



Departamento de Electrónica e Telecomunicações  
Universidade de Aveiro

Modalidade *Ciber*  
do Concurso *Micro-Rato* 2006  
Regras e Especificações Técnicas



(Fevereiro de 2006)

## Conteúdo

<b>1</b>	<b>Introdução</b>	<b>3</b>
<b>2</b>	<b>Sistema de Simulação</b>	<b>3</b>
2.1	Software de Apoio . . . . .	4
<b>3</b>	<b>Corpo do Robô Virtual</b>	<b>5</b>
3.1	Sensores . . . . .	5
3.2	Actuadores . . . . .	7
3.3	Botões . . . . .	8
<b>4</b>	<b>Área de Jogo</b>	<b>8</b>
<b>5</b>	<b>Competição</b>	<b>9</b>
5.1	Estrutura da Competição . . . . .	10
5.2	Requisitos aos Agentes Robóticos . . . . .	10
5.3	Desafio . . . . .	11
5.4	Pontuação . . . . .	11
5.5	Classificação . . . . .	13
5.6	Circunstâncias Anómalas . . . . .	13
5.7	Júri . . . . .	14
5.8	Árbitro . . . . .	14
<b>6</b>	<b>Parâmetros de simulação</b>	<b>14</b>
<b>7</b>	<b>Desenvolvimento de Agentes Robóticos</b>	<b>15</b>
7.1	Protocolos de Comunicação . . . . .	16
7.2	Modelos de Simulação . . . . .	18
7.3	Biblioteca de Suporte em C . . . . .	20
<b>8</b>	<b>Diferenças para a Edição Anterior</b>	<b>22</b>

## 1 Introdução

A modalidade *Ciber* do concurso *Micro-Rato*, daqui em diante designada concurso *Ciber-Rato*, é uma competição entre robôs virtuais, que decorre num ambiente simulado numa rede de computadores. Um sistema de simulação cria uma área virtual de jogo, com uma grelha de partida, várias áreas-alvos, cada uma sinalizada por um farol, e povoada de obstáculos, e nela faz movimentar vários robôs virtuais. Os concorrentes participam com agentes robóticos, programas, cujos papéis são comandar os movimentos dos robôs virtuais, com o propósito de alcançar determinados objectivos.

Os corpos de todos os robôs virtuais são iguais, sendo cada um constituído por um corpo cilíndrico, equipado com sensores, actuadores e botões de comando. O sistema de simulação é responsável por determinar os valores medidos pelos diversos sensores e actuar nos botões de comando. Os agentes robóticos recebem a informação referente aos sensores e botões e comandam os elementos de actuação.

Para a edição de 2006 do concurso *Ciber-Rato*, a cada concorrente (leia-se agente robótico) é posto o seguinte desafio: partir da sua posição na grelha de partida, visitar todas as áreas-alvos e regressar à sua posição na grelha de partida. A classificação do concorrente depende dos seguintes aspectos: concretização dos objectivos estabelecidos, tempo de concretização e penalizações sofridas.

Este documento descreve as regras e especificações técnicas em vigor para a edição 2006 do concurso *Ciber-Rato*. Os leitores familiarizados com as regras da edição do ano anterior podem consultar a secção 8 para uma breve descrição das alterações introduzidas este ano.

## 2 Sistema de Simulação

O sistema virtual de suporte ao concurso *Ciber-Rato* é baseado numa arquitectura distribuída, onde participam 5 aplicações: 1 simulador, 1 visualizador e 3 agentes robóticos (ver figura 1).

O simulador é responsável por:

- Implementar os corpos dos robôs virtuais.
- Determinar os valores medidos pelos sensores dos robôs virtuais em prova, enviando-os aos respectivos agentes robóticos.
- Executar os movimentos dos robôs dentro da área de jogo, de acordo com as ordens de movimento recebidas dos agentes robóticos e pelas restrições impostas pelo ambiente.
- Determinar a pontuação de cada robô em prova, tendo em conta as infracções cometidas.
- Enviar ao visualizador toda a informação necessária para mostrar o movimento dos robôs dentro da área de jogo e mostrar a sua pontuação.

O visualizador é responsável por:

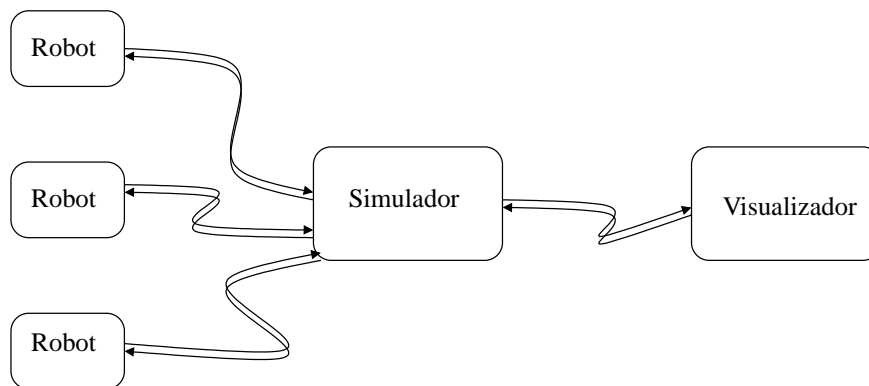


Figura 1: Arquitectura geral do sistema de simulação.

---

- Mostrar graficamente a área de jogo com o movimento dos robôs em prova.
- Mostrar os estados e pontuações dos robôs em prova.
- disponibilizar um painel de controlo para arranque e paragem da prova.

O sistema de simulação é comandado pelo tempo (*cycle driven*). Quer isto dizer que o sistema evolui discretamente no tempo, com um passo dado pela duração temporal do ciclo de simulação. Em cada ciclo o simulador: envia aos agentes robóticos os valores medidos pelos seus sensores, recebe deles os comandos sobre os actuadores, movimenta os corpos dos robôs de acordo com esses comandos e actualiza as pontuações. Para a edição de 2006 o ciclo de simulação tem uma duração entre os 40 e os 80 milissegundos.

