Sniffer: Um Robot Baseado em Arbitragem de Comportamentos Autónomos e Reactivos

Arnaldo Oliveira, Andreia Melo

***Resumo* - Este artigo descreve os aspectos mais importantes das arquitecturas de hardware e de software do robot *Sniffer*, o qual foi construído especificamente para participar no concurso de robótica Micro-Rato 2000 da Universidade de Aveiro, onde lhe foi atribuído o "Prémio Inovação". Os componentes principais do seu hardware são um microcontrolador da família 8051 da Intel e uma FPGA XC4005XL da Xilinx. O primeiro é utilizado para executar os algoritmos de controlo, enquanto a segunda realiza algumas tarefas de baixo nível. O software de controlo é baseado em arbitragem de comportamentos autónomos e reactivos. Esta abordagem para além de tornar a sua programação bastante flexível e completamente extensível, permite também que o robot, sem qualquer conhecimento prévio dos labirintos, cumpra os objectivos para os quais foi construído.**

***Abstract* – This paper describes the most important aspects of the hardware and software architectures of the robot *Sniffer*, built specifically to participate on the Micro-Rato 2000 robotics contest of the University of Aveiro, where it won the "Innovation Prize". The main hardware components are a micro-controller from the Intel 8051 family and a Xilinx XC4005XL FPGA. The former is used to run the control algorithms while the later performs some low-level tasks. The control software is based on the arbitration of autonomous and reactive behaviors. This approach makes its programming very flexible and fully extensible and allows the robot to achieve the goals for which it was built without any previous knowledge about the labyrinths.**

I. INTRODUÇÃO

O robot *Sniffer* (figura 1) foi um dos 26 concorrentes que participaram na 5ª edição do concurso de robótica Micro-Rato da Universidade de Aveiro que se realizou a 17 de Maio deste ano. Este robot resultou de algumas alterações introduzidas no seu antecessor, o robot *Dyno* [1], o qual participou na edição anterior do mesmo concurso. Algumas das alterações foram motivadas por modificações efectuadas pela organização no regulamento e nas especificações técnicas do concurso. As outras visaram melhorar o seu desempenho e resultaram do conhecimento adquirido na participação da equipa na edição do concurso Micro-Rato de 1999.

No seu circuito de controlo foram utilizados um microcontrolador da família 8051 e uma *Field Programmable Gate Array* (FPGA). Sendo um dispositivo lógico programável, a FPGA permitiu por um lado concentrar num único circuito integrado funções normalmente dispersas por vários (ex. *latch* e descodificação de endereços) e por outro desempenhar funções repetitivas de forma eficiente sem a intervenção do microcontrolador (ex. controlo PWM dos motores).

O paradigma usado na programação do robot *Sniffer* é baseado na arquitectura *Subsumption* [2]. Nesta arquitectura os vários comportamentos que controlam um robot possuem diferentes prioridades, executam em paralelo (pelo menos conceptualmente) e quando um comportamento mais prioritário ordena a execução de uma operação, inibe todos os de menor prioridade. A principal diferença entre esta arquitectura e a implementada no robot *Sniffer* reside na adição de um árbitro que para além de implementar o mecanismo de prioridades, permite que a opinião de um comportamento menos prioritário seja considerada no caso das acções ordenadas pelos mais prioritários o permitirem, aumentando assim a probabilidade da decisão tomada ser a mais adequada.

Além desta introdução, este artigo está organizado em três secções. As duas primeiras descrevem de forma sucinta a estrutura do hardware e do software do robot *Sniffer*. Na última secção são apresentadas as conclusões.

Figura 1 – Aspecto físico do robot *Sniffer*.

II. HARDWARE

O hardware fornecido pela organização para construção deste robot consistiu numa base de madeira, motores,

rodas e sensores de infravermelhos (IV) para detecção de obstáculos, do farol e do chão da área de chegada.

Como placa principal de controlo foi utilizada uma XS40 desenvolvida pela *XESS Corporation* [3] contendo os seguintes componentes (figura 2):

- um microcontrolador 80C154 da Intel;
- uma FPGA XC4005XL da Xilinx [4];
- 32 Kbytes de RAM;
- 256 Kbits de EEPROM para configuração da FPGA;
- interface paralelo (*Centronics*) para programação e depuração.

