

# Robótica



## GOVERNO CIVIL DA GUARDA

Gabinete do Governador

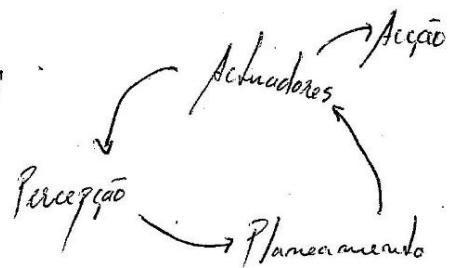
13/4/2

### → Actuadores em robótica móvel

- Basicamente, actuadores "atuam" no ambiente do robô
- Actuadores de vários tipos

Nota: Buzzers, LEDs também são actuadores!

Actuadores tipo motor → movimentam as partes móveis



### → Pneumáticos

- Cilindros ou motores pneumáticos → relativos, como o eléctrico
  - ↳ pistão tem movimento linear
- Utiliza ar comprimido como energia
- Controlo difícil, mas são conhecidos por serem potentes (em termos de força)
- Requerem manutenção extra (mais do que o eléctrico) → só ligar e já está
  - ↳ estruturas auxiliares: bombas, válvulas, óleo, etc.

Nota: Ver músculo artificial, tipo os actuadores do i-Robot

### → Hidráulicos

- Em vez de ar, usam água ou outro tipo de óleo
- Têm mais força que os pneumáticos e precisão (estabilidade)
- Funcionam de forma semelhante que os pneumáticos

### → Actuadores eléctricos (motores)

- Mais utilizados na robótica móvel
- Acessibilidade e não necessitam de toda a estrutura como os hidráulicos e pneumáticos
- Mais fáceis de controlar e mais precisos

• Vários tipos

→ Corrente alternada

→ " continua + usados

→ de gás → controlado por um circuito especial que envia impulsos para controlar muito preciso

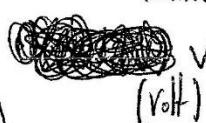
→ Giro motores

↳ mais precisos, mas não como os de gás

→ motores CC

• Funcionamento nos apontamentos da aula

Características



(volt)

• Tensão nominal → alimentação recomendada pelo fabricante

↳ pode funcionar com tensões inferiores, com um custo na performance (menos) superiores, mas dentro de uma tolerância

• Corrente nominal  
(ampere)

A

↳ máxima quando não tem carga (tipicamente pequena)  
↳ máxima quando a carga é tanta que o motor não roda  
↳ Idealmente, o controlador do motor deverá suporar essa corrente máxima  
Nota: mínimos e máximos são sempre jeito saber, para projecção de custos  
A partir daqui escolhe-se a bateria (voltagem, amperagem)

• Potência elétrica

↳ Pode ser interessante para alguns projetos

• Binário, torque, par binário

• Velocidade angular ( $\omega$ )

• Potência mecânica = Binário  $\times \omega$

Nota: conversão de binário em força

⇒ Engenheiros

⇒ Motores CC

• Alimentação

↳ com um motor disk gênero e 2 portas digitais controla-se o motor e o sentido de rotação

• Circuitos de controlo

→ controlo eletrónico por pente-H

Nota: inversão de polaridade

↳ com transistores

↳ motor roda em sentido contrário

No entanto, também pode interessar o cômputo da velocidade

- Sinais de 0 e 1 da ponte H são substituídos por sinais PWM (Pulse width modulation)
    - É ~~possível~~, efectivamente, por exemplo, para um motor de 12 V, alimentar 8 V e fazer a velocidade variar (como consequência)

104

Sata 16 W. (2 points II)

- No inálteram dí para substituir o circuito de controlo de motores
    - ↳ Também por questão de acessibilidade → circuito é muito sensível à corrente e pode falhar facilmente
  - Para programas no inálteram com servo motores é melhor usar Continuous Rotation Servo

2013/6/22

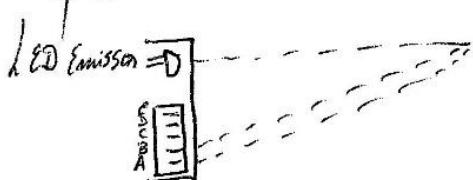
→ Sensores de distância

*[Signature]*

- Tempo de Vôo
    - Considera-se velocidades do vento constantes
    - Som percorre o dobro da distância a que o obstáculo se encontra

- ## • Triangulação

- Determina distância com o Teorema de Pitágoras
  - Em função do fototransistor que recebe a onda, haverá um ângulo apoxionado



Se receber no fototransistor A → ângulo u  
u u u " B → u y

## » Equipamento

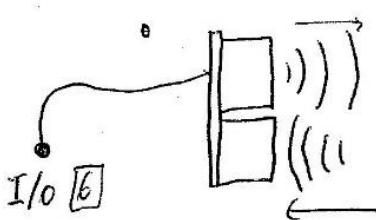
→ Somar SRF05 (de distância)

