Cotação: Cada questão vale 2 valor.

Duração: 2 horas sem tolerância.

1. Com o auxílio da figura 1 explique o princípio de funcionamento de um servomotor.

2. As classes interface **Navigator** e **Localiser** são a base do sistema de navegação implementado para o robô IntelliBrain-Bot. a) Descreva qual a motivação para o uso de classes interface na programação de robôs móveis em Java. b) A técnica de localização usada pelo sistema de navegação foi a Dead Reckoning cujos cálculos foram implementados na classe **OdometricLocalizer**. Se aumentarmos em 50% a distância entre as rodas do robô, qual a diferença na direcção do robô, quando as rodas rodam uma volta completa em sentidos opostos? 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 SRF04 estudado nas aulas. a) Classifique-o quanto ao seu uso típico, ao tipo de informação que obtêm e à energia que utiliza. b) Descreva o seu princípio de funcionamento; c) Escreva uma pequena função em Java para efectuar leituras com esse sensor e apresentar o valor lido na linha 1 do ecrã LCD.

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 o método de planeamento de caminhos "Grafo de Visibilidade". b) O caminho determinado não poderá ser usado directamente pelo robô. Indique porquê e apresente uma possível solução para o problema.

5. Considere que se pretendia desenvolver um robô móvel que fosse capaz de construir na sua memória, à medida que navegasse por uma casa, um mapa do tipo "Grelha de Ocupação". Descreva sucintamente como o robô construiria esse mapa, indicando: os sensores usados pelo robô; a informação armazenada na grelha; o método de navegação/construção.

6. O simulador do concurso CiberRato usa uma arquitectura cliente-servidor. a) Descreva sucintamente o funcionamento do simulador. b) Descreva sucintamente as características dos agentes (robôs virtuais) usados no simulador. c) Descreva através de pseudocódigo, ou de um fluxograma, o algoritmo genérico do robô virtual "Agente008" implementado nas aulas.

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. Desenhe um esquema representativo das ligações entre a placa, a bateria e os sensores e actuadores presentes no desenho.

9. Considere o robô da figura 2 com um sistema diferencial de direcção e com 2 sensores de detecção da chama baseados em fototransistores, iguais aos estudados nas aulas. a) Apresente o circuito electrónico destes sensores. c) Descreva o seu princípio de funcionamento. b) Escreva um pequeno programa em Java para o IntelliBrain-Bot para controlar o robô de modo a que este vá de encontro à chama de uma vela (semelhante ao que foi feito nas aulas).

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.

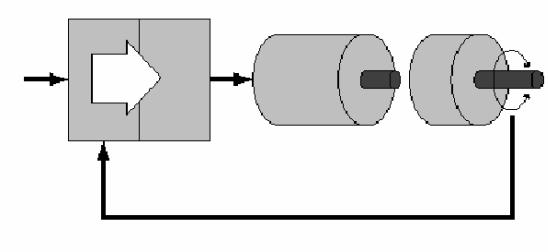


Fig. 1

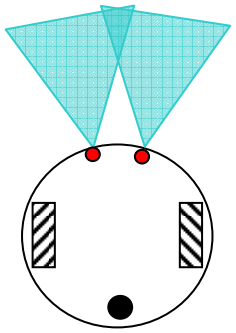


Fig. 2

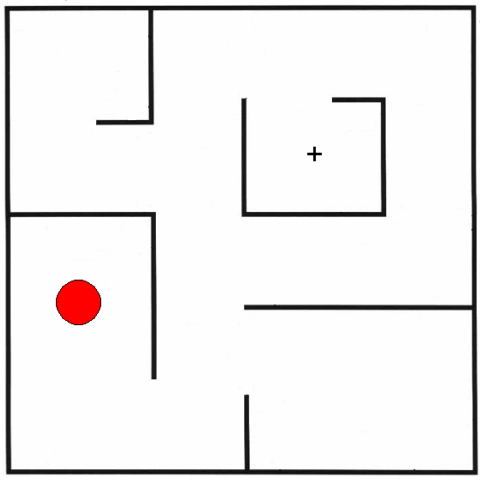
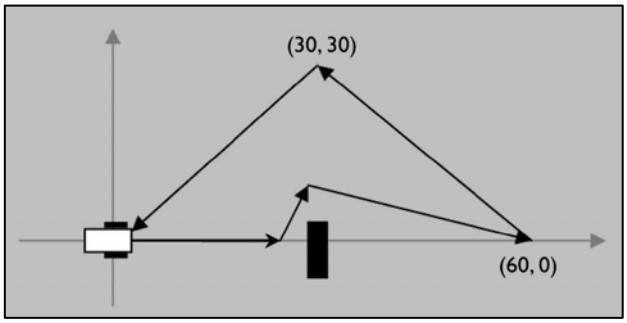


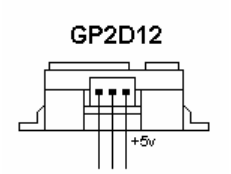
Fig. 3

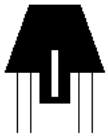


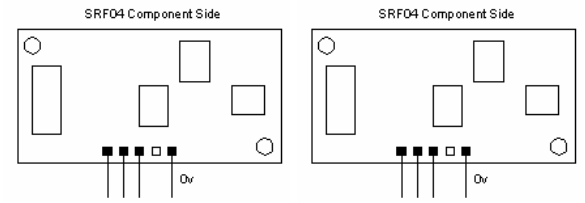

```
public class Vacation implements Runnable {
  private BehaviorArbiter mArbiter;

  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);
}
```

