Introdução e descrição do projeto

1 Introdução

Neste curso, desenvolveremos um carrinho elétrico controlado por uma placa embarcada que roda sistema Linux, usando visão computacional e aprendizagem de máquina.

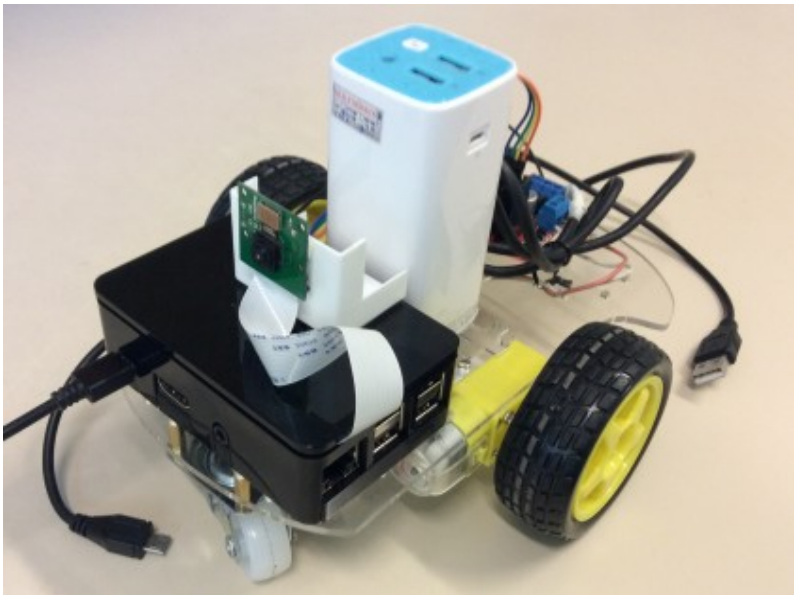


Figura 1: Carrinho elétrico com Raspberry, câmera, powerbank, e ponte-H. Diferentemente da foto, recomendo que montem o carrinho com bateria deitada, pois aumenta estabilidade e permite passar sob estruturas baixas.

No carrinho, haverá uma placa Raspberry com uma câmera. Raspberry (servidor) será responsável por adquirir vídeo, comprimi-lo e transmiti-lo para um computador (cliente). Além disso, Raspberry deve receber os comandos do computador e controlar os motores (através da interface GPIO e ponte-H) de acordo com os comandos recebidos.

No computador, haverá um programa que recebe o vídeo compactado da Raspberry, descompacta-o, interpreta-o e comanda o carrinho, pedindo para controlar as velocidades dos motores esquerdo/direito, para frente/trás. A interpretação do vídeo será feito de três formas, de dificuldades crescentes:

1. O usuário vê o vídeo remotamente e envia manualmente os comandos de controle apertando teclas.
2. Um programa procura uma certa placa nos quadros do vídeo e faz o carrinho seguir automaticamente essa placa.
3. Um programa procura uma placa nos quadros do vídeo e lê um dígito manuscrito escrito dentro da placa, e faz o carrinho dirigir-se à direção indicada pelo dígito.

Raspberry (servidor) e computador (cliente) estarão ligados através de protocolo TCP/IP a um roteador, via wifi.

Note que o foco deste projeto não é a mecânica do carrinho, que é muito simples. Este projeto aborda os seguintes aspectos de sistemas eletrônicos:

- Comunicação TCP/IP.
- Placa embarcada com sistema Linux.
- Processamento de imagens e visão computacional.
- Aprendizagem de máquina.
- Processamento paralelo.
- Máquina de estados finita.
- Prática de programação orientada a objetos.

2 Cuidados

a) Cada grupo deverá trabalhar com um carrinho elétrico, uma placa Raspberry Pi, um computador com Windows+BashWin[1] ou Linux, e um monitor/teclado/mouse. Você pode usar o seu material ou pegá-los emprestados do Departamento. Se você danificar algum material emprestado, deverá repor o material danificado por outro igual.

b) Sempre desligue a energia de Raspberry “de forma ordenada”, dando “shutdown” no ambiente gráfico ou “sudo poweroff” (ou "sudo shutdown –h now") na linha de comando. O cartão SD pode ficar corrompido se desligar Raspberry durante uma operação de escrita. Quando trabalhar remotamente (sem ter um monitor ligado diretamente a Raspberry), você não vai enxergar a tela do Raspberry após o comando "poweroff". Assim, espere o tempo suficiente para ter certeza de que Raspberry se desligou completamente antes de desligar a alimentação. Ao ligar Raspberry, ligue de uma vez, sem provocar liga-desliga intermitente.

