

Universidade Tecnológica Federal do  
Paraná  
Campus Apucarana  
Bacharelado em Engenharia da Computação

**Relatório Atividade Prática 1 -  
Robocode**

Alunos:

Ana Carolina Ribeiro Miranda

Eiti Parruca Adama

Lucas Eduardo Pires Parra

Professor:

Dr. Rafael Gomes Mantovani

Setembro  
2023

# Universidade Tecnológica Federal do Paraná

Campus Apucarana  
Bacharelado em Engenharia da Computação

## **Relatório Atividade Prática 1 - Robocode**

Primeiro Relatório da Atividade Prática apresentado na disciplina de Sistemas Inteligentes 1 do Curso de Engenharia de Computação da Universidade Tecnológica Federal do Paraná, campus Apucarana como requisito parcial para aprovação da disciplina.

Aluno:

Ana Carolina Ribeiro Miranda

Eiti Parruca Adama

Lucas Eduardo Pires Parra

Professor:

Dr. Rafael Gomes Mantovani

Setembro  
2023

# Conteúdo

1	Resumo	1
2	Apresentação	2
3	Descrição de atividades	3
4	Análise dos Resultados	9
5	Trabalhos Futuros	14
6	Referências	15

# 1 Resumo

O presente trabalho representa a primeira atividade prática da disciplina, e tem como seu propósito a concepção de um agente inteligente capaz de participar de competições virtuais, nas quais robôs modelados como tanques de guerra interagem em uma arena virtual. Para tal empreendimento, utiliza-se a plataforma Robocode, implementada na linguagem de programação Java, com uma clara ênfase na Programação Orientada a Objetos.

O objetivo principal deste projeto consiste na criação de um robô que, munido das ferramentas e recursos disponibilizados pela plataforma, juntamente da comunidade, possa destacar-se no cenário competitivo, triunfando sobre seus adversários e alcançando a posição de líder no “*ranking*” de competidores.

O presente relatório engloba a estratégia que adotamos para o desenvolvimento do robô denominado “*Miauzinho*”. Além disso, serão abordados os desafios que surgiram durante o processo de concepção e implementação, bem como os resultados que conseguimos atingir, trazendo à tona as contribuições e implicações deste trabalho para a disciplina e o campo de estudo em questão.

## 2 Apresentação

O presente trabalho representa um marco significativo para o aprendizado e aplicação prática da inteligência artificial e da programação orientada a objetos. Esta atividade é a nossa primeira incursão na disciplina de sistemas inteligentes 1, onde nos dedicamos à criação de um agente inteligente destinado a competir em um ambiente virtual, caracterizado por batalhas épicas entre robôs modelados como tanques de guerra.

Empregando a plataforma Robocode e fazendo uso da linguagem de programação *Java*, nossa missão foi desenvolver um robô capaz de superar todos os adversários e conquistar o cobiçado primeiro lugar no topo do *ranking*. Para atingir esse objetivo, exploramos as amplas possibilidades oferecidas por essa plataforma, aprimorando nossa compreensão da programação orientada a objetos e expandindo nosso repertório de habilidades em inteligência artificial.

Nosso objetivo primordial é desenvolver um robô que, enquanto segue a ideia do “*Crazy*”, que tem como principal característica sua movimentação aleatória, visando dificultar os oponentes de atingi-lo, acrescentar um componente de precisão que se ajusta conforme a distância entre os adversários.

### 3 Descrição de atividades

O robô “*Miauzinho*” é uma expressão do nosso esforço para criar um agente inteligente capaz de competir com eficácia em batalhas virtuais no ambiente Robocode. Inspirado na estratégia do “*Crazy*”, o “*Miauzinho*” se destaca por sua capacidade de realizar ataques precisos com base na distância do adversário. A seguir, detalhamos as principais características e estratégias implementadas no “*Miauzinho*”:

- **Mudança Aleatória de Cores:** Uma característica distintiva do “*Miauzinho*” é sua capacidade de mudar de cores de forma aleatória para surpreender seus adversários. Isso não apenas adiciona um elemento de diversidade visual ao robô, mas também reflete a adaptabilidade do “*Miauzinho*” em suas táticas.
- **Movimento Aleatório:** O movimento aleatório do “*Miauzinho*” é uma característica central que o ajuda a se adaptar tanto às batalhas X1 quanto às batalhas “*Battle Royale*”. Ele calcula a direção do objetivo com base na posição do inimigo e acrescenta uma pequena variação aleatória a essa direção. Essa abordagem permite que o robô mantenha sua capacidade de esquiva e adaptação em ambas as situações de combate.