- O que iremos usar

- 2 modos → 1 porta digital para o disparo

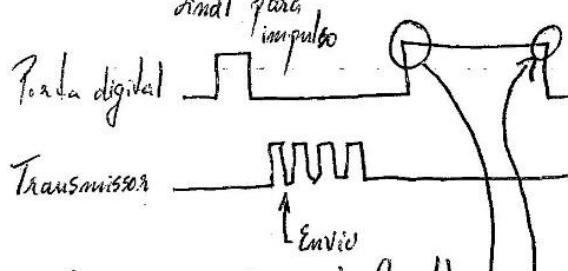
    ↳ 1 porta digital para receber

    ↳ 1 porta para enviar/receber ← Gasta menos bateria



$$d = At \times \frac{1}{2} \sqrt{c}$$

• Impulso mínimo de 10  $\mu s$

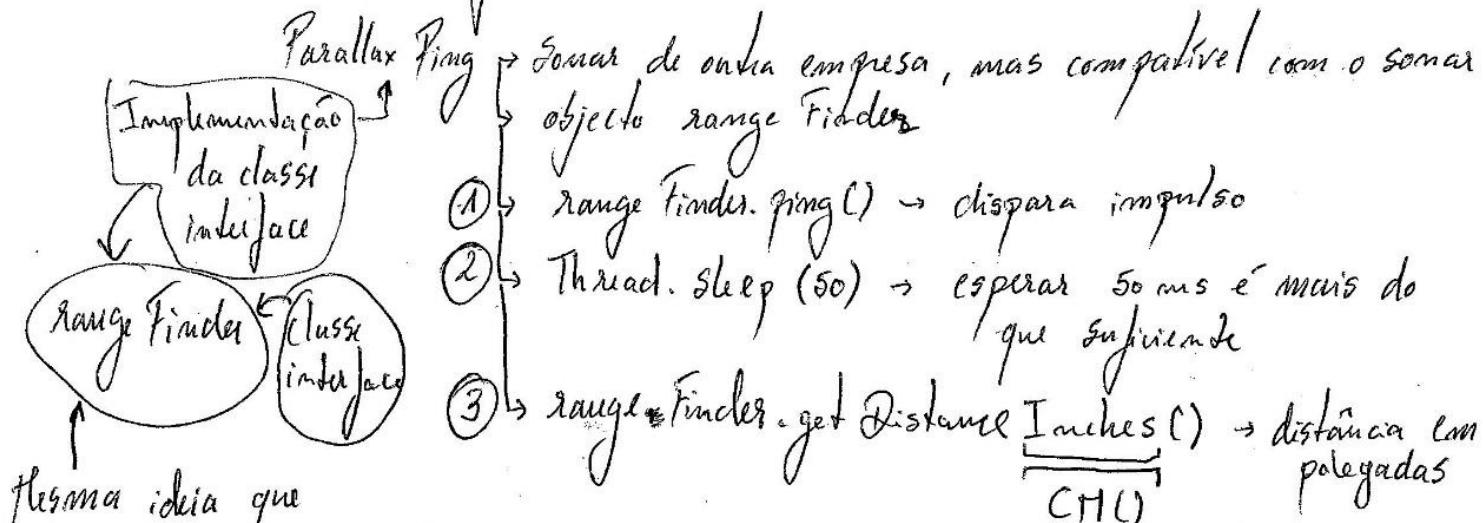


Para evitar ficar preso à espera (atraso)

- Têm limites aos 30 ms (fabricante)
- final é colocado a 0
- Se não receber o eco
- No fundo, define a distância do somar

Porta digital fica a 1 até receber o final

- Classe RangeFinder fornecida



a classe Motor → ~~controle~~ controlo de vários sensores com a mesma classe interface

→ SRF04

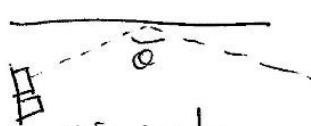
Gabinete do Governador

- Precisa de 2 portas digitais → gasta + bateria
- SRF05 funciona da mesma forma

→ SRF08

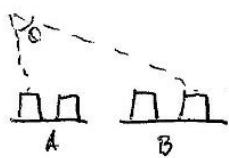
- Para usar no IBrain → declarar objecto Devantech
- Sensor de distância e luminosidade
  - ↳ já é radicalmente diferente
  - ↳ poderia utilizar portas I<sup>2</sup>C
  - ↳ mas não dá para detectar luminosidade porque a chama não provoca alteração significativa
  - ↳ a classe interface ~~Parallax~~ Parallax Ring mas não foi implementada

