Robot Voyager II – Reactividade e eficiência

Milton Ruas, José Luís Azevedo

Resumo **- Este artigo apresenta o projecto de um Robot desenvolvido para o concurso Micro Rato, edição 2005, realizado anualmente na Universidade de Aveiro. Este concurso desafia os entusiastas da robótica a desenvolver o seu próprio robot, com o objectivo de realizar a tarefa proposta dentro do tempo limite, sendo capaz de se desviar de eventuais obstáculos estáticos ou móveis. Este artigo propõe a descrição física do robot e da estratégia de controlo utilizada, bem como a arquitectura do *software*, algo importante para uma implementação transparente na programação de entidades complexas como robots.**

Abstract **- This paper presents a robot's project developed to the Micro Rato contest, edition 2005, that takes place annually at University of Aveiro. This contest challenges all robotics' enthusiasts to develop their own robot with a single common purpose: to fulfil the proposed task within the deadline, being capable to deviate from all static and movable obstacles that can arise. This paper proposes the robot's physical description and the control strategy implemented, as well as its software architecture, something very important for a transparent implementation when programming these complex entities.**

I. INTRODUÇÃO

O projecto de robots móveis em instituições académicas, constitui um verdadeiro desafio para os estudantes no desenvolvimento dos seus conhecimentos e experiência numa área de elevada multidisciplinaridade, permitindo alargar horizontes em diversas áreas como é o caso da mecânica, electrónica, instrumentação, programação e sistemas de controlo. Este tipo de actividades constitui um complemento muito útil para a compreensão das matérias abordadas em várias disciplinas dos cursos tecnológicos e também para a vida activa fornecendo uma visão bastante realista do mundo dos sistemas de controlo.

Com vista a incentivar este género de actividades, o Departamento de Electrónica e Telecomunicações da Universidade de Aveiro realiza todos os anos uma competição de pequenos robots móveis, o MicroRato [1] [2] [7], propondo uma tarefa comum a todos os participantes que a devem tentar cumprir pelo desenvolvimento dos seus próprios robots e do respectivo *software* de controlo: nas diversas mangas, os robots, circulando dentro de um labirinto, devem ser capazes de evitar obstáculos e de se desviarem de outros robots com a finalidade de atingir uma área circular caracterizada pela presença de um farol emissor de infravermelhos (omnidireccional) que os guia na sua direcção; opcionalmente poderá voltar para a posição original do labirinto, desta vez sem a ajuda de nenhum guia, recorrendo simplesmente à memória. O robot que executar esta(s) tarefa(s) dentro do tempo limite, e incorrendo na menor penalização, será o vencedor da competição. Para efeitos de desempate recorre-se ao factor tempo, dando-se a preferência ao robot mais rápido no cumprimento do primeiro objectivo (isto é, a chegada ao farol). De referir ainda que a competição está dividida em quatro mangas e que a configuração do labirinto apenas é desvendada no início de cada manga. Assim, os robots não incorporam, na sua programação, qualquer conhecimento prévio da configuração do labirinto onde vão competir.

Este artigo centra-se no robot Voyager II (Fig. 1), participante da edição 2005, detentor do prémio "Acelera" por ter sido o robot mais rápido nas duas primeiras mangas da competição.

Durante o desenvolvimento do *software* de controlo, apostou-se na transparência e numa cuidada organização do código de modo a evitar quaisquer ambiguidades na sua análise. Como contributo essencial para a sua organização, o sistema foi desenvolvido a partir de um *kernel* de tempo real multi-tarefa (PICMIK [4] [9] destinado ao processador PIC16F876 [9]).

Fig. 1 - Robot Voyager II.

O comportamento da entidade robótica é baseado num modelo reactivo [6], isto é, para cada conjunto de dados sensoriais está associada uma reacção predefinida (neste modelo o sistema reage em função da informação sensorial actual, não entrando em consideração com informação passada). Para a navegação no labirinto optou-se por uma estratégia de seguimento de paredes que,

potencialmente, maximiza a taxa de sucesso na chegada à área de farol.

Este artigo está estruturado do seguinte modo: a secção 2 apresenta o *hardware* constituinte do robot e faz um estudo da resposta dos sensores de obstáculos; a secção 3 descreve a arquitectura do *software* de controlo bem como a plataforma de tempo real que suporta esta arquitectura; a secção 4 demonstra a estratégia de controlo utilizada para conferir estabilidade no comportamento dinâmico; e, finalmente, a última secção apresenta as conclusões e os resultados provenientes das estratégias apresentadas.

II. DESCRIÇÃO DO *HARDWARE*

O robot Voyager II foi desenvolvido a partir do protótipo fornecido pelo departamento (deixando a responsabilidade da sua programação ao participante).

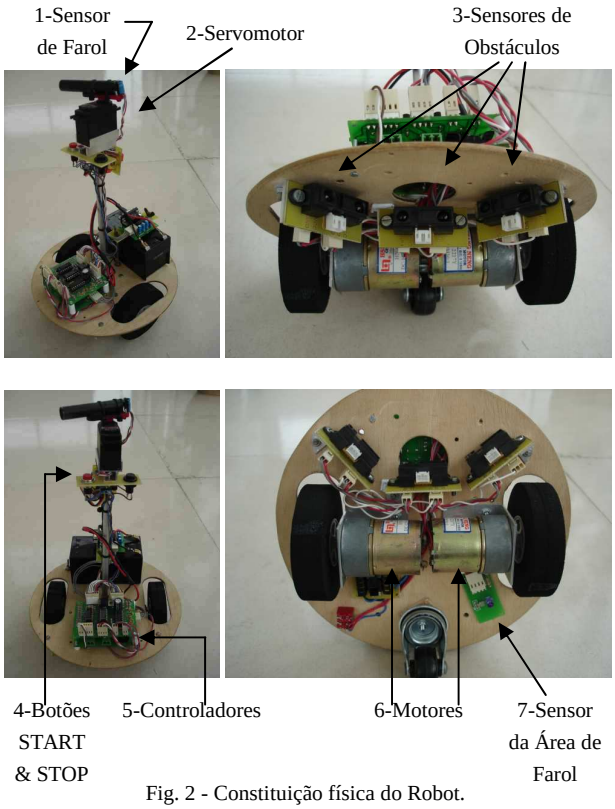

Fig. 2 - Constituição física do Robot.

O estrutura do robot é apresentado na Fig. 2, e é constituído pelo seguinte conjunto de sensores/actuadores:

- Dois motores de tracção (Ref. 6);
- Três sensores de infravermelhos (IV) para detecção de obstáculos (Ref. 3);
- Um receptor de IV (Ref. 1), acoplado a um servomotor (Ref. 2), para medição da direcção do farol;
- Um sensor de IV colocado na base inferior do robot para detecção da área de farol (Ref. 7);
- Botões START/STOP para interrupção/reactivação do funcionamento do robot (Ref. 4).

