Sistema de Controlo para o *robot* Bulldozer baseado no executivo ReTMiK

Alexandre Santos, Ana Sargento

***Resumo* – O uso de um kernel de tempo-real no controlo de robots torna-se fundamental quando estão envolvidos vários comportamentos ou processos com requisitos temporais diferentes. No exemplo do concurso Micro-Rato [1], os vários robots necessitam de ter uma grande reactividade no que toca à detecção e desvio de obstáculos, uma mais lenta atenção ao farol e monitorizar a uma baixa frequência a tensão da bateria ou botões de controlo. O kernel de tempo-real permite ainda que os robots tenham uma noção temporal precisa, necessária ao cumprimento dos vários requisitos temporais relacionados com a prova.**

No âmbito da componente prática da disciplina de Sistemas de Tempo Real, opção do 5º ano da LEET e LECT foi desenvolvido um sistema de controlo para o robot Bulldozer (1ª versão), baseado no executivo de tempo real ReTMiK.

I. INTRODUÇÃO

Este projecto surge no âmbito da componente prática da disciplina de Sistemas de Tempo Real, opção do 5º ano da LEET e LECT da UA, e tem como objectivo desenvolver um sistema de controlo para o robot Bulldozer (1ª versão, desenvolvida para o Concurso Micro-Rato'98), baseado no executivo de tempo real ReTMiK. Este sistema de controlo visa dotar o robot de um conjunto de movimentos reactivos básicos que o permita dirigir-se para um farol (emissor de infravermelhos) evitando os obstáculos que se lhe depararem, usando apenas a informação sensorial.

Neste artigo são inicialmente apresentadas algumas características das plataformas de desenvolvimento utilizadas – o kernel de tempo-real ReTMiK e o robot Bulldozer. Seguidamente é apresentada a técnica utilizada no controlo do robot bem como a abordagem seguida em termos da organização dos vários comportamentos em diferentes tarefas e dos períodos atribuídos a cada tarefa. Finalmente são apresentados os resultados obtidos bem como as conclusões gerais do trabalho desenvolvido.

II. PLATAFORMAS DE DESENVOLVIMENTO

A. ReTMiK

O ReTMiK é um kernel de tempo real, desenvolvido em linguagem C (turboC2.0), que funciona sobre a plataforma Kit Det188. Este kernel multitarefa permite definir tarefas periódicas com relação de fase, activadas automaticamente de uma forma transparente para o utilizador. O kernel permite preempção e o código das tarefas é reentrante. O escalonamento é efectuado tendo como base a atribuição de prioridades fixas, indexadas inversamente ao período das tarefas (*rate monotonic*). O kernel ReTMiK permite assim que rotinas em C correntes se transformem em tarefas periódicas, controlando a sua activação e execução concorrente de um modo transparente para o utilizador.

B. Bulldozer

Este robot dispõe de três pares emissor/receptor de infravermelhos usados na detecção de obstáculos, que se encontram-se dispostos segundo a figura 1. A sua gama de visão limita-se à detecção de obstáculos situados entre 6cm e 22cm de distância do sensor.

Para a detecção do farol, o robot dispõe de um sensor de infravermelhos rotativo, que permite efectuar o varrimento de aproximadamente 270º em redor do robot, centrado na frente, como evidencia a figura 1. O movimento deste sensor não é controlado por software, no entanto é solidário com um potenciómetro que permite determinar a sua posição a cada instante.

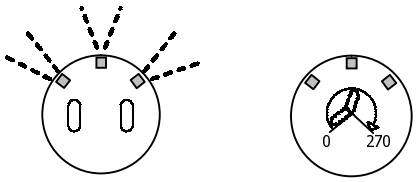


Figura 1 – Disposição dos sensores de obstáculos e do farol

O *robot* possui dois motores independentes, cada um ligado a uma roda, e são ainda disponibilizados 2 botões de pressão usados para iniciar/parar o próprio *robot*.

III. TÉCNICA DE CONTROLO

As técnicas de controlo adoptadas têm o objectivo de tornar os movimentos do *robot* suaves de modo a que este, por um lado, tenha um comportamento elegante e, por outro, mantenha a sua orientação e distância aos obstáculos controlada. O controlo baseia-se na actuação nos motores com valores de velocidade dinâmicos, calculados em função dos valores devolvidos pelos sensores.

A. Orientação pelo farol

No seguimento do farol são aplicadas velocidades diferenciais aos motores em função do desvio medido:

$V_{DIR} = V_{MED} + (POS_{FRENTE} - POS_{EFECTIVA}) * Kp1,$

$V_{ESQ} = V_{MED} - (POS_{FRENTE} - POS_{EFECTIVA}) * Kp1,$

em que V_{MED} corresponde à velocidade média aplicada aos motores, $POS_{FRENTE} - POS_{EFECTIVA}$ corresponde ao desvio angular medido entre a direcção do robot e a direcção do farol e finalmente Kp1 é uma constante de proporcionalidade. O controlo é efectuado por um compensador proporcional, em que Kp1 foi determinada experimentalmente, de forma a aproximar o comportamento do robot ao comportamento desejado. Como resultado o *robot* vira tanto mais quanto mais desviado da trajectória desejada estiver.