- Sensores de distância medem a distância se estiverem perpendiculars ao objecto



- ↳ Devido à reflexão especular
- ↳ Não há grande volta a dar
- ↳ Bumper é uma mitigação

- Cross talk → outro sensor recebe um eco que não lhe era destinado



- B recebe o sinal de A
- Ambos enviam impulso
- Solução: só quando A terminar a batuta é que B envia impulso → em sequência

- Quanto mais denso for o meio, mais baixa será a velocidade do som

- ↳ Laser são os melhores → usam tempo de voo
- ↳ Desvantagem grande → se material for transparente, eco não é recebido porque não há reflexão
- ↳ Muito caro → tem hardware muito rápido para cálculos
- ↳ Funciona como ultrassom → 2D

## • Infra vermelhos

↳ Sharp GP2D12 → muito usado no lab de Robótica

↳ Temos precisos → ângulo é aproximado

↳ Distâncias limitadas → difícil arranjar sensor que detecte a mais  
di 1m

↳ Classe específica → implementação com interface Range Finder  
↳ Classe Sharp GP2D12

Nota

Ping e Sharp GP2D12 são implementações da classe Range Finder

Há com somar é necessário usar o Thread Sleep

Há um tempo mínimo entre pings consecutivos

↳ ter em atenção, procurar o mínimo tempo possível

## ~ Sensors de proximidade

→ De linha branca

• Distância de detecção muito pequena

• Infra vermelhos

• Vamos usar o Fairchild PIRB1134

• Indivisa distinguir o branco, mas outras cores também são possíveis (voltage será diferente)

→ Usados nos smartphones para detectar a proximidade com a face

## ~ Sensors de visão

→ Normalmente juncionam com RGB

→ Sensors de cor → detectam a cor

→ 2 problemas → funcionam bem para distâncias muito pequenas (1cm)  
↳ lento, porque precisa da sequência para detectar

↳ difícil posicionamento

Sugestão

Gabinete do Governador

Para o robô, usar a ITUCar se ainda restar tempo antes do concurso  
férias, utilizar outro sensor

## → Sensores de som

- Exemplo: inicia funcionamento depois de ouvir um sinal sonoro
- Detecta apenas o nível de som
- Pode ser desvantagem: ruído parecido e detectado pode activar o robô (quando não queríamos)

## → Sensores de chama

- Robô tem que detectar a chama e assimilar → circuito simples com sensores de chama
- Fototransistor que funciona como interruptor eletrônico
  - ↳ Radiação atinge determinado limiar → porta fica ligada com +5V
  - ↳ Tem amplificadores para aumentar sensibilidade do sensor
  - ↳ Embora consiga detectar a maior distância, passa a detectar outras fontes de luz (exemplo: luz solar)

## • Arquitecturas de controlo

modelação → planeamento e execução → de ações  
num ambiente já alterado → evitar

### → Deliberativa

#### Modelação

→ Arquitectura exige que a modelação esteja actualizada

→ Cria uma representação do mundo onde o robô actua

para se fazer um planeamento de tarefas (próximo passo)



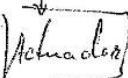
*Nota:* Inteligências não têm capacidade de processamento para implementar este tipo de arquitectura

### → Reactiva

- Idiota oposta da deliberativa



- Tentar ligar de forma mais directa possível os sensores aos actuadores



- Exemplo prático de robô seguir a parede encaixa-se neste tipo de arquitectura

- Rápido e não precisa de grande processamento, mas um pouco limitado

### → Híbrida

- Combinação, tenta eliminar desvantagens da reactiva e deliberativa

- "Posso actuar" → Deliberativa

- "Obstáculo" → Reactiva!

### → Baseada em comportamentos

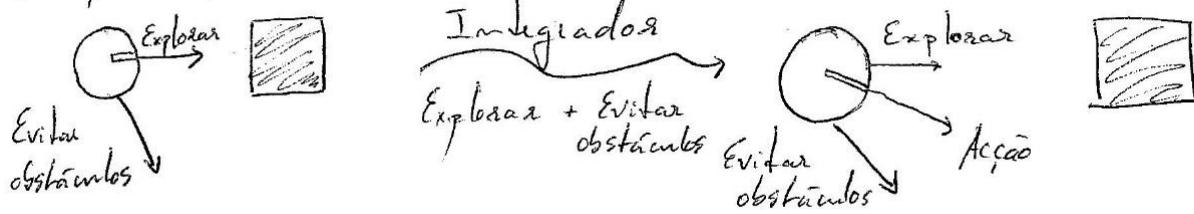
- Cada um controla o robô de forma independente

- Cada comportamento tem um trigger de activação

- Conflito de comportamentos? → híbrido decide (ou então, escala de prioridade)

- Integrador (faz tipo soma de comportamentos)

## Exemplo



\* Pode ser implementada da seguinte forma:

(anotando)

Estado 1 = 1

Estado 2 = 2

Estado  $N$  =  $N$

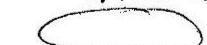
Estado =  $[1, 2, \dots, N]$

Enquanto (~~sair~~) <sup>Sair</sup> /

[Litura de sensores]

Seleção (Estado) (switch) {

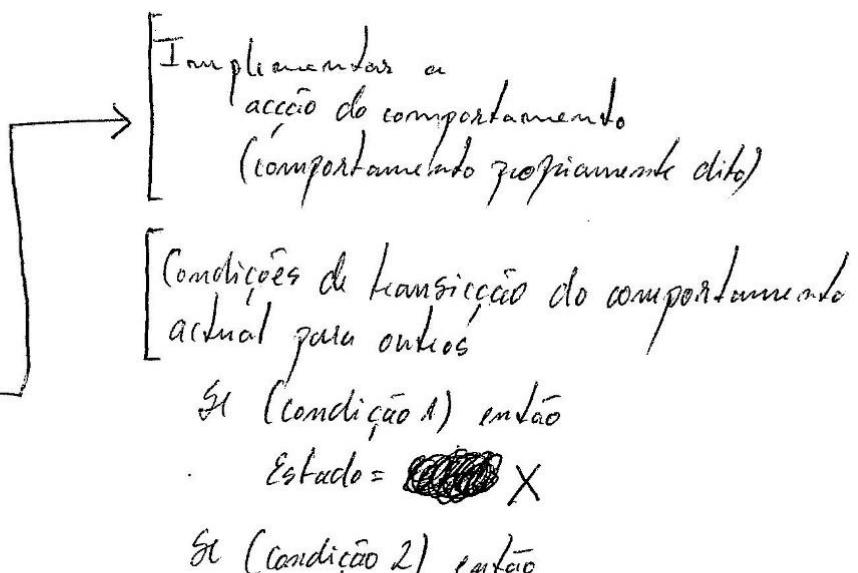
Caso Estado 1:



Caso Estado 2:

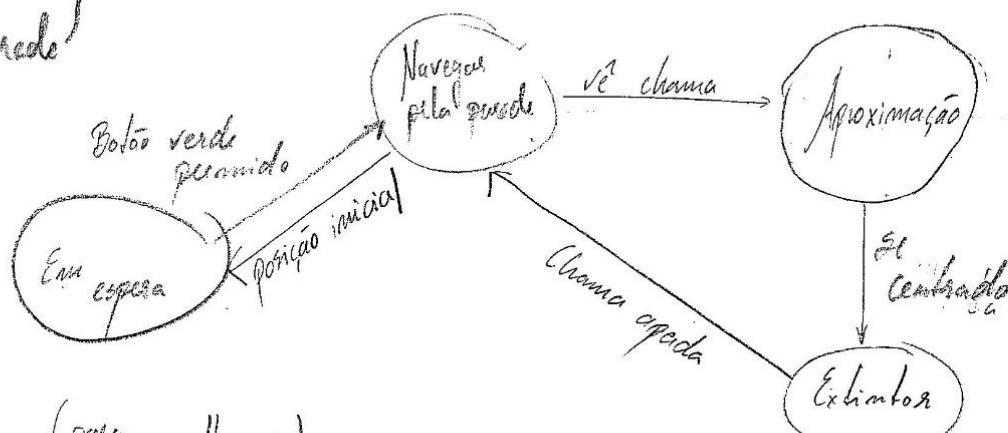
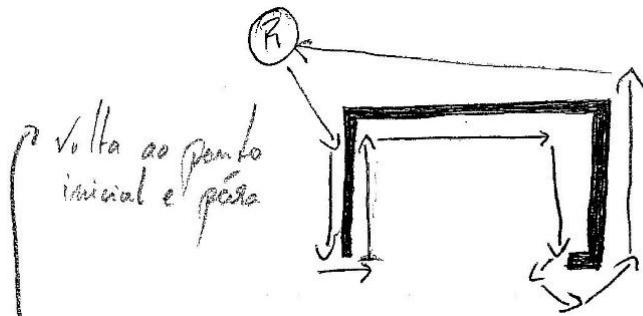


Caso Estado  $N$ :

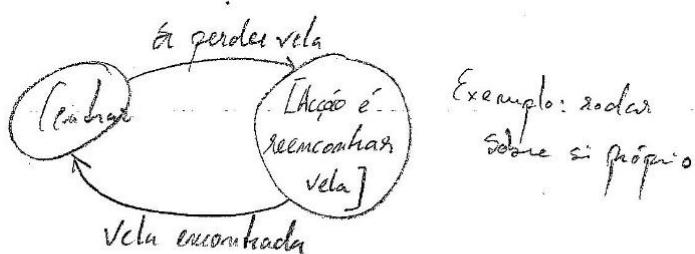


Exercício

- Temporais e ações
- Navegar pela parede
- Identificar chama
- Extinguir chama
- Navegar pela parede

