

Introdução à Experiência 2

PSI3422 - Laboratório de Sistemas Eletrônicos (2025)

Este curso tem 2 experiências. A primeira experiência foi dada pelo prof. Gustavo e a segunda experiência será dada por mim (Hae).

As informações da 2ª experiência estão nos seguintes sites:

- <https://edisciplinas.usp.br/course/view.php?id=132219> (edisciplina do curso 2025)
- <http://www.lps.usp.br/hae/psi3422/index.html> (site da 2ª experiência)
- <http://www.lps.usp.br/hae/apostilaraspi/index.html> (apostilas sobre 2ª experiência)

Os sistemas embarcados podem ser classificados usando diferentes critérios. Uma possível classificação é agrupar em “microcontroladores dedicados” e “computadores completos numa única placa”:

1) Microcontroladores dedicados “parecidos com Arduino/Freedom”: Quando um sistema deste tipo é ligado, ele começa a executar um programa pré-carregado na memória. A edição de programa e compilação são feitas no computador e o programa compilado é carregado na placa. Há interface elétrica com o mundo exterior (GPIO). É o que vocês estudaram na 1ª experiência com prof. Gustavo.

2) Computadores completos numa única placa (parecidos com Raspberry Pi, Odroid, Orange Pi, etc.): Consiste em um computador completo numa única placa que normalmente roda Linux. Como qualquer computador, é possível ligar neste tipo de dispositivo monitor, teclado, mouse, etc. A programação é feita normalmente na própria placa, pois é possível executar editor de texto, compilador e/ou interpretador na própria placa. Diferentemente da maioria dos computadores, as placas deste tipo possuem interface elétrica com o mundo exterior (GPIO). É o que vocês vão estudar na 2ª experiência comigo.

O meu “sonho de criança” era ter um carrinho ou um barco controlado remotamente, podendo ver ao vivo numa tela a visão que teria de dentro do veículo. Esta disciplina permitiu realizar o meu “sonho de criança”.

[controlemanual3.mp4](#)

Vai e volta (fase 6): [vai e volta celular.mp4](#)

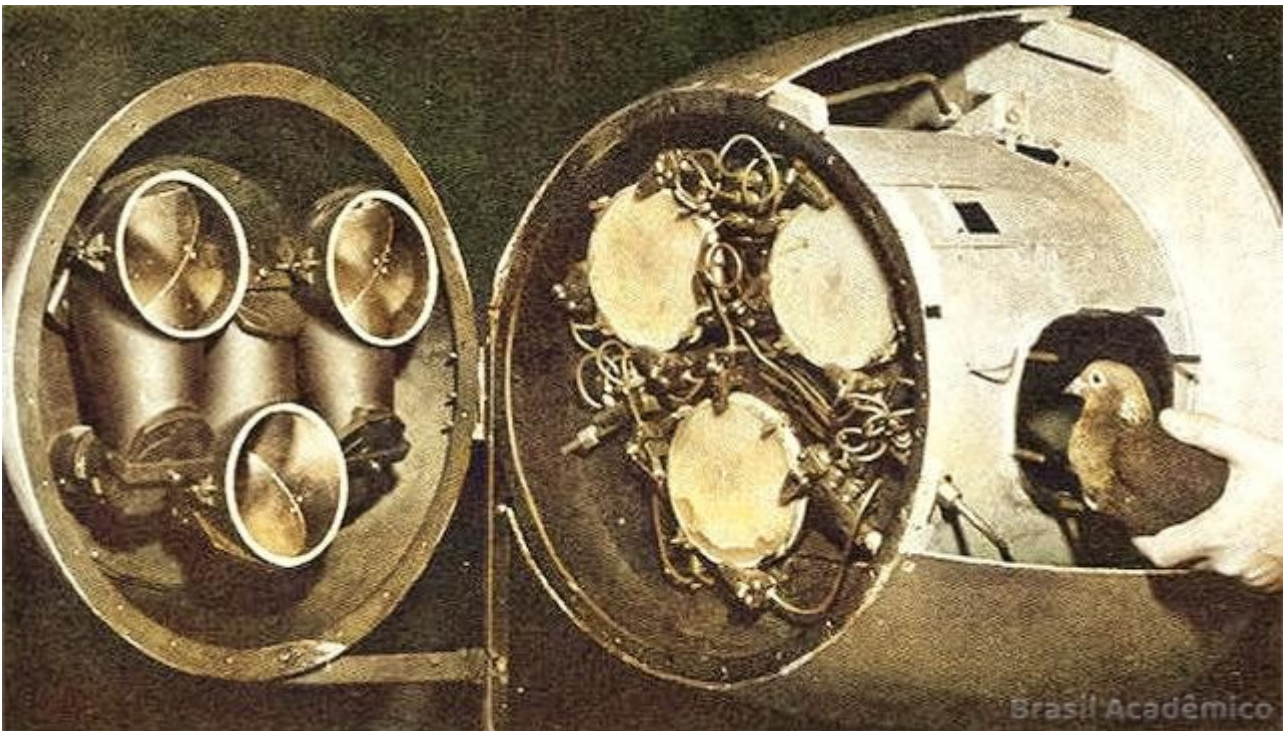
Controle automático/manual (fase 7): [fase7 raspberry.avi](#)

