



Sistemas Autónomos

Trabalho prático 1

Grupo 1

11 de abril de 2021



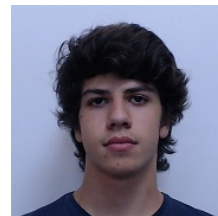
Ana Gil
A85266



Beatriz Rocha
A84003



Hugo Matias
A85370



João Abreu
A84802

Conteúdo

1	Introdução	4
2	Primeira etapa	5
2.1	Odómetro	5
2.2	Circum-navegação	6
2.2.1	Deslocação para a posição inicial	6
2.2.2	Estratégia de navegação	7
3	Segunda etapa	9
3.1	Shepherd	9
3.2	Sheep	9
4	Terceira etapa	11
4.1	Boss	11
4.2	Saviour	12
4.3	Gladiator	13
4.3.1	Movimento	13
4.3.2	Ação	15
4.4	Avenger	15
4.4.1	<i>Hunting mode</i>	15
4.4.2	<i>Avenging mode</i>	17
5	Conclusão e Análise de Resultados	19

Lista de Figuras

2.1	Cálculo do ângulo complementar	6
2.2	Cálculo do raio da circunferência	7
2.3	Estratégia de navegação	8
3.1	Ordens fornecidas pelo robô Shepherd aos robôs Sheep	9
3.2	Cálculo do ângulo que a arma tem de rodar (gun_angle) para atingir o inimigo	10
4.1	Diagrama de estados do robô Saviour	13
4.2	Movimento que o robô Gladiator começa por adotar	14
4.3	Movimento que o robô Gladiator adota quando os robôs Boss e Saviour morrem	14
4.4	Comportamento padrão do robô Avenger	16
4.5	O robô Avenger perde o inimigo de vista	16
4.6	O robô Avenger identifica um novo inimigo	17
4.7	O robô Boss envia uma mensagem ao robô Avenger com o nome do inimigo que o atacou	18
4.8	O robô Avenger ataca o inimigo cujo nome estava na mensagem do robô Boss	18

Lista de Tabelas

2.1	Distância medida por cada um dos odômetros em cada ronda . .	5
2.2	Distância percorrida e perímetro formado pelos três obstáculos em cada ronda	8
5.1	Resultados da competição entre a nossa equipa e a equipa prede- finida no Robocode após 10 batalhas	19

1 Introdução

Sistemas autônomos são definidos como sistemas capazes de completar uma tarefa, atingir um determinado objetivo ou interagir com o ambiente que os rodeia com o menor envolvimento humano possível. Para isso, é essencial que estes sistemas sejam capazes de prever, planejar e absorver o mundo à sua volta.

Com o intuito de nos familiarizar com este subdomínio da inteligência artificial e recorrendo, para isso, ao jogo Robocode, este trabalho prático passou por três etapas:

1. Conceção de um odómetro capaz de medir a distância percorrida e implementação de técnicas de planeamento de trajetórias para circum-navegar por fora a área demarcada pelos três obstáculos no sentido dos ponteiros do relógio na menor distância possível, de modo a partir da posição (18,18) e regressar à mesma;
2. Formação de equipas de robôs capazes de apresentar diversos comportamentos de grupo ou sociais, permitindo a discussão das questões abordadas no contexto da unidade curricular de Agentes Inteligentes, como a cooperação entre agentes para a resolução de problemas;
3. Conceção, formação e avaliação do desempenho de equipas de cinco robôs (iguais ou distintos), cujo objetivo é o de ganhar batalhas e, consequentemente, competições.

Deste modo, nos próximos capítulos, iremos explicar detalhadamente as estratégias desenvolvidas e implementadas por nós para conseguir responder adequadamente a cada uma das tarefas propostas.

2 Primeira etapa

2.1 Odômetro

Na primeira etapa deste projeto, foi-nos pedido que desenvolvêssemos e implementássemos um odômetro capaz de medir a distância percorrida por um robô em cada ronda de uma batalha. Deste modo, a nossa implementação passou por calcular a distância euclidiana (2.1) entre a posição anterior e a posição atual do robô em cada instante (*turn*) e acumular esse valor numa variável que armazena a distância total percorrida.

$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \quad (2.1)$$

Para termos acesso à posição atual do robô em cada *turn*, tivemos de recorrer ao método `onStatus(StatusEvent event)` que, por sua vez, é chamado em cada instante, de modo a devolver o estado atual do robô (tal como as suas coordenadas, a temperatura da sua arma, *etc.*).

```
1 public void onStatus(StatusEvent event) {  
2     this.myOdometer.calculateDistanceTravelled();  
3 }
```

Para além disso, é importante referir que, para o odômetro saber quando deve começar a medir a distância percorrida e quando deve parar, criámos os métodos `start_race()` e `stop_race()` que indicam que a corrida começou e acabou, respetivamente.