B. Seguimento de Parede

No seguimento de parede as velocidades a imprimir aos motores são diferenciais e calculadas em função da distância do robot à parede (esquerda ou direita, respectivamente), técnica conceptualmente semelhante à utilizada no seguimento do farol:

$V_{DIR} = V_{MED} +/- (DIST_{DESEJADA} - DIST_{EFECTIVA}) * Kp2,$

$V_{ESQ} = V_{MED} -/+ (DIST_{DESEJADA} - DIST_{EFECTIVA}) * Kp2,$

sendo V_{MED} conforme referido acima, Kp2 uma contante de proporcionalidade e $DIST_{DESEJADA} - DIST_{EFECTIVA}$ a diferença entre a distância desejada e a medida.

C. Outros Movimentos

Para dotar o *robot* de alguma inteligência no que respeita a efectuar o contorno de cantos de parede por dentro e por fora, foram adicionados movimentos pré-programados simples, tais como rodar sobre si próprio num determinado sentido ou andar em frente durante um determinado tempo (tendo sempre em atenção obstáculos que possam surgir).

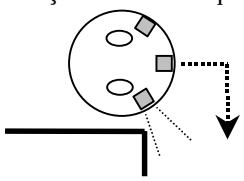


Figura 2 – Disposição dos sensores de obstáculos e do farol

Nesta segunda situação (fig. 2) o *robot* deve seguir em frente durante um pequeno período de tempo, para não chocar com a parede, e de seguida rodar à direita 90 graus, garantindo-se assim um movimento global suave e preciso.

Os movimentos de rotação são ainda utilizados para se desviar de obstáculos que se deparem pela frente, demasiado perto.

V. ABORDAGEM

A. Organização em Tarefas

Foi constituído um sistema com 7 tarefas, de acordo com as várias actividades que o robot tem de desempenhar.

A tarefa *monitor* é responsável por monitorizar o estado de carga da bateria e também dos botões de pressão usados para arrancar/imobilizar o *robot*.

A tarefa *detecta_farol* efectua uma amostragem (a uma frequência relativamente elevada) do valor detectado pelo sensor do farol, associando esse valor à posição lida do potenciómetro. No final de cada varrimento, é actualizada uma variável global do sistema com a posição detectada.

A tarefa *le_sens* faz a amostragem periódica dos sensores de obstáculos, guardando os valores lidos em variáveis globais acessíveis por qualquer tarefa.

As tarefas *mov_farol*, *segue_prd_dir* e *segue_prd_esq* determinam os *setpoints* a enviar para os motores segundo os algoritmos descritos na secção anterior. Estes valores são assim calculados periodicamente independentemente de serem ou não utilizados.

Finalmente, a tarefa *arbitro* determina o movimento a adoptar mediante os valores fornecidos pelos sensores e um histórico relativo ao movimento anterior. Determinado o movimento, são enviados para os motores os valores de velocidade respectivos, previamente calculados por outra tarefa. O diagrama da figura 3 descreve a filosofia usada no controlo global do robot, implementada nesta tarefa.

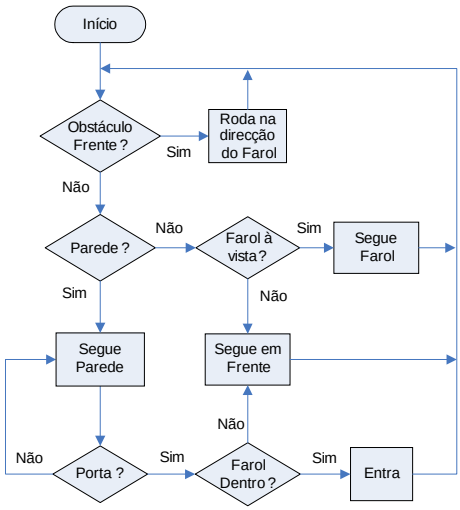


Figura 3 - Algoritmo de navegação implementado no *árbtiro*

B. Períodos

O valor escolhido para o tick do sistema foi de 10ms. Este valor corresponde ao valor mínimo recomendado para que o overhead associado à execução do kernel não seja significativo (inferior a 10%).

A tarefa *monitor* foi criada com um período de 0,5s, valor considerado suficiente para efectuar a amostragem dos botões de pressão, dado que define o tempo máximo de resposta (validação) da pressão de um destes botões.

A tarefa *detecta_farol* foi criada com um período de 20ms, de modo a obter uma resolução razoável para a posição relativa do farol. O valor do sensor de farol é amostrado 35 vezes em cada varrimento (duração de 0,7s), obtendo-se uma resolução de aproximadamente 8º.

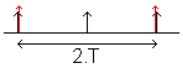
O período escolhido para a *le_sens* foi de 100ms, garantindo a detecção de obstáculos, estáticos ou móveis, que se aproximem do robot a uma velocidade igual à máxima do próprio robot. Os dois casos mais críticos são:

i. Detecção de um outro robot que se movimente na mesma direcção e sentido contrário, à mesma velocidade, considerando jitter máximo.

