem como direccionar os dados sensoriais provenientes dos slaves para a unidade principal.
- As **unidades Slave (escravo),** cujas principais funções são a geração da onda de pulso modulado (PWM) de controlo dos servomotores e a aquisição dos sinais dos diversos sensores da plataforma.

Entre os diversos nós são utilizados como meios de comunicação:

- Linha série ponto-a-ponto, baseada na norma **RS-232**, entre a unidade central e a unidade *Master*: acesso assíncrono byte a byte a um *baudrate* de 115200 bps.
- **CAN** (*Controller Area Network*) entre a unidade *master* e as unidades *slave*: é utilizada a versão *full*CAN 2.0A a uma taxa de transmissão/recepção de 1Mbps.

A unidade central de controlo ainda não está completamente definida, permanecendo em aberto soluções baseadas em PDA, placas de controlo genéricas (como as baseadas no padrão PC104) ou placas de controlo dedicadas (Fig. 4). Por enquanto é utilizado um PC externo com recurso ao software *MatLab* para enviar e receber dados por uma linha série para o controlador *master*.

Para as unidades de controlo local (*master*/*slave*), a escolha recaiu sobre os microcontroladores PIC da série 18F da *Microchip* – PIC18F258(0) – por possuírem diversos periféricos e interfaces para redes de comunicações, incluindo o CAN (Fig. 6).

Fig. 4: Exemplo de uma board PC-104.

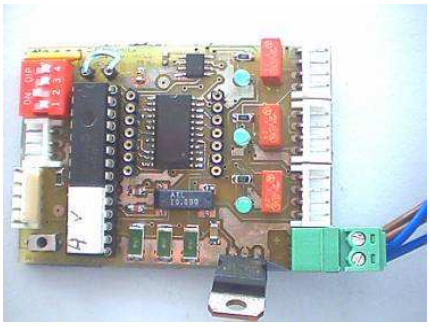


Fig. 5: Placa de controlo Master/Slave.

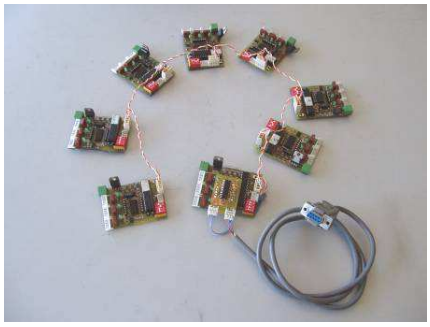


Fig. 6: Rede completa de Microcontroladores.

Até ao momento, a rede implementada é constituída por uma placa *master*, pelo qual designaremos por MCU (*Master Control Unit*), que efectua a interface entre a unidade principal e as unidades *slave*, e oito placas *slave*, designadas por SCU (*Slave Control Unit*), que efectuarão o controlo local até três actuadores através da geração de uma onda de pulso modulado em largura (PWM), e a aquisição de até 16 sinais analógicos usando um *multiplexer* (Fig. 5).

A organização enunciada na tem como objectivo agrupar as juntas que estão directamente relacionadas, como é o caso das juntas do tornozelo e do joelho que possuem um controlador dedicado, que, por aquisição dos sinais analógicos dos sensores de força instalados nos pés, pode controlar o equilíbrio por compensação em malha fechada. Na presente data já é possível fazer a adaptação a irregularidades do solo, com as juntas a corrigir a sua posição de forma reactiva para que a projecção do centro de massa do robot não se situe fora da área de apoio dos pés. Obtém-se assim um controlo localizado independente do resto do sistema sem que haja necessidade permanente de interagir com a unidade central de controlo.

Os sinais analógicos adquiridos actualmente podem ser de quatro origens diferentes:

- Potenciómetros internos aos servomotores, indicativos da sua posição;
- Extensómetros presentes na base de cada pé, para medição da força aplicada, para posterior cálculo do Centro de Pressão exercido;
- Acelerómetros/inclinómetros, que pela medição da aceleração da gravidade em duas componentes ortogonais, medem a inclinação do objecto onde se encontram localizadas;
- Giroscópios para medição da velocidade angular, para compensação das forças dinâmicas exercidas sobre o robot.

Estes valores são convertidos e registados localmente em cada unidade *slave* sendo depois enviados via CAN para o controlador *Master*. Os *Slaves* estão preparados também para receber mensagens via CAN. Estas mensagens consistem basicamente nos dados de actuação a aplicar sobre cada unidade local:

- Posições finais que os actuadores têm de tomar;
- Velocidade a que têm que se mover até atingir a posição final;
- Tipo de controlador em acção sobre as três juntas;
- Parâmetros de compensação para os algoritmos de compensação;
- Flags booleanas de controlo (ex.: PWM activados).

O controlador *master* tem a tarefa de receber a informação enviada pelos *slaves* via CAN e registá-la para que esteja disponível para ser enviada para a unidade central de controlo quando solicitada. Este controlador mantém, por isso uma representação do estado actual das juntas (actuadores e sensores) que disponibilizará ao controlo central sempre que este o pedir. O processo é bidireccional e o controlador *master* também recebe as ordens da unidade central e despacha para o *slave* respectivo.

Tabela 1: Características do Hardware e Software na unidade principal.

Unidade Central de Controlo	**Computador com porta série RS-232** • Software de suporte: MatLab 7.xx • Device Drivers de comunicação série: *mini-toolbox* cport v1.3
Unidades de controlo local *Master/Slave*	**PIC18F258 da Microchip** • Memória de programa: 2 MB • Memória de dados: 4 KB • Velocidade de processamento: 10 MIPS (f_{osc}=40MHz com a PLL activa) • Instruções de 16 bits e *datapath* de 8 bits • Definição de dupla prioridade nas interrupções. • Diversos periféricos: *timers*, módulos CCP, interfaces para redes de comunicação, ADC, etc…

1.1. Protocolos de Comunicação

Desenvolveram-se dois protocolos de comunicação, nomeadamente para…

- a linha série RS-232 entre o PC e a unidade *Master*
- e para o CAN entre a unidade Master e os *Slaves*,

…de modo a poder trocar dados sensoriais e de actuação entre o PC e as unidades *slave*.

Entre os dados sensoriais podem-se enumerar (para um SCU):

- **Posição** dos três servomotores (em graus);
- **Velocidade** estimada de cada servomotor (em graus/s);
- **Corrente** consumida por cada servomotor;
- Valores dos **sensores de força** de cada pé (quatro sensores por pé);
- Saída dos **giroscópios** (em graus/s);
- Saída dos **inclinómetros** (em graus).

Quanto aos dados de actuação:

- **Posição referência** a atingir para cada componente (ângulos para os três servomotores ou as 3 componentes *x*, *y*, *z* do sistema de coordenadas cartesiana);
- **Velocidade** média para a realização da trajectória até à posição final;
- **Tipo de controlador** activo (centro de pressão, inclinação, ou velocidade angular);
- **Parâmetros de compensação** para o controlador activo (K) e para o controlo local (K_P, K_I);
- **Flags booleanas** de des/activação (ex.: PWMs).

Considera-se que o controlo de actuação deve ser feito de forma isolada a cada uma das três juntas, e por isso as diversas juntas podem realizar movimentos com diferentes velocidades e diferentes parâmetros de compensação. Tal é útil no último caso, pois cada junta pode estar sujeita a diferentes esforços, e por isso, é rentável aplicar diferentes ganhos a cada uma.

Um dado importante a ter em conta, é a multiplicidade de significados que cada dado de actuação pode possuir, dependendo do tipo de controlador activo. Assim, para cada tipo de controlador, a posição referência diz respeito ao valor final que o respectivo controlador deve atingir. Quatro tipos de controladores de primeiro nível são implementados:

- **Sem controlador**: a posição referência diz respeito às três posições angulares a aplicar a cada junta, sendo directamente aplicadas sobre o controlador local (ver abaixo);
- **Controlo de Centro de Pressão**: a posição referência diz respeito ao centro de pressão que a estrutura deve possuir no fim do movimento.
- **Controlo de Inclinação**: a posição referência especifica a inclinação que a estrutura deverá possuir.
- **Controlo de Velocidade Angular**: (análogo aos restantes).

De igual modo, a velocidade indicada diz respeito ao controlador em causa, implementado-se, desta forma, trajectórias polinomiais que tanto podem ser de posição angular, centro de pressão, inclinação ou velocidade angular.

Com base nestas posições referência o controlador de primeiro nível determina as posições angulares a aplicar a cada servomotor, que posteriormente será entregue a um controlador local do tipo PI (Proporcional+Integrador) que se responsabilizará por garantir o alcance das posições solicitadas. O ajuste do controlador de primeiro nível é feito através do ganho K, enquanto que para o controlador local o ajuste é feito através dos ganhos K_I e K_P.

No que diz respeito aos dados sensoriais enviados à unidade Master, eles são de natureza constante e independentes do controlador activo.

2. COMUNICAÇÃO RS-232

2.1. Device Drivers de Comunicação na Unidade Principal

A. Device Drivers de Baixo-Nível

Para automatizar o processo de leitura/escrita dos dados sensoriais/actuadores, *device drivers* foram escritos na forma de funções em MatLab. Elas são:

Tabela 2: Lista de *device drivers* da unidade principal.

Ficheiro	*Função*	*Descrição*
initcom.m	`[H,error,errorstr]=initcom(gate,rate)`	Criação de uma nova ligação.
killcom.m	`[error,errorstr]=killcom(H)`	Término da ligação.
testcom.m	`[error,errorstr]=testcom(H)`	Pedido de envio de uma sequência de teste.
readcanstat.m	`[array,...]=readcanstat(H)`	Consulta do estado do barramento CAN.
readjoint.m	`[servos,...]=readjoint(H,scu_id,param)`	Leitura das posições das juntas de um SCU.
readspecial.m	`[special,...]=readspecial(H,scu_id)`	Leitura dos sensores especiais.
applyjoint.m	`[...]=applyjoint(H,scu_id,param,servos)`	Actuação nas juntas de um determinado SCU.
applycontrol.m	`[...]=applycontrol(H,scu_id,param,servos)`	Actualização dos parâmetros PID de uma determinado SCU.

initcom

Estabelecimento de uma nova ligação via RS-232.

```
[handler,error,errorstr]=initcom(gate,rate)

Entradas:
 gate     -> Porta a utilizar (1,2,...)
 rate     -> baudrate a definir (bits/s)
Saídas:
 handler  -> ID da linha de comunicações
 error    -> Código de erro
 errorstr -> String descritiva do erro
```

killcom

Término de uma ligação RS-232 existente.

```
[error,errorstr]=killcom(handler)

Entradas:
 handler -> ID da linha série
Saídas:
 error    -> Código de erro
 errorstr -> String descritiva do erro
```

testcom

Pedido de envio de uma sequencia de teste.

```
[error,errorstr]=testcom(H)

Entradas:
 handler  -> ID da linha série.
Saídas:
 error    -> Código de erro
 errorstr -> String descritiva do erro
```

readcanstat

Leitura do estado do barramento CAN.

```
[array,state,rx,error,errorstr]=readcanstat(H)

Entradas:
 H => Handler para comunicar com o Master
Saídas:
 array    => [estado de erro, #erros de transmissão, #erros de recepção]
 state    => Bits de estado dos servos
 rx       => Mensagem de baixo nível recebida
 error    => Código de erro, se existente
 errorstr => String descritiva do erro
```

readjoint

Leitura de um parâmetro sensorial de um SCU.

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

Entradas:
H        => Handler para comunicar com o Master
scu_id   => Identificador do SCU alvo
param    => Parametro a ler (Tabela 3)

Saídas:
sensors  => Parametros sensoriais relativos a cada servomotor.
state    => Bits de estado dos servos (Tabela 4)
rx       => Mensagem de baixo nível recebida
error    => Código de erro, se existente
errorstr => String descritiva do erro
```

Tabela 3: Valores possíveis do parâmetro param na função readjoint.

Campo param		*Descrição*	*Campo servos*	*Unidades*
PARAM_POSITION	0	Posição angular de cada junta.	[pos_1, pos_2, pos_3]	Graus
PARAM_VELOCITY	1	Velocidade estimada de cada junta.	[vel_1, vel_2, vel_3]	Graus/100ms
PARAM_CURRENT	2	Corrente drenada por cada servo.	[$curr_1$, $curr_2$, $curr_3$]	% do T de PWM

Valor 1	*2*	*3*	*4*	*5*	*6*	*7*	*Valor 8*
PWM	Calib	Deadline	FinAll	FinOne	FinServo3	FinServo2	FinServo1

Conteúdo do vector Status retornado pela função readjoint (ver Tabela 16)

Tabela 4: Valores presentes no vector *status* retornado pela função *readjoint*.

Elemento	***Campo***	***Descrição***
1	*PWM*	Activo se todos os motores possuem o PWM ligado.
2	*Calib*	Calibração dinâmica da posição dos servos activada.
3	*Deadline*	Ocorrência de um erro de violação da largura de banda disponível.
4	*FinAll*	Todos as juntas terminaram a trajectória.
5	*FinOne*	Pelo menos uma das juntas terminou a trajectória.
6	*FinServo3*	A junta 3 terminou a trajectória.
7	*FinServo2*	A junta 2 terminou a trajectória.
8	*FinServo1*	A junta 1 terminou a trajectória.

readspecial

Leitura dos sensores especiais de um SCU.

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

Entradas:
H      => Handler das comunicações com o Master
scu_id => SCU alvo

Saídas:
special  => Valores dos sensores especiais (4 valores)
rx       => Mensagem de baixo nível recebida
error    => Código de erro, se existente
errorstr => String descritiva do erro
```

applyjoint

Aplicação de uma ordem de actuação a cada componente de um SCU.

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

Entradas:
   H        => Handler para comunicar com o Master
   scu_id   => Identificador do SCU alvo
   param    => Parametro a aplicar (Tabela 5)
   servos   => Dados a aplicar
Saídas:
   rx       => Mensagem de baixo nível recebida
   error    => Código de erro, se existente
   errorstr => String descritiva do erro
```

Tabela 5: Valores possíveis do parâmetro param na função applyjoint.

Campo param		***Descrição***	***Campo servos***	***Unidades***
PARAM_POSITION	0	Posição referência a ser atingida.	[ref_1, ref_2, ref_3]	Graus *ou* outro
PARAM_VELOCITY	1	Duração do movimento a efectuar.	[vel_1, vel_2, vel_3]	Ciclos de 20ms
PARAM_SPECIAL	3	Activação/desactivação do PWM aplicado aos motores.	[0/1, 0, 0]	(Valor booleano)

applycontrol

Selecção do controlador de primeiro nível, e ajuste dos seus parâmetros bem como dos do controlador local.

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

Entradas:
   H        => Handler para comunicar com o Master
   scu_id   => Identificador do SCU alvo
   param    => Parametro a modificar (Tabela 6)
   servos   => Dados a aplicar
Saídas:
   rx       => Mensagem de baixo nível recebida
   error    => Código de erro, se existente
   errorstr => String descritiva do erro
```

Tabela 6: Valores possíveis do parâmetro param na função applycontrol.

Campo *param*		*Descrição*	Campo *servos*
PARAM_KI	0	Ganho da componente integral (*Ki*) do controlador local.	[Ki_1, Ki_2, Ki_3]
PARAM_KP	1	Ganho da componente proporcional (*Kp*) do controlo local.	[Kp_1, Kp_2, Kp_3]
PARAM_K	2	Ganho (*K*) do controlador de primeiro nível.	[K_1, K_2, K_3]
PARAM_CONTROLON	3	Tipo de controlo de primeiro nível (*Type*) a aplicar em cada junta (Tabela 7).	[$Type_1$, $Type_2$, $Type_3$]

Tabela 7: Tipos de controladores de primeiro nível (a aplicar com o parâmetro PARAM_CONTROLON na função applycontrol).

Tipo de Controlo (*Type*)		*Descrição*
NO_CONTROL	0	Sem Controlo de primeiro nível
COP_CONTROL	1	Controlo de Centro de Pressão
INC_CONTROL	2	Controlo de Inclinação
GIRO_CONTROL	3	Controlo de velocidade angular

B. Outros *Device Drivers*

Outras funções estão também disponíveis, não sendo nada mais que operações de nível intermédio que fazem uso dos *device drivers* apresentados na secção anterior (secção II. 2.1. A).

Tabela 8: Lista de device drivers da unidade principal.

Ficheiro	*Função*	*Descrição*
inituc.m	`[error,errorstr]=inituc(H,scu_list)`	Activação do PWM para vários SCUs.
killuc.m	`[error,errorstr]=killuc(H,scu_list)`	Desactivação do PWM para vários SCUs.
sensor_reset.m	`[stat,error,errorstr]=sensor_reset(H,scu_list)`	Recalibração dos sensores de um SCU.
statmotionfin.m	`[finall,finone,fin,..]=statmotionfin(H,scu_list)`	Consulta do estado da execução de trajectórias por parte de um SCU.
motionstart.m	`[stat,error,errorstr]=motionstart(H,scu_list)`	Espera que um SCU inicie a execução do movimento.
motionfin.m	`[stat,error,errorstr]=motionfin(H,scu_list)`	Espera que um SCU termine a execução do movimento.
waitmotionfin.m	`[stat,error,errorstr]=waitmotionfin(H,scu_list)`	Espero que o processo de iniciação e término do movimento seja executado.
waitpwmon.m	`[stat,error,errorstr]=waitpwmon(H,scu_list)`	Espera que o PWM seja activado nas três juntas de um SCU.

Ao contrário das funções de interface básicas (secção II. 2.1. A), estas permitem a utilização de listas de endereços de SCUs para aplicação do procedimento. Assim, sempre que precisar de executar uma operação para vários SCUs introduza um vector com a referência a todos eles no parâmetro *scu_list*.

inituc

Activação dos PWMs de uma série de SCUs.

```
[error,errorstr]=inituc(H,scu_list)

Entradas:
   H        => Handler para comunicar com o Master
   scu_id   => Conjunto de SCUs alvo (endereços)
Saídas:
   error    => Código de erro, se existente
   errorstr => String descritiva do erro
```

killuc

Desactivação dos PWMs de uma série de SCUs.

```
[error,errorstr]=killuc(H,scu_list)

Entradas:
   H        => Handler para comunicar com o Master
   scu_id   => Conjunto de SCUs alvo (endereços)
Saídas:
   error    => Código de erro, se existente
   errorstr => String descritiva do erro
```

sensor_reset

Recalibração sensorial de um conjunto de SCUs.

```
[stat,error,errorstr]=sensor_reset(H,scu_list)

Entradas:
   H        => Handler para comunicar com o Master
   scu_id   => Conjunto de SCUs alvo (endereços)
Saídas:
   stat     => Indicador de sucesso na operação
   error    => Código de erro, se existente
   errorstr => String descritiva do erro
```

statmotionfin

Leitura do estado de um conjunto de SCUs relativo à execução de trajectórias.

```
[finall,finone,fin,pwm,err,errstr]=statmotionfin(H,scu_list)

Entradas:
   H        => Handler para comunicar com o Master
   scu_id   => Conjunto de SCUs alvo (endereços)
Saídas:
   finall   => Todos os SCUs indicados terminaram uma trajectória
   finone   => Pelo menos um SCU dos indicados terminaram uma trajectória
   fin      => Vector indicando se a junta i (de 3) de todos os SCUs terminaram
   pwm      => Todos os SCUs indicados possuem os seus sinais de PWM activos
   error    => Código de erro, se existente
   errorstr => String descritiva do erro
```

motionstart

Espera que todos os SCUs indicados na lista comecem a execução de uma trajectória. É necessário que previamente tenha sido enviado um comando de actuação para início do movimento (*applyjoint*).

```
[stat,error,errorstr]=motionstart(H,scu_list)

Entradas:
   H        => Handler para comunicar com o Master
   scu_id   => Conjunto de SCUs alvo (endereços)
Saídas:
   stat     => Indicador de sucesso na operação
   error    => Código de erro, se existente
   errorstr => String descritiva do erro
```

Efeitos colaterais: caso a posição das juntas indicado no comando de actuação corresponda à actual, este comando ficará bloqueado. Como solução pode-se adicionar um *timer* para impor um *timeout*.

motionfin

Espera que todos os SCUs indicados na lista terminem a execução de uma trajectória.

```
[stat,error,errorstr]=motionfin(H,scu_list)

Entradas:
   H        => Handler para comunicar com o Master
   scu_id   => Conjunto de SCUs alvo (endereços)
Saídas:
   stat     => Indicador de sucesso na operação
   error    => Código de erro, se existente
   errorstr => String descritiva do erro
```

waitmotionfin

Espera que todos os SCUs indicados na lista comecem e terminem a execução de uma trajectória. Esta operação é a conjunção das operações *motionstart* e *motionfin*. É necessário que previamente tenha sido enviado um comando de actuação para início de um movimento (*applyjoint*).

```
[stat,error,errorstr]=waitmotionfin(H,scu_list)

Entradas:
   H        => Handler para comunicar com o Master
   scu_id   => Conjunto de SCUs alvo (endereços)
Saídas:
   stat     => Indicador de sucesso na operação
   error    => Código de erro, se existente
   errorstr => String descritiva do erro
```

Efeitos colaterais: caso a posição das juntas indicado no comando de actuação corresponda à actual, este comando ficará bloqueado. Como solução pode-se adicionar um *timer* para impor um *timeout*.

waitpwmon

Espera que todos os SCUs indicados na lista possuam os seus sinais de PWM (actuação sobre os motores) activados. É necessário que anteriormente tenha sido enviado um comando de activação de PWMs (ex.: *inituc* ou *applyjoint*).

```
[stat,error,errorstr]=waitpwmon(H,scu_list)

Entradas:
   H        => Handler para comunicar com o Master
   scu_id   => Conjunto de SCUs alvo (endereços)
Saídas:
   stat     => Indicador de sucesso na operação
   error    => Código de erro, se existente
   errorstr => String descritiva do erro
```

C. Utilização dos *Device Drivers*

Estabelecimento/Término de uma Linha de Comunicações

Para criar uma linha de comunicações série, é utilizado o comando *initcom*. Note que um *handler* nulo significa que não foi possível abrir o canal – apenas *handlers* não nulos são válidos.

```
[handler,error,errorstr]=initcom(gate,rate)
```

Como por exemplo, para abrir um canal utilizando a porta COM1 a um ritmo de transmissão de 115k2 bps, o seguinte comando deve ser enviado:

```
[H,error,errorstr]=initcom(1,115200)
```

Para verificar se as comunicações estão a decorrer sem problemas nenhuns, pode-se utilizar o comando *testcom*. Se o valor 0 for retornado, tudo está ok.

```
[error,errorstr]=testcom(H)
```

Para fechar o mesmo canal, deve-se, em contraposição, utilizar o comando *killcom*:

```
[error,errorstr]=killcom(H)
```

Activação/Desactivação dos Sinais de PWM

Para controlar a presença (ou não) dos sinais de PWM sobre os servomotores, é utilizado o comando de actuação *applyjoint* com o valor 3 no parâmetro *param* (PARAM_SPECIAL), tal como descrito na Tabela 5. Por sua vez o parâmetro *servos* será composto por um vector linha de três elementos, cujo primeiro elemento corresponderá a um valor booleano indicando se o *slave* de endereço *scu_id* deverá ter os seus sinais de PWM activos ou não.

```
[rx,error,errorstr]=applyjoint(H,scu_id,param,servos)
[rx,error,errorstr]=applyjoint(H,scu_id,3,[0/1 0 0])
```

A título de exemplo, para activar os PWMs do slave de endereço 1, seria enviado o seguinte comando:

```
[rx,error,errorstr]=applyjoint(H,1,3,[1 0 0])
```

Para automatização do processo de activação e desactivação dos PWMs a uma série de SCUs podem ser utilizadas as funções *inituc* e *killuc*. Estas funções permitem a atribuição de vectores de SCUs que automaticamente se encarrega de enviar o adequado *applyjoint* a cada um deles.

```
[error,errorstr]=inituc(H,[scu1 scu2 scu3 …])
[error,errorstr]=killuc(H,[scu1 scu2 scu3 …])
```

Leituras Sensoriais

Para ler os diversos sensores, os comandos *readjoint* e *readspecial* são utilizados, conforme se pretenda ler os sensores associados aos servomotores ou os sensores ligados via *piggy-back* sobre a placa controladora respectivamente (Tabela 9).

Tabela 9: Funções para leitura sensorial.

Tipo de sensores	***Função associada***
Sensores associados aos servomotores	`[sensors,state,rx,error,errorstr]=readjoint(H,scu_id,param)`
Sensores via *piggy-back*	`[special,rx,error,errorstr]=readspecial(H,scu_id)`

Com a função *readjoint* (parâmetros sensoriais dos servomotores) é possível ler a posição angular, a velocidade estimada e a corrente drenada que cada servo possui, através do parâmetro *param* (ver Tabela 3). Um vector de três elementos (*sensors*) é retornado com o resultado para cada um deles, tal como um vector de 8 elementos descritivos do estado do SCU consultado (a estrutura e o significado dos seus valores pode ser consultado na Tabela 4).

Tabela 10: Syntax da função readjoint na leitura dos diversos parâmetros sensoriais.

Parâmetro a obter	***Syntax da função***
Posição angular	`[sensors,state,...]=readjoint(H,scu_id,0)`
Velocidade angular estimada	`[sensors,state,...]=readjoint(H,scu_id,1)`
Corrente consumida	`[sensors,state,...]=readjoint(H,scu_id,2)`

Dado que o vector status retorna informação sobre o cumprimento de trajectória por parte do master, algumas funções de médio nível foram desenvolvidas para automatizar o processo de espera na iniciação/finalização de um percurso. A Tabela 8 refere-os como sendo *statmotionfin*, *motionstart*, *motionfin*, *waitmotionfin* e *waitpwmon*. Estas funções de médio nível permitem, tal como com o *inituc* e *killuc* a utilização de vectores de SCUs para simplificação do procedimento.

Através da função *readspecial*, é possível obter os ditos dados sensoriais "especiais", ou seja, as saídas provenientes dos sensores conectados via *piggy-back* à unidade *slave*. Eles são:

- 4 extensómetros localizados em cada pé, que medem a força exercida (sensores de força);
- Inclinómetros que medem a inclinação do robot nas componentes *x* e *y*;
- Giroscópios para medição da velocidade angular sobre as três componentes *x*, *y* e *z*.

Qualquer deste conjunto de sensores pode ser ligado às entradas analógicas (no máximo de quatro) fornecidas na interface *piggy-back*, sendo devolvidas à unidade central as suas quatro saídas digitalizadas em torno do valor 128 utilizando a gama de 0 a 255. O vector *special* devolvido por esta função contém estes quatro valores:

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

Exemplos: leitura sensorial da posição angular e dos sensores "especiais" do SCU de endereço 1:

```
[sensors,state]=readjoint(H,1,0)

      sensors=[1 0 -69]
      state=[1 0 0 1 1 1 1 1]

sensors=readspecial(H,1)

      sensors=[128 129 126 130]
```

Caso, por algum motivo, os valores retornados sejam inconsistentes, pode sempre ser feita a recalibração dos sensores. Para tal é executado o seguinte procedimento:

1. Desligar os PWMs
   `[rx,error,errorstr]=applyjoint(H,scuid,3,[0 0 0])` ou
   `[error,errorstr]=killuc(H,scu_list)`
2. Reactivar os PWMs
   `[rx,error,errorstr]=applyjoint(H,scuid,3,[1 0 0])` ou
   `[error,errorstr]=inituc(H,scu_list)`

Para maior automatização pode ainda ser utilizada a função reset_sensores, que, basicamente, executa os passos enunciados:

```
[stat,error,errorstr]=sensor_reset(H,scu_list)
```

Controlador de Primeiro Nível

Cada unidade *slave*, possui dois controladores em cascata:

1. Controlador de primeiro nível, responsável por realizar trajectórias de vários parâmetros de actuação diferentes, como por exemplo, posição angular ou centro de pressão;
2. Controlador local, responsável pela garantia da aplicação da posição angular solicitada.

Quatro tipos de controladores de primeiro nível estão implementados, tal como se pode constatar através da Tabela 11:

Tabela 11: Controladores de primeiro nível (consultar Tabela 7).

Tipo de controlador	***Parâmetro de actuação***	***Sensores utilizados***
Sem controlador	Posição angular do servo	Potenciómetro do servomotor
Controlo do Centro de Pressão	Centro de Pressão	4 Extensómetros
Controlo de Inclinação	Inclinação	Acelerómetros/Inclinómetros
Controlo de Velocidade Angular	Velocidade Angular	Giroscópios

Se não se seleccionar nenhum controlador, as trajectórias efectuadas terão como base apenas posições angulares referência relativas aos próprios servomotores. Caso contrário, um dos conjuntos de sensores "especiais" serão utilizados, com o controlador a executar a tarefa de calcular a posição angular a atribuir aos motores, de modo a alcançar o valor referência.

A função *applycontrol* é destinada à escolha deste controlador, com o seu parâmetro *param* igual 3 (ver Tabela 6), e o parâmetro *servos* com a selecção do controlador para cada junta (vector de três elementos) (ver Tabela 7):

```
[rx,error,errorstr]=applycontrol(H,scu_id,3,servos)
```

Como por exemplo, para seleccionar o controlador de centro de pressão para as três juntas do SCU 1, é necessário enviar o seguinte comando:

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

Além da selecção do controlador, é necessário também configurar o seu ganho para ajustar a sua reactividade a variações do sinal de erro na sua entrada. Este ganho é controlável também através da função *applycontrol*, mas com o parâmetro *param* igual a 2 (Tabela 6). O parâmetro *servos* é, assim, usado para definição do ganho para cada junta. Note que se definir diferentes controladores para as várias juntas, cada ganho será aplicado ao seu correspondente controlador.

```
[rx,error,errorstr]=applycontrol(H,scu_id,2,servos)
```

Para a definição de um ganho de 30 para as três juntas, deve ser executado o seguinte:

```
applycontrol(H,scu_id,2,[30 30 30])
```

Controlador Local

O controlador local é responsável por garantir a aplicação da posição angular solicitada pelo controlador de primeiro nível, com base na posição medida através do potenciómetro interno ao servomotor. Caso o controlador de primeiro nível esteja desactivado, a posição de actuação enviada pela unidade central é aplicada directamente neste controlador.

Esta componente apenas se trata de um compensador do tipo PI que a partir da posição referência e da medida pelo potenciómetro (sinal de erro) calcula a compensação a atribuir ao servo para que o sinal de erro se anule. Para ajuste deste compensador são usados os ganhos K_I e K_P para as componentes integradora e proporcional respectivamente. Para a definição destes ganhos é usada igualmente a função *applycontrol* com o parâmetro *param* igual a 0 para o ajuste de K_I e 1 para o K_P (Tabela 6). O parâmetro *servos* possuirá o valor numérico do ganho a atribuir a cada junta (vector triplo).

Tabela 12: Syntax da função *applycontrol* na definição dos ganhos K_I e K_P do controlador local.

Ganho a controlar	***Syntax da função***
K_I	`[rx,error,errorstr]=applycontrol(H,scu_id,0,servos)`
K_P	`[rx,error,errorstr]=applycontrol(H,scu_id,1,servos)`

Para a definição de um K_I igual a 5, e um K_P igual a 20 às três juntas do SCU 1, usar-se-á a sintaxe:

```
applycontrol(H,1,0,[ 5  5  5])
applycontrol(H,1,1,[20 20 20])
```

Caso se atribua K_I e K_P nulos, o controlador local será desactivado, e as posições angulares retornadas pelo controlador de primeiro nível são aplicadas directamente sobre os servomotores:

```
[rx,error,errorstr]=applycontrol(H,scu_id,0,[0 0 0])
[rx,error,errorstr]=applycontrol(H,scu_id,1,[0 0 0])
```

Execução de Trajectórias

Para a definição dos setpoints para uma trajectória, a função de actuação *applyjoint* é utilizada, com o parâmetro *param* a 0 para a atribuição da posição referência, e a 1 para a velocidade.

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

Tabela 13: Syntax da função *applyjoint* na definição posição referência e da velocidade.

Parâmetro a controlar	***Syntax da função***
Referência	`[rx,error,errorstr]=applyjoint(H,scu_id,0,servos)`
Velocidade	`[rx,error,errorstr]=applyjoint(H,scu_id,1,servos)`

A velocidade é indicada através da duração da trajectória em ciclos de 20ms (período de PWM), pelo que se pretender uma duração de 2s, o valor 100 deve ser dado. No que respeita à posição referência, este parâmetro depende do controlador de primeiro nível em acção. A Tabela 11 refere a relação entre o parâmetro de actuação dado nesta função e o controlador activo. Por isso, antes da definição deste valor é necessário seleccionar previamente o controlador através da função *applycontrol*, e só depois modificar este parâmetro usando a função *applyjoint*. O mesmo também se aplica à velocidade, podendo definir-se valores diferentes de posição e velocidade para cada controlador – a mudança do controlador carrega automaticamente os valores utilizados na sua utilização anterior.

Para a definição de uma trajectória de inclinação à primeira junta do *slave* 1, até +40º segundo uma duração de 2s, deve-se efectuar o seguinte conjunto de passos:

1. Seleccionar o controlador de inclinação:
   `[rx,error,errorstr]=applycontrol(H,1,0,[0 0 0])`

2. Verificar se o ganho do controlador é nulo:
   `[rx,error,errorstr]=applycontrol(H,1,2,[0 0 0])`

3. Definir a inclinação de referência final:
   `[rx,error,errorstr]=applyjoint(H,1,0,[+40 0 0])`

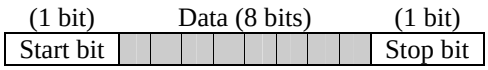
4. Definir a duração da trajectória:
   `[rx,error,errorstr]=applyjoint(H,1,1,[100 0 0])`

5. Iniciar o movimento ao atribuir um ganho não nulo ao controlador de inclinação:
   `[rx,error,errorstr]=applycontrol(H,1,2,[30 0 0])`

Note que antes de tudo, é feita a selecção do controlador de interesse, pois desta forma, a definição do seu ganho e as instruções de actuação serão agulhadas para o parâmetro de controlo associado. Uma estratégia usada neste exemplo, para garantir que a trajectória não é iniciada durante a definição da inclinação final e da velocidade, é a inicialização prévia do ganho do controlador a zero para forçá-lo ao "estacionamento". Só após a atribuição de um ganho válido, ele retoma o seu funcionamento normal.

2.2. Protocolo RS-232

A comunicação RS-232 entre o PC e a unidade *Master* é efectuada assincronamente e é orientada ao byte (1 *start* bit + 8 bits de dados + 1 *stop* bit), ou seja, é transmitido um byte de dados em cada transmissão/recepção. Pretende-se que numa única mensagem esteja contida toda a informação relativa a um parâmetro a ler/actuar nas três juntas de um SCU, o que implica que cada mensagem seja constituída por vários bytes. Como por exemplo, para ordenar o SCU *X* a colocação das juntas nas posições *A*, *B* e *C*, são necessários no mínimo quatro bytes: três para as três posições *A*, *B* e *C*; e um indicando a identificação do SCU.



Estrutura de um pacote USART (transmissão/recepção de um byte)

Comandos PC→Master:

As mensagens no sentido PC→*master* são constituídas por seis bytes. O primeiro byte sinalizará a mensagem como sendo um comando de solicitação à unidade *master* – MESSAGE_REQ (ver Tabela 18); o segundo byte conterá um código (*opcode*) indicativo da operação a realizar, do SCU alvo e de parâmetros adicionais; os três bytes seguintes conterão parâmetros a atribuir às três juntas no caso de um comando de actuação, e finalmente o byte BCC indicará a validade da mensagem (como não tendo sofrido corrupção).

byte 1	*byte 2*	*byte 3*	*byte 4*	*byte 5*	*byte 6*
SOF: *0x*FF	OpCode	Joint 1	Joint 2	Joint 3	BCC

Mensagem USART PC→Master de actuação.

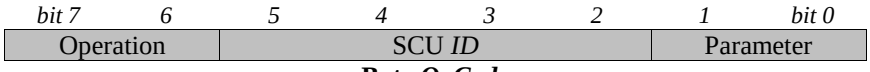
byte 1	*byte 2*	*byte 3*	*byte 4*	*byte 5*	*byte 6*
SOF: *0x*FF	OpCode	0	0	0	BCC

Mensagem USART PC→Master de leitura sensorial.

Tabela 14: Campos das mensagens PC→Master via USART.

Pacotes	Função:	Valores possíveis:
SOF	(*S*tart *O*f *F*rame) Indica o tipo da mensagem	(Ver Tabela 18).
OpCode	Código indicando o que é solicitado e a quem se destina.	(Ver Tabela 15)
Joint 1/2/3	Dados de actuação.	• Mensagem de actuação: Dados de actuação a atribuir a cada componente de um determinado SCU. • Mensagem de leitura sensorial: campos nulos.
BCC	*Block Check Code* Verificação da integridade da mensagem.	$BCC = \left(\sum_{i=1}^{5} byte_i\right) \%256$

A Tabela 15 descreve a estrutura do byte *OpCode*, bem como todos os seus valores possíveis. Os três bytes de dados (*Joint* 1/2/3) dizem respeito à operação indicada neste byte.



Byte *OpCode*

Tabela 15: Campos do pacote OpCode nas mensagens PC→Master via USART.

SCU id	*Operation*	*Parameter*
SCU alvo relativo à operação (endereçável a 15 SCU's: *0x*0001-*0x*1111)	*OP_APPLY_JOINT* (*0b*00) Mensagem de actuação sobre as três componentes do SCU alvo (SCU id): componentes dependem do controlador activo.	*PARAM_POSITION* (*0b*00) Posição referência a ser atingida.
		PARAM_VELOCITY (*0b*01) Velocidade média do movimento a efectuar.
		PARAM_SPECIAL (*0b*11) (Des)Activação do PWM aplicado aos motores.
	OP_APPLY_CONTROL (*0b*01) Mensagem de actuação sobre as três juntas do SCU alvo com a definição dos parâmetros dos controladores.	*PARAM_KI* (*0b*00) Ganho da componente integral do controlador local.
		PARAM_KP (*0b*01) Ganho da componente proporcional do controlo local.
		PARAM_K (*0b*10) Ganho do controlador de primeiro nível.
		PARAM_CONTROLON (*0b*11) Tipo de controlo (de primeiro nível) a aplicar na junta.
	OP_READ_SENSORS (*0b*10) Mensagem de leitura dos sensores.	*PARAM_POSITION* (*0b*00) Posição actual de cada junta.
		PARAM_VELOCITY (*0b*01) Velocidade estimada de cada junta.
		PARAM_CURRENT (*0b*10) Corrente drenada por cada servo.
		PARAM_SPECIAL (*0b*11) Saída dos sensores especiais.
	OP_READ_EXTBUFF (*0b*11) Leitura do buffer externo do Master.	*Status* do barramento CAN.

De notar que a posição referência e a velocidade média da trajectória (*OP_APPLY_JOINT*) depende do controlador de primeiro nível activo (seleccionado através do parâmetro *PARAM_CONTROLON*). Os vários tipos de controlador estão indicados na Tabela 16.

Tabela 16: Tipo de controlo a seleccionar no campo PARAM_CONTROLON.

Tipo de Controlo sobre as juntas	*Designação*	*PARAM_CONTROLON*
Sem Controlo de primeiro nível	*NO_CONTROL*	*0b*00
Controlo de Centro de Pressão	*COP_CONTROL*	*0b*01
Controlo de Inclinação	*INC_CONTROL*	*0b*10
Controlo de velocidade angular	*GIRO_CONTROL*	*0b*11

De notar, que a actuação é feita directamente às três juntas/componentes numa única mensagem, mas com a desvantagem de apenas se poder actuar num parâmetro cada vez (posição final, velocidade média ou um parâmetro do controlador).

Respostas (aos comandos) Master→PC:

Na resposta, o *Master* responde com uma mensagem de 7 bytes, cuja estrutura é a seguinte:

- *SOF*: possui o valor *MESSAGE_SUCESS* (*0x*FB) no caso de uma resposta com sucesso;
- *OpCode*: *opcode* utilizado pela mensagem original PC→Master;
- Data 1-4: dados sensoriais no caso de um pedido de consulta sensorial;
- BCC: Validação da mensagem.

byte 1	*byte 2*	*byte 3*	*byte 4*	*byte 5*	*byte 6*	*byte 7*
*0x*FD	OpCode	Data 1	Data 2	Data 3	Data 4	BCC

Formato geral de uma mensagem de resposta Master→PC.

Esta estrutura é geral e pode assumir várias formas de acordo com a operação envolvida:

- ➔ No caso de uma operação de actuação (*OP_APPLY_**), os bytes 2-5 possuem o mesmo valor que a mensagem original com o sexto byte nulo;
- ➔ Numa leitura sensorial (*OP_READ_SENSORS*), o byte *OpCode* é igual à da mensagem original com os bytes *Data* 1-4 contendo a informação sensorial pedida. Se os dados sensoriais concernem aos servomotores, três bytes são utilizados para conter a informação relativa a cada um deles, e o último é utilizado para transmitir informação de *status* do SCU em causa. Se concernem aos sensores especiais, são utilizados todos os quatro bytes para conter a informação de um dos conjuntos dos sensores – sensores de força (de um pé), inclinómetros ou giroscópios.
- ➔ Se for solicitada a leitura do buffer externo do master (*OP_READ_EXTBUFF*), o estado do barramento CAN (percepcionado pelo *master*) é devolvido.

O formato das mensagens de resposta descritos apresentam-se de seguida (== significa que o byte é o mesmo do da mensagem de solicitação):

byte 1	*byte 2*	*byte 3*	*byte 4*	*byte 5*	*byte 6*	*byte 7*
SOF: *0x*FD	==	==	==	==	0	BCC

Mensagem de actuação aplicada com sucesso

byte 1	*byte 2*	*byte 3*	*byte 4*	*byte 5*	*byte 6*	*byte 7*
SOF: *0x*FD	==	Joint 1	Joint 2	Joint 3	Status*	BCC

Mensagem de leitura sensorial das juntas(servomotores)

byte 1	*byte 2*	*byte 3*	*byte 4*	*byte 5*	*byte 6*	*byte 7*
SOF: *0x*FD	==	Data 1	Data 2	Data 3	Data 4	BCC

Mensagem de leitura dos sensores especiais

byte 1	*byte 2*	*byte 3*	*byte 4*	*byte 5*	*byte 6*	*byte 7*
SOF: *0x*FD	==	Error Code	# TX errors	# RX errors	0	BCC

Mensagem de leitura do buffer externo do Master (status do barramento CAN)

bit 7	*6*	*5*	*4*	*3*	*2*	*1*	*bit 0*
PWM	—	Deadline	FinAll	FinOne	FinServo3	FinServo2	FinServo1

Conteúdo do Byte de Status(*) *nas mensagens de leitura sensorial das juntas*

Tabela 17: Significado dos bits presentes no byte de status nas mensagens de leitura sensorial das juntas.

Bit	***Campo***	***Descrição***
7	*PWM*	Activo se todos os motores possuem o PWM ligado.
6	*——*	
5	*Deadline*	Ocorrência de um erro de violação da largura de banda disponível.
4	*FinAll*	Todos as juntas terminaram a trajectória.
3	*FinOne*	Pelo menos uma das juntas terminou a trajectória.
2	*FinServo3*	A junta 3 terminou a trajectória.
1	*FinServo2*	A junta 2 terminou a trajectória.
0	*FinServo1*	A junta 1 terminou a trajectória.

Repare que, na leitura sensorial das juntas, na mesma mensagem é transportado os valores dos três servos. No entanto, apenas um parâmetro pode ser lido – posição, velocidade média ou corrente.
Quanto à leitura dos sensores especiais, note também, a limitação de quatro valores, daí a limitação imposta anteriormente de apenas usar quatro linhas analógicas dedicadas a este género de sensores: ou o conjunto dos sensores de força, ou inclinómetros, ou giroscópios.
No entanto, podem ocorrer situações anómalas na recepção do comando por parte do *Master*, podendo-se discernir duas situações possíveis:

- ➔ Mensagem de pedido inválido: os parâmetros solicitados pelo PC não fazem sentido (ex.: SCU alvo não existente). Neste caso uma mensagem de SOF de código *MESSAGE_INVREQ* (*0x*FC) (*invalid request*), com todos os restantes bytes iguais à da mensagem original, é retornada ao PC.
- ➔ Mensagem corrompida: o código BCC não está de acordo com a estrutura da mensagem. Neste caso uma mensagem de SOF igual a *MESSAGE_INVALID* (0xFB) é retornado com os restantes bytes iguais à da mensagem recebida.

byte 1	*byte 2*	*byte 3*	*byte 4*	*byte 5*	*byte 6*	*byte 7*
SOF: *0x*FC	==	==	==	==	0	BCC

Mensagem sinalizadora de um pedido inválido.

byte 1	*byte 2*	*byte 3*	*byte 4*	*byte 5*	*byte 6*	*byte 7*
SOF: *0x*FB	==	==	==	==	0	BCC

Mensagem sinalizadora de inconsistência no byte BCC.

Teste das comunicações USART entre PC e *Master*:

Para confirmação da correcta comunicação entre o PC e o *Master* pode ser enviada uma mensagem de teste igualmente de 6 bytes com o SOF como MESSAGE_TEST (0xFE).

byte 1	*byte 2*	*byte 3*	*byte 4*	*byte 5*	*byte 6*
SOF: *0x*FE	X2	X3	X4	X5	BCC

Mensagem USART PC→Master de teste.

Os bytes 2 a 5 podem possuir qualquer valor, desde que o byte BCC esteja de acordo com eles. Por sua vez, o *master* deverá responder com uma mensagem de 7 bytes possuindo os mesmos valores que a primeira:

byte 1	*byte 2*	*byte 3*	*byte 4*	*byte 5*	*byte 6*	*byte 7*
SOF: *0x*FE	X2	X3	X4	X5	0	BCC

Resposta Master→PC a uma mensagem de teste.

A Tabela 18 apresenta sintetizados os diversos tipos de mensagens apresentados nas secções anteriores, com a descrição do byte SOF utilizado para cada caso.

Tabela 18: Tipos de mensagens USART (primeiro byte de cada frame).

Tipo de Mensagem	***Designação***	***Código***	***Descrição***
Mensagem de solicitação	*MESSAGE_REQ*	*0x*FF	Solicitação de dados do PC para o *Master*.
Mensagem de teste	*MESSAGE_TEST*	*0x*FE	Envio de uma mensagem de teste das comunicações.
Mensagem de sucesso	*MESSAGE_SUCESS*	*0x*FD	Resposta a uma mensagem de solicitação (MESSAGE_REQ), indicando que o comando foi executado com sucesso.
Parâmetros inválidos	*MESSAGE_INVREQ*	*0x*FC	Pedido com parâmetros inválidos.
Mensagem inválida	*MESSAGE_INVALID*	*0x*FB	Mensagem de estrutura inválida (BCC incorrecto).

2.3. Unidade Principal

A. Configuração da Unidade Principal

Como configurações gerais optou-se por efectuar as transacções à velocidade máxima permitida sem a aplicação de controlo de comunicações. Elas são:

- ➔ *Baudrate*: 115200 bps
- ➔ Tamanho da palavra de dados: 8 bits
- ➔ Número de *stop bits*: 1
- ➔ Bit de paridade: desactivado
- ➔ Controlo de comunicações por *handshaking*: desactivado

Utilizou-se um computador Pentium com porta série RS-232 embutida, com o sistema operativo Microsoft® Windows XP, como unidade principal, para comunicar com a unidade *master*, embora seja perfeitamente compatível usar qualquer sistema desde que possua suporte a comunicações RS-232. O software utilizado foi o MatLab 7.0 através da mini-toolbox *cport* v1.3 que oferece *device-drivers* para troca directa por RS-232 tanto de caracteres como de *strings* e valores numéricos inteiros.

Para inicializar as comunicações através do *cport* é necessário seguir as seguintes instruções:

1. Abrir as comunicações especificando a porta série a usar (COM?) guardando o *handler* da linha aberta.

   Porta utilizada: COM1 `handler=cportopen('com1')`
   (Se handler=0, a ligação falhou!)

2. Configurar a linha especificando as seguintes características:

   Tabela 19: Configurações gerais do cport.

