

[Aula 1 parte 3. Início]

Nota 06/10/2025: Para este ano (2025) eu tinha planejado não usar Cekeikon para nada. Reparei que não é possível fazer isso na sala GD-06, pois os computadores dessa sala não possuem OpenCV instalado, apenas Cekeikon (que vem com OpenCV v3 dentro). Além disso, os alunos não conseguem instalar OpenCV no computador da sala pois não possuem a senha de administrador. Assim, estou alterando esta apostila para que tanto o computador quanto a Raspberry usem Cekeikon. Com isso, também melhora a compatibilidade das bibliotecas OpenCV de computador e de Raspberry, pois ambas serão v3.

OpenCV, Video4Linux e WiringPi

Resumo: Hoje instalaremos C++, OpenCV e Video4Linux no Raspberry. Depois, testaremos o funcionamento dessas bibliotecas. Além disso, montaremos fisicamente o carrinho.

Nota 2025: Este ano, vamos copiar os códigos-fontes das funções e classes de Cekeikon que formos usar no projeto dentro do arquivo a ser incluído *raspberry.hpp* (a ser usado tanto no computador como no Raspberry):

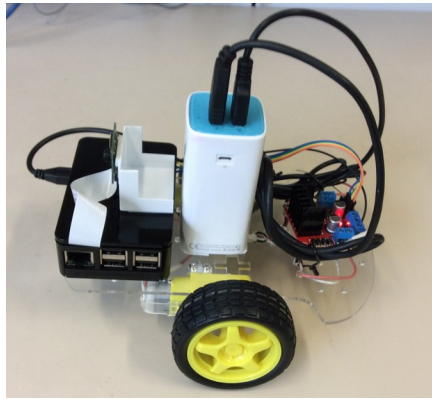
<http://www.lps.usp.br/hae/apostilaraspi/raspberry.hpp>

Isso permitirá que os alunos entendam o que está acontecendo dentro dessas funções.

Nota: Nesta aula, é desejável dispor de um osciloscópio, um multímetro ou uma LED (em série com um resistor de valor adequado - se LED for de 20 mA use em série um resistor de 270Ω) para visualizar as tensões de saída do Raspberry.

Nota: Durante esta aula, montem o carrinho mecanicamente, fixando Raspberry, câmera, ponte-H, etc. na carroceria do carrinho. Sugiro que montem a bateria (powerbank) deitada, pois o carrinho ficará mais estável (figuras 1b e 1c). A caixa do Raspberry deve ser montada de forma que todos os conectores fiquem acessíveis. Sempre trabalhem com Raspberry dentro da sua caixa, pois isto protege-o mecanica e eletricamente. A qualidade da montagem será levada em conta ao atribuir a nota desta aula e da apresentação final.

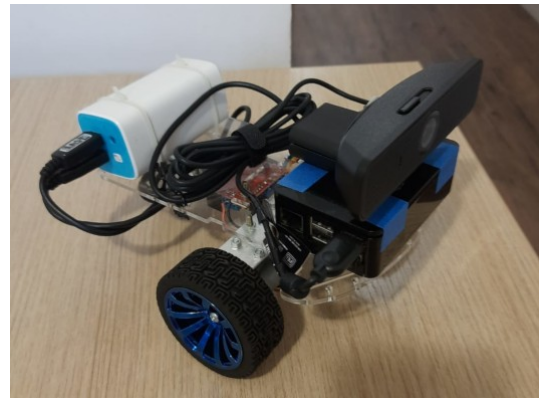
Nota: Tipicamente, demora bastante para carregar o powerbank. Os powerbanks entregues possuem capacidade 20000 mAh (o triplo de um celular típico de 6000 mAh). Carregá-lo pode demorar o triplo do tempo de carga de um celular.



(a)



(b)



(c)

Figura 1: (a) Carrinho usado nos anos 2017-2018. (b) Carrinho de 2019 em diante com novos motores de maior torque e menor rpm usando raspicam. (c) Carrinho com webcam Logitech C925e.

1 Introdução

Iremos desenvolver o nosso projeto em C/C++. Os testes independentes mostram que C++ é algo como 10 a 100 vezes mais rápido do que Python e usa menos memória:

<https://benchmarksgame-team.pages.debian.net/benchmarksgame/fastest/gpp-python3.html>
<https://towardsdatascience.com/how-fast-is-c-compared-to-python-978f18f474c7>
<https://www.freecodecamp.org/news/python-vs-c-plus-plus-time-complexity-analysis/>

Ou seja, não é possível desenvolver sistemas em tempo real “sérios” em Python.