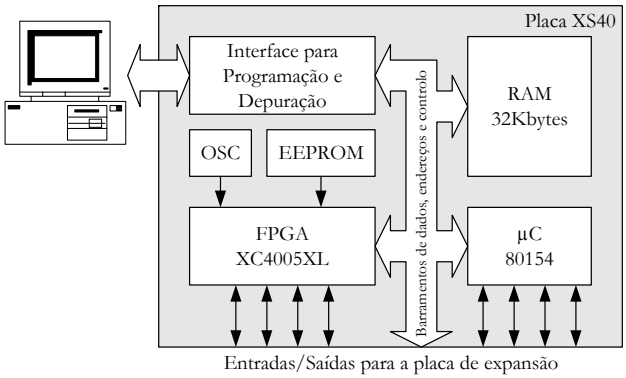


Figura 2 – Diagrama de blocos da placa de controlo XS40.

Especificamente para aplicação no robot foram desenvolvidas as seguintes placas (figura 3):

- Expansão para a XS40 constituída por 16 entradas digitais, 16 saídas digitais, 8 entradas analógicas, 32 Kbytes de RAM protegida por um supervisor de tensão e uma pilha de *backup*;
- Interface para ligação dos sensores e motores contendo circuitos de comutação e de filtragem, *drivers* e conectores.

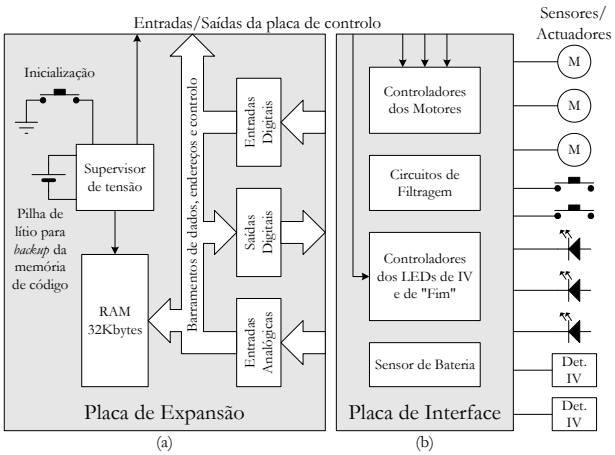


Figura 3 – Diagramas de blocos das placas de (a) expansão (b) interface.

O circuito de supervisão, juntamente com a pilha de *backup*, garante que o conteúdo da memória de programa é preservado mesmo na ausência de tensão de alimentação. Além disso, controla também o sinal de activação da memória, inibindo-o no caso da tensão de alimentação baixar perigosamente. Entre as duas edições do concurso de 1999 e 2000 foi possível observar que o tempo de retenção do programa é superior a um ano.

A figura 4 ilustra a disposição das placas na parte superior da base do robot. À direita temos a placa de controlo, em baixo a placa de expansão e à esquerda a de interface.

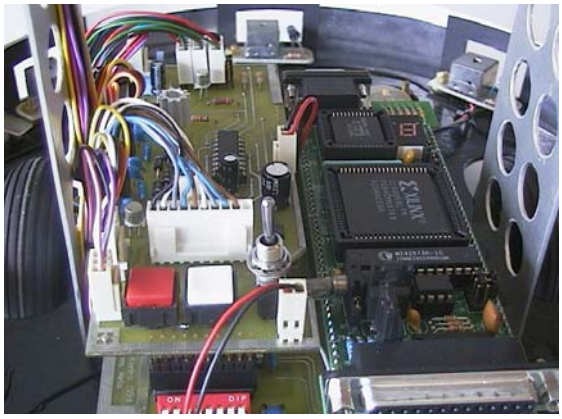


Figura 4 – Localização das placas de controlo, expansão e interface.

A FPGA da placa de controlo é utilizada nas seguintes funções (figura 5):

- Descodificação e *latch* de endereços;
- Geração dos sinais de controlo para os portos de entrada/saída;
- Geração dos sinais PWM para os motores;
- Obtenção de diversas frequências de relógio.