Todos os elementos em jogo, nomeadamente área de jogo, obstáculos e robôs, são virtuais, pelo que não há necessidade de usar uma unidade real de comprimento. Assim, a unidade de comprimento adoptada pelo concurso *Ciber-Rato* designa-se  $u_m$ , e não tem qualquer significado físico.

Todos os tempos medidos pelo sistema de simulação são múltiplos do ciclo de simulação. A unidade de tempo adoptada pelo concurso *Ciber-Rato* designa-se  $u_t$ , e corresponde à duração do ciclo de simulação.

## 2.1 Software de Apoio

Através do seu sítio oficial (<http://microrato.ua.pt>), a *Organização* disponibiliza vários pacotes de software de apoio ao concurso, que incluem: executáveis do simulador, executáveis do visualizador, executáveis do reproduzidor de provas previamente gravadas, bibliotecas de suporte ao desenvolvimento de agentes robóticos, código fonte de agente robóticos para serem usados como modelo e executáveis de agentes robóticos que participaram em edições anteriores do concurso.

Cada pacote de software possui instruções sobre a sua instalação e utilização.

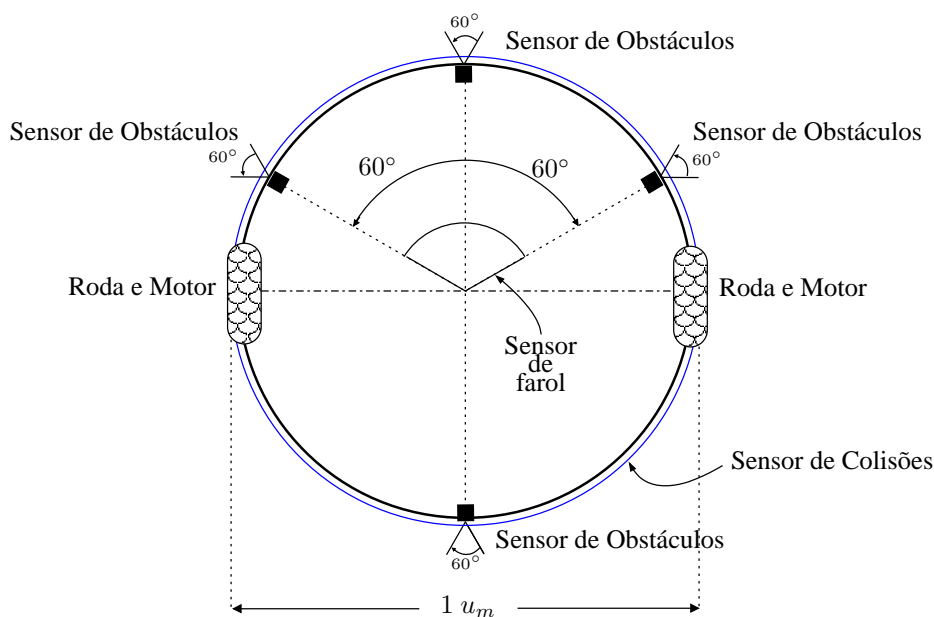


Figura 2: Estrutura geral do corpo do robô virtual.

### 3 Corpo do Robô Virtual

Os corpos dos robôs virtuais têm uma planta circular, com diâmetro  $1 u_m$ , e estão equipados com sensores, actuadores e botões de comando (ver figura 2).

#### 3.1 Sensores

Os elementos sensoriais de cada robô incluem quatro sensores de obstáculos, um sensor de farol por cada farol/área-alvo, uma bússola, um sensor de colisões, um sensor de chão e um GPS. Estes sensores pretendem simular dispositivos sensoriais que se podem encontrar em robôs reais, pelo que as suas medidas estão sujeitas a ruído. Faz-se a seguir uma descrição dos vários tipos de sensores, apresentado-se um resumo na tabela 1. Os valores de ruído apresentados serão os praticados durante a edição do concurso de 2006.

- Os **sensores de obstáculos** permitem medir a distância entre o robô e os obstáculos que o rodeiam, incluindo os outros robôs. Cada um destes sensores possui um ângulo de cobertura de 60 graus, 30 para cada lado do seu eixo. Podem ser colocados em qualquer ponto na periferia do robô, mas sempre em posição radial relativamente ao seu centro. Por defeito, são colocados um sobre o eixo frontal do robô, dois a 60 graus deste, um para cada lado, e um virado para trás (ver figura 2).

Tabela 1: Gama dinâmica, resolução e níveis de ruído dos vários tipos de sensores que equipam um robô virtual.

Tipo de sensor	Gama dinâmica	Resolução	Tipo de ruído	Desvio padrão
Obstáculos	0,0 a 100,0	0,1	aditivo	0,25
Farol	-60 a +60	1	aditivo	5
Bússula	-180 a +180	1	aditivo	5
Colisões	0 ou 1	..... não aplicável .....		
Chão	0 ou 1	..... não aplicável .....		

O valor medido por um sensor de obstáculos é inversamente proporcional à menor das distâncias a que se encontram os obstáculos cobertos pelo seu ângulo de cobertura, e varia entre 0,0 e 100,0, com uma resolução de 0,1. O valor medido está sujeito a ruído aditivo, seguindo uma distribuição normal (gaussiana) com valor médio igual a zero e desvio padrão igual a 0,25.

- Os **sensores de farol** estão montados no centro do robô, virados para a frente, e possuem um ângulo de cobertura de 120 graus, 60 para cada lado. Medem a posição angular do farol relativamente ao eixo frontal do robô. Cada robô está equipado com um sensor por cada farol/área-alvo existente no labirinto.

A visibilidade do farol pode ser anulada por duas razões:

- o robô está de costas para o farol, colocando-o fora do ângulo de cobertura do sensor;
- a existência de paredes com altura superior à do farol pode ocultá-lo, criando zonas de sombra.

Se o farol for visível, o valor medido pelo sensor de farol varia entre -60 e +60 graus, com uma resolução de 1 grau, e está sujeito a ruído aditivo, seguindo uma distribuição normal (gaussiana) com valor médio igual a zero e desvio padrão igual a 5 graus.