Para o controlo do robot é usado um microcontrolador PIC (Ref. 5 da Fig. 2) da Microchip [9] com 8 K*Words* (14 bits) de memória de programa e 368 *bytes* de RAM, constituído por diversos periféricos e por um CPU com uma arquitectura RISC capaz de executar 5×10^6 instruções/segundo (5 MIPS). Segue-se a descrição dos vários sensores/actuadores.

A. Sensor de Farol

O farol é constituído por um sistema emissor de IV que emite, sobre uma portadora de 40 KHz, um sinal digital de 600 Hz com modulação OOK. O robot, por sua vez, possui um receptor sensível a este sinal montado numa configuração que permite determinar a origem da fonte com uma precisão de, aproximadamente, ±5º (Ref. 1 da Fig. 2). Para tal, este é montado no interior de um tubo com paredes opacas que apenas permite detectar o sinal emissor quando este se encontra direccionado para a sua origem: conhecendo a orientação deste tubo é possível conhecer a direcção do farol. Um servomotor com o conjunto montado sobre o seu eixo (Ref. 2 da Fig. 2) permite fazer um varrimento de, aproximadamente, ±96º. Assim, retirando 33 amostras ao longo do varrimento, obtém-se a posição do farol com uma resolução de 6º e uma precisão de ±5º num campo de, aproximadamente, 192º.

Como o sinal captado pelo sensor é um sinal digital, nada se pode inferir sobre a sua distância, pelo que informação extra é necessária para que o robot tenha conhecimento de quando tenha atingido a área de farol.

B. Sensor detector da Área de Farol

A área de farol é caracterizada por um tapete circular não reflector (cor negra) com o farol localizado no centro. Com o robot dotado de um sensor de IV (emissor/receptor) (Ref. 7 da Fig. 2), orientado para o solo, é possível verificar se o robot está sobre uma superfície negra ou não: na ausência de recepção do sinal, o robot encontra-se sobre uma superfície não reflectora, e logo na área de farol. Deste modo, mesmo sem o conhecimento da distância do robot ao farol, é possível saber quando este atinge a área de farol.

C. Sensores de Obstáculos

Três sensores de obstáculos localizados frontalmente (Ref. 3 da Fig. 2), dois deles direccionados para partes laterais opostas e o último direccionado para a dianteira, destinam-se a detectar a existência de obstáculos (paredes ou outros robots móveis).

O modelo utilizado é o sensor SHARP GP2D12 com capacidade de detecção de objectos a distâncias até cerca de 80 cm. As características técnicas mais importantes deste dispositivo são:

- Distância de detecção: 100-800 mm;
- Tensão de saída na gama de detecção (relação inversa com a distância): 0.5-2.5V;
- Corrente consumida: ≈30 mA.

- Taxa de actualização da saída: ≈25 Hz (uma medida a cada 40 ms).

O funcionamento destes sensores baseia-se no princípio da triangulação representado esquematicamente na Fig. 3. O emissor (LED) emite um feixe de luz IV posteriormente focado por uma lente. A luz reflectida no obstáculo, segundo um determinado ângulo α, é focada por uma outra lente e o feixe resultante incide num detector sensível à posição (DSP). A saída do sensor é uma tensão proporcional ao valor detectado pelo DSP (i.e., proporcional ao ponto de incidência do feixe no DSP). Aplicando relações trigonométricas simples facilmente se conclui que $Vo \propto POS_{DSP} = (A * B)/d$.

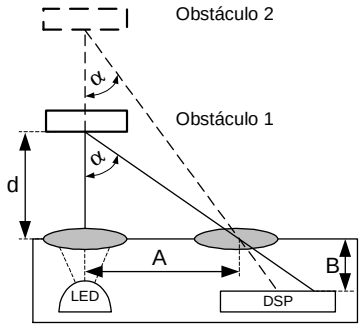

Fig. 3 - Funcionamento dos sensores IV Sharp.

A variação da tensão de saída com a distância, obtida experimentalmente, é apresentada na Fig. 4, observando-se, como seria de esperar, um comportamento não linear.

Um aspecto importante que se observa nesta figura, é o decaimento da tensão V para distâncias inferiores a 10 cm, dando a ideia de que o obstáculo se afasta quando acontece exactamente o contrário. Para evitar este tipo de problema os sensores foram colocados dentro do corpo do robot, a uma distância de, aproximadamente, 10 cm do seu limite físico (vide imagem inferior direita da Fig. 2).

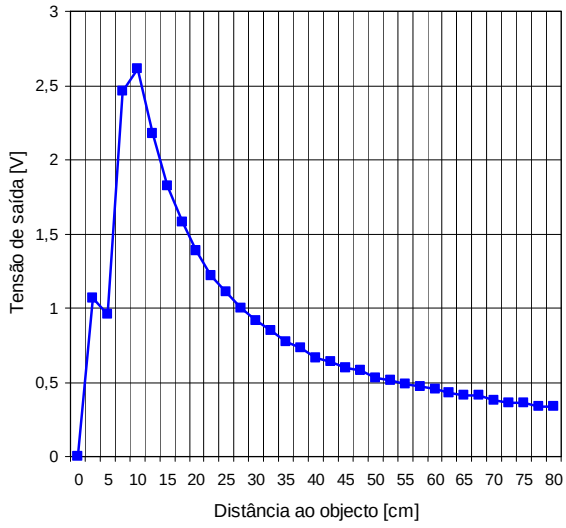

Fig. 4 - Relação tensão de saída *versus* distância.

Com base na Fig. 4, pode-se estimar a distância associada para cada valor de tensão seja através de uma tabela ou de métodos de linearização (por exemplo, aproximando a curva em três segmentos de recta: 10-20, 20-40 e 40-80 cm). Desta forma, as estratégias de controlo baseadas nas informações fornecidas por estes sensores são bastante melhoradas, dado que se contorna o problema da não linearidade da relação tensão/distância.

Um dado igualmente importante é a sensibilidade angular de detecção que estes sensores possuem e que uma vez conhecido permite estudar o seu melhor posicionamento no robot, de modo a maximizar a área de detecção de obstáculos. Para isso efectuaram-se medidas, tomando como obstáculo um objecto liso de 56 mm de espessura, deslocando-o angularmente entre –25 e +25º a diferentes distâncias (100 a 800 mm).

Estudou-se o comportamento do sensor em dois cenários distintos. Considere-se o plano Δ correspondente ao plano de deslocamento do obstáculo e o plano Σ o plano que intersecta o par emissor/receptor e o ponto de incidência do feixe no obstáculo. No primeiro cenário os planos Δ e Σ são paralelos entre si, e no segundo são ortogonais. Se considerarmos o plano Δ como sendo horizontal, no primeiro cenário o sensor está colocado na horizontal e no segundo está na vertical. A Fig. 5 e a Fig. 6 apresentam estes dois casos na forma de diagramas de sensibilidade, em que a representação angular indica o máximo ângulo de detecção e a representação radial indica a distância associada (desde 100 a 800 mm) – um obstáculo só é detectado pelo sensor se estiver posicionado dentro da área delimitada pela linha a cheio nos gráficos.