Campo	*Valor*
Baudrate	115200 bps
Tamanho do byte	8 bits
Descarte dos bytes nulos	desactivado
Controlo de fluxo pela linha DTR	desactivado
Controlo de fluxo pela linha RTS	desactivado
Término das operações de leitura/escrita na ocorrência de um erro	activado
Caracter a usar no caso de um erro de paridade	nenhum
Tempo de timeout entre dois caracteres consecutivos	2 caracteres (ms)
Tempo de timeout para a recepção/transmissão de uma mensagem	10 ms

```
state=cportconfig(handler,'BaudRate',115200,                    ...
                          'ByteSize',8,                         ...
                          'fNull','OFF',                        ...
                          'fDtrControl','OFF',                  ...
                          'fRtsControl','OFF',                  ...
                          'fAbortOnError','ON',                 ...
                          'ErrorChar',-1,                       ...
                          'ReadIntervalTimeout',2*11/baudrate*1000,  ...
                          'ReadTotalTimeoutMultiplier',1,       ...
                          'ReadTotalTimeoutConstant',10,        ...
                          'WriteTotalTimeoutMultiplier',1,      ...
                          'WriteTotalTimeoutConstant',10);
```

Para efeitos de *debugging* é útil usar um terminal RS-232. Recomenda-se a utilização do terminal *R.E.Smith* por permitir a troca de bytes de qualquer código *ASCII* (caracter ou não) e a visualização das saídas/entradas

em formato hexadecimal.
Em caso de uso de um terminal as seguintes opções são suficientes:

Tabela 20: Configurações de um terminal RS-232 (No caso do R.E.Smith, usar COM1, 115200, N-8-1).

Campo	***Valor***
Porta	COM1
Baudrate	115200
Bits de dados	8
Paridade	Nenhum
Bits de paragem	1
Controlo de fluxo	Nenhum

No caso de haver algum problema durante o funcionamento é necessário reinicializar as comunicações. Para isso basta fazer...

```
stat=cportreset(handler)
```

Para terminar as comunicações é só usar o comando *cportclose*:

```
stat=cportclose(handler)
```

Caso haja algum problema na configuração do *cport*, pode obter as definições correctas através do seguinte procedimento:

1. Fechar todos os programas e reiniciar o computador;
2. Ligar as comunicações através do terminal *R.E.Smith* seguindo as opções da Tabela 20;
3. Verificar a conectividade (através de uma mensagem de teste) e de seguida desligar as comunicações;
4. Abrir o MatLab e ligar a porta série através dos comandos do *cport* definindo apenas o *baudrate*;
5. Anotar as configurações tomadas fazendo *stat=cportconfig*(*handler*).
6. Redefinir as configurações do *cport* com estes dados (actualizar Tabela 19).

Sempre que um programa tenha utilizado a porta série antes, as configurações a adoptar pelo *cport* serão as mesmas que desse programa. A partir daqui é só configurar a porta série com estes dados sempre que uma ligação é estabelecida.

Para simplificação, duas funções foram criadas para automatizar o processo de conexão e desconexão com as configurações citadas anteriormente (Tabela 2). Elas são as seguintes:

```
% Estabelecimento de uma nova ligação via RS-232
[handler,error,errorstr]=initcom(gate,baudrate)

% Término de uma ligação RS-232 existente.
[error,errorstr]=killcom(handler)
```

B. Algoritmos de Comunicação

As rotinas de comunicação básicas, já apresentadas na secção II. 2.1 (ver Tabela 2 e Tabela 8), constituem a forma de interface entre a unidade de controlo principal (normalmente um computador) e a unidade *master*. O seu algoritmo básico de funcionamento é apresentado na Fig. 7 e resume-se ao envio de uma mensagem à unidade *master* de acordo com os parâmetros de entrada fornecidos, retornando em seguida os dados sensoriais contidos na mensagem de resposta, caso seja o caso de um comando de leitura sensorial, mais o correspondente estado de erro (se ocorreu algum problema nas comunicações).

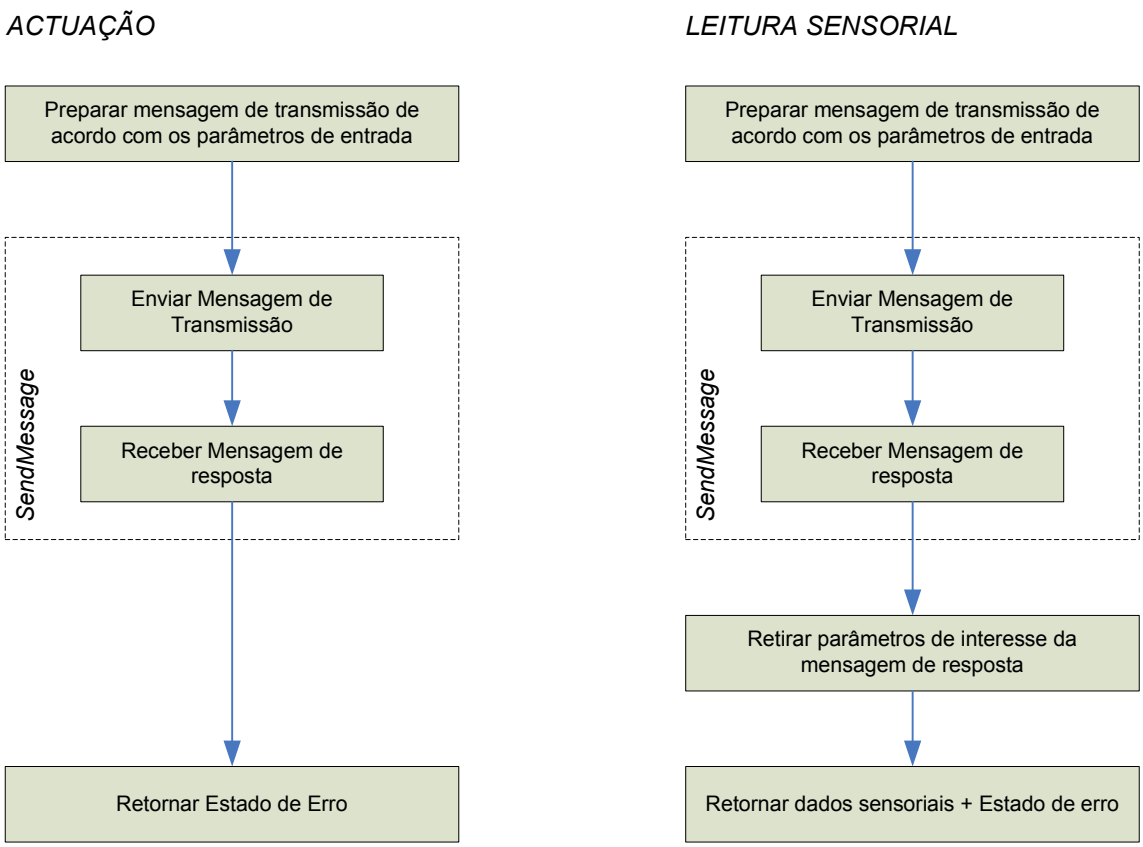


Fig. 7: Algoritmo básico dos device drivers de actuação / leitura sensorial.

Todos os *device drivers* apresentados utilizam uma função base que tem como responsabilidade a implementação do protocolo descrito na secção II. 2.2, ou seja, o envio de uma mensagem de 6 bytes à unidade *master* solicitando determinado dado sensorial ou com uma ordem de actuação, e depois o processamento da mensagem de resposta com a informação solicitada, ou confirmação da actuação, respectivamente. Esta função é denominada por *sendmessage*, e é constituída pelo conjunto de passos apresentado na Fig. 8 e Fig. 9

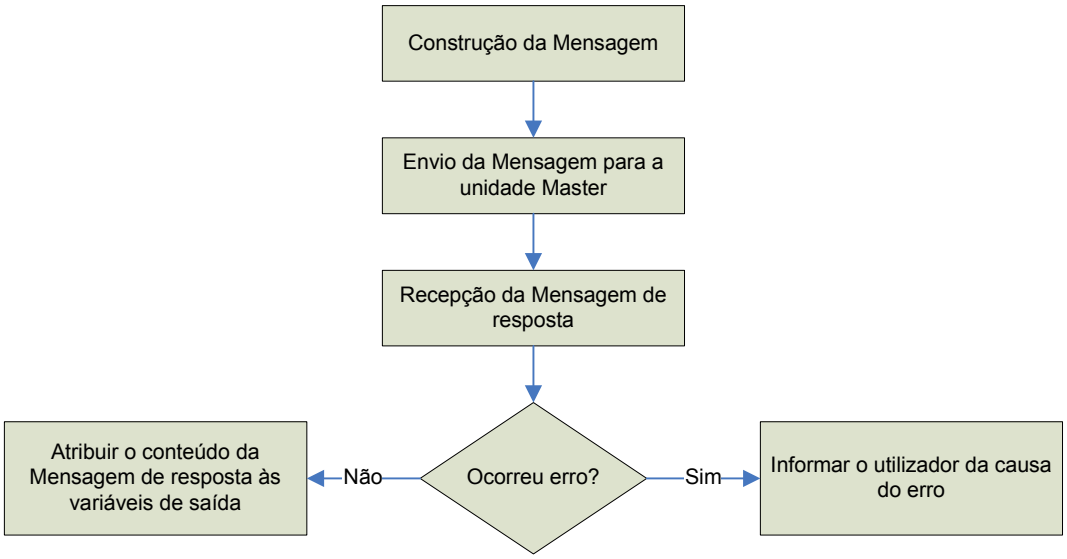


Fig. 8: Algoritmo básico da função *sendmessage*.

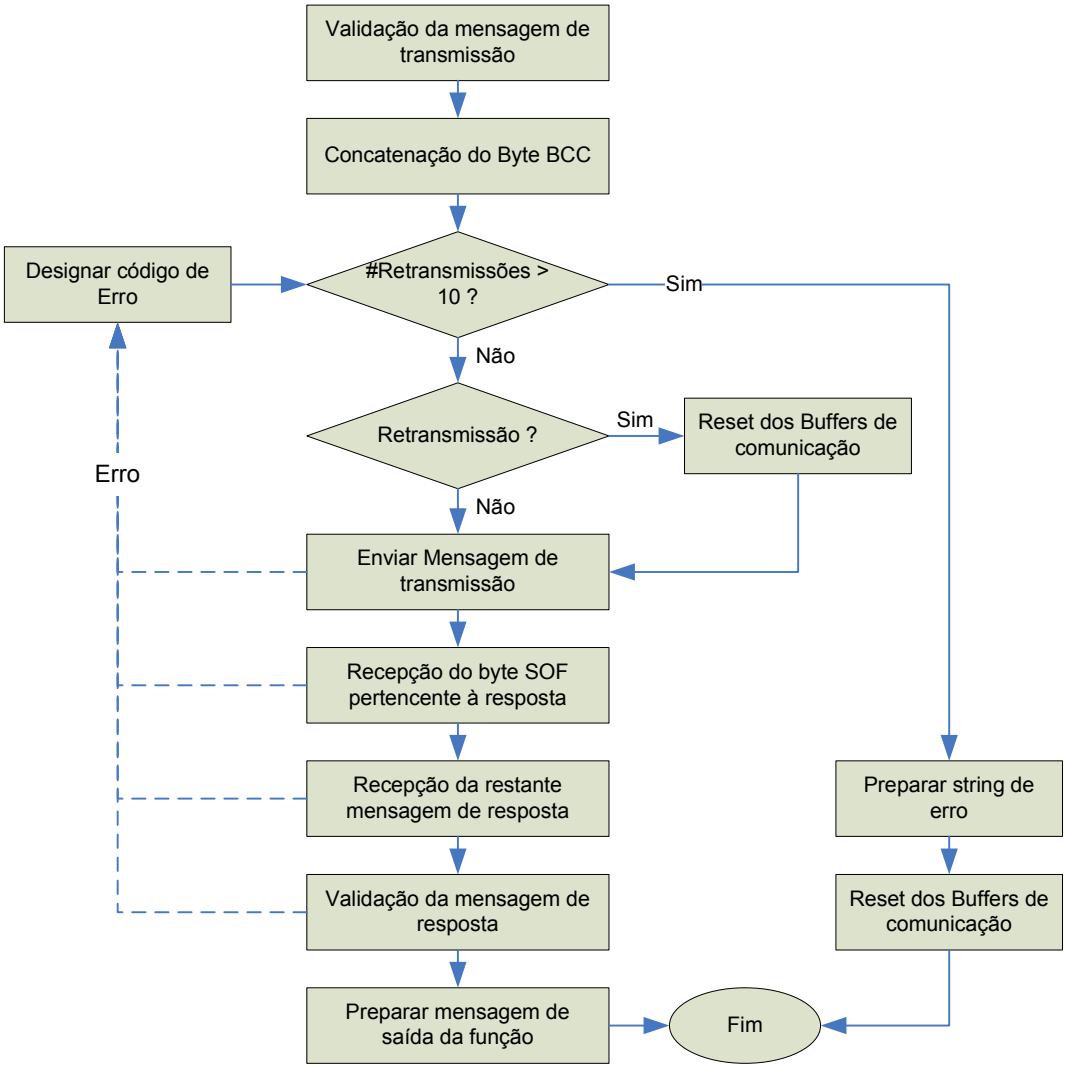


Fig. 9: Algoritmo detalhado da função *sendmessage*.

Um pormenor interessante que se pode extrair da Fig. 9 é a diferenciação dos passos de leitura do byte SOF e da restante mensagem de resposta. Observações experimentais indicam que por vezes a mensagem de resposta não se sincroniza com o envio do comando, começando a ocorrer deslocamentos (o primeiro byte recebido não corresponde ao primeiro byte da mensagem de resposta, mas sim a um byte intermédio da resposta interior. Para evitar este tipo de problemas, pesquisa-se durante algum tempo pelo byte SOF (de valor superior a *MESSAGE_INVALID – 0x*FB) e só depois de recebido, lê-se a restante mensagem.

2.4. Unidade Distribuidora *MASTER*

A. Configuração da unidade *Master*

Para configurar a unidade microcontroladora a operar com a linha RS-232 é necessário efectuar um conjunto de passos que serão descritos a seguir. Primeiramente é necessário configurar os pinos RX (recepção) e TX (transmissão) como entrada e saída respectivamente:

```
TRISC=(TRISC | 0x80) & 0xBF;
```

De seguida é necessário configurar o *baudrate* da comunicação através do registo SPBRG. Para a frequência de CPU F_{CPU}=10MHz e para um *baudrate* desejado de 115200 bps, segundo a fórmula (modo BRGH activo – *high speed*):

$$SPBRG = round\left(\frac{F_{CPU}}{4 * Baud}\right) - 1$$

... o registo SPBRG deve assumir o valor de 21.

Configurar o modo de funcionamento de transmissão através do registo TXSTA e de recepção pelo registo RCSTA.

R/W-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0	R-1	R/W-0
CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D
bit 7							bit 0

Registo TXSTA

Campo	***Parâmetro***	***Valor***	***Descrição***
CSRC	Clock Source Select bit	X	——
TX9	9-bit Transmit Enable bit	0	Selects 8-bit transmission
TXEN	Transmit Enable bit	1	Transmit enabled
SYNC	USART Mode Select bit	0	Asynchronous mode
BRGH	High Baud Rate Select bit	1	High speed
TRMT	Transmit Shift Register Status bit	—	(Read only)
TX9D	9th bit of Transmit Data	X	——

Tabela 1: Configuração do registo TXSTA.

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0	R-0	R-x
SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D
bit 7							bit 0

Registo RCSTA

Campo	***Parâmetro***	***Valor***	***Descrição***
SPEN	Serial Port Enable bit	1	Serial port enabled (configures RX/DT and TX/CK pins as serial port pins)
RX9	9-bit Receive Enable bit	0	Selects 8-bit reception
SREN	Single Receive Enable bit	X	——
CREN	Continuous Receive Enable bit	1	Enables continuous receive
ADDEN	Address Detect Enable bit	X	——
FERR	Framing Error bit	—	(Read only)
OERR	Overrun Error bit	—	(Read only)
RX9D	9th bit of Received Data	—	(Read only)

Tabela 2: Configuração do registo RCSTA.

Finalmente é necessário activar/desactivar as interrupções respectivas e defini-las como alta prioridade (não esquecer de activar a funcionalidade de dupla prioridade):

```
IPR1bits.RCIP=1;              // Interrupções de alta prioridade
IPR1bits.TXIP=1;
PIE1bits.RCIE=1;              // (Des)Activação das interrupções
PIE1bits.TXIE=0;
```

B. Algoritmos de Comunicação

As Fig. 12 e Fig. 13 apresentam os algoritmos para troca de informação entre o master e a unidade principal.

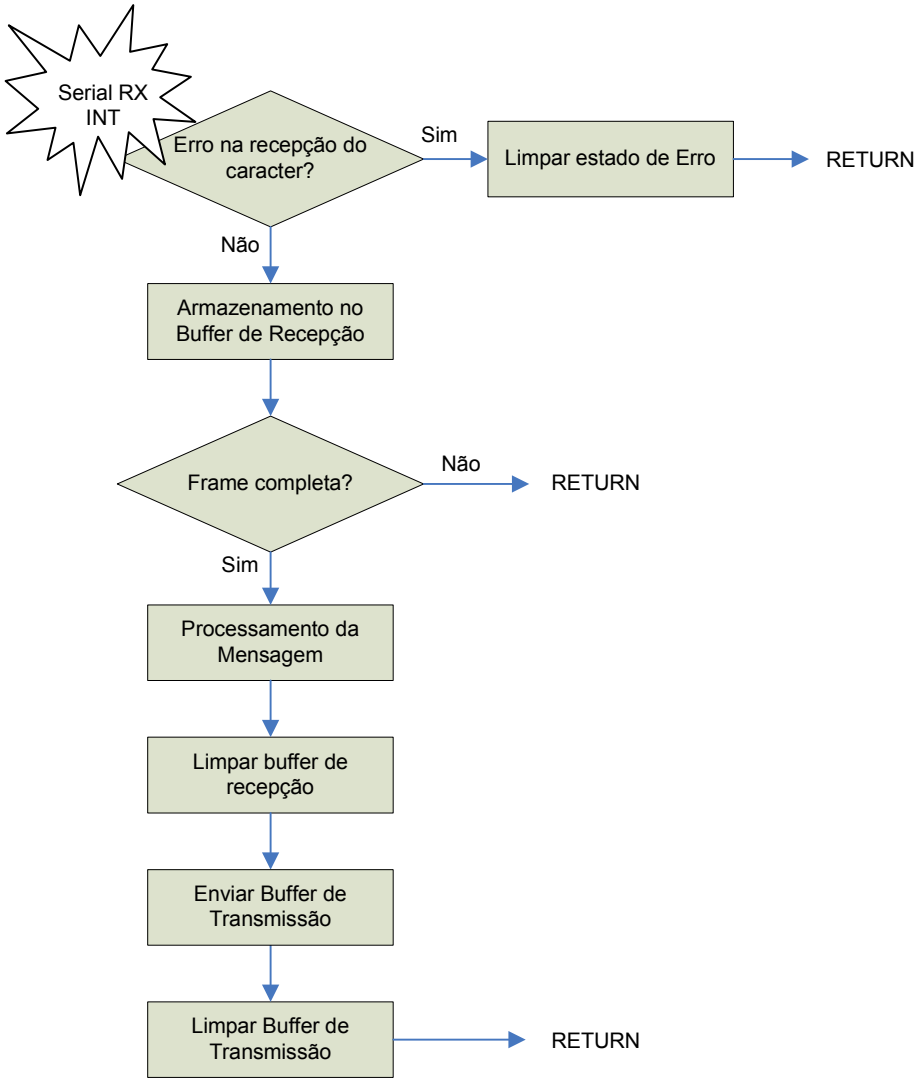


Fig. 10: Algoritmo de gestão das comunicações RS-232 na unidade *Master*.

Pela Fig. 10 podemos observar que o tratamento das mensagens é executado byte a byte a partir de interrupções de recepção pela linha série, com a utilização de *buffers* implementados em software para armazenar a mensagem recebida do PC (*buffer* de recepção) e a mensagem de resposta a enviar (*buffer* de transmissão). A implementação de cada um destes procedimentos está descrita na Fig. 11 e Fig. 12, correspondendo aos passos "armazenamento no *buffer* de recepção" e "enviar *buffer* de transmissão" apontados na Fig. 10.
Como detalhes importantes para a robustez deste algoritmo é a utilização de um timer para contagem de tempo, com vista a impor um tempo de *timeout* máximo para a conclusão de cada um dos procedimentos. Deste modo evitam-se bloqueios que poderiam comprometer o funcionamento do sistema.

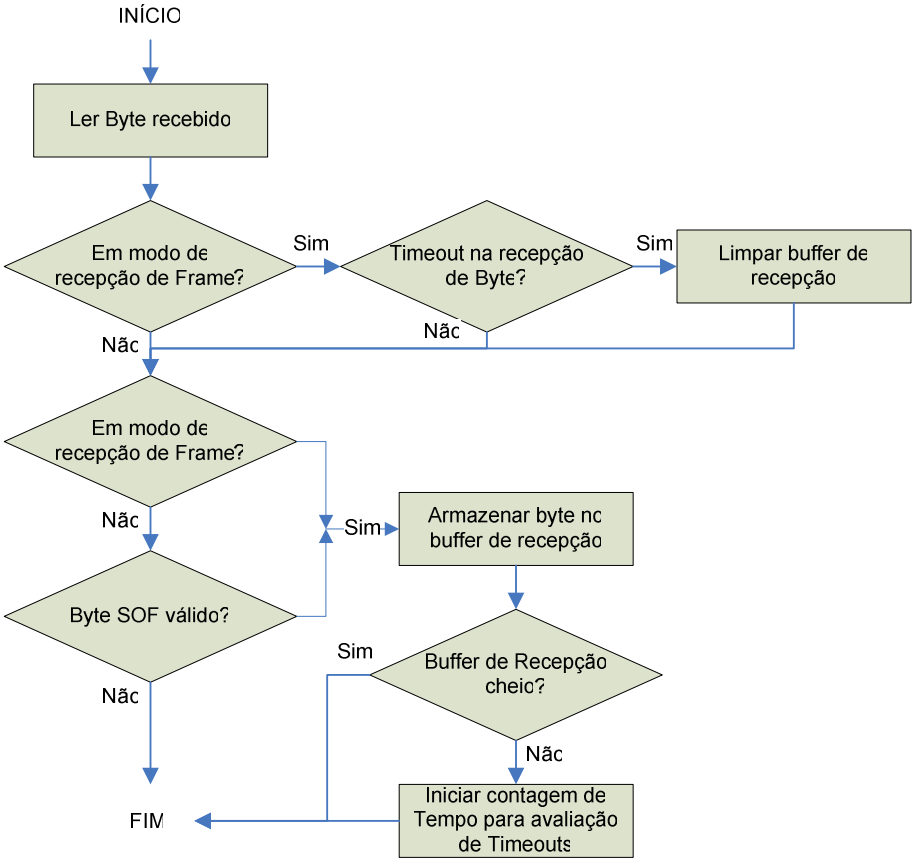


Fig. 11: Recepção de byte por RS-232 e armazenamento no buffer de recepção.

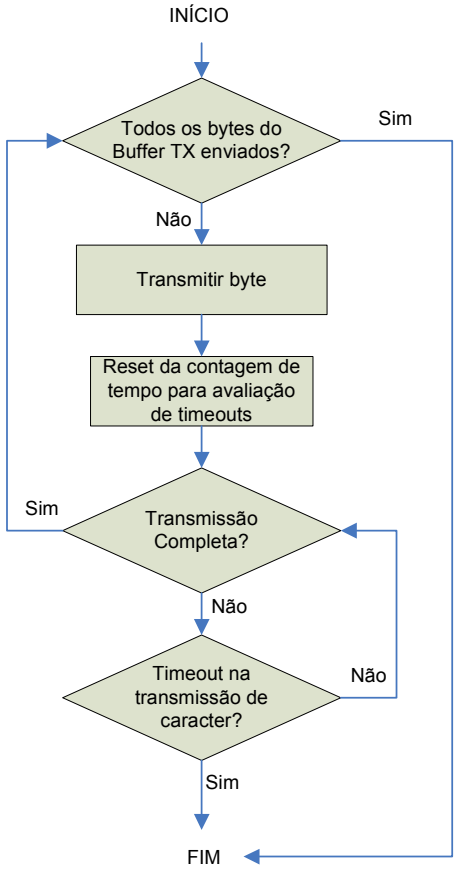


Fig. 12: Envio do buffer de transmissão por RS-232.

A função da tarefa “processamento da mensagem” apontada na Fig. 10 é a de ler a mensagem de solicitação presente no *buffer* de recepção, e construir a mensagem de resposta de acordo com o protocolo enunciado na secção II. 2.2, armazenando-a posteriormente no *buffer* de transmissão. O seu algoritmo é apresentado na Fig. 13 sendo capaz de validar a integridade da mensagem (por verificação do byte BCC), e construir as mensagens de resposta correspondentes a um comando de solicitação (actuação ou leitura sensorial) ou de teste das comunicações.

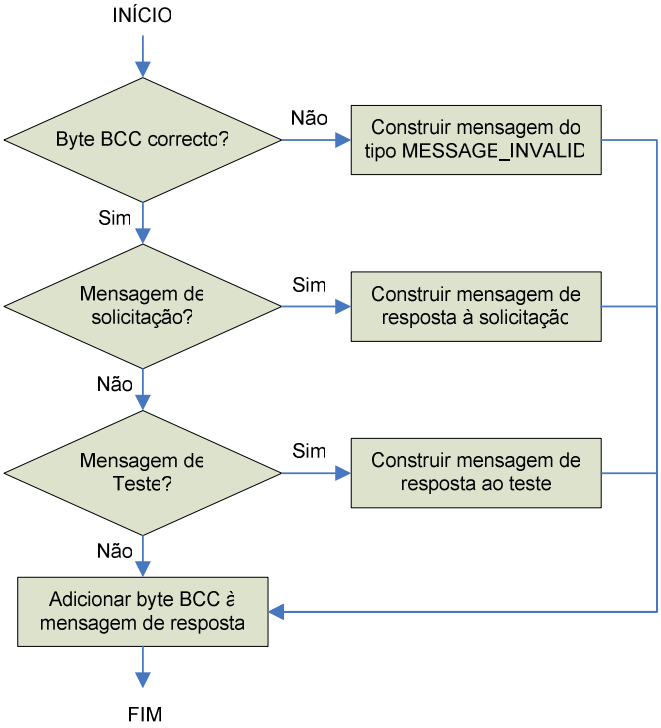


Fig. 13: Processamento de uma mensagem na unidade Master.

No que respeita ao processamento das mensagens de solicitação, a resposta vem de acordo com o byte *OpCode* descrito na Tabela 15 (Fig. 14).

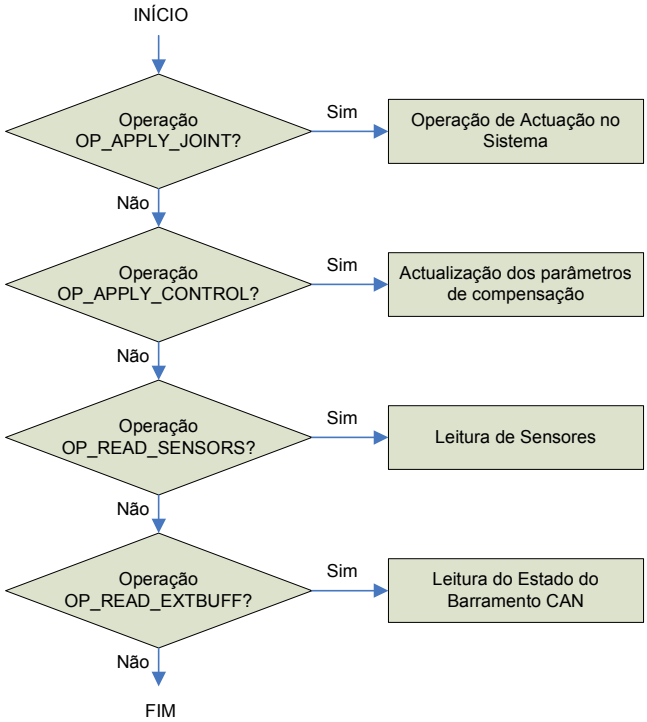


Fig. 14: Processamento de uma mensagem de solicitação na unidade Master.

Um melhoramento importante a conferir a estes algoritmos, para evitar tempos mortos na largura de banda de CPU, sugere-se uma solução baseada na transmissão por interrupção, que em vez da transmissão dedicada a todos os bytes da mensagem de resposta, o CPU só deve ser utilizado para a transmissão de cada byte através da notificação por interrupção do envio do último byte. Desta forma, o período de tempo entre a transmissão de cada par de bytes pode ser utilizado para a execução de outras tarefas, como é o caso do serviço às comunicações CAN.

C. Base de Dados Global do Sistema Humanóide

O processamento da mensagem de solicitação, proveniente da unidade principal, é baseado na análise do byte *opcode* segundo a descrição na Tabela 15. Uma vez decifrado o conteúdo da mensagem, uma base de dados global com a informação sensorial e de actuação de cada uma das unidades *slave*, é acedido para consulta ou actualização, dependendo se se trata de uma operação de consulta sensorial ou de actuação respectivamente. A base de dados é dividida em dois conjuntos de dados principais, uma para a informação sensorial e outra para a de actuação.

Esta base de dados, que pode ser visualizada na página seguinte, descreve o estado do sistema completo para todos os 8 slaves. A secção sensorial descreve o estado do sistema (Tabela 17), as posições, velocidades estimadas e correntes drenadas, fornecidas pelos potenciómetros dos três servomotores, bem como também a saída dos sensores que eventualmente possam estar conectados via *piggy-back* a cada unidade *slave*.
Já a base de dados de actuação é representada por duas estruturas, uma indicando o estado de aplicação dos sinais de PWM sobre os servomotores e os parâmetros de compensação do controlador local e de primeiro nível, e a outra com os valores de referência dos mesmos compensadores. Houve necessidade de duas estruturas, em vez de uma só, por questões de limitação de memória dos microprocessadores PIC, que não permitem que cada estrutura de dados faça a transposição de um banco de dados para o outro (o PIC18F258 possui a memória segmentada em 16 bancos de 256 bytes cada um). Como solução, definiram-se duas estruturas colocando cada uma num banco diferente, através do "pragma" *varlocacate* no ficheiro *header .h*, indicando aos módulos exteriores a localização dos dados, e *udata* no ficheiro *.c* para alocá-los efectivamente no banco desejado.

Como já foi referido, cada comando de solicitação apenas consulta ou actualiza esta base de dados, não representando uma intervenção directa sobre as várias unidades *slave*. Tal procedimento permite que a troca de mensagens série opere à máxima velocidade, dado que não é necessário comunicar com mais nenhuma unidade. Quanto à entrega efectiva dos dados entre o *master* e os *slaves*, é da responsabilidade da comunicação CAN que garante que os parâmetros de actuação são entregues às diversas unidades *slave* com um determinado ritmo periódico, bem como a actualização da base de dados com os dados sensoriais correctos para o instante.
Dado que o período de PWM aplicado aos servomotores é de 20ms, e que apenas tem interesse actualizar os parâmetros de actuação e sensoriais uma vez dentro deste período, para garantir alguma segurança na actualização dos mesmos, definiu-se um período de troca de informação *master-slave* de 1 ms, fazendo com que para 8 *slaves*, ao fim de 8 ms todas unidades locais possuam os seus parâmetros locais actualizados. Desta forma, garante-se que para cada período de PWM, a base de dados global no *master*, e a local em cada *slave* sejam actualizados duas vezes, de modo a incluir alguma redundância no caso de falha esporádica na entrega de mensagens.

Estrutura sensorial

```
// Estrutura descritiva dos sensores
typedef struct {
    byte sysStatus;                             // Estado do sistema
    struct_servo servo[N_SERVOS];               // Sensores dos servos
    unsigned char special[N_SPECIAL_SENSORS];   // Sensores de força dos pés
} struct_sensors;
#pragma varlocate 1 sensors
extern volatile struct_sensors sensors[N_SCU];  // Conjunto global dos sensores
```

Estrutura de actuação

```
// Estrutura descritiva dos actuadores
typedef struct {
    // Estrutura de actuação de status para cada SCU
    struct {
        bool pwm;                               // PWM on/off
        bool calib;                             // Calib on/off
    } sysStatus;
    // Estrutura de Controlo para cada junta
    struct {
        byte ki, kp, kd;
        enum_controlType type;
    } control[N_SERVOS];
} struct_actuators;
#pragma varlocate 2 sensors
extern volatile struct_actuators actuators[N_SCU];  // Conjunto global dos actuadores

// Estrutura descritiva dos valores referência dos controladores
typedef struct {
    struct_paramControl servo[N_SERVOS];
    struct_paramControl cop[N_SERVOS];
    struct_paramControl inc[N_SERVOS];
    struct_paramControl giro[N_SERVOS];
} struct_refControl;
#pragma varlocate 3 sensors
extern volatile struct_refControl refControl[N_SCU];
```

Estruturas adicionais

```
// Tipo enumerado do género de controlador implementado em cada junta
typedef enum {
    NO_CONTROL  = 0b00,                 // Funcionamento em malha aberta
    COP_CONTROL = 0b01,                 // Controlo do Centro de Pressão
    INC_CONTROL = 0b10,                 // Controlo de inclinação
    GIRO_CONTROL= 0b11                  // Controlo de giroscópios
} enum_controlType;

// Estrutura descritiva de um servo (parte sensorial)
typedef struct {
    signed char position;               // Posição
    signed char velocity;               // Velocidade
    unsigned char current;              // Corrente consumida
} struct_servo;

// Estrutura descritiva dos valores referência a aplicar a um SCU
typedef struct {
    signed char position;               // Posição
    signed char velocity;               // Velocidade
} struct_paramControl;
```

Declaração das Estruturas de Dados no ficheiro .c

```
// Variáveis globais descritivas do sistema
#pragma udata bank1=0x100
volatile struct_sensors   sensors[N_SCU];                  // Sensores
#pragma udata

#pragma udata bank2=0x200
volatile struct_actuators actuators[N_SCU];               // Actuadores
#pragma udata

#pragma udata bank3=0x300
volatile struct_refControl refControl[N_SCU];
#pragma udata
```

3. COMUNICAÇÃO CAN

3.1. Introdução

A comunicação CAN é implementada entre o *Master* e as diversas unidades *Slave* e tem como finalidade o redireccionamento dos dados de actuação, provenientes da unidade principal, para cada unidade *Slave*, e, no sentido oposto, o envio dos dados sensoriais para a unidade *Master*, de modo a que esta actualize a base de dados com a informação sensorial de todas as unidades de forma a permitir à unidade principal a sua consulta.

O CAN é um sistema de comunicações série multi-ponto que foi desenvolvido originalmente para a indústria automóvel para possibilitar as comunicações entre diversos componentes em ambientes extremamente ruidosos. Para tal, o sinal que serve de suporte à comunicação é definido em corrente e não em tensão. É um protocolo baseado na mensagem e não no endereço, o que significa que as mensagens são transmitidas na forma de *broadcasting* para todos os nós existentes na rede, cabendo a cada um a decisão de a aceitar ao não. As mensagens são constituídas por um identificador que pode possuir 11 ou 29 bits, de acordo com a versão 2.0A ou 2.0B respectivamente, e até 8 bytes de dados (Fig. 15).

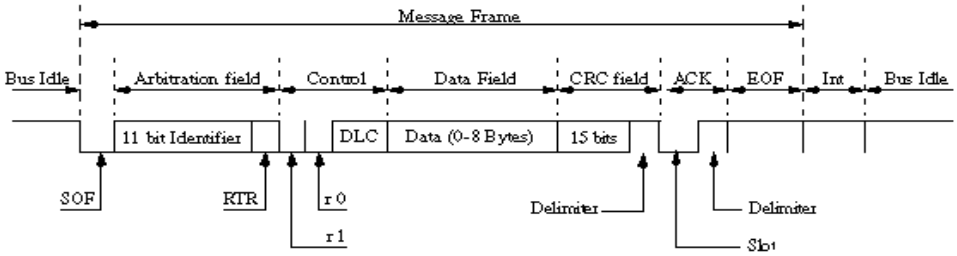


Fig. 15: Formato *standard* das mensagens CAN.

De modo a eliminar a hipótese de colisão destrutiva de pacotes (no caso de envio simultâneo de mensagens por diferentes nós) como acontece nas redes Ethernet, o CAN é dotado de um sistema de colisão determinístico bit a bit (CSMA/BA – Carrier Sense Multiple Access / Bit-wise Arbitration) que no caso de uma colisão de pacotes apenas um prevalece (o de maior prioridade), enquanto os restantes são destruídos. No último caso, os nós associados a esses pacotes devem tentar retransmitir mais tarde. Desta forma garante-se que a largura de banda oferecida pela rede não é desperdiçada, havendo sempre o transporte de mensagens desde que pelo menos um nó transmita.

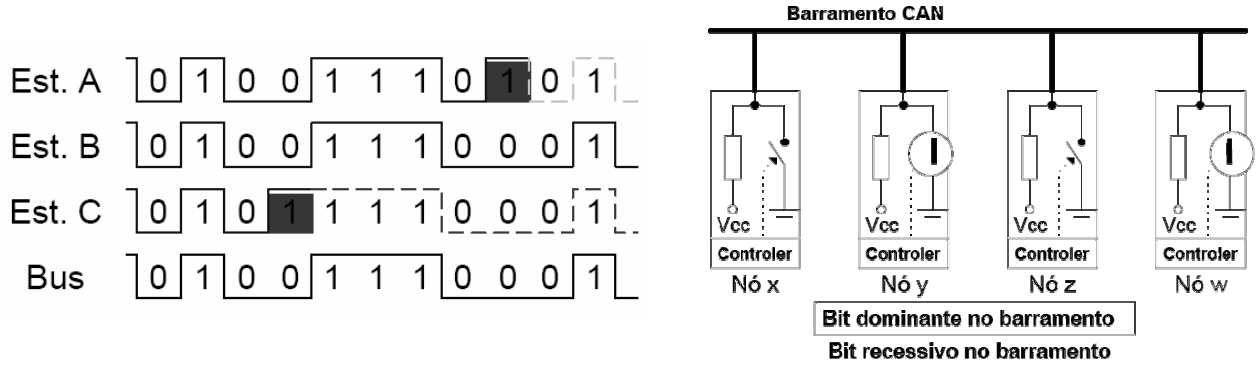


Fig. 16: Arbitragem de mensagens segundo o sistema de colisões CSMA/BA.

Para implementação deste sistema um dos bits é definido como dominante (bit 0) e o outro como recessivo (bit 1), de modo que quando dois bits colidem um com o outro "vence" o dominante. Esta estratégia é implementada bit a bit ao nível do identificador conferindo, deste modo, prioridades de acordo com o seu valor. Assim, quanto menor for o valor do identificador, maior é a prioridade da mensagem (Fig. 16). Segundo esta política é vantajoso atribuir um endereço de menor valor numérico à unidade *Master* de modo

a conferir a máxima prioridade às mensagens de actuação *Master→Slave*, e às restantes unidades é-lhes atribuído endereços de valor crescente à medida que a sua ordem de importância na rede diminui. Desta forma, atribuiu-se o endereço *0b*0000 ao *Master* e os endereços *0b*0001 até *0b*1000 para os oito *Slaves*, dando maior prioridade às unidades respeitantes aos membros inferiores dado que requerem maior atenção ao nível do controlo.

Unidade Controladora	***Secção a que respeita***	***Endereço na Rede***
Master	Unidade mestre	*0b*0000
Slave 1	Perna direita	*0b*0001
Slave 2	Perna esquerda	*0b*0010
Slave 3	Anca direita	*0b*0011
Slave 4	Anca esquerda	*0b*0100
Slave 5	Tronco	*0b*0101
Slave 6	Braço direito	*0b*0110
Slave 7	Braço esquerdo	*0b*0111
Slave 8	Cabeça	*0b*1000

Tabela 21: Endereços atribuídos às diversas unidades de controlo.

Outro benefício da comunicação baseada na mensagem é o facto de se poderem adicionar outros nós à rede sem haver necessidade de reprogramar todos os nós existentes. O novo nó adicionado receberá igualmente as mensagens que circulam na rede, decidindo por si só se deve ou não processar a informação contida. Esta foi uma das principais razões que motivaram a escolha do CAN, permitindo, desta forma que a rede possa ser modificada sem qualquer implicação ao nível do controlo local.

No microcontrolador em causa, é utilizada a versão *full*CAN 2.0A cujas características principais são:

- Mensagens com um identificador de 11 bits e um máximo de 8 bytes de dados;
- Filtragem completa do conteúdo do identificador (todos os bits do identificador podem ser utilizados para configuração da filtragem);
- Associação do identificador a *buffers* de dados;
- Múltiplos *buffers* de transmissão/recepção: 3 *buffers* de transmissão e 2 *buffers* de recepção;
- Processamento e recuperação automática de erros.

3.2. Protocolo CAN

De acordo com os dados a trocar entre o *Master* e cada *Slave*, e utilizando todos os 8 bytes de dados em cada mensagem, é possível efectuar trocas de informação, quer de actuação, quer sensoriais, recorrendo a apenas duas mensagens para cada instanciação (16 bytes). O protocolo é enunciado a seguir.

bit 10 9 8 7	6 5 4 3	2	1 *bit 0*
Source address	Destination address	Index	Operation

Identificador de um pacote CAN

Campo	***Descrição***
Source Address	Endereço do nó emissor da mensagem
Destination Address	Endereço do nó destinatário da mensagem
Index	Índice do pacote dentro de uma mensagem: ➔ 0: primeiro pacote; ➔ 1: segundo pacote.
Operation	Operação associada à mensagem: ➔ *0b*00: actualização sensorial; ➔ *0b*01: actualização de actuadores.

Tabela 22: Campos do identificador de um pacote CAN.

De notar que, do ponto de vista das unidades *Slave*, apenas o campo *destination address* é útil para a recepção de cada pacote, pois é ele quem define o destino do pacote. Já o campo *source address* está presente nos bits mais significativos com o propósito de definir a prioridade do pacote de acordo com endereço do remetente. Desta forma, mensagens enviadas pelo *master* possuem a máxima prioridade, e dentro dos *slaves*, os dos membros inferiores possuem maior prioridade de envio (ver Tabela 22). Como cada transacção é constituída por duas mensagens, o campo *index* identifica a sua ordem, e o campo *operation* refere a função dos dados transportados.

Transacção Master→Slave

As mensagens trocadas do *master* para *slaves*, têm como função a actualização dos actuadores das juntas a que dizem respeito, pelo que transportará as seguintes informações para cada um dos três servomotores associados:

- Posição referência a aplicar ao controlador de primeiro nível;
- Velocidade média do movimento;
- Flags de (des)activação dos controladores locais (ex: PWM);
- Parâmetros de compensação dos controladores local e de primeiro nível.

Source address				Destination address				Index	Operation	
0	0	0	0	X	X	X	X	X	0	1

Identificador de uma mensagem de actualização de actuadores

O identificador destas mensagens possuem o endereço *0b*0000 no *source address* indicativo de que o pacote é proveniente da unidade *master*, e o campo *operation* contém o código *0b*01 descrevendo que o pacote tem como intenção fazer uma actualização dos actuadores. O campo *destination address* identifica o SCU alvo.

byte 1	*byte 2*	*byte 3*	*byte 4*	*byte 5*	*byte 6*	*byte 7*	*byte 8*
Reference 1	Reference 2	Reference 3	Velocity 1	Velocity 2	Velocity 3	Control Flags	K_1 3

Dados da primeira mensagem de actualização de actuadores (*Index*=0)

byte 1	*byte 2*	*byte 3*	*byte 4*	*byte 5*	*byte 6*	*byte 7*	*byte 8*
K_I 1	K_I 2	K_I 3	K_P 1	K_P 2	K_P 3	K_1 1	K_1 2

Dados da segunda mensagem de actualização de actuadores (*Index*=1)

Cada transacção é constituída por duas mensagens em que tanto a posição referência final, como a velocidade média e os parâmetros de compensação para cada servomotor são actualizadas de uma só vez.
O byte *Control Flags* tem como função indicar qual o tipo de controlador de primeiro nível a estar activo em cada um dos servomotores (Tabela 7) e se os sinais de controlo dos motores (PWM) devem estar activos (valores booleanos).

bit 7	*6*	*5*	*4*	*3*	*2*	*1*	*bit 0*
PWM	—	Controller 3		Controller 2		Controller 1	

Byte *Control Flags* da primeira mensagem

Transacção Slave→Master

Por outro lado é necessário actualizar a base de dados na unidade *master* pelo envio de mensagens CAN com os dados sensoriais de cada *slave* para o *master*. Tal é executado logo após a recepção do pacote de actuação e com o retorno das seguintes informações:

- Posição angular dos servomotores;
- Velocidade estimada dos servos;
- Corrente consumida;
- Output dos sensores conectados via *piggy-back*;

Source address				Destination address				Index	Operation	
X	X	X	X	0	0	0	0	X	0	0

Identificador de uma mensagem de actualização sensorial

Os pacotes de actualização sensorial devem possuir no campo *destination address* o endereço da unidade *master* (*0b*0000), dado que a mensagem se destina a ele, e o campo *operation* a *0b*00 indicativo de que a mensagem contém dados sensoriais. O *source address* contém o endereço do *slave* respectivo.

byte 1	*byte 2*	*byte 3*	*byte 4*	*byte 5*	*byte 6*	*byte 7*	*byte 8*
Sensor Flags	Position 1	Position 2	Position 3	Velocity 1	Velocity 2	Velocity 3	Slave Status

Dados do primeiro pacote de actualização sensorial (*Index*=0)

byte 1	*byte 2*	*byte 3*	*byte 4*	*byte 5*	*byte 6*	*byte 7*	*byte 8*
Sensor Flags	Current 1	Current 2	Current 3	Special 1	Special 2	Special 3	Special 4

Dados do segundo pacote de actualização sensorial (*Index*=1)

Os valores de saída dos sensores piggy-back estão descritos nos campos *Special* apenas podendo fazer parte apenas de uma das seguintes classes:

- Sensores de força de um dos pés (4 sensores);
- Inclinómetros e/ou Giroscópios (2 + 2 valores);

No entanto é indiferente de onde provém estes dados, dado que a unidade principal (PC) conhece à priori o género de sensores especiais conectados a cada unidade *slave*.

O byte *sensor flags* é puramente redundante, dado que a informação que contém também é transportado no identificador do pacote, e poderá ser usado no futuro para conter mais dados sensoriais.

bit 7	*6*	*5*	*4*	*3*	*2*	*1*	*bit 0*
Source Address				Index	0	0	0

Byte *Sensor Flags*

Campo	***Descrição***
Source Address	Endereço do slave remetente.
Index	Índice do pacote na mensagem de dados.

Tabela 23: Campos do byte *Sensor Flags*.

Já o byte *slave status* contém *flags* indicadoras do estado de funcionamento da unidade *slave* (consultar Tabela 17).

bit 7	*6*	*5*	*4*	*3*	*2*	*1*	*bit 0*
PWM	—	Deadline	FinAll	FinOne	FinServo3	FinServo2	FinServo1

Byte *Slave Status* da primeira mensagem

3.3. Funcionamento do Barramento CAN

A. Configuração CAN

Inicializações

As definições de funcionamento do CAN devem ser iguais entre a unidade *Master* e as unidades *Slave* de modo a permitir o correcto sincronismo entre todas as entidades envolvidas na rede, pelo que as opções tomadas devem ser implementadas em todas as unidades.

Como condições base seguiram-se as seguintes linhas:

- ✓ Escolha da versão CAN mais universal: *full*CAN 2.0A;
- ✓ Configuração da velocidade na máxima possível (≤ 1 Mbps);
- ✓ Filtragem das mensagens apenas destinadas ao próprio SCU – apenas as mensagens com o endereço destino igual ao do próprio são aceites;
- ✓ Dadas as transacções serem constituídas por duas mensagens, aproveitou-se a presença de dois *buffers* de recepção para redireccionar cada uma para um *buffer* à parte – dupla filtragem;
- ✓ Igualmente, utilizou-se dois *buffers* de transmissão para transmitir cada uma das duas mensagens.

A seguir são descritos as principais operações na configuração deste recurso.

Modo de Configuração

O PIC 18F258 possui seis modos de operação, dos quais apenas 3 nos são úteis:

- ➢ Modo *disable*: o PIC não transmite nem recebe quaisquer mensagens;
- ➢ Modo de configuração: necessário antes de activar as transmissões/recepções;
- ➢ Modo de operação normal: recepção e transmissão de mensagens válidas, via CAN.

Antes de colocar o PIC a processar mensagens via CAN, é necessário antes, colocar o PIC em modo de configuração e definir os seus parâmetros de configuração. Apenas depois deste passo coloca-se o PIC no modo de operação normal.
Para definir o modo de funcionamento, os bits REQOP do registo CANCON devem ser modificados para o modo de interesse, e esperar a mudança efectiva por *polling* dos bits OPMODE do registo CANSTAT.

Baudrate

Todos os nós na rede CAN devem ter o mesmo *bitrate* nominal – número de bits transmitidos por segundo – sem no entanto ser necessário que todos os nós tenham a mesma frequência de oscilação. No entanto é necessário o cuidado em programar o *baudrate prescaler* e o número de *time quanta* (explicado adiante) em cada segmento de bit, de modo a manter o *bitrate* nominal. O *bitrate* máximo depende da qualidade do transmissor e do oscilador, e da presença de ressincronizações, mas poderá atingir o valor máximo de 1 Mbps numa situação ideal.