3 Lista de materiais

3.1 Carrinho elétrico

Kit Chassi 2WD Robô para Arduino
Driver Motor Ponte H L298n para Arduino
Recarregador de celular powerbank 10Ah com duas saídas 5V 2A e 5V 1A.

3.2 Raspberry Pi

Raspberry Pi 3 modelo B
Cartão de memória sdxc classe 10 com pelo menos 16 GBytes
Caixa para Raspberry Pi
Fonte de alimentação 5V 2,5A (ou mais)
Câmera Raspberry versão 1, 5 Mpixels com cabo
Algum suporte para fixar câmera (por exemplo, cotovelo externo para canalera 40mm).

1 Estamos abreviando como BashWin o ambiente "Bash on Ubuntu on Windows", a camada de compatibilidade de Linux disponível em todas as versões de Windows 10, 64 bits.

3.3 Computador

Computador ou notebook com Windows 10 64 bits com BashWin ou com Linux 64 bits (Debian, Ubuntu ou Mint).

3.4 Outros

Teclado
Mouse
Monitor HDMI ou DVI
Cabo HDMI-HDMI ou HDMI-DVI (dependendo do monitor)
Conectores paralelos fêmea-fêmea
2 Cabos de energia USB
Roteador Wifi
Multímetro
Fita dupla face
Fita isolante
Soldador, solda, suporte para soldador
Chaves de fenda, alicates

4 Descrição do projeto

Vamos dividir este projeto em 6 fases:

4.1 Transmissão de vídeo pela Raspberry e transmissão de comandos manuais de controle pelo computador:

Neste fase, vamos criar uma arquitetura servidor-cliente TCP/IP local via roteador wifi.

Faça um programa servidor1.cpp para rodar na Raspberry e outro programa cliente1.cpp para rodar no computador.

Estando Raspberry e computador ligados no mesmo roteador, a forma de chamar os dois programas deve ser:

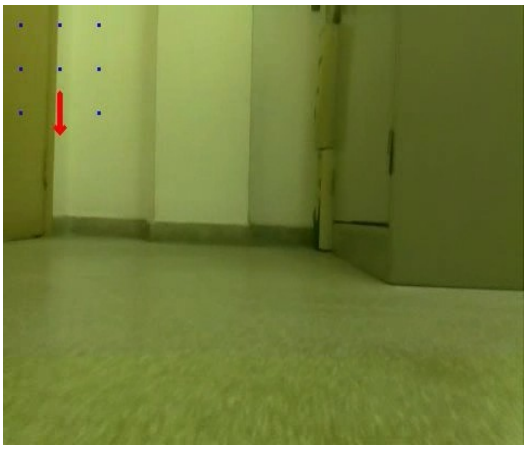
```
raspberrypi$ servidor1
computador$ cliente1 192.168.0.110
```

onde 192.168.0.110 (por exemplo) é o endereço IP da Raspberry. Servidor1 captura vídeo (colorido ou em níveis de cinza, 240x320 ou 360x480 ou 480x640) da câmera e o transmite em tempo real através do wifi do roteador para o computador. Use a compressão JPEG quadro a quadro para diminuir a quantidade de dados a serem transmitidos (sem compressão, a taxa quadros por segundo poderá ser baixa e o vídeo chegar com atraso). O computador pode enviar os seguintes comandos para Raspberry:

7 ou q = Virar à esquerda
8 ou w = Ir para frente
9 ou e = Virar à direita
5 ou s = Parar
1 ou z = Virar à esquerda dando ré
2 ou x = Dar ré
3 ou c = Virar à direita dando ré
ESC = Parar o programa.

Opcionalmente, você pode implementar também:
4 ou a = Virar acentuadamente à esquerda
6 ou d = Virar acentuadamente à direita

(a) Virar à esquerda.



