**01 Controlando GPI no Console**

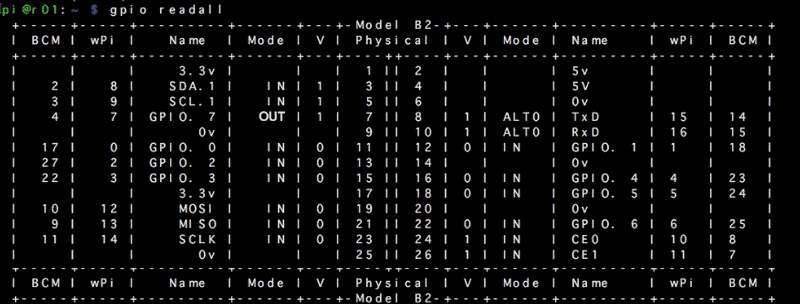
## Transcrição

Neste capítulo vamos dar continuidade na interação com o GPIO. Com o circuito ainda montado, e com a conexão remota ao Raspberry PI, executamos o seguinte comando:

gpio

O comando **gpio** é a base de tudo, com esse comando, seguido dos parâmetros, iremos ter acesso direto à nossa interface. Por exemplo:

gpio readall



Esse comando mostra o número do GPIO, do pino, se a porta está em IN/OUT, se está ligada ou desligada (1 ou 0, respectivamente). É uma forma de visualizar a nossa interface.

## Definindo o estado de uma porta

No capítulo passado, nós já vimos como se define o estado de uma porta (entrada ou saída). Por exemplo, para definir o estado do pino 7 como **saída**, executamos:

gpio mode 7 out

Para ligar o LED, precisamos colocar a porta (o pino) em alta (ligá-lo). Na linha de comando, em alta é representado pelo número **1**:

gpio write 7 1

E para desligar o LED, colocamos a porta em baixa. Na linha de comando, em baixa é representado pelo número **0**:

gpio write 7 0

Podemos fazer isso com qualquer porta, não precisa necessariamente estar com o LED. Para saber o estado de uma porta, usamos o comando:

gpio read 7

E o retorno será **0** para **porta em baixa** e **1** para **porta em alta**.

Com isso conseguimos interagir com o GPIO sem a necessidade de uma linguagem de programação no meio, diretamente da linha de comando.

# 02 Lendo um GPIO

**Vimos no vídeo como escrever e ler um GPIO na linha de comando:**

**Para escrever, devemos primeiro definir o GPIO como saída:**

**gpio mode 7 out**

**O comando acima define o pino 7 como saída. E para realmente escrever, usamos:**

**gpio write 7 1**

**O comando acima escreve para o pino 7 o valor 1 (HIGH).**

**Agora, como podemos ler um GPIO na linha de comando?**

Top of Form

* Alternativa correta



gpio mode 7 in

* Alternativa correta



gpio read 7

* Alternativa correta



gpio in 7

* Alternativa correta



gpio readPin 7

Bottom of Form

Para ler o GPIO específico devemos usar:

gpio read 7

Repare também que o seguinte comando existe:

gpio mode7 in

E define o pino 7 como entrada. Isso pode ser necessário quando você trabalha com botões na sua protoboard, que não é o nosso caso.

# 03 Esquema de numeração #1

**Vimos que através do comando gpio, podemos referenciar os GPIOs do Raspberry Pi de várias formas. Essas formas também são chamadas de esquemas de números. Um esquema define qual pino possui qual número, nada mais do que isso!**

**No entanto existem 3 esquemas. Qual é o esquema padrão do comando gpio?**

Top of Form

* Alternativa correta



Physical

* Alternativa correta



wiringPi (wPI)

* Alternativa correta



Broadcom (BCM)

Bottom of Form

O padrão do comando gpio é o esquema **wiringPi** (wPi). Isso está gerando constantemente confusão e o autor da biblioteca se justificou no site: <http://wiringpi.com/pins/>

A notícia boa é que o comando **gpio readall** mostra todos os esquemas e você sempre pode escolher o esquema que você gostaria de utilizar. Nesse curso usaremos o esquema Physical (ou seja, o número sequencial do pino).

