Cotação: Cada questão vale 2 valor.

Duração: 2 horas sem tolerância.

1. Explique sucintamente como se pode transformar um servomotor genérico para que o sinal PWM passa a representar a velocidade de rotação contínua em vez da posição do eixo. Auxilie-se do diagrama da figura 1, se achar conveniente.

2. As classes interface **Navigator** e **Localiser** usam o mecanismo se sincronização do Java para implementarem o navegador do IntelliBrain-Bot. a) Indique para que é usado esse mecanismo e porquê. b) A técnica de localização usada pelo navegador foi a Dead Reckoning, cujos cálculos foram implementados na classe **OdometricLocalizer**. Demonstre, através das fórmulas da odometria, que a diminuição do diâmetro das rodas aumenta a precisão do cálculo da pose do robô. c) O sistema de navegação é ainda formado pela classe **Pose** e pela classe **DifferentialDriveNavigator**. Considere que também se implementou uma técnica de localização baseada em GPS numa classe chamada **GPSLocalizer**. Represente num diagrama a relação hierárquica entre todas as classes mencionadas no enunciado desta questão 2.

3. Considere o sensor GP2D12 estudado nas aulas. a) Descreva o seu princípio de funcionamento. b) Sendo um sensor que usa luz infravermelha, está sujeito a interferências de outras fontes de luz infravermelha presentes no ambiente. De que maneira essas interferências são bloqueadas pelo sensor? c) Escreva uma pequena função em Java que implemente o algoritmo de seguimento da parede esquerda para um robô como o representado na figura 2 (equipada com 3 sensores GP2D12).

4. Considere o labirinto do concurso Robô Bombeiro representado na figura 3. a) Determine um caminho entre o local onde se encontra o robô e o local assinalado com uma cruz, aplicando um dos métodos de planeamento de caminhos em ambientes conhecidos que estudou.

5. Descreva os tipos de fusão sensorial que conhece e apresente uma possível aplicação da fusão sensorial tipo colaboradora no contexto do concurso Robô Bombeiro.

6. O simulador do concurso CiberRato usa uma arquitectura cliente-servidor. a) Descreva sucintamente a interacção entre o simulador e os agentes (robôs virtuais). b) Descreva sucintamente as características desses agentes, no que se refere a sensores e actuadores. c) Considere que o robô apenas tem que encontrar um farol e que o sensor de farol do robô consegue detectar a direcção do farol a partir de qualquer posição. Descreva através de pseudocódigo, ou de um fluxograma, um algoritmo BUG para o robô encontrar o farol.

7. "Localização Baseada em Marcas Artificiais" e "Localização Baseada em Marcas Naturais", foram duas das técnicas de localização de robôs móveis estudados nas aulas. a) Indique as diferenças entre essas duas técnicas. b) Indique exemplos de aplicações de robôs móveis onde essas técnicas poderiam ser usadas (2 exemplos para cada método). c) Considera essas técnicas de localização úteis no contexto do concurso Robô Bombeiro? Justifique.

8. Considere o diagrama da placa IntelliBrain na folha em anexo. a) Desenhe um esquema representativo das ligações entre a placa, a bateria e os sensores e actuadores presentes no desenho. b) Quantos sensores SRF04 podem ser ligados à placa? Justifique.

9. Nas aulas estudámos um sensor muito simples para detectar a chama da vela, usando um fototransistor. a) Descreva o princípio de funcionamento do circuito electrónico desse sensor. b) Escreva uma pequena função em Java para efectuar leituras com

esse sensor e apresentar o valore lido na linha 1 do ecrã LCD. Considere que o sensor está ligado a uma porta digital.

10. A figura 4 descreve o comportamento de um robô que segue o código implementado na classe Vacation. a) Considere que quando o código foi executado observou-se que o robô ao chegar ao ponto de coordenadas (30, 30) parou durante cerca de 10 segundos e só depois navegou até ao ponto de partida. Tendo em conta os comportamentos implementados, explique esta paragem. b) Descreva o sistema de controlo do robô através de uma máquina de estados finitos.

