

# Projeto 03

## Controle de Acesso – Teoria

Jan K. S. – janks@puc-rio.br

ENG1419 – Programação de Microcontroladores

# Ferramentas do Python



Datas e Horários

```
>>> from datetime import datetime  
  
# ano, mês, dia, hora, minuto e segundo  
>>> data_e_hora = datetime(2015, 10, 21, 16, 29, 37)  
  
>>> data_e_hora  
datetime.datetime(2015, 10, 21, 16, 29, 37)  
  
# apenas ano, mês e dia  
>>> data = datetime(2015, 10, 21)  
  
>>> data  
datetime.datetime(2015, 10, 21, 0, 0)  
  
>>> agora = datetime.now()  
  
>>> agora  
datetime.datetime(2018, 8, 31, 11, 37, 44, 111740)
```

```
>>> from datetime import datetime, timedelta  
>>> agora = datetime.now()  
>>> agora  
datetime.datetime(2018, 8, 31, 11, 38, 14, 111740)  
>>> daqui_a_uma_hora = agora + timedelta(hours=1)  
>>> daqui_a_uma_hora  
datetime.datetime(2018, 8, 31, 12, 38, 14, 111740)  
>>> agora - timedelta(days=300, minutes=50)  
datetime.datetime(2017, 11, 4, 10, 48, 14, 111740)  
>>> daqui_a_uma_hora > agora  
True
```

```
>>> from datetime import datetime, timedelta  
>>> agora = datetime.now()  
>>> daqui_a_uma_hora = agora + timedelta(hours=1)  
>>> intervalo = daqui_a_uma_hora - agora  
>>> intervalo  
datetime.timedelta(0, 3600) # dias, segundos  
>>> intervalo.seconds  
3600
```

```
>>> intervalo2 = agora - daqui_a_uma_hora  
>>> intervalo2  
datetime.timedelta(-1, 82800)
```



-1 dia + 82800 segundos = -3600 segundos. Ok, mas fica esquisito...

```
>>> from datetime import datetime  
>>> agora = datetime.now()  
>>> str(agora.day) + "/" + str(agora.month)  
'26/3'  
  
>>> agora.strftime("%d/%m")  
'26/03'  
  
>>> agora.strftime("%d/%m/%Y às %H:%M:%S")  
'26/03/2018 às 15:12:35'  
  
>>> agora.strftime("%c")  
'Mon Mar 26 15:12:35 2018'
```

# Hardware



Campainha (Buzzer)

Criar um livro  
Descarregar como PDF  
Versão para impressão  
  
Noutros projetos  
Wikimedia Commons  
  
Ferramentas  
Páginas afluentes  
Alterações relacionadas  
Carregar ficheiro  
Páginas especiais  
Hipervínculo permanente  
Informações da página  
Elemento Wikidata  
Citar esta página  
  
Noutros idiomas   
العربية  
Deutsch  
English  
Español  
हिन्दी  
Italiano  
日本語<sup>1</sup>  
한국어<sup>2</sup>  
中文<sup>3</sup>

7 Referências  
8 Bibliografia  
9 Ligações externas

## Mecanismo [editar | editar código-fonte]

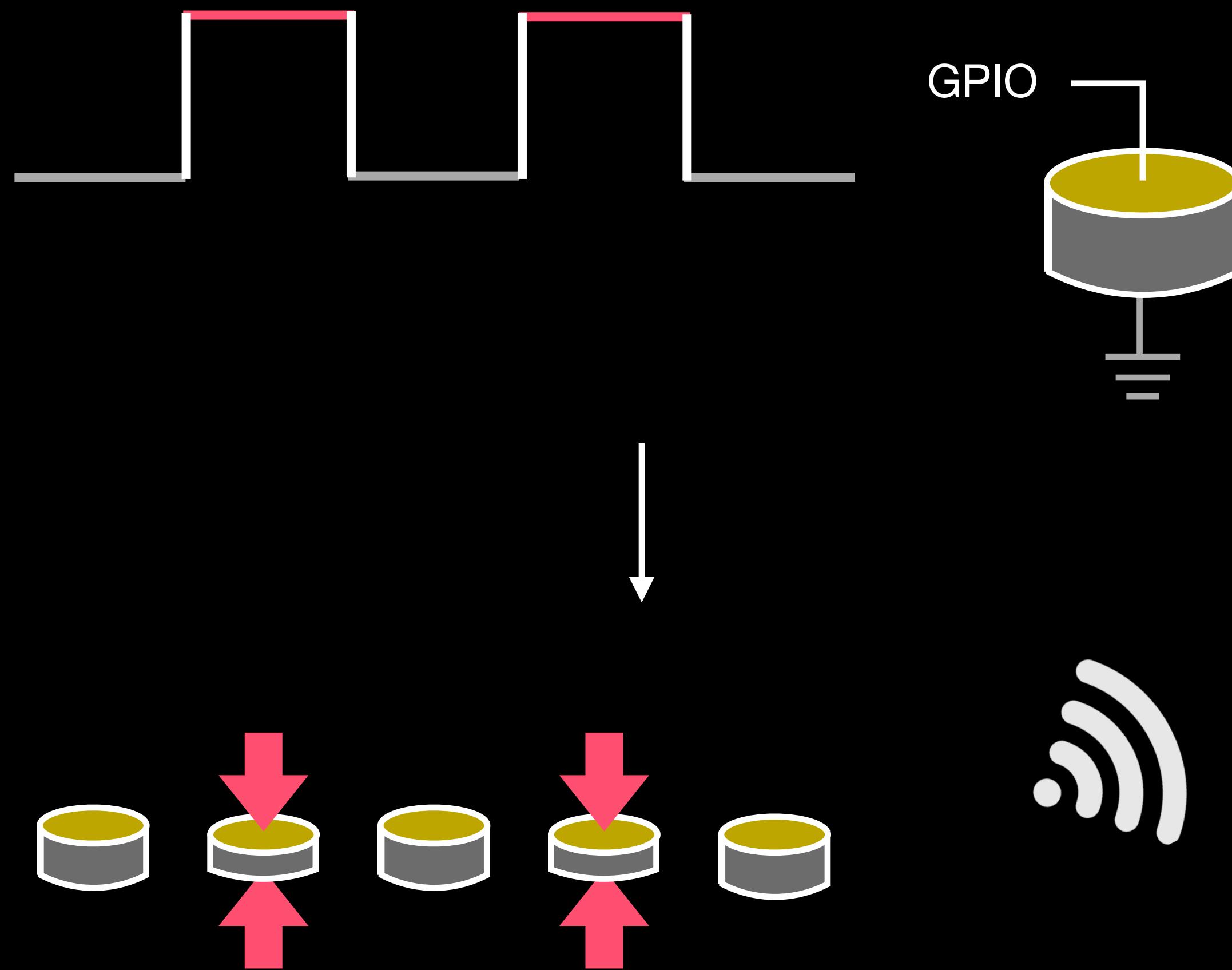
O efeito piezoelétrico é entendido como a interação eletromecânica linear entre a força mecânica e o estado elétrico ([forças de Coulomb](#)) em materiais cristalinos ([cerâmicos](#), [polímeros](#)).

O efeito piezoelétrico é um processo reversível em que os materiais exibem o efeito piezoelétrico direto (a geração interna de [carga elétrica](#) resultante de uma [força](#) mecânica aplicada), mas também exibem o efeito piezoelétrico reverso (a geração interna de uma tensão mecânica resultante de um [campo elétrico](#) aplicado). Por exemplo, os cristais de [titânato zirconato de chumbo](#) irão gerar piezoelectricidade mensurável quando a sua estrutura estática é deformada por cerca de 0,1% da dimensão inicial. Por outro lado, esses mesmos cristais mudam cerca de 0,1% da sua dimensão estática quando um campo elétrico externo é aplicado ao material. Como exemplo, o efeito piezoelétrico inverso é usado na produção de ondas de [ultrassom](#).<sup>[1]</sup>

Um disco piezoelétrico gera uma diferença de potencial quando deformado.