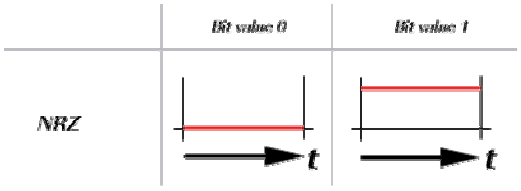


Fig. 17: Codificação NRZ.

Dado que a transmissão utiliza codificação NRZ (non return to zero) (Fig. 17) e que os osciladores e transmissores podem variar de nó para nó, é necessário introduzir *bit stuffing* para possibilitar a extracção de relógio para efeitos de sincronização. Tal é feito de modo a garantir uma alternância de bit pelo menos de 6 em 6 bits.

Uma unidade denominada por DPLL é utilizada para a sincronização dos dados recebidos e para garantir o *bitrate* nominal nos dados transmitidos. Funções de *bus timing* executadas dentro de cada *bit frame*, como a sincronização com o oscilador local, compensação do atraso introduzido pela rede e posicionamento de amostragem, implicam a particionamento de cada bit em vários segmentos definidos a partir de períodos de tempo mínimos chamados *Time Quanta* (T_Q).

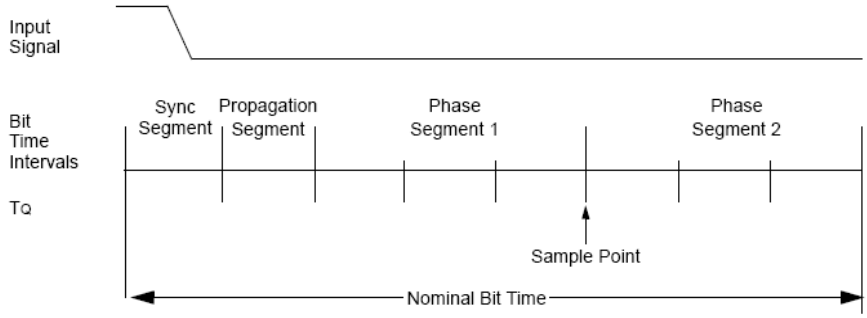


Fig. 18: Particionamento temporal de um bit.

Pela Fig. 18 podemos identificar 4 tipos de segmento:

1. Segmento de sincronização: utilizado para sincronizar os vários nós da rede – duração fixa de 1 T_Q;
2. Segmento de propagação: utilizado para compensar os atrasos introduzidos na rede – duração entre 1 e 8 T_Q.
3. Segmento de fase 1: número de T_Q antes da amostragem do bit – duração entre 1 e 8 T_Q.
4. Segmento de fase 2: fornece um atraso antes da mensagem seguinte – duração de 2 a 8 T_Q.

A duração dos segmentos de fase 1 e 2 não são parâmetros estáticos e podem sofrer variações para efeitos de ressincronização. O parâmetro SJW (*Synchronization Jump Width*) define a forma como se farão estas flutuações contendo o número máximo de T_Q que serão adicionados ao segmento de 1 ou subtraídos ao segmento 2.
Para definição das durações de cada segmento algumas regras deverão ser respeitadas. Elas são:

- ✓ 1 + Seg. de propagação + Seg. de fase 1 + Seg. de fase 2 ≥ 8
- ✓ Seg. de propagação + Seg. de fase 1 ≥ Seg. de fase 2
- ✓ Seg. de fase 2 ≥ SJW

Para o nosso caso, de forma a conseguir a máxima velocidade, definimos definiram-se os seguintes parâmetros:

- Segmento de sincronização: 1 T_Q (não configurável);
- Segmento de propagação: 3 T_Q;
- Segmento de fase 1: 3 T_Q;
- Segmento de fase 2: 3 T_Q;
- SJW: 2 T_Q.

Com base nestes parâmetros, e de modo a conseguir a máxima velocidade de F_{bit}=1Mbps, o *baudrate* a definir no registo BRP seria de 1.5. Como estes registos são inteiros têm que se arredondar para o lado que diminua a velocidade, ou seja, para cima. Neste caso BRP deverá ser de 2!

$$BRP = \frac{2 * F_{CPU}}{F_{bit} * N_{seg}} - 1, \quad com \quad N_{seg} = Sync_{seg} + Propag_{seg} + Phase_1 + Phase_2$$

$$F_{bit} = \frac{2 * F_{CPU}}{(BRP + 1) * N_{seg}} = \frac{2 * 10\ \text{MHz}}{(1+1) * 10} = 1\ \text{Mbps}$$

Para BRP=1, o *bitrate* nominal efectivo atingirá o valor máximo de 1Mbps.

Máscaras e Filtros

Os filtros são utilizados para automatizar o processo de aceitação de mensagens. No caso das mensagens de actuação, apenas se destinam a um SCU particular não possuindo qualquer interesse para os restantes. Desta forma, na perspectiva de cada SCU, apenas algumas das muitas mensagens que passam pelo seu porto de entrada devem ser processadas, pelo que se a verificação do endereço destino fosse feita por software, a largura de banda de CPU atribuída a este processo seria tanto mais pesada quanto maior fosse o número de unidades *slave* na rede.
De modo a automatizar o processo de aceitação, uma máscara pode ser utilizada para indicar quais os bits do identificador que contém o padrão de interesse para a unidade em questão (como por exemplo o endereço destino) e filtros podem ser utilizados para fazer a selecção dos padrões e redireccionar para determinados *buffers*. Desta forma, apenas as mensagens cujos bits da secção do identificador programado coincide com o padrão especificado, são aceites e carregados para os *buffers* de recepção gerando uma interrupção para o posterior processamento.

Máscara do bit n	***Filtro do bit n***	***Valor do bit n***	***Resultado***
0	x	x	Rejeitado
1	0	0	Aceite
1	0	1	Rejeitado
1	1	0	Rejeitado
1	1	1	Aceite

Tabela 24: Resultado da filtragem para cada bit.

Pela Tabela 24 podemos verificar que apenas os bits definidos a 1 pela máscara serão considerados para filtragem. Todos os outros serão rejeitados a priori. Dos que são considerados para filtragem são comparados com um determinado padrão sendo apenas aceites os que coincidem com esse padrão. Apenas as mensagens em que todos os bits do filtro são aceites, são consideradas para processamento por parte do programador.
Para o PIC 18F258 é possível utilizar todos os bits do identificador na máscara e filtragem e definir até dois padrões de filtragem para armazenamento imediato no primeiro *buffer* de recepção e até 4 padrões para armazenamento no segundo *buffer* (Fig. 19).

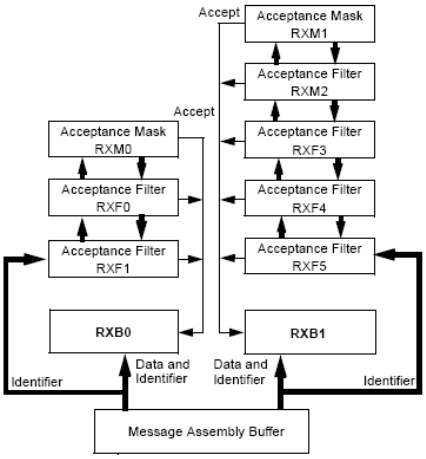


Fig. 19: Registos associados à máscara e filtragem.

Para o nosso caso em concreto, de acordo, com a estrutura do identificador enunciado anteriormente, a máscara a definir cobrirá os bits do endereço destinatário e do *index* do pacote, ou seja, possuirá o seguinte padrão:

bit 10	*9*	*8*	*7*	*6*	*5*	*4*	*3*	*2*	*1*	*bit 0*
Source address				Destination address				Index	Operation	
0	0	0	0	1	1	1	1	1	0	0

Máscara a aplicar ao identificador

De seguida definir-se-ão dois filtros, de modo a aceitar unicamente as mensagens cujo endereço destino corresponda ao próprio SCU, e destes, redireccionar os de *index* 0 (primeira mensagem) para o *buffer* de recepção 0, e os de *index* 1 (segunda mensagem) para o *buffer* de recepção 1 (Tabela 25).

Redirecção dos pacotes	***Registos a configurar***	***Padrão***
Buffer 0	RXF0	0 0 0 0 a_3 a_2 a_1 a_0 0 0 0
	RXF1	0 0 0 0 a_3 a_2 a_1 a_0 0 0 0
Buffer 1	RXF2	0 0 0 0 a_3 a_2 a_1 a_0 1 0 0
	RXF3	0 0 0 0 a_3 a_2 a_1 a_0 1 0 0
	RXF4	0 0 0 0 a_3 a_2 a_1 a_0 1 0 0
	RXF5	0 0 0 0 a_3 a_2 a_1 a_0 1 0 0

Tabela 25: Configuração dos filtros para redireccionamento de pacotes para os dois buffers de recepção (padrão a3 a2 a1 a0 = endereço do SCU).

Desta forma, redireccionando cada uma das duas mensagens para um *buffer* à parte, evita-se a perda de informação caso ocorra *overflow*, pois apenas o *buffer* cuja informação é equivalente à nova mensagem (*index* igual) é substituído.

Transmissão e Recepção de Mensagens

Finalmente, também é necessário configurar a forma como as mensagens serão transmitidas e recebidas.

Para a recepção, é necessário definir a recepção de apenas pacotes válidos cujo identificador segue o formato da versão CAN 2.0A, ou seja de 11 bits, e deve-se assegurar que o *overflow* do *buffer* 0 para o *buffer* 1 está desactivado, dado que estamos a usar um *buffer* para cada índice de mensagem.
Finalmente devemos colocar o bit RXFULL de cada *buffer* a zero, para a sua abertura à recepção de novas mensagens, verificando também se o *interrupt flag* associado está desligado.

Para a transmissão, os identificadores das mensagens deverão estar de acordo com o formato *standard*, ou seja, deverão possuir 11 bits de comprimento, e é necessário definir prioridades para cada um dos três *buffers*, para definição da ordem de transmissão no caso de vários agendamentos simultâneos.
Por questões de similaridade de comportamento, também se reservará um *buffer* de transmissão para cada tipo de mensagem, fazendo apenas uso de dois *buffers*. O *buffer* 0 utilizado para as mensagens de *index* 0 possuirá a máxima prioridade e o *buffer* 1 possuirá a prioridade mínima (Tabela 26).

Buffer	***Pacotes associados***	***Prioridade***	***Registos***
Buffer 0	*Index* 0	Máxima	TXB0CON=*0b*11
Buffer 1	*Index* 1	Mínima	TXB1CON=*0b*10
Buffer 2	——	——	TXB2CON=*0b*01

Tabela 26: Atribuição de prioridades entre cada buffer de transmissão.

Modo de Operação Normal

Para colocar o CAN a funcionar só resta definir o modo de operação normal no registo CANCON.

Resumindo...

1. Colocação do PIC no modo de configuração;
2. Definição da duração de cada segmento de bit;
3. Definição do *bitrate* para a velocidade pretendida;
4. Configuração da máscara de recepção;
5. Configuração dos dois filtros para cada um dos *buffers* de recepção;
6. Definição do formato *standard* para os identificadores dos pacotes a transmitir;
7. Definição de prioridades entre *buffers* de transmissão;
8. Configuração da recepção para apenas pacotes válidos com identificadores de 11 bits;
9. *Overflow* do *buffer* 0 para o *buffer* 1 desactivado;
10. *Clear* dos *interrupt flags* de recepção;
11. Abertura dos *buffers* para recepção;
12. Colocação do PIC no modo de operação normal.

(Os passos 6 a 11 devem ser aplicados a cada um dos buffers de recepção/transmissão.)

B. Troca de Mensagens CAN

Esta secção apresenta os procedimentos a efectuar nos PICs 18F258 para a troca de pacotes utilizando a rede CAN. Estes microcontroladores oferecem uma gama diversificada de registos que permitem controlar diversos aspectos, como por exemplo o comprimento do identificador ou o conjunto de dados e, também, no domínio da recuperação de erros.

Na recepção de um pacote que passe a máscara e um dos dois filtros, o seu conteúdo é carregado no *buffer* associado ao seu *index* n. Os procedimentos a seguir descritos devem-se aplicar relativamente a esse *buffer*:

1. Verificação da recepção de uma mensagem inválida através da *flag* PIR3bits.IRXIF;
2. Confirmação da utilização do *buffer* n através da *flag* RXFULL;
3. Reposição do *interrupt flag* associado ao *buffer* n (PIR3bits.RXBnIF) a zero;
4. Transferência do *buffer* n para o *Acess Bank Area*, de modo a se poder aceder a ele a partir de qualquer banco de memória. Para tal deve-se definir a janela de endereçamento de acordo com o *buffer* em uso (*window address bits*);
5. Verificação da ocorrência de *overflow*;
6. Leitura do identificador do pacote;
7. Validação do comprimento de dados recebidos;
8. Leitura dos dados;
9. Libertação do *buffer* n pela recolocação da *flag* RXFULL a zero;
10. Reposição da janela de endereçamento para o seu valor original;

Transmissão de um pacote usando o *buffer* n:

1. Verificar se o *buffer* n está pronto para transmissão (TXBnCONbits.TXREQ deve ser zero);
2. *Clear* do *interrupt flag* associado ao *buffer* n (TXBnIF).
3. Definição da janela de endereçamento para o *buffer* de transmissão n;
4. Definição do identificador;
5. Definição do comprimento do pacote (8 bytes);
6. Definição do conjunto de dados a enviar;
7. Activação da transmissão do *buffer* n (TXREQ=1);
8. Reposição da janela de endereçamento para o seu valor original;

Os procedimentos de configuração e de troca de mensagens estão definidos numa biblioteca denominada por *candrivers* para utilização por parte do programador.

3.4. Unidade Master

A. Algoritmo de Troca de Mensagens

As trocas de informação *Master-Slave* são desencadeadas a partir da unidade *Master*, estando os Slaves programados para responder imediatamente após a recepção das mensagens oriundas do *Master*. A Fig. 20 apresenta os algoritmos utilizados na unidade *master* para gerir a troca de informação.

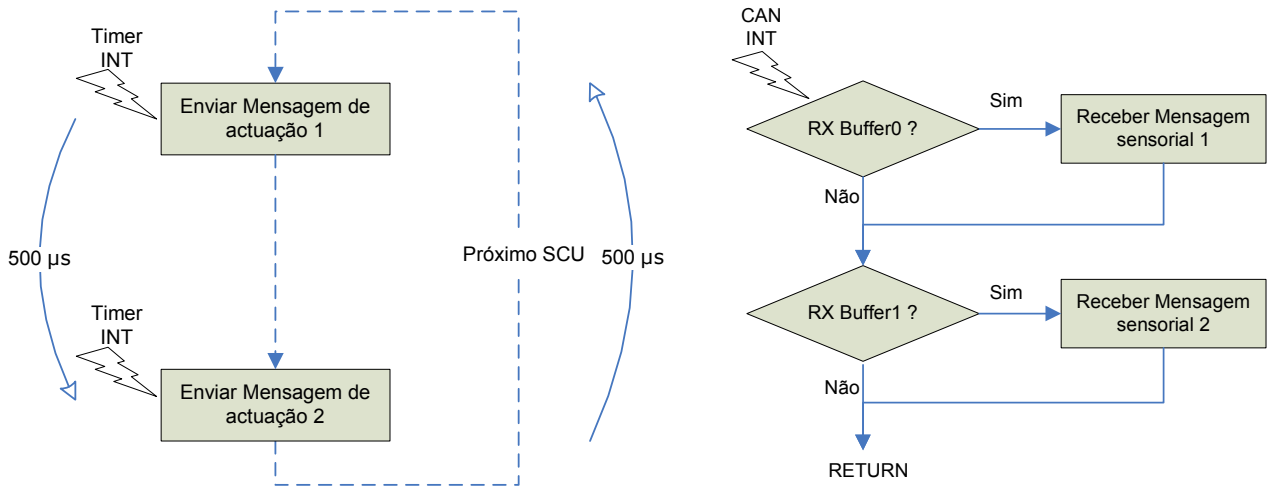


Fig. 20: Algoritmos de transmissão/recepção de mensagens CAN no Master.

As transmissões de mensagens de actuação são desencadeadas a partir de um *timer* programado para gerar interrupções de 500 em 500µs. Como duas mensagens tem de ser transmitidas a cada slave, cada um é servido durante 1 ms, correspondendo a um tempo de *turn-around* de 8 ms para 8 slaves.
Relativamente às recepções das mensagens sensoriais provenientes dos *slaves*, são processadas através de interrupções CAN segundo a política *event-triggered*: sempre que uma nova mensagem chega, uma interrupção CAN é gerada, e quando possível, é processada actualizando a base de dados sensorial.
Estas duas fontes de interrupções foram definidas como sendo de baixa prioridade, relativamente às interrupções da linha série, pelo que o serviço CAN só pode ser atendido quando não há caracteres na porta série. Desta forma, garante-se uma boa velocidade de resposta aos comandos enviados pelo PC, sem comprometer significativamente a performance CAN para o período de transmissão citado de 500 µs.

B. Transmissão e Recepção de Mensagens

Do ponto de vista do *Master* todas as mensagens a enviar são de actuação, e todas as recebidas são de carácter sensorial, pelo que, no último caso, é escusado o processamento do identificador, excepto para o conhecimento do SCU fonte e do *index* do pacote. No caso do endereço destino e da operação apenas são verificados para efeitos de teste do correcto funcionamento.

As mensagens de actuação a enviar vão buscar os valores a aplicar nos *slaves* à base de dados global que o PC periodicamente actualiza. Já as mensagens de carácter sensorial que vão chegando dos *slaves*, actualizam constantemente as variáveis da mesma base de dados. É este mecanismo que garante a periódica actualização da base de dados sensorial e das unidades *slave* no que toca à actuação.

Procedimento de envio de uma mensagem de actuação:

1. Construção do identificador com base no endereço destino e no *index* do pacote (endereço origem e operação conhecidos);
2. Construir o *array* de dados a enviar, a partir da base de dados de actuação, de acordo com o *index* da mensagem;
3. Enviar o pacote (identificador + *array* de dados) pelo *buffer* de transmissão adequado ao *index*;
4. Actualização do estado de erro, se ocorreu algum.

Recepção de um pacote de carácter sensorial:

1. Obtenção do identificador e do *array* de dados do pacote recebido;
2. Actualização da base de dados sensorial, a partir do endereço origem e do *index* do pacote;
3. Actualização do estado de erro, se ocorreu algum.

Operação I/O	***Causa do Erro***	***Label***	***Padrão***
Recepção/Transmissão	Erro geral	CAN_ERROR	*0b*00000001
Recepção	Mensagem inválida	RX_INVALIDMSG_ERROR	*0b*00000010
	Overflow	RX_OVERFLOW_ERROR	*0b*00000100
	Comprimento de dados inválido	RX_INVALID_LENGTH	*0b*00001000
	Identificador inválido	RX_INVALID_ID	*0b*00010000
	Dados inválidos	RX_INVALID_DATA	*0b*00100000
Transmissão	Erro de transmissão	TX_NOTPOSSIBLE	*0b*01000000
	Transmissão impossível	TX_ERROR	*0b*10000000

Tabela 27: Causas de erro na comunicação CAN.

Na ocorrência de erros, uma estrutura estática armazena a causa do erro na forma de um byte fazendo a operação booleana *or* entre os diversas códigos segundo a Tabela 27. Duas outras variáveis fazem a contagem dos erros de transmissão e de recepção.

```
// Estrutura descritiva dos erros ocorridos
volatile struct  {
    enum CAN_STATUS status;     // Flags originais dos dev.drivers sobre a causa de erro
    enum CAN_ERRORS flags;     // Causas
    byte tx_count, rx_count;   // Número de ocorrências
} canErrors={false,0,0};
```

Quando um comando USART de operação *OP_READ_EXTBUFF* é pedido pelo PC, uma mensagem é devolvida contendo a causa do erro e a quantidade de erros de transmissão e de recepção. Uma vez consultados, estes valores são reinicializados a zero.

byte 1	*byte 2*	*byte 3*	*byte 4*	*byte 5*	*byte 6*
*0x*FB	==	Error Code	# TX errors	# RX errors	0

Mensagem de leitura do estado do barramento CAN

3.5. Unidades Slave

A. Algoritmo de Troca de Mensagens

Nas unidades *slave*, a comunicação CAN é efectuada segundo a política *event-triggered*, dado que apenas é accionada quando são recebidas mensagens do *master*. A Fig. 19 evidencia o algoritmo presente em cada unidade slave.

Quando uma mensagem é recebida no *buffer* 0 (primeira mensagem de actuação) uma interrupção é gerada, e na Rotina de Serviço à Interrupção (RSI) os dados de actuação são actualizados e uma mensagem com informação sensorial é enviada (primeira mensagem de resposta). Por sua vez, se a interrupção corresponder à chegada de uma mensagem no *buffer* de recepção 1, o processo é equivalente mas respeitante à segunda mensagem de cada transacção.

De notar ainda, que as interrupções CAN são de baixa prioridade, para não permitir interferência com as tarefas essenciais de controlo de PWM sobre os motores.

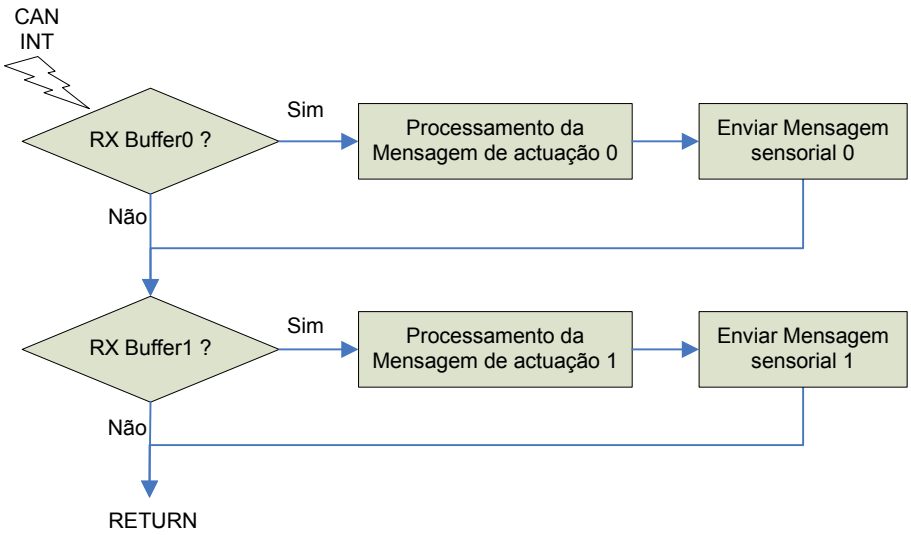


Fig. 21: Algoritmo de troca de informação pelo CAN no Slave.

B. Transmissão e Recepção de Mensagens

Do ponto de vista de cada *slave*, todas as mensagens a enviar são de carácter sensorial, e todas as recebidas são de actuação, pelo que, no último caso, é escusado o processamento do identificador, excepto para o conhecimento do *index* da mensagem. O endereço origem (MCU), endereço destino (o próprio) e da operação apenas são verificados para efeitos de teste do correcto funcionamento da troca de mensagens.

Envio de uma mensagem de carácter sensorial:

1. Construção do identificador com base no *index* da mensagem (endereço origem, destino e operação conhecidos);
2. Construir o *array* de dados a enviar de acordo com o *index* da mensagem, a partir dos dados sensoriais obtidos directamente;
3. Enviar a mensagem (identificador + *array* de dados) pelo *buffer* de transmissão adequado ao *index*;

Recepção de um pacote de actuação:

1. Obtenção do identificador e do *array* de dados da mensagem recebida;
2. Actualização dos actuadores de acordo com o index da mensagem;

No caso das unidades *slave*, dispensa-se o processamento de erros, uma vez que se pretende reduzir ao mínimo o consumo de largura de banda da rede. O processamento de erros, implicaria a transmissão de mensagens para o *master* do estado da unidade, o que poderia aumentar o risco de saturação da rede.

C. Base de Dados Local

Como as novas informações de actuação nunca são aplicadas imediatamente, torna-se necessário armazenar estes valores numa base de dados local para posterior acesso pelas tarefas de controlo local. O mesmo se passa com a informação sensorial que quando é gerada tem que ser armazenada para que a tarefa de comunicação CAN possa acedê-la mais tarde para o envio das mensagens de resposta.
Esta base de dados local é muito semelhante à base de dados apresentada na unidade *master*, mas agora com a diferença de ser de carácter local, ou seja, apenas armazena a informação sensorial ou de actuação de um único SCU – o próprio.

```
// Tipo enumerado do género de controlador implementado em cada junta
typedef enum {
    NO_CONTROL  = 0b00,             // Funcionamento em malha aberta
    COP_CONTROL = 0b01,             // Controlo de equilíbrio no tronco
    INC_CONTROL = 0b10,             // Controlo das forças de reacção nos pés
    GIRO_CONTROL= 0b11              // Controlo de posição e velocidade
} enum_controlType;

// Estrutura descritiva do estado de um servo
typedef struct {
    signed char position;                                   // Posição referência
    signed char velocity;                                   // Velocidade
    unsigned char current;                                  // Corrente consumida
} struct_servo;
```

Base de dados sensorial

```
// Estrutura descritiva dos sensores
typedef struct {
    struct {
        bool pwm;                           // Motores ligados/desligados
        bool calib;                         // Calibração ligada/desligada
        bool deadlineError;                 // Violação de deadline
        bool motionFinAll;                  // Todos os motores terminaram o movimento
        bool motionFinOne;                  // Um dos motores terminaram o movimento
        bool motionFin[N_SERVOS];           // Movimento terminado
    } sysStatus;                                  // Estado sensorial do sistema
    struct_servo servo[N_SERVOS];                 // Sensores dos servos
    unsigned char special[N_SPECIAL_SENSORS];     // Sensores especiais
} struct_sensors;
extern volatile struct_sensors   sensors;         // Sensores
```

Base de dados de actuação

```
// Estrutura descritiva dos actuadores
typedef struct {
    // Estrutura de actualização de status para cada SCU
    struct {
        bool pwm;                           // PWM on/off
        bool calib;                         // Dynamic calibration on/off
    } sysStatus;                                  // Estado da actuação do sistema
    struct {
        byte kp, kd, ki;                    // Ganhos dos controladores
        byte type;                          // Tipo de Controlador de primeiro nivel
    } control[N_SERVOS];                          // Controlador
    struct_servo servo[N_SERVOS];                 // Informação de actuação
(posição, velocidade e corrente)
} struct_actuators;
extern volatile struct_actuators  actuators;      // Actuadores
```

De forma a minimizar a interferência temporal com as restantes tarefas da unidade, as interrupções de recepção de mensagens provenientes do *master* (Fig. 21) foram definidas como sendo de baixa prioridade, enquanto as restantes são de alta. Desta forma a troca de mensagens em cada *slave* só é feita em *background*, ou seja, nos tempos livres do processador.

4. SOFTWARE NAS UNIDADES MICROCONTROLADORAS

4.1. Unidade Master

As relações de inclusão entre os diversos módulos estão esquematizadas na Fig. 20. Cada módulo representa um par de ficheiros *.h* e *.c* com o protótipo das funções externas e a sua implementação respectivamente. Apenas o módulo *Master* apenas possui um ficheiro *.c* dado que é o que contém a função *main*.

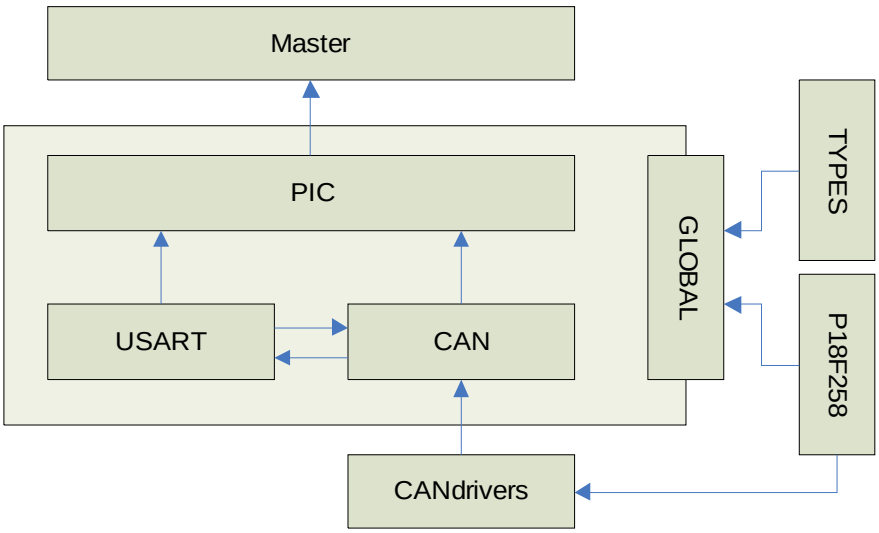


Fig. 22: Relações de inclusão dos módulos de software do Master.

Módulo Master

Este módulo é o primeiro a ser executado (função *main*) e é responsável por fazer a chamada das funções de configuração presentes no módulo PIC. Após os procedimentos de configuração, fica preso numa *dummy task* em que o único software em execução é proveniente da rotina de serviço à interrupção definida no módulo PIC.

Módulo PIC

Módulo com a implementação da rotina de inicialização do PIC e das rotinas de serviço às interrupções provenientes da USART e do CAN. É neste módulo que são inicializados e implementados os mecanismos de comunicação com a unidade principal (secção II. 2.4) e as unidades *slave* (secção II. 3.4).

Tabela 28: Funções do módulo PIC.

Funções	***Descrição***
initPic	Inicialização dos periféricos associados às comunicações USART e CAN.
highISR	Implementação dos mecanismos de comunicação pela USART com a unidade principal.
lowISR	Implementação dos mecanismos de comunicação CAN com as unidades *slave*.

Módulo USART

Módulo com os *device drivers* para inicialização (secção II. 2.4) e implementação do protocolo USART descrito na secção II. 2.2.

Tabela 29: Funções para manipulação dos *buffers* da USART.

Funções	***Descrição***
usartInit	Inicialização da USART no PIC.
usartStreamMode	Esta função indica se o *buffer* de recepção está vazio ou não. Em caso negativo a USART está em modo de recepção de um *stream* de bytes (*true* retornado).
usartStoreRx	Armazenamento de um byte no *buffer* de recepção.
usartGetTx	Obtenção de um byte armazenado no *buffer* de transmissão.
usartResetRxBuff	Reinicialização a zero de todo o *buffer* de recepção.
usartResetTxBuff	Reinicialização a zero de todo o *buffer* de transmissão.
usartStore2extBuff	Armazenamento de um byte no *buffer* externo.

A função *usartStore2extBuff* refere a existência de um terceiro *buffer* denominado por *buffer* externo. Tal destina-se aos módulos externos que pretendam enviar informação adicional pela USART. Tal é o caso do módulo CAN que disponibiliza à unidade principal, através desta função, o estado de funcionamento do barramento CAN (estado de erro). Para o PC aceder a estes dados apenas tem de enviar um comando com o *operation*=OP_READ_EXTBUFF. O formato da mensagem de resposta está descrita na secção II. 2.2.
A Tabela 30 apresenta um conjunto de funções de mais alto nível, que fazem uso dos *device drivers* explícitos na Tabela 29, para processamento dos comandos provenientes do PC e construção da mensagem de resposta a retornar.

Tabela 30: Funções de construção da mensagem de resposta para uso da Rotina de Serviço à Interrupção.

Funções	***Descrição***
usartSendTestMsg	Rotina de envio de uma mensagem com o formato FA F9 F8 F7 F6 F5 (hex). Esta mensagem tem como propósito testar o funcionamento da USART.
usartSendStatus	No caso dos comandos de actuação, dois tipos de mensagens podem ser retornados, excluindo as mensagens com erros de recepção por parte do master. Elas são a confirmação da actuação com sucesso ou parâmetros inválidos. Em tais casos é enviada uma mensagem com todos os bytes iguais ao do comando de actuação à excepção do primeiro com a indicação do estado: MESSAGE_SUCESS (0xFB) ou MESSAGE_INVREQ (0xFC). Esta função tem o objectivo de construir uma mensagem igual à última recebida com o primeiro byte igual ao estado a retornar.
usartProcessMsg	Rotina de processamento dos comandos enviados do PC para o master para construção da mensagem de resposta.

Módulo CAN

Módulo com a implementação das funções de alto nível para a recepção e o envio de mensagens CAN (secção II. 3.4) segundo o protocolo descrito na secção II. 3.2.

Tabela 31: Funções de alto nível para troca de mensagens via CAN.

Funções	***Descrição***
canInit	Inicialização do periférico CAN e dos *timers* e interrupções associados.
canSendMsg	Envio de uma mensagem CAN com dados de actuação a aplicar a um determinado SCU.
canReceiveMsg	Recepção de uma mensagem CAN com os dados sensoriais de um determinado SCU.
canClearStatus	Reinicialização do estado de erro do barramento CAN.

Módulo CANDRIVERS

Módulo com os device drivers básicos para inicialização e transmissão/recepção de pacotes via CAN (secção II. 3.3).

Tabela 32: Device drivers da comunicação CAN.

Funções	*Descrição*
myCANInitialize	Inicialização do periférico CAN.
myCANSendMessage	Envio de um pacote através do primeiro *buffer* de transmissão vazio.
myCANSendMessage0	Envio de um pacote através do *buffer* de transmissão 0.
myCANSendMessage1	Envio de um pacote através do *buffer* de transmissão 1.
myCANSendMessage2	Envio de um pacote através do *buffer* de transmissão 2.
myCANReceiveMessage	Leitura de um pacote no primeiro *buffer* de recepção cheio.
myCANReceiveMessage0	Leitura de um pacote no *buffer* de recepção 0.
myCANReceiveMessage1	Leitura de um pacote no *buffer* de recepção 1.

Módulo GLOBAL

Módulo com a definição da base de dados com os dados de actuação e sensorial de todos os SCUs da arquitectura (estruturas descritas na secção II. 2.4. C).

Tabela 33: Funções presentes no módulo GLOBAL.

Funções	*Descrição*
initGlobal	Inicialização de toda a base de dados para os valores origem.
initSensors	Inicialização somente da base de dados sensorial.
initActuators	Inicialização somente da base de dados de actuação.

Módulo TYPES

Módulo com a definição de tipos de variáveis extra para uso nos restantes módulos. Eles são:

Tabela 34: Tipos de variáveis definidas no módulo TYPES.

Tipo de variável	*Descrição*
bool	Tipo booleano (*true* ou *false*) (tipo enumerado).
byte	Tipo inteiro de 8 bits sem sinal (*unsigned char*).
word	Tipo inteiro de 16 bits sem sinal (*unsigned int*).
dword	Tipo inteiro de 32 bits sem sinal (*unsigned long*).

Módulo P18F258

Biblioteca com a definição de todos os registos e bits correspondentes do PIC 18F258 para o seu controlo (ver *datasheet* da *Microchip*, PIC18F258).

4.2. Unidades Slave

A Fig. 23 apresenta as relações de inclusão entre os vários módulos de *software* da unidade *slave*. Os módulos a verde são iguais aos utilizados na unidade *master*, pelo que dispensam apresentações. Quanto aos módulos a vermelho não estão relacionados com as comunicações e por isso não serão referidos neste capítulo.

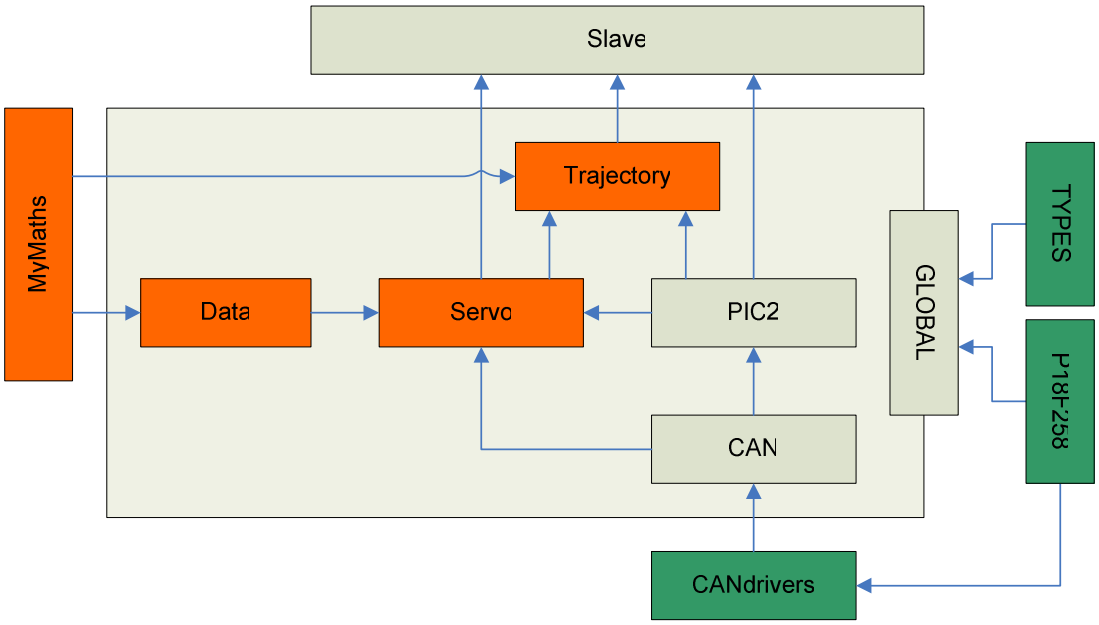


Fig. 23: Relações de inclusão dos módulos de software de cada Slave.

Módulo Slave

Módulo com a função iniciadora *main*, responsável pela chamada das funções de configuração presentes nos outros módulos e por executar tarefas em *background* durante o funcionamento normal.

Módulo PIC2

Módulo com a implementação da rotina de inicialização do PIC e da rotina de serviço às interrupções, uma das quais provenientes do CAN. É neste módulo que são inicializados e implementados os mecanismos de comunicação com a unidade *master* (secção II. 3.5. A).

Tabela 35: Rotinas do módulo PIC2 responsáveis por gerir as comunicações CAN.

Funções	***Descrição***
initPic	Inicialização das comunicações CAN e de outros periféricos destinados ao controlo local.
lowISR	Rotina de Serviço à Interrupção responsável por gerir as interrupções provenientes do CAN.
...	(outras tarefas para controlo local)

Módulo CAN

Módulo com a implementação das funções de alto nível para a recepção e o envio de mensagens CAN (secção II. 3.5. B) segundo o protocolo descrito na secção II. 3.2, mas no que respeita à unidade *slave*.

Tabela 36: Funções de alto nível para troca de mensagens via CAN.

Funções	***Descrição***
initCan	Inicialização do periférico CAN e das interrupções associadas.
sendCanMessage	Envio de uma mensagem CAN com dados de sensoriais para a unidade master.
receiveCanMessage	Recepção de uma mensagem CAN proveniente do master com os dados de actuação a aplicar.
checkCan	Verificação do bloqueio do CAN, no caso do PIC se encontrar no modo *bus-off*.

A função *checkCan* é útil para verificar se a unidade ainda possui a interface CAN activa e em funcionamento. Cada nó na rede pode comportar-se de três formas diferentes durante o seu funcionamento:

- Modo *error-active*: mensagens normais e frames de erro (na ocorrência de erros) com bits dominantes são trocadas com os outros nós para lhes indicar a ocorrência de anomalias;
- Modo *error-passive*: as frames de erro passam a ser constituídas por bits recessivos para evitar a interferência destrutivas das mensagens provenientes de outros nós;
- Modo *bus-off*: o nó é bloqueado em termos de recepções e transmissões.

Na ocorrência de erros, um contador é incrementado, sendo também decrementado na ausência deles. Quando atinge um valor limite o modo de funcionamento vai alternando para o modo seguinte até atingir o modo *bus-off*, bloqueando as recepções e as transmissões com os outros nós.
Esta função foi construída devido ao facto de as unidades esporadicamente bloquearem as suas comunicações e ter assim um meio para verificar a causa.

Módulo GLOBAL

Módulo com a definição da base de dados com os dados de actuação e sensorial do próprio SCU (secção II. 3.5).

5. PROBLEMAS REPORTADOS

Implementando os algoritmos descritos em cada nó, observou-se uma eficiente troca de mensagens sem corrupção de dados e sem atrasos aparentes.

No entanto, verificava-se esporadicamente que, nalgumas unidades *slave*, as comunicações CAN bloqueavam ao fim de vários minutos de funcionamento. Testes mais aprofundados revelaram que este problema acontecia de igual em qualquer unidade, seja *Master* ou *Slave*, o que levou a crer que a fonte do problema reportava-se ao nível dos *device drivers* CAN (ficheiros *CANdrivers.**). Mais ainda, observou-se que na unidade *Master*, levava muito menos tempo a acorrer bloqueios, o que fazia crescer ainda mais esta suspeita dado que os device drivers no *Master* são executados mais frequentemente que em cada *slave* (8 vezes mais para 8 *slaves*).
O bloqueio começava a revelar-se na unidade *master*, pela ausência de actualização da base de dados sensorial por parte de alguns *slaves* (aleatórios), muito embora se assegurasse que as mensagens CAN de conteúdo sensorial estavam a ser transmitidas por parte deles. Tal sugeria que o *Master*, a determinado ponto, deixava de processar as mensagens CAN provenientes de alguns *slaves*.
Do ponto de vista das unidades *slave*, o bloqueio observava-se pela incapacidade de recepção de mensagens num determinado buffer de recepção. Mais tarde, o segundo buffer seria igualmente afectado, privando esta unidade a processar as duas mensagens de actuação provenientes do *Master*.

Como o atendimento às mensagens recebidas, é feita a partir das interrupções geradas pelo módulo CAN quando uma nova mensagem é armazenada num *buffer*, a suspeita recaiu que esta interrupção deixava de ser produzida, ou então que a *flag* indicativa deste evento (*interrupt flag*) deixava de ser actualizada.
Por análise dos device drivers de recepção de mensagens, verificou-se que, após processamento de uma nova mensagem recebida, a *interrup flag* só era reposta a *false* depois de disponibilizar o buffer para uma nova recepção (*RXFULL* a *false*). Para mais pormenores queira ler a secção II. 3.3. B. Tal procedimento podia gerar a seguinte situação:

1. Buffer era disponibilizado para uma nova recepção: *RXFULL* a zero;
2. Uma nova mensagem CAN chegava à unidade, sendo carregada na MAB (*Message Assembly Buffer*) e executados os procedimentos automáticos de aceitação de mensagem através da máscara e filtros definidos. No caso de uma mensagem válida e destinada à unidade em causa, ela é carregada no respectivo *buffer* de recepção, sinalizando a correspondente *interrupt flag* a 1.
3. Reposição da *interrupt flag* associada ao *buffer* em causa a 0.

Note que o segundo passo, é um procedimento automático do PIC, constituindo uma interrupção de baixo-nível durante a execução da rotina de processamento de uma nova mensagem. Como se pode observar, mesmo que a interrupção seja produzida, a *interrupt flag* correspondente é reposta a zero, apagando quaisquer pistas acerca da sua origem, e impedindo desta forma que interrupções futuras sejam processadas (a flag *RXFULL* nunca mais é reposta), resultando num bloqueio de recepção de uma das mensagens. Claro que mais tarde, o mesmo acontecerá ao segundo buffer de recepção, incapacitando o processamento de quaisquer novas mensagens. Embora, isto apenas aconteça, se a interrupção ocorrer neste ponto específico, o que representa apenas uma probabilidade de 0.0025% de ocorrência para uma instância, do ponto de vista dos *slaves* (considerando a recepção de duas mensagens com uma periodicidade de 8 ms para um determinado *slave* – 1ms*8 *slaves* – existem 2 possibilidades em 80000 instruções executadas durante 8 ms para $F_{\text{instrução}}=F_{\text{CPU}}$=10 MHz), tal valor não pode ser desprezado. Isto porque, a probabilidade de não bloqueio, que é de 99.9975% para uma instância, ao fim de cinco minutos desce para apenas 39%!

$$P_{sucesso} = (1 - P_{falha})^{N} = (1 - 0.000025)^{(5min*60s/8ms)} = 0.392 = 39\%$$

Para a unidade *Master*, a recepção de mensagens é de 8 vezes superior (uma transacção a cada milisegundo), o que ainda diminui mais esta probabilidade.

Ao efectuar a reposição da interrupt flag para antes da disponibilização do buffer, este problema é imediatamente eliminado sem quaisquer efeitos colaterais, pelo que neste momento o sistema se encontra a operar à máxima eficiência.

Um fenómeno que ainda no presente se observa, é na situação em que o *drive* RS-232 é removido sobre a unidade *Master*, o que impossibilita a comunicação CAN de forma permanente. Quando este *drive* é removido acontece que o pino de recepção série do PIC, RX, fica em alta impedância, fazendo com que sucessivos caracteres "fantasma" surjam nesta entrada ocupando toda a largura de banda de CPU. Ora, como a comunicação série é de prioridade superior em relação à CAN, os algoritmos de comunicação relativos ao CAN vêem-se sem possibilidade de execução. Para evitar este efeito, sugere-se a adição de uma resistência suficientemente elevada entre este pino e a alimentação, conferindo o estado "1" na ausência de *driver* (o estado *idle* corresponde ao estado "1").

6. CONCLUSÕES

A arquitectura de comunicações adoptada para a estrutura humanóide, revelou-se bastante simples de implementar seguindo uma lógica bastante intuitiva. Esta arquitectura segue um formato distribuído, com a associação de cada trio de juntas (normalmente associadas a um membro físico) a uma unidade de controlo local baseada num PIC que se responsabiliza em aplicar os sinais de controlo e ler a informação sensorial. Estas unidades de controlo interligam-se umas às outras por meio de um único barramento que partilha as mensagens em circulação a todas as unidades de controlo a ele ligadas. Este barramento também se interliga a uma unidade mestre (*master*) que por sua vez está conectada à unidade principal, neste caso um PC, que é responsável por enviar os comandos de actuação e de leitura sensorial a cada unidade local. Deste modo, cada unidade tem uma função específica, deslocalizando assim, da unidade principal, todas as tarefas de controlo dos servomotores e de distribuição das mensagens entre os diversos *slaves*.

Implementando os protocolos de comunicação tanto série entre o PC e o *master*, como o CAN entre o *master* e os *slaves* seguindo a metodologia de funcionamento enunciada, a fiabilidade das comunicações revelou-se bastante boa com a troca de mensagens à velocidade máxima possível e sem a detecção de corrupção de dados ou de atrasos excessivos na sua entrega.

Como interface, utiliza-se o *software MatLab* na unidade principal (PC) para fazer a interface uma vez que oferece uma linguagem de alto nível e intuitiva para programação das sequências de comandos a aplicar nas juntas e também para monitorar os sinais de saída (ver secção II. 2.1).

III. LOCOMOÇÃO BASEADA EM SENSORES DE FORÇA

Resumo:
Este capítulo pretende explorar os sensores de força presentes na base dos pés, para a realização de trajectórias específicas, por parte da perna de suporte. Deste modo, abrange-se a utilidade destes sensores que, além de poderem garantir o equilíbrio estático da estrutura humanóide, podem fazê-lo de uma forma dinâmica pela variação de um parâmetro referência que se relaciona com a projecção do centro de massa do sistema sobre o solo.
Posteriormente também se fará um estudo sobre a utilidade destes sensores para a perna livre, de modo a ser possível detectar o impacto do respectivo pé sobre o solo, ou a sua elevação.

1. INTRODUÇÃO

1.1. Os sensores de Força

Para o *sensing* das forças aplicadas sobre os pés do robot humanóide, foram usados sensores de força baseados em extensómetros resistivos, cuja resistência varia de acordo com a sua deformação.

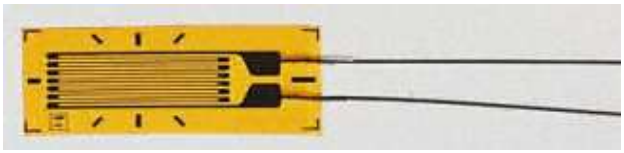


Fig. 24: Extensómetro resistivo.

1.2. Estrutura do Pé

Aproveitando esta propriedade elástica que relaciona proporcionalmente a deformação com a força aplicada, podemos colocar vários extensómetros ao longo da área de cada pé, e, pela medição da sua deformação, medir a força aplicada sobre cada um deles. Seleccionou-se a utilização de quatro sensores, um em cada canto, de modo a permitir uma estimativa bidimensional do centro de pressão aplicado sobre o pé.
A montagem é apresentada na Fig. 25, e baseia-se na colagem de cada extensómetro a uma placa de acrílico com boas propriedades elásticas (Fig. 27), em que, por sua vez, são colocados em várias aberturas de uma plataforma que permitem a sua livre deformação aquando da aplicação de uma força sobre elas.