De modo a testar a eficácia do nosso odômetro, decidimos observar os resultados obtidos pelo mesmo e pelo odômetro que nos foi fornecido pela equipa docente em 10 rondas (Tabela 2.1) e calcular a média entre os mesmos. Feito isto, obtivemos uma eficácia de cerca de 99.9%, como já era de esperar, visto que os valores são muito semelhantes.

Ronda	MyOdometer	Odometer
1	1776.38	1772.35
2	1801.67	1799.5
3	1831.36	1830.07
4	1811.21	1807.81
5	1747.53	1744.4
6	1875.52	1871.9
7	1838.29	1834.99
8	1901.84	1900.77
9	1824.03	1821.45
10	1735.97	1734.71

Tabela 2.1: Distância medida por cada um dos odômetros em cada ronda

2.2 Circum-navegação

2.2.1 Deslocação para a posição inicial

Visto que, no início de cada ronda, o robô está localizado numa posição aleatória, é necessário começar por deslocá-lo para o canto inferior esquerdo (coordenada (18,18)), de modo a cumprir os requisitos desta fase. Assim sendo, a tática a aplicar apresenta-se de seguida:

1. Cálculo da distância entre a posição atual do robô e a posição para a qual ele se deseja deslocar (posição (18,18), neste caso);
2. Cálculo do ângulo complementar (Figura 2.1);
3. Rotação de x graus para a esquerda, de modo a ficar a apontar para a posição desejada, sendo que x é a soma do ângulo complementar calculado anteriormente com a direção do robô;
4. Deslocação da distância calculada no primeiro passo, chegando, assim, à posição desejada.

Adicionalmente, é importante referir que, se pelo caminho o robô encontrar algum obstáculo, este recua ligeiramente, gira 45° à direita e avança novamente, de maneira a tentar contornar o mesmo.

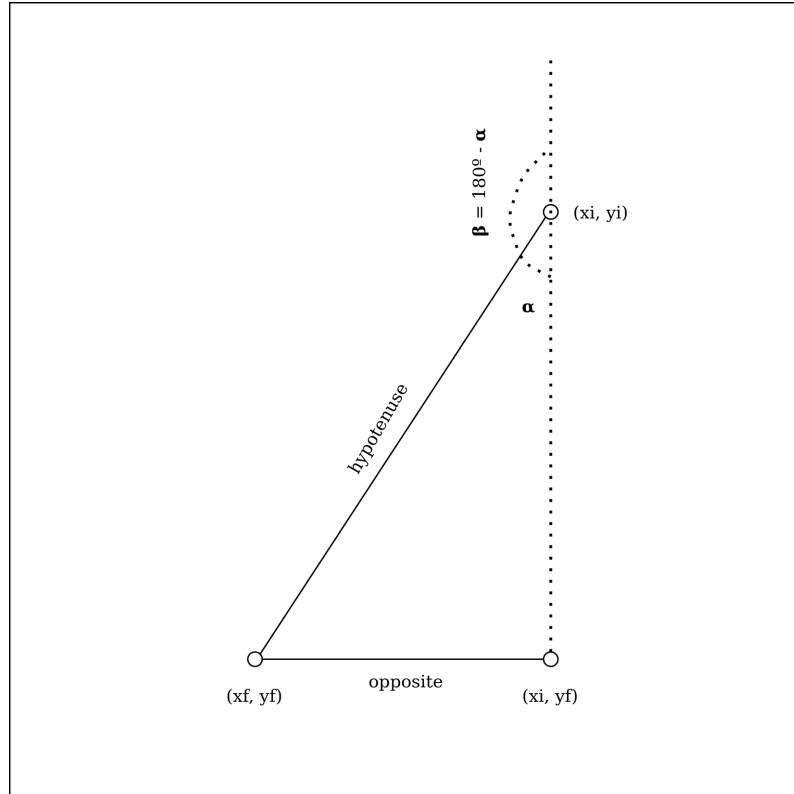


Figura 2.1: Cálculo do ângulo complementar

2.2.2 Estratégia de navegação

Feita a deslocação do robô para a posição (18,18), há que fazê-lo contornar os três obstáculos sem colidir neles. No nosso caso, optámos por descrever uma trajetória circular à volta dos obstáculos e, portanto, primeiro, torna-se importante calcular qual será o raio dessas circunferências.

O raio mínimo que as circunferências podem tomar de modo a que o robô não colida com os obstáculos corresponde à soma de $d1$ com $d2$ (mais um coeficiente de segurança), tal como podemos ver na Figura 2.2. Visto que a dimensão de um robô é 36x36 píxeis, $d1$ será igual a $18\sqrt{2}$ píxeis e $d2$ será igual a 18 píxeis, o que resultará em, aproximadamente 43.46 píxeis. Acrescentando o tal coeficiente de segurança, obtemos um raio igual a 50 píxeis.