Sugestão do prof

(para melhorar)



Exemplo: rodar  
sobre si próprio

2013/5/13

→ Data limite até dia do concurso → 30 de Junho

→ Gráficos, só adicionam  
alturas cores, no máxi

→ Pôr a fazer com uma semana de margem (+ -), formato técnico com template (AO)

→ Para participar com vídeo, pôr aí 30 de Junho, senão aí 6 de Julho

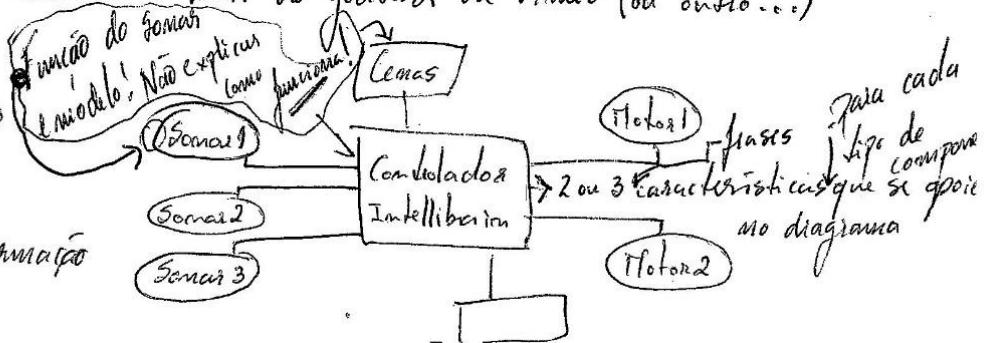
→ Vídeo, tem que ser enviado o link do youtube ou vimeo (ou outro...)

Nota

Diagrama de blocos é importante!

Ex.: Atividade às cores!

É para descrever informação  
técnica



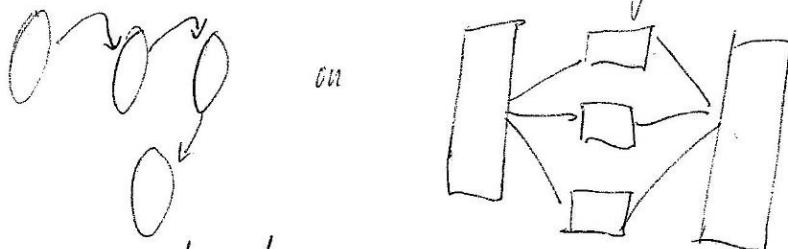
↳ Descrição mecânica é gás o óptimo para jatos.

Nota → Pensar na geometria



↳ Descrição software

↳ máquina estados ou de threads (threads se for dedicado a tempo)



↳ Indicar comportamentos únicos

↳ O que distingue o robô dos outros?

↳ Aspectos mais importantes

↳ aspectos inovadores (se houverem) → algum algoritmo interessante, que tinha valor...

↳ Resultados!

↳ Descrivêr também situações que não corram bem!

↳ Exemplo: problema surge porque há falta de material, p.e.: um quarto que o robô empenha sempre mais, caso houvesse mais um zonal, seria possível resolver o problema do ângulo morto.

↳ Taxa de sucesso: ex: em 10 incêndios, apagam 7!

↳ em termos de ganhos

- ↳ no modo de funcionamento  $X \rightarrow Y$  sucesso
- ↳ aumentando o grau de dificuldade  $\rightarrow Z$  sucesso
- ↳ tempo que o robô demora a apagar

Robustez com os resultados!

## ~ Navegação

"Para onde vou?" → Missão do Robô

Navegação

- ↳ Basada em auto-localização → Deliberativa → Para quando queremos localizar o robô, construir mapa...
- ↳ Reactiva (sem auto-localização) → Comportamento
  - ↳ se próximo da parede, desvia-se
  - ↳ se é canto, conforma...

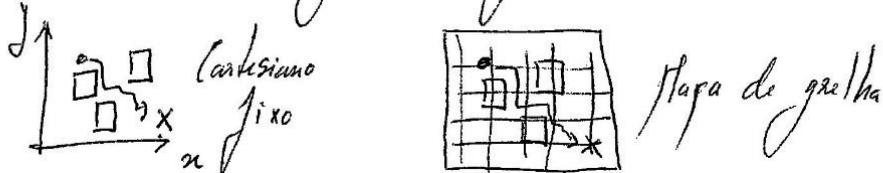
## ~ Auto-localização

→ Referencial cartesiano fixo → posição ( $x, y$ )

→ Mapa de ambiente 2D/3D

↳ Robô sabe onde está e para onde quer ir → Algoritmo para chegar lá

Navegação por algoritmos também pode ser implementado no navegação por referencial, se soubermos localização do objectivo e obstáculos.