O sensor de farol é lento, não conseguindo produzir medidas em todos os ciclos de simulação. Para a edição de 2006 tem uma latência de 2, ou seja, produz medições de 2 em 2 ciclos.

- A **bússula** está montada no centro do robô e mede a distância angular da frente do robô relativamente ao *Norte virtual*. Considera-se que o semi-eixo positivo das abcissas está apontado para o *Norte virtual*.

O valor medido pela bússula varia entre -180 e +180 graus, com uma resolução de 1 grau, e está sujeito a ruído aditivo, seguindo uma distribuição normal (gaussiana) com valor médio igual a zero e desvio padrão igual a 5 graus.

- O **sensor de colisões** é constituído por uma estrutura em anel, colocada à volta do robô, que é activada sempre que este colide com obstáculos ou outros robôs. O valor de saída é binário, sendo igual a 1 sempre que haja colisão e 0 no caso contrário.

Tabela 2: Gama dinâmica, resolução e níveis de ruído dos vários tipos de actuadores que equipam um robô virtual.

Tipo de actuador	Gama dinâmica	Resolução	Tipo de ruído	Desvio padrão
Motor	-0,15 a +0,15	0,001	multiplicativo	3%
led Fim	On ou Off	.....	não aplicável	.....
led Regresso	On ou Off	.....	não aplicável	.....
led Visita	On ou Off	.....	não aplicável	.....

- O **sensor de chão** é um dispositivo que detecta se o robô se encontra sobre uma área-alvo. O seu valor de saída é um inteiro no intervalo  $[-1, n - 1]$ , em que  $n$  é o número de faróis no labirinto.<sup>1</sup> O sensor devolve  $i$ , com  $i = 0, 1, \dots, n - 1$ , quando o robô se encontra por cima da área-alvo número  $i$ . Quando o robô não se encontra em cima de nenhuma área-alvo o sensor devolve  $-1$ .
- O **GPS** é um dispositivo que determina o posicionamento espacial do robô na área de jogo e está montado sobre o centro do robô. O valor medido não está sujeito a ruído. Este sensor apenas deve ser usado na fase de desenvolvimento dos agentes robóticos. Durante a competição está desactivado.

### 3.2 Actuadores

O robô virtual possui 2 motores e 3 leds de sinalização. Os motores pretendem simular, embora de uma forma rudimentar, motores reais, pelo que o seu funcionamento está sujeito a inércia e ruído. Faz-se a seguir uma descrição dos vários tipos de actuadores, apresentado-se um resumo na tabela 2. Os valores de ruído apresentados serão os praticados durante a edição do concurso de 2006.

- Os **motores** comandam o movimento de duas rodas, colocadas na periferia do robô, sobre a linha diametral perpendicular ao seu eixo frontal (ver figura 2). O movimento do robô é função da potência aplicada aos motores que comandam as duas rodas. Movimentos de translação e rotação do robô são possíveis por aplicação de potência adequada a cada um dos motores. Assim, aplicando a mesma potência aos dois motores, o robô desloca-se no sentido do seu eixo; aplicando potências com o mesmo valor absoluto mas sinais contrários, o robô roda sobre o seu centro.

A potência aceite pelos motores à entrada varia entre  $-0,15$  e  $+0,15$ , com uma resolução de  $0,001$ . A relação de transformação da potência de entrada em movimento do robô está descrita na secção 7.2, estando sujeita a um ruído multiplicativo, seguindo uma distribuição normal (gaussiana) com valor médio igual a 1 e desvio padrão igual a 3%.

<sup>1</sup>O simulador numera os faróis sequencialmente a partir de 0.

Uma ordem de comando sobre um motor tem efeito até que haja uma nova ordem. Assim, se aplicar uma potência de 0,15 ao motor de uma roda num determinado instante de tempo, essa potência será continuamente aplicada nos instantes de tempo seguintes, até que o simulador receba nova ordem de comando sobre o mesmo motor.

- Os três *leds*, designados *Visita*, *Regresso* e *Fim*, são usados para sinalizar a concretização de determinados objectivos. O *led Visita* deve ser usado por um robô para sinalizar que se encontra sobre a área-alvo de um farol. O *led Regresso* deve ser usado por um robô para sinalizar que vai iniciar o regresso ao seu ponto de partida. Finalmente, o *led Fim* deve ser usado por um robô para sinalizar que concluiu a sua prova. Compete ao robô acender os *leds* de acordo com as regras do concurso.

### 3.3 Botões

O robô virtual está ainda equipado com dois botões, designados *Arranque* e *Paragem* que comandam o arranque e as interrupções das provas. O comando destes botões é realizado pelo simulador, sendo os agentes robóticos informados sempre que isso acontecer. O botão *Arranque* é accionado pelo simulador para dar início à prova e para reiniciar uma prova previamente interrompida. O botão *Paragem* é accionado pelo simulador para interromper uma prova.

## 4 Área de Jogo

A área de jogo é constituída por uma superfície rectangular, delimitada por paredes, contendo no seu interior obstáculos, faróis, áreas-alvos e uma grelha de partida. A grelha de partida define as posições e orientações iniciais dos robôs que participam numa determinada prova. As áreas-alvos definem zonas na área de jogo, cada uma sinalizada por um farol, por onde os robôs têm necessariamente que passar no sentido de concretizarem um dos objectivos do concurso. Um farol encontra-se no centro de cada área-alvo. Os obstáculos são blocos espalhados pela área de jogo, no sentido de dificultarem a progressão dos robôs. A figura 3 mostra uma configuração possível para a área de jogo. Em relação à edição do concurso de 2006 serão observadas as seguintes regras:

### Rectângulo de jogo

1. A superfície rectangular da área de jogo tem, no máximo 14  $u_m$  de altura por 28  $u_m$  de largura.
2. As paredes delimitadoras da área de jogo são detectáveis pelos sensores de obstáculos.

### Obstáculos

3. Os obstáculos têm plantas poligonais, com faces de dimensões mínimas de 0,4  $u_m$  e com ângulos internos entre faces adjacentes (esquinas) de valor nunca inferior a 90 graus.