Dada a gama de distâncias para detecção de obstáculos se situar no intervalo 6cm a 22cm, o período da tarefa deverá ser inferior ao seguinte valor:

- Velocidade máxima do robot: $v \approx 0.25 m/s$
- Velocidade máxima de aproximação: $2 * v \approx 0.50 m/s$
- Jitter máximo (o tempo de percorrer toda a gama de 22cm-6cm=16cm tem que abarcar 2 períodos):

$$2 * T \leq \frac{22cm - 6cm}{2 * v} \quad \Leftrightarrow \quad T \leq 160ms$$



ii. Detecção de um obstáculo estático quando o robot se encontra em rotação, considerando jitter máximo

A velocidade máxima de aproximação será a velocidade linear máxima de rotação de um dos sensores laterais. O robot demora aproximadamente 2.4s a rodar 4π (duas voltas) e tem um raio de 12.75cm.

- Velocidade angular do robot: $w \approx \frac{4\pi}{2.4s} = 5.23 rad/s$
- Velocidade linear do sensor: $v \approx r * w \approx 0.69 m/s$
- Jitter máximo (o tempo de percorrer os 16cm tem que abarcar 2 períodos):

$$2 * T \leq \frac{22cm - 6cm}{v} \quad \Leftrightarrow \quad T \leq 116ms$$

Assim, utilizando um período de 100ms está salvaguardado o caso mais crítico que se refere à aproximação de um obstáculo quando o robot se encontra em rotação.

O período da tarefa *mov_farol* foi escolhido de forma a limitar a validade do valor da posição do farol, e respectivamente dos setpoints a actuar nos motores, quando estes valores não reflectem a actual posição do *robot* (posição do farol actualizada apenas cada 0.7s). Assim, esta tarefa após usar a informação da posição do farol, marca-a como "fora de prazo", sendo a tarefa *detecta_farol* responsável pelo refrescamento desta informação. O período escolhido, determinado experimentalmente, foi de 350ms.

O período das tarefas *seg_prd_dir* e *segue_prd_esq* foi escolhido em função, por um lado, do período da tarefa *le_sens*, e por outro, do tempo de resposta dos motores (da ordem das dezenas de milisegundos), uma vez que não faz sentido efectuar cálculos sem que haja novos valores dos sensores de obstáculos e também antes sequer do robot se ter movido. Assim, o período escolhido foi de 100ms.

A tarefa *arbitro* é responsável por tomar a decisão mais difícil: "Que movimento adoptar em cada instante?". Assim, o seu período deve ser igual ao menor de entre as tarefas que lhe fornecem dados, isto é 100ms (*le_sens*).

VI. RESULTADOS

O *robot* segue efectivamente o farol, desviando-se de obstáculos que surgem no seu campo de visão, ainda que em situações concretas não adopte o comportamento que mais rapidamente o conduziria ao seu objectivo (ir ter com o farol). Este comportamento é aceitável, pois o robot possui inteligência limitada por não ter uma visão global do ambiente que o rodeia.

Foi ainda testada a reactividade do robot na presença de obstáculos estáticos ou móveis, observando-se uma rápida e atempada resposta por parte do robot, tendo-se desviado sem qualquer problema dos obstáculos em causa.

VI. CONCLUSÕES

Sempre que num sistema estão envolvidas actividades com características temporais diferentes torna-se útil a utilização de um kernel de tempo-real. Esta utilidade é notória no controlo dos *robots* que participam no concurso Micro-Rato, os quais necessitam de ter elevada reactividade na detecção e desvio dos obstáculos, uma reactividade mais lenta relativamente á detecção do farol e uma reactividade ainda mais lenta utilizada na monitoração da tensão na bateria e botões de controlo.

A utilização de um kernel de tempo-real permite, por um lado, tirar partido das diferentes velocidades dos vários elementos físicos que compõem o sistema, por outro lado, permite dotar o sistema de uma noção temporal precisa, necessária para estabelecer coerência temporal com o meio envolvente. Assim, cada decisão pode ser tomada ao seu ritmo, independentemente das outras actividades que se encontram a decorrer, optimizando o tempo de ocupação do processador e actuando no momento certo.

Neste artigo é descrito o desenvolvimento de um sistema de controlo para o robot Bulldozer (1ª versão), baseado no executivo de tempo real ReTMiK.

Obteve-se um conjunto de tarefas com características temporais diferentes, fazendo um aproveitamento dos diferentes ritmos de execução e resposta dos vários componentes do sistema. Coincidentemente, as tarefas *le_sens*, *segue_prd_dir*, *segue_prd_esq* e *árbitro* acabaram por ter períodos iguais, mas este facto deve-se a condicionantes diferentes, tal como foi já referido.

REFERÊNCIAS

[1] http://microrato.ua.pt, Página do concurso Micro-Rato da UA;

[2] http://sweet.ua.pt/~lda/retmik/retmik.html, Página do ReTMiK;

[3] Frederico M. Santos, Valter F. Silva, Luís Almeida, "Auto-localização em pequenos robôs móveis e autónomos: O caso do robô Bulldozer IV", Acta do Encontro Científico do ROBOTICA 2002 – Festival Nacional de Robótica.