## • Estimativa da pose

Medidas → Relativas → processamento relativamente simples (dá com imprecisão)  
→ Absolutas → se o GPS não fizer o processamento, é mundo pesado

Exemplo de processamento com medidas relativas:

$$\begin{aligned} & \text{Initial Position: } (x_i, y_i, \theta_i) \\ & \text{Movement: } (\delta x, \delta y) \\ & \text{New Position: } (x', y', \theta') \end{aligned}$$
$$\begin{aligned} x' &= x + \delta x \\ y' &= y + \delta y \end{aligned}$$

Plus é necessário fazer medições

Gabinete do Governador

regulares de forma a saber a distância percorrida.

↳ Odometria → método para medir deslocamento em função da rotação das rodas  
↳ Vai ser implementado

### • Odometria

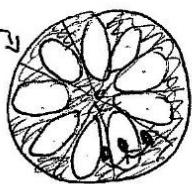
→ Shaft encoders

↳ No IB, com sensores de linha nas rodas →

Passagem de sólido

pelo buraco e ao  
sólido outra vez

↳ Passou um buraco



$$P = 2\pi R$$
$$= D \cdot \pi$$

Se 8 buracos → volta completa tem 16 pulsos

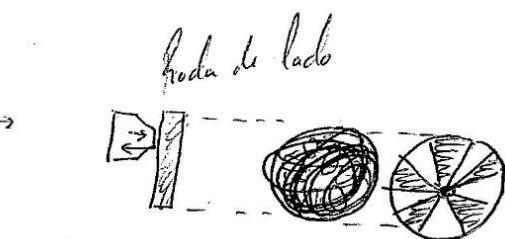
Distância dos 16? → Perímetro da roda

E de um pulso? → Perímetro → 16 pulsos

Nota → Esta forma só funciona

para quando o robô anda em frente.

Para curvas é preciso mais cálculos, então  
simplifica-se.



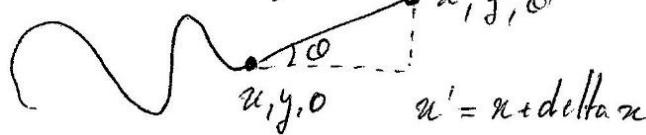
$$n \rightarrow 1 \text{ pulso}$$
$$x = P/16$$

↳ Simplificação → descrição dos movimentos através de 2:

No final, distância é obtida através da média ~~entre~~ de pulsos entre as 2 rodas.

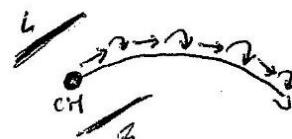
$$(Sd)^{\frac{1}{2}}$$

delta distance ( $\Delta d$ )

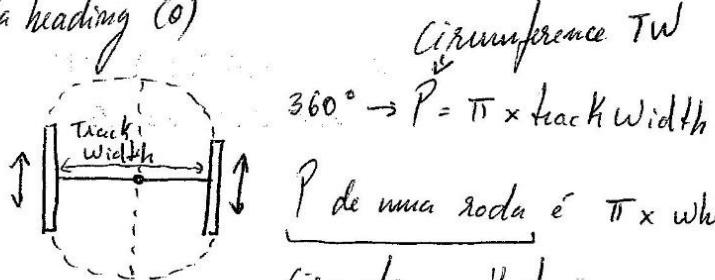
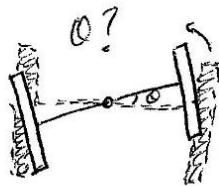


Note

Considerar que é capaz APENAS de  
fazer 2 movimentos



• Fazia saber como calcular o delta heading ( $\theta$ )



P de uma roda é  $\pi \times \text{wheel diameter}$   
Circumference wheel

Página 35!

Circumference Wheel

→ Counts per Revolution      Uma roda gera 16 pulsos  
uma "rodagem" completa

Circumference TW →  $(\pi) L$

Portanto, para uma roda: counts Per Rotation = (Circumference TW / Circumference Wheel)

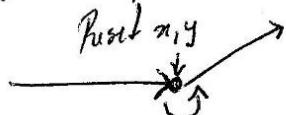
X counts per Revolution

Fazer simplificações

Nota → classe localizer implementa dead reckoning

Nota → Não implementar só odometria

Nota → Fazer reset quando trocar o movimento



Ex.: 2 robôs:

$TW_A = 10 \text{ cm}$        $TW_B = 15 \text{ cm}$       ] → Resto tudo igual → Qual é o mais preciso?  
(com odometria)

Precisão → cálculo das 2 fórmulas → delta → heading → precisa do track width!  
Distance