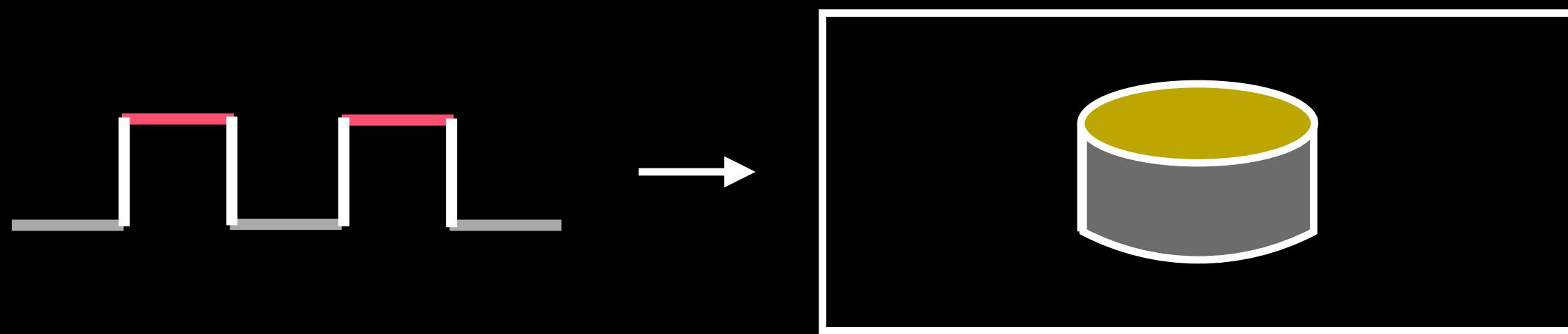
## Cristais [editar | editar código-fonte]

Utilizando argumentos referentes à [simetria](#), o efeito piezoelétrico não existe em materiais que apresentam simetria central, e desta forma, podem ser [polarizados](#), ou seja, a piezoelectricidade pode ser explicada pela assimetria de polarização iônica. Porém, elementos puros, tais como [selênio](#) (Se) e [telúrio](#) (Te) também exibem a propriedade de piezoelectricidade. Nestes casos, a [polarização elétrica induzida](#) é atribuída à [distribuição eletrônica](#), que é alterada pela ação externa.

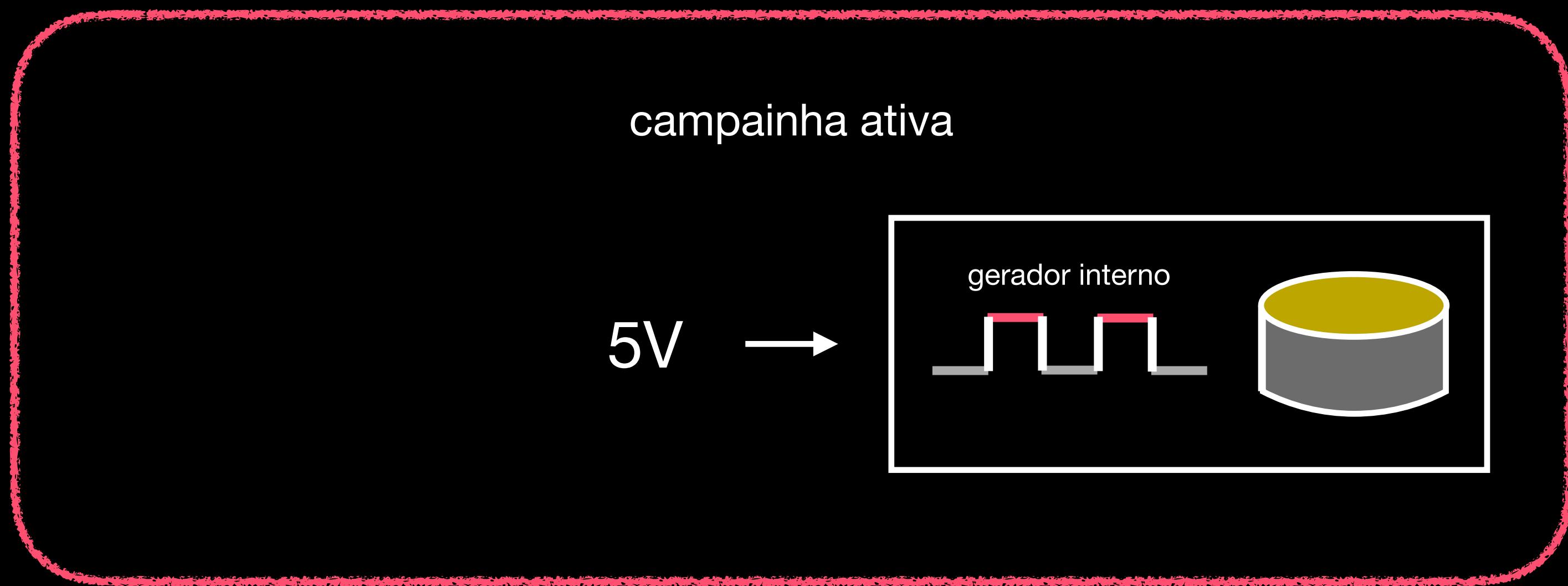


Geração de Som por Pulso

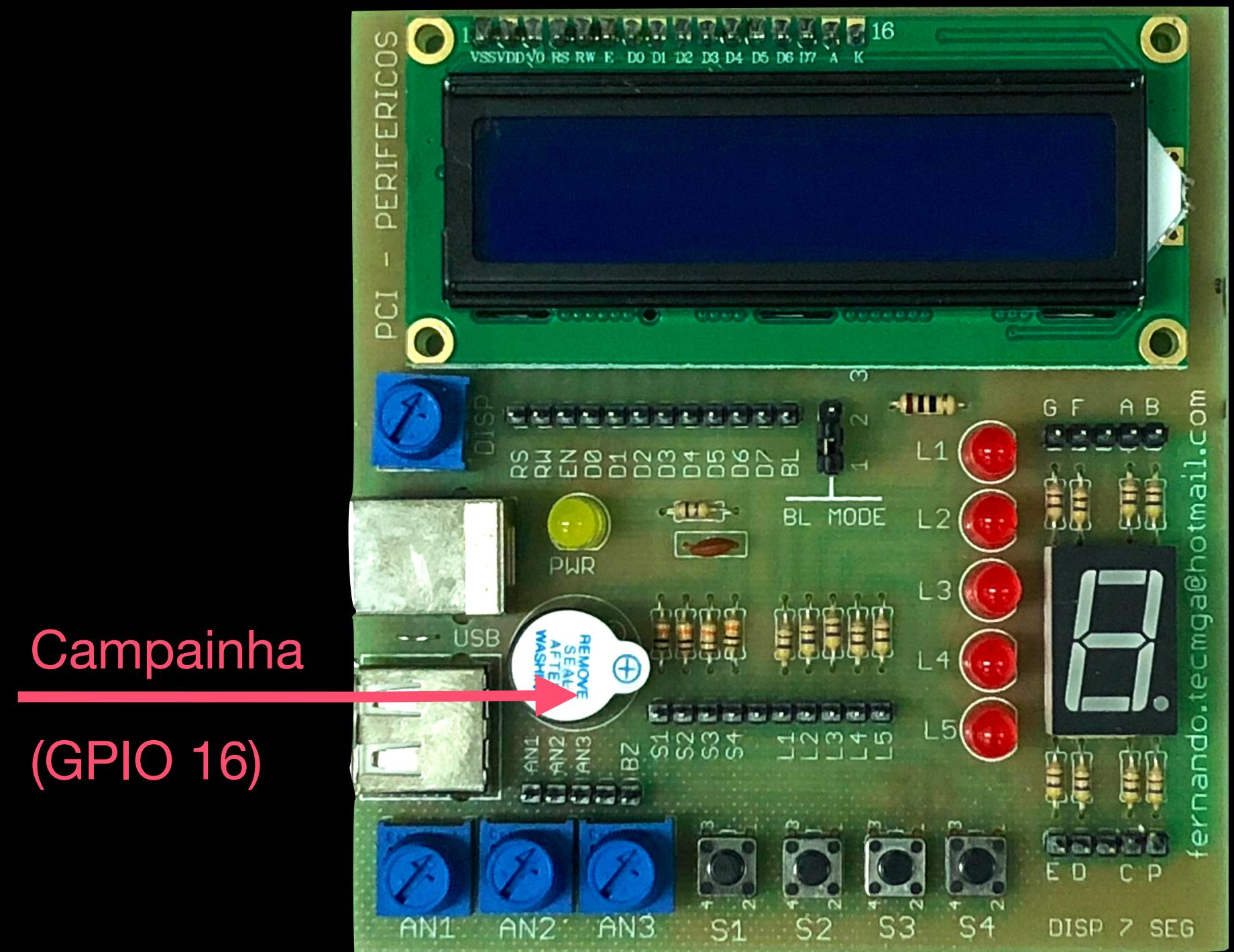
campainha passiva



campainha ativa



Campainhas Ativas e Passivas



Campainha  
(GPIO 16)