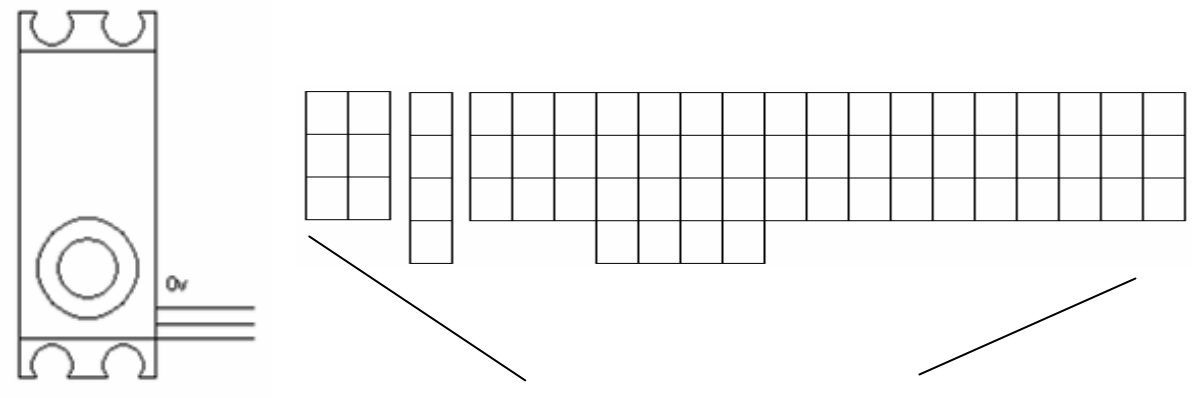
Fig. 4

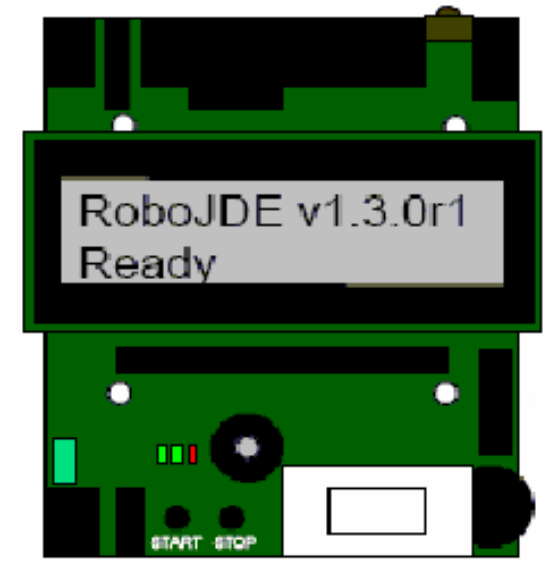
Nome: _____ Nº: _____

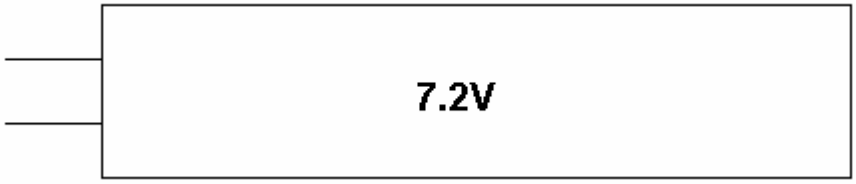

GP2D12
+5v



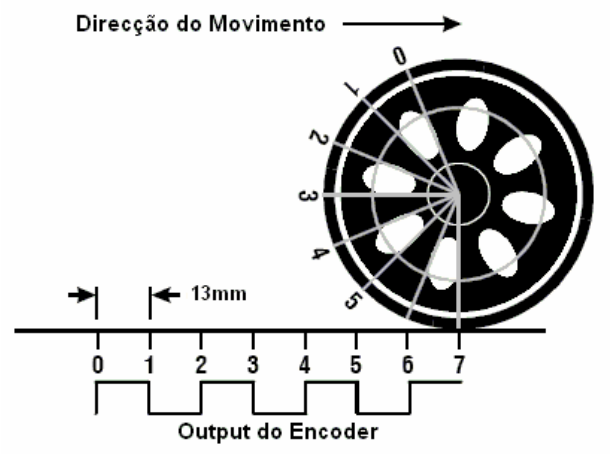

SRF04 Component Side
0v
SRF04 Component Side
0v


0v

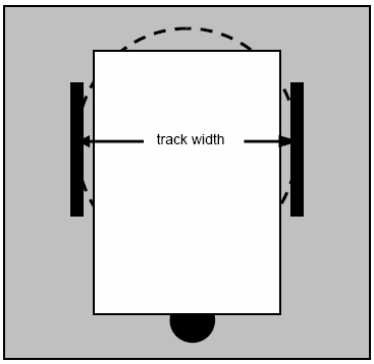

RoboJDE v1.3.0r1
Ready
START
STOP


7.2V

Formulas da Odometria



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

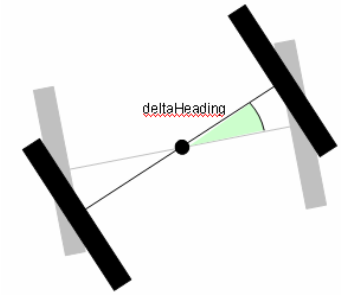


O perímetro da circunferência é:
circunferenceTw = Pi * trackWidth

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