De forma mais específica, no código usa variáveis como a variável “*distancia*” para manter o controle da distância entre ele e seu alvo, a “*contDirection*” e “*moveTime*” que são usadas para controlar o movimento do robô, alternando sua direção de movimento, o “*moveDirection*” que define a direção atual de movimento, sendo para frente ou trás, a “*velocidadeBalinha*” usada para calcular o tempo de movimento com base na potência do tiro. Também é utilizado a variável “*desejadaDirection*” do robô que determina para onde ele se move no campo de batalha. Esta direção é calculada com base em informações sobre a posição do alvo (inimigo) e a orientação atual do robô, seguindo o cálculo de subtrair um ângulo de desvio da direção do alvo para determinar a direção desejada, em seguida adiciona uma variação aleatória à direção desejada, isso introduz uma imprevisibilidade no movimento do robô, tornando-o menos previsível para os adversários, essa aleatoriedade produz um valor entre -0.5 a 0.5 enquanto gera outro valor entre 1 e 3 que multiplica essa variação.

- **Esquiva de Paredes:** Para evitar colisões com as paredes da arena, o “*Miauzinho*” utiliza um algoritmo que calcula a trajetória do robô e faz ajustes à medida que se aproxima das bordas. Isso garante que o robô possa contornar as paredes de forma suave, além de eficaz em qualquer

tipo de batalha, garantindo que o robô possa contornar obstáculos e paredes sem problemas, ou seja, ele verifica se o próximo movimento fará o robô colidir com a parede e, caso positivo, ajusta a direção, isso é feito dentro de um “*loop while*” que continua até que o robô esteja em uma posição segura.

No código é usado a variável “*randomDistParede*” para determinar a distância aleatória da parede para adicionar variação ao movimento, ela varia entre 120 e 160 “*pixels*” para adicionar variação ao movimento e é novamente ajustada na função “*run*” para adicionar mais variações à distância da parede.

Também é capaz de contornar paredes evitando colisões, o processo de contorno de paredes é detalhado a seguir:

```
// Para contornar a parede sem bater
double x = getX();
double y = getY();
double smooth = 0;

// Cria um retângulo e verifica se o ponto calculado
//com base na direção desejada do robô está dentro
//desse retângulo.
Rectangle2D fieldRect = new Rectangle2D.Double(18, 18,
getBattleFieldWidth()-36, getBattleFieldHeight()-36);

while (!fieldRect.contains(x+Math.sin(desejadaDirection)*
randomDistParede, y+ Math.cos(desejadaDirection)*
randomDistParede)) {
// Vira pro inimigo e tenta dnovo
    desejadaDirection += moveDirection*0.1;
    smooth += 0.1;
}

// Se for mt, inverte a direção
if(smooth > 0.5 + Math.random()*0.125) {
    moveDirection = -moveDirection;
    contDirection = 0;
}
```

Nesse trecho em questão, encontra-se o controle das coordenadas X e Y da sua posição atual, em seguida é criado um retângulo “*fieldRect*” que representa a área de jogo, com uma margem de 18 “*pixels*” a partir das bordas do campo, assim o robô entra em um “*loop*” enquanto verifica se o ponto calculado com

base na “*desejadaDirection*” está dentro do retângulo “*fielReact*”, se o ponto estiver fora do retângulo, o robô ajusta sua “*desejadaDirection*” girando na direção do inimigo, e incrementa a variável “*smooth*”, se o valor for maior que 0.5 acrescido de uma pequena variação aleatória, o “*Miauzinho*” inverterá sua direção de movimento e irá redefinir o “*contDirection*” como zero, uma forma de ter controle para caso indique que provavelmente está preso em uma situação perigosa ou beirando uma parede.

- **Mira Precisa com Base na Distância:** Uma característica fundamental do “*Miauzinho*” é sua capacidade de mirar com precisão, levando em consideração a distância do inimigo, sendo bem aplicável tanto nas batalhas X1 quanto nas batalhas “*Battle Royale*”. Ele ajusta o poder de fogo da bala com base na distância, garantindo que tiros mais fortes sejam usados à queima-roupa e tiros mais fracos a distâncias maiores, otimizando sua capacidade de ataque em ambas as situações.