Nota: Um programa em Python e o equivalente em C/C++ até rodam com velocidades semelhantes se a maior parte do tempo é gasto dentro de funções escritas numa outra linguagem (como no TensorFlow ou PyTorch). Porém, há diferença substancial no tempo de execução se o programa gastar boa parte do tempo dentro de laços escritos na própria linguagem.

Além disso, as funções de baixo nível de Linux estão escritas para serem chamadas de C/C++, e não de Python.

2 C++ e OpenCV

Instale Cekeikon no seu computador e no Raspberry seguindo os passos descritos em:

<http://www.lps.usp.br/hae/software/cekeikon56.html>

Nota: Em 2025, Cekeikon já vem instalada no Raspberry. Cekeikon também já está instalada nos computadores da sala GD-06. Assm, só precisa instalar no seu notebook (para fazer os exercícios em casa).

Nota: Não vamos usar as funções do Cekeikon. Vamos usar apesar a biblioteca *OpenCV3* (incluído no Cekeikon) no computador e o programa *compila* no computador e no Raspberry.

Nota: No Raspberry, mantenha a seguinte linha no `.bashrc`:

```
source ~/cekeikon5/bin/inst-cek5-raspberry.sh
```

Nota: Na sala GD-06, é necessário executar no computador o comando:

```
$ source ~/cekeikon5/bin/ativa_cekcpu
```

no início da sessão para ativar Cekeikon.

Para verificar se Cekeikon está funcionando no seu computador ou raspberry, digite:

```
$ compila
```

Se chamar corretamente o programa “compila” é por que Cekeikon está funcionando.

Com Cekeikon instalado no computador, você tem as seguintes opções principais de compilação. É possível linkar com OpenCV versões 2 ou 3, com ou sem Cekeikon. **Em 2025, vamos trabalhar com OpenCV3, sem linkar com a biblioteca Cekeikon. Os comandos para compilar e linkar com OpenCV3 são:**

No computador:

comp\$ compila prog -ocv => Compila prog.cpp e linka com OpenCV2 (sem Cekeikon).

Computador\$ compila prog -ocv -v3 => Compila prog.cpp e linka com OpenCV3 (sem Cekeikon).

comp\$ compila prog -cek => Compila prog.cpp e linka com OpenCV2 e Cekeikon.

comp\$ compila prog -cek -v3 => Compila prog.cpp e linka com OpenCV3 e Cekeikon.

No Raspberry, não há opção “-v3”, pois sempre irá linkar com OpenCV3.

Raspberry\$ compila prog -ocv => Compila prog.cpp e linka com OpenCV3 (sem Cekeikon).

Rasp\$ compila prog -cek => Compila prog.cpp e linka com OpenCV3 e Cekeikon.

Nota: Se você não quer ficar escrevendo ./ toda hora, faça a alteração sugerida na apostila “raspberry.odt”: acrescente a linha:

```
export PATH=".:$PATH"
```

no arquivo “~/bashrc”.

Nota: Nesta aula, trabalharemos exclusivamente com Raspberry (sem usar computador) até chegar ao último item, quando voltaremos a trabalhar com Raspberry controlado por computador.

3 V4L (Video4Linux)

Muitos modelos de webcam são incompatíveis com Raspberry 3. Assim, vamos usar a câmera própria para Raspberry (versão 1 ou 2) ou algum webcam USB que seja compatível com Raspberry 3, como webcam Logitech C925e.

Em 2025, todos os grupos devem ter recebido webcam Logitech C925e.

1) Se você recebeu câmera Raspberry, para instalá-la **primeiro desligue Raspberry**. Depois, abra a porta pressionando os lados (figura 3a). Insira o cabo da câmera com o lado azul voltado para o conector ethernet (figura 3b). Depois, pressione a tampa da porta para prender o cabo, ao mesmo tempo em que mantém o cabo dentro do conector com a outra mão (figura 3c). Puxe levemente o cabo para verificar se está fixada firmemente. Para instruções mais detalhadas, visite o site abaixo.