Para o projecto do circuito na FPGA foi utilizada a ferramenta *Xilinx Foundation Express* [5] para captura do esquemático, implementação e geração do ficheiro de configuração. A programação do robot consiste no carregamento desse ficheiro na FPGA e na transferência do programa para a memória de código do microcontrolador, sendo ambas as operações efectuadas através do porto paralelo de um PC.

Os sinais de controlo PWM dos motores e dos *leds* emissores de IV usam pinos dedicados da FPGA, sendo gerados por esta sem a intervenção do microcontrolador. Este interage com a FPGA somente para especificar um novo valor para a velocidade dos motores ou para alterar o estado de activação dos *leds* de IV.

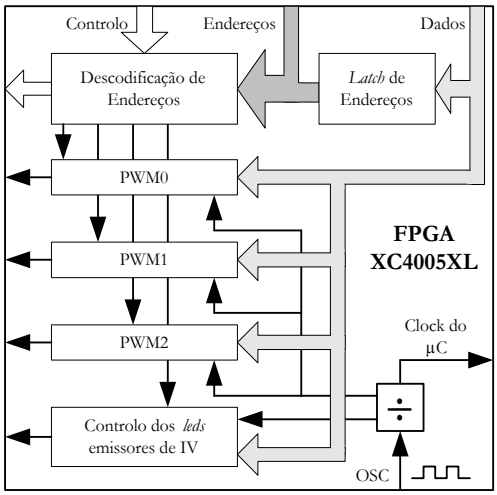


Figura 5 – Componentes implementados na FPGA.

Foi também desenvolvido um sensor de farol rotativo que possui a vantagem de detectar o farol independentemente da posição do robot, desde que não existam obstáculos no labirinto que o impeçam.

A estrutura deste sensor está ilustrada na figura 6, sendo constituído por um motor DC e um sensor IV sensível ao sinal do farol, montado sobre uma base rotativa. A ligação eléctrica entre as partes fixa e móvel deste sensor é realizada através de escovas de grafite que deslizam sobre pistas de cobre circulares (figura 7 (a)).

Para detectar a passagem do sensor por um ponto de referência, foi usado um par emissor-receptor de IV semelhante ao sensor de chão. A figura 7 mostra fisicamente este sensor.

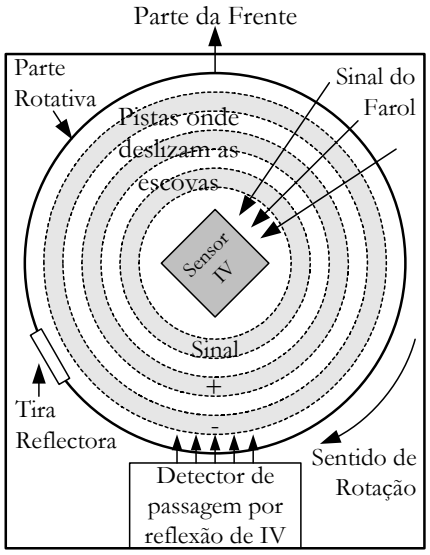
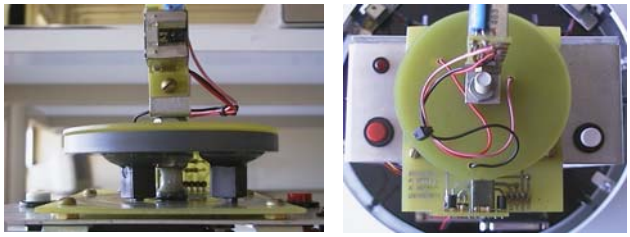


Figura 6 – Estrutura do sensor de farol rotativo.

Na parte inferior do robot encontram-se montados os motores, as rodas e o sensor de chão. Os sensores para detecção de obstáculos estão montados sobre a base do robot. A sua disposição permite cobrir razoavelmente toda a parte da frente.

Um dos melhoramentos introduzidos foi precisamente a diminuição do diâmetro da base para que com o mesmo número de sensores de obstáculos a cobertura da área de interesse seja mais efectiva. Devido à inexistência de sensores na traseira não são executados movimentos para a retaguarda, somente rotações sobre si próprio.

Os botões de "Início", "Paragem", o *led* de "Fim" e o sensor de farol encontram-se na parte superior do robot (figura 7 (b)).