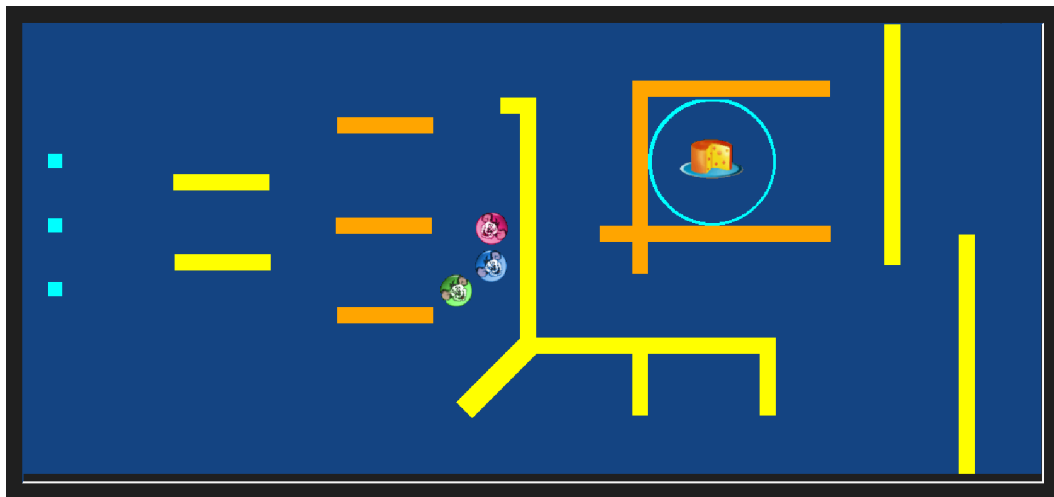


Figura 3: Panorâmica da área de jogo.

4. A altura de alguns obstáculos poderá ser superior à altura do farol, criando zonas de sombra.
5. A distância mínima entre quaisquer dois obstáculos que não estejam juntos será de  $1,5 u_m$ . Para este efeito as paredes delimitadoras da área de jogo são consideradas obstáculos.

### Grelha de partida

6. A grelha de partida define as posições e orientações iniciais de 3 robôs. A *Organização* assegurará, na medida do possível, condições de equidade entre os 3 pontos da grelha de partida.

### Áreas-alvos

7. Uma área-alvo é definida por um círculo com raio mínimo de  $1,0 u_m$ .
8. Um farol virtual, que não constitui obstáculo ao movimento dos robôs, encontra-se colocado no centro de cada área-alvo. A “luz” por si emitida é apenas detectada pelo respectivo sensor de farol, desde que nenhuma parede o impeça.

## 5 Competição

A modalidade *Ciber* do concurso *Micro-Rato* prevê a participação de um máximo de 21 agentes robóticos. Se o número de propostas de participação for superior a esse valor a *Organização* seleccionará

21, após elaboração duma seriação tendo por base um relatório técnico em que as equipas descrevem sucintamente a estratégia utilizada.

A pré-inscrição dentro dos prazos estabelecidos pelo calendário do concurso será tida em conta na seriação anterior.

A *Organização* poderá prescindir da entrega do relatório se o número de pré-inscrições for inferior a 21.

### 5.1 Estrutura da Competição

A competição está estruturada em 4 mangas. Na primeira e segunda mangas participam todos os concorrentes devidamente inscritos, num máximo de 21. Na terceira manga participam os 9 mais bem classificados ao fim das duas mangas anteriores. Finalmente, na quarta manga, a final, participam apenas os 3 mais bem classificados no fim da terceira manga.

Cada manga está dividida em provas, nas quais competem 3 robôs de cada vez. Os participantes numa determinada manga serão distribuídos por diversas provas, sendo observadas as seguintes regras:

1. A distribuição será feita por sorteio, antes do início da manga.
2. Se o número de participantes a uma manga não for múltiplo de 3, a *Organização* acrescentará 1 ou 2 robôs próprios, de forma a garantir a participação de 3 robôs em cada prova. Estes robôs acrescentados pela *Organização* não contam, obviamente, para efeitos de classificação.
3. Os 3 robôs participantes numa mesma prova serão colocados nas 3 posições da grelha de partida segundo a ordem de sorteio.

A manga final tem 3 provas. No entanto, o sorteio das posições na grelha de partida é apenas feito para a primeira. Os robôs que ocupam as posições 1, 2 e 3 na primeira manga, ocuparão as posições 2, 3 e 1 na segunda e 3, 1 e 2 na terceira.

A configuração da área de jogo, i.e., a distribuição na área de jogo de obstáculos, grelha de partida, farol e área-alvo, pode ser diferente de manga para manga, mas é a mesma para todas as provas de uma manga. Esta configuração é divulgada imediatamente antes do início da manga e após recolha dos agentes robóticos que nela participam.

### 5.2 Requisitos aos Agentes Robóticos

A estrutura computacional de suporte à realização de cada prova de cada manga da edição 2006 do concurso *Ciber-Rato* será constituída por 4 computadores ligados em rede. Um dos computadores correrá as aplicações de simulação e de visualização, sendo os restantes usados para execução dos agentes robóticos dos concorrentes, um por computador.

Todos os computadores têm instalados os sistemas operativos Linux e Windows e possuem processadores e quantidade de memória idênticos. Para a edição 2006 do concurso *Ciber-Rato* a *Organização*

assegurar que os recursos mínimos disponibilizados serão: um processador a 2,0 GHz e 256 MBytes de memória.

O lançamento de cada prova será realizado unicamente pelos elementos da *Organização*, pelo que é definido o formato de execução dos agentes robóticos. Estabelece-se que os agentes robóticos, em Windows ou Linux, serão lançados executando o comando

```
agente -robname nome -host endereço[:porto] -pos posição
```

em que:

`agente` é o nome do programa que implementa o agente robótico;

`nome` é o seu nome, tal como será identificado no painel de pontuações;

`endereço[:porto]` é o endereço IP do simulador, eventualmente com indicação do porto a usar; e

`posição` é a posição do robô na grelha de partida, podendo ter o valor 1, 2 ou 3.