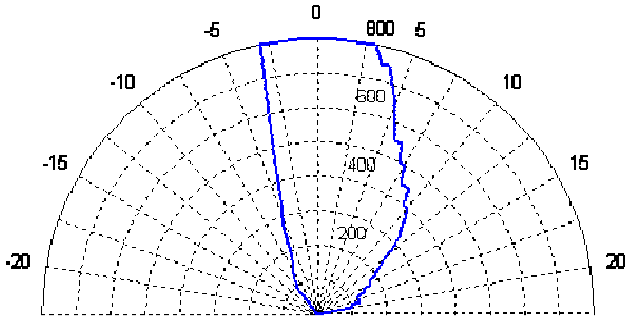

Fig. 5 - Diagrama de sensibilidade do sensor Sharp na horizontal.

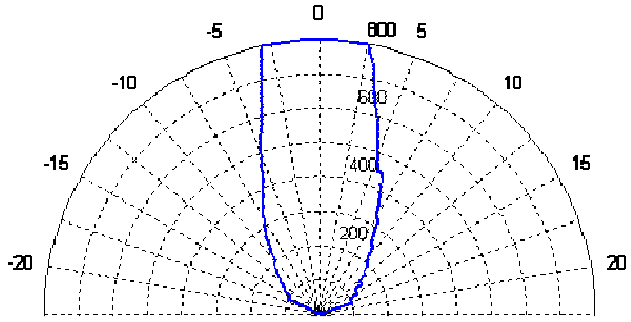

Fig. 6 - Diagrama de sensibilidade do sensor Sharp na vertical.

Comparando as duas disposições podem-se evidenciar algumas diferenças notórias. A disposição horizontal apresenta a vantagem de possuir uma maior abertura angular em termos gerais. Basta verificar que, para a distância intermédia de 40 cm, a abertura angular é de 14º contra os 12.5º da disposição vertical. No entanto, esta disposição apresenta dois problemas: 1) elevada assimetria angular com favorecimento para o lado do receptor (como pode ser verificado pela Fig. 5); 2) incorrecção acentuada da medida da distância fornecida sempre que o obstáculo se encontra deslocado para o lado da componente receptora do sensor (situação igualmente verificada de forma experimental). Com o sensor na disposição vertical, o diagrama de sensibilidade é perfeitamente simétrico e o problema da incorrecção da medida não ocorre. No entanto, e apesar dos inconvenientes referidos, optou-se pela configuração horizontal devido a limitações físicas.

Um facto que se pode constatar em ambas as situações, é o estreito ângulo de detecção – o máximo verificado é de apenas 20º, e apenas para a distância mínima. Por isso, de modo a minimizar os ângulos mortos, optou-se pela configuração apresentada na imagem esquerda da Fig. 7.

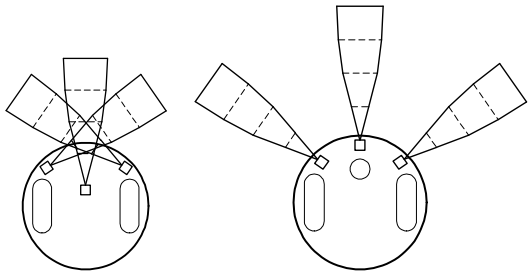

Fig. 7 – Área de detecção de obstáculos em duas configurações possíveis dos sensores.

D. Motores

Dois motores DC controlados por tensão foram usados para a locomoção. Estes motores, localizados em linha no centro radial do robot (Ref. 6 da Fig. 2), permitem uma grande liberdade de movimentos tais como viragens, retrocessos e rotações que aliados à sua forma circular lhe conferem uma elevada flexibilidade de movimentação. Com uma roda livre na parte traseira e o peso todo concentrado nesta parte (pela bateria) é garantido o equilíbrio do conjunto.

Características dos motores:

- Alimentação: 12V DC;
- Velocidade máxima: 200 rpm.
- Modelo matemático de 2ª ordem;

A velocidade do motor é controlada através da aplicação de um sinal PWM (modulação por largura de impulso, em que um *duty-cycle* de 100% corresponde à velocidade máxima do motor). O *drive* dos motores é feito com recurso ao circuito L293 (que implementa 2 pontes-H) que permite controlar dois motores independentes com possibilidade de inversão do sentido de rotação. Este circuito é atacado directamente pelo controlador do robot com o sinal PWM (velocidade) e um sinal digital (direcção) para cada motor.

Note-se que a relação "valor médio da tensão aplicada/velocidade do motor" não é linear dada a natureza de 2ª ordem do sistema: variações de tensão próximas do valor máximo resultam numa pequena variação da velocidade do motor.

III. ARQUITECTURA E INTERFACES DO *SOFTWARE*

A. Kernel de Tempo Real PICMiK

Para gerir a execução das actividades necessárias ao controlo do robot foi utilizado o *kernel* PICMiK [4] [5] [9] que permite encapsular as várias actividades em tarefas independentes. Este *kernel* possui a vantagem de um controlo temporal mais preciso garantindo deste modo que cada tarefa é executada no prazo estipulado.

Quando as tarefas são activadas, o *kernel* funciona como uma máquina de estados agendando e executando as várias tarefas segundo uma determinada política de escalonamento (Fig. 8). Para o *kernel* em causa, a política de escalonamento considerada é a de prioridades fixas, configuráveis à priori pelo programador.

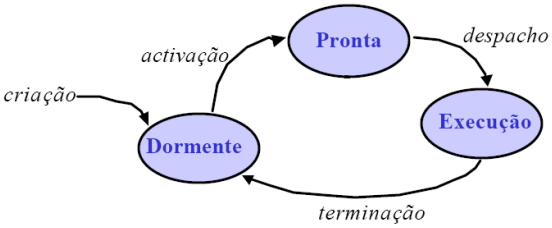

Fig. 8 - Máquina de estados do *Kernel*.

Quando é prevista a execução de uma tarefa, ela é agendada para execução passando do estado “dormente” para o estado “pronta”. Tendo em conta que várias tarefas podem estar neste estado, o *kernel* verifica qual a de maior prioridade e, de acordo com a política de escalonamento implementada, essa mesma será executada. Como o *kernel* em causa não é preemptivo, a tarefa não será interrompida até finalizar a sua execução; nessa altura voltará ao estado “dormente” e a seguinte de maior prioridade na fila de espera (no estado “pronto”) será executada.

Cada tarefa possui um conjunto de atributos internos que determinam a forma como será feito a agendamento de execução, como é o caso do período de activação. Se este parâmetro variar para as várias tarefas, a sua ordem de activação pode variar ao longo do tempo, pelo que não se pode garantir que determinada tarefa é activada antes de uma outra.