(a) (b)

Figura 7 – Vistas do sensor de farol – (a) frontal; (b) superior.

III. SOFTWARE

A arquitectura do software desenvolvido para controlo do robot *Sniffer* encontra-se representada na figura 8. A linguagem de programação escolhida foi o C tendo sido utilizado um compilador para a família 8051 da Intel.

As operações realizadas pelo robot resultam da arbitragem das sugeridas individualmente por cada um dos seguintes quatro comportamentos: "**Arranque**", "**Detecção de armadilhas**", "**Detecção de obstáculos**" e "**Ir para o farol**". Cada um destes comportamentos possui uma prioridade diferente na determinação da acção a realizar pelo robot.

O comportamento "Detecção de obstáculos" possui maior prioridade do que o "Ir para o farol" porque a movimentação do robot no labirinto até à área de chegada deve ser efectuada sem colisões. Por outro lado, o comportamento "Detecção de armadilhas" possui uma prioridade ainda mais elevada, pois em determinadas situações, as acções sugeridas pelo comportamento de detecção de obstáculos podem levar o robot a cair numa armadilha. Esta situação ocorre quando, devido ao ambiente que o rodeia, as ordens emitidas por um comportamento fazem com que o robot execute movimentos oscilatórios sem sair da posição em que se encontra. De notar que esta atribuição de prioridades não constitui perigo de colisão, uma vez que no caso de ser detectada uma armadilha, o robot inicia um movimento rotativo sobre si próprio até encontrar uma zona suficientemente livre por onde possa escapar. Finalmente, o mais prioritário é o comportamento "Arranque" que, tal como o próprio nome indica, só é utilizado na fase inicial da prova.

Para aumentar o nível de abstracção a que são programados os comportamentos, isto é, para esconder os detalhes relacionados com a activação de cada um dos motores, foi definido um conjunto de acções que o robot pode executar, como por exemplo, andar para a frente, virar para os lados, rodar sobre si próprio, etc.

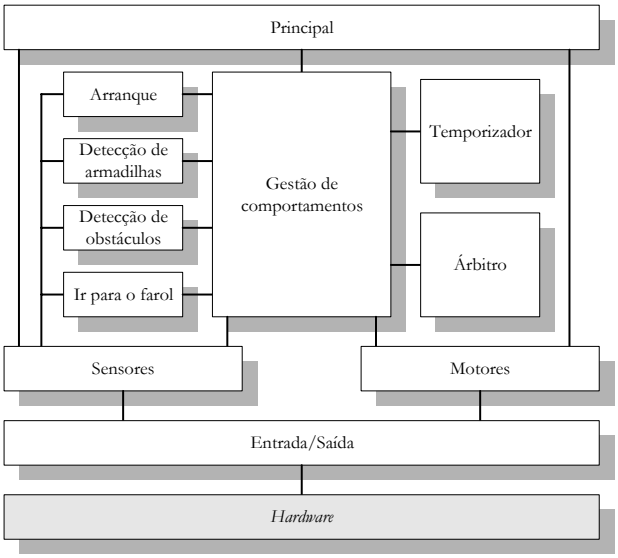


Figura 8 – Arquitectura do software de controlo.

Uma acção é armazenada em 8 bits divididos em três campos (figura 9(a)): movimento (para a frente/para trás/rotação), direcção (esquerda/direita) e intensidade (0…7). A intensidade pode significar, por exemplo, a velocidade com que o robot se desloca ou a rapidez com que vira.

Do ponto de vista de grau de definição, as acções podem ser de três tipos:

- **Completamente definida** – quando existe apenas um bit activo em cada um dos campos "Movimento" e "Direcção" (figura 9 (b)) - utilizada quando a operação estipulada por um dado comportamento não pode ser alterada por um comportamento de menor prioridade;
- **Parcialmente definida** – quando existe mais do que um bit activo nos campos "Movimento" e/ou "Direcção" (figura 9 (c)) - utilizada quando a operação estipulada por um dado comportamento pode ser refinada com a opinião de um comportamento de menor prioridade;
- **Indefinida** - quando todos os bits dos campos "Movimento" e "Direcção" estão activos (figura 9 (d)) - utilizada quando um dado comportamento não pretende estipular nenhuma operação a realizar pelo robot, remetendo toda a responsabilidade para os outros comportamentos.