Assim, os agentes robóticos aceites para participação terão de satisfazer este requisito.

### 5.3 Desafio

No início de cada prova, os robôs nela participantes são colocados nas respectivas posições da grelha de partida. O desafio colocado a cada participante desdobra-se em dois objectivos:

1. No primeiro, o robô deverá partir da sua posição na grelha de partida e visitar todas as áreas-alvos. Dentro de cada área-alvo o robô deve acender a *led Visita* que deverá apagar antes que dela saia, sob pena de ser penalizado.
2. No segundo, apenas possível para quem concretiza o primeiro, o robô deverá, dentro de uma área-alvo à sua escolha, assinalar que vai iniciar o regresso, acendendo o *led Regresso*, após o que deverá aproximar-se o mais possível da sua posição na grelha de partida, acendendo o seu *led Fim* quando desejar dar a sua prova por concluída.

A prossecução dos dois objectivos anteriores está sujeita a um tempo máximo, que para a edição do concurso de 2006 estará situado no intervalo de 1800 a 3600  $u_t$ .

### 5.4 Pontuação

No fim de cada prova será atribuída uma pontuação a cada robô que nela participou. Esta pontuação, calculada automaticamente pelo simulador, tem em consideração a concretização de objectivos e o incorrimento em penalizações. É calculada aplicando as seguintes regras:

1. No início de cada prova é atribuída a cada robô participante uma pontuação dada pela expressão  $200 + 100n$ , onde  $n$  representa o número de faróis. Para uma prova com 2 faróis a pontuação inicial é de 400 pontos, para 3 de 500 e assim sucessivamente.
2. Sempre que um robô colida com um obstáculo ou uma parede é penalizado em 5 pontos, que serão automaticamente somados à sua pontuação.
3. Sempre que um robô colida com outro robô é penalizado em 5 pontos, que serão automaticamente somados à sua pontuação. Compete ao simulador decidir sobre a responsabilidade pela colisão entre robôs, podendo, eventualmente, ser imputada a ambos.
4. Por cada nova área-alvo visitada e devidamente assinalada (por acendimento do *led Visita*) o robô ganha uma bonificação de 100 pontos, que serão automaticamente subtraídos à sua pontuação. A entrada numa área-alvo sem o acendimento do *led Visita* não concretiza a visita.
5. Por cada ciclo de simulação em que mantenha aceso o *led Visita* fora de uma área-alvo o robô sofre uma penalização de 5 pontos, automaticamente adicionados à sua pontuação.
6. Ao assinalar devidamente o início da viagem de regresso, (acendimento do *led Regresso*/ numa área-alvo após visitar todas as áreas-alvos), um robô ganha uma bonificação de 100 pontos, que serão automaticamente subtraídos à sua pontuação.
7. O acendimento do *led Regresso*/ fora de uma área-alvo provoca o fim da prova do robô infractor, com a pontuação tida até ao momento.
8. Seja  $D$  a distância mais curta que um robô tem de percorrer para ir do ponto onde assinala o início da sua viagem de regresso até ao ponto correspondente à sua posição na grelha de partida. Sempre que um robô se aproxima da sua posição na grelha de partida de uma distância  $D/100$  a sua pontuação é decrementada de 1 ponto. Sempre que se afasta é incrementada de igual valor.
9. Seja  $T$  o número de  $u_t$  (ciclos de simulação), arredondados à unidade, que um robô demora a percorrer a distância  $D$  anterior, se se deslocar à velocidade máxima. Por cada  $25 u_t$  em excesso relativamente a  $T$  que um robô gastar a regressar ao seu ponto na grelha de partida ser-lhe-á acrescentado 1 ponto à pontuação.
10. Considera-se que um robô terminou a sua prova no momento em que acende o seu *led Fim*, ficando com a pontuação tida nesse momento. (Se um robô realizar a sua prova sem incorrer em penalizações, terminar a uma distância da sua posição na grelha de partida inferior a  $D/100$  e demorar a regressar um tempo inferior a  $T + 25$ , ficará com uma pontuação de 0 pontos.)
  - (a) O robô que no fim do tempo máximo de ida e volta não tenha acendido o seu *led Fim* é penalizado em 15 pontos, somados automaticamente à sua pontuação.
11. A nenhum robô é atribuída uma pontuação superior a 600 pontos:
  - (a) A um robô que no fim da sua prova atinja uma pontuação superior a 600 pontos, é-lhe atribuído aquela pontuação.

- (b) A um robô que tenha sido retirado de prova por decisão do Júri é atribuída uma pontuação de 600 pontos (ver secção 5.6).

Além da pontuação, será também atribuído a cada robô o tempo que demorou a concretizar a visita à sua primeira área-alvo.

## 5.5 Classificação

No final de cada prova é atribuída a cada robô uma pontuação e um tempo (o tempo de visita). A classificação dos robôs será estabelecida pela ordenação ascendente das pontuações atribuídas. Os robôs com igual número de pontos serão seriados de acordo com a ordenação ascendente dos tempos atribuídos.

A atribuição de pontuações e tempos ao longo das várias mangas do concurso é feita da seguinte forma:

1. No final da primeira manga a pontuação e o tempo atribuídos a cada robô correspondem respectivamente à pontuação e ao tempo obtidos na prova realizada nessa manga. Todos os participantes passam à manga seguinte.
2. No final da segunda manga a pontuação e o tempo atribuídos a cada robô correspondem respectivamente à soma das pontuações e à soma dos tempos obtidos nas provas da segunda e primeira mangas. Passam à manga seguinte os 9 mais bem classificados.
3. No final da terceira manga a pontuação e o tempo atribuídos a cada robô correspondem respectivamente à soma das pontuações e à soma dos tempos obtidos na prova da terceira manga e na prova de mais baixa pontuação das primeira e segunda mangas. Passam à manga seguinte os 3 mais bem classificados.
4. No final da quarta e última manga a pontuação e o tempo atribuídos a cada robô correspondem respectivamente à soma das pontuações e à soma dos tempos obtidos nas duas provas dessa manga com mais baixas pontuações.