Um documentário sobre a 2ª Guerra Mundial mostra militares americanos treinando pombos para que fiquem bicando uma tela de vidro através do qual os pombos podiam ver o navio-alvo:

<https://www.youtube.com/shorts/3m2JEDj1cw> (50s)

<https://www.youtube.com/watch?v=JY-RPBqkYCK> (4min iniciais do vídeo)

A ideia era colocar esses pombos treinados dentro de mísseis ou torpedos para que eles pudessem guiar o armamento ao navio inimigo, através das bicadas com bico coberto por uma peça metálica, para fazer contato elétrico. Parece que esta ideia não chegou a ser colocada em prática. Naquele tempo, não havia sistema embarcada, câmera digital, nem havia técnicas de visão computacional que pudesse guiar o míssil/torpedo ao alvo.



Hoje, podemos fazer o que era impossível naquele tempo. Vamos fazer algo bem parecido usando Raspberry.

[segueplaca2.mp4](#)

[segueplaca celular.mp4](#)

Se pesquisar “drone Ucrânia” na Internet, aparecem várias notícias sobre drones aéreos ou marítimos que Ucrânia e Rússia estão usando na guerra, como as figuras abaixo. O nosso projeto tem semelhança com esses dispositivos.

Drones navais da Ucrânia



Fonte: Rybar



Alguns vídeos dos alunos de 2019.

- [2019_erik_isabela_celular.mp4](#), [2019_erik_isabela_raspberry.mp4](#).
- [2019_gustavo_lucas_celular.mp4](#), [2019_gustavo_lucas_raspberry.mp4](#).
- [2019_matheus_thomas_celular.mp4](#), [2019_matheus_thomas_raspberry.mp4](#).
- [2019_matias_caio_celular.mp4](#), [2019_matias_caio_raspberry.mp4](#).
- [2019_rubens_thomas_celular.mp4](#), [2019_rubens_thomas_raspberry.mp4](#).

Fases 1 e 2: No início da 2ª experiência, vocês vão controlar remotamente o carrinho de forma manual, tendo a visão da câmera do carrinho no monitor do seu computador. Para fazer esta fase, vão praticar:

- Placa embarcada Raspberry com sistema Linux.
- Comunicação TCP/IP (para transmitir vídeo do carrinho para computador).
- Compressão de imagem (pois os quadros precisam ser transmitidos compactados).
- Prática de programação orientada a objetos em C++.

Fases 3 e 4: Depois, em vez de um ser humano controlá-lo, o carrinho vai se dirigir automaticamente em direção a uma placa. O processamento é feito no computador, pois possui maior poder de processamento. Vão praticar:

- Visão computacional (template matching para detectar placa no vídeo).
- Processamento paralelo (para acelerar a detecção da placa).