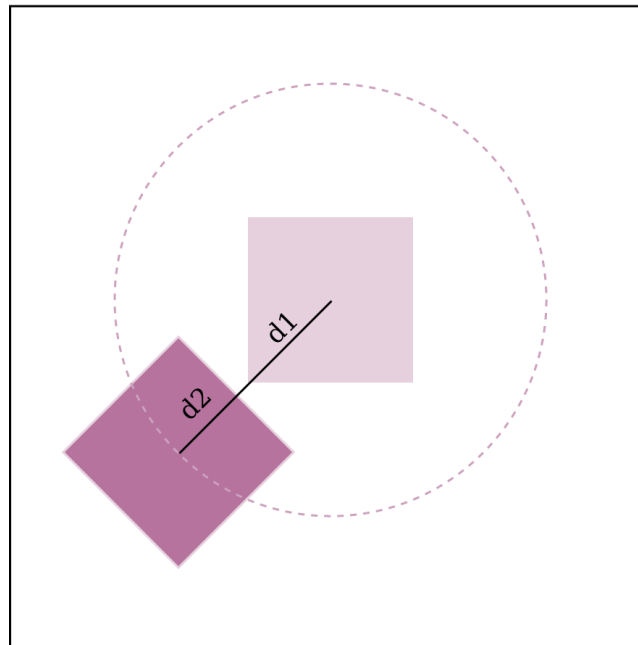


Figura 2.2: Cálculo do raio da circunferência

Posto isto, resta-nos enumerar os passos da estratégia implementada:

1. Esperar durante um curto período de tempo, de modo a evitar que o robô comece a navegar antes de os obstáculos estarem posicionados nos seus respetivos lugares;
2. Girar o robô em direção a norte;
3. Enquanto o número de obstáculos detetados for inferior a 3:
 - (a) Rodar o radar do robô para a direita, de forma a detetar o obstáculo;
 - (b) Rodar o robô de forma a ficar alinhado com o obstáculo detetado;
 - (c) Deslocar o robô até 50 píxeis (raio da circunferência) antes da posição do obstáculo;

- (d) Contornar o obstáculo;
4. Regressar à posição inicial.

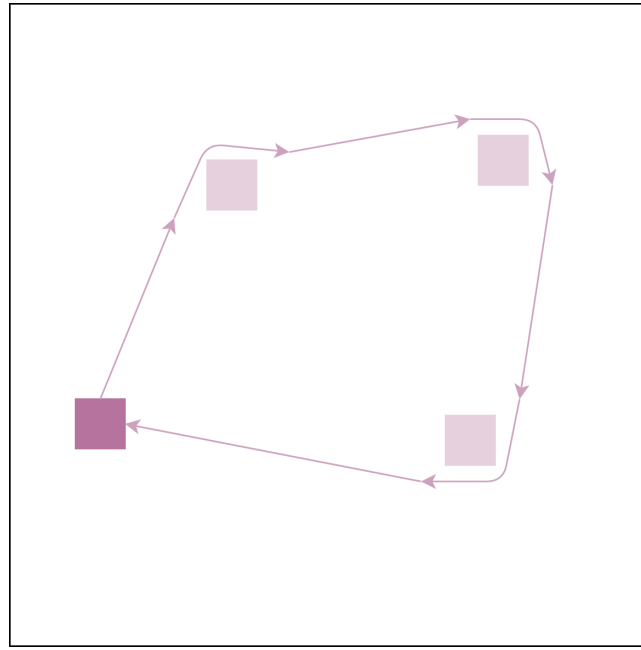


Figura 2.3: Estratégia de navegação

De modo a testar a eficácia da nossa estratégia, decidimos observar a distância percorrida e o perímetro formado pelos três obstáculos em 10 rondas (Tabela 2.2) e calcular a média entre esses valores. Feito isto, obtivemos uma eficácia de cerca de 87%, o que demonstra que o nosso robô está a executar uma trajetória bastante próxima da área demarcada pelos três obstáculos.

Ronda	Perímetro	Distância percorrida
1	1499.46	1736.37
2	1666.19	1912.45
3	1511.2	1746.23
4	1622.77	1868.22
5	1519.86	1757.49
6	1619.85	1866.46
7	1580.4	1821.24
8	1532.32	1770.78
9	1579.47	1818.91
10	1559.12	1800.89

Tabela 2.2: Distância percorrida e perímetro formado pelos três obstáculos em cada ronda

3 Segunda etapa

Nesta fase, foi-nos pedido que criássemos equipas de robôs capazes de apresentar diversos comportamentos de grupo ou sociais, portanto (e já a pensar na fase seguinte) optámos por criar uma equipa formada por quatro robôs **Sheep** do tipo **Droid** (robôs que não possuem radar, mas, em contrapartida, possuem 20 unidades de energia extra) e um robô **Shepherd** que lidera esses quatro robôs (este, por sua vez, possui 100 unidades de energia extra).