[<https://www.dexterindustries.com/howto/installing-the-raspberry-pi-camera/>]

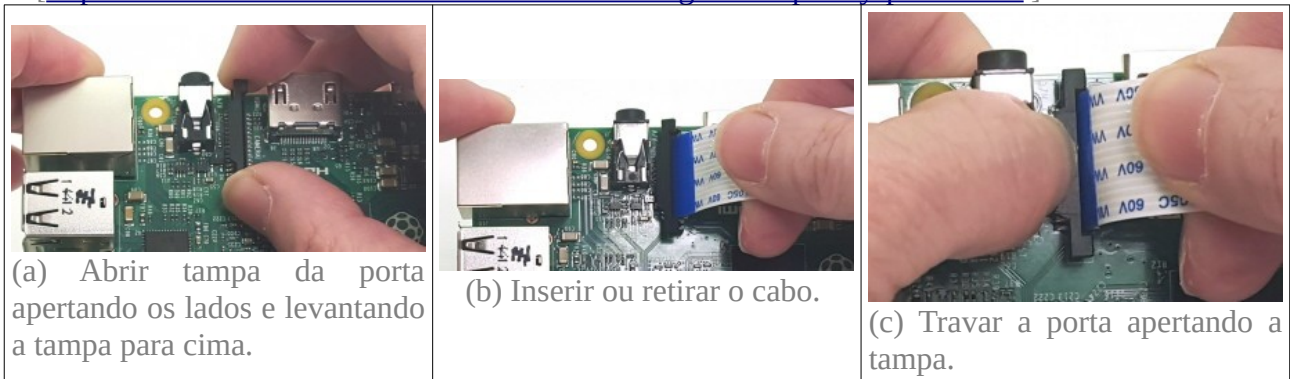


Figura3: Como inserir e retirar cabo da câmera Raspberry.

Para retirar a câmera, **primeiro desligue o Raspberry**. Depois, pressione os lados da porta e puxe o cabo sem fazer força – não puxe cabo sem pressionar os lados ou com força que isso irá quebrar o cabo. Se a caixa do Raspberry impedir apertar os lados da porta, use alguma ferramenta.

Nota: O comando abaixo só funciona se a sua câmera for Raspberry (não funciona se a sua câmera for webcam).

Após montar fisicamente a câmera, teste-a capturando uma imagem

[<https://www.filipeflop.com/blog/modulo-camera-raspberry-pi/>]:

```
raspberrypi>raspistill -o img_teste.png
```

Depois, verifique se *img_teste.png* foi capturada corretamente.

2) Se você recebeu webcam USB, é só conectar a câmera numa porta USB.

Para acessar a sua câmera (Raspberry ou webcam) de OpenCV/C++:

1) Verifique se a câmera está habilitada:

início → preferences → Raspberry Pi Configuration → Camera=enable.

2) Carregue driver V4L2:

```
sudo modprobe bcm2835-v4l2
```

[<https://www.raspberrypi.org/forums/viewtopic.php?t=126358>]

Cuidado: o caracter no meio de 4l2 não é o dígito “um”, mas a letra “le”.

3) Se o comando acima não funcionou, instale primeiro v4l-utils:

```
$ sudo apt-get update
```

```
$ sudo apt-get install v4l-utils
```

[<https://medium.com/@petehouston/install-v4l-utils-on-debian-based-distros-d4f5c2f4dc61>]

E teste novamente “sudo modprobe bcm2835-v4l2”.

O programa abaixo captura os quadros da câmera (tanto câmera Raspberry como webcam USB) e mostra-os numa janela como um vídeo:

```
//raspcam2.cpp
//compila raspcam_ocv -ocv (usando "compila" do Cekeikon) OU
//g++ raspcam_ocv.cpp -o raspcam_ocv `pkg-config opencv --libs --cflags` -O3 -s
#include <opencv2/opencv.hpp>
using namespace std;
using namespace cv;
int main() {
    VideoCapture w(0);
    if (!w.isOpened()) {
        fprintf(stderr, "Erro: Abertura de webcam 0.");
        exit(1);
    }
    w.set(CAP_PROP_FRAME_WIDTH, 640);
    w.set(CAP_PROP_FRAME_HEIGHT, 480);
    Mat_<Vec3b> a;
    namedWindow("janela");
    while (true) {
        // w.grab(); // Esvazia o buffer da camera (15/10/2025)
        w >> a; // get a new frame from camera
        imshow("janela", a);
        int ch=(signed char)(waitKey(30)); // E necessario (signed char)
        if (ch>=0) break;
    }
}
```

Copie e cole o programa acima num editor de texto e salve-o como `raspcam_cek.cpp`. Depois, digite no terminal:

```
Rasp$ compila raspcam2 -ocv
Rasp$ ./raspcam2
```

Deve aparecer uma janela com os quadros capturado pela câmera.

Nota (15/10/2025): Descobri que o comando “`w >> a`” pode devolver um quadro velho da câmera armazenado no buffer. Isso faz que a latência (o intervalo de tempo entre algo acontecer e esse acontecimento aparecer visualmente na janela) aumente. Para jogar fora o quadro velho, pode acrescentar o comando “`w.grab()`”. Isso diminui a latência, mas também diminui quadros por segundo.