Nota: Com template matching, o reconhecimento da placa é suficientemente bom. Porém, para melhorar ainda mais, seria possível usar deep learning. Só que, para isso, precisaria criar um conjunto de vídeos para treinar o modelo.

Fases 5 e 6: Depois, o carrinho vai se navegar autonomamente. O carrinho lê os dígitos escritos nas placas e efetua movimento de acordo.

- 0: Pare o carrinho.
- 1: Pare o carrinho.
- 2: Vire 180 graus à esquerda imediatamente.
- 3: Vire 180 graus à direita imediatamente.
- 4: Passe por baixo da placa e continue em frente.
- 5: Passe por baixo da placa e continue em frente.
- 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.

Para fazer esta fase funcionar, vocês vão praticar:

- Aprendizado de máquina (leitura do dígito manuscrito na placa).
- Máquina de estados finita (controlar a navegação autônoma do carrinho).

Fases 7: Finalmente, faremos com que o usuário possa escolher entre controlar o carrinho manual ou automaticamente.

Nota: Na 2ª experiência, precisamos que um computador com sistema Linux se comunique com Raspberry que também roda Linux. Para isso, cada grupo precisa ter um computador Linux. Os computadores do laboratório possuem sistema Linux. É possível o aluno trabalhar no seu próprio computador/notebook que só possui Windows. Como dá trabalho instalar uma partição do sistema Linux no computador Windows (além do perigo de perder todo o conteúdo do disco), Maurício Perez achou uma ótima alternativa: Instalar o sistema Linux no “VirtualBox” dentro do sistema Windows. Assim, não precisa criar uma partição Linux. Assim, vocês podem continuar o trabalho do laboratório em casa, se necessário. Uma outra alternativa para rodar Linux dentro de Windows é usar WSL (Windows Subsystem Linux).

Nota: Além do kit que receberam e das ferramentas comuns como alicate, chave de fenda, etc, precisa dispor:

- Um computador (notebook ou desktop) com Windows 64 bits. Se o computador tiver sistema Linux Mint ou Ubuntu, melhor ainda.
- Um monitor ou televisão com entrada HDMI, para servir de monitor de Raspberry. Depois que tudo estiver funcionando, será possível enxergar a saída de Raspberry remotamente na tela do computador. Porém, no início (e também quando der algum problema que impeça controlar Raspberry do computador), é

recomendável ter um monitor/televisão. Parece que no site do Raspberry ensina como configurá-lo sem precisar de monitor.

- Um teclado e um mouse USB para usar no Raspberry. Depois que tudo estiver funcionando, não vai precisar mais, pois será possível controlar Raspberry remotamente do computador. Porém, no início (e também quando der algum problema que impeça controlar Raspberry do computador), precisa de teclado e mouse.
- Um cabo HDMI para ligar Raspberry com monitor/televisão.
- Um roteador de wifi com acesso à internet para ligar Raspberry à internet e conectar Raspberry com computador.

[Aula 0 - introdutória. Fim.]

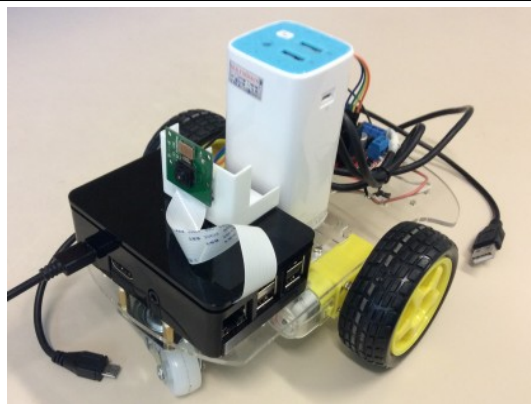
[Aula 1 parte 1. Início.]