3.1 Shepherd

Inicialmente, este robô começa por dirigir-se para o canto inferior esquerdo do campo de batalha (coordenada (18,18)), de modo a estar mais protegido de possíveis ataques de inimigos. Mais uma vez, caso colida com um obstáculo pelo caminho, este irá recuar ligeiramente, rodar 45° à esquerda e avançar novamente. Contudo, caso já tenha tentado cinco vezes sem sucesso, tenta virar à direita em vez de virar à esquerda.

De seguida, irá coordenar os restantes membros da sua equipa, segundo quatro ordens distintas (enviadas sob o formato de mensagem), tal como se pode observar na Figura 3.1.

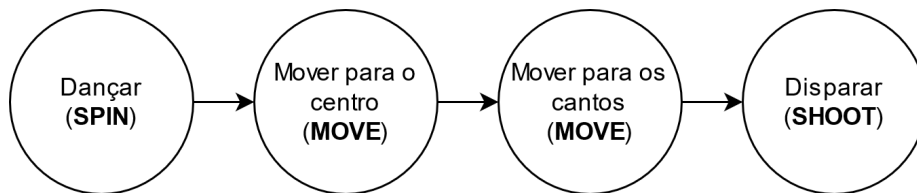


Figura 3.1: Ordens fornecidas pelo robô **Shepherd** aos robôs **Sheep**

3.2 Sheep

Como o próprio nome indica, os robôs **Sheep** limitam-se a cumprir as ordens fornecidas pelo robô **Shepherd**. Assim sendo, para cada tipo de ordem recebida, irá ser executado um comportamento diferente, nomeadamente:

1. **Dançar (mensagem do tipo SPIN)** - Girar duas vezes sobre si mesmos (720°), de modo a garantir que as mensagens estão a ser recebidas;
2. **Mover para o centro (mensagem do tipo MOVE)** - Mover para o centro do campo de batalha (neste caso, para as coordenadas (400,300));
3. **Mover para os cantos (mensagem do tipo MOVE)** - Mover para cada um dos quatro cantos do campo de batalha, de modo a evitar *friendly fire* (fogo amigo) na hora de disparar sobre os inimigos, sendo que o robô que

for para o canto onde já se encontra o **Shepherd**, ficará umas unidades à frente deste, de modo a protegê-lo;

4. **Disparar (mensagem do tipo SHOOT)** - No caso de o **Shepherd** detetar um robô inimigo com recurso ao seu radar (que, entretanto, se ligou e passou a rodar 360° sobre 360°), disparar contra o mesmo.

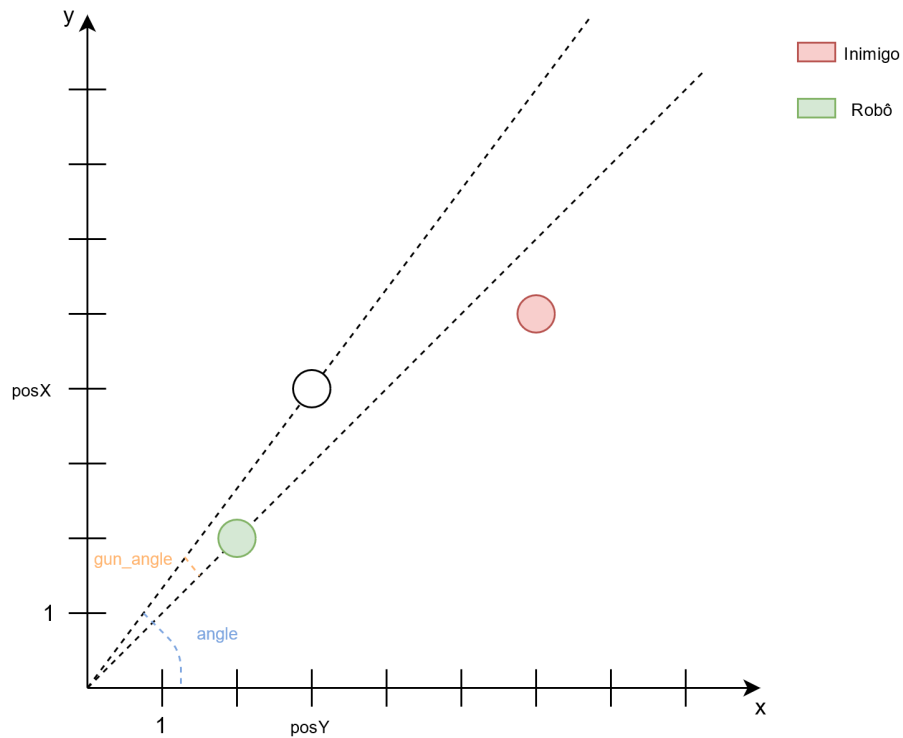


Figura 3.2: Cálculo do ângulo que a arma tem de rodar (**gun_angle**) para atingir o inimigo

Novamente, se o robô colidir com algum obstáculo durante a sua deslocação para uma determinada posição (e enquanto se encontrar a mais de 50 unidades da mesma), vai recuar ligeiramente, rodar 45° à direita e avançar novamente.