Campainha Ativa na Placa de Periféricos Básicos

```
>>> from gpiozero import Buzzer  
>>> campainha = Buzzer(16)  
>>> campainha.on()  
>>> campainha.is_active  
True  
  
>>> campainha.off()  
>>> campainha.is_active  
False  
  
>>> campainha.toggle()  
>>> campainha.is_active  
True  
  
>>> campainha.toggle()  
>>> campainha.is_active  
False
```

```
>>> from gpiozero import Buzzer  
>>> sirene = Buzzer(16)  
>>> sirene.beep()  
>>> sirene.off()  
>>> sirene.beep(on_time=0.5, off_time=2, n=3)
```

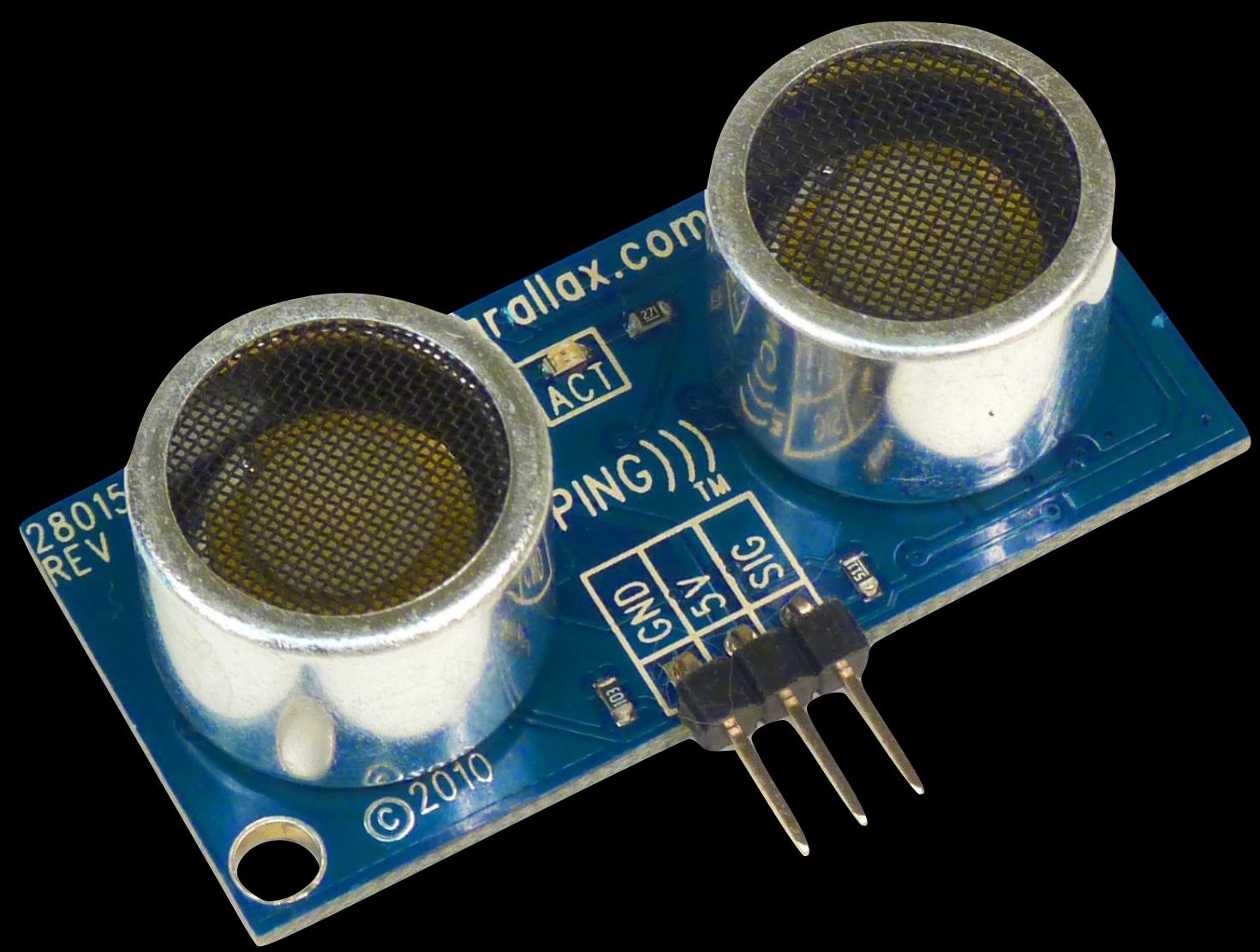


.on()  
.off()  
.toggle()  
.is\_active  
.beep(on\_time=, off\_time=, n=)

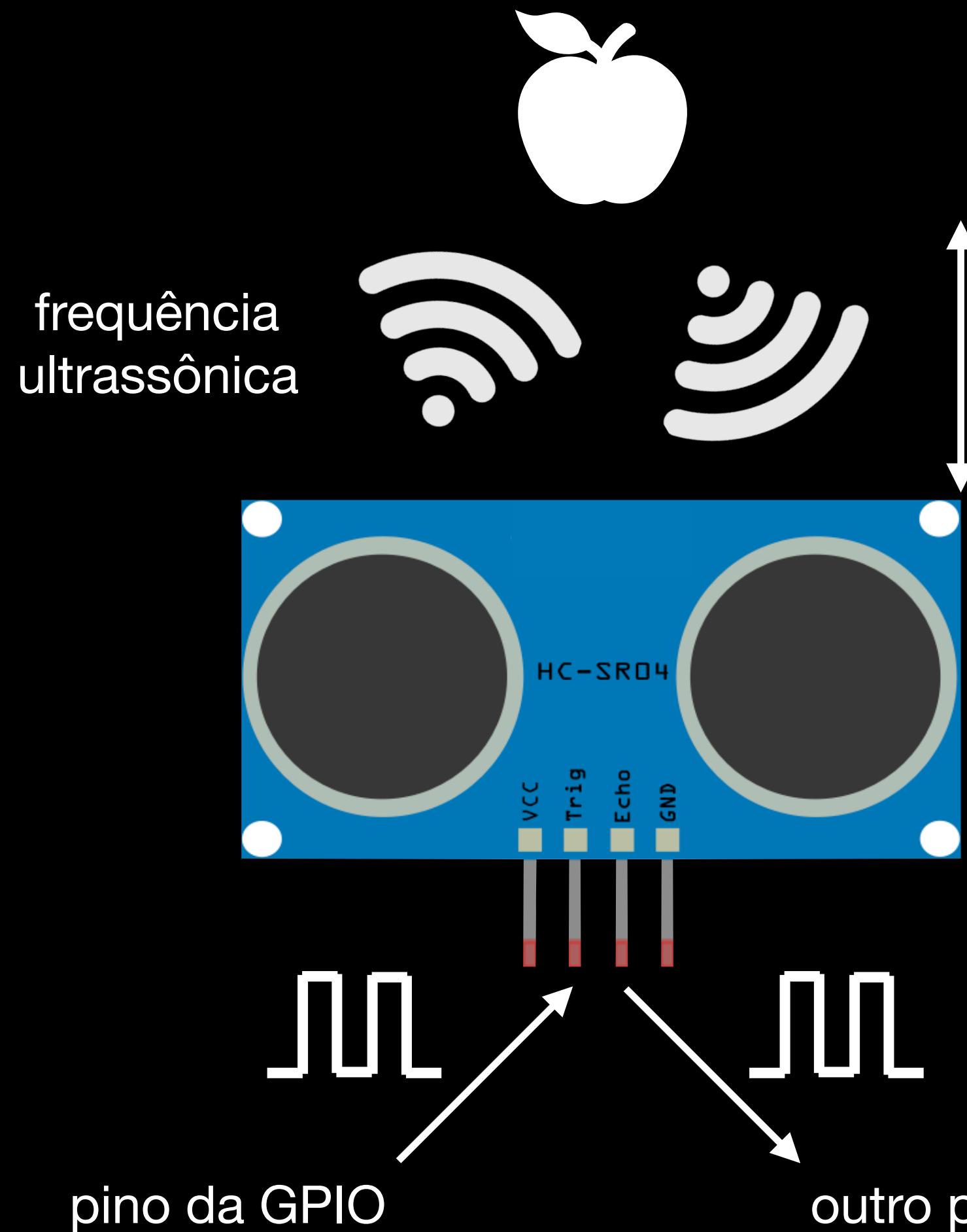


.on()  
.off()  
.toggle()  
.is\_lit  
.blink(on\_time=, off\_time=, n=)