No delta Heading → Robô A  $\left[ \begin{smallmatrix} \text{e.g.} \\ \text{Robô B} \end{smallmatrix} \right]$  radians per count  $\rightarrow 0.7 \leftarrow +\text{puxo!}$   
 $\rightarrow 0.9$

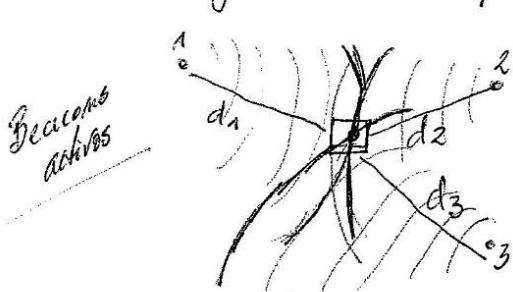
Radians Per Count =  $\pi * (\text{wheel Dia} / \text{track width}) / \text{counts per Revolution}$   
 $\left( \begin{smallmatrix} 10 \\ 10 \end{smallmatrix} \right) \quad \left( \begin{smallmatrix} 10 \\ 15 \end{smallmatrix} \right) \quad \left( \begin{smallmatrix} 16 \\ 16 \end{smallmatrix} \right)$

Quando menor o Radians Per Count, mais preciso é a medição!  
Logo, maior track width implica menor Radians Per Count!

→ medidas absolutas

- Cada vez que se actualiza a posição do robô, a medida é nova e nada tem a ver com a anterior (ex.: GPS).

→ Triangulação e filtração



Minimo dos máximos para filtração

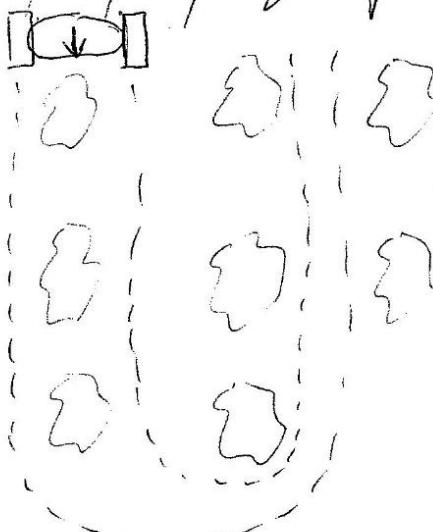
- Com 3 beacons é possível determinar a pose do robô (filtração)

→ Landmarks

→ Tracker externo

→ Mapa topológico

Ex: Robô para guiar as alfaces



Podia ser por:

Navegação com marcas naturais

- ↳ câmara para detectar alface
- ↳ alface verde, terra constante
- ↳ identificação fácil

Navegação com marcas artificiais

- ↳ cabo eléctrico nos carreiros (?) para o robô detectar a radiação e navegar

Qual a melhor? Mais barata?

2013/6/3

[Mapas]





Nota

"para onde vou?" → Máquina de estados  
↳ Comportamentos para cumprir a missão

- Na arquitetura deliberativa é comum usar-se mapa
  - ↳ Na de comportamentos já não, uma vez que é reactiva
- SLAM → robô cria o mapa ~~no momento~~ no momento
- Mapas representam ambiente e permitem a localização do robô
  - ↳ Sensores → Comparação dos dados obtidos com o mapa
- Mapas sensoriais
  - ↳ O que é armazenado como mapa é a informação extraída dos sensores
  - ↳ Número de pontos (sonar, IR, laser)
  - ↳ Imagens (câmaras)
  - ↳ Se o que aponta for igual ao que tem armazenado, robô sabe localizar-se
- Mapas móveis
  - ↳ Mais usados
  - ↳ Descreve o ambiente com medidas
  - ↳ Pode ser fornecido a-priori ou SLAM
  - ↳ De ocupação 2D ou 3D → Célula de ocupação → (Cada célula contém informação que)
    - ↳ Geométrico 2D ou 3D (linhas, segmentos de recta descriptivos do ambiente)
    - ↳ Tem aumentado em popularidade → Evolução dos sensores

Nota

Variante algoritmo Dijkstra (Célula de ocupação)

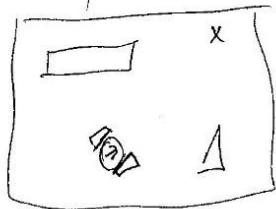
Precinge com labels a partir da pose ate chegar ao destino  
Depois é escolhido caminho