No código temos as variáveis “*balaPower*” e “*balaSpeed*”, o poder do tiro varia aleatoriamente entre 1 e 3, tornando os tiros menos previsíveis, já a velocidade da bala é calculada inversamente proporcional ao poder do tiro, ou seja, quanto maior o poder do tiro, menor a velocidade da bala. Para o cálculo da precisão da bala, utiliza-se “*velLatIni*” para calcular o componente da velocidade lateral do inimigo e o “*anguloFuga*” representa o ângulo de inclinação da mira do robô em relação ao movimento do inimigo.

```
// BANG BANG
double balaPower = 1 + Math.random()*2;
double balaSpeed = 20 - 3 * balaPower; // Quanto maior
//a potência do tiro, menor será a velocidade do projétil.

// Mira em um deslocamento aleatório na direção geral
//para a qual o inimigo está indo
double velLatIni = e.getVelocity()*
Math.sin(e.getHeadingRadians() - anguloAlvo);
double anguloFuga = Math.asin(8 / balaSpeed);

// Signum produz 0 se não estiver se movendo, o que
//significa que atiraremos diretamente de frente
//para um alvo imóvel
double iniDirection = Math.signum(velLatIni);
double angleOffset = anguloFuga*iniDirection*Math.random();
setTurnGunRightRadians(Utils.normalRelativeAngle(
```



```

    sanguloAlvo + angleOffset - getGunHeadingRadians()));

distancia = e.getDistance();

// Não atirar se isso for fazer ficarmos com 0 de energia
if(getEnergy() > balaPower) {

    if (distancia < 100){ // De perto, força máxima
        //Volta atras quando esta muito perto do inimigo
        setBack(50);
        fire(3);
    }else if (distancia >= 100 && distancia < 300){
        //Caso a distancia do robo inimigo seja menor que
        //100 pixels o robo ira dar um tiro forte, caso
        //seja entre 100 e 300 um tiro médio
        setFire(2);
    }//else
    else{
        setFire(balaPower);
    }//else
}//if

```

A variável “*iniDirection*” recebe a função “*Math.signum()*” que retorna o argumento do sinal passado, que se for maior que zero, retornará 1, se for igual a zero receberá 0 e se for menor que zero, retornará -1, no caso em questão, utiliza o sinal “*velLatIni*” determinando que se for positivo receberá 1 indicará que o inimigo está se movendo lateralmente para a direita em relação ao “*Miauzinho*” e se negativo -1 então estará se movendo lateralmente para a esquerda, ou seja, determina a direção de fuga em relação ao inimigo, assim o “*angleOffset*” calcula o deslocamento aleatório na direção geral para a qual o inimigo está se movendo, introduzindo uma pequena aleatoriedade na mira para ser menos imprevisível para o adversário.

Para a tomada de decisão para disparar o tiro, primeiramente verifica se o nível de energia do robô “*getEnergy()*” é maior do que o poder do tiro planejado “*balaPower*”, caso seja verdadeira, o robô efetuará o disparo sem esgotar toda sua energia, assim com base na distância, se for menor que 100 *pixels* irá disparar um tiro com a força máxima, que é 3, caso esteja entre 100 a 300 *pixels*, um tiro na potência média, que é 2, e no caso do inimigo estar mais distante, o disparo será com o poder calculado previamente.

- **Rotação do Radar e Movimento:** O controle do radar e o movimento

é uma parte essencial da estratégia de forma eficaz em qualquer tipo de batalha. Ele mantém o radar girando constantemente para detectar inimigos e ajusta sua direção de movimento de acordo com a posição do adversário, independentemente do formato da batalha.