Nota: Este programa só funciona corretamente quando ligar monitor diretamente no Raspberry. Se Raspberry estiver conectado remotamente ao computador via “VNC” ou “ssh -X”, a quantidade de quadros por segundo diminui substancialmente, pois não é possível transmitir quantidade grande de quadros por segundo do Raspberry para computador via Wi-Fi.

[Lição de casa #3/4 da aula 1. Vale 2,5.] Modifique o programa raspcam para criar raspvid.cpp que, além de mostrar a captura de câmera na tela, grava os quadros num vídeo. Exemplo:

```
Rasp$ raspvid saida.avi
```

Deve gerar vídeo saida.avi com os quadros capturados pela câmera. Os comandos para gerar vídeo de saída são basicamente:

```
...
Videowriter vo("nomevideo.avi",CV_FOURCC('X','V','I','D'),
               30,Size(640,480));
...
while (true) {
    w >> a;
    ...
    vo << a;
    ...
}
```

Nota: Se o seu Raspberry estiver conectado ao computador via Wi-Fi, o número de quadros capturados por segundo deve ser bem menos que 30 fps. Neste caso, o vídeo gravado vai parecer “acelerado”. Neste caso, altere 30 fps no programa para um número menor.

A apostila “vídeo”:

[<http://www.lps.usp.br/hae/apostila/video.pdf> ou <http://www.lps.usp.br/hae/apostila/video.odt>]

traz mais exemplos de como gerar vídeo.

Cuidado: OpenCV utiliza duas convenções diferentes para especificar o tamanho de imagem:

- 1) Mat_<uint8_t> a(rows,cols) Mat_<uint8_t> a(240,320) - OpenCV puro
- 2) Size(width,height) Size(320,240)

No primeiro caso, utiliza convenção de matriz (linhas, colunas). No segundo caso, utiliza convenção do espaço cartesiano (x, y). Assim, as especificações do tamanho da imagem e resolução de vídeo devem ter os parâmetros invertidos. Caso contrário, será gerado um vídeo nulo sem quadros e a biblioteca não emitirá nenhuma mensagem de erro.

4 Driver Motor Ponte-H L298n

Este driver (figuras 4 e 5) já foi utilizado na experiência #1 deste curso.

Nota: Há alguma controvérsia sobre como devem ser feitas as ligações de alimentação e dos jumpers da Ponte-H, com sites diferentes fornecendo informações contraditórias:

<http://www.instructables.com/id/Control-DC-and-stepper-motors-with-L298N-Dual-Moto/>
<https://blog.eletrogate.com/guia-definitivo-de-uso-da-ponte-h-l298n/>

O esquema de ligação que funcionou foi colocar a alimentação entre os conectores 4 e 5 (figura 5, respectivamente +5V e 0V). Mantive o jumper 3 no lugar. Nada foi ligado no conector 6. Os conectores 1 e 2 devem alimentar o motor A, e 13 e 14 o motor B (todas as numerações dos pinos referem-se à figura 5).

Os conectores 8 e 9 são as entradas dos sinais para acionar o motor A e 10 e 11 do motor B. O motor A vai girar de acordo com a tabela 1. O mesmo esquema é aplicado aos conectores 10 e 11 para controlar o motor B (figura 5).

Tabela 1: Sinal de entrada do ponte-H e o sentido de rotação do motor.

Motor A (B)	conector 8 (10)	conector 9 (11)
horário	5V	GND
anti-horário	GND	5V
ponto morto	GND	GND
freio	5V	5V

Se quisesse usar PWM (Pulse Width Modulation) por hardware, os jumpers 7 e 12 (figura 5) deveriam ser removidos e os sinais de PWM deveriam ser injetados nesses pinos. Como vamos usar PWM por software, mantenha esses jumpers nos seus lugares.

Nota: O manual do CI usado nesta ponte está em:

https://www.sparkfun.com/datasheets/Robotics/L298_H_Bridge.pdf

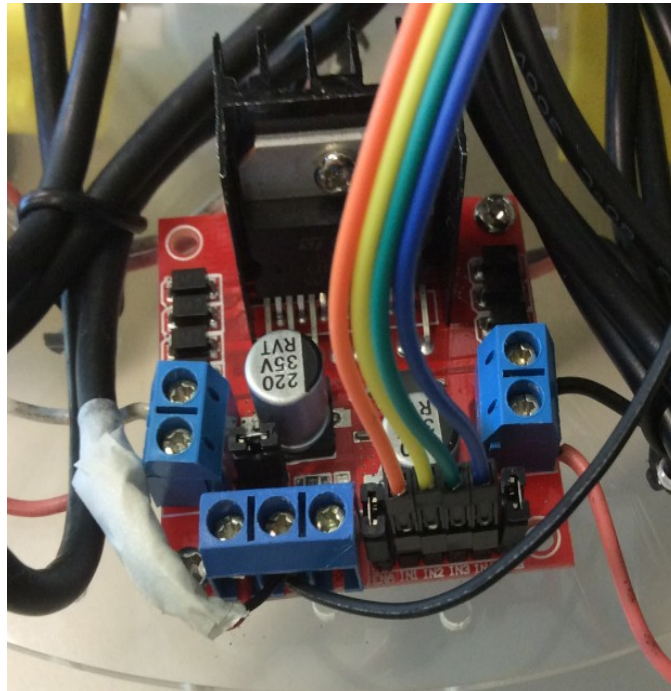


Figura 4: Ponte H L298n para Arduino montado no carrinho.

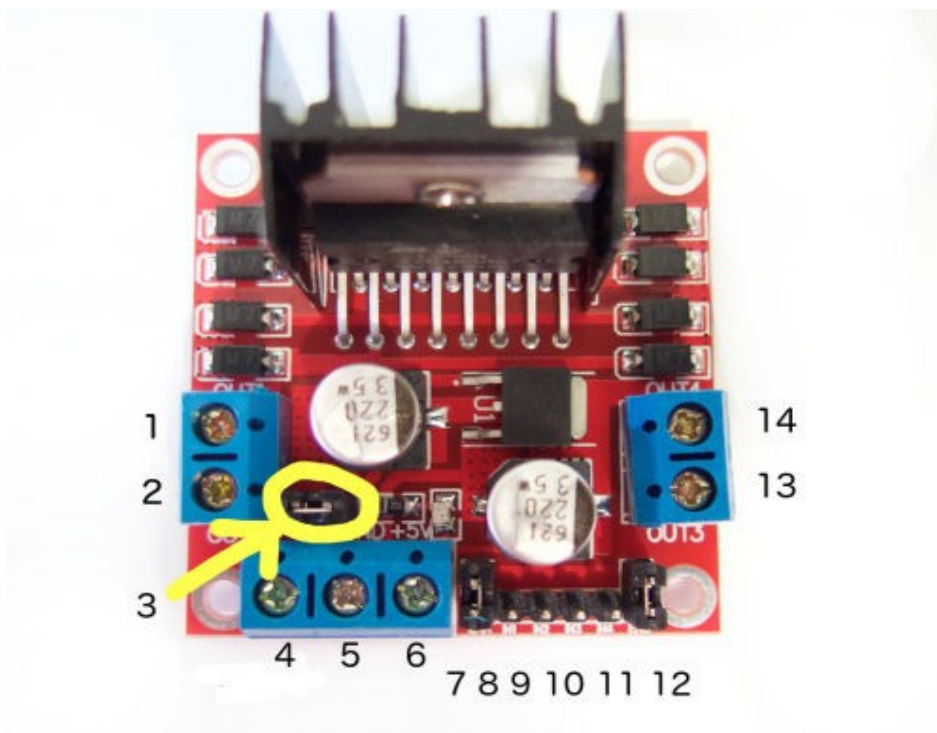


Figura 5: Ligações da ponte H L298N
(figura retirado de <https://www.instructables.com/Control-DC-and-stepper-motors-with-L298N-Dual-Moto/>).

5 WiringPi

<https://github.com/WiringPi/WiringPi>

<https://github.com/WiringPi/WiringPi-Node/blob/master/DOCUMENTATION.md#soft-pwm>

Raspberry Pi 3 possui 40 pinos, dos quais 26 são GPIO (General Purpose Input/Output, figura 6 e tabela 2).