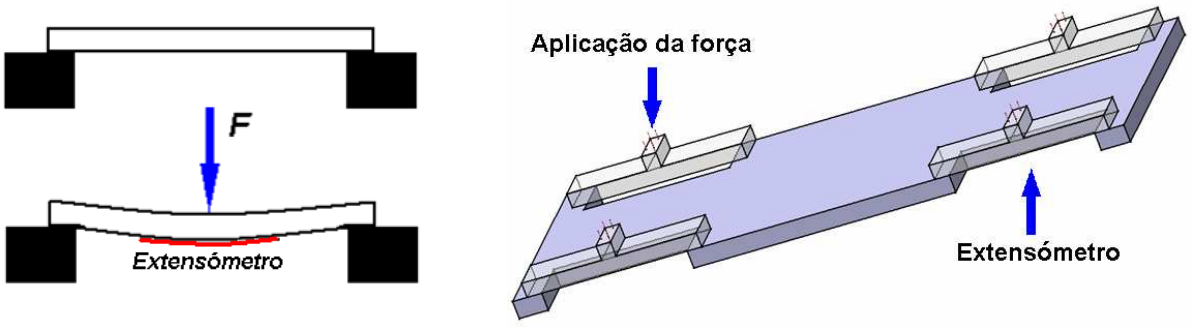


Fig. 25: Montagem dos extensómetros na estrutura do pé.

A aplicação da força é feita através de uma segunda plataforma colocada superiormente com vários parafusos que são aplicados directamente sobre as placas de acrílico. Uma vista completa do pé pode ser visualizada na Fig. 26, e mais em pormenor os pontos de contacto sobre os extensómetros na Fig. 28.

Fig. 26: Visão completa da base do pé.

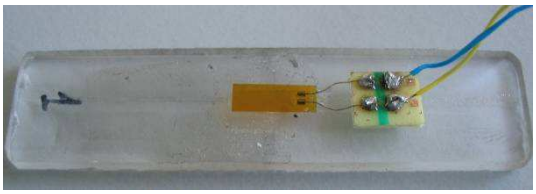


Fig. 27: Peça de acrílico contendo o extensómetro para medição da sua deformação.

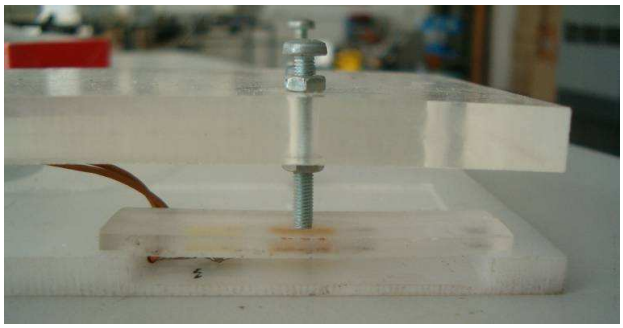


Fig. 28: Pontos de contacto entre as 2 plataformas do pé.

Sobre esta segunda plataforma está assente a estrutura da perna, que pela variação da projecção do seu centro de massa, fazem com que uns sensores se deformem mais do que outros. As relações entre as saídas destes quatro sensores fornecem-nos uma estimativa da aplicação do centro de pressão, que por sua vez será utilizado para fazer compensação em ordem a manter o equilíbrio da estrutura.

1.3. Condicionamento de Sinal

Como as variações de força, se traduzem em variações óhmicas dos extensómetros, é necessária electrónica de condicionamento, de modo a converter este sinal para uma tensão, e a amplificá-la o suficiente para que seja útil para o microcontrolador.

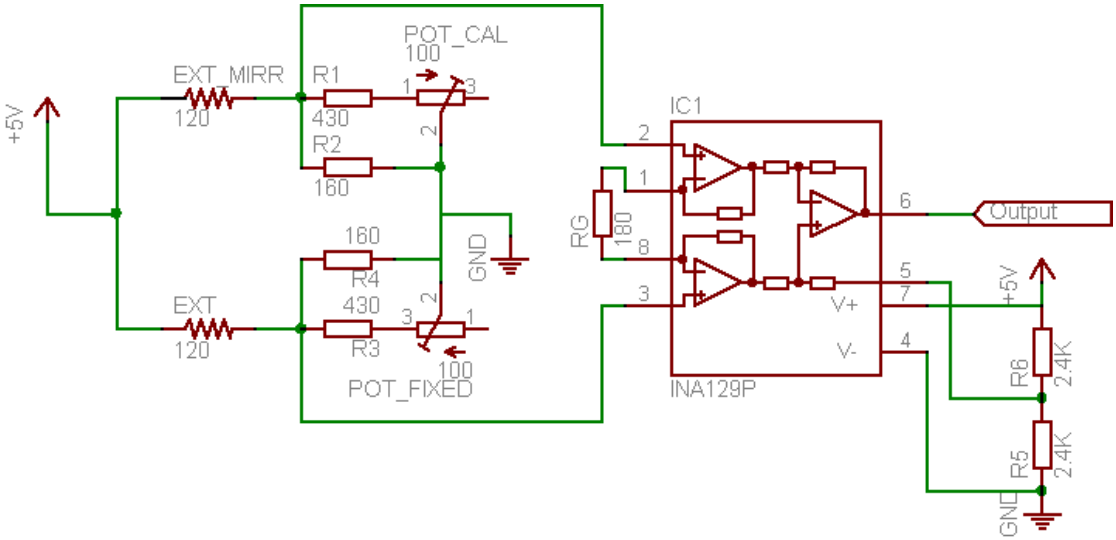


Fig. 29: Circuito de condicionamento do sinal proveniente de um extensómetro (EXT).

Este circuito, é aplicado sobre cada extensómetro e pode ser dividido em dois andares. O primeiro, composto pela ponte de Wheatstone, pretende converter as variações óhmicas do extensómetro a deformar num sinal eléctrico (tensão), e finalmente o amplificador instrumental para amplificação dos pequenos sinais presentes à saída da ponte.

A ponte, composta por dois ramos, inicialmente pensou-se que possuiria num dos ramos o extensómetro a ser medido (*EXT*) em série com uma resistência fixa, e o segundo ramo teria duas resistências, de valores iguais aos do primeiro, mas uma delas variável para fazer a compensação do ponto de equilíbrio (quando nenhuma força é aplicada). Neste ponto, a tensão nos pontos intermédios de cada ramo deve ser igual, resultando numa diferença de tensão nula à entrada do amplificador.

Contudo verificou-se que possuía demasiadas assimetrias entre os dois ramos, observando-se variações na saída ao longo do tempo. A verdade é que cada componente apresentava uma relação óhmica com a temperatura diferente, fazendo com a corrente que o atravessava variasse de forma não constante entre eles, e por isso, a observação das variações na saída.

Para resolver este problema, atribuiu-se simetria ao circuito, ou seja, em ambos os ramos teríamos um extensómetro e uma resistência variável de propriedades semelhantes: no primeiro seria com o extensómetro a ser medido (*EXT*) em série com um potenciómetro que não se pretende variar uma vez regulado inicialmente (*POT_FIXED*); e o segundo teria um extensómetro "espelho" (*EXT_MIRR*) localizado num ponto o mais próximo possível do primeiro, de modo a que as variações de temperatura os afectem de forma semelhante, mas sem a possibilidade de aplicação de qualquer força, este em série com o potenciómetro a ser usado para calibração do ponto de equilíbrio (*POT_CAL*).
Como as variações de resistência dos extensómetros são muito pequenas (na ordem dos 4% do valor nominal: 120Ω), e pretendia-se uma calibração muito fina, utilizaram-se potenciómetros multi-volta de 100 Ω que em série com a resistência R_S, e em paralelo com R_P, provocam uma variação efectiva na ordem dos 5Ω em torno do valor nominal (Fig. 30).

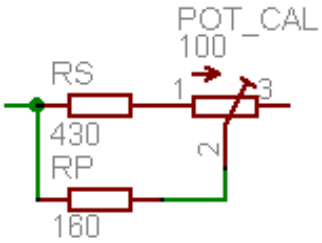


Fig. 30: Circuito de calibração da ponte de Wheatstone.

Finalmente o amplificador instrumental é o INA129P que pela configuração actual, com R_G=180Ω, possibilita a amplificação do sinal de entrada em 65V/V. Desta forma, variações de 60.98 mV à saída da ponte de *Wheatstone*, transformam-se num sinal com uma gama de 4V passível de ser utilizada pela ADC do microcontrolador com as tensões de referência 0-5V.

Uma visão global da perna assente sobre o pé sensível a forças, com a respectiva electrónica de condicionamento, pode ser visualizada na Fig. 31.

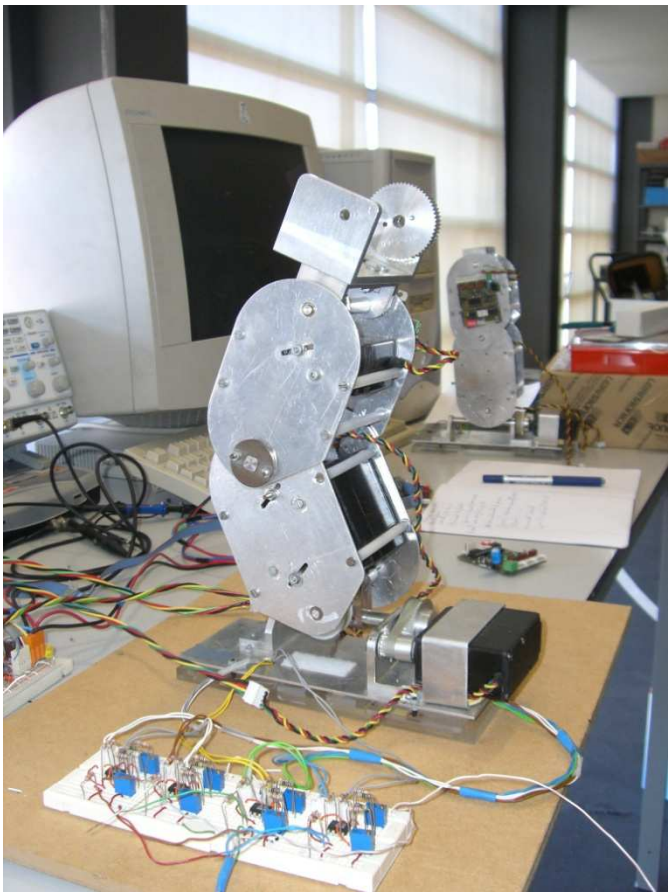


Fig. 31: Perna completa sobre um pé sensível a forças.

1.4. Processamento do Sinal pelo Microcontrolador

Finalmente, a tensão presente à saída do amplificador é entregue à ADC do microprocessador que, uma vez digitalizada, é utilizada pelos algoritmos de compensação de força para ajuste de equilíbrio.

No entanto, antes de tudo, é necessário efectuar uma segunda calibração, de modo a sinalizar que a tensão presente na entrada da ADC corresponde a uma determinada postura referência da perna.
Já foi referido que, considerando o uso das tensões referência de 0/5V, o sinal eléctrico à saída do amplificador apresenta uma gama inferior a 5V, tendo em conta um amplificador *non rail-to-rail*; por isso é necessário convencionar uma determinada tensão para a posição vertical; como por exemplo, se definirmos esta tensão como 1V, variações abaixo deste valor reflectem que o centro de pressão se deslocou para a direcção oposta ao sensor medido, mas acima deste, reflectem o seu deslocamento na sua direcção.
Para que este sistema seja independente deste valor escolhido, efectua-se inicialmente uma calibração por *software*, medindo a tensão presente à saída e convencionando-se este valor como correspondendo à postura vertical da perna. Por isso, é importante que quando o sistema é iniciado, que a perna esteja na chamada posição de equilíbrio, para que os algoritmos de controlo funcionem como o desejado.

Finalmente podemos efectuar leituras da força aplicada sobre cada sensor. O valor presente à saída de cada amplificador é digitalizado através de uma ADC de 8 bits, apresentando como resultado um valor numérico entre 0 e 255. Como podemos obter medições acima e abaixo do ponto de equilíbrio (postura vertical da perna), mas sempre representando forças positivas (as placas de acrílico deformam-se sempre para um único sentido: para baixo), convencionou-se a representação da força sempre como um valor positivo, mas em torno de um ponto intermédio. Este ponto foi definido como o 128, de forma que:

- Uma medição com o valor 128 corresponde à postura de equilíbrio;
- Medições acima de 128 (129-255), correspondem ao deslocamento do centro de pressão no sentido do sensor medido;
- Medições abaixo de 128 (0-127), correspondem ao afastamento do centro de pressão relativamente ao sensor medido.

Note que o valor de 128 corresponde à postura de equilíbrio, pelo que após a calibração por *software*, deveremos obter este valor independentemente da tensão presente à saída do amplificador. Desta forma, o resultado da força medida corresponde a efectuar o seguinte cálculo, em que $Sensor_{\text{output}}$ corresponde à saída do sensor no momento, e $Sensor_{\text{calib}}$ também à saída do mesmo, mas no momento da calibração:

$$Força = Sensor_{output} - Sensor_{calib} + 128$$

Repare, que após a calibração, $Sensor_{\text{output}}$ e $Sensor_{\text{calib}}$ são iguais, resultando numa força igual a 128! Para que obtenhamos uma força compreendida entre 0 e 255, passível de ser armazenada numa variável de 8 bits, ao valor proveniente da ADC (ADC_{output}) é removido o bit menos significativo, modificando a sua gama de 8 bits para apenas 7, fazendo com que a diferença ($Sensor_{\text{output}} - Sensor_{\text{calib}}$) se situe entre –127 e +127, resultando numa gama de 1 a 255 no resultado final da força.

$$Sensor_{output} = ADC_{output} >> 1 \quad (\text{Right Shift})$$

Resumindo, o processo de leitura da força aplicado segue o seguinte processo:

1. Com a aplicação de uma força, a placa de acrílico em conjunto com o extensómetro colado, deformam-se;
2. A resistência óhmica do extensómetro varia em torno do valor nominal (120Ω);
3. A ponte de Wheatstone, previamente calibrada (pelo potenciómetro), regista a variação pelo desequilíbrio de corrente que circula entre os dois ramos, resultando numa variação da tensão à saída;
4. O amplificador instrumental amplifica esta tensão;
5. A ADC do microcontrolador digitaliza esta tensão num valor de 8 bits (0-255);
6. O valor estimado da força aplicada, é determinado com base numa calibração inicial (por *software*) presumindo a postura vertical da perna nessa circunstância.

Como se pode constatar, a força estimada não corresponde ao seu valor absoluto, dado que ele depende de inúmeros factores:

- Desgaste na deformação do acrílico (resultante das quebras nas ligações poliméricas);
- Precisão das extensómetros e resistências envolvidas na ponte de Wheatstone: pode variar para os diversos sensores;
- Precisão da ADC;
- Calibração física sobre o potenciómetro;
- Calibração por *software*.

Embora os motivos resultantes de desgaste e precisão, possam ser resolvidos pela adopção de materiais e componentes de boa qualidade, o problema agrava-se no que respeita aos procedimentos de calibração. Como se sabe, no momento da inicialização do sistema, os sensores de força dos pés já possuem uma determinada força aplicada sobre eles, resultante da massa da estrutura superior do pé, da perna e do restante corpo, se presente. Estas condições inicias são desconhecidas, e ao fixar este valor em 128 está-se a pressupor que um acréscimo de 10 unidades na força estimada corresponde a um acréscimo de 7.8% na força absoluta, o que pode não ser verdadeiro. Pelo contrário, se definirmos o valor referência como 50, o mesmo acréscimo de 10 unidades já corresponde a um acréscimo de 20% na força absoluta. Tudo depende da sensibilidade das placas de acrílico e do peso aplicado inicialmente. Embora a sensibilidade seja praticamente constante entre as várias placas, no que respeita ao peso inicial tal já não é verdade: repare que com a postura vertical da perna, mais peso é aplicado no par de sensores traseiros dada a presença de um servomotor nessa posição. O mesmo também se verifica relativamente a um dos pares de sensores laterais, devido à assimetria do eixo de rotação lateral do pé em relação ao centro do mesmo. Ou seja, dois problemas surgem:

- O valor inicial da força absoluta é desconhecido;
- E a força absoluta inicial difere de sensor para sensor.

A resolução do problema da força inicial, por enquanto é baseada em métodos empíricos, pela aplicação de um ganho sobre o controlador que utiliza estes valores. Relativamente à variação de sensor para sensor, ainda não se tornou suficientemente importante quando usando apenas a perna, pois o seu reduzido peso torna estas variações desprezáveis. Contudo, experiências subsequentes com cargas aplicadas sobre a perna, tornaram este problema evidente, requerendo a aplicação de ganhos de amplificação independentes para cada sensor.

1.5. Cálculo do Centro de Pressão (CoP)

Com a força estimada sobre cada sensor, só nos falta determinar a projecção do centro de massa (CoM) da estrutura humanóide sobre cada pé. Para velocidades relativamente reduzidas, tal corresponde a determinar o centro de pressão (CoP) com base nos quatro sensores de força. Este ponto está representado na Fig. 32, e para a siituação de equilíbrio (perna vertical), tal deve corresponder às coordenadas (x,y)=(0,0), ou seja, o centro da base do pé.

Medindo as coordenadas de cada sensor relativamente ao centro do pé:

$$\vec{d}_1 = (x_1, y_1) \qquad \vec{d}_2 = (x_2, y_2)$$
$$\vec{d}_3 = (x_3, y_3) \qquad \vec{d}_4 = (x_4, y_4)$$

Podemos determinar as coordenadas do centro de pressão, através do peso da força individual de cada sensor:

$$\vec{CoP} = \frac{\sum_{i=0}^{4} \left(F_i * \vec{d}_i\right)}{\sum_{i=0}^{4} F_i}$$

$$CoP_x = \frac{\sum_{i=0}^{4} (F_i * x_i)}{\sum_{i=0}^{4} F_i} \qquad CoP_y = \frac{\sum_{i=0}^{4} (F_i * y_i)}{\sum_{i=0}^{4} F_i}$$

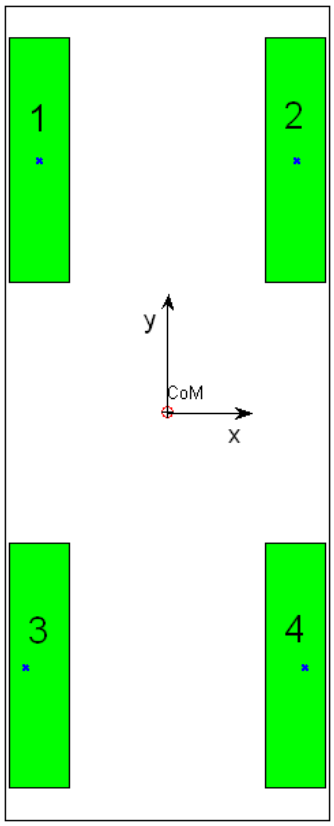


Fig. 32: Projecção do Centro de Massa na situação de equilíbrio.

Contudo, como já foi referido anteriormente, a força inicial de cada sensor é desconhecida, o que, se ignorada, afectará também o valor do centro de pressão de uma forma não sistemática:

$$\vec{CoP} = \frac{\sum_{i=0}^{4}\left(F_i * \vec{d}_i\right)}{\sum_{i=0}^{4} F_i} = \frac{\sum_{i=0}^{4}\left[\left(F_i^{med} + F_i^{0}\right) * \vec{d}_i\right]}{\sum_{i=0}^{4}\left(F_i^{med} + F_i^{0}\right)} = \frac{\sum_{i=0}^{4}\left(F_i^{med} * \vec{d}_i\right) + \sum_{i=0}^{4}\left(F_i^{0} * \vec{d}_i\right)}{\sum_{i=0}^{4} F_i^{med} + \sum_{i=0}^{4} F_i^{0}}$$

Tendo em conta que a estimação da força já inclui a adição de um *offset* (128) de forma a prever esta força inicial, basta a seguinte expressão para estimar o CoP:

$$\vec{CoP} = \frac{\sum_{i=0}^{4}\left(F_i^{med} * \vec{d}_i\right)}{\sum_{i=0}^{4} F_i^{med}}$$

Contudo, dado o formato não simétrico dos pés (rectangular), a medição do CoP na componente *xx* possuirá maior sensibilidade que na componente ortogonal *yy*. Para compensar esta assimetria, adicionou-se um segundo *offset* (subtractivo) à força medida em cada sensor de modo que quanto maior for o seu valor maior será a sensibilidade atribuída. Este offset está expresso nas equações seguintes, possuindo um valor superior para a componente *yy*:

$$CoP_x = \frac{\sum_{i=0}^{4}\left[\left(F_i^{med} - offset_x\right) * x_i\right]}{\sum_{i=0}^{4}\left(F_i^{med} - offset_x\right)} \qquad CoP_y = \frac{\sum_{i=0}^{4}\left[\left(F_i^{med} - offset_y\right) * y_i\right]}{\sum_{i=0}^{4}\left(F_i^{med} - offset_y\right)}$$

Por observação experimental, convencionou-se $offset_x$=0 e $offset_y$=50 – dado que a força normalizada é 128, os *offsets* devem ser inferiores a 128 de modo a evitar resultados negativos.

2. O CONTROLADOR DE EQUILÍBRIO

2.1. Introdução

Uma vez que já possuímos um meio de calcular o centro de pressão em cada instante, apenas nos falta apresentar o controlador de equilíbrio que tem como função o movimento das juntas com o objectivo de atingir um determinado CoP referência, que tanto pode ser o centro do pé, se apenas pretendemos garantir a postura vertical, ou então outros valores de modo a conferir alguma dinâmica ao sistema.

A Fig. 33 apresenta um diagrama com a descrição dos controladores envolvidos. Após a leitura dos sensores de força, o cálculo do Centro de Pressão é executado, que por comparação com o Centro de Pressão desejado (CoP_{ref}), a variação de posição a aplicar a cada junta do pé e da perna é determinado. Esta variação v(n) é adicionada à posição actual de cada junta obtendo assim o valor absoluto da posição u(n). Finalmente o controlador local é usado para assegurar a aplicação da posição calculada.

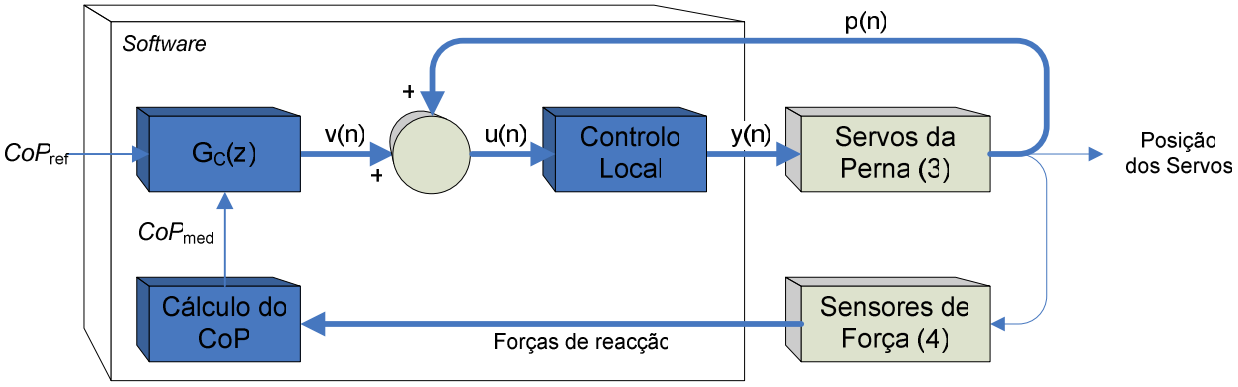


Fig. 33: Diagrama com os controladores aplicados para a compensação de equilíbrio.

2.2. Controlo do Centro de Pressão

De modo a permitir a utilização de todas as juntas da perna, nomeadamente a ortogonal e a paralela ao pé, e a do joelho, utilizou-se a matriz Jacobiana para o efeito, que além dessa possibilidade também permite um maior rigor no cálculo da variação angular a aplicar (transformação das coordenadas cartesianas do CoP para coordenadas angulares dos servos).

Sabendo que a matriz Jacobiana corresponde à derivada do centro de massa em função das posições angulares dos três servos:

$$\vec{J} = \frac{\partial \vec{CoM}}{\partial \vec{\theta}} = \begin{bmatrix} \frac{\partial CoM_x}{\partial \theta_1} & \frac{\partial CoM_x}{\partial \theta_2} & \frac{\partial CoM_x}{\partial \theta_3} \\ \frac{\partial CoM_y}{\partial \theta_1} & \frac{\partial CoM_y}{\partial \theta_2} & \frac{\partial CoM_y}{\partial \theta_3} \\ \frac{\partial CoM_z}{\partial \theta_1} & \frac{\partial CoM_z}{\partial \theta_2} & \frac{\partial CoM_z}{\partial \theta_3} \end{bmatrix} \Rightarrow \partial\vec{\theta} = \vec{J}^{-1} \times \partial\vec{CoM}$$

Podemos associar a variação angular a aplicar a cada um deles, com base no erro do centro de massa, em que ***K*** corresponde a um ganho a aplicar às variações determinadas:

$$\Delta\vec{\theta} = \vec{K} \cdot \left(\vec{J}^{-1} \times \vec{e}_{CoM}\right)$$

Por simplificação, em vez da inversa do Jacobiano e do erro do Centro de Massa, podemos usar o Jacobiano transposto e o erro do Centro de Pressão respectivamente, dado o último ser bastante próximo do centro de massa para baixas velocidades:

$$\Delta\vec{\theta} = \vec{K}\cdot\left(\vec{J}^{T}\times\vec{e}_{CoP}\right) = \vec{K}\cdot\left(\vec{J}^{T}\times\left(\vec{e}_{CoP}^{ref}-\vec{e}_{CoP}^{med}\right)\right)$$

Expandindo os parâmetros vectoriais, obtém-se:

$$\begin{bmatrix}\Delta\theta_1\\ \Delta\theta_2\\ \Delta\theta_3\end{bmatrix} = \begin{bmatrix}K_1\\ K_2\\ K_3\end{bmatrix}\cdot\left(\begin{bmatrix}\frac{\partial CoM_x}{\partial\theta_1} & \frac{\partial CoM_y}{\partial\theta_1} & \frac{\partial CoM_z}{\partial\theta_1}\\ \frac{\partial CoM_x}{\partial\theta_2} & \frac{\partial CoM_y}{\partial\theta_2} & \frac{\partial CoM_z}{\partial\theta_2}\\ \frac{\partial CoM_x}{\partial\theta_3} & \frac{\partial CoM_y}{\partial\theta_3} & \frac{\partial CoM_z}{\partial\theta_3}\end{bmatrix}\times\begin{bmatrix}CoP_x^{ref}-CoP_x^{med}\\ CoP_y^{ref}-CoP_y^{med}\\ CoP_z^{ref}-CoP_z^{med}\end{bmatrix}\right)$$

As variáveis θ_i e K_i estão indicadas no modelo da Fig. 34 descritivo da perna e do pé humanóide.

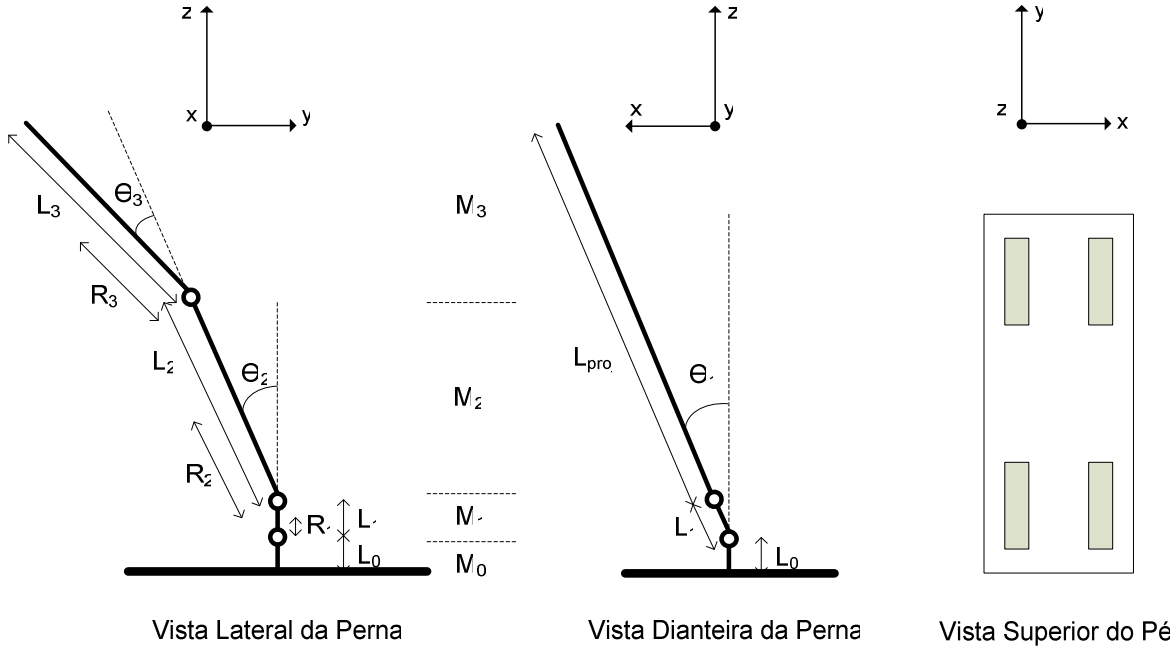


Fig. 34: Modelo da perna humanóide com a associação das diversas variáveis de interesse.

Para cálculo da matriz Jacobiana, seja:

$$\begin{cases}s_1 = \sin\theta_1\\ s_2 = \sin\theta_2\\ s_{23} = \sin(\theta_2+\theta_3)\end{cases}\qquad\qquad\begin{cases}c_1 = \cos\theta_1\\ c_2 = \cos\theta_2\\ c_{23} = \cos(\theta_2+\theta_3)\end{cases}$$

O Jacobiano está relacionado com o Centro de Massa, que pela Fig. 34 pode ser deduzido como:

$$\vec{CoM} = \begin{cases}CoM_x = \\ \\ \\ \end{cases}$$

$$\overrightarrow{CoM}:\begin{cases} CoM_x = \dfrac{M_1R_1s_1 + M_2(L_1 + R_2c_2)s_1 + M_3(L_1 + L_2c_2 + R_3c_{23})s_1}{M_1 + M_2 + M_3} \\ CoM_y = \dfrac{M_2R_2s_2 + M_3(L_2s_2 + R_3s_{23})}{M_2 + M_3} \\ CoM_z = \dfrac{M_0R_0 + M_1(L_0 + R_1c_1) + M_2[L_0 + (L_1 + R_2c_2)c_1] + M_3[L_0 + (L_1 + L_2c_2 + R_3c_{23})]c_1}{M_0 + M_1 + M_2 + M_3} \end{cases}$$

Calculando as suas derivadas parciais em ordem a θ_1, θ_2 e θ_3 obtém-se:

$$\frac{\partial\overrightarrow{CoM}}{\partial\vec{\theta}}:\begin{cases} \dfrac{\partial CoM_x}{\partial\theta_1} = \dfrac{M_1R_1c_1 + M_2(L_1 + R_2c_2)c_1 + M_3(L_1 + L_2c_2 + R_3c_{23})c_1}{M_1 + M_2 + M_3} \\ \dfrac{\partial CoM_x}{\partial\theta_2} = \dfrac{-M_2R_2s_2s_1 - M_3(L_2s_2 + R_3s_{23})s_1}{M_1 + M_2 + M_3} \\ \dfrac{\partial CoM_x}{\partial\theta_3} = \dfrac{-M_3R_3s_{23}s_1}{M_1 + M_2 + M_3} \\ \dfrac{\partial CoM_y}{\partial\theta_1} = 0 \\ \dfrac{\partial CoM_y}{\partial\theta_2} = \dfrac{-M_2R_2c_2 - M_3(L_2c_2 + R_3c_{23})}{M_2 + M_3} \\ \dfrac{\partial CoM_y}{\partial\theta_3} = \dfrac{M_3R_3c_{23}}{M_2 + M_3} \\ \dfrac{\partial CoM_z}{\partial\theta_1} = \dfrac{-M_1R_1s_1 - M_2(L_1 + R_2c_2)s_1 - M_3(L_1 + L_2c_2 + R_3c_{23})s_1}{M_0 + M_1 + M_2 + M_3} \\ \dfrac{\partial CoM_z}{\partial\theta_2} = \dfrac{-M_2R_2s_2c_1 - M_3(L_2s_2 + R_3s_{23})c_1}{M_0 + M_1 + M_2 + M_3} \\ \dfrac{\partial CoM_z}{\partial\theta_3} = \dfrac{-M_3R_3s_{23}c_1}{M_0 + M_1 + M_2 + M_3} \end{cases}$$

De modo a simplificar estas expressões consideremos que $M_0 \approx M_1 \approx 0$ e que $L_1 \approx 0$:

$$\frac{\partial\overrightarrow{CoM}}{\partial\vec{\theta}}:\begin{cases} \dfrac{\partial CoM_x}{\partial\theta_1} = \dfrac{M_2R_2c_2 + M_3(L_2c_2 + R_3c_{23})}{M_2 + M_3} \cdot c_1 \\ \dfrac{\partial CoM_x}{\partial\theta_2} = -\dfrac{M_2R_2s_2 + M_3(L_2s_2 + R_3s_{23})}{M_2 + M_3} \cdot s_1 \\ \dfrac{\partial CoM_x}{\partial\theta_3} = -\dfrac{M_3R_3s_{23}}{M_2 + M_3} \cdot s_1 \\ \dfrac{\partial CoM_y}{\partial\theta_1} = 0 \\ \dfrac{\partial CoM_y}{\partial\theta_2} = -\dfrac{M_2R_2c_2 + M_3(L_2c_2 + R_3c_{23})}{M_2 + M_3} \\ \dfrac{\partial CoM_y}{\partial\theta_3} = \dfrac{M_3R_3c_{23}}{M_2 + M_3} \\ \dfrac{\partial CoM_z}{\partial\theta_1} = -\dfrac{M_2R_2c_2 + M_3(L_2c_2 + R_3c_{23})}{M_2 + M_3} \cdot s_1 \\ \dfrac{\partial CoM_z}{\partial\theta_2} = -\dfrac{M_2R_2s_2 + M_3(L_2s_2 + R_3s_{23})}{M_2 + M_3} \cdot c_1 \\ \dfrac{\partial CoM_z}{\partial\theta_3} = -\dfrac{M_3R_3s_{23}}{M_2 + M_3} \cdot c_1 \end{cases}$$

Fazendo com que:

$$S_{23} = \frac{M_3 R_3 s_{23}}{M_2 + M_3} \qquad SS_{23} = \frac{M_2 R_2 s_2 + M_3 (L_2 s_2 + R_3 s_{23})}{M_2 + M_3}$$
$$C_{23} = \frac{M_3 R_3 c_{23}}{M_2 + M_3} \qquad CC_{23} = \frac{M_2 R_2 c_2 + M_3 (L_2 c_2 + R_3 c_{23})}{M_2 + M_3}$$

Então as diversas derivadas parciais podem ser descritas de uma forma mais simples:

$$\frac{\partial \overrightarrow{CoM}}{\partial \vec{\theta}} : \left\{ \begin{array}{|l|} \hline \frac{\partial CoM_x}{\partial \theta_1} = CC_{23} \cdot c_1 \\ \frac{\partial CoM_x}{\partial \theta_2} = -SS_{23} \cdot s_1 \\ \frac{\partial CoM_x}{\partial \theta_3} = -S_{23} \cdot s_1 \\ \hline \end{array} \right. \left\{ \begin{array}{|l|} \hline \frac{\partial CoM_y}{\partial \theta_1} = 0 \\ \frac{\partial CoM_y}{\partial \theta_2} = -CC_{23} \\ \frac{\partial CoM_y}{\partial \theta_3} = C_{23} \\ \hline \end{array} \right. \left\{ \begin{array}{|l|} \hline \frac{\partial CoM_z}{\partial \theta_1} = -CC_{23} \cdot s_1 \\ \frac{\partial CoM_z}{\partial \theta_2} = -SS_{23} \cdot c_1 \\ \frac{\partial CoM_z}{\partial \theta_3} = -S_{23} \cdot c_1 \\ \hline \end{array} \right\}$$

Desta forma, a matriz Jacobiana é dada por:

$$\vec{J} = \frac{\partial \overrightarrow{CoM}}{\partial \vec{\theta}} = \begin{bmatrix} \frac{\partial CoM_x}{\partial \theta_1} & \frac{\partial CoM_x}{\partial \theta_2} & \frac{\partial CoM_x}{\partial \theta_3} \\ \frac{\partial CoM_y}{\partial \theta_1} & \frac{\partial CoM_y}{\partial \theta_2} & \frac{\partial CoM_y}{\partial \theta_3} \\ \frac{\partial CoM_z}{\partial \theta_1} & \frac{\partial CoM_z}{\partial \theta_2} & \frac{\partial CoM_z}{\partial \theta_3} \end{bmatrix} = \begin{bmatrix} J_{11} & J_{12} & J_{13} \\ J_{21} & J_{22} & J_{23} \\ J_{31} & J_{32} & J_{33} \end{bmatrix} = \begin{bmatrix} CC_{23} \cdot c_1 & -SS_{23} \cdot s_1 & -S_{23} \cdot s_1 \\ 0 & -CC_{23} & C_{23} \\ -CC_{23} \cdot s_1 & -SS_{23} \cdot c_1 & -S_{23} \cdot c_1 \end{bmatrix}$$

E a sua transposta por:

$$\vec{J}^T = \left(\frac{\partial \overrightarrow{CoM}}{\partial \vec{\theta}} \right)^T = \begin{bmatrix} \frac{\partial CoM_x}{\partial \theta_1} & \frac{\partial CoM_y}{\partial \theta_1} & \frac{\partial CoM_z}{\partial \theta_1} \\ \frac{\partial CoM_x}{\partial \theta_2} & \frac{\partial CoM_y}{\partial \theta_2} & \frac{\partial CoM_z}{\partial \theta_2} \\ \frac{\partial CoM_x}{\partial \theta_3} & \frac{\partial CoM_y}{\partial \theta_3} & \frac{\partial CoM_z}{\partial \theta_3} \end{bmatrix} = \begin{bmatrix} J_{11} & J_{21} & J_{31} \\ J_{12} & J_{22} & J_{32} \\ J_{13} & J_{23} & J_{33} \end{bmatrix} = \begin{bmatrix} CC_{23} \cdot c_1 & 0 & -CC_{23} \cdot s_1 \\ -SS_{23} \cdot s_1 & -CC_{23} & -SS_{23} \cdot c_1 \\ -S_{23} \cdot s_1 & C_{23} & -S_{23} \cdot c_1 \end{bmatrix}$$

Concluindo, as variações angulares a aplicar são determinadas da seguinte forma:

$$\begin{bmatrix} \Delta\theta_1 \\ \Delta\theta_2 \\ \Delta\theta_3 \end{bmatrix} = \begin{bmatrix} K_1 \\ K_2 \\ K_3 \end{bmatrix} \cdot \left(\begin{bmatrix} CC_{23} \cdot c_1 & 0 & -CC_{23} \cdot s_1 \\ -SS_{23} \cdot s_1 & -CC_{23} & -SS_{23} \cdot c_1 \\ -S_{23} \cdot s_1 & C_{23} & -S_{23} \cdot c_1 \end{bmatrix} \times \begin{bmatrix} CoP_x^{ref} - CoP_x^{med} \\ CoP_y^{ref} - CoP_y^{med} \\ CoP_z^{ref} - CoP_z^{med} \end{bmatrix} \right)$$

$$\begin{bmatrix} \Delta\theta_1 \\ \Delta\theta_2 \\ \Delta\theta_3 \end{bmatrix} = \begin{bmatrix} K_1 \\ K_2 \\ K_3 \end{bmatrix} \cdot \left(\begin{bmatrix} J_{11} & J_{12} & J_{13} \\ J_{21} & J_{22} & J_{23} \\ J_{31} & J_{32} & J_{33} \end{bmatrix} \times \begin{bmatrix} e_{CoPx} \\ e_{CoPy} \\ e_{CoPz} \end{bmatrix} \right)$$

$$\begin{cases} \Delta\theta_1 = K_1 \cdot (J_{11} \cdot e_{CoPx} + J_{12} \cdot e_{CoPy} + J_{13} \cdot e_{CoPz}) \\ \Delta\theta_2 = K_2 \cdot (J_{21} \cdot e_{CoPx} + J_{22} \cdot e_{CoPy} + J_{23} \cdot e_{CoPz}) \\ \Delta\theta_3 = K_3 \cdot (J_{31} \cdot e_{CoPx} + J_{32} \cdot e_{CoPy} + J_{33} \cdot e_{CoPz}) \end{cases}$$

Dado que o erro do CoP na componente *zz* não tem significado, este parâmetro não será considerado, pelo que:

$$\begin{bmatrix} \Delta\theta_1 \\ \Delta\theta_2 \\ \Delta\theta_3 \end{bmatrix} = \begin{bmatrix} K_1 \\ K_2 \\ K_3 \end{bmatrix} \cdot \left(\begin{bmatrix} J_{11} & J_{12} & J_{13} \\ J_{21} & J_{22} & J_{23} \\ J_{31} & J_{32} & J_{33} \end{bmatrix} \times \begin{bmatrix} e_{CoPx} \\ e_{CoPy} \\ 0 \end{bmatrix} \right)$$

$$\begin{cases} \Delta\theta_1 = K_1 \cdot (J_{11} \cdot e_{CoPx} + J_{12} \cdot e_{CoPy}) \\ \Delta\theta_2 = K_2 \cdot (J_{21} \cdot e_{CoPx} + J_{22} \cdot e_{CoPy}) \\ \Delta\theta_3 = K_3 \cdot (J_{31} \cdot e_{CoPx} + J_{32} \cdot e_{CoPy}) \end{cases}$$

Pela análise a estas expressões, a terceira coluna do Jacobiano transposto deixa de ser utilizada, pelo que a matriz é suficiente na forma:

$$\vec{J}^T = \begin{bmatrix} J_{11} & J_{21} & 0 \\ J_{12} & J_{22} & 0 \\ J_{13} & J_{23} & 0 \end{bmatrix} = \begin{bmatrix} CC_{23} \cdot c_1 & 0 & 0 \\ -SS_{23} \cdot s_1 & -CC_{23} & 0 \\ -S_{23} \cdot s_1 & C_{23} & 0 \end{bmatrix}$$

2.3. Controlo de Altura

No entanto, além do centro de pressão, torna-se igualmente importante o controlo da altura da anca, para efeitos de estabilização da cintura ao longo dos processos de locomoção. Por isso, para as coordenadas *x* e *y* seria utilizado o controlador de CoP, enquanto para a coordenada *z* utilizaríamos um controlador de altura baseado na cinemática directa.

Sendo assim, a altura da anca é dada por:

$$h = L_0 + (L_1 + L_2 c_2 + L_3 c_{23}) \cdot c_1$$

Utilizando igualmente uma matriz Jacobiana, mas baseada na cinemática directa, podemos determinar a variação angular a aplicar com base na altura h:

$$\Delta\vec{\theta} = \vec{K} \cdot (\vec{J}^T \times e_h)$$

$$\begin{bmatrix} \Delta\theta_1 \\ \Delta\theta_2 \\ \Delta\theta_3 \end{bmatrix} = \begin{bmatrix} K_1 \\ K_2 \\ K_3 \end{bmatrix} \cdot \left(\begin{bmatrix} \frac{\partial h}{\partial \theta_1} \\ \frac{\partial h}{\partial \theta_2} \\ \frac{\partial h}{\partial \theta_3} \end{bmatrix} \times \left[h^{ref} - h^{med} \right] \right)$$

Dado que a altura corresponde a uma das coordenadas cartesianas da anca, podemos expandir estas matrizes de modo a conciliá-la, em termos de dimensão, com as matrizes anteriores:

$$\begin{bmatrix} \Delta\theta_1 \\ \Delta\theta_2 \\ \Delta\theta_3 \end{bmatrix} = \begin{bmatrix} K_1 \\ K_2 \\ K_3 \end{bmatrix} \cdot \left(\begin{bmatrix} 0 & 0 & \frac{\partial h}{\partial \theta_1} \\ 0 & 0 & \frac{\partial h}{\partial \theta_2} \\ 0 & 0 & \frac{\partial h}{\partial \theta_3} \end{bmatrix} \times \begin{bmatrix} 0 \\ 0 \\ h^{ref} - h^{med} \end{bmatrix} \right)$$

Nesta forma, facilmente se percebe que a matriz Jacobiana da cinemática directa pode ser combinada com a da do centro de massa:

$$\vec{J}^T = \vec{J}_{CoP}^T + \vec{J}_h^T = \left(\frac{\partial \overrightarrow{CoM}}{\partial \vec{\theta}} \right)^T + \left(\frac{\partial \vec{h}}{\partial \vec{\theta}} \right)^T = \begin{bmatrix} \frac{\partial CoM_x}{\partial \theta_1} & \frac{\partial CoM_y}{\partial \theta_1} & 0 \\ \frac{\partial CoM_x}{\partial \theta_2} & \frac{\partial CoM_y}{\partial \theta_2} & 0 \\ \frac{\partial CoM_x}{\partial \theta_3} & \frac{\partial CoM_y}{\partial \theta_3} & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 & \frac{\partial h}{\partial \theta_1} \\ 0 & 0 & \frac{\partial h}{\partial \theta_2} \\ 0 & 0 & \frac{\partial h}{\partial \theta_3} \end{bmatrix}$$

$$\vec{J}^T = \begin{bmatrix} \frac{\partial CoM_x}{\partial \theta_1} & \frac{\partial CoM_y}{\partial \theta_1} & \frac{\partial h}{\partial \theta_1} \\ \frac{\partial CoM_x}{\partial \theta_2} & \frac{\partial CoM_y}{\partial \theta_2} & \frac{\partial h}{\partial \theta_2} \\ \frac{\partial CoM_x}{\partial \theta_3} & \frac{\partial CoM_y}{\partial \theta_3} & \frac{\partial h}{\partial \theta_3} \end{bmatrix}$$

Calculando as derivadas parciais da altura:

$$\begin{cases} \frac{\partial h}{\partial \theta_1} = -(L_1 + L_2 c_2 + L_3 c_{23}) \cdot s_1 \\ \frac{\partial h}{\partial \theta_2} = -(L_2 s_2 + L_3 s_{23}) \cdot c_1 \\ \frac{\partial h}{\partial \theta_3} = -L_3 \cdot s_{23} \cdot c_1 \end{cases}$$

E fazendo com que:

$$\begin{cases} CC'_{23} = L_1 + L_2 c_2 + L_3 c_{23} \\ SS'_{23} = L_2 s_2 + L_3 s_{23} \\ S'_{23} = L_3 s_{23} \end{cases}$$

Então:

$$\begin{cases} \frac{\partial h}{\partial \theta_1} = -CC'_{23} \cdot s_1 \\ \frac{\partial h}{\partial \theta_2} = -SS'_{23} \cdot c_1 \\ \frac{\partial h}{\partial \theta_3} = -S'_{23} \cdot c_1 \end{cases}$$

Assim sendo, a definitiva matriz Jacobiana passa a ser:

$$\vec{J}^T = \begin{bmatrix} J_{11} & J_{21} & J_{31} \\ J_{12} & J_{22} & J_{32} \\ J_{13} & J_{23} & J_{33} \end{bmatrix} = \begin{bmatrix} CC_{23} \cdot c_1 & 0 & -CC'_{23} \cdot s_1 \\ -SS_{23} \cdot s_1 & -CC_{23} & -SS'_{23} \cdot c_1 \\ -S_{23} \cdot s_1 & C_{23} & -S'_{23} \cdot c_1 \end{bmatrix}$$

$$\begin{bmatrix} \Delta\theta_1 \\ \Delta\theta_2 \\ \Delta\theta_3 \end{bmatrix} = \begin{bmatrix} K_1 \\ K_2 \\ K_3 \end{bmatrix} \cdot \left(\begin{bmatrix} J_{11} & J_{12} & J_{13} \\ J_{21} & J_{22} & J_{23} \\ J_{31} & J_{32} & J_{33} \end{bmatrix} \times \begin{bmatrix} e_{CoPx} \\ e_{CoPy} \\ e_h \end{bmatrix} \right)$$

Com as seguintes expressões para cálculo da variação angular dos servos:

$$\begin{aligned} \Delta\theta_1 &= K_1 \cdot (J_{11} \cdot e_{CoPx} + J_{12} \cdot e_{CoPy} + J_{13} \cdot e_h) \\ \Delta\theta_2 &= K_2 \cdot (J_{21} \cdot e_{CoPx} + J_{22} \cdot e_{CoPy} + J_{23} \cdot e_h) \\ \Delta\theta_3 &= K_3 \cdot (J_{31} \cdot e_{CoPx} + J_{32} \cdot e_{CoPy} + J_{33} \cdot e_h) \end{aligned}$$

Estas três equações, constituem assim a lei de controlo do centro de pressão e da altura, com base nos respectivos sinais de erro. O erro do centro de pressão é calculado a partir dos sensores de força, pela comparação entre o CoP desejado (CoP^{ref}) e o medido (CoP^{med}):

$$e_{CoP} = CoP^{ref} - CoP^{med}$$

Já o erro de altura é calculado pela comparação entre a altura desejada (h^{ref}) e a estimada através da cinemática directa das juntas (h^{med}):

$$h^{med} = L_0 + (L_1 + L_2 c_2 + L_3 c_{23}) \cdot c_1$$

$$e_h = h^{ref} - h^{med}$$

2.4. Estudos Experimentais

Estudos detalhados sobre os sensores de força, relativos à sua resposta face a vários cenários, bem sobre o controlador de Centro de Pressão no que respeita ao controlo de equilíbrio na aplicação de forças externas sobre os respectivos sensores, variação de declive do plano de suporte e presença de perturbações externas aplicadas à estrutura, foram intensivamente executados, e encontram-se descritos detalhadamente no relatório final de projecto 2005/2006 "Desenvolvimento de Algoritmos de Controlo para Locomoção de um Robot Humanóide" que se encontra disponível para livre acesso.