No código o robô ajusta a configuração para controlar o radar e executa a ação “*setTurnRadarRightRadians(Double.POSITIVE\_INFINITY)*” para girar o radar infinitamente. Isso permite que o robô mantenha o radar sempre ativo para detectar inimigos. “*execute()*” é chamado para executar as ações.

```
// Calcular a quantidade de rotação necessária para
//alinhar a direção atual do robô com a direção desejada
double viraVirou = Utils.normalRelativeAngle(
desejadaDirection - getHeadingRadians());

// Ajusta para ir para trás
if (Math.abs(viraVirou) > Math.PI/2) {
    viraVirou = Utils.normalRelativeAngle(viraVirou +
    Math.PI);
    setBack(100);
} else {
    setAhead(100);
} //else

setTurnRightRadians(viraVirou);
```

Utilizamos a função “*Utils.normalRelativeAngle()*” para calcular a menor diferença angular entre dois ângulos e retornar um valor que representa a quantidade de rotação necessária para alinhar a direção atual do robô com a direção desejada. No código em questão, verificamos se a quantidade de rotação necessária do “*viraVirou*” é maior que 90 graus, se for, indica que o “*Miauzinho*” precisa realizar uma rotação para alinhar-se a direção desejada, o que pode ser mais eficiente de ser feito movendo-se para trás, logo, se a condição for verdadeira o código ajusta a direção desejada adicionando 180 graus no “*viraVirou*” e define o movimento para trás com “*setBack(100)*”, permitindo que faça um ajuste rápido de direção enquanto se move em sentido contrário, caso não seja verdadeiro, se moverá para frente. Por fim define a rotação com o “*setTurnRightRadians(viraVirou)*”, fazendo que execute a rota necessária para o alinhamento da direção com a direção desejada, garantindo que aponte na direção correta enquanto se move.

- **Reação a Balas Inimigas:** Quando atingido por balas inimigas, o “*Miauzinho*” avalia a situação e ajusta sua estratégia, incluindo a

possibilidade de mudar de alvo se o robô agressor estiver mais próximo. Essa capacidade é valiosa principalmente em cenários mais caóticos de “*Battle Royale*”.

- **Dança da Vitória:** Em momentos de vitória, o “*Miauzinho*” celebra com uma dança giratória, demonstrando uma personalidade lúdica.

Os principais métodos presentes na plataforma que podemos enfatizar, permitindo que ele tome decisões inteligentes em resposta a eventos específicos durante a batalha podem ser classificados como:

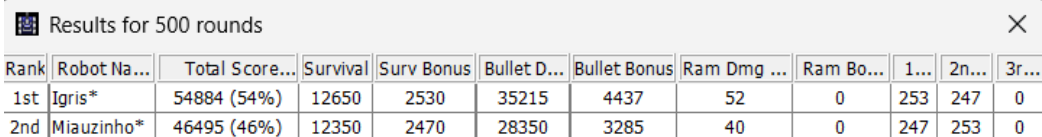
- ***setBodyColor, setGunColor, setRadarColor, setScanColor e setBulletColor*** : São utilizados para definir a cor do corpo, arma, radar, “*scanner*” e projétil do robô;
- ***run*** É o ponto de partida para a execução do robô. No *loop while (true)*, o robô ajusta a configuração para controlar o radar e executa a ação para girar o radar infinitamente. Isso permite que o robô mantenha o radar sempre ativo para detectar inimigos. “*execute()*” é chamado para executar as ações.
- ***onScannedRobot*** Chamado sempre que o robô detecta um inimigo no radar. Neste método, o robô realiza cálculos complexos para determinar a posição e a direção do inimigo, ajustar a mira e decidir quando atirar com base na distância e na velocidade do inimigo.
- ***onHitByBullet*** Chamado quando o robô é atingido por uma bala inimiga. O robô calcula a distância estimada do atacante com base na potência da bala e na velocidade do projétil. Se o atacante estiver mais próximo do que o inimigo atual, o robô muda de alvo.
- ***onHitRobot*** Chamado quando o robô colide com outro robô, no caso do “*Miauzinho*” reage rapidamente, ajustando sua mira e disparando uma bala de alta potência.

## 4 Análise dos Resultados

Com base nas simulações feitas utilizando uma arena de dimensões “800x600”, onde foi realizado batalhas com 500 *rounds* cada, contra diferentes adversários, com o intuito de testar o “*Miauzinho*” em diferentes situações, visto que os robôs enfrentados implementavam diferentes estratégias, obteve-se contra os robôs do semestres 2022-2 e 2023-1 um desempenho notável, demonstrando sua capacidade de competir efetivamente nas batalhas.

No entanto, é importante salientar os principais desafios encontrados em suas batalhas, em principal contra os robôs denominados “*Igris*”, “*Neo*” e “*WallHack*”, onde, durante essas batalhas, observou-se dificuldades ao enfrentar suas estratégias de movimentação, previsão de movimento e suas estratégias de ataque, juntamente com suas miras precisas, impossibilitando a sobrevivência do “*Miauzinho*”.