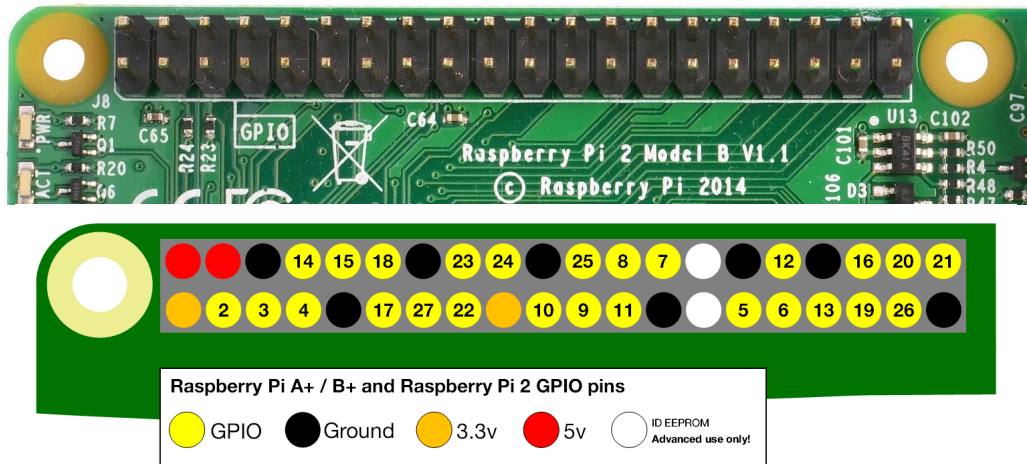


Figura 6: Pinagem de GPIO da Raspberry 2 e 3 (figura retirada do site de Raspberry).

Utilizaremos 4 pinos GPIO como saída, para controlar os dois motores do carrinho.

WiringPi é uma biblioteca para acessar GPIO do Raspberry Pi, escrita em C. No Raspberry Pi OS atual (2022 em diante), WiringPi já vem pré-instalada. Entre as características interessantes desta biblioteca, está “software PWM”: gera PWM via software, sem hardware extra.

WiringPi adota uma numeração de pinagem diferente do Raspberry (tabela 2). Por exemplo, o pino físico 5 (em cinza na tabela 2) é o pino 3 segundo a numeração de Raspberry, que por sua vez é o pino 9 segundo a numeração de WiringPi. O comando abaixo lista os pinos de GPIO:

```
pi@raspberrypi:~ $ gpio readall
```

Tabela 2: Numeração de pinagem do WiringPi. BCM = numeração de Raspberry. wPi = numeração de WiringPi. Physical = numeração física.

Pi 3						
BCM	wPi	Name	Physical	Name	wPi	BCM
		3.3v	1	2	5v	
2	8	SDA.1	3	4	5v	
3	9	SCL.1	5	6	0v	
4	7	GPI0. 7	7	8	TxD	15
		0v	9	10	RxD	16
17	0	GPI0. 0	11	12	GPI0. 1	1
27	2	GPI0. 2	13	14	0v	
22	3	GPI0. 3	15	16	GPI0. 4	4
		3.3v	17	18	GPI0. 5	5
10	12	MOSI	19	20	0v	
9	13	MISO	21	22	GPI0. 6	6
11	14	SCLK	23	24	CE0	10
		0v	25	26	CE1	11
0	30	SDA.0	27	28	SCL.0	31
5	21	GPI0.21	29	30	0v	
6	22	GPI0.22	31	32	GPI0.26	26
13	23	GPI0.23	33	34	0v	
19	24	GPI0.24	35	36	GPI0.27	27
26	25	GPI0.25	37	38	GPI0.28	28
		0v	39	40	GPI0.29	29