Descrição do projeto

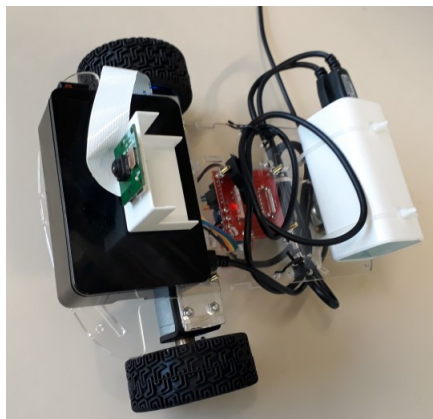
Resumo: No segundo experimento, desenvolveremos um carrinho elétrico controlado por uma placa embarcada conectada a um computador via TCP/IP sem fio (wifi). No carrinho, a placa embarcada irá adquirir a imagem vista pela câmera, comprimi-la e transmiti-la para o computador. No computador, haverá um programa que recebe a imagem compactada, descompacta-a, procura nela uma placa, faz o carrinho se mover em direção à placa, lê um comando escrito na placa e faz o carrinho se mover de acordo com o comando. Este projeto aborda os seguintes aspectos de Eletrônica e Sistemas Computacionais: comunicação TCP/IP; placa embarcada com sistema Linux; processamento de imagens; aprendizagem de máquina; processamento paralelo; máquina de estados finita; e programação orientada a objetos.

1 Introdução

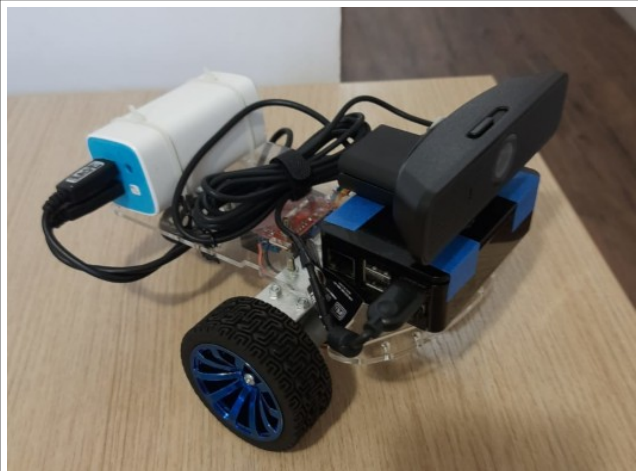
No segundo experimento, desenvolveremos um carrinho elétrico controlado por uma placa embarcada (Raspberry Pi) que roda sistema Linux, usando visão computacional e aprendizagem de máquina.



(a)



(b)



(c)

Figura 1: Carrinho elétrico com Raspberry, câmera, powerbank e ponte-H. (a) Primeira montagem usando motores amarelos de alta rotação – o carrinho andava rápido demais e o posicionamento do powerbank tornava carrinho instável. (b) Segunda montagem usando motores de baixa rotação e alto torque – powerbank baixo deixa o carrinho mais estável. (c) Terceira montagem usando webcam Logitech C925e.

No carrinho, haverá uma placa Raspberry com uma câmera (Câmera Raspberry ou alguma webcam compatível). 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, controlando as velocidades dos motores esquerdo/direito, avante/reverso. 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, clicando mouse num teclado virtual [https://drive.google.com/file/d/1IjAcvsbrXhrS5_0I95E4_MWPAtdKxAC9/view?usp=sharing].
2. Um programa procura uma certa placa nos quadros do vídeo e faz o carrinho seguir automaticamente essa placa [<https://drive.google.com/file/d/14NZWYULSyNik5utFIGm9YkUQTZz5N-YS/view?usp=sharing>].
3. Um programa procura uma placa nos quadros do vídeo, lê um dígito manuscrito dentro da placa e faz o carrinho obedecer o comando indicado pelo dígito.
[<https://drive.google.com/file/d/1OflWzi2F4nitedi8lvVVZVY796gQQkxM/view?usp=sharing>].