Este relatório faz a prossecução desses estudos, agora examinando os seguintes fenómenos:

- Impacto do pé sobre o solo, por parte da perna livre;
- Execução de trajectórias da perna de suporte, por manipulação do Centro de Pressão referência.

3. ESTUDO DO IMPACTO DO PÉ SOBRE O SOLO

3.1. Introdução

Pretende-se que a locomoção siga algoritmos o mais dinâmicos possíveis, que sejam adaptáveis a irregularidades presentes no solo, bem como a perturbações que possam ser aplicadas sobre a estrutura humanóide.
Cada passo realizado pelas pernas, envolvem o movimento da perna livre terminando com o impacto do seu pé sobre o solo. Como existe a possibilidade de o solo não ser perfeitamente plano, desconhece-se com rigor o momento deste impacto, podendo ocorrer antecipadamente ao momento previsto, bem como também posteriormente. Devido a tal, decidiu-se prosseguir com o estudo de um método que permita a geração de um evento por parte da unidade de controlo presente na perna livre que indique à unidade central que o pé atingiu o solo e que se encontra perfeitamente adaptado a ele, permitindo, desta forma, sinalizar à mesma unidade que pode prosseguir com o seguinte passo no processo de locomoção, tornando a perna livre na de suporte, e o inverso na outra perna.

Este capítulo propõe a análise do cenário de impacto sobre o solo, bem como a indicação de possíveis métodos permitam a automática detecção destes eventos e a sua sinalização à unidade principal.

3.2. Setup Experimental

Uma das pernas do robot humanóide foi segura a estrutura de altura fixa, de modo a simular o deslocamento da perna livre que supõe que a anca permaneça de altura fixa (Fig. 35).

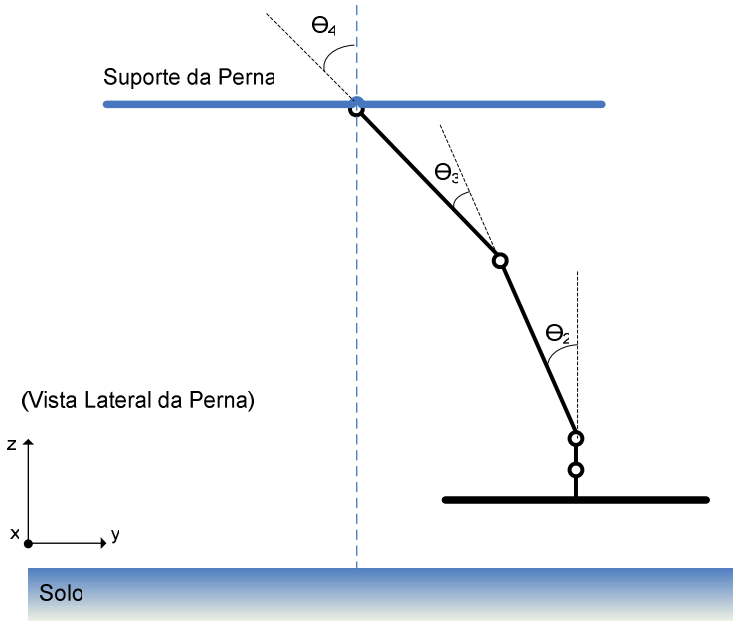


Fig. 35: Setup experimental para movimentação da perna com a anca fixa.

A altura a que se encontra esta estrutura fixa possui uma altura ao solo sensivelmente inferior à altura total da perna, de modo a permitir que, quando a perna atinge a postura vertical, esta toque o chão e levante a própria estrutura exercendo, desta forma, uma força não desprezável sobre o pé. Os movimentos serão realizados através da actuação ao servo paralelo ao pé e ao joelho (θ_2 e θ_3), mais ainda um terceiro servo presente na anca que permite o movimento da perna superior relativamente à estrutura de altura fixa (θ_4). Relativamente ao servo ortogonal ao pé que controla a sua rotação (θ_1), este manter-se-á constante de modo a que a base do pé se mantenha sempre alinhada com o solo.

Desta forma, as únicas variáveis independentes de controlo correspondem aos servos associados aos ângulos θ_2 e θ_3, com θ_4 dependente destes dois parâmetros de modo a manter o plano do pé alinhado com o solo:

$$\theta_1 = 0^o$$
$$\theta_4 = -(\theta_2 + \theta_3)$$

Como os ângulos θ_1, θ_2 e θ_3 fazem parte da unidade de controlo localizada na perna, e θ_4 já pertence a uma segunda unidade de controlo, teremos de enviar comandos de actuação a duas unidades simultaneamente.

3.3. Controlo da Perna

Para simplificar o controlo de movimentos da perna, foi desenvolvido um programa em *MatLab* (*mouse.m*[1]) que permite controlar as quatro juntas da perna (juntas ortogonal e paralela ao pé, joelho e anca) através de uma interface gráfica oferecida ao utilizador que com um simples *click* do rato pode seleccionar a postura desejada da perna (Fig. 36).

```
[<error>,<error_string>]=mouse(<com_handler>,<Tspeed>)
```

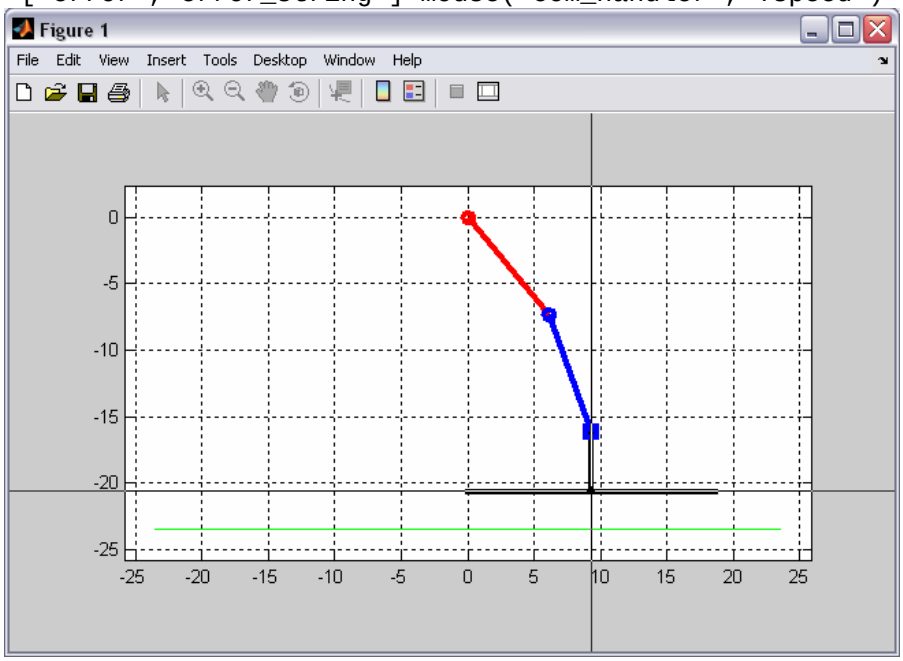


Fig. 36: Interface gráfica para selecção da posição da base do pé.

Através da interface gráfica da Fig. 36, o utilizador apenas tem de seleccionar, através do rato, a posição desejada do pé (ponto central da base do pé). A Fig. 36 representa um caso em que o ponto $(x,y)=(9,-21)$ é seleccionado. Em seguida são efectuados os seguintes procedimentos para a determinação das posições a solicitar aos servomotores:

a) Cálculo das coordenadas da anca mudando o referencial para o ponto central da base do pé;
b) Cálculo dos ângulos das juntas ortogonal e paralela ao pé e da junta do joelho (θ_1,θ_2,θ_3), usando a convenção da Fig. 34, pela cinemática inversa das coordenadas cartesianas da anca calculadas no ponto 1;
c) Cálculo do ângulo a aplicar à junta da anca (θ_4), de forma a assegurar que a base do pé fica paralela ao solo;
d) Adaptação dos ângulos determinados, para as posições (angulares) a aplicar aos servomotores;
e) Aplicação das posições nos servomotores, com uma determinada velocidade.

a) Cálculo das coordenadas da anca:

Considerando $(x,y,z)_{pé}$ a posição cartesiana da base do pé seleccionada pelo utilizador (tendo em conta a convenção da Fig. 34), então as coordenadas da anca $(x,y,z)_{anca}$ mudando o referencial para o ponto

[1] CD: /MatLab/Control/Movimento Cartesiano

seleccionado corresponde ao seu simétrico:
$(x, y, z)_{anca} = -(x, y, z)_{pé}$

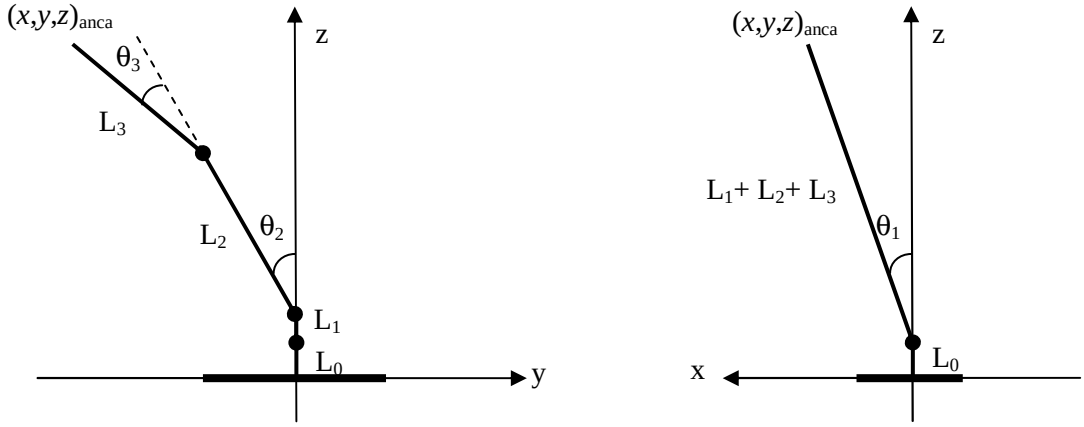


Fig. 37: Representação da Perna após a mudança de referencial.

b) Cinemática inversa para cálculo dos ângulos θ_1, θ_2 e θ_3:

Com a mudança de referencial efectuado atrás, podemos aplicar as equações da cinemática inversa do pêndulo duplo para determinar os ângulos θ_2 e θ_3.

$$\theta_i = \text{atan2}(K_1, K_2) \mp \text{atan2}\left(\frac{\sqrt{K_1^2 + K_2^2 - K_3^2}}{K_3}\right) - \theta_{offset}$$

$$Para\ \theta_2: \begin{cases} K_1 = 2 \cdot z \cdot L_2 \\ K_2 = 2 \cdot y \cdot L_2 \\ K_3 = y^2 + z^2 + L_2^2 - L_3^2 \end{cases} \qquad Para\ \theta_3: \begin{cases} K_1 = 0 \\ K_2 = 2 \cdot L_2 \cdot L_3 \\ K_3 = y^2 + z^2 - L_2^2 - L_3^2 \end{cases}$$

Contudo, dado que a perna pode estar sujeita a alguma inclinação devido ao ângulo θ_1, as variáveis de entrada (x, y, L_1, L_2 e L_3) vão estar sujeitas a transformações.
O ângulo θ_1 é facilmente obtido através das coordenadas da anca:

$$\theta_1 = \text{atan2}(z - L_0, x)$$

Dado que a inclinação da perna irá alterar os valores L1, L2 e L3 a aplicar na equação da cinemática inversa, temos de determinar as projecções dos elos sobre o plano *yoz*:

$$L_1' = L_1 \cdot \cos\theta_1$$
$$L_2' = L_2 \cdot \cos\theta_1$$
$$L_3' = L_3 \cdot \cos\theta_1$$

Além disso ainda temos de adaptar as coordenadas (x,y,z) da anca de modo a transformarmos a perna num pêndulo duplo:
$(x', y', z') = (x, y, z - L_1' - L_0)$

O cálculo das constantes K1, K2 e K3 na equação da cinemática inversa, para a ser feito do seguinte modo:

$$Para\ \theta_2: \begin{cases} K_1 = 2 \cdot z' \cdot L_2 \\ K_2 = 2 \cdot y' \cdot L_2 \\ K_3 = y'^2 + z'^2 + L_2'^2 - L_3'^2 \end{cases} \qquad Para\ \theta_3: \begin{cases} K_1 = 0 \\ K_2 = 2 \cdot L_2' \cdot L_3' \\ K_3 = y'^2 + z'^2 - L_2'^2 - L_3'^2 \end{cases}$$

Assim sendo, os ângulos θ_2 e θ_3 são determinados da seguinte forma:

$$\theta_2 = \text{atan2}(K_1, K_2) - \text{atan2}\left(\frac{\sqrt{K_1^2 + K_2^2 - K_3^2}}{K_3}\right) - \frac{\pi}{2}$$

$$\theta_3 = \text{atan2}(K_1, K_2) + \text{atan2}\left(\frac{\sqrt{K_1^2 + K_2^2 - K_3^2}}{K_3}\right)$$

A expressão para cálculo de θ_2, sem o factor de ajuste $-\pi/2$, devolve o ângulo do elo L_2 relativamente ao eixo *yy*. Como pretendemos o ângulo relativamente ao eixo *zz*, retira-se $\pi/2$ radianos. Daí este ajuste.

c) Cálculo do ângulo θ_4:

O ângulo da anca θ_4 é determinado de modo a que o pé esteja paralelo ao solo. Assim sendo, resulta que:

$$\theta_4 = -\theta_2 - \theta_3$$

d) Adaptação para as posições dos servomotores:

Tendo que conta que a unidade de controlo da perna apenas controla as juntas θ_1, θ_2 e θ_3, e que uma segunda unidade de controlo controla a junta da anca θ_4, podemos agrupar os ângulos em dois grupos destinados a cada unidade:

$$\theta_{scu1} = (\theta_1 \quad \theta_2 \quad \theta_3)$$
$$\theta_{scu2} = (\theta_4 \quad 0 \quad 0)$$

A adaptação passa por três fases:

1. Conversão do sinal do ângulo dependendo se o sentido de deslocamento coincide com o da Fig. 34 ou não (*signal*=±1);
2. Adaptar o ângulo através da relação de transmissão da junta (*nrel*);
3. Adicionar a posição do servomotor correspondente ao 0º (*offset*).

Desta forma, a equação usada para adaptação é a seguinte, sendo aplicada a cada junta:

$$\varphi_{servo} = \theta \cdot signal \cdot nrel + offset$$

e) Aplicação da posição nos servomotores:

Finalmente, a posição calculada $\varphi=(\varphi_1,\varphi_2,\varphi_3)$ é aplicada nos servomotores de cada unidade de controlo, com a velocidade T_{speed}, através da função *applyjoint*:

```
applyjoint(<com_handler>,<scu_id>,1,[Tspeed Tspeed Tspeed])
applyjoint(<com_handler>,<scu_id>,0,[φ1 φ2 φ3])
```

Outras duas funções foram desenvolvidas que recorrem à linha de comandos para manipular a perna humanóide. A rotina *motioncartesian* controla o movimento através de coordenadas cartesianas, executando os mesmos passos que a função *mouse*, mas sem a interface gráfica. Finalmente, também está disponível a rotina *motion* que controla também a perna, mas directamente através das posições angulares das suas juntas. Apenas realiza os passos c), d) e e) descritos atrás. Esta rotina é útil para a realização de testes de amostragem sensorial.

```
[theta,error,errorstr]=motioncartesian(H,xyz,leg,speed)
[error,errorstr]=motion(H,theta,leg,speed)
```

Para mais informações sobre estas funções, consulte a documentação fornecida directamente por elas.

3.4. Impacto do Pé com Todos os Motores Ligados

Com todos os motores da perna activados (em estado rígido), analisaram-se a saída dos sensores de força nas situações de descida provocando impacto contra o solo, e também de subida até o pé estar completamente suspenso. As grandezas medidas foram os quatro sensores de força presentes no pé e a corrente consumida por todos os servomotores intervenientes (juntas do pé, do joelho e da anca).

A configuração do setup experimental é a seguinte:

- Apenas é utilizada uma perna da estrutura humanóide, que por sua vez está fixa a uma estrutura fixa através da anca;
- Considerando que as coordenadas introduzidas usam como referencial o ponto do solo imediatamente abaixo da perpendicular que passa pela anca, as coordenadas (x, y, z)=(0, 0, 0) correspondem à postura vertical da perna;
- É possível descer o pé até atingir o solo, mas tal não corresponde à postura completamente vertical, estando um tanto flectida. As coordenadas associadas a este ponto são (x, y, z)=(0, 0, 0.4) cm.
- Quando a perna atinge a verticalidade máxima – (x,y,z)=(0, 0, 0) – a estrutura fixa à anca sobe (massa de alguns Kg).

A. Descida do Pé

A primeira parte das experiências correspondem a descer o pé a partir de um determinado ponto acima do solo, fazendo-o descer até efectuar impacto. A experiência em detalhe está descrita a seguir:

1. O pé inicialmente encontra-se em leve contacto no solo com as seguintes coordenadas iniciais: (x,y,z) = (0, 0, 0.4) cm;
2. O pé é elevado até à altura máxima de coordenadas (x,y,z) = (0, 0, 2.0) cm, mantendo a superfície do pé horizontal ao solo;
3. A perna é descida até atingir a verticalidade máxima: coordenadas (x,y,z) = (0, 0, 0). Durante este procedimento os sensores de pressão e corrente são amostrados à velocidade máxima permitida pelo *MatLab* e pelo sistema operativo Windows XP. O parâmetro tempo também será amostrado para associação aos dados de força e corrente.
4. No fim da amostragem, a perna é reposta na altura máxima: (x,y,z) = (0, 0, 2.0) cm.

Os dados dos sensores de força obtidos durante a descida podem ser consultados na Fig. 38. A numeração atribuída aos quatro sensores obedece aos indicados na Fig. 26.

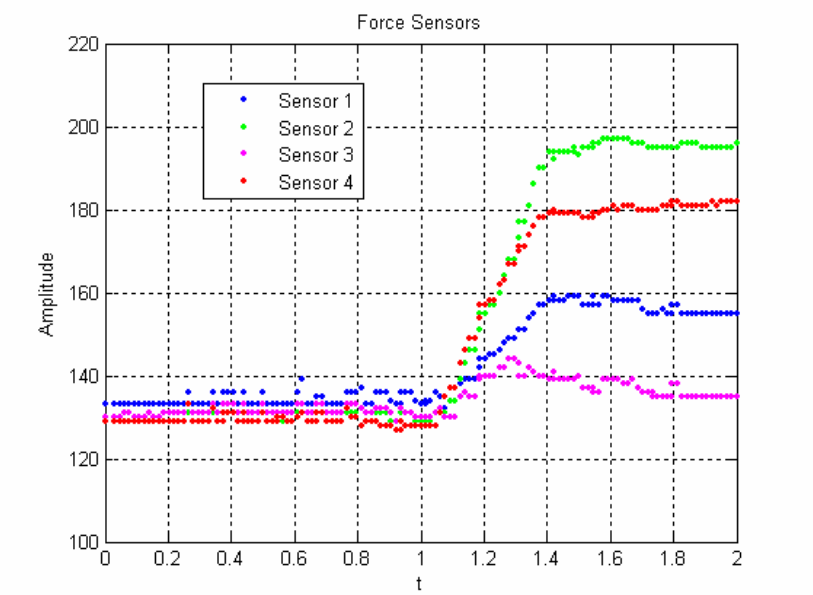


Fig. 38: Medição dos sensores de força durante a descida do pé contra o solo.

No início da experiência, quando o pé ainda está suspenso, podemos confirmar a correcta calibração dos sensores através do seu valor inicial ≈128. A partir de t=1s pode-se constatar que começa a ocorrer variação da saída destes sensores associando-se obviamente ao momento inicial do impacto contra o solo. Tendo em conta que à medida que a perna se aproxima da postura vertical, a estrutura sobe impondo uma força crescente sobre o pé, o processo de deflexão termina para t=1.4s quando a estrutura fica em modo estacionário. A partir deste ponto, a perna fica estática com uma força semelhante aplicada sobre cada sensor.
Contudo os sensores não reflectem semelhança nas forças aplicadas aos quatro sensores, revelando uma maior força sobre o sensor 2 (direito dianteiro), seguido do 4 (direito traseiro), e finalmente o 1 (esquerdo dianteiro) e o 3 (esquerdo traseiro). Esta informação é um tanto contraditória uma vez que a perna se encontra na postura vertical sobre o pé horizontal perfeitamente assente no solo. Tal indica que a sensibilidade dos quatro sensores difere por alguma razão. Analisando a estrutura do pé verifica-se algumas assimetrias que podem estar relacionadas com este efeito:

- O servomotor de controlo da junta ortogonal encontra-se sobre a parte traseira do pé, ao contrário da parte dianteira que não possui qualquer componente: a parte traseira possui um maior peso associado;
- E o eixo que permite o deslocamento de rotação lateral do pé não está centrado sobre a base do pé, mas ligeiramente para a direita, conferindo uma maior força aplicada sobre os sensores da direita.

Tendo em conta que a assimetria do eixo de rotação lateral impõe uma maior variação de força ao longo dos sensores, dado esta variação depender da força imposta sobre a anca (que para a estrutura considerada pode assumir vários quilogramas), ao contrário da presença do servomotor sobre a parte traseira do pé que apenas possui algumas décimas de quilograma, podemos deduzir que os sensores com maior força aplicada, por ordem decrescente, são os seguintes:

1. Direito traseiro;
2. Direito dianteiro;
3. Esquerdo traseiro;
4. Esquerdo dianteiro.

Esta ordem indicada apenas difere da observada na Fig. 38 no que respeita aos sensores dianteiro e traseiro, mas como já foi indicado, a influência do servomotor na parte traseira é desprezável quando comparável à introduzida pelo eixo de rotação lateral, pelo que poderíamos tanto obter em primeiro os sensores dianteiros como os traseiros com semelhante probabilidade.

Como a análise individual de cada sensor é difícil dada esta assimetria de sensibilidades, efectuando uma média dos quatro sensores, podemos obter a mesma informação de impacto e estabilização só pela análise de um único sinal, tal como se observa na Fig. 39.

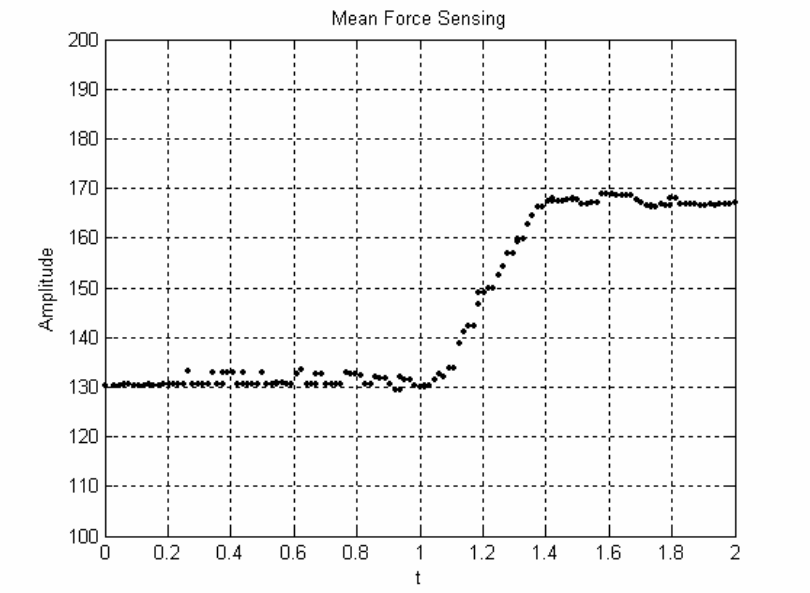


Fig. 39: Média aritmética dos sensores de força durante a descida do pé.

Pela análise à Fig. 39 podemos deduzir igualmente o momento do impacto (t=1s) e o instante inicial da estabilização (t=1.4s), usando os mesmos critérios de análise aplicados à Fig. 38.

Se em vez dos sensores de força, analisarmos a corrente consumida pelos diversos servomotores envolvidos (Fig. 40), podemos observar várias flutuações dos três servomotores inferiores da perna, tanto antes do impacto como depois, o que não nos transmite muita informação útil sobre este fenómeno. Quanto ao servomotor da anca (motor 1 do 2.º microcontrolador) ainda menos informação se pode extrair, sem qualquer consumo registado.
Além disso, várias leituras revelaram-se viciadas com alguns servomotores a imitaram o consumo de corrente feito por outros, tendo a certeza de que estes nada consumiam. Este efeito apenas se observou por entre os três servos da mesma unidade de controlo o que revela que algum isolamento nas linhas de alimentação é necessário.
Por estes motivos não se encontrou motivos para analisar estes sinais para o propósito de impacto no solo.

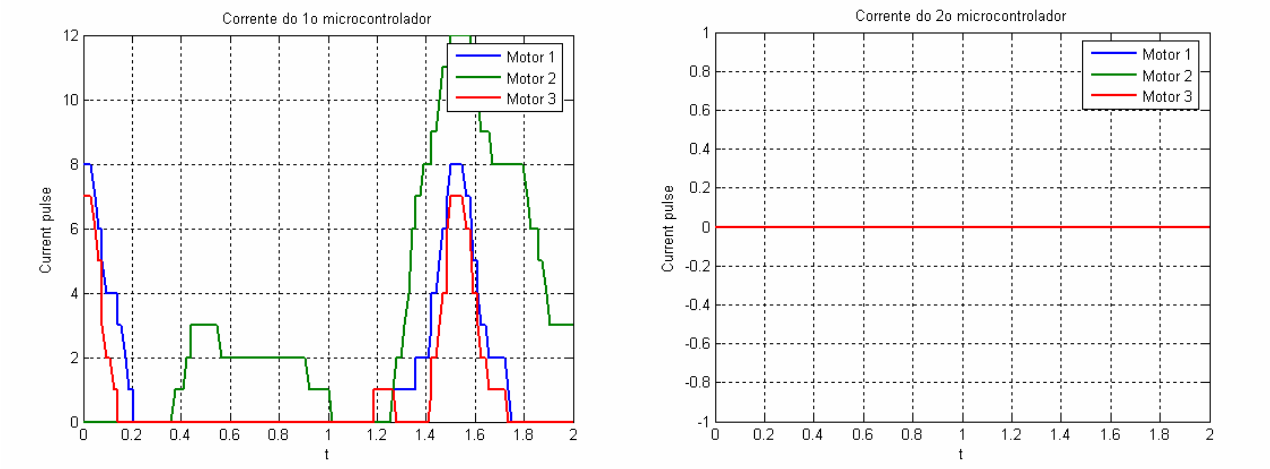


Fig. 40: Consumo de corrente por parte dos servomotores durante a descida do pé.

B. Subida do Pé

A experiência de subida do pé a partir do solo, foi também efectuada, através do seguinte procedimento:

1. A perna inicialmente encontra-se na postura vertical – (x,y,z)=(0,0,0) – com a estrutura ligeiramente suspensa acima do solo. A máxima força encontra-se aplicada sobre os sensores de força;
2. O pé é elevado até à altura máxima de coordenadas (x,y,z) = (0, 0, 2.0) cm, mantendo a superfície do pé horizontal ao solo. Durante este procedimento os sensores de pressão e corrente são amostrados à velocidade máxima permitida pelo *MatLab* e pelo sistema operativo Windows XP. O parâmetro tempo também será amostrado para associação aos dados de força e corrente.

Os gráficos demonstrativos da saída dos sensores de força individuais, bem como da sua média aritmética, podem ser visualizados na Fig. 41 e Fig. 42 respectivamente.

Pela análise individual dos sensores (Fig. 41) pode-se observar o mesmo comportamento que durante a descida do pé. Os mesmos sensores, 2 e 4, revelam a superior sensibilidade durante o início do percurso, com todos os sensores a estacionar no valor ≈128 quando o pé já se encontra suspenso acima do solo.

Tanto a análise individual dos sensores (Fig. 41) como da média dos quatro (Fig. 42), são capazes de nos fornecer a informação de começo de inflexão da perna (t=0.8s) e início da elevação do pé acima do solo (t=1.2s), pelo que a análise à média aritmética revela-se bastante mais vantajosa em termos de simplicidade e independência relativamente à sensibilidade de cada sensor de força.

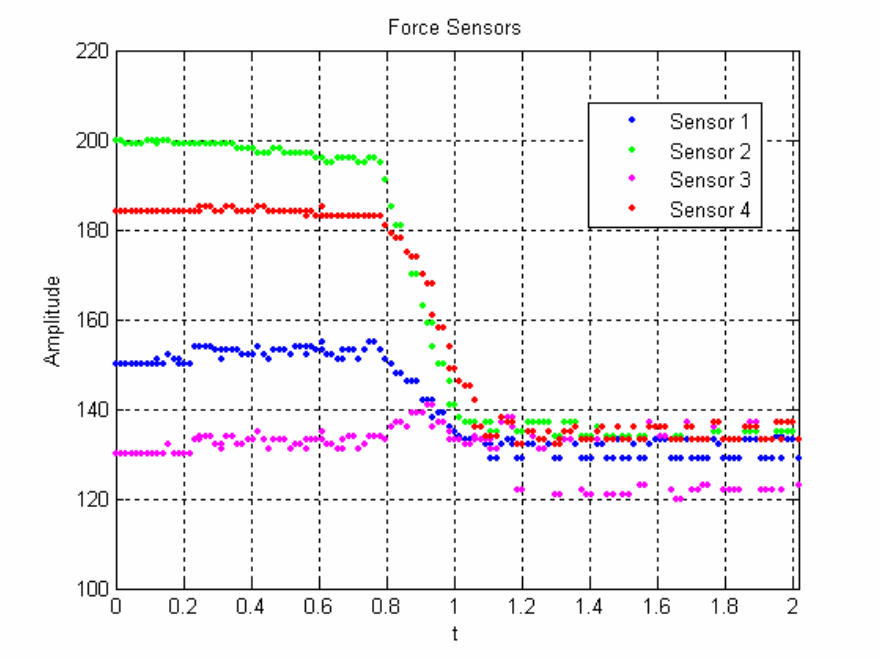


Fig. 41: Medição dos sensores de força durante a subida do pé a partir do solo.

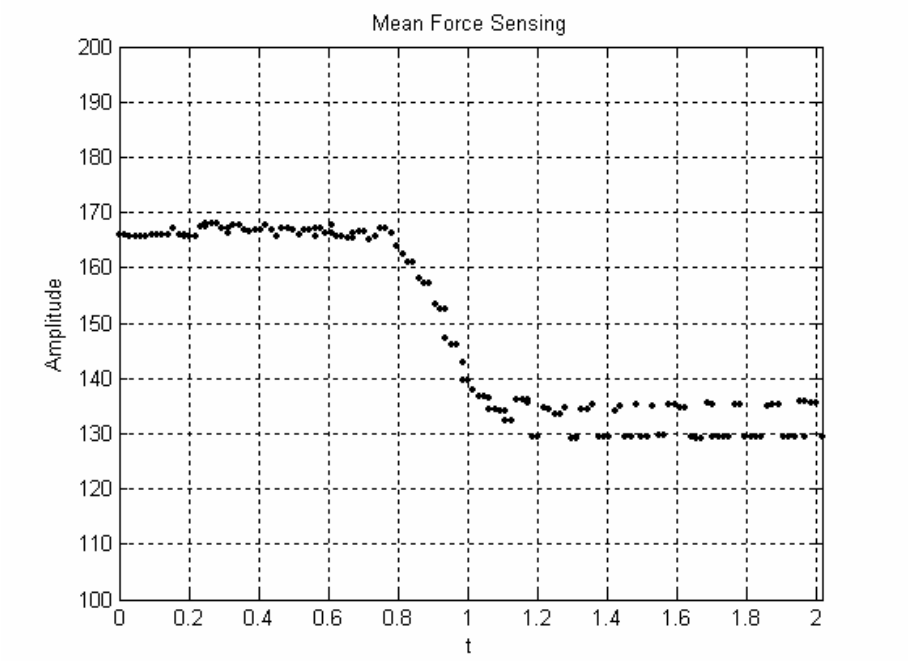


Fig. 42: Média aritmética dos sensores de força durante a subida do pé.

Os gráficos demonstrativos do consumo de corrente indicados na Fig. 43, confirmam mais uma vez a ideia transmitida na análise dos dados durante a descida do pé. Embora agora apenas se registe consumo a partir do momento de elevação do pé acima do solo, verificam-se algumas contradições, como é o caso de ausência de consumo quando a máxima força se encontrava aplicada (início do percurso), e a presença de consumo por parte do motor de rotação lateral do pé (motor 1) quando este nunca foi sujeito a nenhum deslocamento nem a nenhuma força lateral.

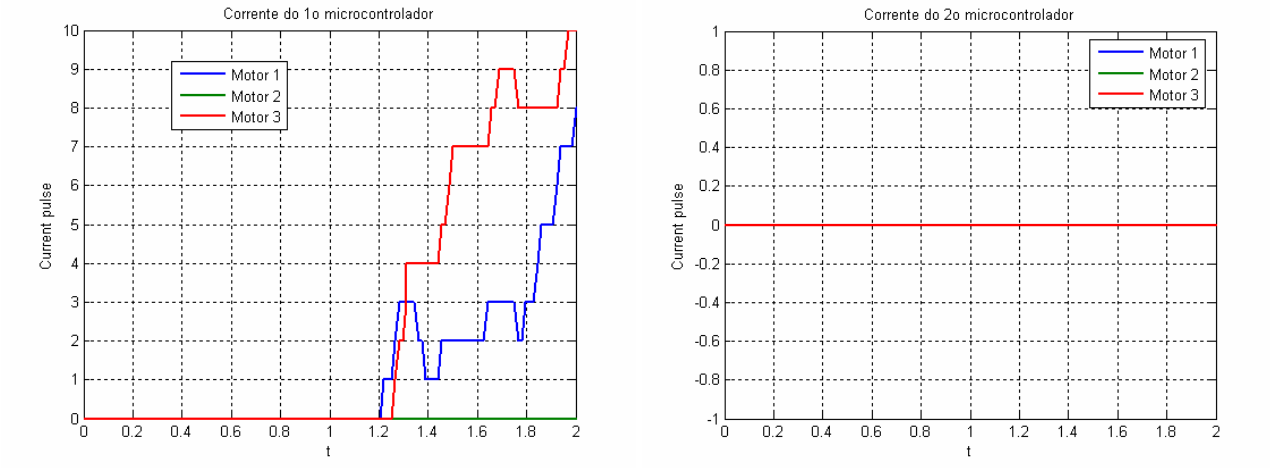


Fig. 43: Consumo de corrente por parte dos servomotores durante a subida.

3.5. Impacto do Pé com os seus Motores Desligados

Nas experiências realizadas nesta secção, desligaram-se os motores das juntas do pé para permitir a sua acomodação ao solo durante o impacto. Apenas o percurso de descida foi realizado, uma vez que a subida é semelhante à situação de motores ligados estudado na secção anterior.

Nesta secção, serão realizadas experiências adicionais em que um objecto estranho se encontra entre o pé e o solo para simular superfícies irregulares. Este objecto possui as seguintes características:

- Natureza do objecto: Placa metálica rectangular;
- Dimensões do objecto: 25cm (comprimento) x 8cm (largura) x 0.85cm (altura)

Todas as experiências foram realizadas da seguinte forma:

1. O pé inicialmente encontra-se em leve contacto no solo com as seguintes coordenadas iniciais: (x,y,z) = (0, 0, 0.4) cm;
2. O pé é elevado até à altura máxima de coordenadas (x,y,z) = (0, 0, 2) cm, mantendo a superfície do pé horizontal ao solo;
3. Os motores do pé (ortogonal e paralelo) são desligados, de modo a permitir a sua acomodação aquando à chegada ao solo;
4. Caso a experiência vise a simulação de planos irregulares, é adicionado a placa rectangular entre o pé e o solo, cobrindo apenas alguns dos sensores de força;
5. **A perna é descida até atingir a verticalidade máxima: coordenadas (x,y,z) = (0, 0, 0). Durante este procedimento os sensores de pressão e corrente são amostrados à velocidade máxima permitida pelo *MatLab* e pelo sistema operativo Windows XP. O parâmetro tempo também será amostrado para associação aos dados de força e corrente.**
6. No fim da amostragem, a perna é reposta na altura máxima: (x,y,z) = (0, 0, 2) cm.

A. Descida do Pé numa Superfície Regular

Esta experiência é semelhante à executada na secção 0 (
Descida do Pé), em que o pé desce e impacta contra o solo, mas com a excepção de que os motores que controlam as juntas do pé estão desactivados.
A Fig. 44 apresenta os dados experimentais amostrados no que concerne às saídas individuais dos sensores de força, média dos quatro sensores de força e corrente consumida pelas quatro juntas intervenientes.

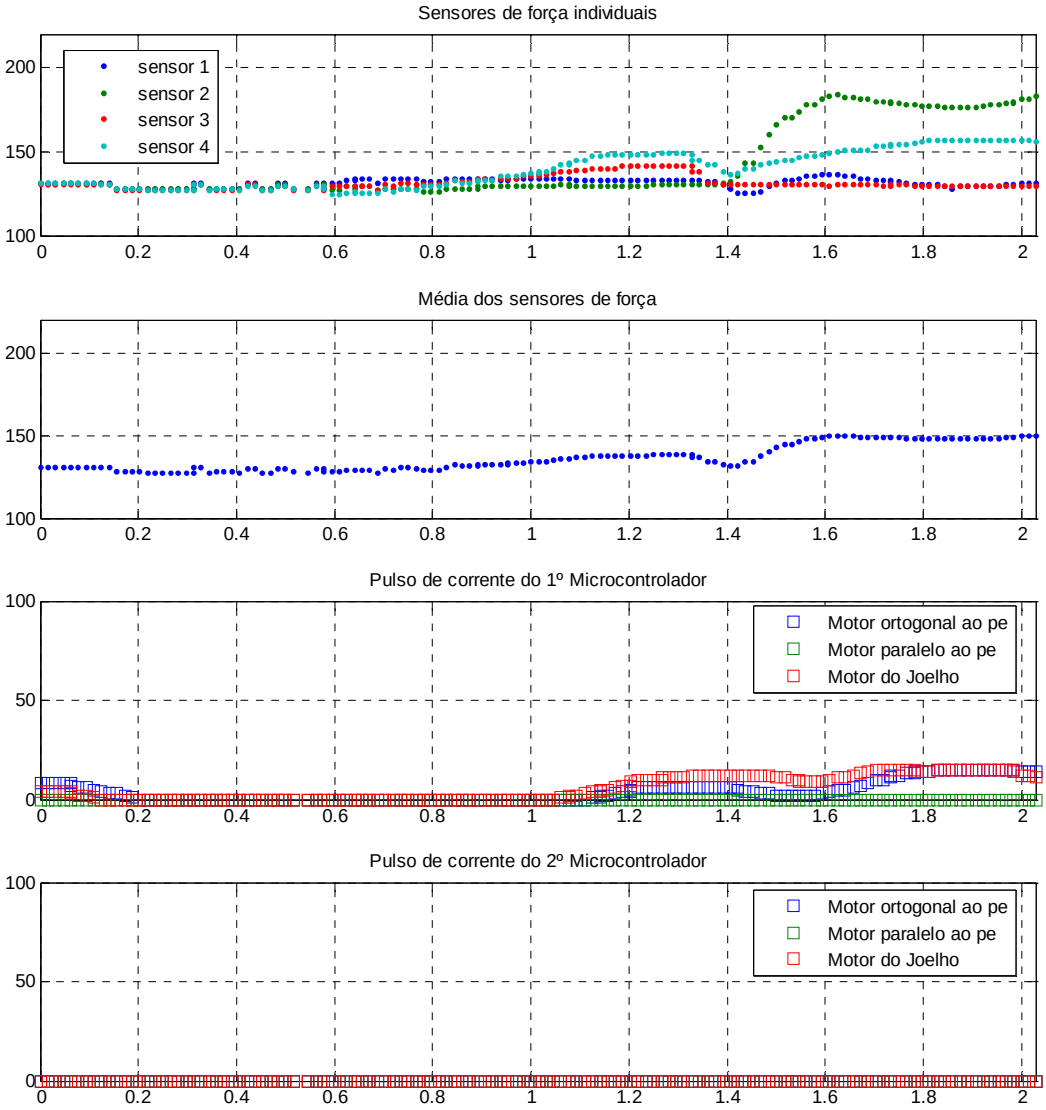


Fig. 44: Dados experimentais obtidos durante a descida contra o solo.

Como agora os motores do pé estão desligados, irá acontecer um efeito: inicialmente o pé encontra-se paralelo ao solo, pelo que à medida que a perna deflecte este paralelismo perde-se fazendo com que a parte traseira do pé impacte primeiramente com o solo, e mais tarde a restante estrutura do pé. Tal como a Fig. 44 reflecte, para t=1s, os sensores traseiros (3 e 4) revelam o impacto inicial nesta parte do pé, e mais tarde, a partir de t=1.4s, os restantes sensores informam o contacto total no solo.
Aqui, tal como na secção III. 3.4, os sensores da direita (2 e 4) apresentam maior sensibilidade, muito embora na prática todos os sensores possuam a mesma força aplicada.

A média aritmética dos quatro sensores, revelam também este fenómeno, mas de uma forma muito pouco pronunciada no que respeita ao contacto inicial dos sensores traseiros. Já para o contacto total é mais evidente registando-se primeiramente uma descida e novamente uma subida pronunciada para t>1.4s.

B. Descida do Pé contra um Objecto cobrindo os Sensores Dianteiros (1 e 2)

A mesma experiência foi realizada agora colocando uma placa metálica sobre os sensores dianteiros do pé (sensores 1 e 2). A Fig. 45 apresenta os dados registados.

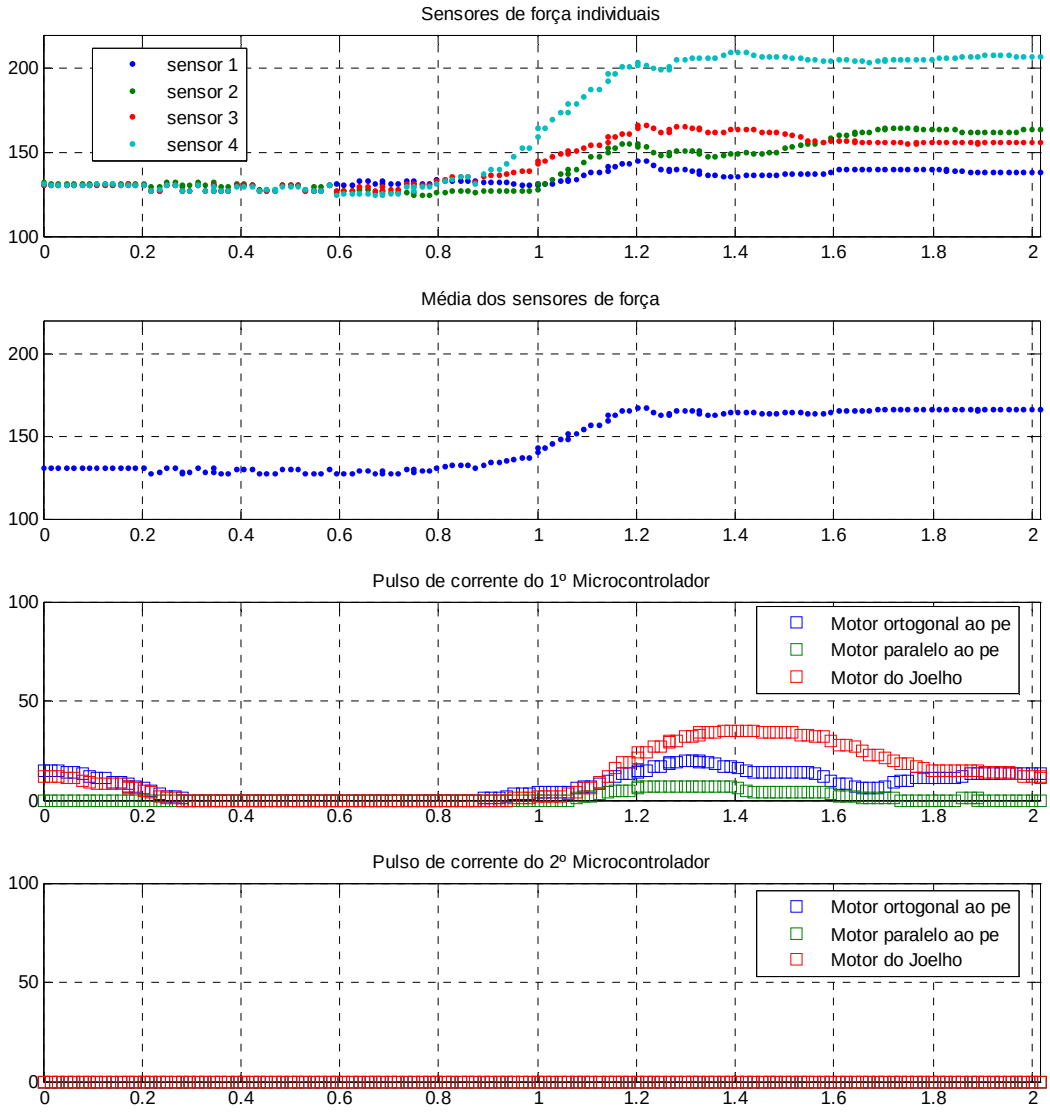


Fig. 45: Dados experimentais obtidos no impacto contra um objecto cobrindo os sensores dianteiros (1 e 2).

Pela análise aos sensores de força individuais, podemos detectar o impacto inicial sobre a parte dianteira do pé para t>0.9s, seguida do impacto total para t>1s. A estabilização apenas se consegue para t>1.2s, mas ainda com algumas flutuações.

O sinal da média aritmética não diz muito acerca do impacto inicial, observando-se apenas uma ligeira subida para t>0.9s. Apenas a partir do impacto total, a variação da média se torna mais evidente. Convém notar que o impacto inicial ocorre na zona do pé mais distante do seu centro, pelo que a inércia para o deslocamento das juntas é muito pequena, e por isso a reduzida variação registada nos sensores dianteiros. É de esperar igualmente o mesmo efeito quando a placa metálica se encontra sobre os sensores traseiros.

C. Descida do Pé contra um Objecto cobrindo os Sensores Traseiros (3 e 4)

Nesta experiência, a placa metálica foi introduzida por baixo dos sensores traseiros (sensores 3 e 4).