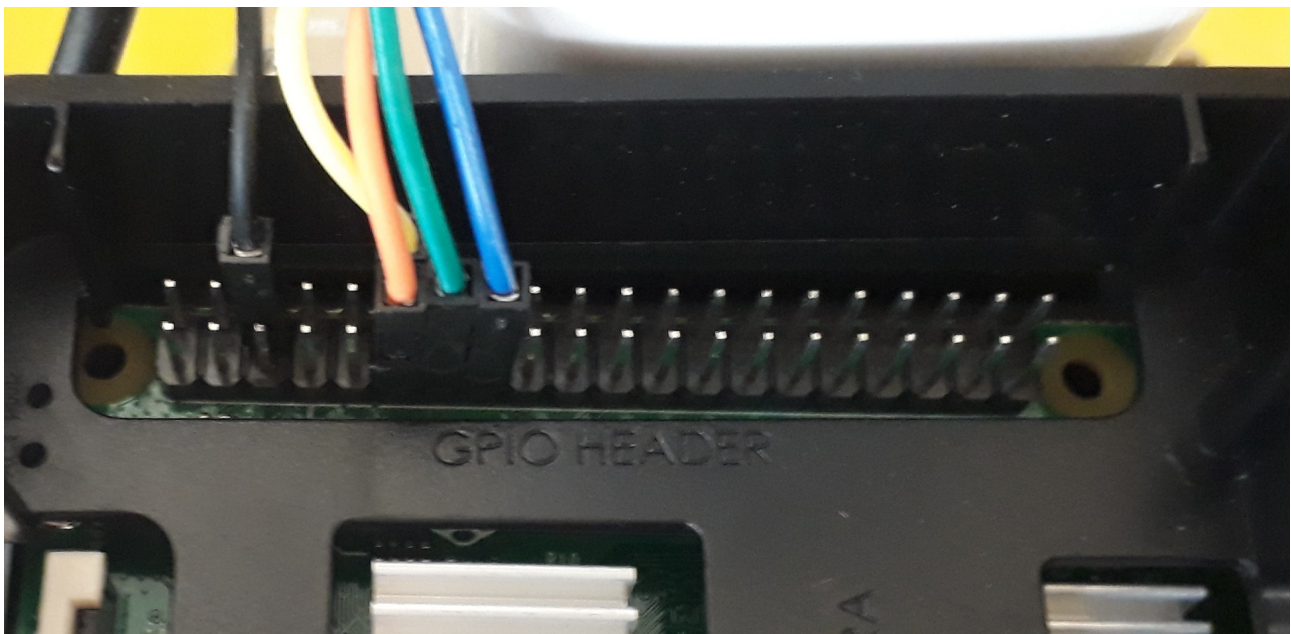


Figura 7: Conexões no Raspberry.

Escolhi usar os pinos wiringPi (0, 1) para o motor A e (2, 3) para o motor B, isto é, os pinos físicos (11, 12) e (13, 15). Você pode escolher outros. Ligue Raspberry, rode o seguinte programa e monitore as saídas dos pinos 0, 1, 2 e 3 com um multímetro, osciloscópio ou LEDs (em série com resistores de 270Ω). Verifique que as saídas GPIO estão ligando (3,3V) e desligando (0V) a cada 2 segundos.

Importante: Compile com opção -w para linkar com biblioteca wiringPi.

Cuidado: Para não causar curto-circuito acidental, não encoste pontas do multímetro diretamente nos pinos do GPIO. Use sempre fios com conectores fêmeas para fazer ligações nos pinos GPIO do Raspberry.

```
//blink3.cpp
//compila blink3 -w
#include <wiringPi.h>
int main () {
    wiringPiSetup () ;
    pinMode (0, OUTPUT) ;
    pinMode (1, OUTPUT) ;
    pinMode (2, OUTPUT) ;
    pinMode (3, OUTPUT) ;
    for (int i=0; i<4; i++) {
        digitalWrite (0, HIGH) ;
        digitalWrite (1, LOW) ;
        digitalWrite (2, HIGH) ;
        digitalWrite (3, LOW) ;
        delay (2000) ;
        digitalWrite (0, LOW) ;
        digitalWrite (1, HIGH) ;
        digitalWrite (2, LOW) ;
        digitalWrite (3, HIGH) ;
        delay (2000) ;
    }
    digitalWrite (0, LOW) ;
    digitalWrite (1, LOW) ;
    digitalWrite (2, LOW) ;
    digitalWrite (3, LOW) ;
}
```

Exercício: Mostre o programa blink3 funcionando, medindo tensões com um voltímetro ou osciloscópio. Caso não tenha nem voltímetro nem osciloscópio, pode fazer piscar dois LEDs (com resistores de 270Ω em série).

Nota: Se você não tem nem voltímetro, nem osciloscópio, nem LED com resistor em série, monte primeiro os motores do carrinho e depois rode o programa blink3. Neste caso, a tensão de saída será indicada pela rotação do motor: o carrinho vai andar para frente e para trás 4 vezes. Porém, não recomendo fazer isto, pois neste caso você estará fazendo as ligações dos motores sem ter a certeza de que GPIO está gerando as tensões corretas. Se algo der errado, não vai saber se o problema está antes ou depois do GPIO.

Pode acontecer do seu programa terminar com os motores do carrinho ligados. Neste caso, rode o programa abaixo para desligar os motores. É bom deixar o programa abaixo compilado e pronto para usar numa emergência.

```
//paramotor.cpp
//compila paramotor -w
#include <wiringPi.h>
int main () {
    wiringPiSetup () ;
    pinMode (0, OUTPUT) ;
    pinMode (1, OUTPUT) ;
    pinMode (2, OUTPUT) ;
    pinMode (3, OUTPUT) ;
    digitalWrite (0, LOW) ;
    digitalWrite (1, LOW) ;
    digitalWrite (2, LOW) ;
    digitalWrite (3, LOW) ;
}
```