B. Arquitectura do Software

Oito tarefas descritoras dos comportamentos básicos do robot foram definidas com um determinado período de activação e prioridade. Elas são:

- **Actualização de Velocidade**: actualização da velocidade dos motores por afectação do PWM aplicado a cada um deles.
- **Verificação dos Botões**: verificação da pressão dos botões START ou STOP para arranque ou paragem do robot.
- **Verificação da Área de Farol**: através do sensor da área de farol é verificado se o robot atingiu o farol.
- **Localização de Farol**: estimação da direcção do farol pelo varrimento da área frontal do robot através do sensor de farol.
- **Leitura dos Sensores de Obstáculos**: medição da distância a que o robot se encontra dos obstáculos pela leitura dos sensores de IV frontais.
- **Desvio de Obstáculos**: Pelos dados fornecidos pela tarefa anterior, é determinado se existem obstáculos eminentes de colisão, efectuando, se necessário, os procedimentos necessários para o seu contorno.
- **Seguimento de Parede**: Seguimento lateral da parede caso se apresente no percurso no caminho para o farol. Um algoritmo de compensação em malha fechada do tipo proporcional derivativo é aplicado para garantir um controlo suave.
- **Navegação em Direcção ao Farol**: na ausência de obstáculos, o robot encaminha-se na direcção do farol com base nos dados fornecidos pela tarefa "*Localização de Farol*". Um algoritmo de compensação em malha fechada do tipo proporcional é aplicado para o controlo.

A prioridade das tarefas foi definida pela ordem apresentada desde a mais prioritária à menos. A Fig. 9, apresenta a arquitectura de execução das várias tarefas.

O conjunto de tarefas pode ser dividido em três grupos:

- Tarefas de máxima prioridade: tarefas que, sob determinadas condições, podem bloquear a total execução das tarefas de menor prioridade;
- Leitura de sensores: tarefas responsáveis pela leitura dos vários sensores (detecção de farol e de obstáculos);
- Navegação: tarefas responsáveis pela navegação do robot.

A comunicação entre tarefas é realizada através de estruturas de dados partilhadas (qualquer tarefa pode ter acesso a elas). Dado que o escalonador não é preemptivo não há a necessidade do uso de semáforos de exclusão mútua pois não ocorrem acessos simultâneos. Estas estruturas dizem respeito aos sensores e aos actuadores (Fig. 10).

Sempre que uma tarefa pretenda atribuir uma determinada velocidade aos motores deverá indicá-lo através da estrutura de dados de actuação, e nunca directamente pelo *device driver* que afecta imediatamente o PWM dos motores. Finalmente, será a tarefa "*actualização de velocidade*" que, por leitura desta estrutura, actualizará efectivamente a velocidade pelo uso do *device driver* apropriado. Este procedimento é importante para evitar situações de conflito, em que duas ou mais tarefas pretendam alterar a velocidade, garantindo que apenas uma tenha o seu pedido atendido. No que concerne à leitura dos sensores, o motivo é semelhante: para evitar leituras sucessivas que podem resultar em valores diferentes para a mesma iteração de execução, apenas uma tarefa efectua a leitura directa dos sensores – "*localização de farol*" para o sensor de farol e "*leitura dos sensores de obstáculos*" para os sensores de obstáculos – e as tarefas que fazem uso destes valores apenas têm de ler a estrutura de informação sensorial.

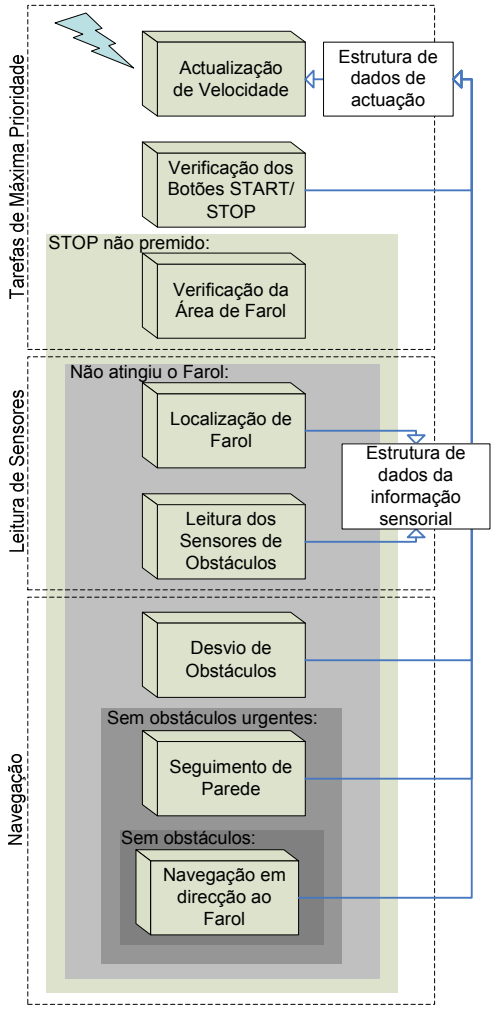

Fig. 9 - Arquitectura do *Software*.

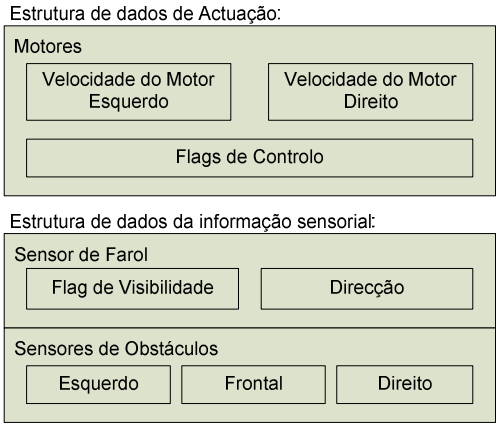

Fig. 10 - Interfaces de comunicação entre tarefas.

Note-se que todas as tarefas são activadas pelo escalonador do *kernel* e, por isso, todas são executadas segundo o seu período de activação definido. No entanto, tal é desfavorável pois cada uma lida com um determinado cenário, não devendo ser executada se esse cenário não se verificar. Um exemplo deste tipo de circunstâncias ocorre quando o botão de paragem forçada (STOP) é premido ou quando a área de farol é detectada (tarefas *"verificação dos botões START/STOP"* e *"verificação da área de farol"* respectivamente), o que implica que as operações de leitura de sensores e de navegação devem ser interrompidas, e por isso, todas as tarefas de prioridade inferior a estas não devem ser executadas. Um motivo alternativo prende-se com a situação de uma outra tarefa de maior prioridade e de operação equivalente ter sido executada e, por isso, a actual não deve ter lugar, pois uma acção de prioridade superior foi tomada em conta. Esta circunstância ocorre para a operação de navegação, em que apenas uma das actividades – *"desvio de obstáculo"*, *"seguimento de parede"* e *"navegação em direcção ao farol"* – deve ser executada pois cada uma lida com um cenário específico. De modo a garantir esta última condição é fundamental que as tarefas relativas à navegação cumpram os seguintes requisitos:

- Sejam activadas na mesma iteração de execução – garantido pelo *kernel* quando os períodos de activação destas três tarefas são iguais;
- Sejam executadas segundo a sua ordem de importância – garantido quando os índices de prioridade são correctamente atribuídos;
- E que cada tarefa apenas é executada quando nenhuma de prioridade superior (deste grupo) tenha sido considerada.