## 5.6 Circunstâncias Anómalas

O árbitro (ver secção 5.8) poderá decidir pela interrupção da prova sempre que for necessário consultar o Júri. Ao fazê-lo todos os robôs são notificados através do botão *Paragem*, sendo imobilizados no âmbito do simulador. A contagem do tempo também é interrompida.

Da decisão do Júri poderá resultar a eliminação de robôs participantes e a continuação, termo ou repetição da prova.

A eliminação de um robô justificar-se-á se o seu comportamento prejudicar os restantes robôs em prova para além das interferências normais resultantes da coexistência no recinto de jogo. A título de exemplo refira-se:

- colisões deliberadas e/ou repetitivas com os restantes robôs em prova;
- envio repetitivo de mensagens, procurando saturar o sistema.
- bloqueio de uma área-alvo, impedindo outros robôs de a alcançarem.

### **Continuação ou Repetição de uma Prova**

1. O processo de rearranque é controlado pelo árbitro, sendo os robôs notificados através do seu botão *Arranque*. As posições e orientações dos robôs no momento do rearranque são as mesmas que detinham no momento da interrupção.
2. A repetição de uma prova far-se-á com os robôs não eliminados, sendo incorporado um robô da *Organização* por cada robô eliminado.

## **5.7 Júri**

O Júri é a entidade máxima na interpretação e aplicação das regras. Tem por missão verificar a conformidade dos robôs às regras e apoiar o árbitro na fiscalização do cumprimento das mesmas. Apenas o Júri pode aplicar aos robôs as penalizações mais gravosas, nomeadamente a desqualificação da prova ou do concurso.

Através da sua autoridade, o Júri garante a justiça na aplicação das regras e regulamentos. Das decisões do Júri não há recurso.

O Júri é nomeado pela *Organização*.

## **5.8 Árbitro**

O árbitro assegura o cumprimento das regras do concurso. O árbitro poderá interromper a prova sempre que achar necessário consultar o Júri. Nas questões omissas nestas regras o árbitro deverá, obrigatoriamente, consultar o Júri.

O árbitro é nomeado pela *Organização*.

## **6 Parâmetros de simulação**

A configuração do simulador para uma prova (manga) faz-se fornecendo-lhe os seguintes dados:

- Duração do ciclo de simulação e tempo total de prova.
- Níveis de ruído dos sensores e motores.
- Descrições do labirinto e da grelha de partida.

As configurações fazem-se usando ficheiros de texto em formato XML. Existem 3 tipos de ficheiros, correspondendo a 3 *tags* diferentes: `Parameters`, `Lab` e `Grid`. Dada a natureza verbosa da linguagem XML limitamos-nos a apresentar uma descrição de cada tipo.

```
<Lab Name="Default LAB" Height="14" Width="28">
  <Beacon X="24" Y="7,0" Height="4,0" />
  <Target X="24" Y="7,0" Radius="1,5" />
  <Beacon X="14" Y="7,0" Height="4,0" />
  <Target X="14" Y="7,0" Radius="1,5" />
  <Wall Height="5,0">
    <Corner X="10,0" Y="4,0" />
    <Corner X="11,0" Y="4,0" />
    <Corner X="11,0" Y="10,0" />
    <Corner X="10,0" Y="10,0" />
  </Wall>
</Lab>

<Grid>
  <Position X="4,0" Y="9,0" Dir="0,0" />
  <Position X="5,0" Y="7,0" Dir="0,0" />
  <Position X="4,0" Y="5,0" Dir="0,0" />
</Grid>

<Parameters SimTime="1800" CycleTime="60"
  CompassNoise="5,0" BeaconNoise="5,0" ObstacleNoise="0,25"
  MotorsNoise="3,0"
  GPS="Off"
  Lab="lab.xml" Grid="grid.xml" />
```

Note que na última descrição são incluídos atributos que indicam os ficheiros com as descrições do labirinto e da grelha de partida. Qualquer um dos atributos pode estar ausente. Nesse caso assumem os valores por defeito. São definidos valores por defeito para todos os elementos.

## 7 Desenvolvimento de Agentes Robóticos

Nesta secção apresentam-se vários itens que pretendem ser elementos de apoio aos concorrentes no desenvolvimento dos seus agentes robóticos.

## 7.1 Protocolos de Comunicação

A comunicação entre o simulador e os agentes robóticos é baseada em *sockets* UDP, sendo as mensagens textuais e formatadas em estruturas XML. Há 5 mensagens a considerar: pedido de registo de um robô, resposta de recusa, resposta de aceitação, dados sensoriais e ordens de actuação.

### Pedido de registo

O registo de um agente robótico no simulador efectua-se enviando a mensagem de registo para o porto 6000 do endereço IP do computador onde o simulador está a correr. Esta mensagem tem o formato:

```
<Robot Name="nome" Id="pos">
  <IRSensor Id="sid" Angle="sangle"/>
  :
</Robot>
```

em que:

- nome é uma sequência de uma ou mais palavras representando o nome do robô;
- pos é um número indicando a sua posição na grelha de partida.
- sid é um número entre 0 e 3 representando um sensor de obstáculos; e
- sangle é um número entre -180,0 e +180,0 representando a posição angular do sensor na periferia do robô.

As *tags* `IRSensor` são opcionais, devendo ser incluída uma por cada sensor de obstáculos que se pretende posicionar.

### Resposta de recusa

Se o simulador recusa o pedido de registo, responde com a mensagem

```
<Reply Status="Refused"></Reply>
```

### Resposta de aceitação

Se o simulador aceita o pedido de registo, responde com a mensagem

```
<Reply Status="Ok">
  <Parameters SimTime="stime" CycleTime="ctime" NBeacons="nbeacons"
    CompassNoise="cnoise" BeaconNoise="bnoise"
    ObstacleNoise="onoise" MotorsNoise="mnoise"
  </Reply>
```

em que:

- `stime` e `ctime` são números inteiros positivos representando respectivamente o tempo total de simulação e a duração do ciclo de simulação;
- `nbeacons` é um número inteiro representando o número de áreas-alvos/faróis;
- `cnoise`, `bnoise` e `onoise` são números reais representando respectivamente os níveis de ruído da bússola, do sensor de farol e dos sensores de obstáculos;
- `mnoise` é um número real representando o nível de ruído na transformação da potência dos motores em movimento das rodas.