# 04 Esquema de numeração #2

**Podemos definir um GPIO como saída ou entrada usando o comando gpio.**

**Qual é a sintaxe correta para definir PIN 3 como saída usando o esquema *Physical*?**

Top of Form

* Alternativa correta



gpio mode 3 out

Bottom of Form

  Alternativa correta



gpio -g mode 3 out

 A flag **-g** usa a numeração BCM.

 Alternativa correta



gpio -1 mode 3 out

Top of Form

* Correto, a flag **-1** habilita a numeração *Physical*.

Bottom of Form

Para usar a numeração pelo PIN (*Physical*) devemos adicionar a flag -1:

gpio -1 mode 3 out

Para usar a numeração BCM devemos usar a flag -g:

gpio -g mode 3 out

E lembrando também, sem flag é utilizada a wiringPi.

# 05 Esquema de numeração #3

**Responda as 3 perguntas:**

## Qual é o número do pino 15 (Physical) no esquema BCM ?

## Qual é o número do GPIO 18 (BCM) no esquema wiringPI?

## Qual é o número do GPIO 4 (BCM) no esquema Physical?

**Use o comando gpio readall para fazer o mapeamento!**

Top of Form

* Alternativa correta



3, 5, 7.

* Alternativa correta



22, 5, 3.

* Alternativa correta

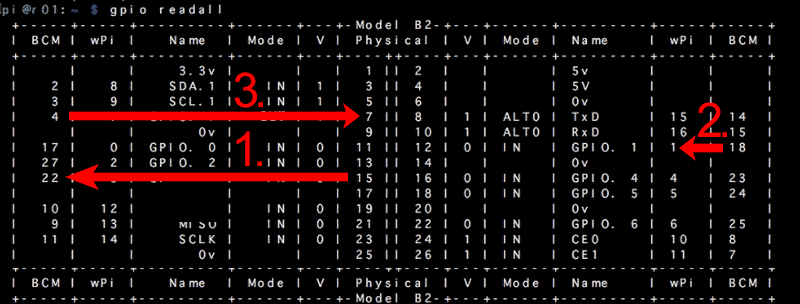


22, 1, 7.

Bottom of Form

* Qual é o número do pino 15 (Physical) no esquema BCM ?
  + O pino 15 é o 22 (BCM).
* Qual é o número do GPIO 18 (BCM) no esquema wiringPI?
  + GPIO 18 é pino 1 (wPi).
* Qual é o número do GPIO 4 (BCM) no esquema Physical?
  + GPIO 4 é pino 7.

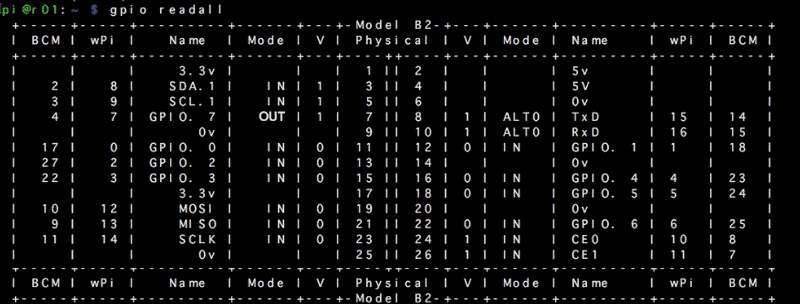
Segue também a saída do comando **gpio readall**:



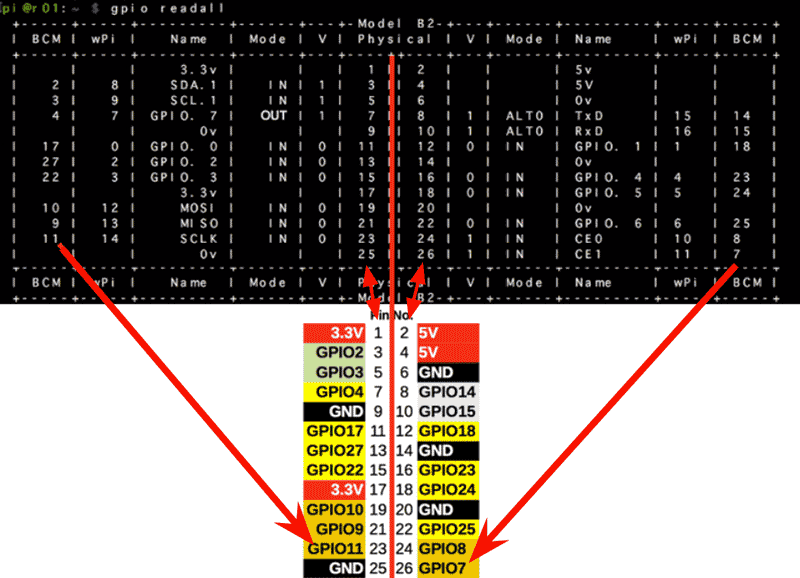
# 06 Resumo: Esquemas de numeração

No vídeo, mostramos o comando gpio readall que mostra o estado, nomes e os esquemas de numeração de todos os GPIOs.

Na imagem é apresentada a saída do um Raspberry PI com 26 pinos:



Para entender melhor, é bom dividir a tela em dois lados e lembrar do esquema dos GPIOs:



Repare que o lado esquerdo mostra os detalhes dos pinos (physical) com números impares (1, 3, 5 até 25). No lado direito os pinos pares (2, 4, 6 até 26). Repare também que as colunas se repetem, no lado esquerdo tem as mesmas colunas do lado direito.

Vamos passar pelas colunas e dar uma breve explicação:

## Coluna BCM (Broadcom)

É aquela numeração que leva em conta o número do GPIO e descarta os GPIOs especiais, como 3.3V, GND etc. Por isso tem alguns campos vazios, pois esses lugares são os pinos especiais.

## Coluna wPi (wiringPi)

É aquele esquema de numeração padrão da biblioteca **wiring Pi**. Essa biblioteca foi escrita na linguagem C para programar os GPIOs e existem vários wrappers em outras linguagens que usam essa biblioteca por baixo dos panos. Aliás, o comando gpio é apenas um wrapper para chamar as funções da biblioteca wiringPi no seu Raspberry PI! E como a biblioteca possui o seu próprio jeito de numeração, o comando gpio herdou desse jeito (mas você já sabe qual flag usar para mudar isso). Você pode saber mais no site do [Wiring Pi](http://wiringpi.com/).

## Coluna Name

É o nome do GPIO. Aqui pode ficar confuso, pois alguns nomes dos GPIOs não batem com o nome na coluna BCM. Por exemplo, o GPIO 4 (BCM) tem o nome "GPIO 7". É isso mesmo, confusão!

## Coluna Mode

O modo de operação: IN ou OUT. Para ser correto, existem mais estados (alt0 até alt5) que podem ser utilizados para componentes externos através de bibliotecas específicas. No nosso curso usaremos apenas IN ou OUT.

## Coluna V

O valor atual do GPIO, pode ser 0 (LOW) ou 1 (HIGH)

## Coluna Physical

O número do pino: aquela numeração sequencial mais simples de todas, que usaremos no curso.

# 07 Programando com GPIO no Python

## Transcrição

Aprendemos a interagir com o GPIO via console, mas também conseguimos interagir via Python!

Para isso, podemos utilizar a interface gráfica do Raspberry PI. Para uso remoto, precisamos do **VNC**, conforme vimos no primeiro curso. Podemos abrir o Python clicando em **Menu -> Programming -> Python 2 (IDLE)**.

Começamos criando um novo arquivo, clicando em **File -> New File**. Agora podemos escrever o nosso código. Primeiramente, para podermos nos comunicar com o GPIO, precisamos importar a sua biblioteca:

import RPi.GPIO as GPIO

Vamos importar também a biblioteca de temporização **time**:

import RPi.GPIO as GPIO

import time

Neste curso estamos usando como referência a numeração da pinagem do GPIO (e não o número do GPIO em si), mas precisamos "dizer" isso ao GPIO:

import RPi.GPIO as GPIO

import time

GPIO.setmode(GPIO.BOARD)

Se quisermos utilizar o número do GPIO como referência, basta trocar **BOARD** por **BCM**.