Similaridades entre a Campainha e o LED

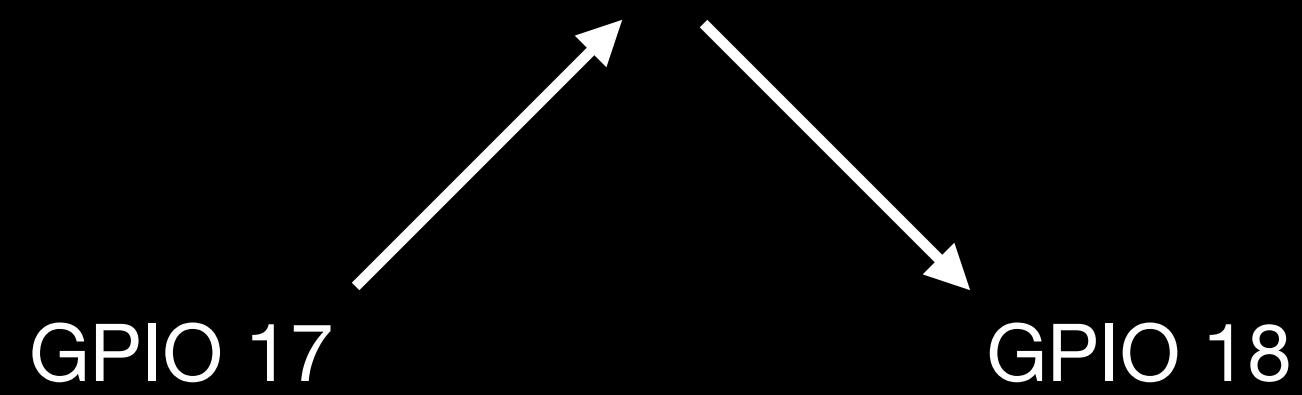
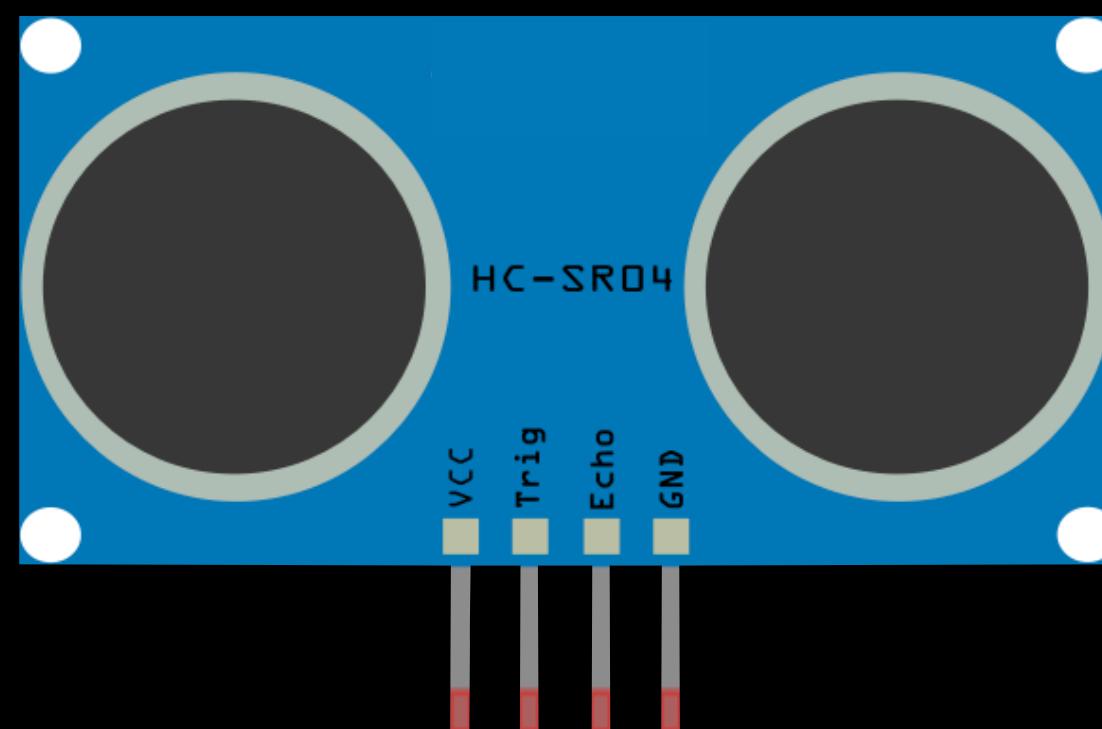


Sensor de Distância Ultrassônico



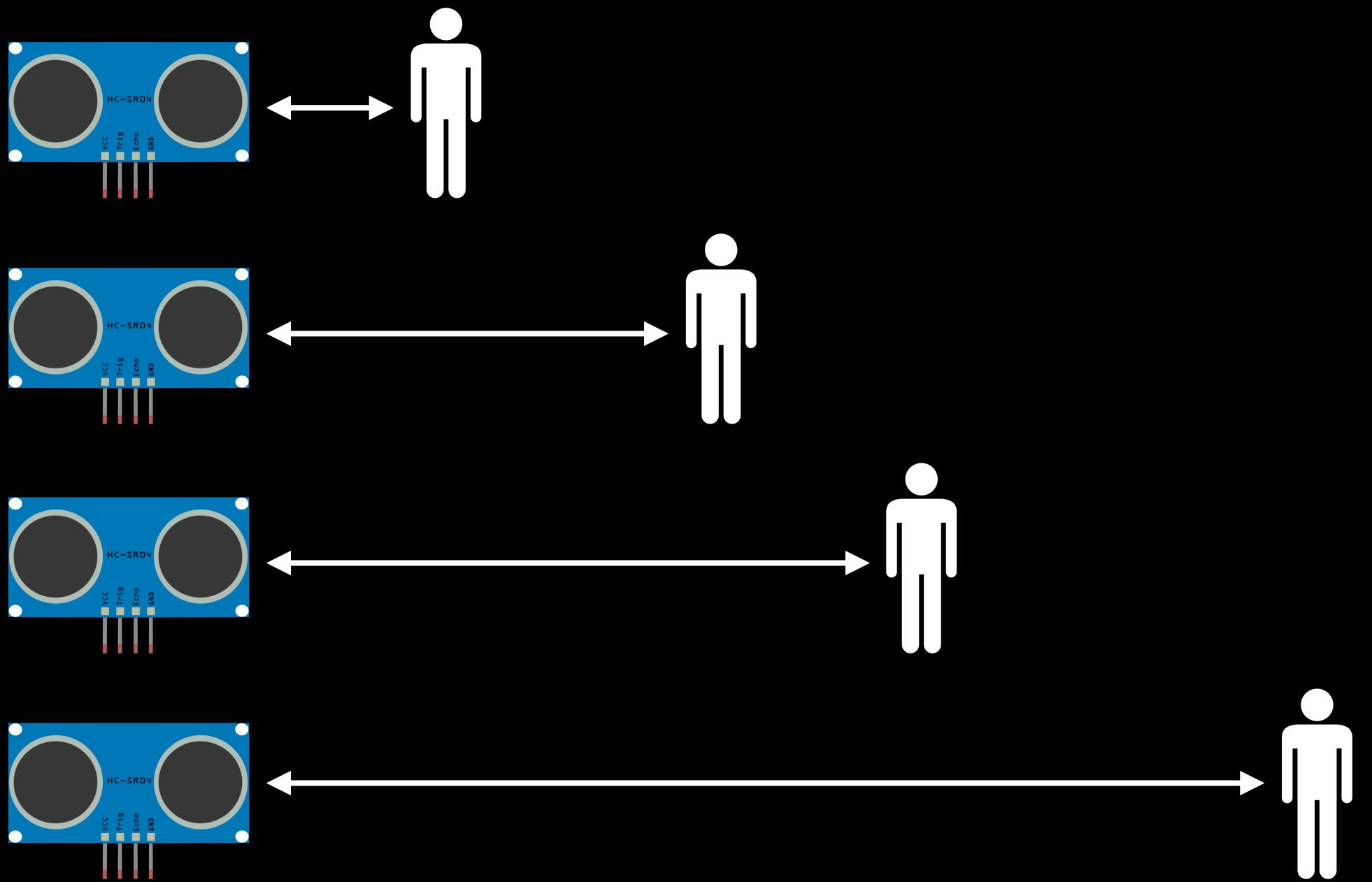
$$d = \frac{v_{\text{som}} \Delta t}{2}$$