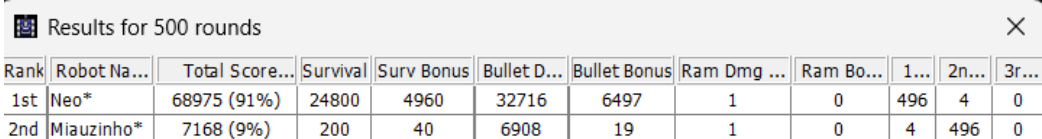
Figura 1: Placar do X1 contra o Igris



Rank	Robot Na...	Total Score...	Survival	Surv Bonus	Bullet D...	Bullet Bonus	Ram Dmg ...	Ram Bo...	1...	2n...	3r...
1st	Igris*	54884 (54%)	12650	2530	35215	4437	52	0	253	247	0
2nd	Miauzinho*	46495 (46%)	12350	2470	28350	3285	40	0	247	253	0

Fonte: Autoria própria.

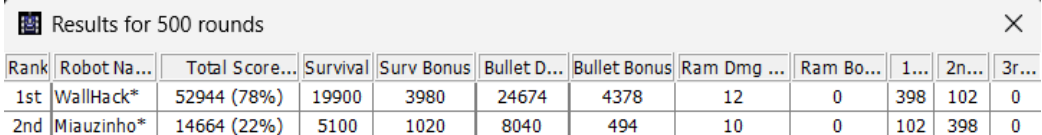
Figura 2: Placar do X1 contra o Neo



Rank	Robot Na...	Total Score...	Survival	Surv Bonus	Bullet D...	Bullet Bonus	Ram Dmg ...	Ram Bo...	1...	2n...	3r...
1st	Neo*	68975 (91%)	24800	4960	32716	6497	1	0	496	4	0
2nd	Miauzinho*	7168 (9%)	200	40	6908	19	1	0	4	496	0

Fonte: Autoria própria.

Figura 3: Placar do X1 contra o WallHack



Rank	Robot Na...	Total Score...	Survival	Surv Bonus	Bullet D...	Bullet Bonus	Ram Dmg ...	Ram Bo...	1...	2n...	3r...
1st	WallHack*	52944 (78%)	19900	3980	24674	4378	12	0	398	102	0
2nd	Miauzinho*	14664 (22%)	5100	1020	8040	494	10	0	102	398	0

Fonte: Autoria própria.

No início do desenvolvimento do “*Miauzinho*”, foram conduzidos testes fundamentais de combate individual contra os adversários “*Walle*”, “*MKIII\_X1*”,

“Freeza”, “Barretinho”, “Ussop”, “SAMU” e “CoadjuvanteBR”. O objetivo era alcançar a vitória decisiva sobre esses oponentes.

Figura 4: Placar do X1 contra o Walle

Results for 500 rounds											
Rank	Robot Na...	Total Score...	Survival	Surv Bonus	Bullet D...	Bullet Bonus	Ram Dmg ...	Ram Bo...	1...	2n...	3r...
1st	Miauzinho*	52613 (52%)	15250	3050	30214	4079	19	0	305	195	0
2nd	Walle*	47892 (48%)	9750	1950	32727	3445	19	0	195	305	0

Fonte: Autoria própria.

Figura 5: Placar do X1 contra o MKIII\_X1

Results for 500 rounds											
Rank	Robot Na...	Total Score...	Survival	Surv Bonus	Bullet D...	Bullet Bonus	Ram Dmg ...	Ram Bo...	1...	2n...	3r...
1st	Miauzinho*	65244 (55%)	18100	3620	37611	5669	100	145	365	135	0
2nd	MkIII_X1*	53283 (45%)	6750	1350	31436	2008	10363	1376	138	362	0

Fonte: Autoria própria.

Figura 6: Placar do X1 contra o Freeza

Results for 500 rounds											
Rank	Robot Na...	Total Score...	Survival	Surv Bonus	Bullet D...	Bullet Bonus	Ram Dmg ...	Ram Bo...	1...	2n...	3r...
1st	Miauzinho*	66330 (88%)	24250	4850	31126	6039	65	0	485	15	0
2nd	Freeza*	8903 (12%)	750	150	7820	153	30	0	15	485	0

Fonte: Autoria própria.