4 Terceira etapa

Nesta fase, foi-nos pedido que criássemos equipas de cinco robôs (iguais ou distintos) capazes de ganhar batalhas e, conseqüentemente, competições. Assim, optámos por criar uma equipa formada por três robôs **Gladiator** do tipo **Droid**, um robô **Boss** que lidera esses três robôs e um robô **Saviour** que dá suporte ao **Boss**, na medida em que o protege e assume o seu lugar quando ele morre.

4.1 Boss

Este robô é a peça essencial da nossa equipa. Tal como já foi referido anteriormente, a sua principal função é comandar os robôs do tipo **Droid**. No entanto, antes de chegar a esta fase, tem de passar por uma etapa de preparação, que explicamos de seguida:

1. Informa os seus colegas de equipa acerca da sua posição atual;
2. Pede ao robô **Saviour** para se movimentar ao seu lado;
3. Começa a girar o radar, de modo a procurar por inimigos.

Depois de efetuar estes 3 passos, o robô **Boss** começará a desempenhar o seu papel. No entanto, este tomará dois tipos de comportamento diferentes, dependendo do estado do jogo:

Padrão	Ataque (todos os robôs Gladiator morreram)
<ul style="list-style-type: none">• Permanece no mesmo sítio;• Não ataca;• Gira o radar continuamente;• Calcula o melhor alvo (rival que está mais próximo dos robôs Gladiator) e manda os últimos atacá-lo;• Quando o alvo morre, pede as posições atuais dos robôs Gladiator para calcular um novo;• Quando leva um tiro, desvia-se um pouco e chama pelo robô Saviour.	<ul style="list-style-type: none">• Permanece no mesmo sítio;• Ataca o rival mais próximo dele;• Quando leva um tiro, desvia-se perpendicularmente ao sentido da bala e chama pelo robô Saviour.

O seu comportamento padrão está presente na coluna da esquerda, sendo que, quando toma o mesmo, fica parado, com o seu radar a girar continuamente, não atacando ninguém, para poupar ao máximo a sua energia. Quando encontra

todos os inimigos, guarda as suas posições e calcula o melhor alvo para os robôs **Gladiator** atacarem. Optámos por escolher o robô que esteja mais próximo dos mesmos, para eliminarmos os inimigos o mais cedo possível. Depois, ordena os mesmos robôs atacarem esse alvo e, quando este é eliminado, recalculamos um novo. Ao permanecer no mesmo sítio, o robô **Boss** torna-se bastante vulnerável e, por esse motivo, implementámos o último ponto, ou seja, quando é atingido, desvia-se ligeiramente no sentido perpendicular da bala, para evitar levar com o próximo tiro. É nesta situação que ele também volta a chamar o robô **Saviour** para junto dele.

O outro tipo de comportamento que o robô **Boss** pode assumir encontra-se na coluna da direita. Este passará a agir desta maneira, quando todos os robôs **Gladiator** morrerem, pois não fará mais sentido ficar à procura de inimigos para esses robôs atacarem. Sendo assim, neste estado, ele próprio passará a ter o papel de atacar. Apesar disso, permanecerá no mesmo sítio, continuando a desviar-se, caso seja atingido.

4.2 Saviour

Este robô depende diretamente do robô apresentado na secção 4.1 e tem como principais objetivos protegê-lo e assumir a liderança, caso este seja destruído:

- **Proteger o robô Boss**

Este é o tipo de comportamento que o robô **Saviour** começa por adotar, quando a batalha começa. Para isso, este começa por se posicionar junto ao robô **Boss**, permanecendo ao seu lado durante o resto da batalha até que um deles seja destruído. Assim sendo, sempre que o robô **Boss** se movimenta, o robô **Saviour** vai segui-lo. Para que esta ação aconteça, o robô **Saviour** recebe uma mensagem do robô **Boss** contendo as suas novas coordenadas. Uma vez que o robô **Boss** não ataca os inimigos (pelo menos, numa fase inicial), o robô **Saviour** (que se encontra ao seu lado) adota um comportamento inspirado no robô **TrackFire** que, por sua vez, ataca o inimigo mais próximo dele e, neste caso, também mais próximo do robô **Boss**, não se movimentando ao longo do ataque.

- **Assumir a liderança**

No cenário em que o robô **Boss** morre primeiro que o robô **Saviour**, o último assume a liderança. Para facilitar esta implementação, a classe **Saviour** vai ser uma subclasse da **Boss**. Posto isto, o robô **Saviour** passa a ter exatamente o mesmo comportamento descrito na secção 4.1.