Cálculo da Distância pelo Tempo Entre Emissão e Eco



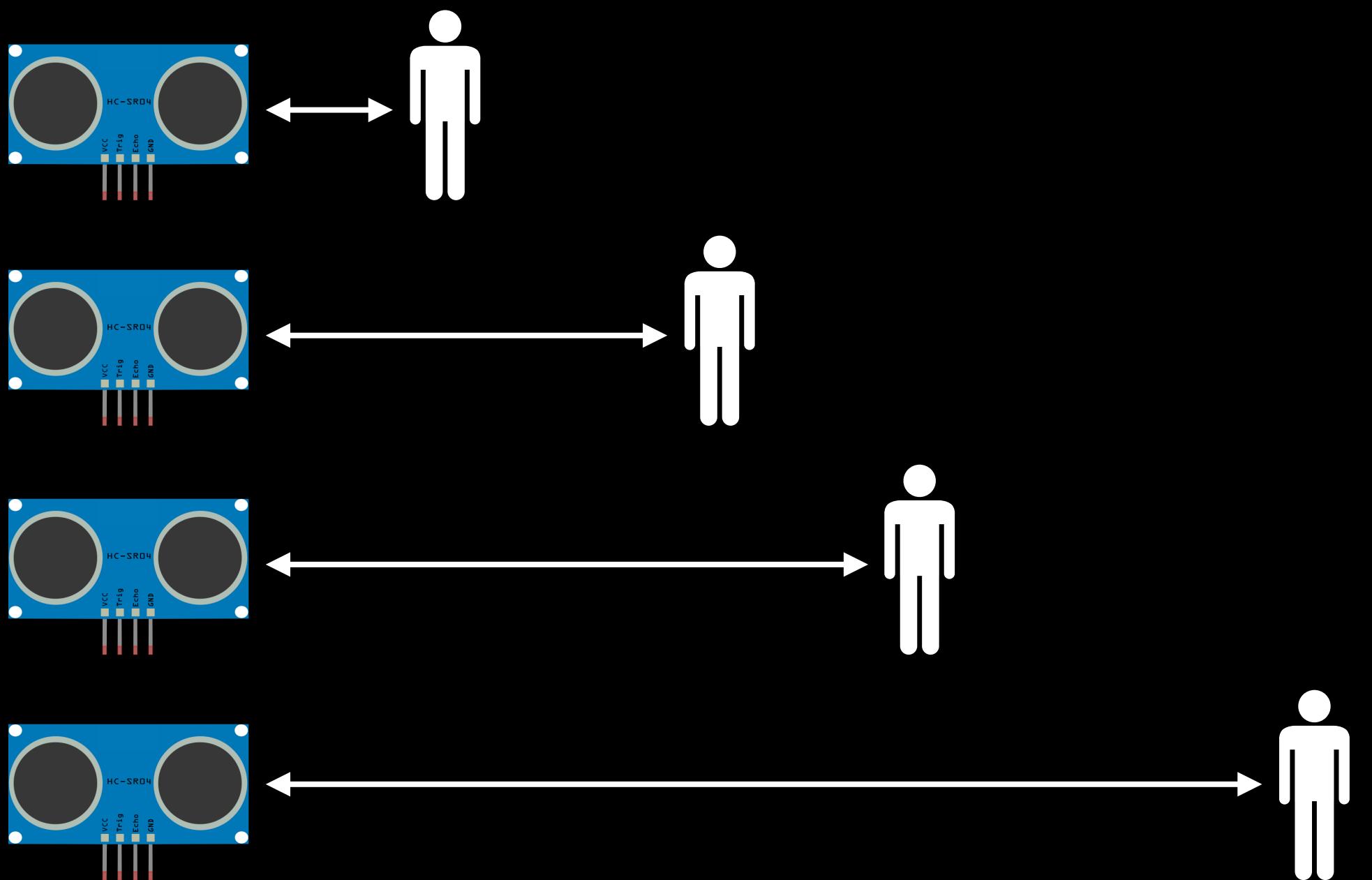
Pinos Usados pela Nossa Placa

```
>>> from gpiozero import DistanceSensor  
>>> sensor = DistanceSensor(trigger=17, echo=18)  
>>> sensor.distance  
0.2682768273353576  
>>> sensor.distance  
0.8682768273353576  
>>> sensor.distance  
1.0  
>>> sensor.distance  
1.0
```



Medição de Distância (em Metros)

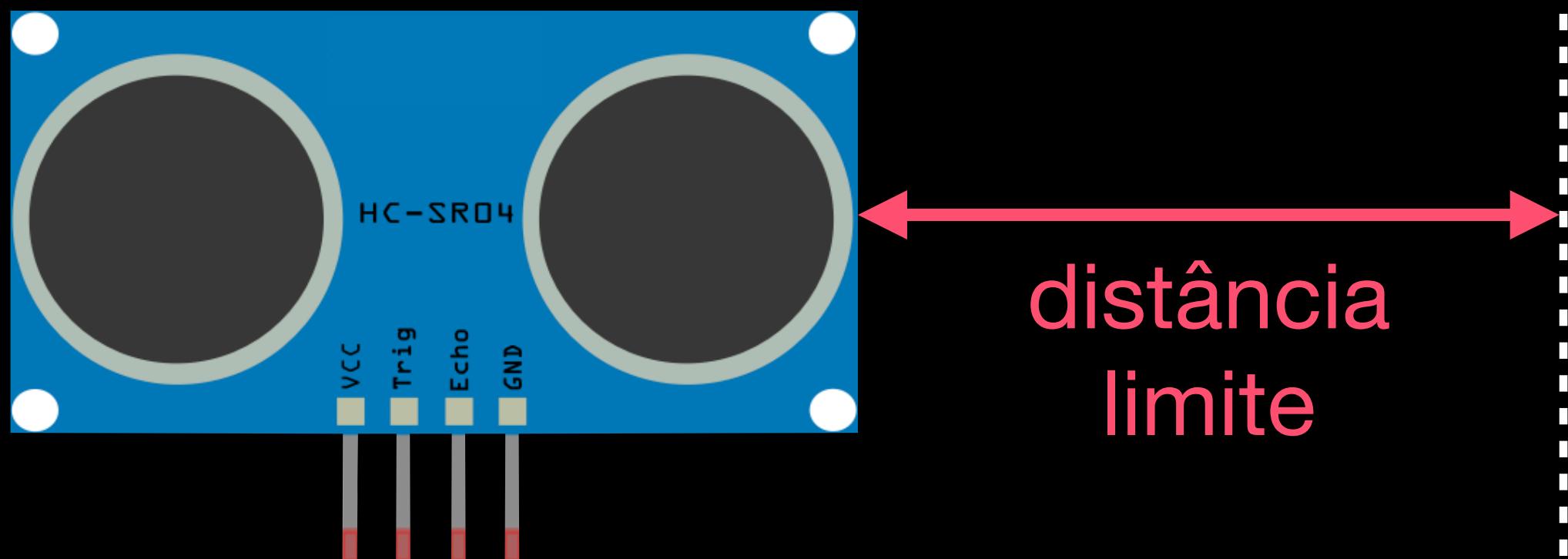
```
>>> from gpiozero import DistanceSensor  
>>> sensor = DistanceSensor(trigger=17, echo=18)  
>>> sensor.max_distance = 2 # valor máximo em metros  
>>> sensor.distance  
0.2682768273353576  
>>> sensor.distance  
0.8682768273353576  
>>> sensor.distance  
1.0057887244224546  
>>> sensor.distance  
1.4057887244224546
```



Medição de Distância (em Metros) com Parâmetro de Distância Máxima

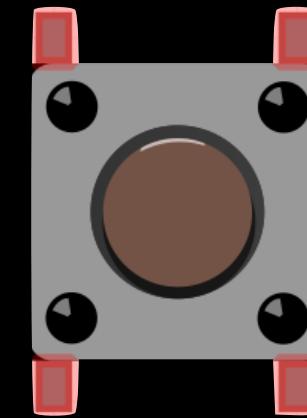
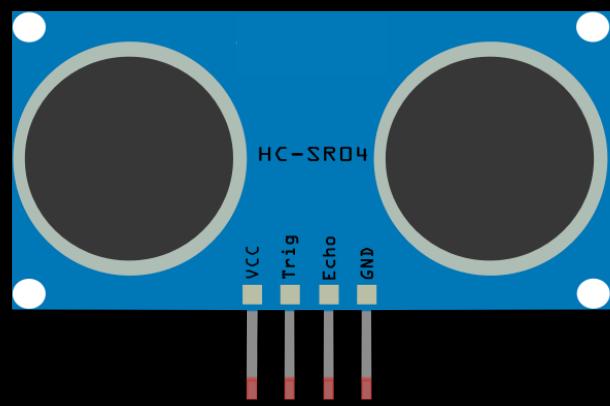
proximidade

afastamento



Eventos de Proximidade

```
>>> from gpiozero import DistanceSensor  
>>> sensor = DistanceSensor(trigger=17, echo=18)  
>>> sensor.threshold_distance = 0.5 # limite em metros  
  
>>> def alertar_proximidade():  
...     print("Objeto próximo!")  
  
...  
>>> sensor.when_in_range = alertar_proximidade  
  
>>> led = LED(21)  
>>> sensor.when_out_of_range = led.toggle
```



```
.when_in_range = funcao1  
.when_out_of_range = funcao2
```

```
.when_pressed = funcao1  
.when_released = funcao2
```