Figura 7: Placar do X1 contra o Barretinho

Results for 500 rounds											
Rank	Robot Na...	Total Score...	Survival	Surv Bonus	Bullet D...	Bullet Bonus	Ram Dmg ...	Ram Bo...	1...	2n...	3r...
1st	Miauzinho*	73247 (56%)	17650	3530	45091	6764	84	128	354	146	0
2nd	Barretinho*	56972 (44%)	7300	1460	32047	2422	12448	1296	147	353	0

Fonte: Autoria própria.

Figura 8: Placar do X1 contra o Barretinho

Results for 500 rounds											
Rank	Robot Na...	Total Score...	Survival	Surv Bonus	Bullet D...	Bullet Bonus	Ram Dmg ...	Ram Bo...	1...	2n...	3r...
1st	Miauzinho*	73247 (56%)	17650	3530	45091	6764	84	128	354	146	0
2nd	Barretinho*	56972 (44%)	7300	1460	32047	2422	12448	1296	147	353	0

Fonte: Autoria própria.

Figura 9: Placar do X1 contra o Ussop

Results for 500 rounds											
Rank	Robot Na...	Total Score...	Survival	Surv Bonus	Bullet D...	Bullet Bonus	Ram Dmg ...	Ram Bo...	1...	2n...	3r...
1st	Miauzinho*	83516 (96%)	24900	4980	44705	8925	6	0	498	2	0
2nd	Ussop*	3211 (4%)	100	20	3044	40	7	0	2	498	0

Fonte: Autoria própria.

Figura 10: Placar do X1 contra o Samu

Results for 500 rounds											
Rank	Robot Na...	Total Score...	Survival	Surv Bonus	Bullet D...	Bullet Bonus	Ram Dmg ...	Ram Bo...	1...	2n...	3r...
1st	Miauzinho*	63440 (82%)	24150	4830	28861	5580	19	0	483	17	0
2nd	SAMU*	14079 (18%)	850	170	12758	296	5	0	17	483	0

Fonte: Autoria própria.

Figura 11: Placar do X1 contra o CoadjuvanteBR

Results for 500 rounds											
Rank	Robot Name	Total Score...	Survival	Surv Bonus	Bullet D...	Bullet Bonus	Ram Dmg ...	Ram Bo...	1...	2n...	3r...
1st	Miauzinho*	60579 (80%)	23300	4660	27435	5116	46	23	466	34	0
2nd	CoadjuvanteBR*	15084 (20%)	1700	340	11395	407	1174	68	34	466	0

Fonte: Autoria própria.

Em seguida, foi realizado um combate em que todos os adversários atuaram simultaneamente, com a alteração daqueles que originalmente foram projetados para um combate individual para os que foram desenvolvidos para combate em grupo.

Figura 12: Placar do Battle Royale

Results for 500 rounds											
Rank	Robot Name	Total Score...	Survival	Surv Bonus	Bullet D...	Bullet Bonus	Ram Dmg ...	Ram Bo...	1...	2n...	3r...
1st	MkIII*	233563 (11%)	156250	7400	60928	4688	3744	553	75	75	72
2nd	Igris*	220598 (11%)	132600	8300	63050	4894	11190	564	84	33	42
3rd	Miauzinho*	205580 (10%)	143450	8800	48705	2645	1880	100	88	44	47
4th	SAMU*	194957 (9%)	139200	4200	47055	3272	1170	60	42	74	52
5th	Walle*	194624 (9%)	126750	3800	54026	3405	6487	156	40	43	48
6th	CoadjuvanteBR*	193254 (9%)	119100	5100	48870	3380	15529	1275	52	45	31
7th	Neo*	187629 (9%)	129350	5300	40951	2353	9307	368	53	45	49
8th	Freeza*	180488 (9%)	127900	3200	46164	2593	611	20	33	58	42
9th	Barretinho*	164578 (8%)	104050	400	44457	3280	11119	1272	6	36	47
10th	WallHack*	154803 (8%)	101300	2100	45950	1944	3331	178	21	25	41
11th	Ussop*	131553 (6%)	92600	1000	33974	1106	2749	124	10	19	30

Fonte: Autoria própria.