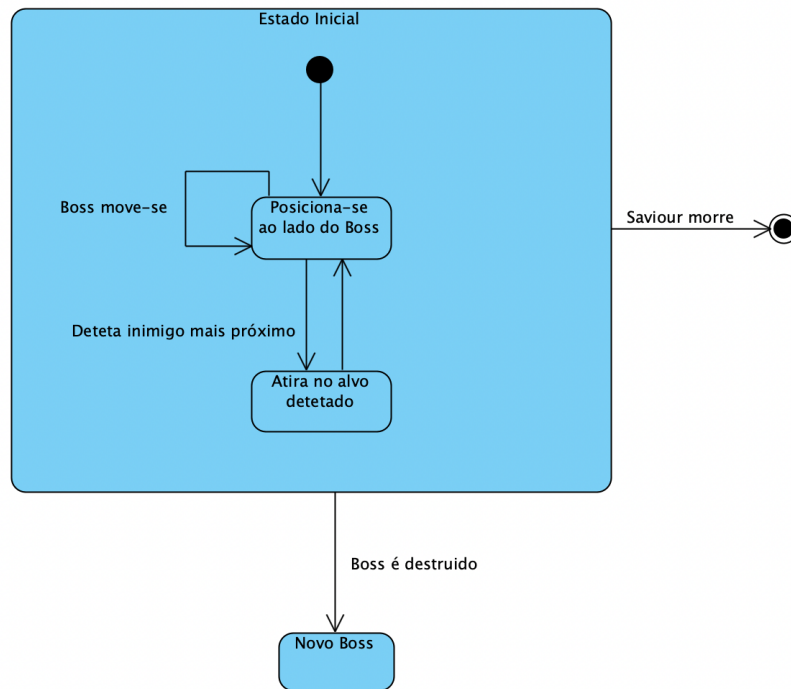


Figura 4.1: Diagrama de estados do robô Saviour

4.3 Gladiator

4.3.1 Movimento

Este robô apresenta dois diferentes tipos de movimento dependendo das circunstâncias.

Inicialmente, começa por ter um movimento circular no sentido dos ponteiros do relógio (Figura 4.2), tal como o robô **SpinBot**. Decidimos adotar este comportamento para o nosso robô, uma vez que, após analisar com atenção todos os robôs predefinidos no Robocode, apercebemo-nos que o **SpinBot** ganha várias batalhas graças ao seu movimento imprevisível, visto que dificulta a tarefa de os rivais lhe atingirem.

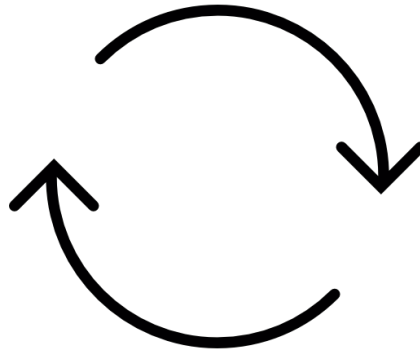


Figura 4.2: Movimento que o robô **Gladiator** começa por adotar

Quando os robôs **Boss** e **Saviour** morrem, este robô deixa de ter quem o auxilie no que toca ao radar e, portanto, adota outro tipo de movimento. Nesta situação (e à semelhança do movimento do robô **Walls**), irá começar a circular à volta das paredes do campo de batalha (Figura 4.3) sem disparar, de modo a poupar energia e tentar esgotar o tempo da batalha. Este movimento poderá constituir uma ameaça para os robôs rivais, uma vez que estes perdem bastante energia a disparar contra o robô **Gladiator** sem quase nunca lhe acertar, devido ao facto de este estar em constante movimento na zona mais resguardada do campo de batalha.

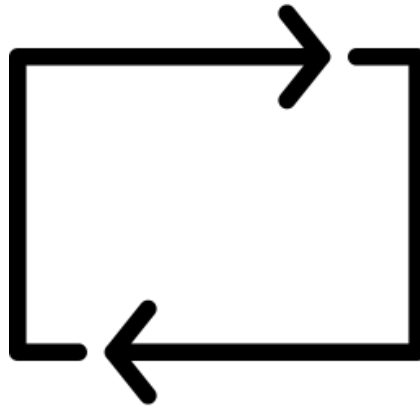


Figura 4.3: Movimento que o robô **Gladiator** adota quando os robôs **Boss** e **Saviour** morrem

Caso o robô **Gladiator** colida com algum obstáculo, irá executar três comportamentos distintos dependendo da situação:

- Caso ainda não esteja a circular à volta das paredes do campo de batalha e o obstáculo com o qual colidiu seja um colega de equipa, vira-se para a sua direcção no sentido oposto e avança ligeiramente;

- Caso ainda não esteja a circular à volta das paredes do campo de batalha, o obstáculo com o qual colidiu seja um rival e:
 - O ângulo a rodar para ficar na mesma direção que o último seja muito reduzido (entre -10° e 10°), dispara;
 - Tenha sido culpa dele, gira 10° à direita para continuar a rodar;
- Caso já esteja a circular à volta das paredes do campo de batalha e:
 - O obstáculo esteja à sua frente, recua ligeiramente;
 - O obstáculo esteja atrás dele, avança ligeiramente.