De forma a cumprir a política de decisão referida na execução das tarefas, uma estratégia de arbitragem torna-se indispensável para o correcto desempenho do robot.

C. Gestão de Arbitragem

Como todas as tarefas são iniciadas pelo escalonador, optou-se por uma estratégia de responsabilização de cada uma acerca da sua própria execução, ou seja, antes de cada tarefa executar o seu código intrínseco, deve previamente fazer um conjunto de testes para averiguar se estão reunidas todas as condições para a sua total execução. Caso assim não seja, a execução deve ser imediatamente terminada, não permitindo, deste modo, que a actividade associada à tarefa em questão seja realizada. As condições referidas estão indicadas nos rectângulos a cinzento da Fig. 9, pelo que cada tarefa deve verificá-las logo no seu cabeçalho antes de executar a sua actividade.

Note-se que como as várias tarefas estão organizadas segundo prioridades decrescentes, as condições para a total execução crescem (mantendo as anteriores) à medida que a prioridade decresce, pelo que a última tarefa é a mais dependente das condições do meio externo. Uma consequência desta característica é o facto de que, quando a condição para a execução de uma determinada tarefa falha, a execução de todas as restantes tarefas de prioridade inferior é comprometida, e por isso, a tarefa seguinte a executar será a primeira do conjunto – *"actualização de velocidade"*.

As condições a efectuar são baseadas num conjunto de variáveis indicadoras do estado actual do sistema:

- Robot em paragem forçada (botão STOP premido) – modificada pela tarefa *"verificação dos botões START/STOP"*.
- Chegada ao destino (detecção da área de farol) – modificada pela tarefa *"verificação da área de farol"*;
- Pedido prévio de alteração de velocidade – modificado pelas tarefas *"desvio de obstáculo"*, *"seguir parede"* e *"navegação em direcção ao farol"*.

Estas condições estão apresentados na Fig. 9 nas condições para execução de cada conjunto de tarefas, incluindo os casos de não haver obstáculo e parede representados pela *flag* de alteração de velocidade prévia. Nestes últimos há um ponto em comum – ambos interferem na navegação do robot pela capacidade de alteração de velocidade – após a activação de qualquer uma destas tarefas, é necessário verificar se a velocidade já foi previamente alterada. Em caso afirmativo, significa que uma acção mais prioritária foi realizada e, por isso, garante-se que apenas uma é executada – a de maior prioridade.

IV. CONTROLO

Nesta secção abordaremos as estratégias de controlo usadas para realizar as actividades respeitantes às tarefas relativas à navegação:

- Desvio de obstáculos;
- Seguimento de parede;
- Navegação em direcção ao farol.

Cada uma usa uma estratégia diferente de acordo com o objectivo em causa: o robot deve efectuar o movimento pretendido o mais rapidamente possível com o mínimo de *overshoot* e erro em regime estacionário.

A. Desvio de Obstáculos

Em circunstâncias normais, sempre que uma parede do labirinto ou outro robot é detectado a uma distância relativamente segura, a tarefa *"seguimento de parede"* efectuará o desvio do obstáculo recorrendo a técnicas de compensação apropriadas. No entanto, o limitado campo que os sensores cobrem e os ângulos mortos existentes entre eles podem potenciar a aproximação excessiva de obstáculos sem a sua prévia detecção, tornando as técnicas de compensação de desvio ineficientes. Para este tipo de circunstâncias, foi criada a tarefa *"desvio de obstáculos"*, que recorre a manobras de emergência sempre que um obstáculo é detectado demasiadamente próximo do robot.

De modo a realizar as manobras adequadas para todas as situações possíveis, três cenários frequentes foram descriminados, nomeadamente:

1. Obstáculo lateral iminente de colisão (Fig. 11): efectuar marcha atrás.
2. Obstáculo apenas frontal iminente de colisão (Fig. 12): efectuar marcha atrás virando para o lado mais apropriado;
3. Obstáculo frontal ou lateral a uma distância segura (Fig. 13): virar para o lado mais apropriado.

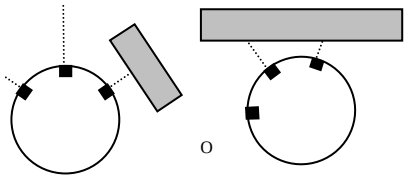

Fig. 11 - Obstáculo lateral iminente de colisão.

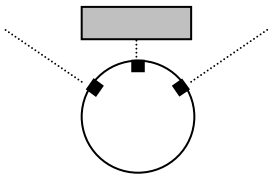

Fig. 12 - Obstáculo frontal iminente de colisão.

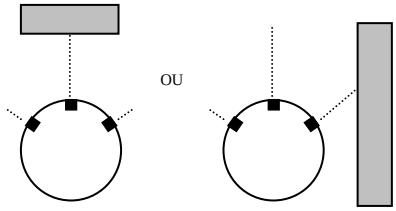

Fig. 13 - Obstáculo frontal/lateral a uma distância segura.

Nas situações 2 e 3 (Fig. 12 e Fig. 13) fala-se numa viragem para a direcção mais apropriada. Esta decisão é provavelmente umas das mais importantes, pois determina o rápido cumprimento do objectivo global. Em cada uma destas situações, dois cenários podem-se discernir com base na informação do sensores laterais livres:

- Um obstáculo é detectado a uma distância relativamente próxima: neste caso a decisão mais segura é seguir pelo lado mais livre de modo a evitá-lo.
- não é detectado nenhum obstáculo, pelo menos para a distância de interesse: neste caso é apropriado decidir pela direcção onde se encontra o farol.

Nesta última situação, no caso do robot se localizar numa esquina de uma abertura, dirigir-se-á logo na direcção do farol.

A mudança de direcção do robot é efectuada pela paragem de uma das rodas, com a outra a circular a uma velocidade de segurança (inferior à velocidade de "cruzeiro"), e nunca pela rotação, pois esta efectua movimentos demasiado bruscos provocando comportamentos imprevisíveis. No entanto, o método adoptado apresenta uma desvantagem: pode ocorrer a situação de colidir com um obstáculo numa zona morta para os sensores, e, como os motores não possuem odometria, nada de anormal seria reportado provocando o bloqueio na movimentação. Para evitar este tipo de anomalias, optou-se pela manobra de retrocesso sempre que um obstáculo muito próximo é avistado, pois é nestas situações que o risco de bloqueio aumenta. Esta solução pode igualmente provocar o bloqueio, dado que não existem sensores traseiros, por isso para minimizar este problema incorporou-se um período de *timeout* para efectuar esta manobra retomando a marcha para a frente caso este período seja ultrapassado.