Agora, vamos testar softPwm. Rode o seguinte programa e monitore as saídas dos pinos 0, 1, 2 e 3 (pinos físicos 11, 12, 13 e 15) com um multímetro ou osciloscópio. No multímetro, deve observar tensões aproximadamente 3V, 2V e 0V ($0,9 \times 3,3V = 2,97V$; $0,6 \times 3,3V = 1,98V$; e 0V). No osciloscópio, deve observar uma onda retangular de 100Hz com duty cycle de 90%, 60% e 0%.

```
//pwmroda4.cpp
//compila pwmroda4 -w -ocv
#include <wiringPi.h>
#include <softPwm.h>
#include <cstdlib>
int main() {
    wiringPiSetup();
    if (softPwmCreate(0, 0, 100)) exit(1);
    if (softPwmCreate(1, 0, 100)) exit(1);
    if (softPwmCreate(2, 0, 100)) exit(1);
    if (softPwmCreate(3, 0, 100)) exit(1);
    for (int i=0; i<2; i++) {
        softPwmWrite(0, 90); softPwmWrite(1, 0);
        softPwmWrite(2, 0); softPwmWrite(3, 90); delay(2000);

        softPwmWrite(0, 0); softPwmWrite(1, 90);
        softPwmWrite(2, 90); softPwmWrite(3, 0); delay(2000);

        softPwmWrite(0, 60); softPwmWrite(1, 0);
        softPwmWrite(2, 0); softPwmWrite(3, 60); delay(2000);

        softPwmWrite(0, 0); softPwmWrite(1, 60);
        softPwmWrite(2, 60); softPwmWrite(3, 0); delay(2000);
    }
    softPwmStop(0); softPwmStop(1); //alterado em 23/10/2024
    softPwmStop(2); softPwmStop(3);
}
```

Depois de verificar que a biblioteca WiringPi está funcionando, vamos fazer as ligações entre GPIO do Raspberry com ponte-H e motores (figuras 1, 4, 5 e 7). Faça todas as ligações com Raspberry e Ponte-H desenergizados.

Ligue primeiro as saídas da ponte-H aos dois motores (conectores 1 e 2 no motor A e 13 e 14 no motor B, figura 5). Depois, conecte os 4 fios GPIO do Raspberry (pinos físicos 11, 12, 13 e 15 da tabela 2 e figura 7) às 4 entradas do Ponte-H (conectores 8, 9, 10 e 11 na figura 5).

Além disso, você deve ligar entre si as terras da Raspberry (0V, pino físico 6 da tabela 2, figura 7) e da ponte-H (conector 5 na figura 5; conector onde chegam os fios pretos na figura 4). Isto é necessário para que Raspberry e Ponte-H possam utilizar fontes de alimentação independentes, para que todas as fontes usem a mesma terra.

Após se certificar de que não há problemas nas ligações, ligue a alimentação da ponte-H (5V 1A do powerbank) e Raspberry (5V 2A do powerbank). Se o seu powerbank tiver duas saídas “smart”, qualquer uma delas pode ser usada para alimentar ponte-H ou Raspberry. Teste a rotação dos motores executando o programa pwmroda4.cpp. Os motores esquerdo e direito devem rodar para frente e para trás com velocidade total e depois com meia velocidade (PWM 60%).

Nota (06/10/2025): O programa acima às vezes terminava com os motores ligados. As linhas marcadas em amarelo foram alteradas em para corrigir este problema. Por estranho que pareça, é necessário linkar com OpenCV (além de WiringPi) para o motor parar no fim:

```
raspberrypi$ compila pwmroda4 -w -ocv
```

Por outro lado, linkando OpenCV, o carrinho só começa se mover depois de passados algo como 2s.

Nota: Não iremos usar nenhum pino de GPIO como entrada, mas é bom lembrar que o autor de WiringPi avisa: “Remember: The Raspberry Pi is a 3.3 volt device! Attempting to directly connect to any 5V logic system will very likely result in tears...”

[Lição de casa #4/4 da aula 1. Vale 2,5.]

(a) Monte o carrinho “de forma caprichada”, com todos os componentes firmemente fixados no chassis e sem “fios soltos”. A qualidade de montagem do carrinho será levada em consideração ao atribuir a nota neste e nos próximos exercícios. Ligue o Raspberry e os motores no Powerbank.

(b) Faça conexão VNC ou SSH entre o Raspberry e o computador. A partir do computador, rode o programa `pwmroda4` no Raspberry. O carrinho deve girar duas vezes para direita com velocidade alta (90%) e depois com meia velocidade (60%). Se o carrinho se movimentar excessivamente, acerte os parâmetros do programa. Mostre ao professor (ou grave no vídeo) o carrinho se movimentando.

Exercício: Escreva um programa que faz o carrinho andar aproximadamente um metro, dá meia-volta, e anda novamente um metro para voltar ao ponto de partida. Mostre o programa funcionando. Provavelmente, o carrinho não vai retornar exatamente ao ponto de partida – não tem problema.

[Aula 1 parte 3. Fim]