O agente robótico deverá fixar o porto de origem da mensagem resposta e dirigir todas as mensagens subsequentes para esse porto.

### Dados sensoriais

A partir da aceitação do registo, o simulador envia ao agente robótico uma mensagem todos os ciclos de simulação, contendo os dados sensoriais do corpo do robô. Esta mensagem tem a forma:

```
<Measures Time="curtime">
  <Sensors Compass="compassvalue"
    Collision="collisionvalue" Ground="groundvalue">
    <BeaconSensor Id="0" Value="bs0value"/>
    <BeaconSensor Id="1" Value="bs1value"/>
    :
    <IRSensor Id="0" Value="irs0value"/>
    <IRSensor Id="1" Value="irs1value"/>
    <IRSensor Id="2" Value="irs2value"/>
    <IRSensor Id="3" Value="irs3value"/>
    <GPS X="xvalue" Y="yvalue" Dir="dirvalue"/>
  </Sensors>
  <Leds EndLed="endledvalue" ReturningLed="retledvalue"/>
  <Buttons Start="startvalue" Stop="stopvalue"/>
</Measures>
```

em que:

- `curtime` é um número inteiro positivo representando o tempo transcorrido de simulação;
- `compassvalue` é um número real representando a medição da bússola;

- `bs0value` e `bs1value` são números reais representando as medições dos sensores de farol ou a palavra "NotVisible" se o farol não é visto pelo sensor; (na edição de 2006 esta medição apenas é enviada de 2 em 2 ciclos de simulação;)
- `collisionvalue` e `groundvalue` são uma das palavras "Yes" ou "No" indicando respectivamente se houve colisão e se o robô se encontra totalmente dentro da área-alvo;
- `irs0value`, `irs1value`, `irs2value` e `irs3value` são números reais com as medições dos sensores de obstáculos números 0, 1, 2 e 3, respectivamente;
- `xvalue`, `yvalue` e `dirvalue` são números reais com as medições do GPS;
- `endledvalue` e `retledvalue` são as palavras "On" ou "Off" indicando se os *leds Fim* e *Regresso* estão acesos;
- `startvalue` e `stopvalue` são as palavras "On" ou "Off" indicando o estado do simulador, a correr (a simular) ou parado.

### Ordens de actuação

Pelo seu lado, os agentes robóticos poderão enviar ao simulador uma mensagem por ciclo de simulação, com a sua ordem de comando sobre os actuadores. Se for enviada mais que uma ordem de comando sobre o mesmo actuador no mesmo ciclo de simulação, apenas a última será considerada. Esta mensagem tem a forma:

```
<Actions LeftMotor="lpow" RightMotor="rpow"
    VisitingLed="vact" EndLed="eact" ReturningLed="ract"/>
```

em que:

- `lpow` e `rpow` são números reais representando as potências a aplicar respectivamente aos motores esquerdo e direito;
- `vact`, `eact` e `ract` são as palavras "On" ou "Off" indicando a intenção de acender ou apagar os *leds Visita*, *Fim* e *Regresso*, respectivamente.

## 7.2 Modelos de Simulação

Um sistema de simulação complexo, como é o caso do do concurso *Ciber-Rato*, envolve frequentemente a introdução de modelos simplificados da realidade que se pretende simular. Alguns destes modelos poderão ter impacto no desenvolvimento dos agentes robóticos, pelo que importa aqui mencioná-los:

### Tempo discreto

A simulação processa-se em tempo discreto e não em tempo contínuo. As posições dos robôs virtuais são modificadas, em simultâneo para todos, no início de cada ciclo de simulação, nada se passando no entretanto.

### Movimento dos robôs

O movimento de um robô depende da potência aplicada aos motores que o controlam. A transformação da potência aplicada aos dois motores em movimento do robô segue uma abordagem que divide o movimento em duas componentes, uma linear seguida de uma rotativa. Assim, em consequência das potências aplicadas, considera-se que o robô primeiro se desloca linearmente na direcção do seu eixo e depois roda em torno do seu centro. Considera-se, ainda, que a aplicação de uma potência à entrada de um motor não se manifesta imediatamente em movimento, havendo como que uma inércia dos motores.

Os deslocamentos linear e angular de um robô num dado instante de tempo  $t$  obedecem às seguintes equações:

$$\begin{aligned} lOutPow_t &= (lOutPow_{t-1} + lInPow_t) / 2 \\ rOutPow_t &= (rOutPow_{t-1} + rInPow_t) / 2 \\ lin_t &= (lOutPow_t * lNoise_t + rOutPow_t * rNoise_t) / 2 \\ rot_t &= (rOutPow_t * rNoise_t - lOutPow_t * lNoise_t) / diam \end{aligned}$$

em que:

- $lin_t$  e  $rot_t$  são os deslocamentos linear e angular do robô no instante  $t$ ;
- $lInPow_t$  e  $rInPow_t$  são as potências aplicadas no instante  $t$  aos motores esquerdo e direito;
- $diam$  é o diâmetro do robô;
- $lOutPow_{t-1}$  e  $rOutPow_{t-1}$  são os valores de  $lOutPow$  e  $rOutPow$  no instante  $t - 1$ , ou seja, no ciclo de simulação anterior;
- $lNoise_t$  e  $rNoise_t$  são os ruídos aleatoriamente calculados para os dois motores.

### Colisões

A movimentação dos robôs é processada em simultâneo para todos, de acordo com os seguintes passos:

1. São determinadas as novas posições dos robôs, de acordo com as ordens dadas e assumindo que não há obstáculos.
2. São identificados os robôs que em consequência do movimento para as novas posições provocam colisões com os obstáculos ou com outros robôs.