(b) Dar ré.

Figura 2: Indicação no quadro do comando recebido.

A direção a seguir deve ser indicada de alguma forma na imagem pelo programa servidor1 antes de ser transmitida ao cliente1, para mostrar que recebeu corretamente o comando. Por exemplo, na figura 2a, foi selecionada "virar à esquerda". Na figura 2b, foi selecionada "dar ré".

4.2 Controle remoto manual do carrinho com transmissão de vídeo:

Neste fase, vamos controlar manualmente o carrinho à distância, onde o usuário vai enxergar no computador o vídeo capturado pela raspberry, e vai comandar manualmente os motores do carrinho apertando as teclas do computador especificadas no item anterior.

Modifique o programa servidor1.cpp para criar o programa servidor2.cpp. O programa servidor2.cpp deve efetivamente controlar os dois motores do carrinho (em vez de apenas mostrar o estado virtual dos motores no vídeo).

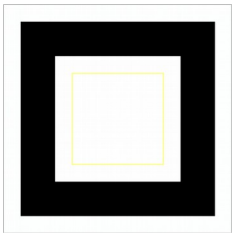
Se necessário, modifique cliente1.cpp para cliente2.cpp. O vídeo abaixo mostra o controle manual do carrinho:

`http://www.lps.usp.br/hae/apostilaraspi/controlemanual.avi`

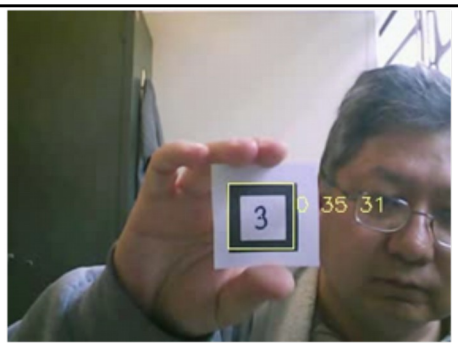
4.3 Localização da placa:

Faça um programa fase3.cpp que lê um vídeo capturado.avi, um modelo de placa quadrado.png e localiza a placa-modelo nos seus quadros, toda vez que a placa estiver entre aproximadamente 30 e 150 cm da câmera, gerando o vídeo localiza.avi (figura 1).

Quadro de câmera capturado (capturado.avi).



Placa quadrado.png.



Vídeo com placa localizada (localiza.avi).

Figura 1: Localização da placa.

O vídeo-teste (capturado.avi) e a placa (quadrado.png) estão em:

http://www.lps.usp.br/hae/apostilaraspi/capturado.avi
http://www.lps.usp.br/hae/apostilaraspi/quadrado.png

Executando o seu programa compilado:

```
$ fase3 capturado.avi quadrado.png seu_localiza.avi
```

deverá ler o vídeo capturado.avi, a imagem quadrado.png e gerar o vídeo seu_localiza.avi. Um exemplo de saída está em:

http://www.lps.usp.br/hae/apostilaraspi/localiza.avi

No interior da placa quadrado.png, há um quadrado menor amarelo pouco visível. Dentro do quadrado amarelo, haverá dígitos manuscritos. Durante a localização da placa, você deve considerar o conteúdo no interior do quadrado amarelo como "don't care", para que o manuscrito não atrapalhe a busca do modelo. Todos os pixels brancos dentro do quadrado amarelo possuem valores (255, 255, 255), enquanto que os pixels brancos fora do quadrado amarelo possuem valores (255, 255, 254) ou (255, 254, 255) - em ordem (b, g, r).

4.4 Fazer o carrinho seguir automaticamente a placa:

Modifique o programa da fase 3 para controlar o carrinho de forma que ele siga continuamente a placa quadrado.png. O carrinho deve parar nas duas situações:

a) Quando não conseguir localizar a placa no quadro do vídeo capturado.
b) Quando a câmera estiver muito próximo da placa, para evitar uma colisão.

Este programa pode rodar no modo servidor-cliente (servidor4.cpp e cliente4.cpp), onde o processamento mais pesado (a localização da placa) pode ser realizado no computador. Se você quiser (depois de constatar que é possível), todo o processamento pode ser feito no Raspberry.