Similaridades entre a Sensor de Distância e o Botão

# Software

Os dados se perdem se o programa for encerrado.  
E essa lista pode ficar muito grande com o tempo...

```
>>> from gpiozero import Button  
>>> from datetime import datetime  
>>> instantes_pressionados = []  
>>> def botao_pressionado():  
...     novo_instante = datetime.now()  
...     instantes_pressionados.append(novo_instante)  
...  
>>> botao = Button(11)  
>>> botao.when_pressed = botao_pressionado
```





meus\_dados

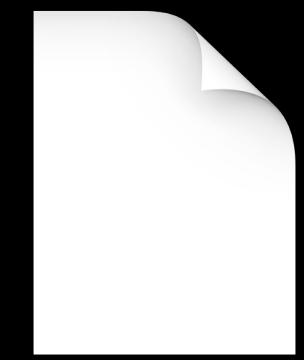
100 kB

com o tempo



meus\_dados

1.2 GB



meus\_dados\_2017

20 MB



meus\_dados\_2018

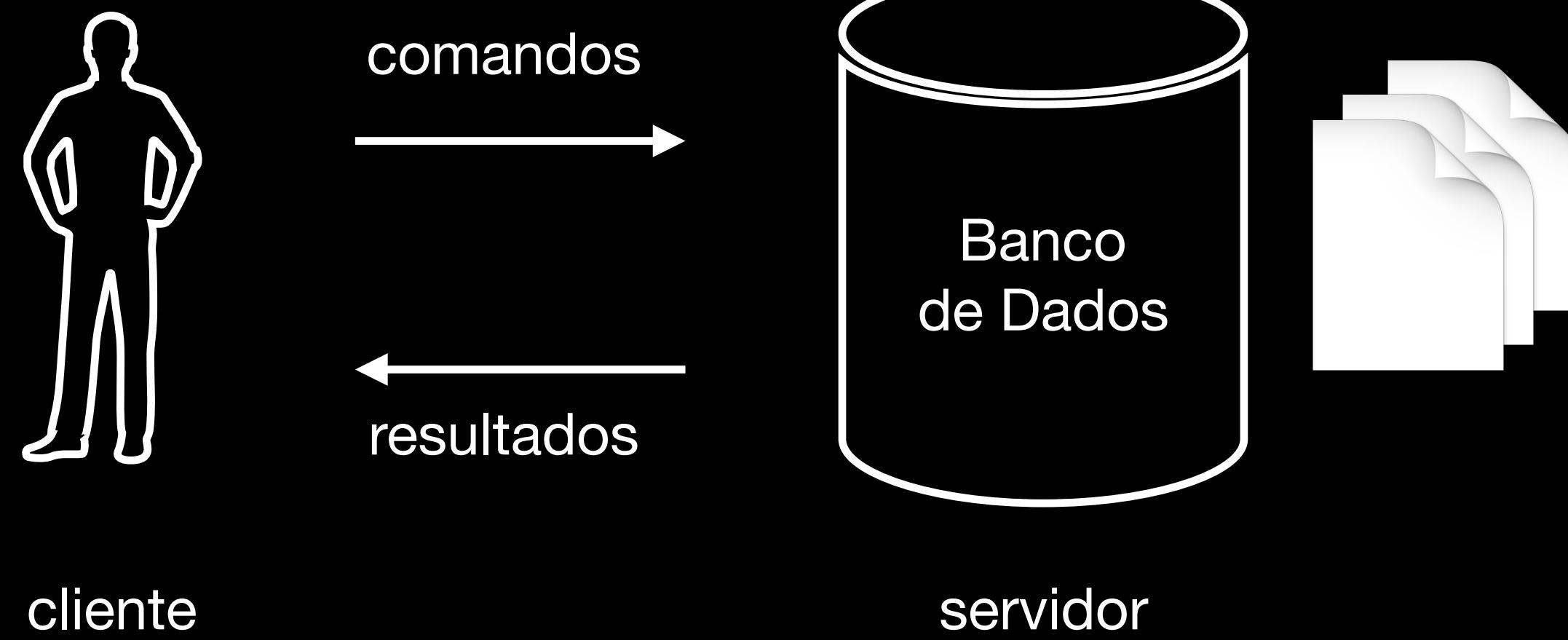
12 MB

...

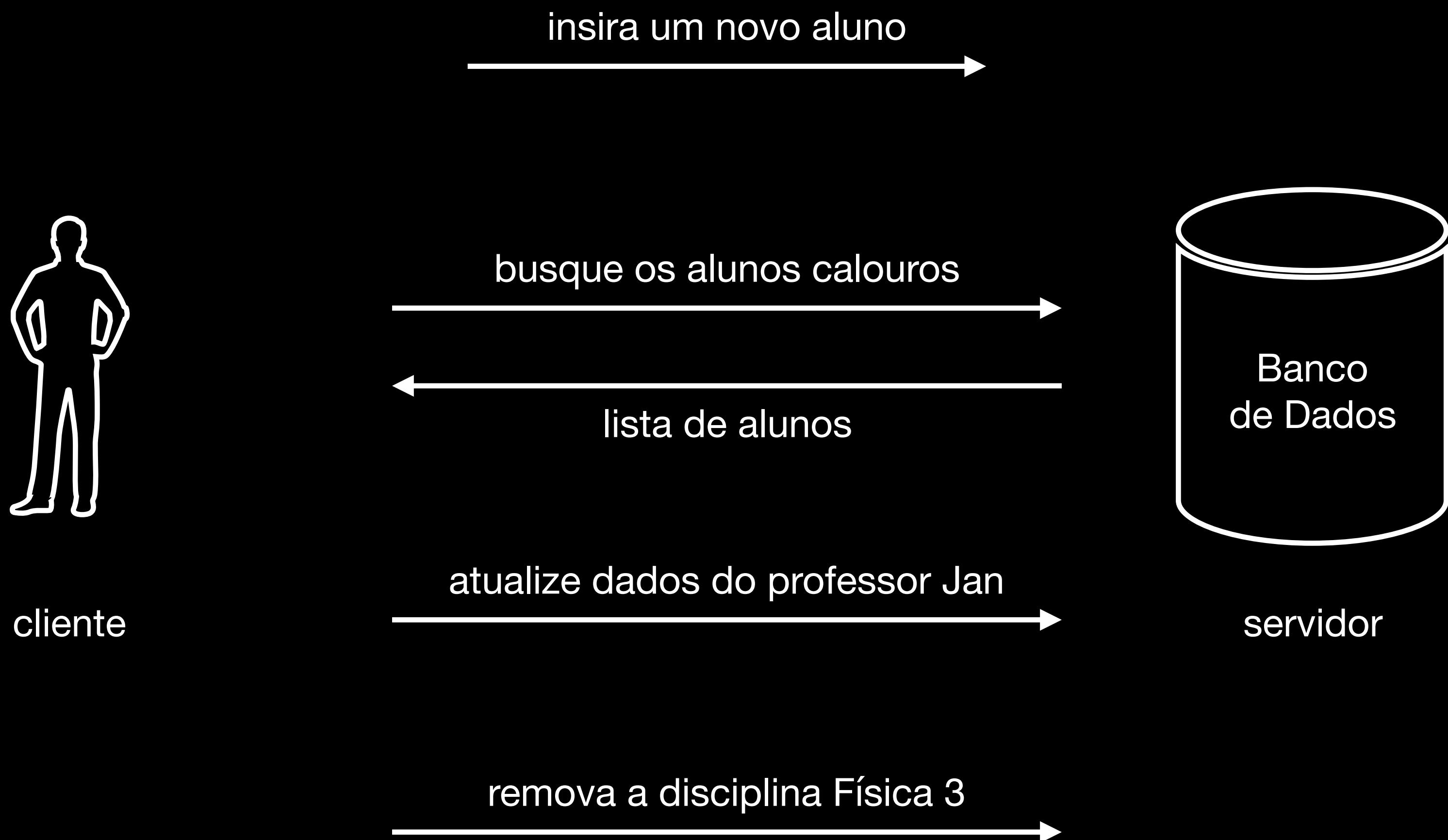
Complicado de  
gerenciar...