3. Os robôs não colidentes assumem as novas posições.
4. Os robôs colidentes apenas efectuam o movimento rotativo, sendo accionado o sensor de colisão e aplicada a penalização correspondente.

### 7.3 Biblioteca de Suporte em C

É fornecida pela *Organização* uma biblioteca de funções de apoio ao desenvolvimento de agentes robóticos na linguagem de programação C, no sistema operativo Linux. Corresponde aos ficheiros `libRobSock.a` e `RobSock.h` e inclui as seguintes primitivas:

```
int InitRobot(char *name, int pos, char *host);
```

Realiza as operações de registo do agente robótico junto do simulador. Os parâmetros `name`, `pos` e `host` representam respectivamente o nome pelo qual o agente robótico é identificado no painel de pontuações, a posição do robô na grelha de partida e o nome da máquina onde o simulador está a correr. Se o registo for feito usando esta função, o simulador coloca os sensores de obstáculos nas posições por defeito.

```
int InitRobot2(char *name, int pos, double angles[4], char *host);
```

Esta primitiva é funcionalmente equivalente à anterior. mas os sensores de obstáculos são colocados nas posições angulares definidas pelo parâmetro `angles`.

```
void DriveMotors(double lPow, double rPow);
```

Aplica potência aos motores esquerdo e direito, respectivamente.

```
unsigned int GetCycleTime(void);
```

Devolve a duração do ciclo de simulação.

```
unsigned int GetFinalTime(void);
```

Devolve o tempo máximo de ida e volta.

```
unsigned int GetTime(void);
```

Devolve o tempo actual de simulação, aquando da última invocação da função `ReadSensors`.

```
int ReadSensors(void);
```

Comunica com o simulador no sentido de obter os dados sensoriais do robô. A função espera até que chegue uma mensagem do simulador com os dados. Os dados recebidos são guardados internamente pela biblioteca, podendo ser acedidos através de outras funções.

```
double GetObstacleSensor(int pos);
```

Devolve o valor medido pelo sensor de obstáculos da posição `pos`, aquando da última invocação da função `ReadSensors`.

```
int GetNumberOfBeacons(void);
```

Devolve o número de faróis no labirinto.

```
bool IsBeaconReady(int id);
```

Indica se houve recepção da medição do sensor de farol, aquando da última invocação da função `ReadSensors`.

```
struct beaconMeasure GetBeaconSensor(int id);
```

Se a função `IsBeaconReady` devolver `true`, esta função devolve o valor medido pelo sensor de farol, aquando da última invocação da função `ReadSensors`. Caso contrário devolve informação sem significado. O tipo do valor de retorno é dado por

```
struct beaconMeasure
{
    bool beaconVisible; /* indica se o farol é visível */
    double beaconDir; /* valor medido, se visível */
};
```

```
double GetCompassSensor(void);
```

Devolve o valor medido pela bússola, aquando da última invocação da função `ReadSensors`.

```
bool GetGroundSensor(void);
```

Indica se, aquando da última invocação da função `ReadSensors`, o robô se encontrava totalmente sobre a área-alvo.

```
bool GetBumperSensor(void);
```

Indica se, aquando da última invocação da função `ReadSensors`, o robô havia colidido com obstáculos ou outros robôs.

```
bool GetStartButton(void);
```

Indica se, aquando da última invocação da função `ReadSensors`, o botão *Arranque* estava premido.

```
bool GetStopButton(void);
```

Indica se, aquando da última invocação da função `ReadSensors`, o botão *Paragem* estava premido.

```
bool GetVisitingLed(void);
```

Indica se, aquando da última invocação da função `ReadSensors`, o *led Visita* estava aceso.

```
bool GetReturningLed(void);
```

Indica se, aquando da última invocação da função `ReadSensors`, o *led Regresso* estava aceso.

```
bool GetFinished(void);
```

Indica se, aquando da última invocação da função `ReadSensors`, o *led Fim* estava aceso.

```
bool SetVisitingLed(bool val);
```

Envia uma ordem para o simulador para acender/apagar o *led Visita*.

```
bool SetReturningLed(bool val);
```

Envia uma ordem para o simulador para acender/apagar o *led Regresso*.

```
bool Finish(void);
```

Envia uma ordem para o simulador para acender o *led Fim* e aplicar uma potência nula aos dois motores.

```
double GetX(void);
```

Devolve o posicionamento do robô na área de jogo segundo o eixo dos X.

```
double GetY(void);
```

Devolve o posicionamento do robô na área de jogo segundo o eixo dos y.

```
double GetDir(void);
```

Devolve o posicionamento angular do robô relativamente ao norte virtual.

## 8 Diferenças para a Edição Anterior

Relativamente à edição de 2005 do concurso *Ciber-Rato* há várias alterações nas regras e especificações técnicas da edição deste ano. Faz-se a seguir uma enumeração das diferenças sem qualquer preocupação de ordem:

- Cada prova passa a conter várias áreas-alvos que os robôs deverão visitar (ver secção 5.3). Em consequência:
  - o corpo do robô virtual é diferente, passando a conter vários sensores de faróis (ver secção 3.1), e um *led Visita* (ver secção 3.2);
  - a área de jogo é diferente, passando a conter várias áreas-alvos (ver secção 4);
  - o processo de cálculo das pontuações é diferente (ver secção 5.4);
  - o formato XML da mensagem de resposta de aceitação é diferente (ver secção 7.1);
  - o formato XML da mensagem com os dados sensoriais é diferente (ver secção 7.1);
  - o formato XML da mensagem com as ordens de actuação é diferente (ver secção 7.1);
  - o formato XML de descrição do labirinto é diferente (ver secção 6);
  - as funções da biblioteca C para manipulação dos dados dos sensores de faróis são diferentes (ver secção 7.3).
  - Há *leds* diferentes para assinalar visita a uma área-alvo e início da viagem de regresso.
- Deixou de existir o tempo de concretização do primeiro objectivo. Em consequência:

- o ficheiro com os parâmetros de simulação perdeu o atributo `RUNNINGTIMEOUT`. (Ver secção 7.1.)