B. Seguimento de Parede

Na ausência de obstáculos iminentes de colisão, é executada a tarefa seguinte – *"Seguimento de Parede"* – em que, no caso de ocorrer um obstáculo no percurso, é feito um acompanhamento lateral do mesmo. Este procedimento corresponde a uma das estratégias que têm apresentado bastante sucesso no cumprimento do objectivo global.

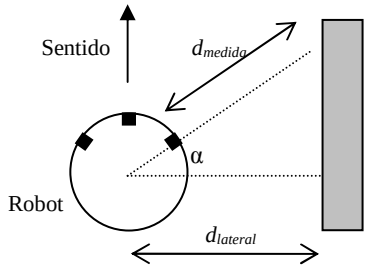

Fig. 14 - Seguimento de uma parede.

Com base na distância medida pelo sensor mais próximo da parede (Fig. 14), pode-se determinar a distância

efectiva a ela pelo ângulo que o sensor faz com a normal lateral (1).

$$d_{lateral} = d_{medida} \cdot \cos(\alpha) \qquad (1)$$

Pelo conhecimento da distância lateral efectiva, basta fazer a compensação apropriada da velocidade dos motores de modo a ajustar o robot na distância pretendida (*velocidade*$_{motores}$=*f*(*erro*)) (Fig. 15).

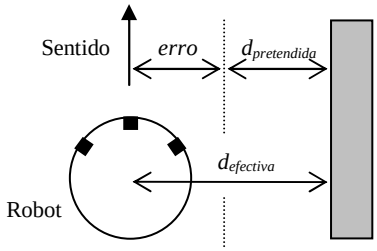

Fig. 15 - Análise do erro no seguimento de uma parede.

A compensação da distância lateral é realizada pela redução de velocidade de apenas um dos motores, enquanto que o outro permanece à velocidade de cruzeiro, logo, para aproximar o robot da parede basta reduzir a velocidade do motor mais próximo dela, e o oposto para o afastamento. Para facultar uma aproximação/afastamento rápida e com o mínimo de oscilação, aplicou-se um compensador em malha fechada do tipo proporcional derivativo (Fig. 16) ao motor a reduzir. Embora este tipo de compensadores seja de risco em aplicações que envolvem ruído (as medidas de distância fornecidas pelos sensores apresentam alguma ruído), tal não é problemático neste caso, pois a inércia do robot impede a propagação do ruído.

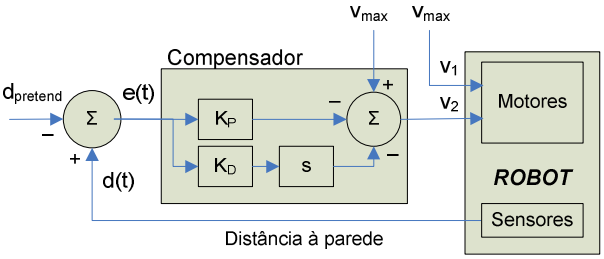

Fig. 16 - Compensador PD aplicado aos motores.

A velocidade a aplicar a cada motor é:

$$\begin{aligned} &v_1(t+T) = v_{max} \\ &v_2(t+T) = \begin{cases} v_{max} - \Delta v_2(t), & se\ \Delta v_2 \le v_{max} \\ 0, & se\ \Delta v_2 > v_{max} \end{cases} \\ &\Delta v_2(t) = K_P \cdot e(t) + K_D \cdot \frac{d}{dt} e(t) \end{aligned} \qquad (2)$$

$$e(t) = d(t) - d_{pretendida} \qquad (3)$$

em que v_1 é a velocidade do motor a manter, e v_2 é a velocidade do motor a reduzir. Como estas expressões são aplicadas em *software*, é necessário discretizá-las para o espaço digital:

$$\begin{aligned} &v_1(n+1) = v_{max} \\ &v_2(n+1) = \begin{cases} v_{max} - \Delta v_2(n), & se\ \Delta v_2 \le v_{max} \\ 0, & se\ \Delta v_2 > v_{max} \end{cases} \\ &\Delta v_2(n) = K_P \cdot e(n) + K_D \cdot \left[e(n) - e(n-1) \right] \end{aligned} \qquad (4)$$

$$e(n) = d(n) - d_{pretendida} \qquad (5)$$

Um detalhe importante a referir é que v_2 deve ser sempre inferior a v_{max}, o que implica que a variação de velocidade $\Delta v_2(n)$ deve ser sempre positiva.

Supondo $e(n) \approx e(n-1)$ pode-se assumir que o sinal de $\Delta v_2(n)$ é o mesmo que o de $e(n)$ (6) (mesmo para a condição inicial em que $e(n-1)=0$):

$$\Delta v_2(n) = (K_P + K_D) \cdot e(n) - K_D \cdot e(n-1) \qquad (6)$$

Com K_P e K_D>0, e $e(n) \approx e(n-1)$:

$$\Delta v_2(n) \approx K_P \cdot e(n) \Rightarrow \text{sign}(\Delta v_2) = \text{sign}(e) \qquad (7)$$

Por isso, Δv_2 positivo significa que o sinal de erro também é positivo, ou seja, o robot está demasiado distanciado sendo a diminuição de velocidade aplicado ao motor mais próximo da parede. Caso $\Delta v_2(n)$ seja negativo (robot próximo da parede) é necessário determinar o módulo de Δv_2 e aplicar a redução de velocidade correspondente ao motor mais afastado da parede.

Resumindo:

1. O motor sobre o qual se aplica a redução de velocidade é determinado a partir do sinal de Δv_2;
2. A velocidade do motor a reduzir é calculada a partir do módulo de Δv_2 (8).

$$v_2(n+1) = \begin{cases} v_{max} - |\Delta v_2(n)|, & se\ |\Delta v_2| \le v_{max} \\ 0, & se\ |\Delta v_2| > v_{max} \end{cases} \qquad (8)$$

Os parâmetros do compensador K_P e K_D foram obtidos de forma experimental. Inicialmente, procurou-se pelo melhor valor de K_P no que toca a tempo de estabelecimento para a velocidade escolhida (80% da velocidade máxima). Como já seria de esperar (dado estarmos perante um sistema de segunda ordem), verificou-se uma elevada oscilação. A Fig. 17 ilustra a resposta ao degrau típico de um sistema deste tipo.

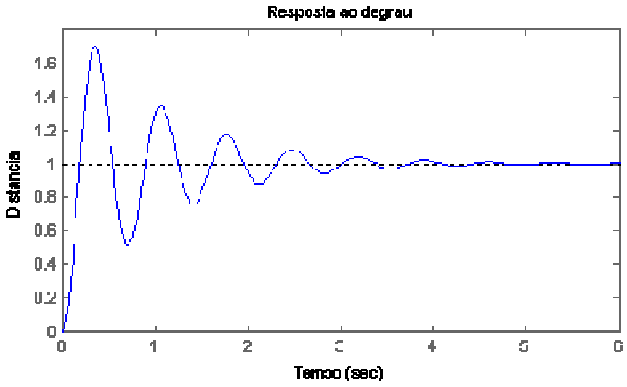

Fig. 17 - Resposta ao degrau típica de um sistema de 2ª ordem com compensação P.