Outro detalhe que merece ser falado, é que há mensagens que o GPIO fica jogando na tela, por exemplo se a porta já estiver alocada. Para não exibi-las, adicionamos a seguinte linha:

GPIO.setwarnings(False)

O próximo passo é definir o pino 7 como saída (**OUT**). Então fazemos:

import RPi.GPIO as GPIO

import time

GPIO.setmode(GPIO.BOARD)

GPIO.setwarnings(False)

GPIO.setup(7,GPIO.OUT)

Para definir como entrada, basta trocar **OUT** por **IN**. Após isso, precisamos acender o LED, mas antes vamos exibir uma mensagem, utilizando o **print**:

import RPi.GPIO as GPIO

import time

GPIO.setmode(GPIO.BOARD)

GPIO.setwarnings(False)

GPIO.setup(7,GPIO.OUT)

print('LED ON')

E colocamos o pino 7 em **alta** (**HIGH**):

import RPi.GPIO as GPIO

import time

GPIO.setmode(GPIO.BOARD)

GPIO.setwarnings(False)

GPIO.setup(7,GPIO.OUT)

print('LED ON')

GPIO.output(7,GPIO.HIGH)

Isso já faz com que o LED fique aceso. Vamos deixá-lo aceso por 2 segundos e depois apagá-lo. Para apagá-lo, basta colocar o pino 7 em baixa (**LOW**) e para aguardar os 2 segundos, utilizamos o **time.sleep(2)**

import RPi.GPIO as GPIO

import time

GPIO.setmode(GPIO.BOARD)

GPIO.setwarnings(False)

GPIO.setup(7,GPIO.OUT)

print('LED ON')

GPIO.output(7,GPIO.HIGH)

time.sleep(2)

print('LED OFF')

GPIO.output(7,GPIO.LOW)

Com isso tudo feito, podemos testar o nosso código. Mas antes vamos criar um diretório, com o nome de **GPIO** e salvar o nosso arquivo Python dentro deste diretório, com o nome **teste.py**.

Agora podemos executar o arquivo, clicando em **Run -> Run Module**. Repare no LED ligando e desligando, e nas respectivas mensagens no console.

Podemos ainda deixar o LED piscando, envolvendo o código em um loop while e adicionando mais um **time.sleep(2)** após desligamento do LED:

import RPi.GPIO as GPIO

import time

GPIO.setmode(GPIO.BOARD)

GPIO.setwarnings(False)

GPIO.setup(7,GPIO.OUT)

while(1):

print('LED ON')

GPIO.output(7,GPIO.HIGH)

time.sleep(2)

print('LED OFF')

GPIO.output(7,GPIO.LOW)

time.sleep(2)

Salvamos e executamos novamente o arquivo, e vemos o LED piscando infinitamente.

Uma vez entendida essa codificação e conseguindo controlar o LED, acabaram os nossos problemas, qualquer dispositivo é passível do nosso controle, basta ele estar dimensionado e alimentado, mas a lógica que usaremos é bem parecida com a feita neste capítulo. Nos próximos capítulos vamos começar a construir a automação, só que ao invés de utilizar o LED, utilizaremos o **Módulo Relé**.

# 08 Python com GPIO

**A biblioteca necessária para utilizarmos o Python com o GPIO é:**

Top of Form

* Alternativa correta



**GPIO.LIB**

* Alternativa correta



**RPI.LIB**

* Alternativa correta



**RPI.GPIO**

* Alternativa correta



**RPi.GPIO**

Bottom of Form

Para que possamos utilizar o Python em nossa programação do GPIO, nosso programa deve fazer a importação da biblioteca **RPi.GPIO**, através da linha:

import RPi.GPIO as GPIO

Cuidado: Letra **i** minúscula!

# 09 Formas de se interagir com o GPIO

**Podemos interagir com o GPIO de diversas formas, portanto é correto afirmar que:**

Top of Form

* Alternativa correta



O GPIO só pode ser configurado através de comandos diretos ou através da linguagem de programação C.