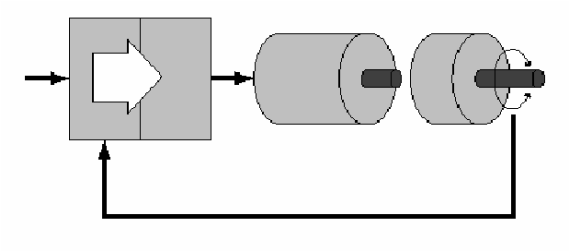


Fig. 1

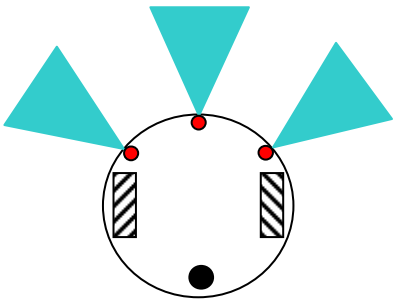


Fig. 2

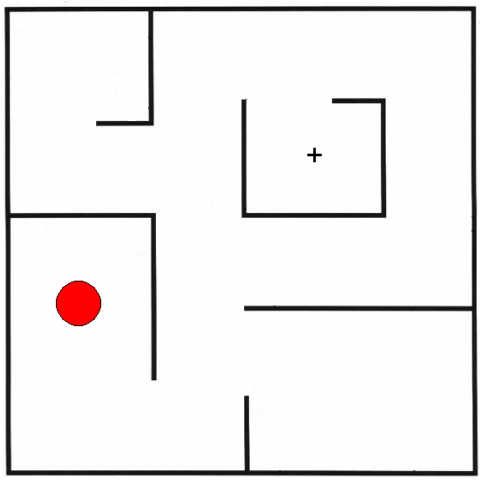
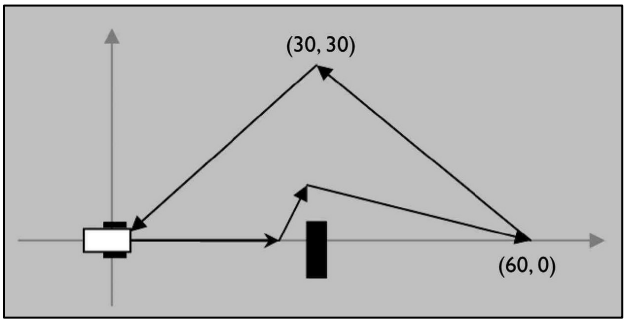


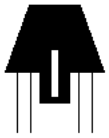
Fig. 3

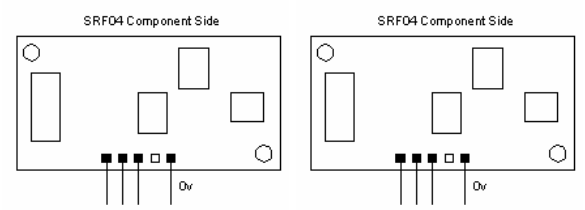


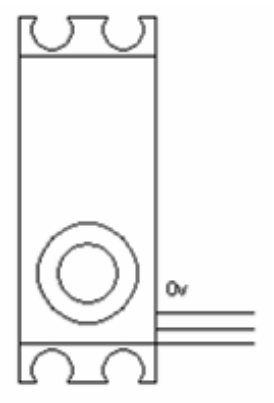

```
public class Vacation implements Runnable {
  private BehaviorArbiter mArbiter;

  public Vacation(Localizer localizer,
                  Navigator navigator,
                  AnalogInput leftRange,
                  AnalogInput rightRange,
                  int priority) {
  Behavior behaviors[] = new Behavior[] {
    new AvoidBehavior(localizer, navigator, leftRange,
                      rightRange, 200, 0.7f, 3000),
    new ReturnHomeBehavior(navigator, 45),
    new GoToBehavior(navigator, 60.0f, 0.0f),
    new GoToBehavior(navigator, 30.0f, 30.0f),
  };

  mArbiter = new BehaviorArbiter(behaviors, 200, null);
  mArbiter.setPriority(priority);
}
```