→ Ver A\*

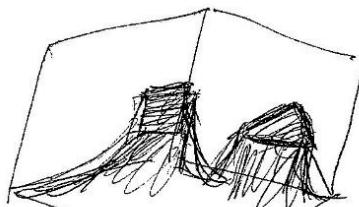
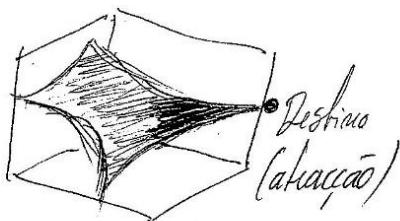
5	5	5	5	5	6	7
4	4	4	4	5	6	X
3	3	3	4	5	6	7
2	2	3	4	5	6	7
1		3	4	5	6	7
0	1		3	4	5	6
1	1	2	3	4	5	6
2	2	2	3	4	5	6

(bulha de ocupação)

↳ flamas de potencial



Gabinete do Governador



Obstáculos (repulsão)

Juros, áreas gradiente  
de navegação

o flamas métricas

↳ Geométrico

Ter ideia, algoritmos muito complicados

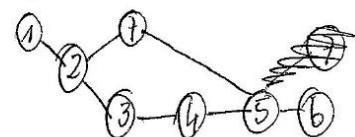
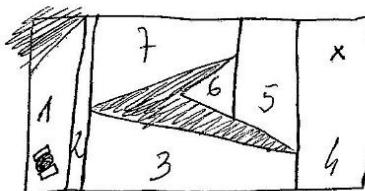
↳ Analogia a ~~planilha~~ planta de um piso  
segmentos de recta com medidas

Planeamento de caminhos → Decomposição celular

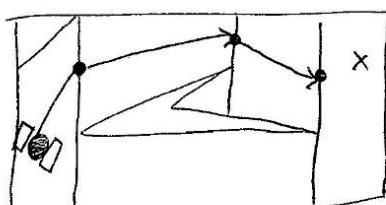
↳ Traçar segmentos de recta que passe nos vértices

↳ Grafo com células adjacentes

↳ Aplicar algoritmo ( $A^*$ , Dijkstra) para caminho



Exemplo: robô não vai da 1 a 5 directamente porque não são adjacentes

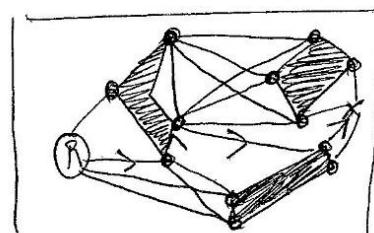


Passar pelo centro do segmento de recta que une 2 vértices

↳ ponto  $\oplus$  afastado dos 2

↓ ao mesmo nível → Grafo de visibilidade

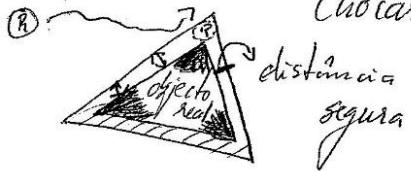
↳ Traçar arestas entre vértices visíveis



Aplicar  $A^*$ , Dijkstra

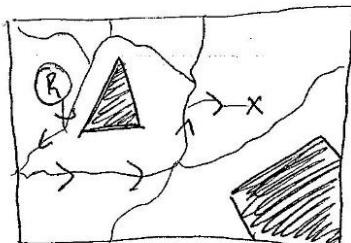
19

- Grafo de visibilidade → Caminho pode ser usado diretamente? Não, porque se passar pelo vértice pode chocar
  - ↳ cálculo de distância segura
  - ↳ Diâmetro do robô para poder mandar em segurança pelos vértices



## → Diagrama de Voronoi

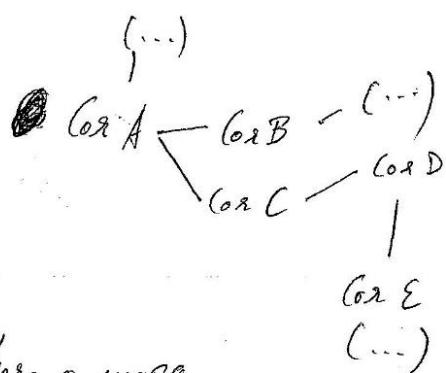
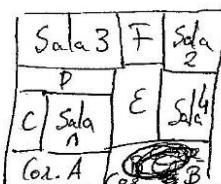
- ↳ Ideia é mesma → processar o mapa para traçar um caminho



cálculos matemáticos para traçar caminhos à volta dos obstáculos → como se fosse um mapa de estradas

## • Mapas topológicos

- ↳ Saber informação de conectividade
- ↳ Identificar regiões de interesse
- ↳ A sequência de navegação no robô também é, de certa forma, uma implementação desse método (embora o mapa não esteja em memória, é implícito)



2016/6/11

## Nikos UV T<sub>gama</sub>

- Deteta entre os 170 ~ 250 mm
- Com condensadores → pulsos extinguidos
  - ↳ ler porta → se valor  $\geq 1$  → fog.
- Com condensadores → ler número de pulsos na porta digital
  - ↳ Se pulsos  $\geq$  → fog.