* Alternativa correta



O GPIO deve ser exclusivamente configurado via linha de comando através do comando **gpio**.

* Alternativa correta



O GPIO só pode ser configurado através do Python, utilizando a biblioteca **GPIO.RPi**.

* Alternativa correta



O GPIO pode ser configurado através de comandos diretos ou através de linguagens de programação (Python, por exemplo).

Bottom of Form

Uma das facilidades na utilização do GPIO está no fato dele poder ser programado por Python, mas também é possível fazer o *setup* das portas diretamente através da linha de comando.

Parabéns, você acertou!

# 10 Esquema de numeração #4

**Na hora de programar os GPIOs com Python, também devemos decidir qual esquema de numeração usar. Você se lembra dos esquemas, certos?**

**No entanto, no mundo da biblioteca RPi.GPIO, só existem dois esquemas:**

## BCM

## Physical

**O esquema do wiringPI não existe na biblioteca RPi.GPIO.**

**Agora ,você se lembra como configurar o esquema physical (número sequencial) no Python?**

Top of Form

* Alternativa correta



GPIO.setmode(GPIO.BOARD)

Bottom of Form

 Correto!

 Alternativa correta



GPIO.setmode(GPIO.PHYSICAL)

 Errado, não existe.

 Alternativa correta



GPIO.setmode(GPIO.PIN)

  Alternativa correta



GPIO.setmode(GPIO.BCM)

Top of Form

Bottom of Form

Devemos usar :

GPIO.setmode(GPIO.BOARD)

E para usar os números dos GPIOs (BCM):

GPIO.setmode(GPIO.BCM)

# 11 Desligando as mensagens

**Para desligarmos as mensagens do sistema referente ao GPIO, devemos incluir em nosso programa a seguinte linha:**

Top of Form

* Alternativa correta



GPIO.setwarnings(False)

* Alternativa correta



GPIO.setmessages(False)

* Alternativa correta



GPIO.warnings(False)

* Alternativa correta



GPIO.messages(False)

Bottom of Form

Por default, o GPIO "joga" para a tela todas as mensagens, o que em muitas vezes pode atrapalhar o andamento do nosso programa. Para desabilitá-las, basta incluir em seu código a linha:

GPIO.setwarnings(False)

# 12 Configurando a porta

**Qual das linhas abaixo deve ser incluída em nosso código Python para configurar a porta 11 em HIGH?**

Top of Form

* Alternativa correta



Output(11, GPIO.HIGH)

* Alternativa correta



GPIO.11(GPIO.HIGH)

* Alternativa correta



Output(GPIO,11.HIGH)

* Alternativa correta



GPIO.output(11, GPIO.HIGH)

Bottom of Form

Através do Python, é muito simples configurarmos o GPIO, para definirmos uma porta em **HIGH**, utilizamos a seguinte sintaxe:

GPIO.output(Número do Pino, GPIO.HIGH)

No caso do pino 11:

GPIO.output(11, GPIO.HIGH)

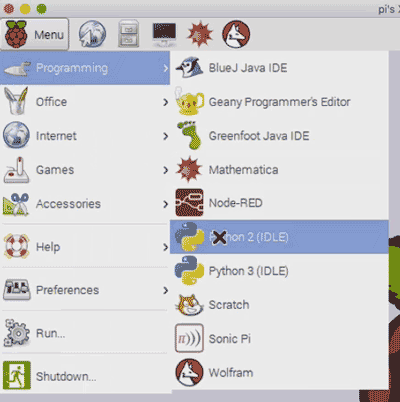
# 13 Mãos na massa: Ligando o LED

O nosso circuito está pronto e já testamos ele na linha de comando.

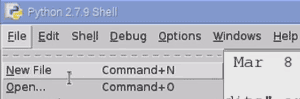
Agora vamos começar a usar o Python e escrever um pequeno programa para acender o LED programaticamente. Você precisa estar conectado ao seu Raspberry PI, usando um monitor, mouse e teclado dedicado, ou usando VNC, como explicamos no primeiro curso.

Mãos à obra!