4.3.2 Ação

O robô **Gladiator** executa três ações distintas, dependendo do tipo de mensagem que recebe:

- **Informa acerca da sua posição (mensagem do tipo REQUEST)** - Comunica aos seus colegas de equipa as suas coordenadas atuais;
- **Ataca (mensagem do tipo ATTACK)** - Dispara contra o rival que o robô **Boss** lhe ordenou atingir (recorrendo, mais uma vez, à estratégia explicada na Figura 3.2 para calcular o ângulo que a arma tem de rodar);
- **Atualiza o *HashMap* dos colegas de equipa (mensagem do tipo INFO)** - Coloca o remetente da mensagem no *HashMap* dos colegas de equipa.

4.4 Avenger

Este robô caracteriza-se pela sua excecional agressividade na procura e destruição dos robôs inimigos. O robô **Avenger** apresenta dois tipos de comportamento: *hunting mode* e *avenging mode*. A sua movimentação depende destes dois modos, ou seja, depende sempre dos inimigos que ele localiza e decide atacar ou das ordens do robô **Boss**.

Contudo, apesar do seu comportamento complexo e, à primeira vista, útil em relação ao objetivo para o qual foi designado (eliminar os inimigos diretos do robô **Boss**), a equipa atingia uma melhor eficácia em batalha, quando o robô **Avenger** era trocado por mais um **Gladiator**. Os seus movimentos previsíveis, o tempo que demorava a chegar perto do inimigo e a sua inabilidade para reagir a robôs que o atacavam faziam com que o robô **Avenger** fosse desnecessário e ineficaz, portanto tivemos de optar por descartá-lo.

4.4.1 *Hunting mode*

Este é o comportamento padrão do robô **Avenger**. Neste modo, o robô **Avenger** funciona como um robô **Tracker**, ou seja, começa por rodar o radar para identificar um inimigo e, de seguida, persegue e ataca esse inimigo. O robô **Avenger** para de perseguir esse inimigo, caso o mate ou o perca de vista por um certo

período de tempo. Tanto num caso como no outro, o robô **Avenger** volta a usar o radar para identificar um novo alvo.