Também identificamos uma estratégia específica que demonstrou um desempenho significativamente superior ao enfrentar o denominado robô “*Barretinho*”, esta melhoria envolveu a redução do valor fixo utilizado para determinar a quantidade de deslocamento angular, conhecida como variável “*anguloFuga*” de 8 para 3. Entretanto, é importante ressaltar que essa estratégia foi mais eficaz quando utilizada contra robôs com função “*Tracker*” semelhantes ao “*Barretinho*”, mas quando aplicada contra os outros robôs testados anteriormente, essa alteração resultou em uma pequena redução no desempenho.

Figura 13: Placar do X1 contra o Barretinho valor fixo 8

Results for 500 rounds											×
Rank	Robot Na...	Total Score...	Survival	Surv Bonus	Bullet D...	Bullet Bonus	Ram Dmg ...	Ram Bo...	1...	2n...	3r...
1st	Miauzinho*	73247 (56%)	17650	3530	45091	6764	84	128	354	146	0
2nd	Barretinho*	56972 (44%)	7300	1460	32047	2422	12448	1296	147	353	0

Fonte: Autoria própria.

Figura 14: Placar do X1 contra o Barretinho valor fixo 3

Results for 500 rounds											×
Rank	Robot Na...	Total Score...	Survival	Surv Bonus	Bullet D...	Bullet Bonus	Ram Dmg ...	Ram Bo...	1...	2n...	3r...
1st	Miauzinho*	86279 (70%)	23300	4660	48907	9172	101	139	466	34	0
2nd	Barretinho*	37214 (30%)	1700	340	22847	355	11219	754	34	466	0

Fonte: Autoria própria.

Alguns desafios não esperados no comportamento do “*Miauzinho*” durante as batalhas foram duas em questão: a mudança de alvo ao ser atingido e a colisão ocasional com parede. Durante algumas batalhas observa-se que o seu comportamento para um novo alvo após ser atingido por uma bala não estava funcionando conforme o esperado, o que resultou em ocasionadas perdas. Já a colisão contra as paredes mesmo havendo uma função destinada a evitar essa situação, o que resultou em um impacto negativo em sua posição tática e sua capacidade de manobra, principalmente se outro robô estiver lhe prendendo.

A estratégia de não atirar quando o robô está prestes a ficar sem energia foi de suma importância para garantir que o robô continuasse se movendo e executando suas ações. Se o robô atirasse mesmo quando a energia estivesse quase esgotada, ele desativaria, ficando parado, vulnerável a ataques inimigos sem a capacidade de reagir. Essa abordagem de tomar decisões com base na distância para o inimigo e no nível de energia do robô torna a estratégia de disparo do “*Miauzinho*” mais adaptativa e eficiente, permitindo que ele se adapte às condições da batalha e aumente suas chances de acerto.

Além disso, é importante ressaltar que a aplicação dessa estratégia foi fundamental para nossos sucessos em batalhas, especialmente durante a competição em sala de aula. Muitas de nossas vitórias ocorreram devido a outros robôs ficarem sem energia, incapacitando-os de continuar a luta. Isso demonstra a eficácia de nossa abordagem, considerando não apenas o ataque, mas também a gestão de recursos como a energia, resultando em um desempenho vitorioso.

Em suma, embora “*Miauzinho*” tenha demonstrando um desempenho notável em muitas batalhas, é natural enfrentar desafios específicos, e mediante a batalha final ocorrida em sala de aula, obteve-se o estimado primeiro lugar no *ranking* no combate individual, *battle royal* contra os demais participantes da sala e por fim contra todos participantes somado com os da turma 2022-2. Sendo uma competição de 101 *rounds* na arena de dimensões “1000x1000”.



## 5 Trabalhos Futuros

Para futuros desenvolvimentos, planeja-se explorar estratégias com ênfase em um melhor desempenho em confrontos contra adversários que possuam uma movimentação de difícil previsão, melhorando a taxa de acerto de tiros do “*Miauzinho*”, além do aprimoramento de algumas das implementações ao qual não funcionaram como o planejado, como o melhor desvio das balas, permitindo ser mais imprevisível ainda, possibilitando o tornar mais competitivo em ambientes de combate robótico.

## 6 Referências

### Referências

- [1] Robocode. Disponível em: <https://robocode.sourceforge.io/>. Acesso em: 06/09/2023.
- [2] Robocode Documentação oficial. Disponível em: <https://robocode.sourceforge.io/docs/robocode/>. Acesso em: 06/09/2023.