O vídeo abaixo mostra o carrinho seguindo a placa:

http://www.lps.usp.br/hae/apostilaraspi/segueplaca.avi

4.5 Reconhecimento do dígito manuscrito:

Modifique o programa fase3.cpp para que, além de detectar a placa, leia o dígito manuscrito inscrito dentro da placa quando a placa estiver suficientemente próximo da câmera. Fase5.cpp deve gravar o vídeo com a localização da placa e o dígito reconhecido.

Quadro de câmera capturado (capturado.avi).

Vídeo com placa localizada e dígito reconhecido (locarec.avi).

Figura 1: Localização da placa com reconhecimento do dígito manuscrito.

Note que, no interior da placa quadrado.png, há um quadrado menor amarelo pouco visível. Os dígitos manuscritos estão sempre dentro do quadrado amarelo. Durante a localização da placa, você deve considerar o conteúdo no interior do quadrado amarelo como "don't care".

Nota: Pode usar a base de dados de dígitos manuscritos MNIST para treinar o seu sistema de reconhecimento:

http://www.lps.usp.br/hae/apostilaraspi/mnist.zip

Site original: http://yann.lecun.com/exdb/mnist/

Nota: Deixe todos os arquivos de entrada necessários no seu diretório default. Os arquivos necessários são:

```
capturado.avi (http://www.lps.usp.br/hae/apostilaraspi/capturado.avi)
quadrado.png (http://www.lps.usp.br/hae/apostilaraspi/quadrado.png)
t10k-images.idx3-ubyte
t10k-labels.idx1-ubyte
train-images.idx3-ubyte
train-labels.idx1-ubyte
```

Os quatro últimos arquivos são obtidos descompactando mnist.zip. Use obrigatoriamente os nomes dos arquivos acima. Executando:

```
$ fase5
```

o seu programa deve ler os arquivos necessários do diretório default e gerar o arquivo `locarec.avi` também no diretório default. Um exemplo de saída está em:

http://www.lps.usp.br/hae/apostilaraspi/locarec.avi

4.6 Navegação autônoma com auxílio das placas:

Faça o sistema cliente-servidor cliente6.cpp e servidor6.cpp, onde o carrinho será controlado pelas placas penduradas em pequenos "semáforos". O carrinho deve seguir automaticamente o caminho indicado pelos dígitos manuscritos nas placas, com a convenção (em 2017):

0: Pare o carrinho.

1: Pare o carrinho.

2: Vire 180 graus à esquerda imediatamente.

3: Vire 180 graus à esquerda imediatamente.

4: Vire 180 graus à direita imediatamente.
5: Vire 180 graus à direita imediatamente.
6: Vire 90 graus à esquerda imediatamente.
7: Vire 90 graus à esquerda imediatamente.
8: Vire 90 graus à direita imediatamente.
9: Vire 90 graus à direita imediatamente.

O programa deve dirigir o carrinho autonomamente de acordo com as indicações das placas. Por exemplo:

```
raspberrypi$ servidor6
computador$ cliente6 192.168.0.110
```

Nota: Se algum grupo quiser, pode usar algum outro problema de reconhecimento de imagens por aprendizagem de máquina para controlar o carrinho. Por exemplo, pode usar reconhecimento de faces ou reconhecimento de objetos. Converse com os professores sobre a sua ideia antes de implementar.

A demonstração final consistirá de 3 sub-demonstrações, referentes às fases 2, 4 e 6 do projeto. Em cada sub-demonstração, terão 3 tentativas para completar a tarefa. Se completar uma vez, está valendo.