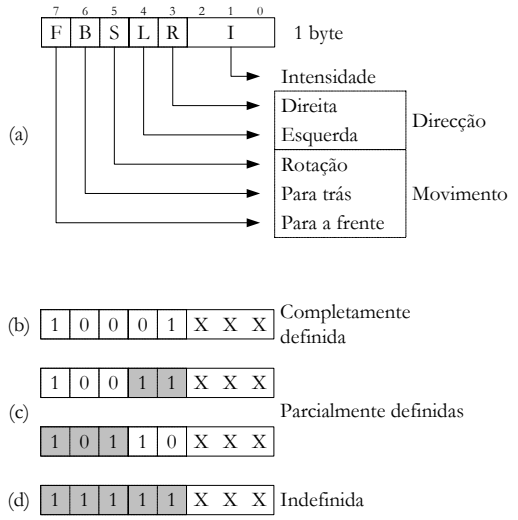


Figura 9 – Acções (a) definição dos campos; (b, c, d) exemplos de vários tipos.

Por questões de simplicidade, apenas foram criadas acções completamente definidas, parcialmente definidas em termos da direcção e indefinidas.

As próximas subsecções descrevem resumidamente cada um dos módulos da figura 8.

Entrada/Saída

Disponibiliza macros e funções para leitura/escrita de todos os portos de entrada/saída do hardware.

Sensores

Contém funções para ler todos os sensores do robot colocando os valores em variáveis globais acessíveis aos restantes módulos do programa. Após a leitura, alguns dos valores dos sensores são pós-processados de forma a simplificar a determinação das acções pelos comportamentos. Por exemplo, os valores lidos dos sensores de detecção de obstáculos são discretizados em apenas cinco distâncias: "muito perto", "perto", "moderado", "longe" e "muito longe".

Motores

Efectua a gestão dos três motores do robot. Devido à variação da velocidade dos motores responsáveis pelo deslocamento do robot, a sua aceleração é limitada, o que contribui por um lado para a suavização dos movimentos do robot e por outro para um aumento da duração das baterias.

Esta limitação é transparente ao utilizador deste módulo. No caso do motor do sensor de farol não é feita qualquer limitação da aceleração, uma vez que roda com velocidade constante.

Este módulo possui, para cada motor de tracção, dois registos de velocidades. O primeiro é utilizado no mecanismo de limitação de aceleração acima descrito. O

segundo permite verificar se o deslocamento do robot foi nulo durante um dado período de tempo.

Temporizador

Este módulo é utilizado para estabelecer o tempo do ciclo de processamento do robot e para controlar variáveis temporizadas de outros módulos. Por parecer razoável, o valor utilizado para o tempo do ciclo de processamento, foi 50ms. Com o microcontrolador a operar a 6MHz, o tempo medido de processamento efectivo foi de 15ms.

Gestão de comportamentos

Este módulo é controlado pelo temporizador, efectuando de forma sequencial a leitura dos sensores, a execução dos comportamentos, a invocação do árbitro e a actualização da velocidade dos motores em função da acção imposta pelo árbitro.

Cada um dos quatro comportamentos recolhe informação dos sensores relevantes, determina a acção a sugerir ao árbitro e coloca-a numa variável global, no módulo de gestão de comportamentos para posterior processamento (figura 10).

A utilização deste módulo facilita a alteração do número de comportamentos do robot, sendo para tal necessário alterar a dimensão da variável global e adicionar ou remover a invocação da respectiva função. Os comportamentos implementados são descritos de seguida.

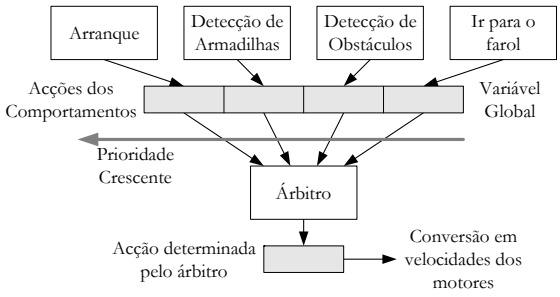


Figura 10 – Gestão dos comportamentos.