É então que entra a componente derivativa para tentar resolver este inconveniente. A Fig. 18 ilustra o resultado do sistema compensado com um compensador PD.

Como se pode constatar pela Fig. 18, a utilização da componente derivativa permite obter respostas com um *overshoot* e um tempo de estabelecimento aceitáveis.

Devido ao facto de não existir erro em regime estacionário, dada a própria natureza do sistema realimentado (Fig. 16), não foi necessário incorporar o integrador no compensador, sendo suficiente as componentes proporcional e derivativa.

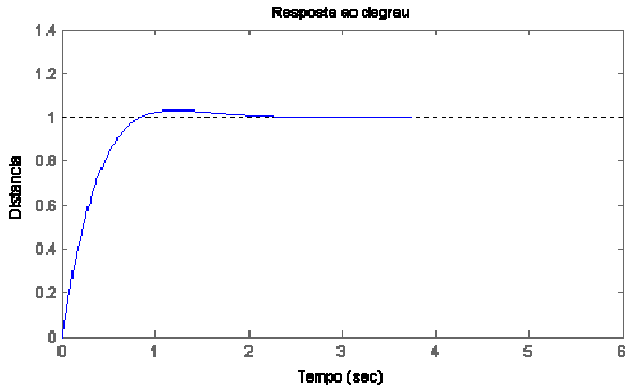

Fig. 18 - Resposta ao degrau da distância robot/parede com compensação PD.

Para um refinamento do controlo, nas situações em que o robot se distancia demasiado da parede (erro positivo), o sinal de erro é atenuado de um determinado factor. Como consequência, o robot tende a dispender maiores períodos de tempo afastado da parede do que próximo. Deste modo, a probabilidade de colisão com a própria parede, durante a fase de estabilização, diminui consideravelmente fazendo com que o robot responda mais rapidamente quando está muito próximo da parede, do que no caso oposto (Fig. 19).

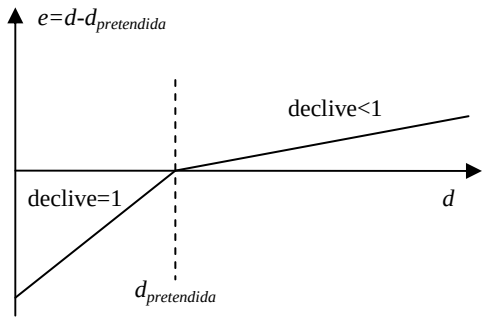

Fig. 19 - Sinal de erro a usar de acordo com a proximidade à parede.

Uma circunstância que pode ocorrer é a detecção de parede em ambos os lados, como é o caso dos corredores. Nestas situações considera-se como sinal de erro, a diferença das distâncias medidas pelos dois sensores laterais (9), sendo o restante processamento semelhante ao caso de uma só parede.

$$e(n) = d_{esquerda}(n) - d_{direita}(n) \qquad (9)$$

C. Navegação em Direcção ao Farol

Esta tarefa conduz o robot até à área de farol e apenas é executada quando nenhum objecto, considerado obstrutivo para a realização deste objectivo, é detectado pelos sensores de obstáculos.

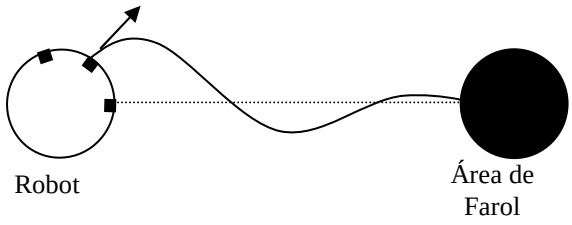

Fig. 20 - Comportamento típico da navegação do robot em direcção ao farol com compensação.

A Fig. 20, apresenta o comportamento típico da navegação do robot em direcção ao farol quando também é aplicado um compensador para o efeito. Como já foi referido anteriormente, a tarefa *"Localização do Farol"* determina a direcção dele, pelo que, quando comparado com a direcção frontal constitui o sinal de erro (10).

$$e(n) = angulo_{farol} - 0^{o} = angulo_{farol} \qquad (10)$$

Para este objectivo concreto, mostrou-se experimentalmente ser suficiente um compensador proporcional, obtendo-se *overshoots* mínimos para tempos de atraso muito reduzidos. O valor do parâmetro K_P foi obtido experimentalmente e baseou-se na procura do valor a partir do qual o sistema entra em oscilação.

Matematicamente:

$$\begin{aligned} &v_1(n+1) = v_{max} \\ &v_2(n+1) = \begin{cases} v_{max} - \Delta v_2(n), & se\ \Delta v_2 \le v_{max} \\ 0, & se\ \Delta v_2 > v_{max} \end{cases} \\ &\Delta v_2(n) = K_P \cdot |e(n)| \end{aligned} \quad (11)$$

$$e(n) = angulo_{farol} - 0º = angulo_{farol} \quad (12)$$

Nesta representação, já se garante que a velocidade do motor a reduzir é sempre inferior à máxima, pelo que apenas é necessário definir qual o motor a reduzir através do sinal de $e(n)$, de modo a orientar o robot na direcção do farol. Deste modo o comportamento do robot é baseado na Fig. 21 – note-se que a velocidade do motor não compensado é igual a v_{max}.

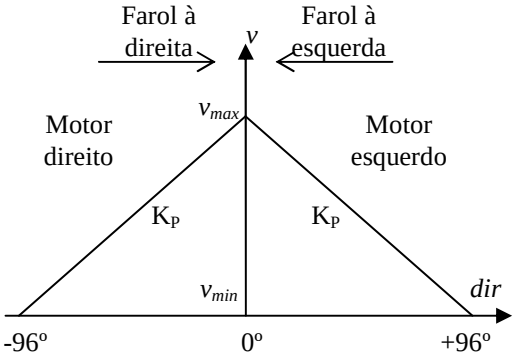

Fig. 21 – Relação "velocidade dos motores/direcção do farol".

Embora em espaço aberto este algoritmo funcione correctamente, na presença de paredes surgem algumas dificuldades, como por exemplo na entrada em aberturas: se o farol não estiver muito desviado relativamente ao robot ele pode não virar o suficiente para fazer a entrada com sucesso correndo o risco de falhar a entrada (Fig. 22).

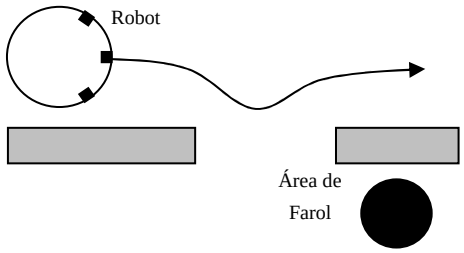

Fig. 22 - Entrada mal sucedida numa abertura.

De modo a evitar estes cenários, modificou-se o algoritmo de modo a contemplar dois parâmetros K_P diferentes (Fig. 23).