**1)** Na interface gráfica do Raspberry PI, vá no ***Menu -> Programming -> Python 2***:



Depois vá para ***File -> New File***:



**2)** Vamos importar as bibliotecas **GPIO** e **time** do Python:

import RPi.GPIO as GPIO

import time

**3)** Logo abaixo configure os GPIOS pelo numeração dos pinos e com *warnings* desabilitados:

GPIO.setmode(GPIO.BOARD)

GPIO.setwarnings(False)

**4)** Configure a GPIO 7 como saída:

GPIO.setup(7, GPIO.OUT)

**5)** Imprima uma mensagem e coloque o GPIO 7 como HIGH:

print("LED ON")

GPIO.output(7, GPIO.HIGH)

**6)** Mande o Python *dormir* pois 2 segundos:

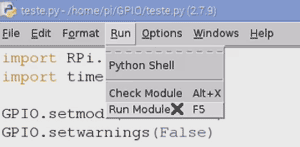
time.sleep(2)

**7)** Imprima novamente uma mensagem e coloque o GPIO 7 com LOW:

print("LED OFF")

GPIO.output(7, GPIO.LOW)

**8)** Salve o código dentro de um arquivo (por exemplo liga-led.py) e rode pelo Python: ***Run -> Run Module***:



**9)** Fique atento para pegar possíveis erros de sintaxe. Se tudo der certo, o LED acenderá!

Segue uma vez o código completo para sua comparação:

import RPi.GPIO as GPIO

import time

GPIO.setmode(GPIO.BOARD)

GPIO.setwarnings(False)

GPIO.setup(7, GPIO.OUT)

print("LED ON")

GPIO.output(7, GPIO.HIGH)

time.sleep(2)

print("LED OFF")

GPIO.output(7, GPIO.LOW)

# 14 Dificuldade com Python?

Se você encontra dificuldade com Python ou gostaria de aprender mais sobre a sintaxe e os comandos da linguagem, temos cursos específicos sobre o Python na Alura: [**Curso 1 sobre Python**](https://cursos.alura.com.br/course/python-3-introducao-a-nova-versao-da-linguagem).

Faz muito sentido aprender mais sobre essa linguagem fantástica não só por causa do Raspberry Pi!

# 15 Mãos na massa: Pisca LED

Agora crie um script que fará com que o LED fique piscando! Esse código é bem parecido com o código anterior, com a diferença que usaremos um laço para deixar o LED piscando.

Seguem os passos a serem realizados:

**1)** Com o Python aberto crie um novo arquivo.

**2)** Importe as bibliotecas do GPIO e de temporização. Lembra os seus respectivos nomes?

**3)** "Diga" para o GPIO usar como referência a numeração dos PINs da placa (GPIO.BOARD).

**4)** Defina o pino 7 como saída e remova as mensagens de alerta do GPIO.

**5)** Diferentemente do código anterior, inicialize um laço infinito com o comando:

while(1):

Obs: Muito cuidado com as próximas linhas. tudo abaixo do while(1) precisa estar indentado corretamente. Isso é, nas linhas abaixo do while, use um tabulador ou espaços, mas não misture os dois (ou você usa tabulador ou espaços para a indentação).

**6)** Imprima uma mensagem dizendo que o LED está ligado e ligue-o, deixando a porta em **alta** (***HIGH***).

**7)** Antes de desligar o LED, peça para o script aguardar 2 segundos.

**8)** Imprima mais uma mensagem, só que dessa vez dizendo que o LED está desligado e desligue-o em seguida, deixando a porta em **baixa** (***LOW***).

Ao final, o script ficará assim, novamente muito cuidado com a indentação:

import RPi.GPIO as GPIO

import time

GPIO.setmode(GPIO.BOARD)

GPIO.setwarnings(False)

GPIO.setup(7,GPIO.OUT)

while(1):

print('LED ON')

GPIO.output(7,GPIO.HIGH)

time.sleep(2)

print('LED OFF')

GPIO.output(7,GPIO.LOW)

time.sleep(2)

Salve-o e execute-o, clicando em ***Run -> Run Module***. Repare no LED ligando e desligando (ou seja, piscando), e nas respectivas mensagens no console!