Armazenamento de Dados em Arquivos



Banco de Dados



Exemplos de Comandos entre Cliente e Servidor de Banco de Dados

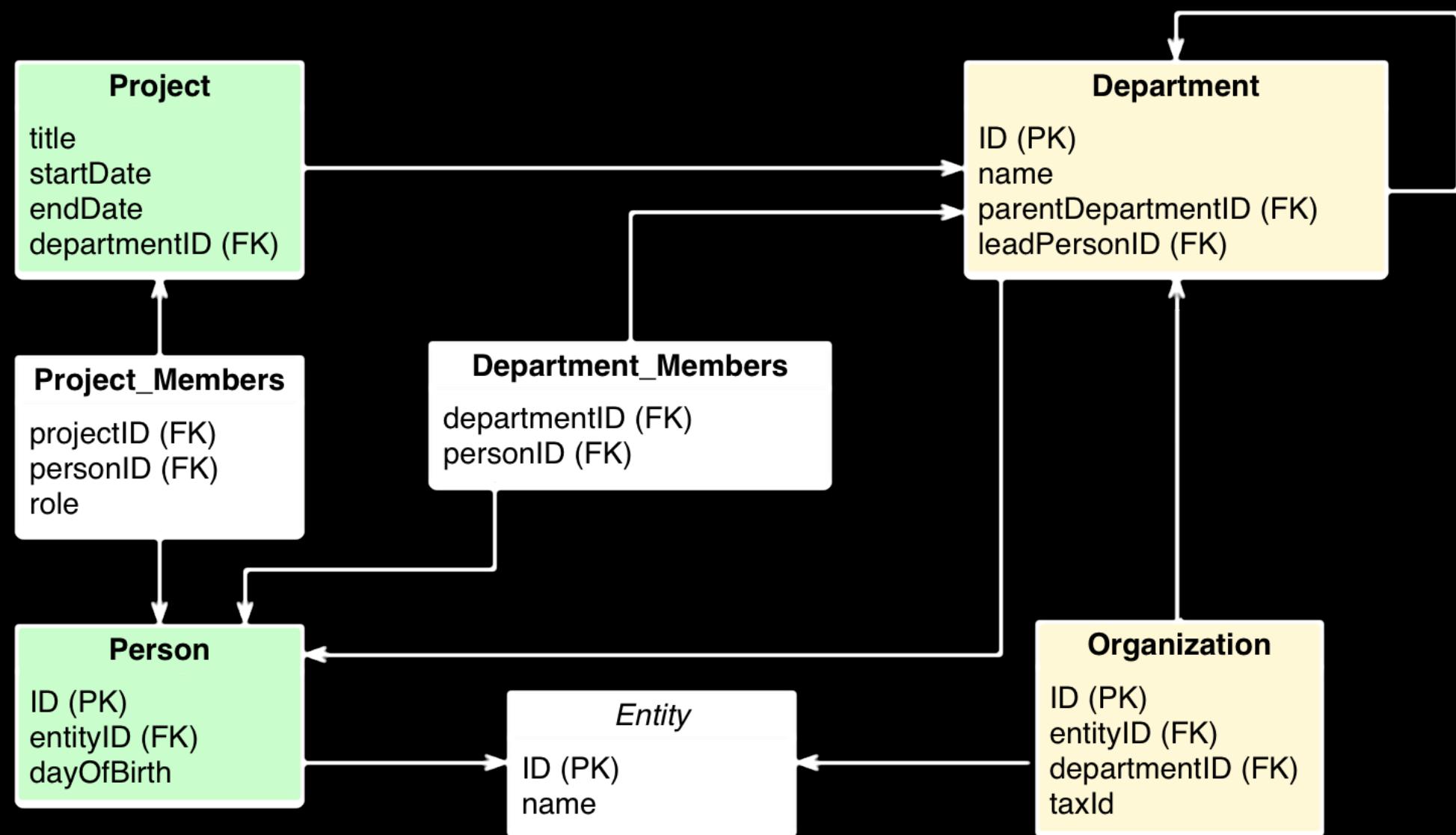
# CRUD

Create, Read, Update, Delete

*foco do nosso projeto*

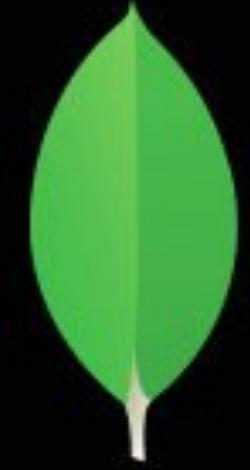
Comandos Básicos em um Banco de Dados

# relacional



# não-relacional

```
{ "order":{ "id":450789469, "total_price":"409.94", "name": "#1001", "line_items": [ { "id":466157049, "variant_id":39072856, "title": "IPod Nano - 8gb", "quantity":1, "price": "199.00", "grams":200, "sku": "IPOD2008GREEN", "variant_title": "green", "vendor":null, "fulfillment_service": "manual", "product_id":632910392, "requires_shipping":true, "taxable":true, "gift_card":false, "name": "IPod Nano - 8gb - green", "variant_inventory_management": "shopify", "location": "Default Location", "metafields": [ { "key": "Custom Field 1", "value": "Value 1" }, { "key": "Custom Field 2", "value": "Value 2" } ] } ] } }
```



mongoDB®

Banco de Dados Não-Relacional MongoDB

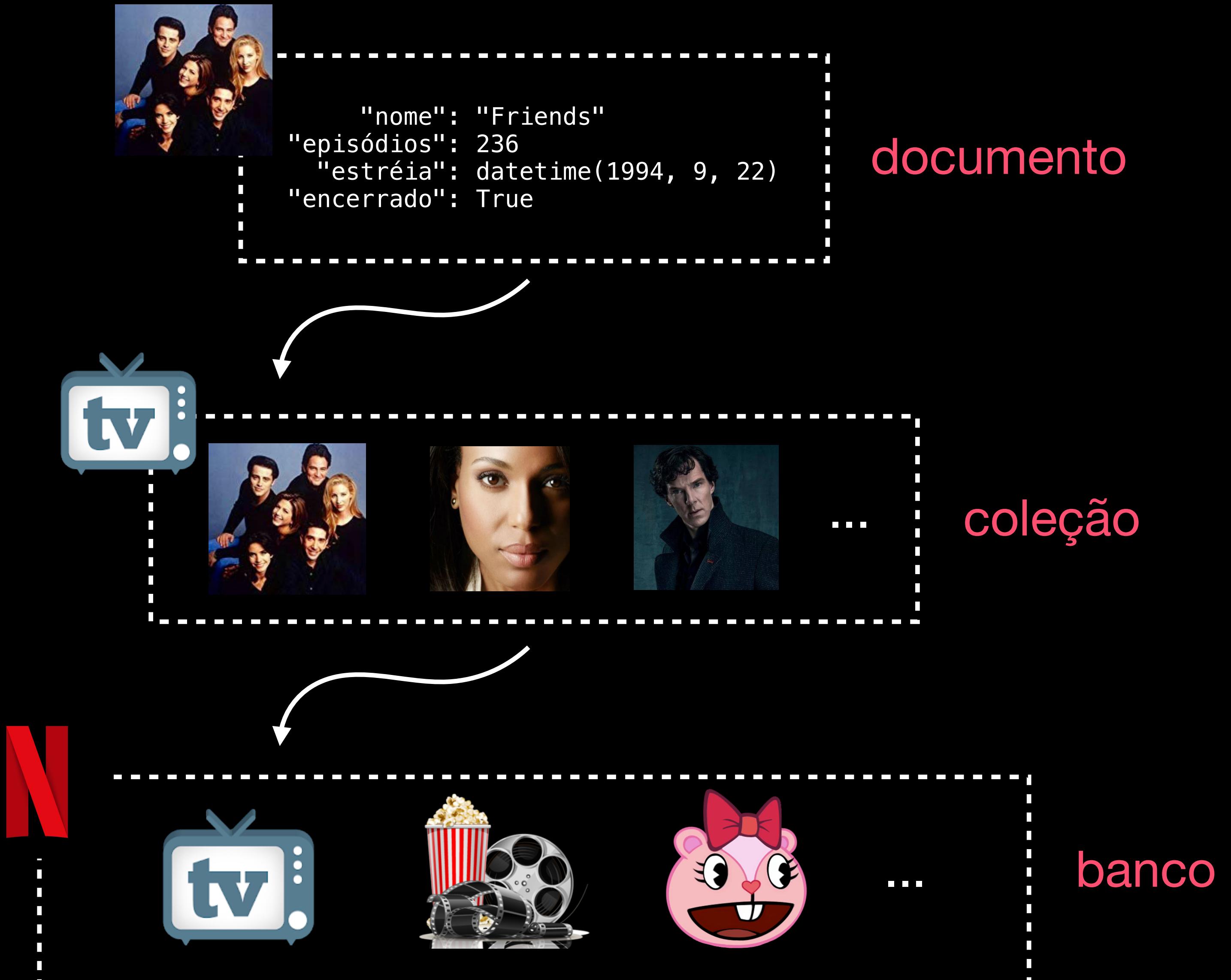
"nome": "Friends"