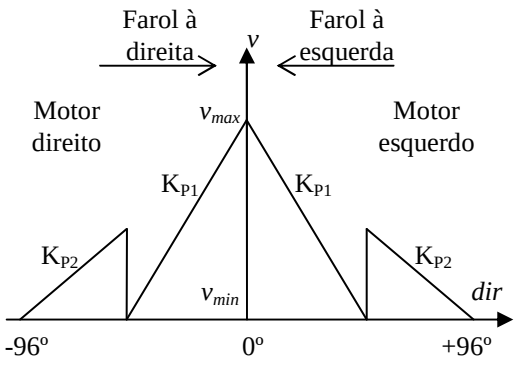

Fig. 23 - Compensação proporcional modificada.

Segundo esta estratégia, para erros reduzidos um K_P elevado é aplicado, forçando-o a entrar em aberturas mesmo com o farol pouco desviado. Já para erros elevados um K_P mais baixo é usado para evitar a oscilação e ao mesmo tempo o toque em esquinas de paredes quando a contorná-las.

No entanto, vale a pena referir que o sucesso destes algoritmos está dependente da capacidade de determinar rápida e fiavelmente a direcção do farol, pelo que a tarefa *"Localização de Farol"* necessita de um algoritmo suficientemente robusto para evitar erros no cálculo da direcção. A Fig. 24, apresenta um exemplo dos valores fornecidos pelo sensor de farol num varrimento completo (±96º).

De modo a minimizar o efeito do ruído na detecção da posição do farol, efectuou-se uma média pesada entre as amplitudes medidas para todas as direcções (13).

$$dir_{farol} = \frac{\sum_{dir} dir \cdot Amp_{sensor}(dir)}{\sum_{dir} Amp_{sensor}(dir)} \quad (13)$$

Pela análise da expressão 13, efectuando a média pesada das amplitudes medidas pelo sensor de farol ($Amp_{sensor}(dir)$) para cada direcção (dir) que supere um determinado limiar, pode-se determinar a direcção do farol (dir_{farol}) mais provável. Para tal é necessário armazenar as medições efectuadas para todas as direcções num *array* de 33 elementos (±96º de deslocamento angular com uma resolução de 6º), estimando a direcção do farol após cada medida. De forma a minimizar o tempo de actualização desta grandeza, evita-se o varrimento completo do servomotor sempre que tal se torne desnecessário. Assim, se o farol deixa de ser detectado (ainda era visível na medida anterior), inverte-se de imediato o sentido de rotação do servomotor.

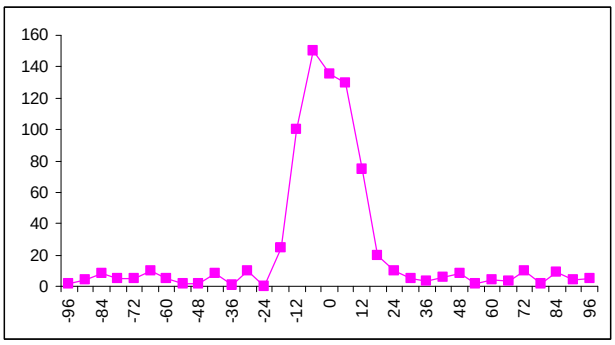

Fig. 24 – Exemplo dos valores fornecidos pelo sensor de farol num varrimento completo.

Com este algoritmo, observa-se que, quando o farol é visível a partir do robot, o sensor está sempre apontado na sua direcção. Isto tem como consequência que o tempo de actualização da direcção do farol seja minimizado, por oposição a um algoritmo de varrimento completo. Adicionalmente, o *debugging* é simplificado, dado que é possível verificar visualmente se o sistema está a detectar o farol ou se está a efectuar uma falsa detecção.

V. CONCLUSÃO

Este artigo descreve, no essencial, a implementação de um robot participante no concurso MicroRato. Apesar da utilização de componentes de baixo custo (como é o caso dos sensores de infravermelhos SHARP e do microcontrolador baseado num PIC16F876), provou-se que, com algoritmos eficientes de controlo, é possível ultrapassar as limitações do *hardware* obtendo um bom comportamento dinâmico. Para tal foram implementados compensadores nos algoritmos de navegação em direcção ao farol e de seguimento de paredes. No caso do seguimento de farol foi apresentada a técnica de compensação proporcional modificada, através do uso de dois parâmetros K_P diferentes, adoptada com o objectivo de tornar mais eficaz a navegação no labirinto em direcção ao farol.

De forma a garantir a eficiência dos algoritmos de navegação, é fundamental assegurar a qualidade dos valores devolvidos pelos sensores. Para isso torna-se necessário um pré-processamento desses valores, como é o caso de:

- conversão da tensão de saída dos sensores de obstáculos para distância;
- determinação da distância do robot à parede de seguimento, para o caso dos sensores laterais;
- cálculo da média pesada para a determinação da direcção do farol.

Para a organização do *software* utilizou-se um *kernel* de tempo real que permite o isolamento das várias actividades em tarefas, o que facilita o desenvolvimento (concepção e *debugging*).

Os resultados destas estratégias puderam ser comprovados pelo cumprimento do primeiro objectivo do Concurso Micro Rato em tempos reduzidos, devendo-se salientar o suave comportamento dinâmico do robot na realização das diferentes tarefas de navegação no labirinto.

No entanto, ainda existem alguns aspectos a melhorar no comportamento global do robot, como é o caso do aumento da área de detecção de obstáculos e da melhoria do controlo com a possibilidade da inclusão de compensação *Fuzzy-logic*.

REFERÊNCIAS

[1] L. Almeida, P. Fonseca, J. Azevedo, B. Cunha, "The Micro Rato Contest: Mobile Robotics for All", The Portuguese Control Conference, Outubro 2000.

[2] L. Almeida, P. Fonseca, J. Azevedo, "The Micro Rato Contest: a popular approach to improve self-study in electronics and computer science", Proceedings of SMC'2000, IEEE Int. Conference on Systems, Man and Cybernetics, Nashville USA, October 2000.

[3] Concurso Micro Rato 2005: Regras e Especificações Técnicas da Modalidade Micro Rato. Disponível em [7].

[4] A. Quinta, R. Andrade, "PICMiK – PIC Micro Kernel", Revista do DETUA, Abril 2004.

[5] L. Almeida: "PICMiK – PIC Micro Kernel: Breve Manual", Abril 2004. Disponível em [9].

[6] A. Oliveira, A. Melo, "Sniffer: Um Robot baseado em Arbitragem de Comportamentos Autónomos e Reactivos", Revista do DETUA: Vol.3 Nº2, Outubro 2000.

[7] Concurso Micro Rato: http://microrato.ua.pt (2005).

[8] PICMiK Webpage:http://sweet.ua.pt/~lda/retmik/picmik.html (2005).

[9] *Microcontrollers PIC16F87X Datasheet*. The Microchip Worldwide Site. Disponível.em http://microchip.com (2005).