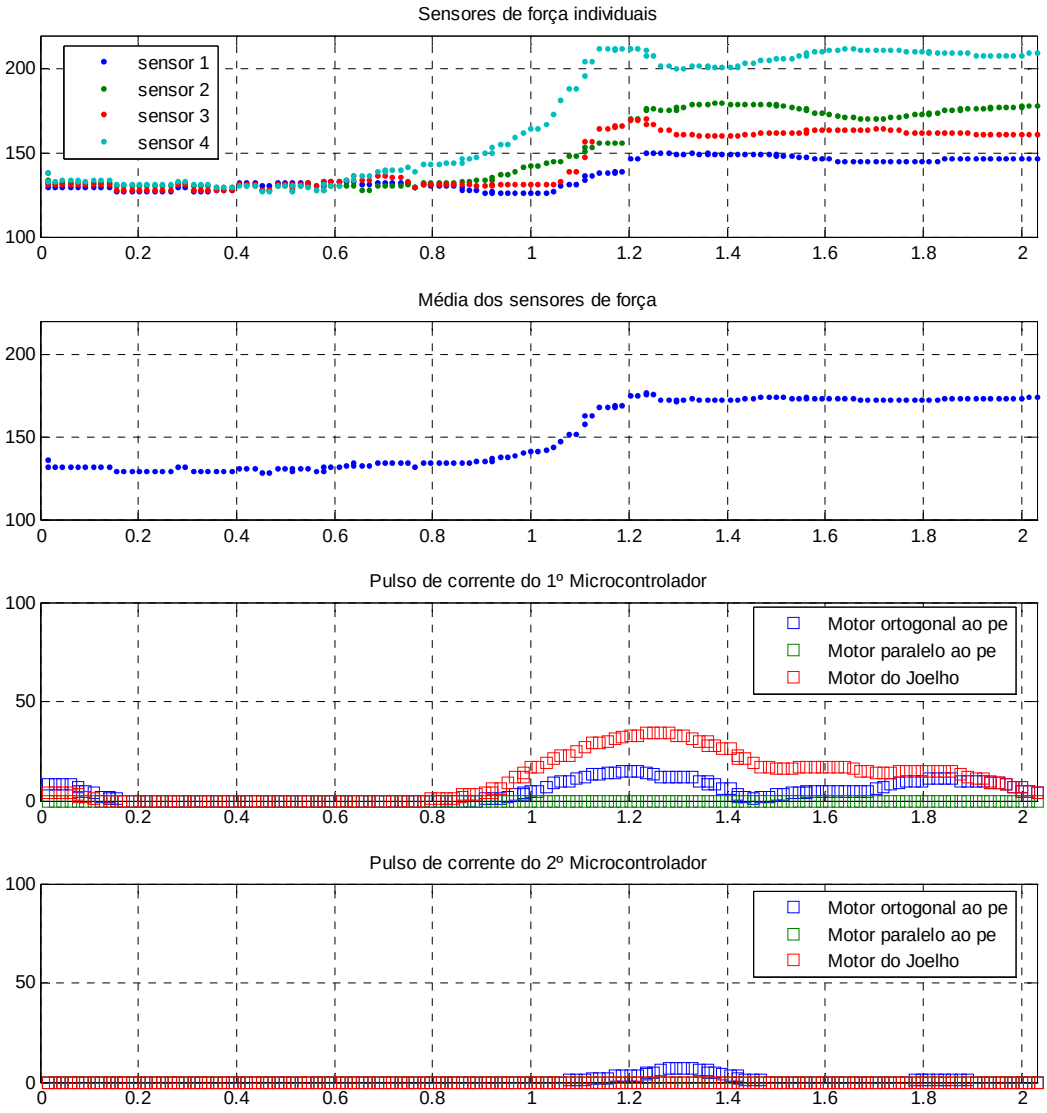


Fig. 46: Dados experimentais obtidos no impacto contra um objecto cobrindo os sensores traseiros (3 e 4).

Estudando o gráfico dos sensores individuais, pode-se visualizar uma notória subida por parte do sensor 4 a partir de t=0.6s, bem como um quase imperceptível acréscimo por parte do sensor 3, revelador do impacto inicial dos sensores traseiros. Esta diferença de variações é facilmente justificável pela diferença de sensibilidades por parte destes sensores: o sensor 4 corresponde a um dos mais sensíveis, e o 3 ao menos sensível de todos. Finalmente o impacto total chega para t>0.9s, em que o sensor 4 revela uma maior sensibilidade que o tradicional predominante 3, dado agora o peso estar a incidir sobretudo sobre o primeiro.

O gráfico da média revela os mesmos dados, bem como o instante inicial de estabilização (t>1.2s). De notar a dificuldade de detectar o momento do impacto inicial, tal como sugerido na situação de objecto sobre os sensores dianteiros.

D. Descida do Pé contra um Objecto cobrindo os Sensores da Esquerda (1 e 3)

A partir de agora, a placa metálica será colocada na parte lateral do pé, cuja distância ao seu centro é menor que nos casos de localização dianteira/traseira.

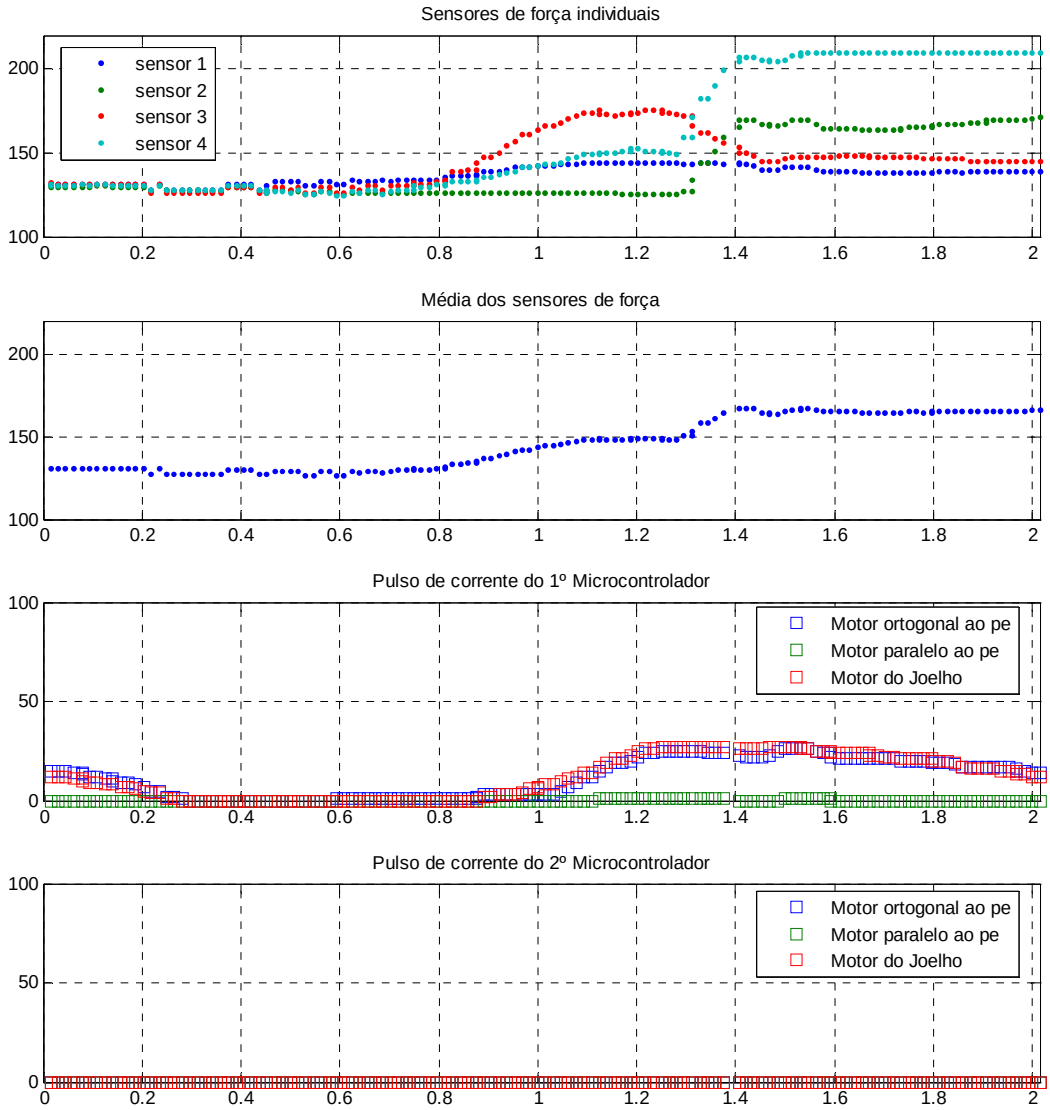


Fig. 47: Dados experimentais obtidos no impacto contra um objecto cobrindo os sensores da esquerda (1 e 3).

Dado que agora a placa está localizada por baixo dos sensores laterais da esquerda, maior inércia se apresenta para o deslocamento das juntas do pé, e por isso maior força será sentida pelos mesmos sensores, tal como se pode visualizar nos gráficos individuais dos sensores e da média, em que o impacto inicial é facilmente detectável para t>0.8s. O momento do impacto total e da estabilização também podem ser visualizados para t=1.3 e t=1.4s respectivamente.

De observar que, após a estabilização, os sensores 2 e 4 (direita) são os que registam maior força, dada a sua superior sensibilidade relativamente aos sensores 1 e 3 (esquerda).

E. Descida do Pé contra um Objecto cobrindo os Sensores da Direita (2 e 4)

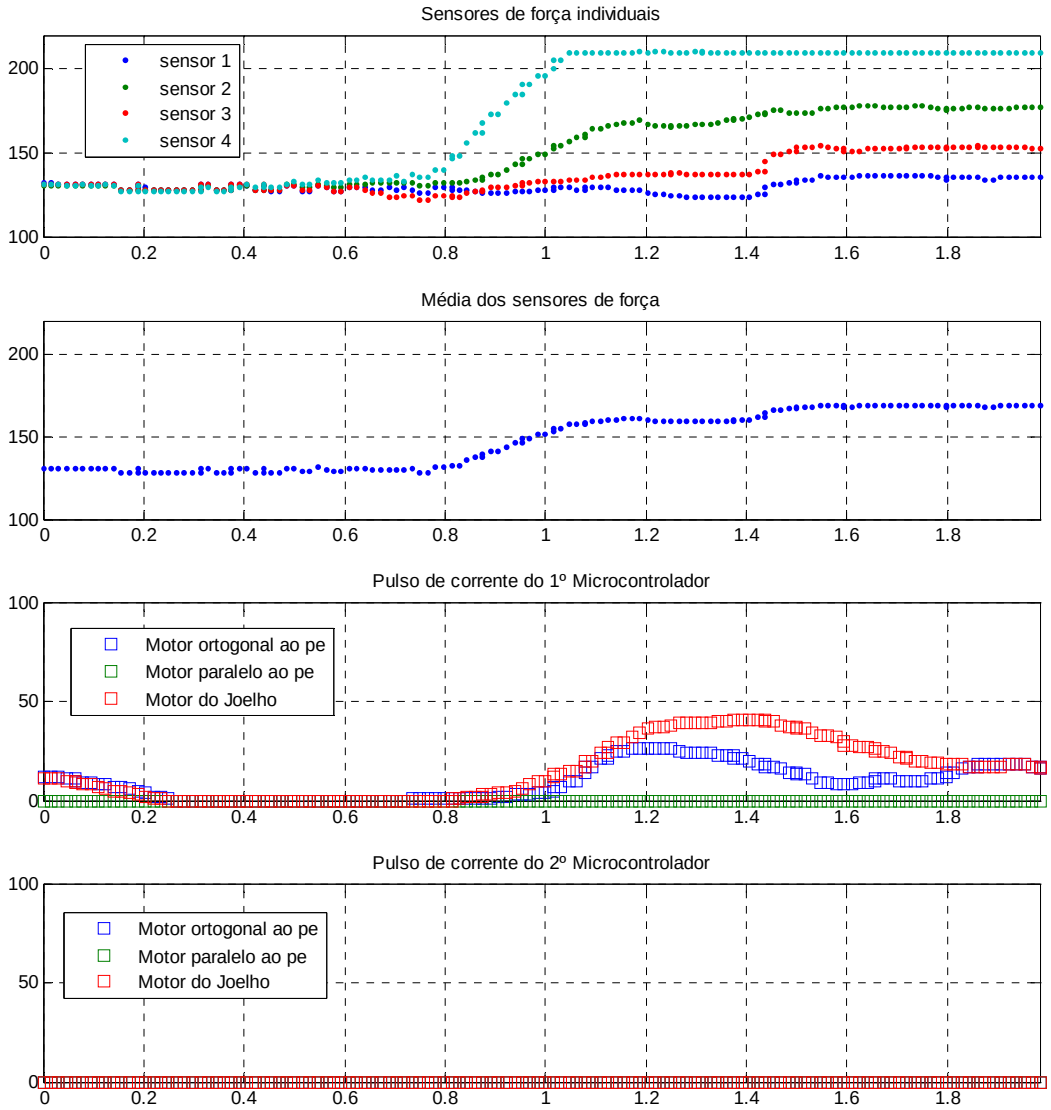


Fig. 48: Dados experimentais obtidos no impacto contra um objecto cobrindo os sensores da direita (2 e 4).

Como a placa metálica está agora por baixo do conjunto de sensores mais sensível (sobre a qual o eixo de rotação lateral do pé está a incidir), vamos obter dados bastante consistentes sobre os fenómenos físicos decorrentes.
Os sensores da direita (2 e 4) registam o impacto inicial a partir de t=0.8s, atingindo a máxima deformação, sensivelmente a partir de t=1.1s. Finalmente para t=1.4s, os restantes sensores 1 e 3 registam o impacto total, com estabilização dos sensores para t>1.5s.

O gráfico da média é bastante demonstrativo dos efeitos verificados, e regista um comportamento contrário ao verificado quando a placa era colocada sobre os sensores traseiros ou dianteiros. Nessas experiências passadas, era mais fácil a detecção do impacto total, dada a baixa inércia para deslocamento das juntas, mas agora, é muito mais notório o momento do impacto inicial, justificado precisamente pelo mesmo motivo da inércia existente para o deslocamento: agora é necessária uma força maior sobre o pé para deslocar as mesmas juntas do pé, e, por isso, os sensores registam uma maior variação neste momento.

3.6. Conclusões

Pela análise aos dados amostrados nas experiências realizadas, podemos extrair duas causas principais para que a saída dos sensores de força se desviem do valor esperado da força aplicada:

- Assimetria dos pontos de ligação do pé com a restante perna: servomotor localizado na traseira, e o eixo de rotação lateral desviado para a direita;
- Inércia existente sobre as juntas do pé depende do tipo da superfície irregular.

Devido à assimetria dos pontos de ligação do pé com a perna, nomeadamente a do eixo rotacional lateral do pé, que corresponde à maior influência, tem-se uma maior sensibilidade por parte dos sensores de força da direita (2 e 4), o que leva a informação contraditória aquando da análise de cada sensor na presença de irregularidades sobre a parte lateral esquerda, ou dianteira/traseira. Por este motivo, é de evitar a análise da informação sensorial individual de cada sensor, mas sim da média ou da soma dos quatro sensores.

Pela análise da média aritmética, pode-se obter dados mais consistentes do momento do impacto inicial e total, muito embora a sua notoriedade dependa da sensibilidade dos sensores envolvidos. Se o impacto se aplicar ao conjunto mais sensível de sensores, esse momento é mais facilmente detectável, do que no caso dos menos sensíveis, sendo este último mais susceptível a erros de detecção.
Finalmente, também o factor inércia existente nas juntas do pé, prejudica a detecção do momento de impacto inicial em que pouca inércia existe. Isto acontece quando a irregularidade do solo se apresenta sobre a parte dianteira ou traseira do pé: por isso a grande dificuldade da detecção do impacto inicial. Já nos casos de irregularidades sobre as partes laterais do pé, maior inércia está presente, o que torna mais facilmente identificável o momento do impacto inicial, mas mais difícil o do impacto total – corresponde à situação inversa.

Resumindo, podemos concluir o seguinte:

- A assimetria de sensibilidades, levam a informação contraditória de quais os sensores envolvidos no impacto inicial e total. Daí a sugestão da utilização do sinal média;
- Esta assimetria, faz com que as irregularidades aplicadas sobre os sensores da direita (os mais sensíveis) sejam muito mais pronunciadas na informação sensorial;
- A existência de inércia para o deslocamento dos servomotores do pé, exige uma maior força no caso das irregularidades se localizarem nas secções laterais. Por este motivo, o momento do impacto inicial é muito mais pronunciado quando a irregularidade se localiza nas partes laterais, do que nas restantes.

Analisando cada caso, podemos prever os resultados à luz das conclusões enunciadas. A Tabela 37 descreve a capacidade de detecção do momento de impacto inicial e total para cada cenário possível.

Localização da irregularidade	***Impacto Inicial***	***Impacto Total***
Não existe	Difícil de detectar devido à baixa inércia.	Fácil de detectar graças à elevada sensibilidade dos sensores da direita.
Traseira		
Dianteira		
Esquerda	Fácil de detectar devido à elevada inércia.	Relativamente fácil de detectar graças à elevada sensibilidade dos sensores da direita.
Direita	Fácil de detectar devido à elevada inércia e elevada sensibilidade dos sensores envolvidos.	Difícil de detectar devido à baixa sensibilidade dos restantes sensores.

Tabela 37: Capacidade de detecção dos momentos de Impacto para vários cenários.

A Fig. 49 até à Fig. 53, exemplificam o que foi descrito na Tabela 37. Estes gráficos são baseados nas experiências discutidas na secção III. 3.5 e visualizam a variação do sinal média, mais precisamente a diferença do sinal em cada instante com o de 0.2s atrás, para cada cenário.

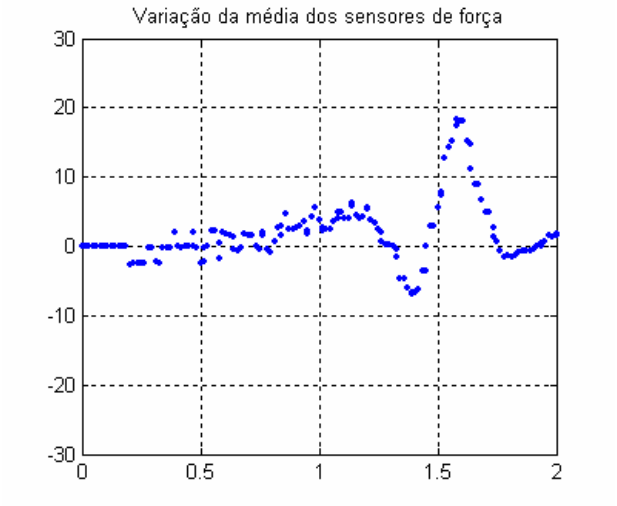


Fig. 49: Variação da Média na ausência de irregularidades.

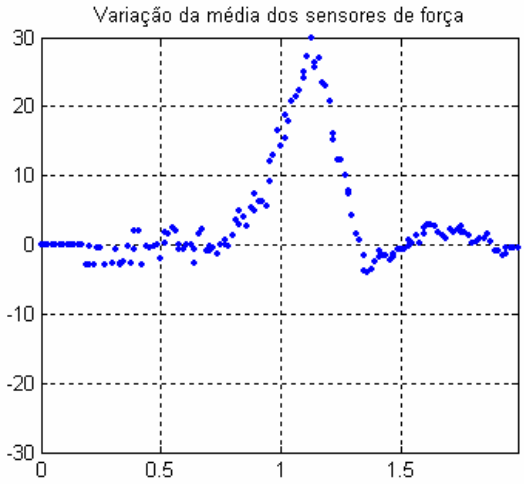


Fig. 50: Variação da Média com irregularidade na secção traseira.

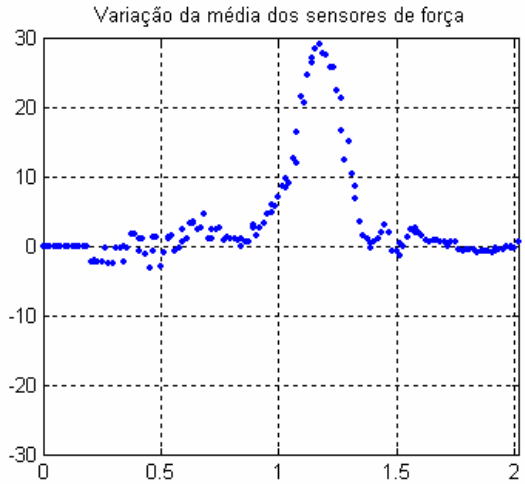


Fig. 51: Variação da Média com irregularidade na secção dianteira.

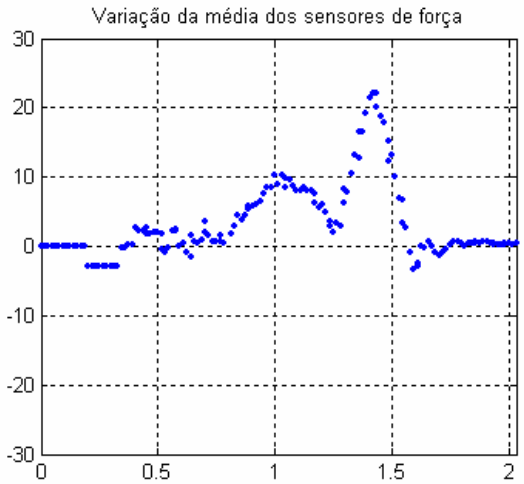


Fig. 52: Variação da Média com irregularidade na secção esquerda.

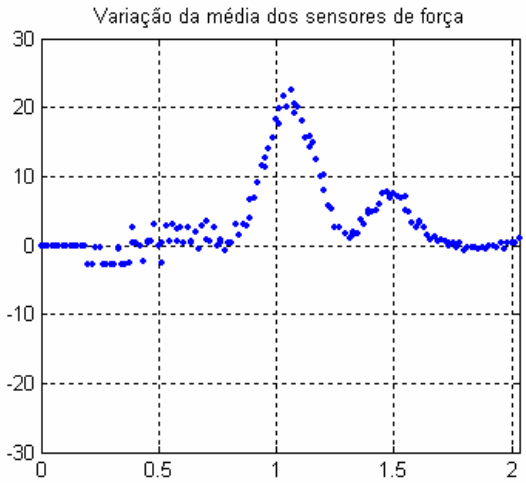


Fig. 53: Variação da Média com irregularidade na secção direita.

Como se pode observar na Fig. 49 à Fig. 51, o impacto inicial é difícil de ser observado nas circunstâncias de irregularidade na secção dianteira ou traseira do pé, não se observando mesmo, nas duas últimas figuras, impulso nenhum descrevendo este fenómeno (apenas se observa uma ligeira flutuação dos dados sensoriais).

Apenas o impulso descritivo do impacto total é claramente visualizado.
Já no caso de irregularidade na secção esquerda do pé, os dois impulsos indicativos do impacto inicial e total estão claramente pronunciados. O impulso de impacto inicial é favorecido pela elevada inércia ao deslocamento das juntas do pé, e o segundo impulso, de impacto total, favorecido pela elevada sensibilidade dos sensores da direita. Este cenário é claramente o melhor pois permite uma boa definição dos dois momentos.
Relativamente ao cenário de irregularidade na secção direita do pé, tem-se um claro impulso indicativo do impacto inicial, favorecido tanto pela elevada inércia como pela elevada sensibilidade dos sensores envolvidos, e em seguida, o impulso de impacto total menos pronunciado, motivado pela baixa sensibilidade dos restantes sensores.

É importante notar que embora os efeitos de assimetria na sensação de força, tornem uns sensores mais sensíveis que outros na detecção de mudanças físicas do sistema, eles continuam a medir com rigor a força aplicada – não se tratam de falsas medições. Contudo, para o objectivo em causa, que é a detecção de impacto, seria importante a ausência destas assimetrias para poder analisar com exactidão quais as partes do pé envolvidas no impacto. Dado que tal não é possível, temos de recorrer a estratagemas para possibilitar pelo menos a detecção do impacto inicial e total, que é o que interessa para a unidade central de processamento para a execução das sequências de locomoção.

Por isso, o algoritmo para detecção destes dois momentos, basicamente deverá obedecer aos seguintes critérios:

1. O primeiro pico positivo do sinal *média* corresponderá ao momento do impacto inicial;
2. O segundo pico positivo do sinal *média* diz respeita ao momento inicial do impacto total.

O algoritmo será, pois, baseado na variação do sinal média, devendo ser suficientemente fiável para ser capaz de detectar os menos pronunciados impulsos, sem o risco de falsa detecção. Para isso dever-se-á definir um limiar para detecção de picos que não seja demasiado baixo para não correr o risco de falsas detecções, nem demasiado elevado para evitar a não detecção de picos.

4. LOCOMOÇÃO ATRAVÉS DO CONTROLO DE COP

4.1. Introdução

Trabalhos anteriores (ver relatório 2006/07), mostraram que é possível utilizar o controlador de equilíbrio para sustentar as pernas na postura vertical, face à inclinação do plano de suporte. Dois possíveis controladores para esta tarefa foram desenvolvidos para alcançar esse objectivo:

- Controlo proporcional de cada junta ortogonal do pé (nos eixos *xx* e *yy*);
- Controlo usando a matriz Jacobiana.

Dado que o controlador usando a matriz Jacobiana permite controlar todas as três juntas da perna, e de uma forma pouca sensível a imprecisões no cálculo desta matriz, este foi o controlador adoptado, obtendo-se resultados bastante satisfatórios.

No passado o único objectivo era o de equilibrar a perna face a variações e irregularidades do solo. Contudo, também poderá ser utilizado para a realização de trajectórias, pela variação do CoP desejado. Se em vez de utilizarmos como referência o CoP=(0,0) correspondente à postura vertical, usarmos outros valores, a perna desloca-se para outras posturas efectuando movimento. Ora, é do interesse que quando o robot se encontra sobre a perna de suporte durante a locomoção, além de fornecer o equilíbrio também seja capaz de a mover ajudando a realizar as sequências de locomoção. Para tal, e como já foi enunciado, ir-se-á actuar no valor referência do CoP (CoP_{ref}) no controlador de equilíbrio para cumprir este objectivo (Fig. 33).

Neste capítulo, vários estudos serão feitos sobre os sensores de força, de modo a averiguar os seguintes aspectos:

- Relação entre o Centro de Pressão medido e o Centro de Gravidade, em circunstâncias estáticas;
- Relação entre o Centro de Pressão medido e o Centro de Gravidade, durante a realização de trajectórias;
- Execução de trajectórias de vários formatos, e comparação dos trajectos efectuados com os previstos.

4.2. Controlo do CoP Referência

A. Funções Básicas

Como já foi discutido na secção II. 2.1, é possível seleccionar um de quatro controladores possíveis, sendo um deles o de equilíbrio (Tabela 11). A função *applycontrol*, com o parâmetro *param* igual 3 (ver Tabela 6), é destinada à escolha deste controlador. O parâmetro *servos* é um vector de 3 elementos e selecciona o controlador a usar para cada junta: de acordo com a Tabela 7, para activar o controlador de centro de pressão para todas as juntas devemos atribuir o vector [1, 1, 1]:

```
[rx,error,errorstr]=applycontrol(H,scu_id,3,[1 1 1])
```

Para configurar o ganho de compensação deste controlador, de modo a ajustar a reactividade a variações do sinal de erro na sua entrada, a função *applycontrol* é igualmente utilizada, mas com o parâmetro *param* igual a 2 (Tabela 6). O parâmetro *servos* é, assim, usado para definição do ganho de compensação para cada junta:

```
[rx,error,errorstr]=applycontrol(H,scu_id,2,[K1 K2 K3])
```

Só depois de activo o controlador adequado, podem ser usadas as funções de actuação *applyjoint*, quer para a definição do valor referência (CoP_{ref}) quer para o período da trajectória (Tabela 13).

Note que a função *applyjoint* é a única cujo comportamento depende do controlador activo, sendo assim possível definir um valor referência e um período de trajectória para cada controlador, com uma única função.

B. Funções Gráficas

Tal como existe uma aplicação gráfica para visualização do Centro de Pressão aplicado sobre um dos pés (*cop_realtime.m*), obtido a partir das saídas dos sensores de força, utilizando o grafismo apresentado na Fig. 32, também foi criada a equivalente para actuação utilizando o rato como interface para introdução das coordenadas do centro de pressão de referência do controlador (*CoPref_exe.m*).
A Fig. 54 apresenta um exemplo de como as coordenadas bidimensionais do Centro de Pressão desejado são introduzidas: a mira que se observa é controlada pelo rato que selecciona o ponto espacial desejado. Além disso também é possível controlar a altura desejada da anca: para tal, utiliza-se o botão direito do rato que fará aparecer uma caixa de diálogo para introduzir numericamente a altura a atribuir à anca (Fig. 55).

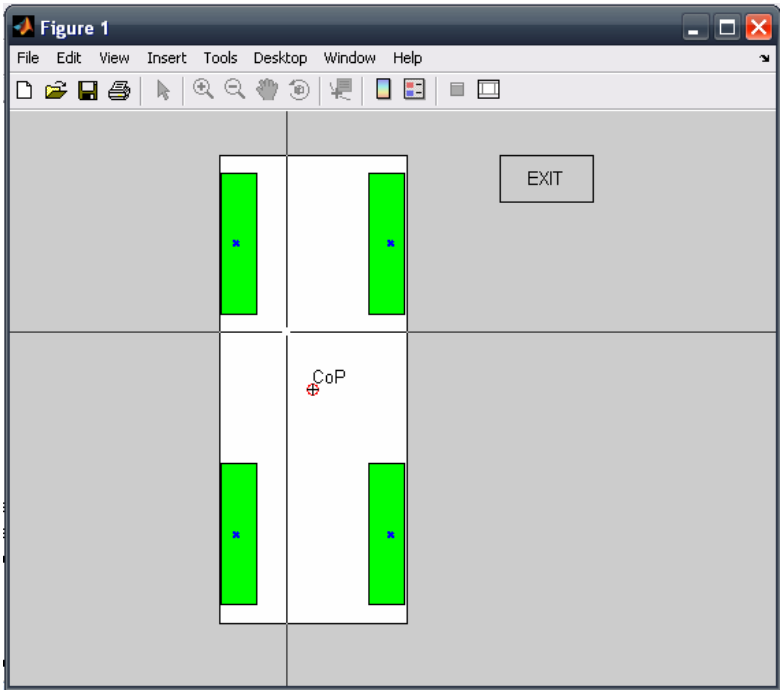


Fig. 54: Actuação sobre o Centro de Pressão referência.

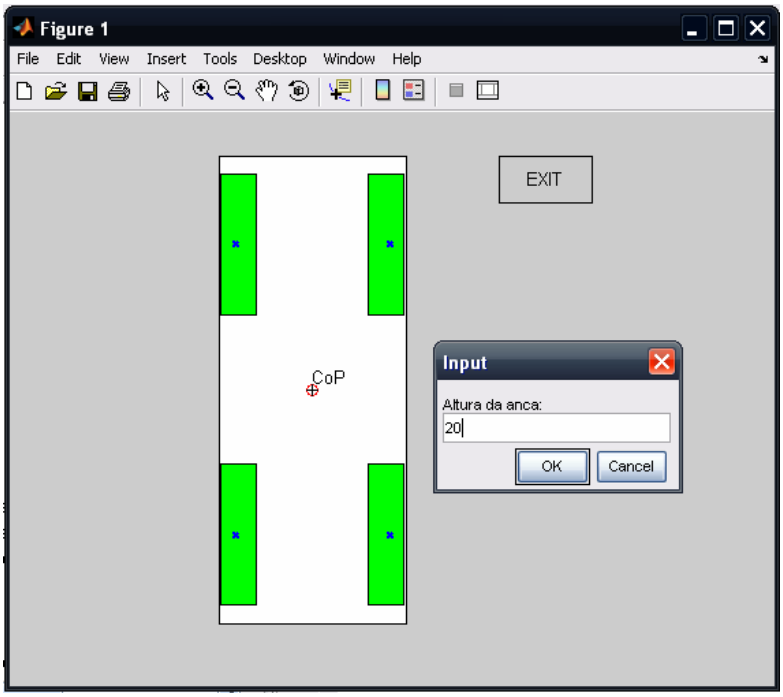


Fig. 55: Actuação sobre a altura desejada da anca.

Para executar estas duas rotinas é necessário primeiramente inicializar as comunicações. À função *cop_realtime.m* apenas é necessário fornecer o *handler* da linha de comunicações (<*H_handler*>), e o

endereço da unidade controladora do pé de interesse (<*scu_id*>).

```
cop_realtime(<H_handler>,<scu_id>)
```

Já para a função *copref_exe.m* é necessário adicionalmente o ganho de compensação do controlador de equilíbrio (<*controller_gain*>). Este valor é único e aplica-se de igual forma a todas as três juntas da unidade seleccionada.

```
[error,errorstr]=copref_exe(<H_handler>,<scu_id>,<controller_gain>)
```

Rotinas semelhantes a estas serão úteis adiante para a análise do comportamento estático dos sensores de força quando o centro de pressão referência é alterado.

4.3. Análise Estática do CoP Referência

A análise estática quando o CoP referência é alterado, é importante para verificar se o centro de pressão medido corresponde ao actuado, analisando assim se existe algum erro associado. Também pela comparação do CoP de referência com a cinemática directa das juntas do joelho e da anca, é possível obter uma relação entre os valores actuados do CoP e as coordenadas reais das juntas, permitindo assim controlar a posição destas juntas em termos de coordenadas espaciais.

As experiências realizadas a seguir utilizam um código semelhante à rotina *copref_exe* fazendo a actuação do CoP para vários pontos específicos. A seguir é realizada a leitura sensorial:

1. Actualização do CoP referência;
2. Esperar que o controlador atinja o equilíbrio;
3. Esperar alguns segundos, para assegurar a paragem dos servos;
4. Leitura dos dados sensoriais;
5. Voltar a 1.

Os dados sensoriais lidos são os seguintes:

- Sensores de força para cálculo do centro de pressão actual;
- Ângulos das diversas juntas da perna, para cálculo da cinemática directa.

As direcções dos eixos cartesianos aplicados seguem a convenção da Fig. 34.

A. Variação do CoP no eixo *xx*

Efectuando 30 passos desde as coordenadas CoP=(30,0,0) até CoP=(-30,0,0), obtiveram-se os dados dos sensores de força e da cinemática directa do joelho e da anca apontados na Fig. 56 e na Fig. 57.

Pela análise à Fig. 56 (coordenada *x*), podemos concluir que o controlador de CoP está a funcionar regularmente cumprindo todos os pontos de referência atribuídos. Já o CoP na componente *y* apresenta variações desprezáveis, tal como esperado, dado que o percurso do CoP referência é apenas feito no eixo *xx*.

No que respeita à cinemática directa, as juntas do joelho e da anca apresentam percursos aproximadamente lineares perto da coordenada nula (postura vertical), mas para pontos mais distantes (perna com maior inclinação) a curva arqueia assemelhando-se à função arco-seno. Tal comportamento deve-se à gravidade que atribui uma maior inércia ao movimento nos pontos mais inclinados. A componente *y* apresenta igualmente flutuações desprezáveis tal como o esperado.

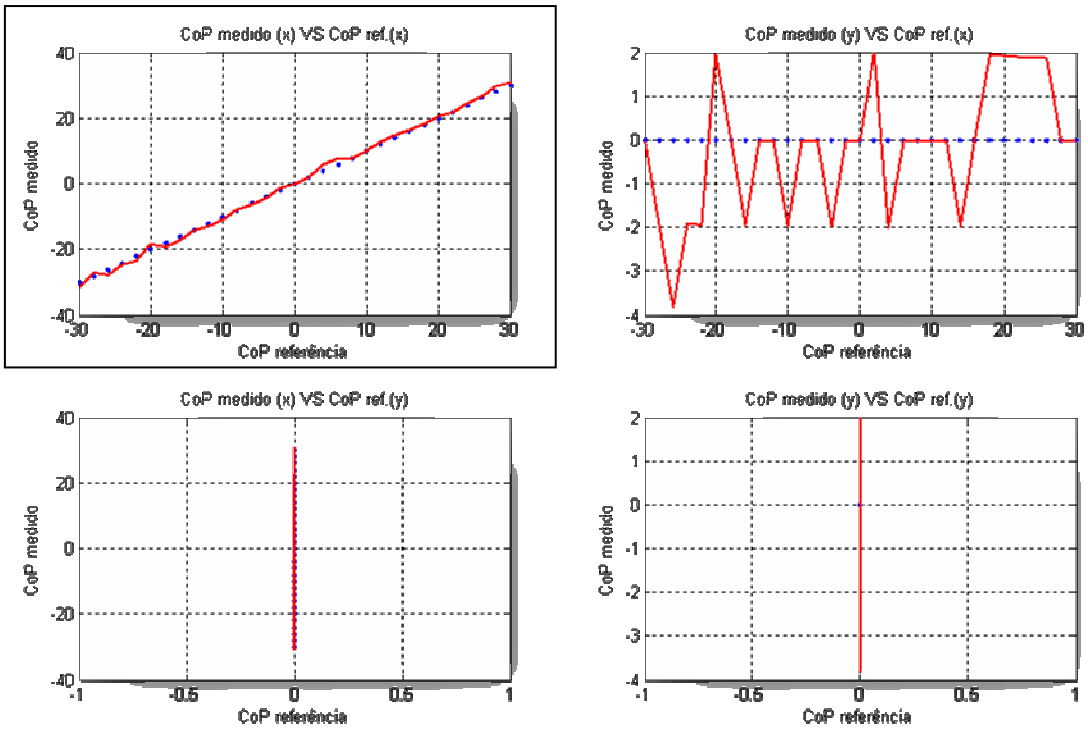


Fig. 56: Relação entre o CoP medido e o actuado, para 30 passos ao longo do eixo *xx*.

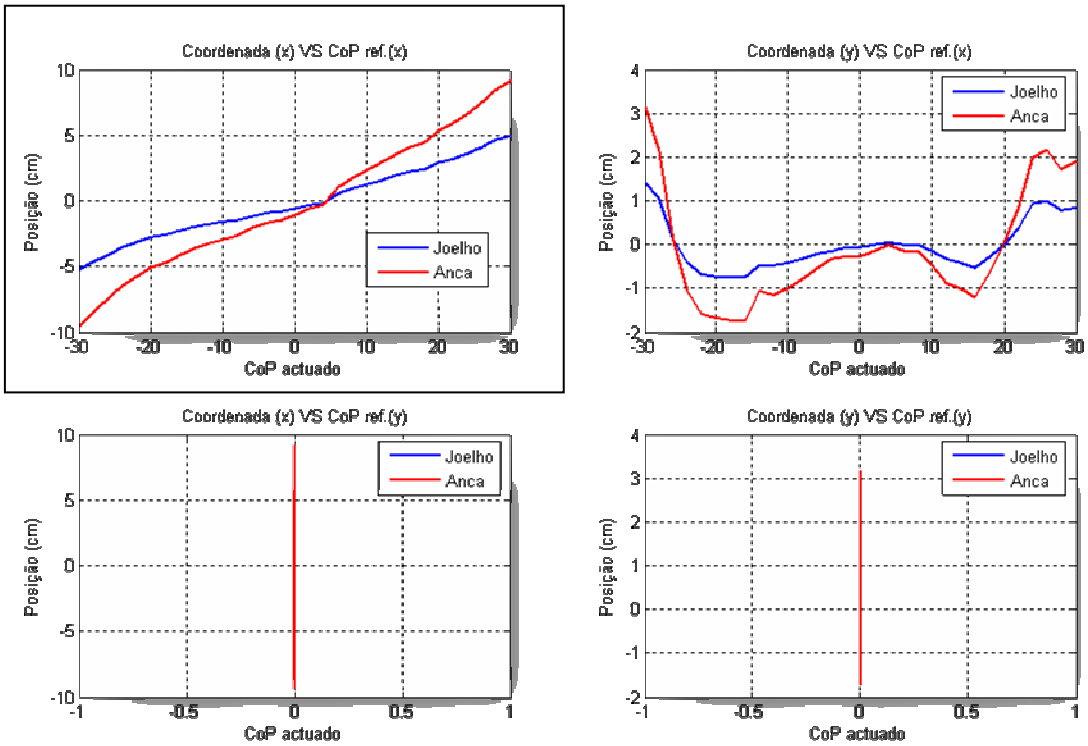


Fig. 57: Relação entre o CoP medido e a cinemática directa do joelho e da anca, ao longo do eixo *xx*.

B. Variação do CoP no Eixo *yy*

Efectuando a mesma experiência, mas agora realizando 32 passos ao longo do eixo yy: CoP=(0,–30,0)→(0,34,0), obtiveram-se os dados da Fig. 58 e Fig. 59. Como se pode comprovar, o comportamento é muito semelhante ao observado na experiência anterior.

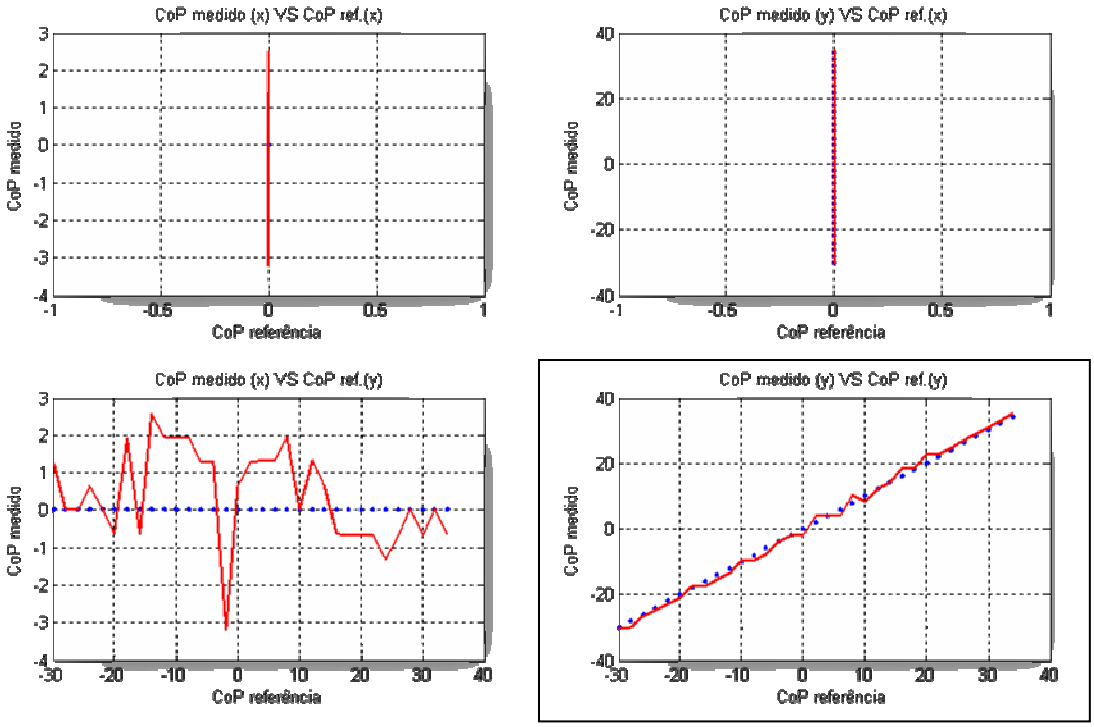


Fig. 58: Relação entre o CoP medido e o actuado, para 32 passos ao longo do eixo *yy*.

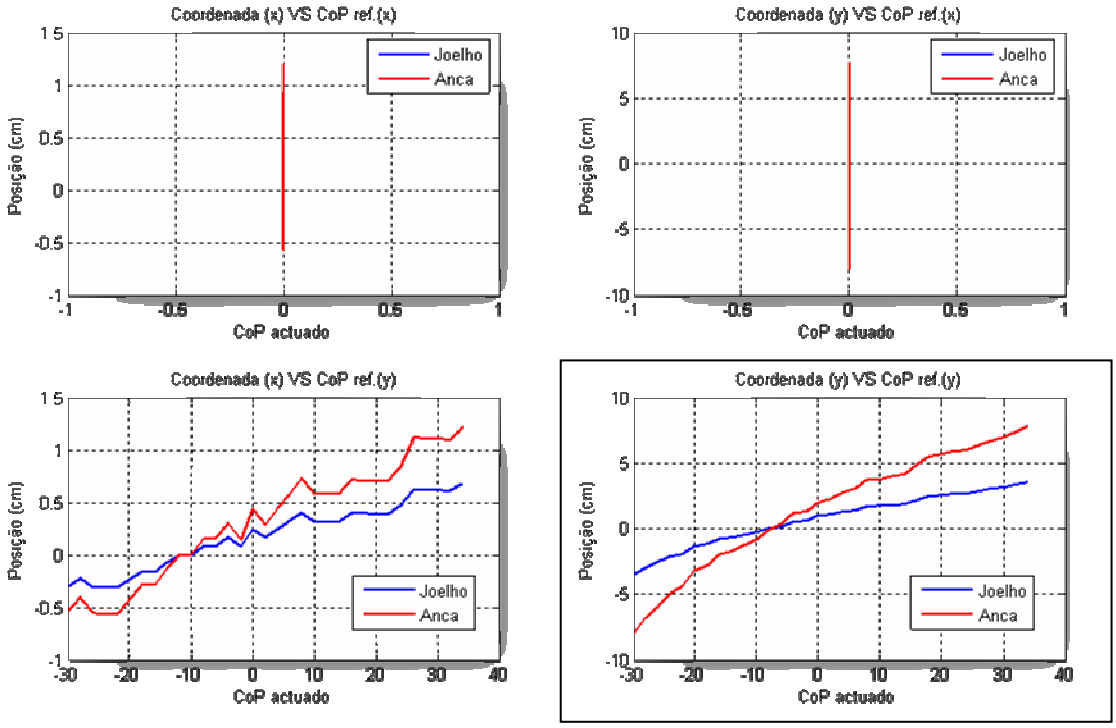


Fig. 59: Relação entre o CoP medido e a cinemática directa do joelho e da anca, ao longo do eixo *yy*.

C. Variação do CoP no eixo *xy*:

Efectuando agora 25 passos ao longo da diagonal entre os eixos xx e yy – CoP=(25,-25,0)→(-25,25,0) – amostraram-se os dados apresentados na Fig. 60 e Fig. 61.

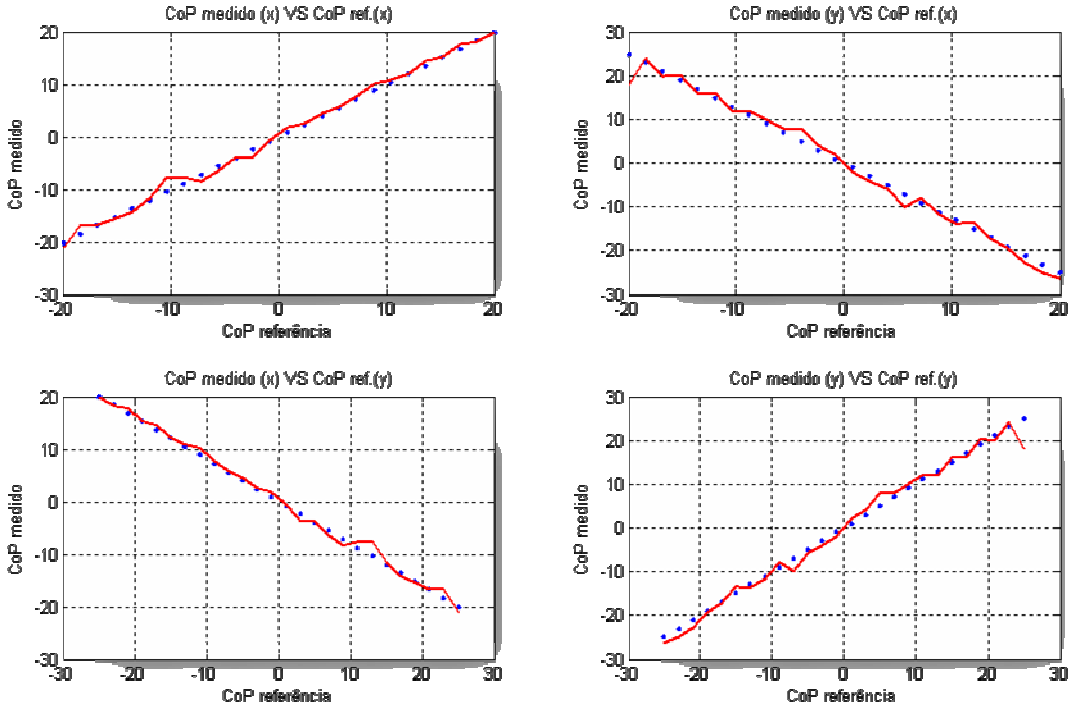


Fig. 60: Relação entre o CoP medido e o actuado, para 25 passos ao longo do eixo *xy*.