Raspberry (servidor) e computador (cliente) estarão ligados através de protocolo TCP/IP a um roteador, via wifi. O foco deste projeto não é a mecânica do carrinho, que é muito simples. Este projeto aborda os seguintes aspectos de Eletrônica e Sistemas Computacionais:

- 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

Cada grupo deverá trabalhar com um carrinho elétrico, uma placa Raspberry Pi, um computador com Linux (Ubuntu ou Mint), 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.

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 de pelo menos 2A e 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 ou 2) com cabo ou uma webcam compatível com Raspberry Pi (como webcam Logitech C925e).
- Algum suporte para fixar câmera de Raspberry (por exemplo, cotovelo externo para canalera 40mm).

3.3 Computador

Computador ou notebook com Linux 64 bits (Ubuntu ou Mint) ligado à internet. Pode ser computador com Windows - neste caso Linux Mint pode ser instalado através de VirtualBox.

3.4 Outros

- Teclado
- Mouse
- Monitor ou televisão com entrada HDMI ou DVI
- Cabo HDMI-HDMI ou HDMI-DVI (dependendo do monitor)
- Conectores paralelos fêmea-fêmea
- 2 Cabos de energia USB
- Roteador Wifi com acesso à internet
- Multímetro
- Fita dupla face
- Fita isolante
- Chaves de fenda, alicates, etc.

[Aula 1 parte 1. Fim.]

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, 240×320 ou 360×480 ou 480×640) 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 (se não comprimir, 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

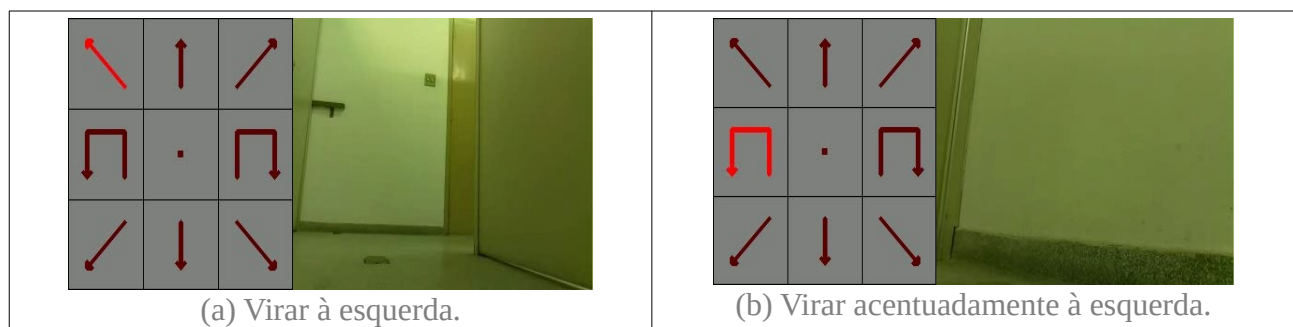


Figura 2: Indicação no quadro do comando recebido.

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 “virar acentuadamente à esquerda”.

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 virtuais 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.

[<http://www.lps.usp.br/hae/apostilaraspi/controlemanual3.mp4>]

4.3 Localização da placa:

Faça um programa fase3.cpp que lê o 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).