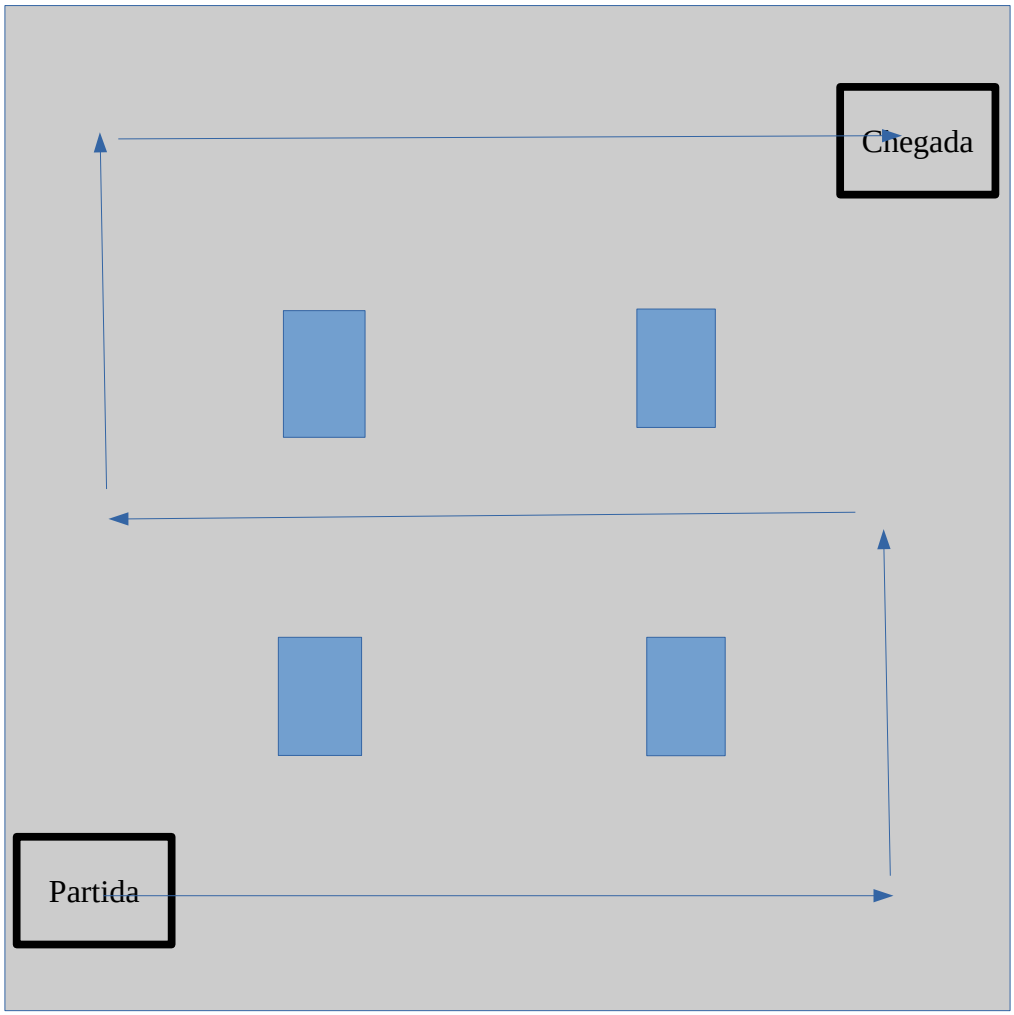


Figura 1: Sub-demonstrações das fases 2 e 4.

Nas sub-demonstrações das fases 2 e 4, o carrinho deve completar o percurso da figura 1.

Na fase 2, os alunos controlam o carrinho remotamente, visualizando no monitor a visão da câmera do Raspberry. Os alunos não terão visão direta do carrinho.

Na fase 4, os alunos guiam o carrinho mostrando a placa para a câmera do Raspberry.