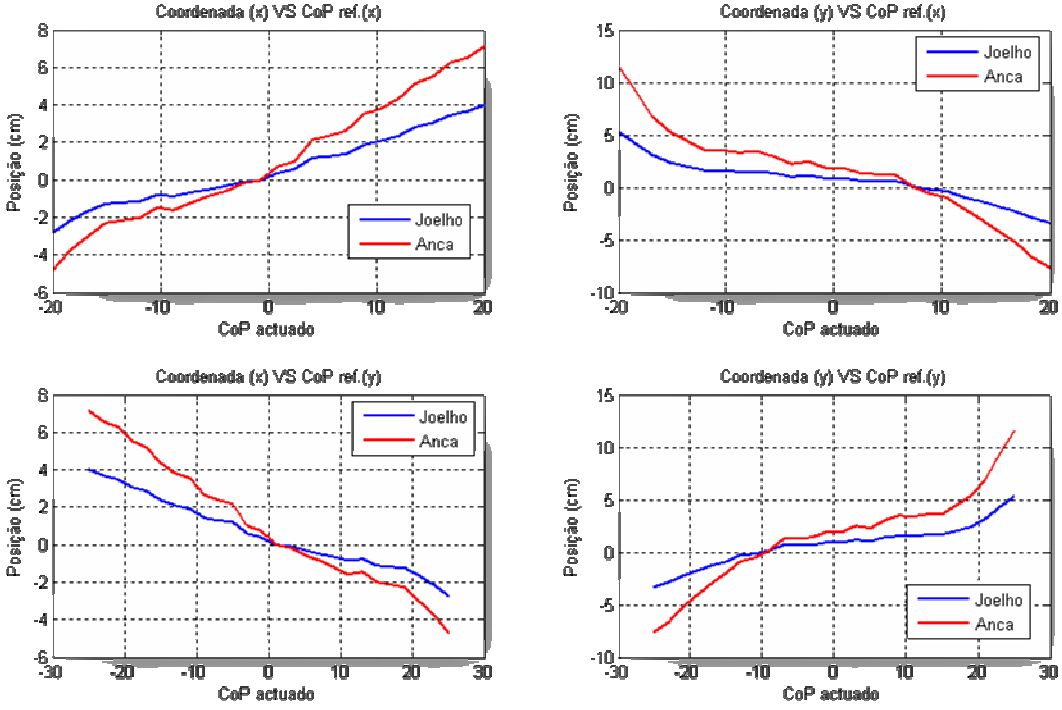


Fig. 61: Relação entre o CoP medido e a cinemática directa do joelho e da anca, ao longo do eixo *xy*.

4.4. Análise Dinâmica do CoP Referência

A análise dinâmica é entendida como a amostragem da resposta durante a aplicação de uma trajectória específica às diversas juntas do sistema. Na situação em concreto aplicar-se-á uma trajectória polinomial de quinta ordem (velocidade e aceleração inicial e final nulas) ao centro de pressão de referência (CoP_{ref}) e a saída dos sensores de força, bem como a posição angular das várias juntas, serão amostrados durante a execução da trajectória mencionada.

O procedimento das experiências executadas está descrito a seguir:

1. Inicialização do CoP_{ref} para a execução de uma trajectória polinomial de 5ª ordem;
2. Activação do controlador de equilíbrio;
3. Definição do período da trajectória e do ganho do controlador de equilíbrio;
4. Execução da trajectória para um determinado CoP_{ref} final;
5. Ao longo da trajectória registar a saída dos sensores de pressão e de posição dos servos;
6. No fim da trajectória, guardar os resultados sensoriais, e reposicionar a perna para o CoP_{ref} inicial;
7. Inicializar ganhos de compensação, desligar os controladores e sinais de PWM das juntas.

A. Variação do CoP no Eixo *xx*

Começando por executar uma trajectória polinomial desde as coordenadas (30,0,0) até à posição (-30,0,0) no que respeita ao centro de pressão, apenas iremos efectuar um movimento ao longo do eixo *xx*. O ganho escolhido para o controlador de equilíbrio é de 50, e o período da trajectória é de 2s. Por questões de análise dos dados, foi feita uma amostragem de 4 segundos tendo em vista o estudo da resposta em regime transiente e estacionário. A Fig. 62 apresenta os resultados da experiência.

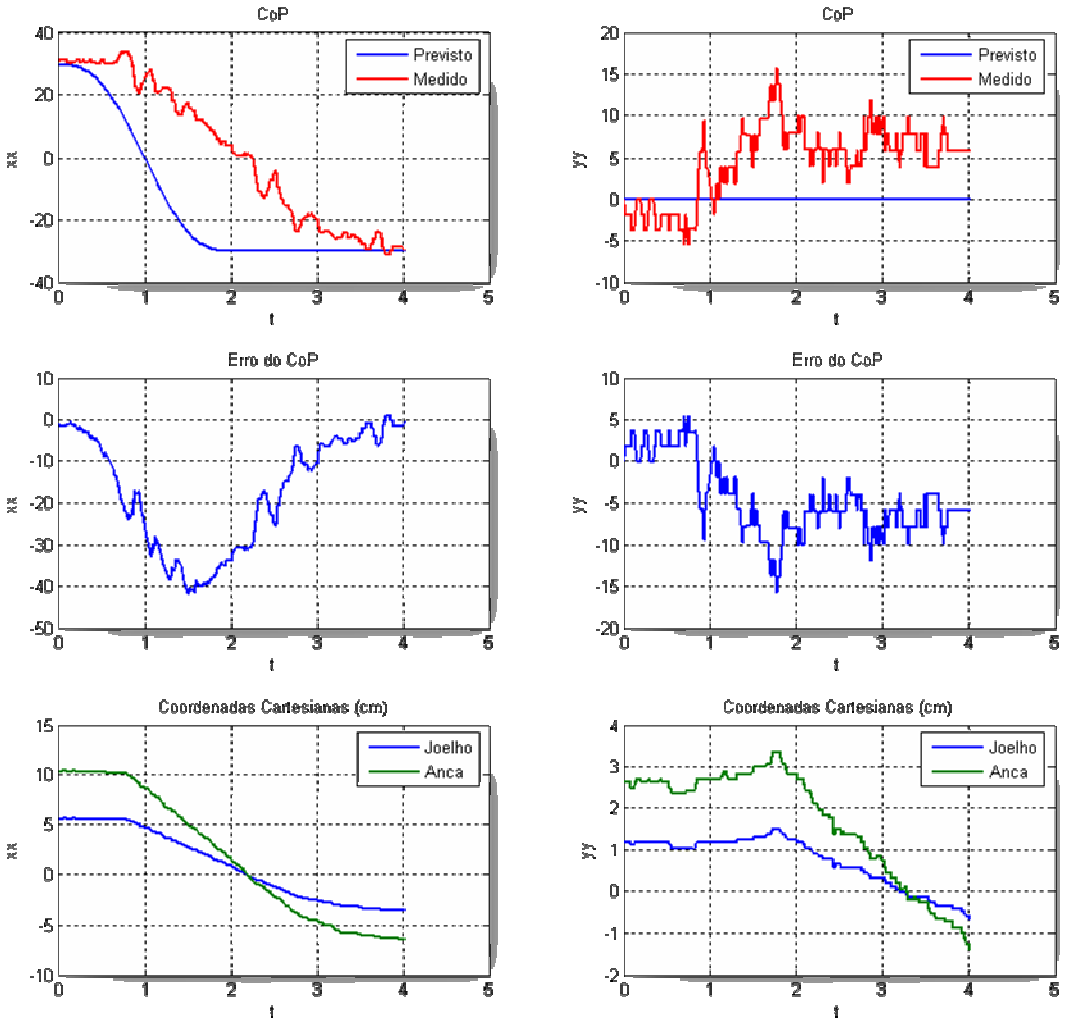


Fig. 62: Resposta ao polinómio aplicado no eixo *xx* com K=50.

Como se pode observar na Fig. 62, a resposta ao polinómio, segundo o eixo *xx*, apresenta um elevado atraso, com um erro máximo de 40 unidades para t=1.5s. O tempo de estabelecimento também apresenta um valor elevado, apenas atingindo o valor final para t=4s. Embora não fosse de esperar variações significativas na componente *yy* do CoP, dado o movimento ser efectuado no eixo *xx*, mesmo assim registaram-se variações com um erro máximo de -16 unidades o que revela alguma propagação destes movimentos para o eixo ortogonal.

B. Variação do CoP no Eixo *yy*

Repetindo a mesma experiência, mas agora aplicando um polinómio ao longo do eixo *yy* (Fig. 63) – CoP_{ref}=(0,-25,0) até (0,35,0) – verifica-se um comportamento muito semelhante: elevado tempo de atraso e de estabelecimento na componente *yy*, com um erro máximo de 45 unidades e um tempo de estabelecimento de 4 segundos, bem como também algumas flutuações na componente *xx* muito embora mais reduzidas que no caso anterior (±5 unidades). As condições da experiência – ganho de controlo e período de trajectória – mantiveram-se.

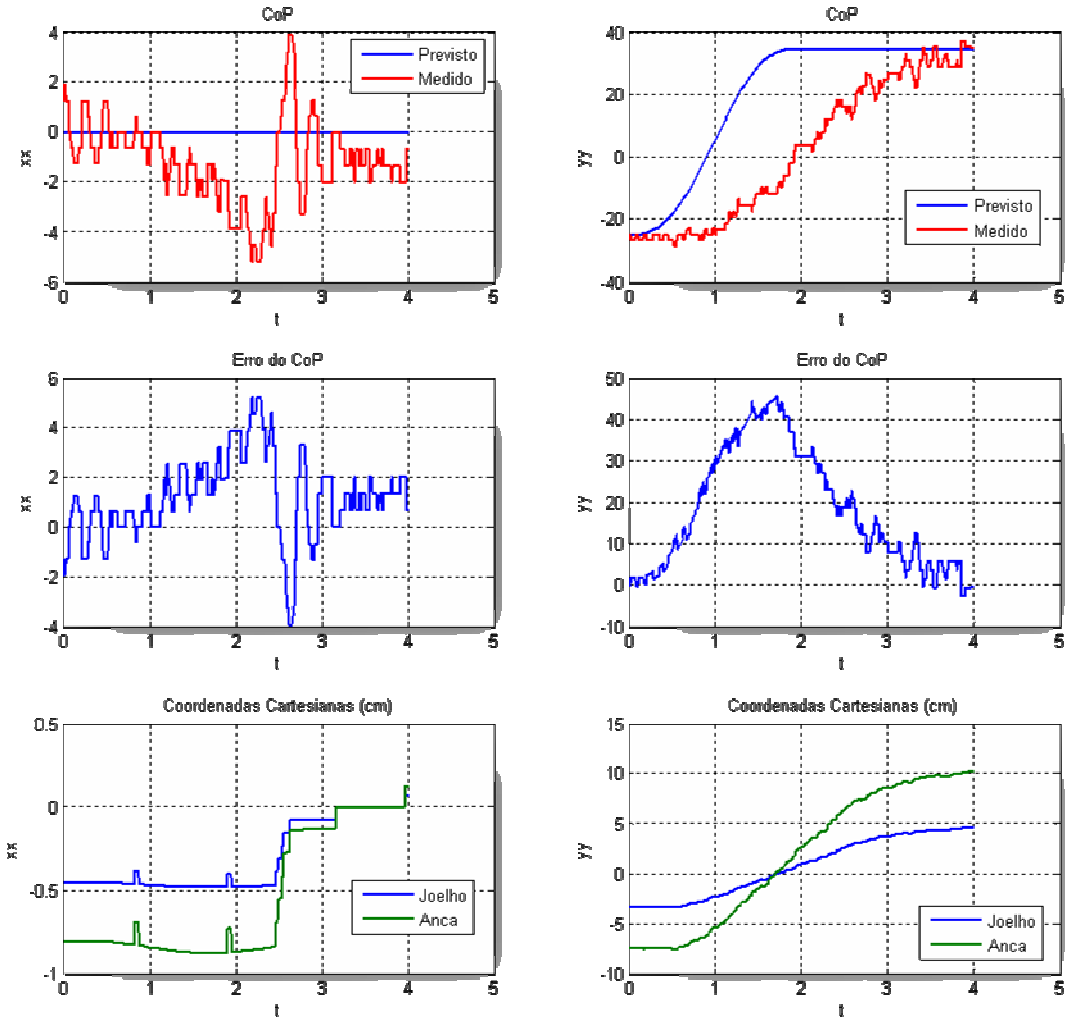


Fig. 63: Resposta ao polinómio aplicado no eixo *yy* com K=50.

C. Variação do CoP no Eixo *xy*

Aplicando a mesma trajectória polinomial na diagonal entre os eixos *xx* e *yy*, desde as coordenadas CoP_{ref}=(25,-25,0) até (-25,45,0) (Fig. 64), obtiveram-se praticamente os mesmos tempos de estabelecimento com 3 segundos para a componente *xx* e 4 segundos para a *yy*, mas com algumas diferenças no que respeita ao erros de CoP comparativamente às circunstâncias em que cada eixo era utilizado de forma independente.
Como os sinais de erro obtidos para cada componente testada individualmente (*xx* e *yy*) apresentam sinais opostos, a conjugação das duas trajectórias, deveria resultar num sinal de erro mais reduzido pois estes sinais tenderiam a anular-se. Tal de facto observa-se na componente *xx*, que quando tínhamos erros de mais de 40 unidades, agora foi reduzido para 35. Esta diferença bate certo com os resultados experimentais da Fig. 62, pois as flutuações introduzidas pelo movimento no eixo *yy* deveriam subtrair cerca de 5 unidades a este sinal, que é o que de facto se observa.
Contudo o erro na componente *yy* deixa de fazer sentido, pois neste caso aumentou de 45 para 60 unidades (Fig. 64), quando deveria baixar 15 unidades para o valor 30 (ver Fig. 63). A razão para este evento ainda não é muito clara e merece alguma reflexão, pois é do interesse em reduzir ao máximo possível o erro de *tracking* da trajectória prevista.

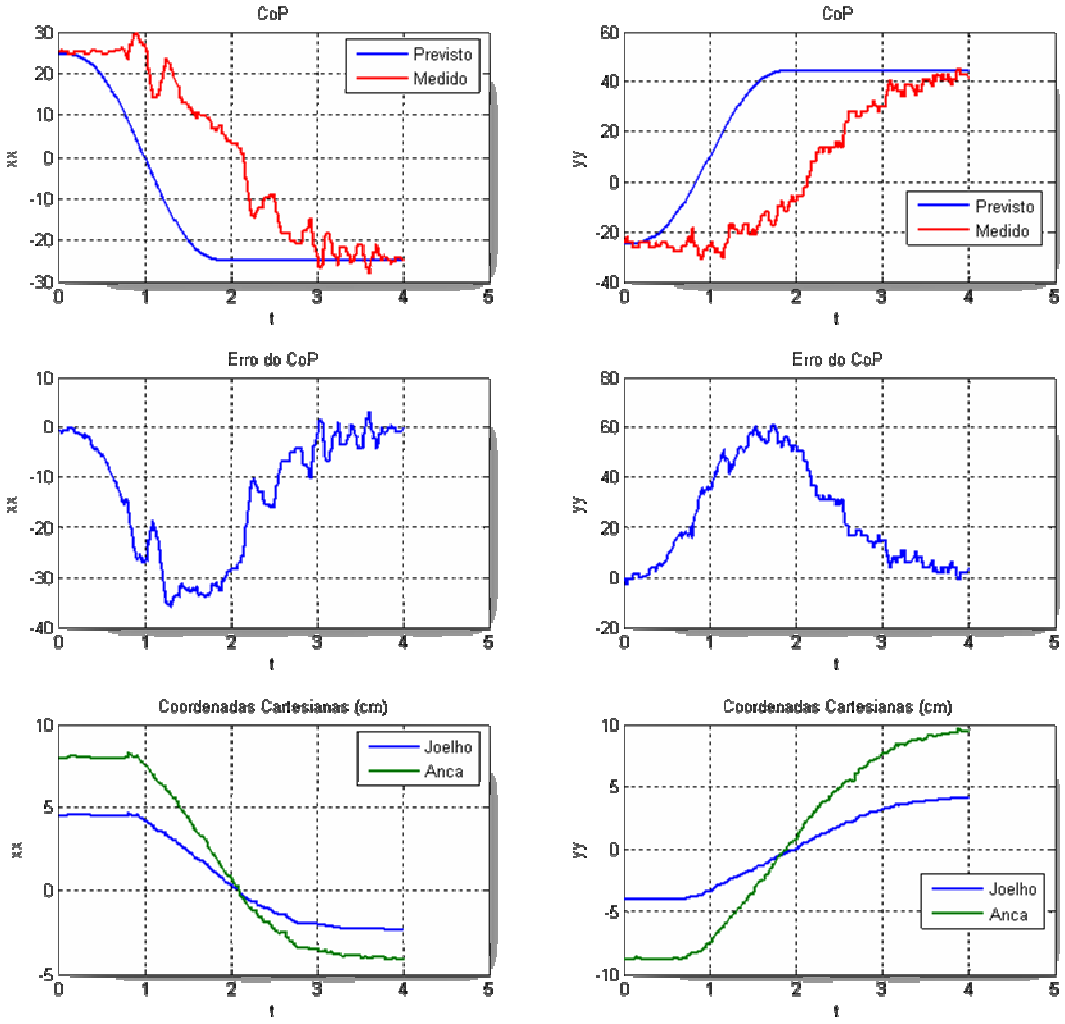


Fig. 64: Resposta ao polinómio aplicado nos eixos *xy* com K=50.

Tendo em vista a redução do sinal de erro, modificaram-se alguns parâmetros como é o caso do ganho do controlador de equilíbrio e o período da trajectória.

Diminuindo o ganho do controlador de 50 para 30 (Fig. 65), evidencia-se um aumento dos tempos de estabelecimento – na componente *xx* aumentou de 3 para 3.5 segundos, e em *yy* subiu além dos 4 segundos – bem como igualmente um aumento do erro para a resposta na componente *xx*: 35 para 40 unidades. O sinal de erro na componente *yy* manteve-se constante.

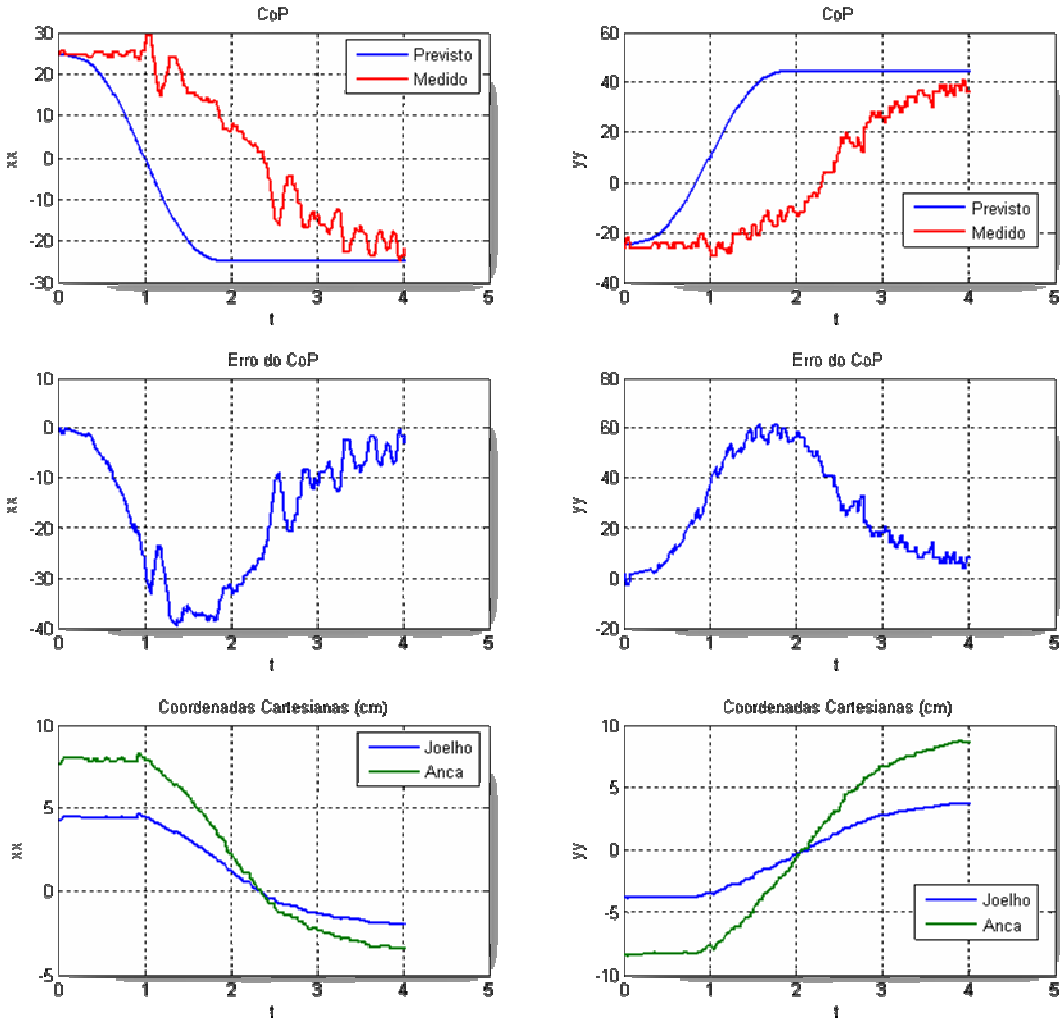


Fig. 65: Resposta ao polinómio aplicado nos eixos *xy* com K=30.

Se em vez de diminuirmos o ganho, aumentarmo-lo para K=70 (Fig. 66) verifica-se o oposto: os tempos de estabelecimentos tendem a diminuir – 3 segundos para a componente *xx* e 3.5 segundos para *yy* – e o erro de *tracking* também tende a ficar mais pequeno: 30 unidades para *xx*, e 58 unidades para *yy*.

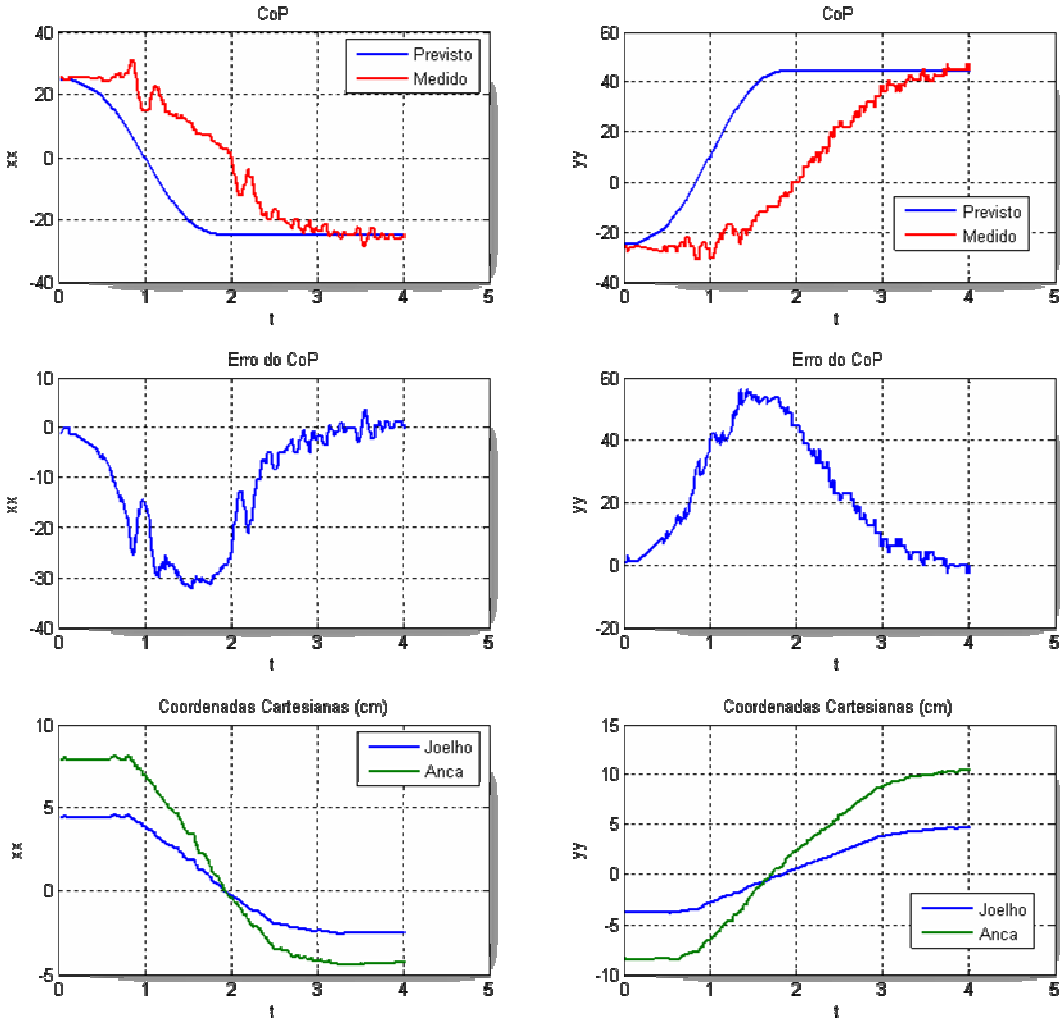


Fig. 66: Resposta ao polinómio aplicado nos eixos *xy* com K=70.

Conclui-se assim que o aumento do ganho de compensação do controlador de equilíbrio tende a melhorar o tempo de estabelecimento e o erro de *tracking*. Contudo, existe um limite para o qual o sistema começa a instabilizar observando-se significativas vibrações. Este fenómeno começa a ser observado precisamente para o último ganho testado (K=70), pelo que poucas possibilidades há para melhorar o erro que em si ainda se apresenta um tanto longe para possuir valores satisfatórios.

Se em vez de manipular o ganho de compensação, variar a velocidade da trajectória, ou por outras palavras, o seu período, é possível que o erro possa ser melhorado. A Fig. 67 e a Fig. 68 exploram esta possibilidade.

A Fig. 67 apresenta o caso, em que o período da trajectória foi reduzido para apenas 1s – duplicou-se a velocidade. As consequências traduzem-se numa variação demasiado elevada para o sistema acompanhar, e por isso os tempos de estabelecimento e os erros máximos pioram para valores incomportáveis.

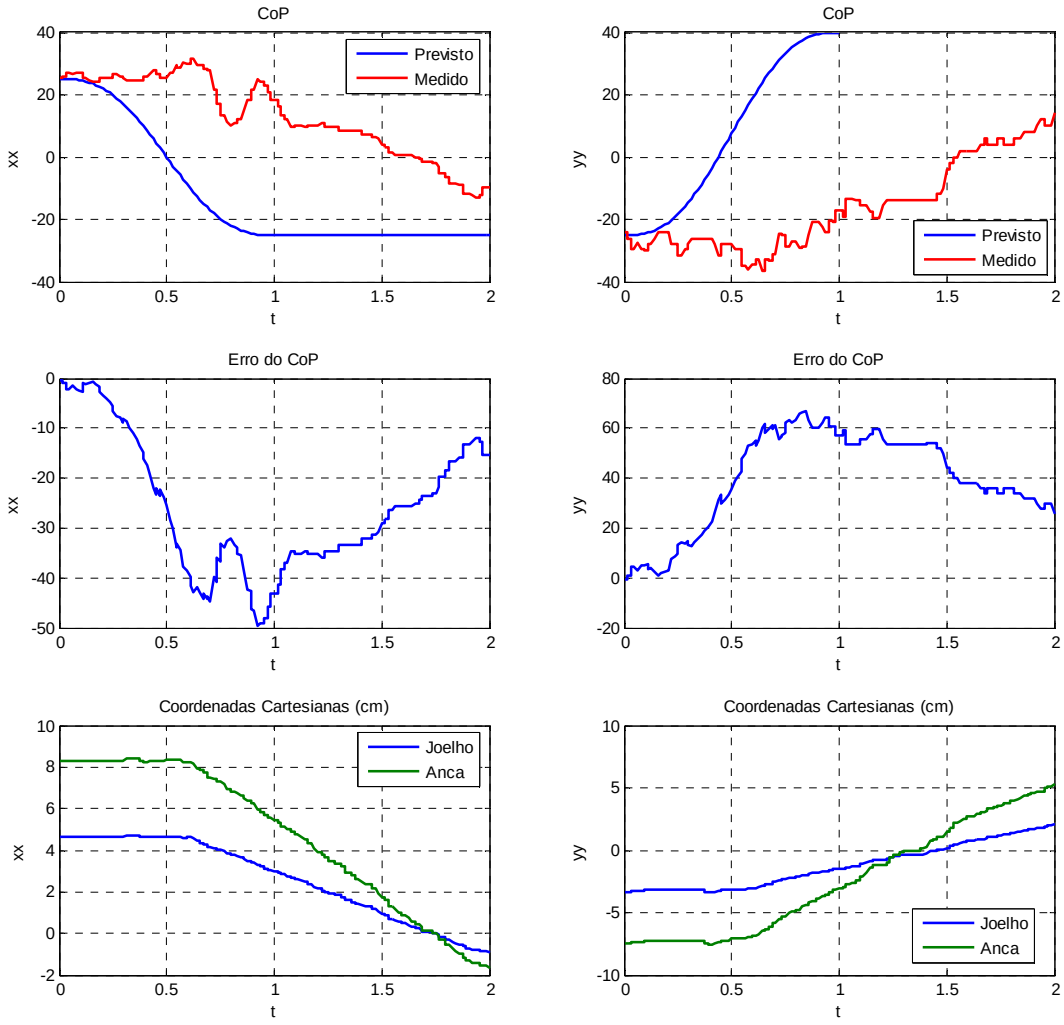


Fig. 67: Resposta ao polinómio aplicado de duração 1s (K=50).

Se, ao invés, aumentarmos o período da trajectória para o dobro – 4 segundos – (Fig. 68), as características da resposta melhoram em muito: o tempo de estabelecimento diminui para 4 segundos para as duas componentes (corresponde ao instante final da trajectória), e o erro diminui para 20 e 32 unidades para as componentes *xx* e *yy* respectivamente.

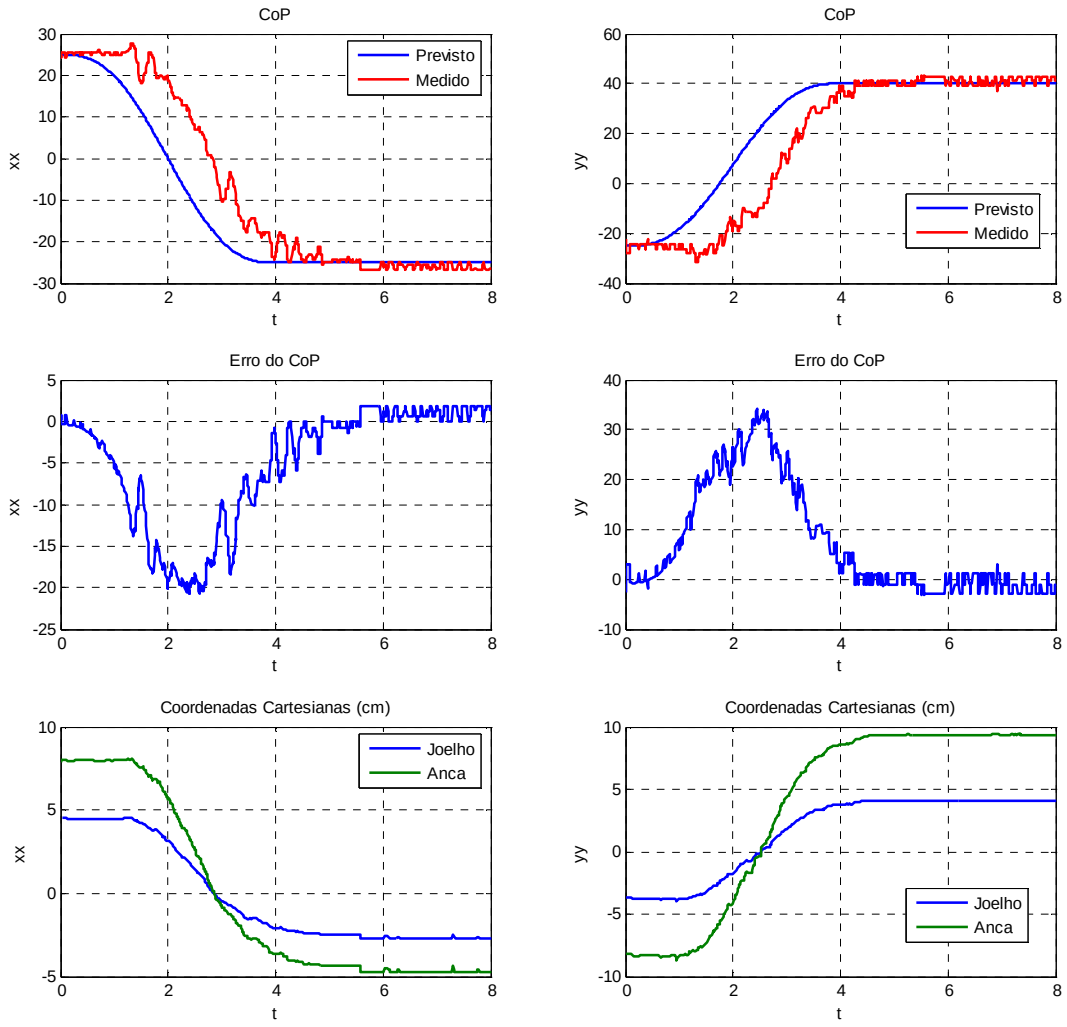


Fig. 68: Resposta ao polinómio aplicado de duração 4s (K=50).

Como conclusão podemos dizer que a melhor estratégia para melhorar ao máximo a resposta do sistema, é de usar ganhos de compensação (do controlador de equilíbrio) próximos do limite, sem provocar instabilidade, e aplicar velocidades as mais reduzidas possíveis. Como se pretende que o robot se locomova a uma velocidade bastante reduzida numa fase inicial, tais exigências não deverão constituir um problema para a implementação do controlador de equilíbrio.

4.5. Execução de Trajectórias Rectilíneas pela Perna de Suporte

Como complemento ao estudo dinâmico da resposta do sistema a um polinómio de quinta ordem, foram executadas experiências similares, mas com a repetição da mesma trajectória várias vezes para análise do comportamento do sinal do erro ao longo do tempo.

Cada experiência seguiu o seguinte procedimento:

1. Inicialização do CoP_{ref} para a execução de uma trajectória polinomial de 5ª ordem;
2. Activação do controlador de equilíbrio;
3. Definição do período da trajectória e do ganho do controlador de equilíbrio;
4. Execução da trajectória para um determinado CoP_{ref} final;
5. Esperar que o sistema termine a trajectória;
6. Execução uma trajectória para o CoP_{ref} inicial;
7. Esperar que o sistema termine a trajectória;
8. Repetir os passos 4 a 7 mais nove vezes.
9. No final, inicializar os ganhos de compensação e desligar os controladores, bem como os sinais de PWM.

Ao longo da execução das trajectórias, os sensores de pressão e as posições angulares dos servomotores são registados para um ficheiro para posterior análise.
De modo a fazer uma comparação válida com as trajectórias previstas, foram também registados os instantes de tempo inicial e final de cada trajecto, de modo a que, aquando da comparação, os polinómios de quinta ordem previstos sejam mapeados no tempo de forma correcta.

Finalmente os resultados foram dispostos de forma a analisar o CoP medido e a correspondente cinemática directa do joelho e da anca ao longo do tempo, comparando os valores esperados com os registados.

Todas as experiências executadas nesta secção consideraram as seguintes condições de funcionamento:

- Controlador de equilíbrio activado, e com ganho de compensação igual a 50;
- Controlador local PID desactivado;
- Período das trajectórias igual a 2 segundos.

A. Trajectória ao longo do eixo *xx*

Executando trajectórias rectilíneas ao longo do eixo *xx* desde as coordenadas CoPref=(-30,0,10) até (+30,0,10), podemos observar na Fig. 69 que o sinal de erro mantém-se estabilizado na gama entre +20 e -20 unidades do centro de pressão medido, com +20 no sentido positivo da trajectória e -20 no sentido oposto.

A cinemática directa também se apresenta estabilizada, sem a presença de acumulação de erros, contida entre ±3cm para o joelho, e ±5cm para a anca (segundo o eixo *xx*).

Analisando o eixo ortogonal *yy*, que supostamente não deveria apresentar variações, verifica-se algumas flutuações tal como visto na secção anterior, mas contidas entre ±15 unidades do centro de pressão.

A cinemática directa também apresenta flutuações compreendidas entre 1 e 3cm para o joelho, e entre -3 e +2cm para a anca. Estas flutuações caracterizam-se por uma forte dependência com o movimento no eixo ortogonal *xx*, e não podem de forma nenhuma ser ignorados.

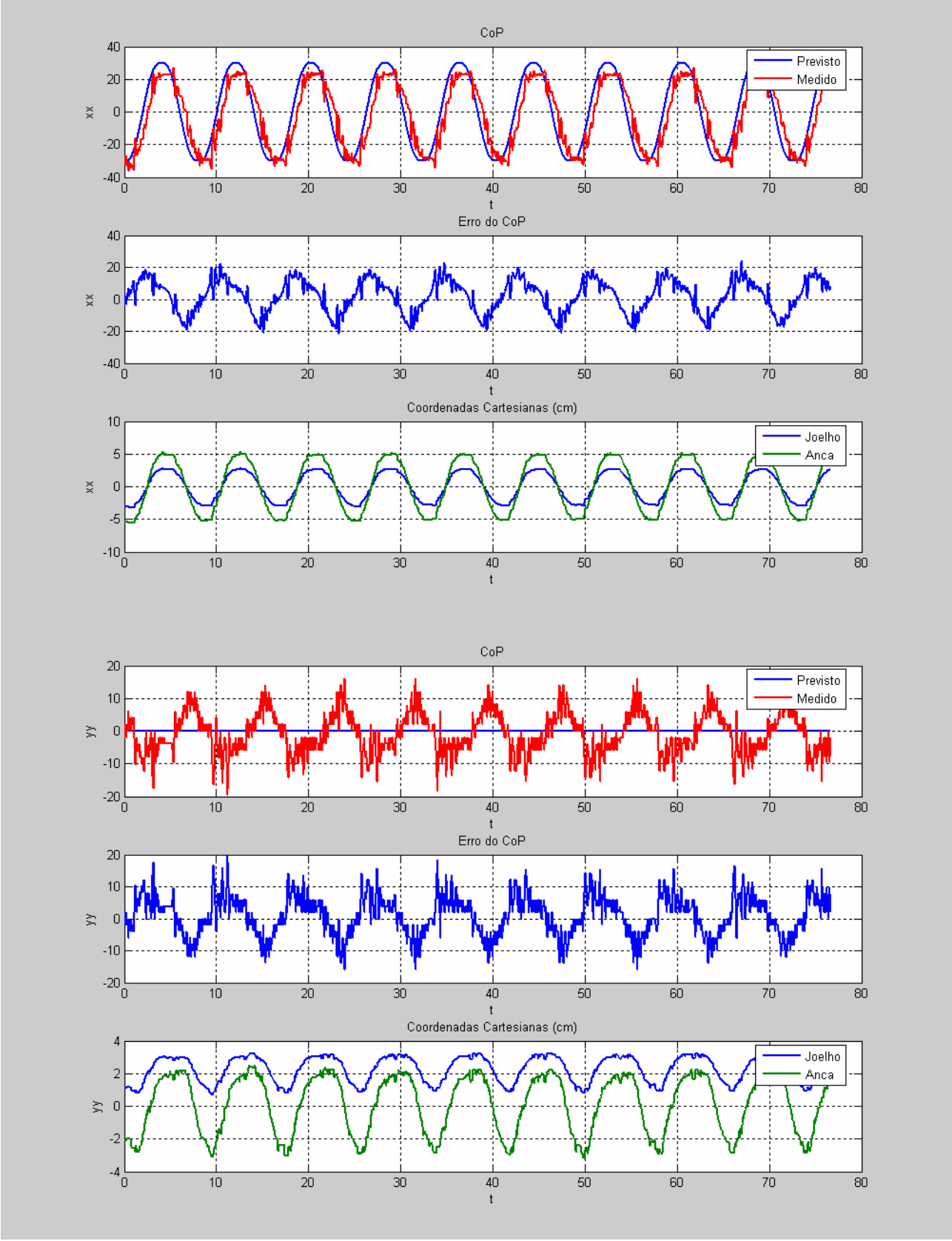


Fig. 69: Trajectória rectilínea ao longo do eixo *xx* (Período=2s e K=50).

B. Trajectória ao longo do eixo *yy*

Repetindo as mesmas experiências, mas agora movendo o sistema apenas segundo o eixo *yy*, entre as coordenadas CoP_{ref}=(0,-60,10) a (0,40,10), podemos observar erros de tracking na ordem das 30 unidades

para a componente *yy* na qual está a ser aplicado o movimento. Já na componente ortogonal verificam-se flutuações que podem atingir as 15 unidades. Contudo em termos de cinemática directa, a perna apenas se desloca no máximo 1 cm, o que representa uma baixa interferência que os movimentos segundo o eixo *yy* induzem no eixo *xx*.

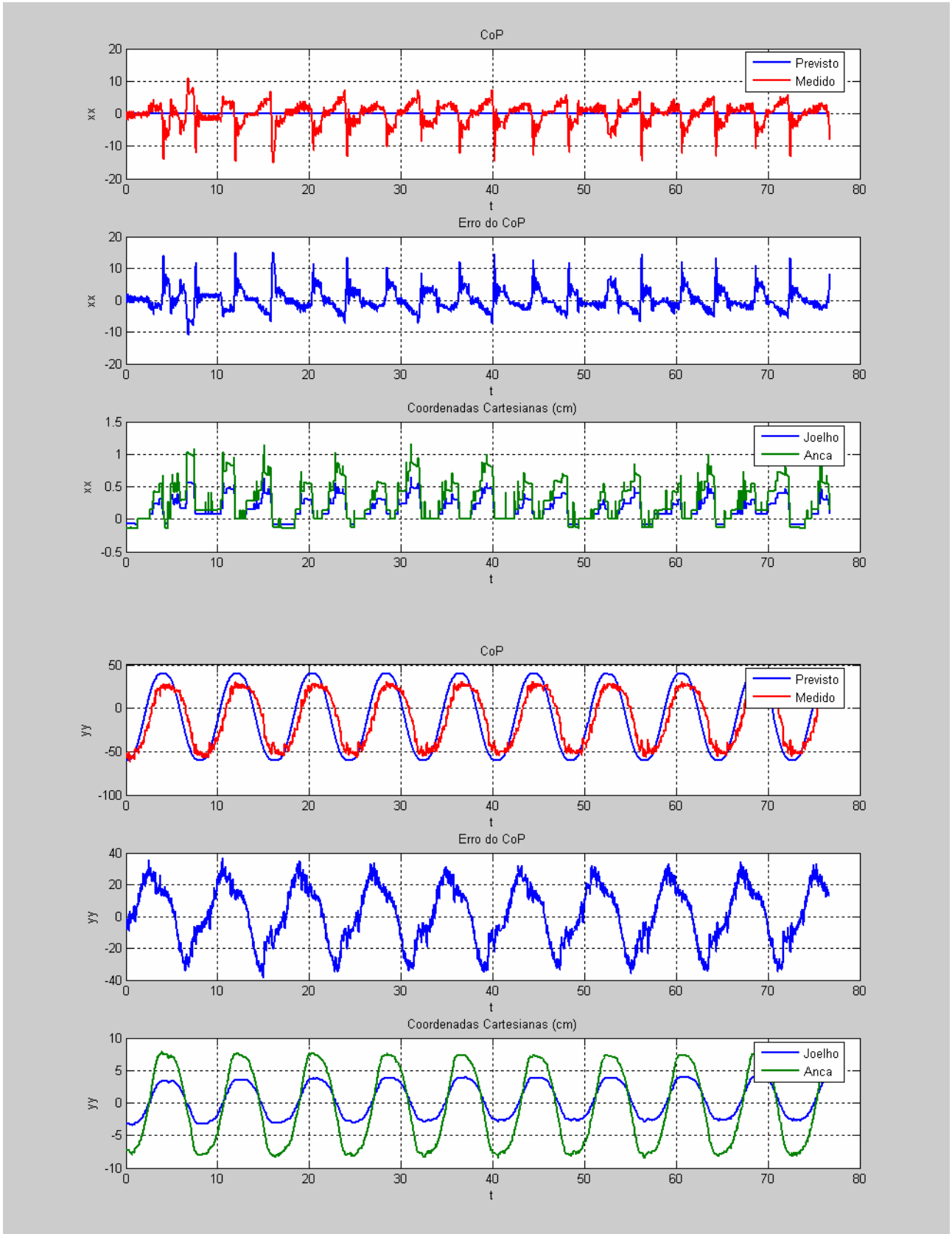
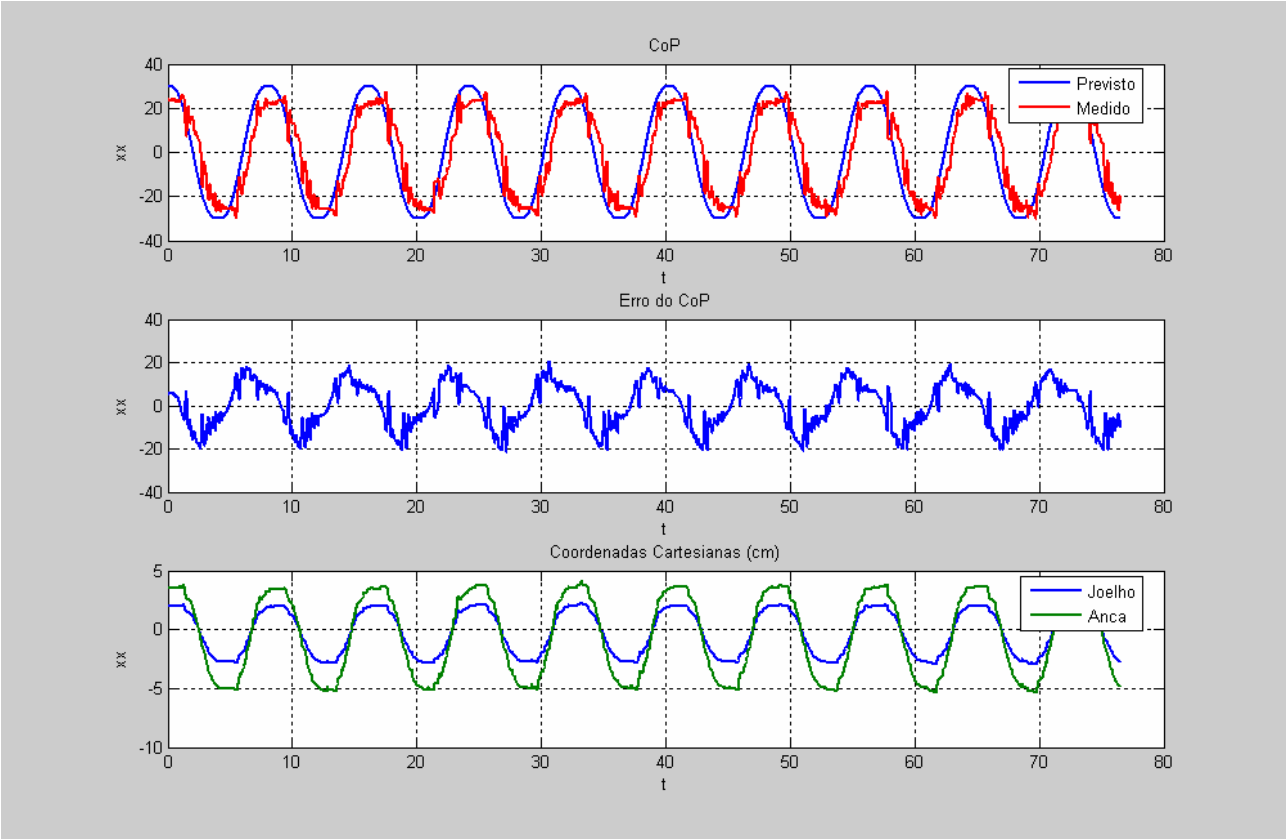


Fig. 70: Trajectória rectilínea ao longo do eixo *yy* (Período=2s e K=50).

Como que resumindo, os dados apontados pelos movimentos em cada um dos eixos mostram uma sensibilidade muito maior no eixo *yy* no que respeita às flutuações sensoriais. Tal pode ser induzido pela introdução dos *offsets* no cálculo do centro de pressão a partir das saídas dos sensores de força (normalizados a 128) (ver secção III. 1.5). Por um lado possuem a vantagem de assemelhar as sensibilidades de cada componente do centro de pressão, mas trazem consigo a desvantagem de introduzir maiores flutuações no sinal de erro de *tracking*. Infelizmente estamos perante um dilema: quanto maior for este *offset*, maior é a reactividade do controlador, sob pena de maiores oscilações do sinal de erro; e quanto menor for, maior é a oposição ao ruído, mas a resposta do controlador torna-se lenta. Uma solução de compromisso torna-se necessária. Percursos com ganho de compensação de valor 50, e períodos de trajectória superiores a 2 segundos são recomendados para minimização do sinal de erro.

C. Trajectória ao longo dos eixos *xy*

Executando uma trajectória rectilíneo que abranja tanto o eixo *xx* como o *yy* – CoP=(30,-60,10) até (-30,40,10) – revelam o superior atraso na componente *yy* (30 contra 20 unidades), tal como foi observado nas figuras anteriores. Uma solução de compromisso, derivado do dilema reactivade/ruído, é necessário para um bom equilíbrio.