Comportamento "arranque"

Este comportamento é utilizado apenas no início da prova forçando o deslocamento do robot para a frente à velocidade máxima. A sua desactivação é feita automaticamente após um intervalo de tempo preestabelecido ou quando são detectados obstáculos muito próximos.

Comportamento "detecção de armadilhas"

O objectivo deste módulo é detectar a paragem ou um comportamento oscilatório repetitivo do robot que impeça o seu deslocamento e, consequentemente, que não prossiga a prova. A detecção é baseada no registo longo de cada um dos motores descrito acima. Nesse caso o robot roda sobre si próprio até encontrar uma zona suficientemente livre por onde possa sair da armadilha.

Comportamento "detecção de obstáculos"

Este comportamento baseia-se apenas na informação recolhida dos sensores de obstáculos de forma a evitar os que forem surgindo no seu caminho. Ao contrário do anterior, a sua operação depende apenas das condições actuais do ambiente, não possuindo portanto qualquer memória.

Comportamento "ir para o farol"

Este comportamento baseia-se essencialmente na informação lida pelo sensor de farol. Ao longo dos 360º de rotação são retiradas cerca de 12 amostras. Uma amostra de máximo só é considerada válida se a diferença entre esta e a amostra de mínimo for superior a um valor predefinido. Desta forma garante-se que na ausência de sinal de farol o movimento do robot é aceitável. Este comportamento integra também mecanismos para seguir paredes, pelo que utiliza informação dos sensores de detecção de obstáculos. De facto, as experiências realizadas permitiram concluir que para os objectivos do concurso só é possível garantir que as decisões tomadas pelo robot são as mais correctas quando se combinam os dois comportamentos.

Árbitro

Este módulo, com base nas acções determinadas pelos vários comportamentos, calcula a acção que o robot deve efectuar de acordo com o esquema de prioridades atrás descrito. Se um comportamento de maior prioridade (ex. detecção de armadilhas) especificar uma acção parcialmente definida (com duas direcções possíveis), o árbitro recorre à acção sugerida por um comportamento de menor prioridade e extrai-lhe a direcção de forma a construir uma acção completamente definida e assim tomar a melhor decisão possível. Se após percorrer todos os comportamentos a acção continuar parcialmente definida é escolhida uma direcção aleatoriamente. Se por outro lado a acção final for indefinida, o robot deslocar-se-á para a frente à velocidade máxima.

Principal

É responsável pela inicialização do robot entrando num ciclo "infinito" quando for premido o botão de "Início". A partir daí o processamento passa a ser controlado pelos módulos de temporização e gestão de comportamentos até que seja detectada uma condição de paragem, como por exemplo, a actuação do botão de "Paragem" ou a conclusão da prova.

IV. CONCLUSÕES

Neste artigo foram descritos alguns dos aspectos mais importantes do hardware e do software do robot *Sniffer*.

As modificações introduzidas relativamente ao seu antecessor, o robot *Dyno*, permitiram melhorar o seu desempenho, o que se traduziu na obtenção do 4º lugar da classificação final do concurso e na conclusão em primeiro lugar de todas as mangas em que participou.

Mais importante foi o Prémio Inovação que lhe foi atribuído, em grande parte devido à arquitectura modular, extensível e baseada em comportamentos autónomos e reactivos do seu software de controlo.

AGRADECIMENTOS

Os autores gostariam de agradecer à comissão organizadora do concurso Micro-Rato o convite para elaborar este artigo e em particular ao Prof. Dr. Luís Almeida pelos seus comentários e sugestões.

REFERÊNCIAS

[1] A. Oliveira, A. Melo, “Dyno – Um Robot com Hardware Reconfigurável”, *Electrónica e Telecomunicações*, vol. 2, nº4, Setembro 1999, pp. 808-810.

[2] R. A. Brooks, "A Robust Layered Control System for a Mobile Robot", *IEEE Journal of Robotics and Automation*, 1986.

[3] XESS Corporation, Catálogo de Produtos, http://www.xess.com.

[4] Xilinx Corporation, “The Programmable Logic Data Book”, 1999.

[5] Xilinx Corporation, “Xilinx Foundation Series Software 2.1I”, 1999.