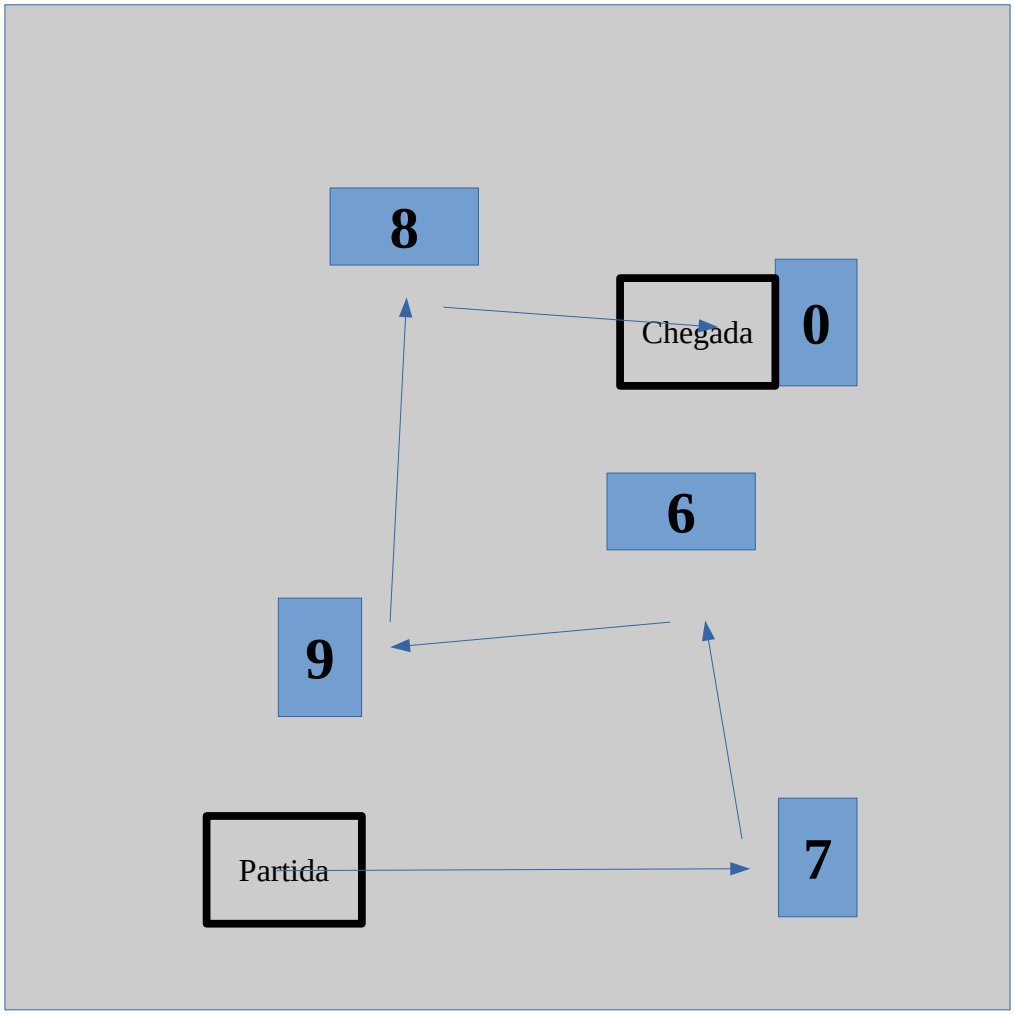


Figura 2: Sub-demonstração da fase 6.

Na sub-demonstração da fase 6, o carrinho deve completar sozinho a trajetória da figura 2. Nas 5 caixas estarão afixadas placas com manuscritos. A trajetória do carrinho, entre uma placa e outra, será de aproximadamente 90 cm.

Por fim, sem valer nota, podem testar a seguinte trajetória em que o carrinho vai e volta indefinidamente.

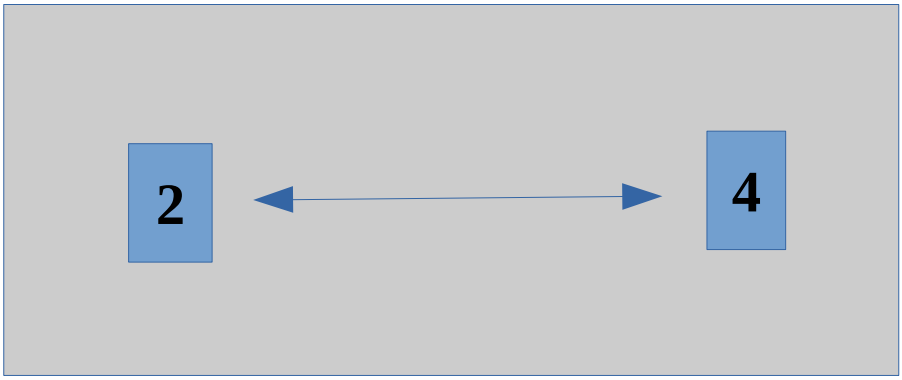


Figura 3: Carrinho vai e volta indefinidamente.