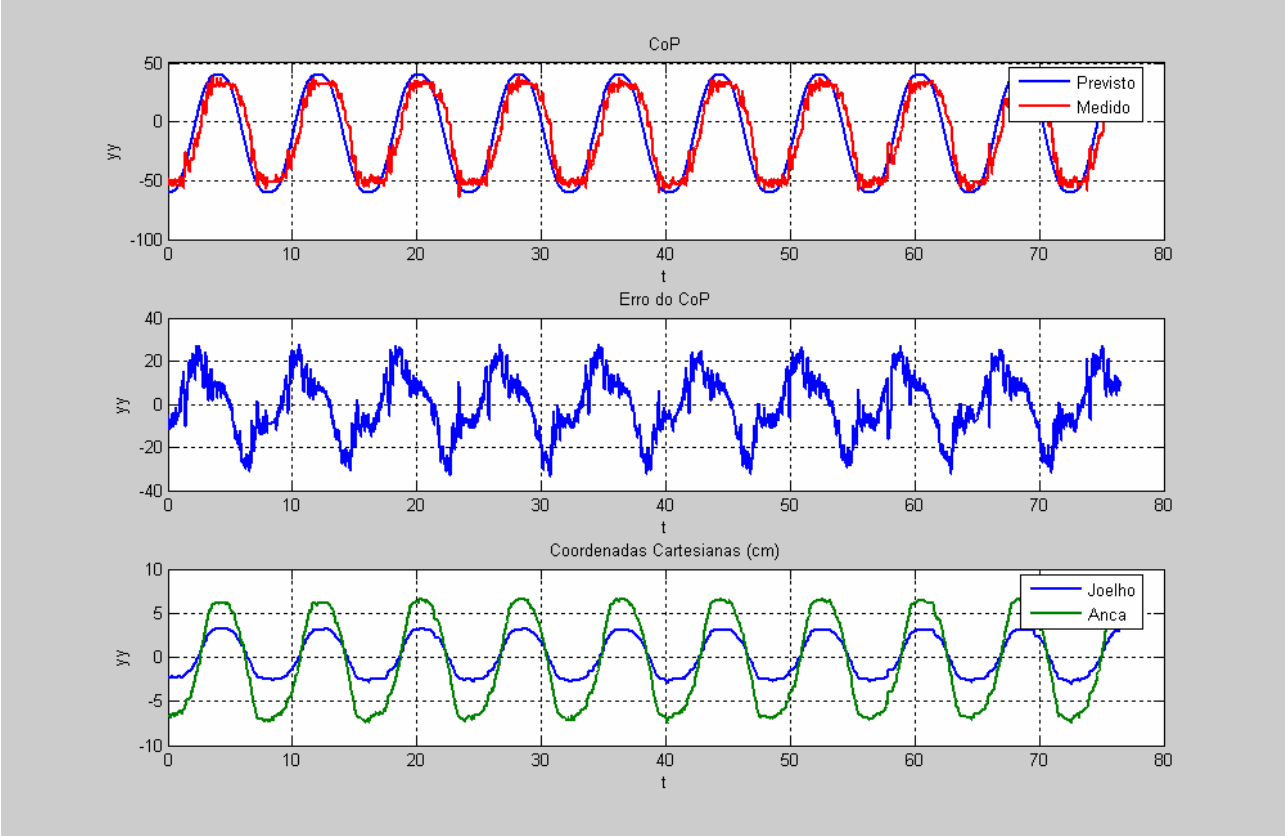


Fig. 71: Trajectória rectilínea ao longo dos eixos *xy* (Período=2s e K=50).

4.6. Execução de Trajectórias Elípticas pela Perna de Suporte

Executando trajectórias elípticas, especificando vários pontos de passagem segundo um formato elíptico, e com uma duração temporal entre cada par, podemos visualizar a resposta do sistema na Fig. 72. Pode-se confirmar mais uma vez a superior sensibilidade ao ruído por parte da componente *yy*.

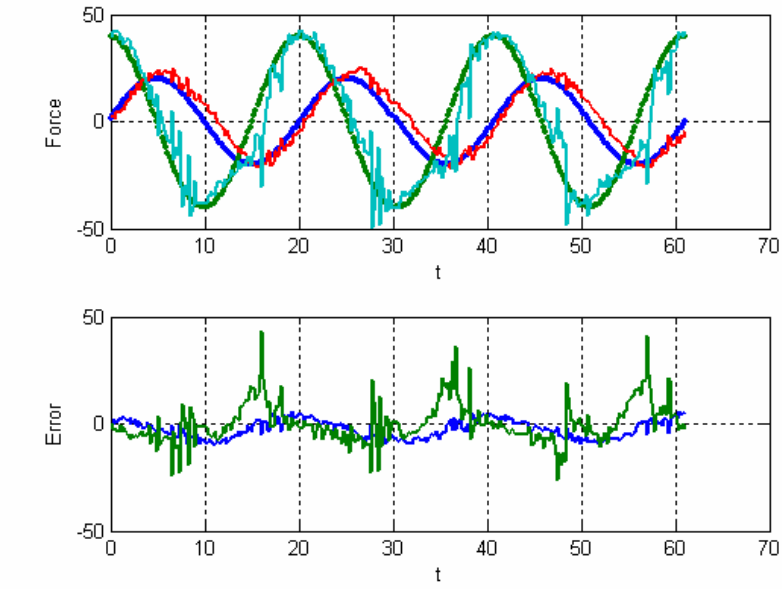


Fig. 72: Variação do CoP ao longo do percurso elíptico - componentes x (azul) e y (verde) - em comparação com o real, e o respectivo erro.

Um pormenor interessante a observar na Fig. 72 é o facto do erro na componente *xx* ser máximo quando o CoP_{ref} se encontra nas extremidades do pé (componente *yy* nos extremos). Efectuando uma análise espacial, em vez de temporal, podemos ver esta anomalia mais evidentemente na Fig. 73 e na Fig. 74, segundo a perspectiva 3D e usando o eixo *zz* como referência, respectivamente.

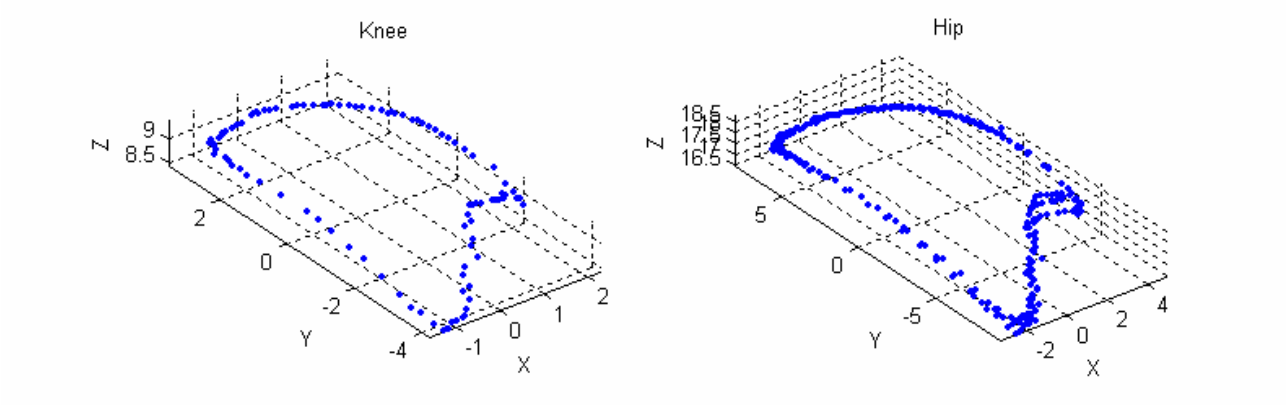


Fig. 73: Cinemática directa do joelho e da anca (vista 3D).

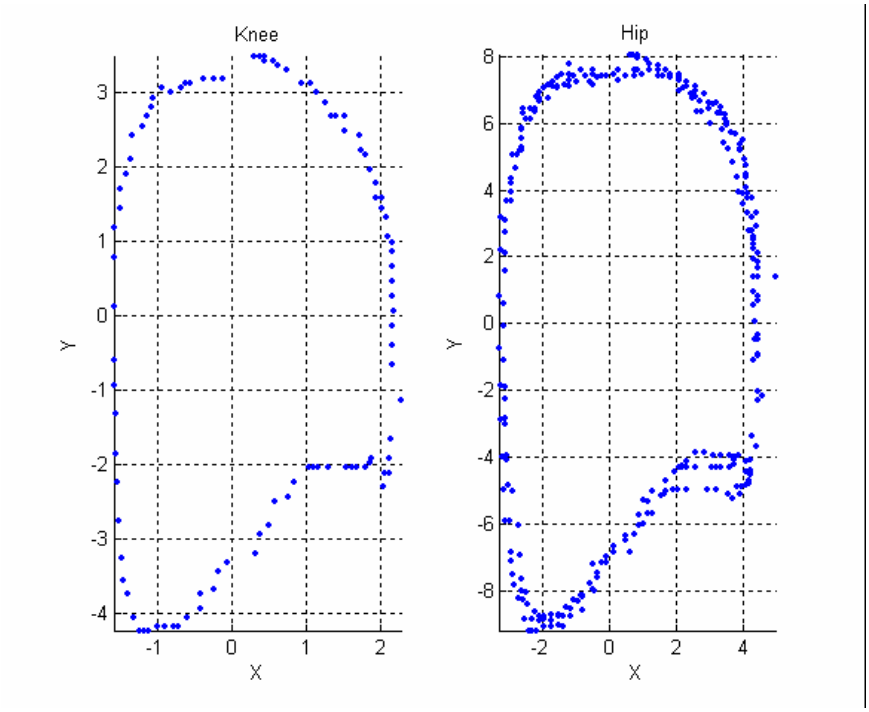


Fig. 74: Cinemática directa do joelho e da anca (vista 2D de cima).

De facto, a trajectória real do centro de pressão afasta-se da elipse projectada, no canto inferior direito. Tal anomalia pode-se dever a diversos factores, sobretudo físicos, tais como:

- Baixa sensibilidade do sensor de força, podendo ser causada pelo próprio sensor, ou pela placa de acrílico subjacente com um nível de rigidez superior ao normal;
- Saturação da saída sensorial provocada pelo circuito de acondicionamento de sinal. Se for este o caso, o potenciómetro de ajuste do *offset* deverá ser regulado.

4.7. Execução de Trajectórias Rectangulares pela Perna de Suporte

Experimentando o formato rectangular, definiram-se quatro pontos de referência do centro de pressão correspondentes aos quatro vértices, e aplicaram-se sucessivamente em instantes de tempo periódicos, até completarem 10 voltas.

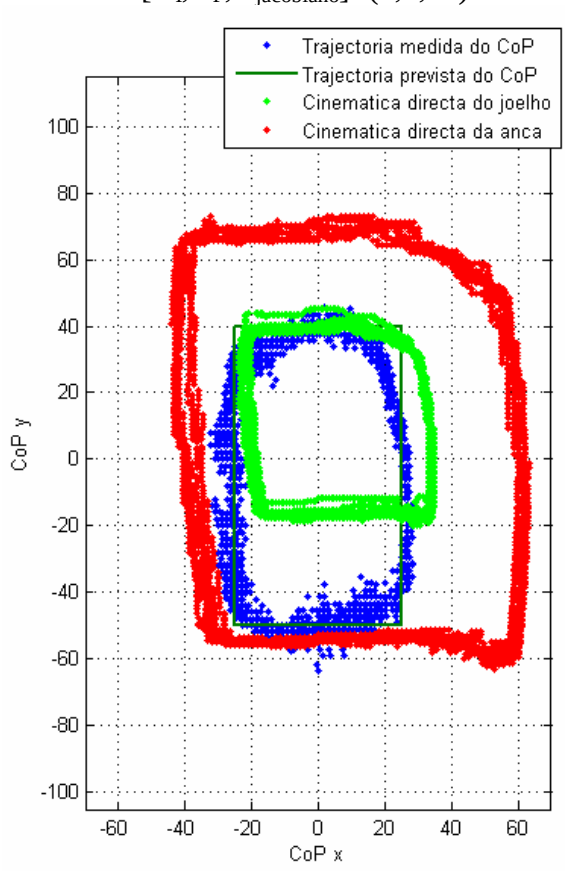
A Fig. 75, apresenta a execução de 10 trajectos rectangulares, cujas arestas são compostas por trajectórias polinomiais de quinta ordem com um período de execução de 4 segundos. A visualização é do tipo espacial, podendo observar o trajecto do CoP esperado (a preto), o CoP medido (pontos a azul), e a cinemática directa do joelho (verde) e da anca (vermelho) vista do eixo *zz*.
Os diferentes gráficos da Fig. 75 mostram o efeito do controlador local variando os seus parâmetros de controlo – ganho de integração KI e proporcional KP – para um ganho fixo do controlador de equilíbrio (K=30).

Começando pelo controlo em malha aberta (sem controlo local), podemos observar o seguimento do trajecto proposto com alguma exactidão, mas ao nível dos vértices a dificuldade de seguimento aumenta conferindo ao trajecto medido um formato semelhante ao elíptico. Ligando o controlo local e aumentando os seus parâmetros, observa-se um seguimento mais exacto, mas ao mesmo uma densidade de pontos (de CoP) muito mais intensa em torno do trajecto aplicado. Tal só pode evidenciar a presença de oscilações resultante da instabilidade do controlador: quanto maiores os parâmetros de controlo, maior a instabilidade.

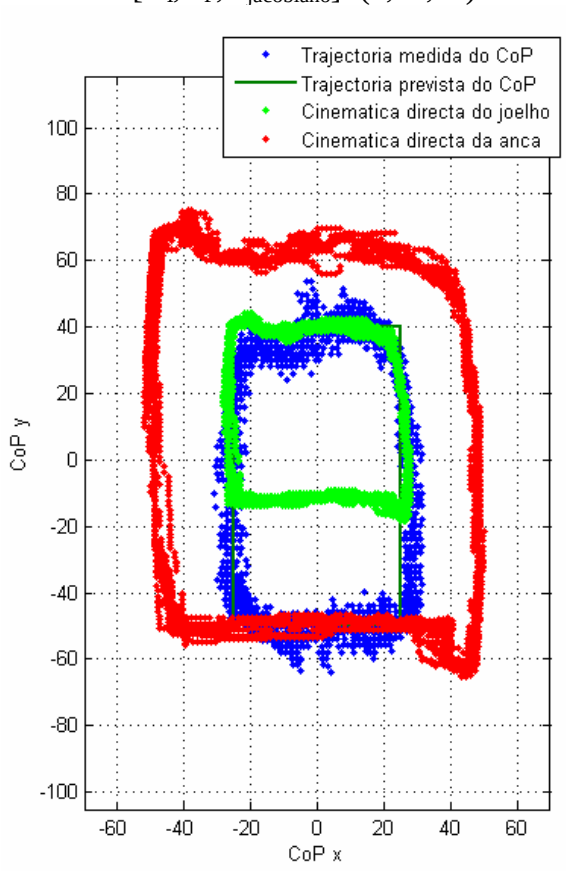
Efectuando os mesmos trajectos, mas agora com um aumentado ganho no controlador de equilíbrio (K=50) (ver Fig. 76), pode-se observar as mesmas características evidenciadas, mas com o surgimento mais precoce das oscilações. Tal é consistente com o esperado, pois o aumento do ganho do controlador de equilíbrio, tende a aumentar a instabilidade.

A Fig. 78 propõe variar a velocidade para um conjunto fixo de parâmetros de compensação. Para $[K_I,K_P,K_{jacobiano}]=(5,40,30)$, observa-se que quanto maior é a velocidade, mais instável é o seguimento de trajectória. Por observação o período de 4 segundos aparenta ser o melhor para a execução de trajectórias rectangulares.

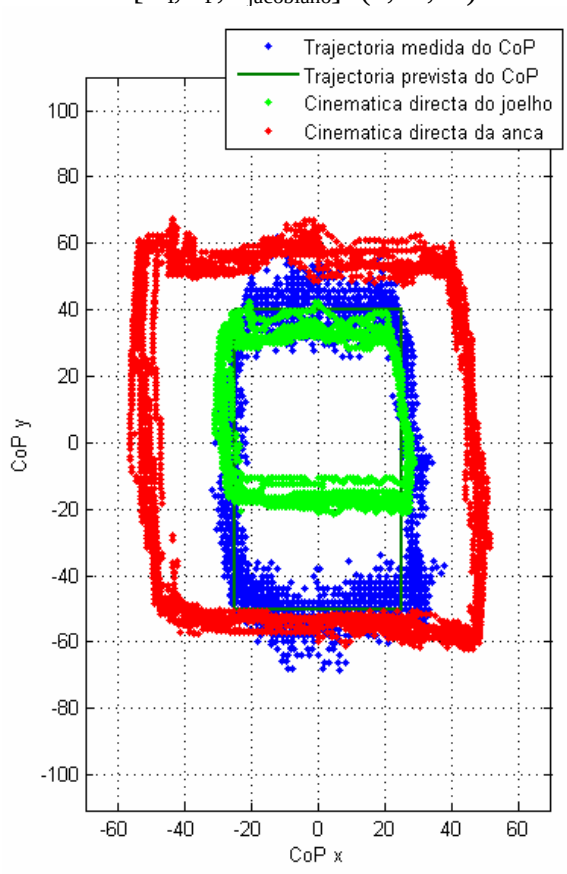
$[K_I,K_P,K_{jacobiano}]=(0,0,30)$



$[K_I,K_P,K_{jacobiano}]=(5,40,30)$



$[K_I,K_P,K_{jacobiano}]=(7,40,30)$



$[K_I,K_P,K_{jacobiano}]=(10,40,30)$

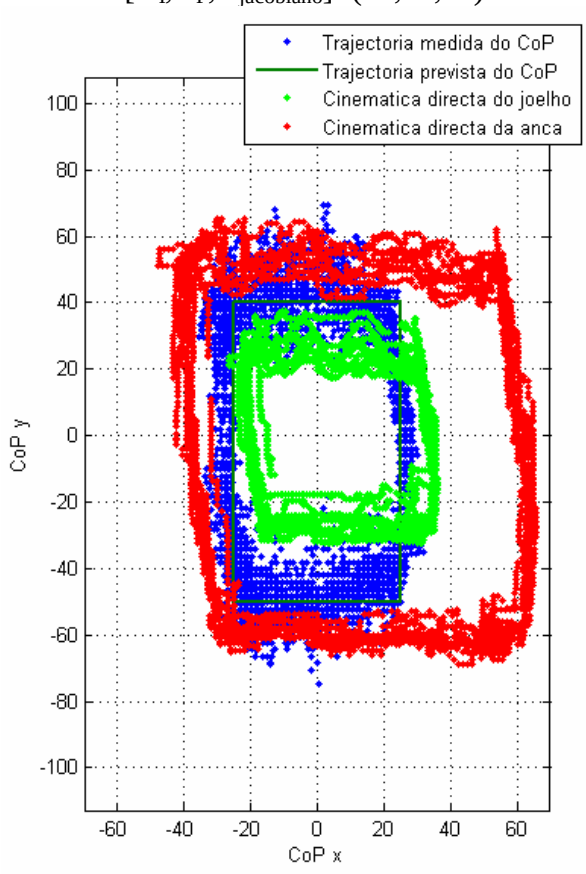


Fig. 75: Resposta a 10 voltas de uma trajectória rectangular para vários parâmetros do controlador local.

Sem Controlo Local: $[K_I,K_P,K_{jacobiano}]=(0,0,50)$ Com Controlo Local: $[K_I,K_P,K_{jacobiano}]=(5,40,50)$

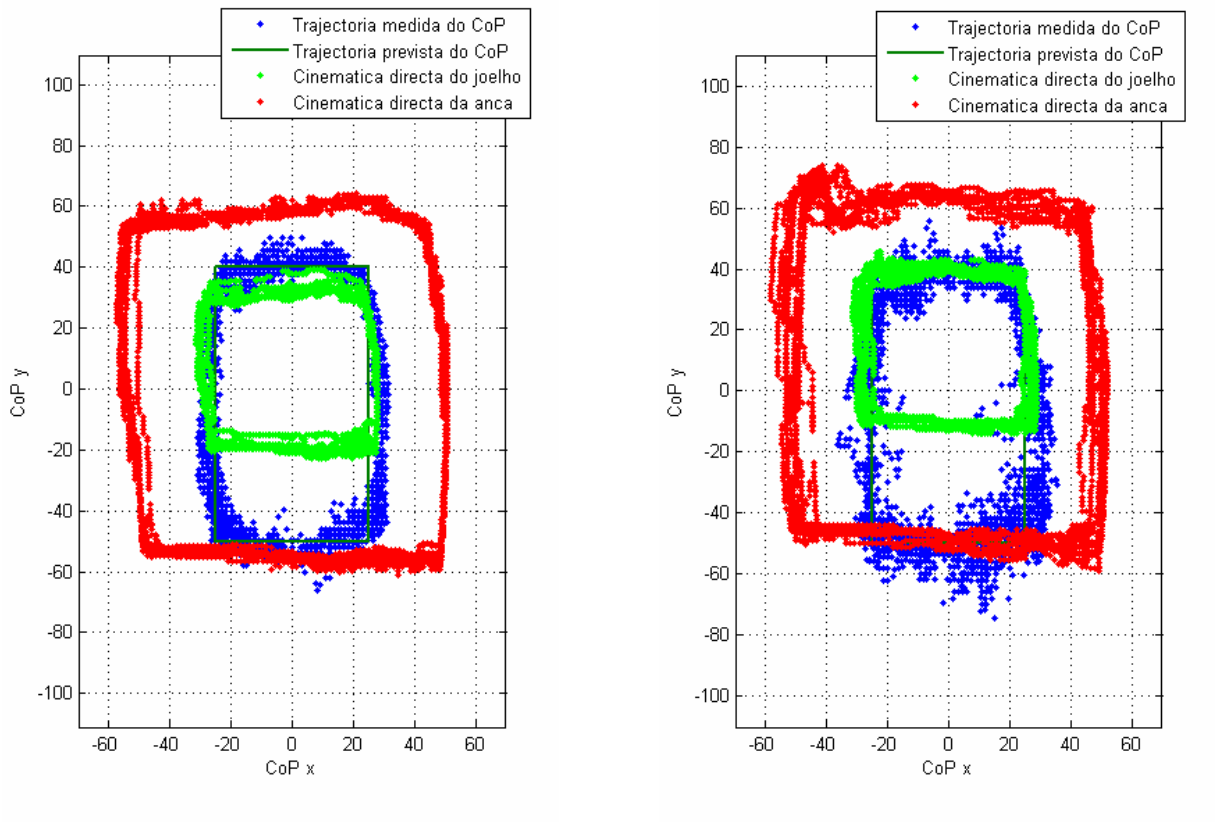


Fig. 76: Resposta a 10 voltas de uma trajectória rectangular sem e com controlo local ($K_{jacobiano}$=50).

Período=4s Período=2s

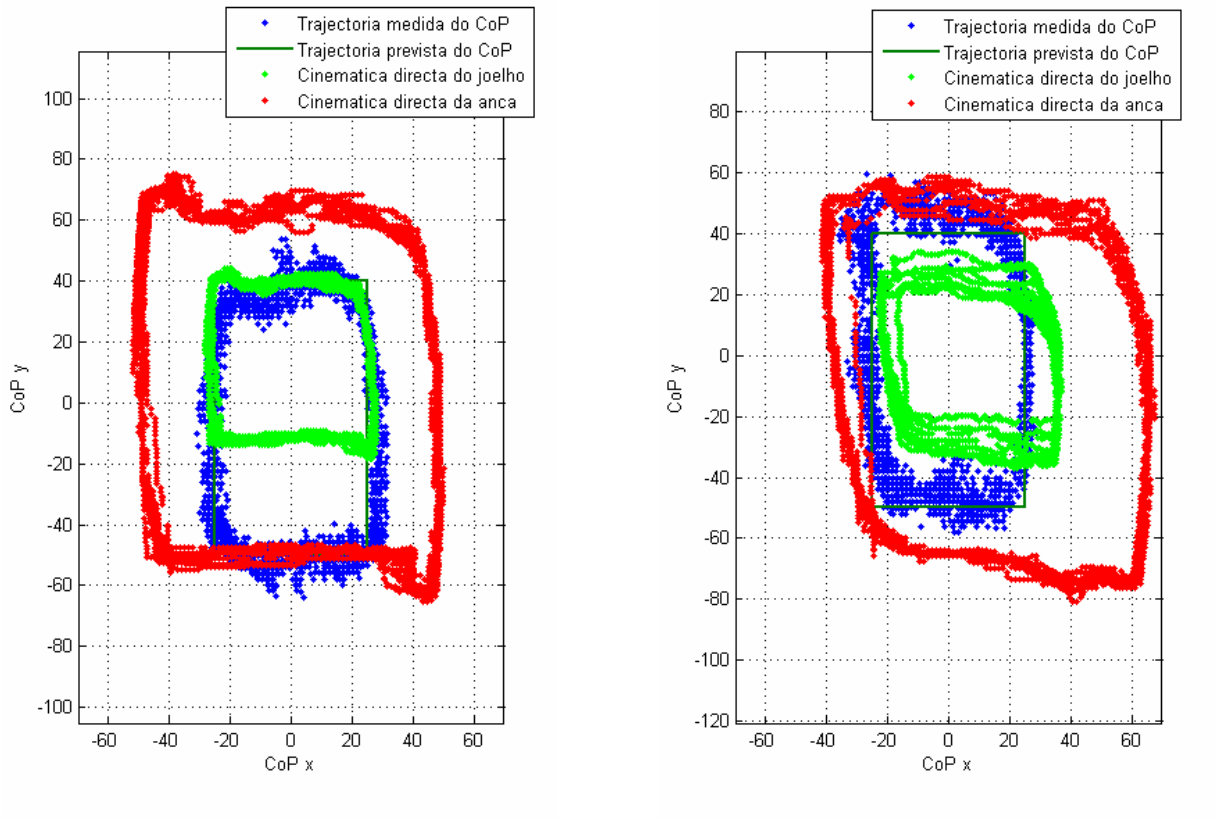


Fig. 77: Resposta a 10 voltas de uma trajectória rectangular para diferentes velocidades - $[K_I,K_P,K_{jacobiano}]=(5,40,30)$.

Apesar dos resultados animadores observados nas figuras anteriores, mostrando bons seguimentos de trajectória, desde que os parâmetros de compensação e a velocidade estejam devidamente ajustados, mesmo assim os resultados estão excessivamente dependentes das condições físicas do sistema, mais em concreto dos elementos adjacentes aos sensores de força. Diferenças na sensibilidade destes sensores, rigidez desigual nas aplacas de acrílico ligadas a estes, e a necessidade constante de ajuste dos potenciómetros que controlam o nível de tensão à saída dos sensores, resultam em significativas diferenças no cumprimento das trajectórias propostas, além que os valores solicitados ao controlador de equilíbrio respeitante a uma determinada localização física estão continuamente em mudança.

A Fig. 78 visualiza o problema descrito, com duas experiências executadas em instantes de tempo diferentes, mantendo iguais os ganhos de compensação, coordenadas referência do CoP e período de execução.

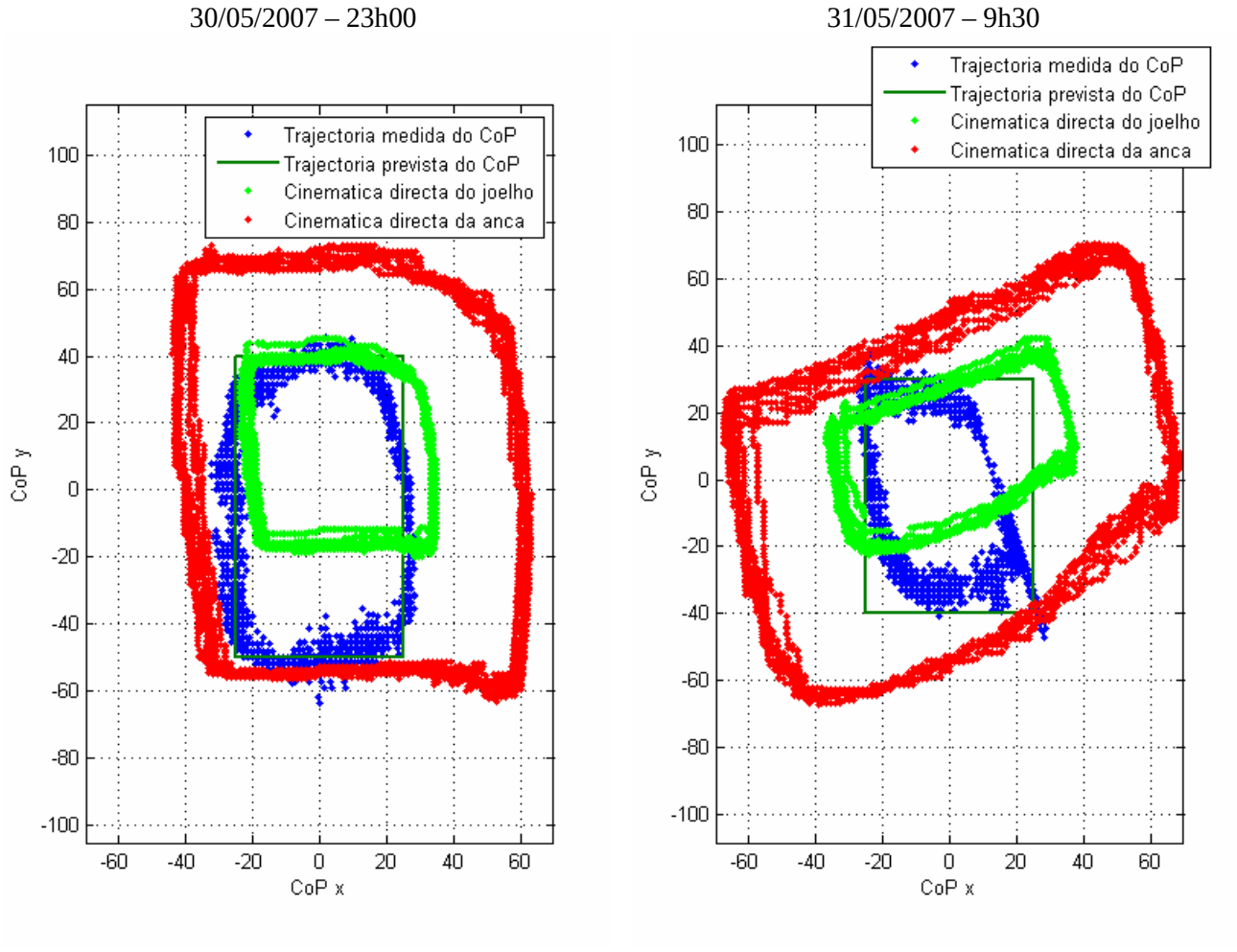


Fig. 78: Duas experiências realizadas em diferentes momentos, com iguais parâmetros de controlo.

Esta variabilidade de comportamento dificulta em grande parte a construção de um algoritmo que permita fazer a correspondência da localização espacial do centro de gravidade com as coordenadas do centro de pressão calculadas (secção III. 1.5).

Por estas razões, ainda não é possível indicar uma lei de correspondência entre estas duas grandezas, o que possibilitaria, um dia mais tarde, indicar as coordenadas espaciais desejadas da anca, e o *firmware* presente nos microcontroladores calcularia o centro de pressão referência a aplicar ao respectivo controlador.

5. CONCLUSÕES

Neste capítulo foram feitos vários estudos no que concerne à execução de trajectórias por parte da perna pela manipulação do CoP referência do controlador de equilíbrio. Três estudos principais foram feitos:

- Análise estática do controlador de equilíbrio;
- Análise dinâmica do controlador;
- Execução de multi-trajectórias e análise do desempenho.

A análise estática pode demonstrar o correcto funcionamento do controlador de equilíbrio com os cumprimentos das diversas posições solicitadas.

A análise dinâmica demonstrou a relação entre os parâmetros de compensação do controlador de equilíbrio e do período com a resposta, no que respeita ao erro de seguimento de trajectória e tempo de estabelecimento. Concluiu-se assim que aumentando o ganho de compensação, o erro tende a diminuir bem como o tempo de estabelecimento, mas desde que dentro de certos limites de modo a evitar instabilidade. De igual forma o período da trajectória também contribui para a diminuição do erro, recomendando-se, por isso, a aplicação de trajectórias com um período superior ou igual a quatro segundos para a minimização do sinal de erro.

Em seguida realizaram-se vários tipos de trajectória multi-volta nomeadamente rectilíneas, elípticas e rectangulares.
As trajectórias rectilíneas permitiram observar a interferência que o movimento em cada eixo introduz no eixo perpendicular, mais concretamente, oscilações de carácter não desprezável e relacionáveis com o movimento aplicado no primeiro eixo, concluindo, desta forma, que as trajectórias aplicadas em cada eixo não são independentes entre si. Verificou-se a susceptibilidade ao ruído que a componente *yy* apresenta, devido ao atribuído aumento de sensibilidade no cálculo do centro de pressão no mesmo eixo (*offset* subtraído à saída dos sensores de força).
Finalmente com as trajectórias elípticas e rectangulares, pudemos conhecer a limitação que os factores físicos inerentes à estrutura dos pés introduzem no cumprimento das trajectórias solicitadas. Verificou-se que quando a trajectória aplicada passa sobre um dos sensores, por vezes, o controlador é incapaz de garantir o cumprimento do CoP solicitado. O motivo deste comportamento ainda não está completamente esclarecido, mas como suspeita considera-se as seguintes causas:

1. As assimetrias de conexões entre o pé e a perna da estrutura humanóide podem interferir de alguma forma, ao fazer com que o centro de pressão não possa cobrir toda a área do pé. Relembra-se que os dois eixos que ligam estas duas partes não estão centrados sobre o pé, pelo que a trajectória realizada pode apresentar anomalias à medida que o centro de gravidade viaja ao longo do plano do pé.

2. Saturação na saída de um dos sensores: pouco provável dado que os potenciómetros de ajuste dos extremos de tensão dos sensores sempre foram verificados;

3. Diferente dinâmica por parte de cada extensómetro (variação da resistência em função da deformação): embora esta diferença possa ser insignificante, face ao valor nominal, não nos podemos esquecer que a seguir o sinal é amplificado 65 vezes pelo amplificador de instrumentação, pelo que qualquer erro será ampliado 65 vezes. Os potenciómetros de ajuste presentes na electrónica de acondicionamento apenas ajustam os extremos de tensão (*offsets*), não a dinâmica, pelo que, para as mesmas condições de calibração entre os quatro sensores, a tensão de saída da electrónica pode variar para uma determinada deformação. Desta forma, a relação tensão de saída *versus* força aplicada seria diferente entre os quatro sensores, e um factor de ajuste seria necessário para que o cálculo do centro de pressão não seja viciado.

4. Na condição da utilização de extensómetros de elevada qualidade, em que as diferenças na dinâmica sejam praticamente desprezáveis (mesmo após a amplificação), também há a possibilidade da dinâmica das placas de acrílico, sobre as quais os sensores estão colados, diferirem entre si. Esta diferença, se existente, é mais difícil de controlar que no caso dos extensómetros, uma vez que a este polímero, antes da entrada no mercado, não são aplicados critérios rigorosos de controlo de

qualidade no que toca à homogeneidade da estrutura. Por isso recomenda-se a aquisição deste material a fornecedores que garantam alguma qualidade, e a uma única peça criar as quatro placas usando uma máquina de precisão, de modo a evitar ao máximo as diferenças de comportamento entre as várias placas.

5. Na suposição de que a dinâmica das quatro placas de acrílico diferirem entre si, pode existir uma segunda causa para o efeito, além de supostas variação na estrutura molecular. Na ausência de qualquer força sobre a perna, às placas de acrílico é atribuída uma deformação inicial, através de parafusos de fixação que interligam as plataformas inferior e superior do pé. Embora tenha já sido demonstrado que qualquer que seja a deformação inicial a força aplicada é proporcional à deformação observada, ainda não se garantiu que a relação entre estes dois parâmetros seja independente desta deformação inicial. Empiricamente, parece fazer sentido que esta relação não seja independente, pois quanto maior for a deformação no momento da calibração, aparentemente menor será o seu acréscimo na aplicação de uma determinada força, quando comparada a outros casos. Como existem quatro parafusos de fixação (um para cada sensor), e como estes parafusos são ajustados manualmente, o controlo no ajuste da deformação inicial é praticamente inexistente, pelo que será de esperar respostas um tanto desiguais por parte de cada sensor, afectando, obviamamente, a resposta a resposta do controlador.

6. Também a forma como os parafusos de fixação estão montados, pode causar algumas anomalias. Pode ser observado que um parafuso liga as duas plataformas do pé através de uma porca sobre a plataforma superior, mas os restantes três, são directamente inseridos sobre peças metálicas presentes no topo do pé (a base do parafuso está livre, mas a outra extremidade está fixa à plataforma superior). Estes dois formatos introduzem diferenças significativas sobre a resposta de cada sensor. A fixação através de porca, permite a aproximação sem limitações das duas plataformas, mas impede o afastamento: desta forma pode-se definir a deformação inicial, sem comprometer o movimento das plataformas. Já a fixação por inserção directa, faz com que a liberdade de movimento dependa apenas da base do parafuso que se encontra livre da plataforma inferior, o que, por sua vez, limita o movimento das plataformas pela distância que esta extremidade tem ao solo (a base do parafuso está ligeiramente inserida sobre a plataforma, pelo que algum movimento ainda é possível). Contudo esta distância é escassa e impede a livre deformação das placas de acrílico. Como é fácil de concluir, quanto menor for a distância livre entre a base do parafuso e o solo, maior será a limitação.

Além destes problemas, também se observam respostas variáveis com o tempo causados com tendência a deteriorar-se ao longo do tempo, necessitando de reajustes no ganho do controlador, bem como na deformação inicial dos sensores. As causas podem ser identificadas como as seguintes:

- Postura da perna no momento da calibração inicial dos sensores de força: como é posicionada manualmente (por intervenção humana, nunca se garante duas posturas exactamente iguais, pelo que ocorrem sempre diferenças no centro de gravidade inicial). Mesmo efectuando o posicionamento através dos servomotores, os esforços envolvidos em cada experiência fazem com que a correia de transmissão sofra fadiga, ou o próprio eixo do servo adquira desfasamentos, pelo que este problema ocorrerá sempre;

- Assimetria ao nível da electrónica de instrumentação: note que para cada extensómetro, um circuito de acondicionamento é utilizado, pelo que se aplicarem componentes de baixa qualidade, podem-se registar variações no seu comportamento à medida que a temperatura vai aumentando. Além disso, se existirem ligações não suficientemente seguras no PCB, o movimento da perna, pode provocar oscilações nos sinais eléctricos;

- A baixa qualidade dos potenciómetros multi-volta usados na electrónica de instrumentação para ajuste dos *offsets* na tensão de saída, requer o seu reajuste constante o que constitui um sério problema tendo em vista a necessidade de autonomia do robot;

- Modificação acidental da deformação inicial das placas de acrílico, motivado por alguns acidentes de trabalho que forçam o deslocamento dos parafusos de fixação;

- É possível também que a relação deformação/força aplicada varie ao longo do tempo. Várias observações registaram diferenças de comportamento por parte dos sensores após várias horas de funcionamento, mas no dia seguinte, as condições iniciais voltavam a ser repostas. Note que isto acontecia quando movimentos intensivos sobre o pé eram executados, pelo que pode ocorrer alguma fadiga por parte da estrutura molecular do acrílico. Infelizmente esta suposição não pode ser verificada, uma vez que a realização de trajectórias durante um período de amostragem suficientemente extenso implica a presença constante de um observador dada a instabilidade inerente ao sistema.

Tendo em conta estes problemas apontados, sugere-se a reconstrução do protótipo seguindo as seguintes recomendações:

- Adopção de extensómetros de alta qualidade com o máximo de similaridade entre eles;

- Utilização de acrílico com características bastante homogéneas, e resistente à fadiga, de modo a minimizar a variação do seu comportamento ao longo do tempo;

- Utilização de montagens parafuso+porca, em todos os pontos de fixação das plataformas do pé, para homogeneização da resposta sensorial;

- Adopção de potenciómetros de alta qualidade, no que respeita a folgas mecânicas e dependência com a temperatura, para minimizar a necessidade de reajuste do *offset* das tensões de saída da electrónica de acondicionamento sensorial;

- Uso de PCB's de qualidade com ligações bastante seguras para garantir a estabilidade dos sinais eléctricos, bem como a montagem de componentes de boa qualidade que apresentem uma boa estabilidade no seu comportamento para uma boa gama de temperatura;

- Minimização das assimetrias na ligação pé/perna: procurar que os eixos de ligação entre estas duas partes se aproximem do centro do pé, de modo a conferir ao centro de pressão medido, uma relação bastante consistente com o centro de gravidade.

Uma ideia que surgiu, mas que ainda precisa de alguma dedicação, consiste na diminuição da electrónica de acondicionamento de sinal de quatro para apenas um circuito. Em vez de usar um extensómetro para cada ponte de Wheatstone (Fig. 29), utilizar-se-iam todos os quatro extensómetros numa única ponte, requerendo, deste modo, apenas um amplificador de instrumentação. Contudo há o problema de este circuito apenas permitir a medição do centro de pressão numa única dimensão (dimensão dependente da localização das saídas na ponte de Wheatstone). Um mecanismo de *switching* para alternar os pontos de medição na ponte seria necessário para possibilitar o cálculo do CoP nas duas dimensões. Segundo esta estratégia, reduziríamos a utilização de quatro sistemas independentes com diferentes condições iniciais (o hardware pode apresentar diferenças de comportamento) para apenas um, permitindo um cálculo mais preciso do centro de pressão.

IV. NOTAS FINAIS

1. CONCLUSÕES FINAIS

Este projecto permitiu aprofundar os conhecimentos práticos sobre barramentos de comunicação, mais concretamente a comunicação CAN muito usada em ambientes industriais, tendo-se desenvolvido vários protocolos de forma a conferir a máxima robustez e maximização da largura de banda disponível. Relativamente ao último protocolo implementado, os resultados observados são bastante animadores, sendo possível observar um funcionamento contínuo sem quaisquer falhas aparentes nas comunicações. Estas afirmações são baseadas em testes exaustivos que foram realizados durante períodos de funcionamento contínuos que nalguns casos atingiam dez horas, pelo que é possível confirmar a elevada robustez nas comunicações.
Apenas algumas correcções são apontadas ao nível do hardware para minimização dos efeitos colaterais na comunicação CAN quando o módulo RS-232 da unidade *Master* é removido. Sugere-se por isso a adição de uma resistência na entrada série RX, de forma a evitar a ocupação total da largura de banda de CPU pelo módulo de comunicação RS-232, que sendo de alta prioridade, impediria o normal funcionamento do módulo CAN.

Relativamente ao trabalho relativo aos algoritmos de controlo, aconteceu a oportunidade de tomar conhecimento do mercado existente no que concerne aos sensores de força que podem ser utilizados para conferir equilíbrio ao robot humanóide durante a locomoção, bem como também várias estratégias para a sua aplicação. No que respeita aos sensores de força usados actualmente (extensómetros), foi possível utilizá-los na detecção de impactos (pés contra o solo) e na realização de movimentos pela variação do ponto de referência do controlador de equilíbrio que utiliza estes sensores como realimentação. Contudo, os resultados ainda não são óptimos, observando-se frequentes descalibrações por parte destes sensores, bem como uma excessiva sensibilidade a interferências externas. Várias causas foram apontadas em detalhe bem como possíveis soluções para melhoria do funcionamento do controlador de equilíbrio, pelo que se espera, que uma vez seguidas, melhorem o desempenho e a estabilidade dos algoritmos de controlo implementados.

Apesar de algumas dificuldades no que respeita ao equilíbrio, todo o *firmware* desenvolvido para o controlo local já se encontra numa fase suficientemente madura, que em conjunto com outros trabalhos realizados paralelamente, nomeadamente a melhoria da parte mecânica e sensorial, permitirá, a curto prazo, a locomoção e a realização de outras tarefas pela unidade central de processamento de modo a podermos participar na competição mundial de robots humanóides RoboCup em 2008.

A partir destas observações, considero que os principais objectivos propostos para esta bolsa de investigação, foram realizados com relativo sucesso, tendo aberto as portas para novos caminhos que permitirão a maturidade deste projecto e a iminente concretização dos objectivos últimos, que são a criação de uma estrutura humanóide de carácter pedagógico capaz de gerar várias sequências de locomoção semelhantes ao ser humano, de uma forma autónoma e recorrendo unicamente a materiais e componentes de baixo custo.

2. DOCUMENTAÇÃO ADICIONAL

Toda a documentação produzida pela equipa Humanóide nos últimos tempos está apontada a seguir:

Artigos:

- Milton R. Silva, Pedro M. Ferreira, Filipe M. Silva, Vítor M. Santos, "Local-Level Control of a Humanoid Robot Prototype with Force-Driven Balance", submetido à International Conference on Humanoid Robots 2007.

- Milton Ruas, Filipe M. T. Silva, Vítor M. F. Santos, "*Parameter Measurement for Speed and Torque Control of RC Servomotors on a Small-Size Humanoid Robot*", nas actas do Encontro Científico Robótica2006, Guimarães, 2006.

- Milton Ruas, Filipe M. T. Silva, Vítor M. F. Santos, "*Techniques for Velocity and Torque Control of RC Servomotors for a Humanoid Robot*", aceite para publicação nos proceedings da 9th International Symposium on Climbing and Walking Robots and Associated Technologies, 11-14 September 2006

- Milton Ruas, Filipe M. T. Silva, Vítor M. F. Santos, "*A Low-Level Control Architecture for a Humanoid Robot*", submetido à International Conference on Humanoid Robots 2006.

- Vítor M. F. Santos, Filipe M. T. Silva, "*Engineering Solutions to Build an Inexpensive Humanoid Robot Based on a Distributed Control Architecture*", proceedings of the IEEE International Conference on Humanoid Robots 2005.

- Vítor M. F. Santos, Filipe M. T. Silva, "*Development of a Low-Cost Humanoid Robot: Components and Technological Solutions*", proceedings of the CLAWAR 2005.

Relatórios:

- Milton R. Silva, "*Desenvolvimento de Algoritmos de Controlo para Locomoção de um Robot Humanóide*", Relatório final de projecto 2005/06.

- Milton R. Silva, "*Desenvolvimento de Algoritmos de Controlo para Locomoção de um Robot Humanóide*", Manual do Programador – Projecto 2005/06.

- Milton R. Silva, "*Desenvolvimento de Algoritmos de Controlo para Locomoção de um Robot Humanóide*", Manual do Utilizador – Projecto 2005/06.

- Luís Gomes, Mauro Silva, "*Concepção e Desenvolvimento de Unidades de Percepção e Controlo para um Robot Humanóide*", Relatório final de projecto 2004/05 – Departamento de Mecânica

- Nuno Beça, Ângelo Cardoso, "*Desenvolvimento e Integração das Sub-estruturas Inferior e Superior para a Locomoção de uma Plataforma Humanóide*", Relatório final de projecto 2004/05 – Departamento de Mecânica.

3. BIBLIOGRAFIA

Recomendam-se as seguintes fontes bibliográficas para prossecução deste trabalho…

Artigos:

- Vítor M. F. Santos, "Robótica Industrial", Apontamentos teóricos, exercícios para aulas práticas e problemas de exame resolvidos – Departamento de Engenharia Mecânica, Universidade de Aveiro, 2003/04.

- Youbin Peng, Damir Vrancic, and Raymond Hanus, "*Anti-Windup, Bumpless, and conditioned Transfer Techniques for PID Controllers*", IEEE Control Systems, 1996.

- The PID Algorithm: http://members.aol.com/pidcontrol/pid_algorithm.html, 2006

- Tim Pike, "*Gait Generation for a Humanoid Robot*", The School of Information Technology and Electrical Engineering, The University of Queensland, 2003

- Michael Hardt and Oskar von Stryk, "*The Role of Motion Dynamics in the Design, Control and Stability of Bipedal and Quadrupedal Robots*", RoboCup 2002 International Symposium – Fukuoka, Japan, June 24-25 (2002).

- Dusko Kati´C and Miomir Vukobratovi´C, "*Survey of Intelligent Control Techniques for Humanoid Robots*", Journal of Intelligent and Robotic Systems 37: 117–141, 2003.

- T.Sugihara, Y.Nakamura, "*Whole-body Cooperative Balancing of Humanoid Robot using COG Jacobian*", Proceedings of the 2002 IEEE/RSJ, Intl. Conference on Intelligent Robots and Systems – EPFL, Lausanne, Switzerland, 2002

Livros:

- Karl Johan Åström, "*Control System Design*", 2002

- K. S. Fu, R. C. Gonzalez, C. S. G. Lee; "*Robotics: Control, Sensing, Vision and Intelligence*", MacGraw-Hill, 1987

4. AGRADECIMENTOS

Agradece-se toda a cooperação por parte do Departamento de Mecânica pela disponibilidade de recursos e toda a ajuda fornecida para a realização deste projecto.

Em igual medida, também agradeço aos meus orientadores, Filipe Silva e Vítor Santos, por todo o apoio e paciência que entregaram gratuitamente para que este trabalho desse os seus frutos.

Até breve!