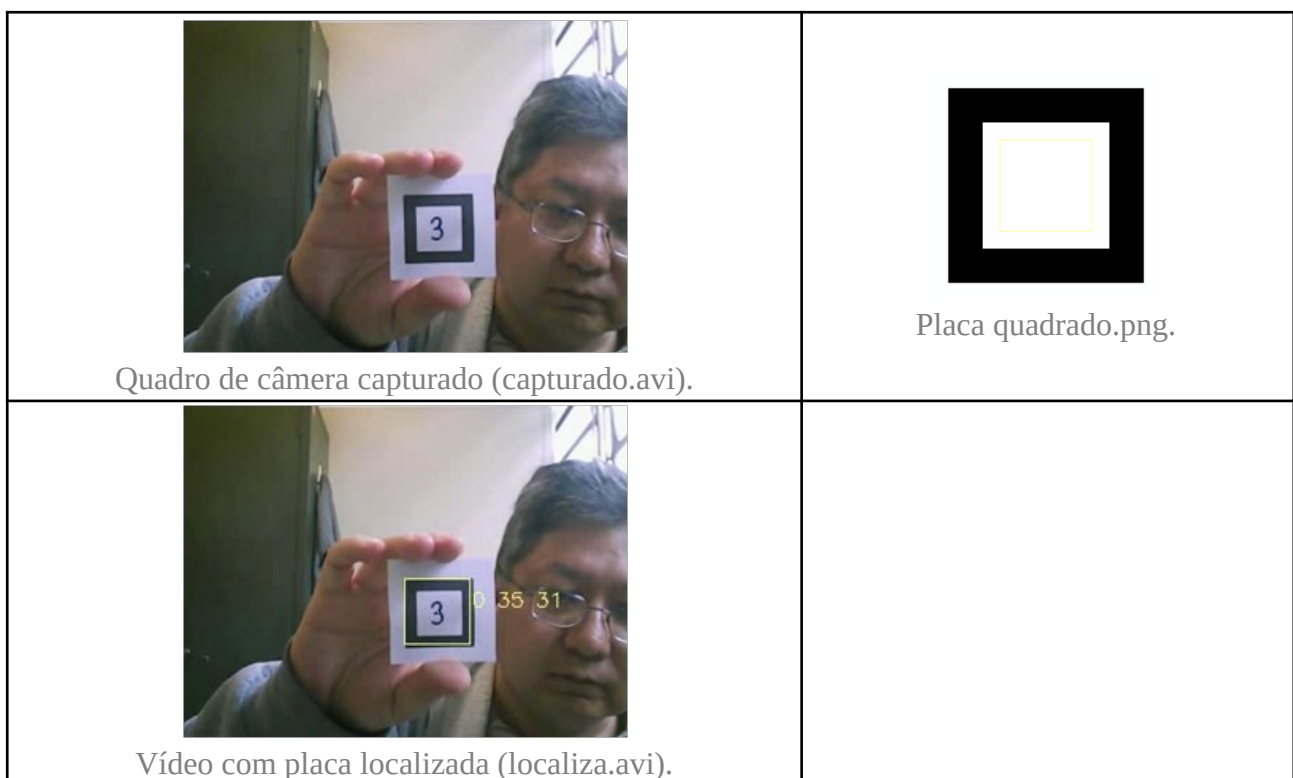


Figura 1: Localização da placa.

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>]

O seu programa deve localizar placa independentemente do dígito escrito no seu interior. Para isso, no interior da placa *quadrado.png* há um quadrado menor amarelo pouco visível. Dentro do quadrado amarelo será escrito um dígito. 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).

A localização da placa foi feita até agora (2023) usando template matching. Porém, provavelmente, usando deep learning conseguiremos obter uma localização com menos erros. Para isso, precisamos de vídeos-exemplos para poder treinar a CNN. Vou pedir que vocês gravem os vídeos vistos do carrinho e entreguem para mim, para criar exemplos de treinamento. Provavelmente, este ano não será possível usar CNN para localizar a placa. Mas será usada nos próximos anos.

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:

- Quando não conseguir localizar a placa no quadro do vídeo capturado.
- 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) deve ser realizado no computador. O vídeo abaixo mostra o carrinho seguindo a placa:

[<http://www.lps.usp.br/hae/apostilaraspi/segueplaca2.mp4>]

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.



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:

- 0: Pare o carrinho.
- 1: Pare o carrinho.
- 2: Vire 180 graus à esquerda imediatamente.
- 3: Vire 180 graus à direita imediatamente.
- 4: Passe por baixo da placa e continue em frente.
- 5: Passe por baixo da placa e continue em frente.
- 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.

No site:

<http://www.lps.usp.br/hae/apostilaraspi/index.html>

há vários exemplos de projetos de outros anos.

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

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