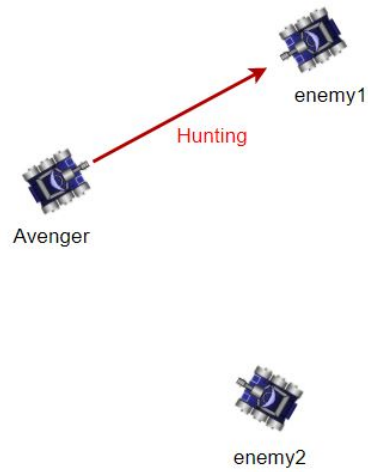


Figura 4.4: Comportamento padrão do robô **Avenger**

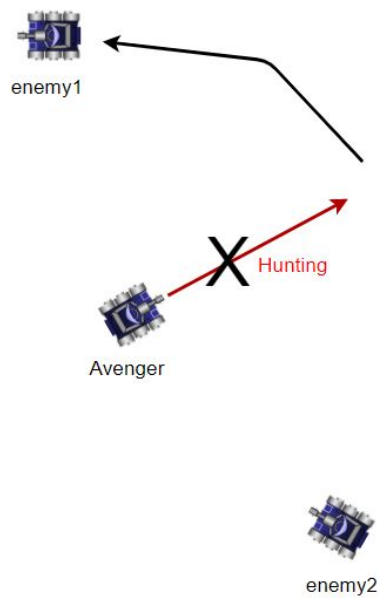


Figura 4.5: O robô **Avenger** perde o inimigo de vista

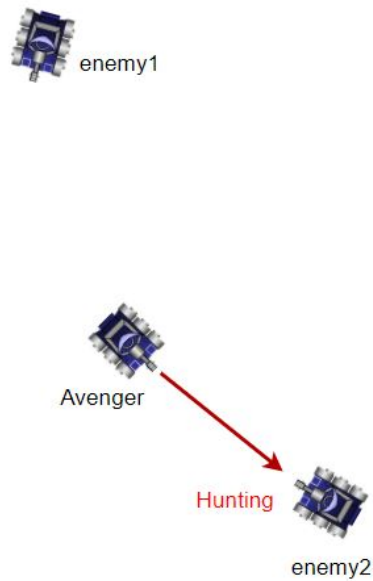


Figura 4.6: O robô **Avenger** identifica um novo inimigo

4.4.2 *Avenging mode*

Este é o comportamento que o robô **Avenger** adota, quando o robô **Boss** é atacado. Quando o robô **Boss** é atacado por um inimigo, este envia uma mensagem ao robô **Avenger** com o nome do inimigo que o atacou. O robô **Avenger** começa a perseguir este inimigo, ignorando qualquer perseguição que estivesse a fazer anteriormente, ou seja, o inimigo que vem na mensagem do robô **Boss** tem prioridade e, portanto, passa a ser o seu alvo. Ao contrário do que acontece quando o robô **Avenger** está em *hunting mode*, neste modo, este robô é incrivelmente persistente, pois, mesmo que perca de vista o inimigo, continua a procurá-lo indefinidamente. Apenas há duas situações onde o robô **Avenger** para de perseguir o inimigo: se tiver sucesso na destruição do inimigo em questão ou se morrer. É de notar que nem mesmo uma nova mensagem do robô **Boss** com um novo inimigo a abater retira o efeito que este modo tem no robô **Avenger**. Caso isto aconteça, o último ignora esta ordem até cumprir a atual.

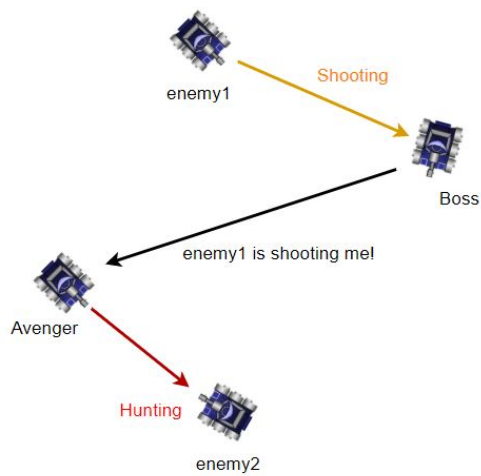


Figura 4.7: O robô **Boss** envia uma mensagem ao robô **Avenger** com o nome do inimigo que o atacou

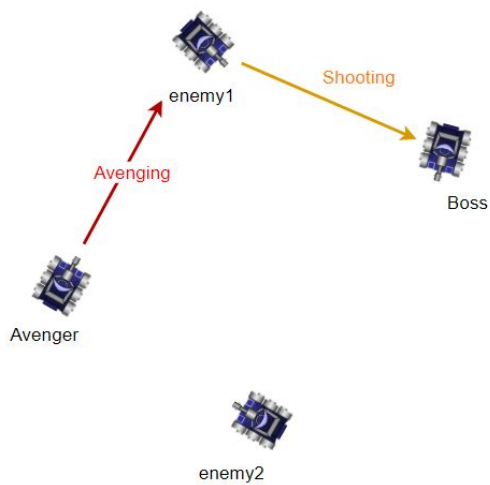


Figura 4.8: O robô **Avenger** ataca o inimigo cujo nome estava na mensagem do robô **Boss**

5 Conclusão e Análise de Resultados

A realização deste projeto, em particular com recurso à programação de robôs, permitiu-nos ambientar ao planeamento de trajetórias e a estratégias e arquiteturas de controlo.

Na primeira etapa, procedemos ao desenvolvimento de um odómetro e à circum-navegação de três obstáculos estáticos, com o objetivo de percorrer a menor distancia possível. Nesta fase, tivemos especial dificuldade em contornar os obstáculos sem colidir com eles. Porém, após várias tentativas, acabámos por optar percorrer uma distância ligeiramente maior com o benefício de nunca atingir os mesmos.

Na segunda etapa, formámos uma equipa capaz de demonstrar vários comportamentos de grupo, o que nos possibilitou a familiarização com a troca de mensagens entre os robôs e todos os métodos inerentes à classe `TeamRobot`. Esta fase permitiu-nos, ainda, aplicar os conhecimentos adquiridos na unidade curricular de Agentes Inteligentes acerca da cooperação entre agentes.

Por último, na terceira etapa, concebemos uma equipa de cinco robôs (um robô `Boss`, um robô `Saviour` e três robôs `Gladiator`), capazes de ganhar batalhas e, consequentemente, competições. Apesar de alguns problemas, nomeadamente no que toca a *friendly fire*, a nossa equipa apresenta um excelente desempenho quando compete com a equipa predefinida no Robocode (por exemplo), tal como podemos ver na Tabela 5.1, o que demonstra que todos os requisitos foram cumpridos com sucesso.

Rank	1st	2nd
Robot Name	sa.fase3.BossTeam (1)	sampleteam.MyFirstTeam (2)
Total Score	16629 (71%)	6634 (29%)
Survival	9700	2800
Surv Bonus	1350	100
Bullet Dmg	4697	3444
Bullet Bonus	831	282
Ram Dmg * 2	52	8
Ram Bonus	0	0
1sts	8	2
2nds	2	8
3rds	0	0

Tabela 5.1: Resultados da competição entre a nossa equipa e a equipa predefinida no Robocode após 10 batalhas