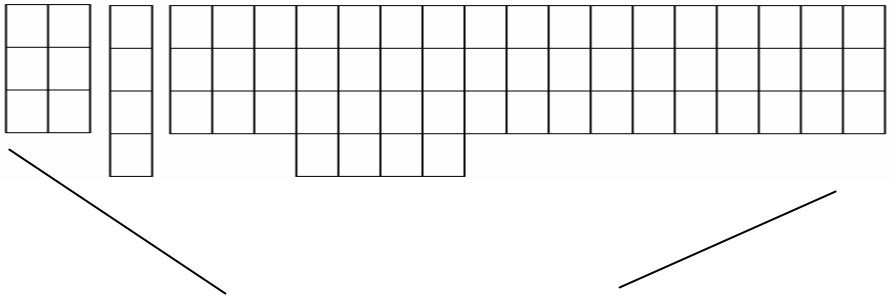
Fig. 4

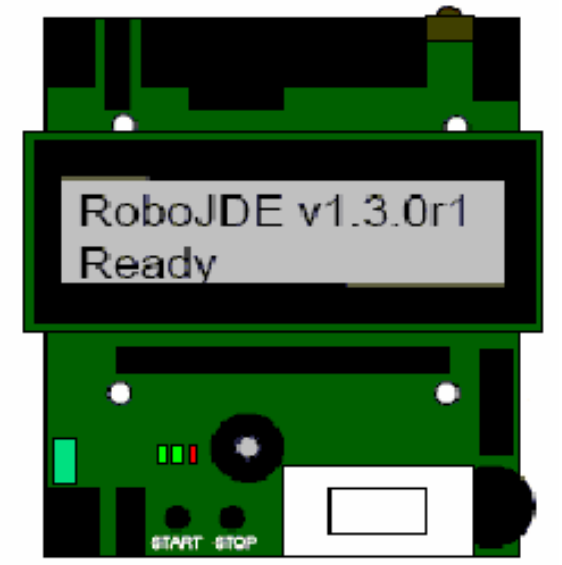
Nome: _____ Nº: _____

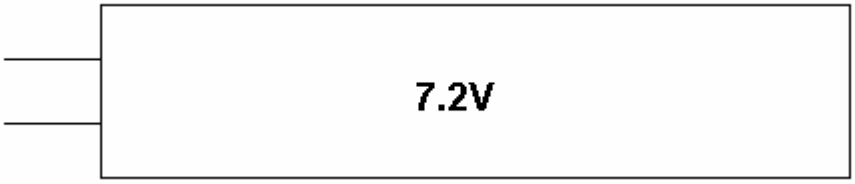



SRF04 Component Side
0v
SRF04 Component Side
0v

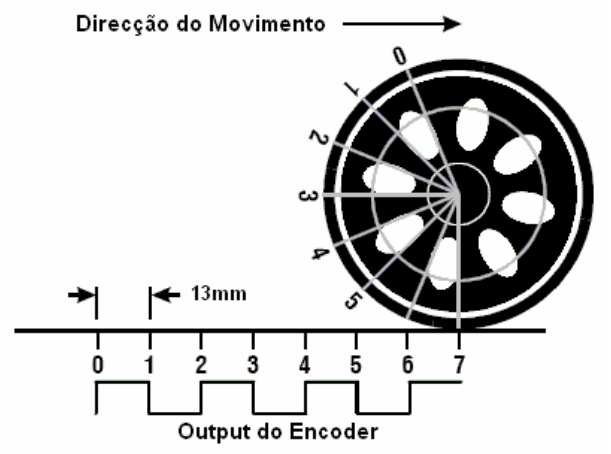

0v



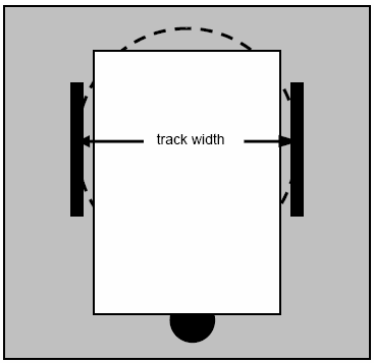

RoboJDE v1.3.0r1
Ready
START
STOP


7.2V

Formulas da Odometria



deltaDistance = (leftCounts + rightCounts) / 2 * distancePerCount
distancePerCount = Pi * diameterWheel / countsPerRevolution

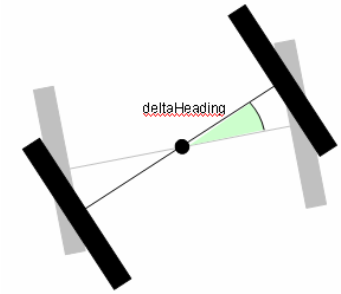


O perímetro da circunferência é:
circunferenceTw = Pi * trackWidth

O perímetro de uma roda é:
circunferenceWheel = Pi * wheelDiameter

countsPerRotation = (circunferenceTw / circunferenceWheel) * countsPerRevolution
countsPerRotation = (trackWidth/wheelDiameter) * countsPerRevolution
countsPerRotation = 2 * (trackWidth/wheelDiameter) * countsPerRevolution

radiansPerCount = 2 * Pi / countsPerRotation
radiansPerCount = 2 * Pi / (2 * (trackWidth / wheelDiameter) * countsPerRevolution)
radiansPerCount = Pi / ((trackWidth / wheelDiameter) * countsPerRevolution)
radiansPerCount = Pi * (wheelDiameter / trackWidth) / countsPerRevolution



deltaHeading = (rightCount – leftCount) * radiansPerCount