"episódios": 236

"estréia": datetime(1994, 9, 22)

"encerrado": True

Dados na Forma de Chave → Valor



Organização de Dados dentro do MongoDB

The screenshot shows a web browser window with the URL [api.mongodb.com/python/3.0.3/](http://api.mongodb.com/python/3.0.3/) in the address bar. The page title is "PyMongo 3.0.3 Documentation". On the left side, there is a sidebar with a "Table Of Contents" section listing various documentation topics like Overview, Getting Help, Issues, Contributing, Changes, About This Documentation, and Indices and tables. Below that are sections for "Next topic" (Installing / Upgrading) and "This Page" (Show Source). At the bottom of the sidebar is a "Quick search" input field with a "Go" button. The main content area features a large heading "PyMongo 3.0.3 Documentation" followed by a sub-section "Overview". A paragraph explains that PyMongo is a Python distribution for MongoDB. Below this are several links: "Installing / Upgrading" (with a sub-note about instructions), "Tutorial" (with a note about a quick overview), "Examples" (with a note about specific tasks), "Frequently Asked Questions" (with a note about common questions), "Python 3 FAQ" (with a note about Python 3 support), and "API Documentation".

PyMongo 3.0.3 documentation » [next](#) | [modules](#) | [index](#)

## Table Of Contents

- PyMongo 3.0.3 Documentation
  - Overview
  - Getting Help
  - Issues
  - Contributing
  - Changes
  - About This Documentation
  - Indices and tables

«

### Next topic

[Installing / Upgrading](#)

### This Page

[Show Source](#)

### Quick search

[Go](#)

# PyMongo 3.0.3 Documentation

## Overview

PyMongo is a Python distribution containing tools for working with MongoDB, and is the recommended way to work with MongoDB from Python. This documentation attempts to explain everything you need to know to use PyMongo.

*[Installing / Upgrading](#)*  
Instructions on how to get the distribution.

*[Tutorial](#)*  
Start here for a quick overview.

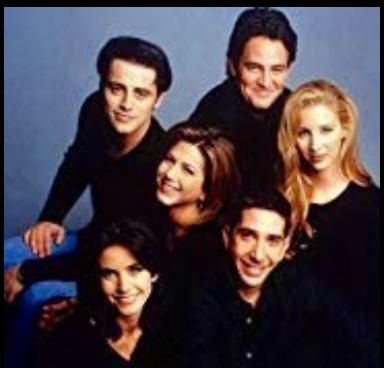
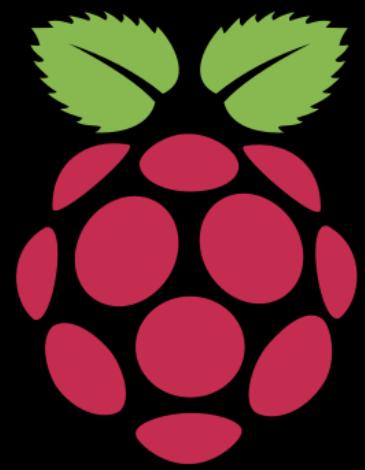
*[Examples](#)*  
Examples of how to perform specific tasks.

*[Frequently Asked Questions](#)*  
Some questions that come up often.

*[Python 3 FAQ](#)*  
Frequently asked questions about python 3 support.

*[API Documentation](#)*

```
>>> from pymongo import MongoClient  
# cria um vínculo cliente/servidor com o MongoDB  
>>> cliente = MongoClient("localhost", 27017)  
# carrega banco, ou cria um se não existir  
>>> banco = cliente["netflix"]  
# carrega coleção, ou cria uma se não existir  
>>> colecao = banco["séries"]
```



inserir 1 documento



inserir vários documentos

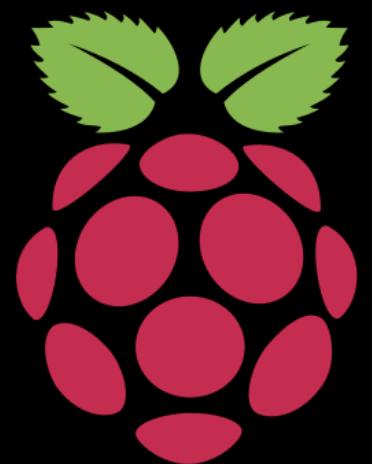


Banco  
de Dados

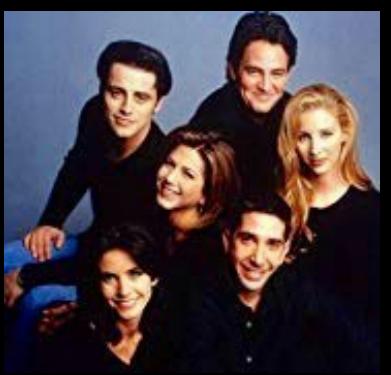
Inserção de Dados

```
>>> from datetime import datetime  
  
>>> documento1 = {"nome": "Friends", "episódios": 236,  
"estréia": datetime(1994, 9, 22), "encerrado": True}  
  
>>> documento2 = {"nome": "Scandal", "episódios": 124,  
"estréia": datetime(2012, 4, 5), "encerrado": True}  
  
>>> documento3 = {"nome": "Sherlock", "episódios": 13,  
"estréia": datetime(2010, 7, 25), "encerrado": False}  
  
>>> colecao.insert(documento1)  
  
>>> colecao.insert( [documento2, documento3] )
```

```
>>> documento4_incompleto = {"nome": "Lord of the Rings"}  
# sem problemas se não tiver todos os campos  
>>> colecao.insert(documento4_incompleto)  
  
>>> documento5 = {"nome": "Black Mirror", "episódios": 19,  
"estréia": datetime(2011, 12, 4), "encerrado": False}  
>>> colecao.insert(documento5)  
>>> colecao.insert(documento5)  
Traceback (most recent call last):  
...  
pymongo.errors.DuplicateKeyError: E11000 duplicate key  
error collection: netflix.séries index: _id_ dup key
```



buscar 1 documento  
com nome = "Friends"



buscar todos os documentos  
com encerrado = False



Busca de Dados

```
>>> busca = {"nome": "Friends"}  
>>> documento = colecao.find_one(busca)  
>>> documento  
{'_id': ObjectId('5b7e05ff530a69054262bba5'), 'nome':  
'Friends', 'episódios': 236, 'estréia':  
datetime.datetime(1994, 9, 22, 0, 0), 'encerrado': True}  
>>> documento["episódios"]  
236
```

Caso haja vários documentos com o critério, a `find_one` retornará o primeiro que ela encontrar



```
>>> busca = {"encerrado": False}  
>>> documento = colecao.find_one(busca)  
>>> documento["nome"]  
'Sherlock'  
  
# busca mais específica  
>>> busca2 = {"encerrado": False, "episódios": 19}  
>>> documento2 = colecao.find_one(busca2)  
>>> documento2["nome"]  
'Black Mirror'
```

```
>>> busca = {"encerrado": False}  
>>> resultado = colecao.find(busca)  
>>> resultado  
<pymongo.cursor.Cursor object at 0x10bba80f0>  
>>> lista_de_documentos = list( colecao.find(busca) )  
>>> lista_de_documentos  
[{..., 'nome': 'Sherlock', 'episódios': 13, 'estréia':  
datetime.datetime(2010, 7, 25, 0, 0), 'encerrado': False},  
{..., 'nome': 'Black Mirror', 'episódios': 19, 'estréia':  
datetime.datetime(2011, 12, 4, 0, 0), 'encerrado': False}]  
>>> for documento in lista_de_documentos:  
...     print(documento["nome"] )
```

...

Sherlock

Black Mirror

```
>>> busca = {"nome": "Community"}  
  
>>> documento = colecao.find_one(busca)  
>>> documento  
None  
  
>>> documentos = list( colecao.find(busca) )  
>>> documentos  
[]
```



Buscas com Nenhum Resultado

Busca	Comando
valor exato	{ "chave": valor }
valor diferente de	{ "chave": { "\$ne": valor } }
valor maior que	{ "chave": { "\$gt": valor } }
valor maior ou igual que	{ "chave": { "\$gte": valor } }
valor menor que	{ "chave": { "\$lt": valor } }
valor menor ou igual que	{ "chave": { "\$lte": valor } }
valor dentro de uma lista de opções	{ "chave": { "\$in": lista } }
valor fora de uma lista de opções	{ "chave": { "\$nin": lista } }

Outras Comparações de Busca

```
>>> busca = {"episódios": {"$lt": 15}}
```

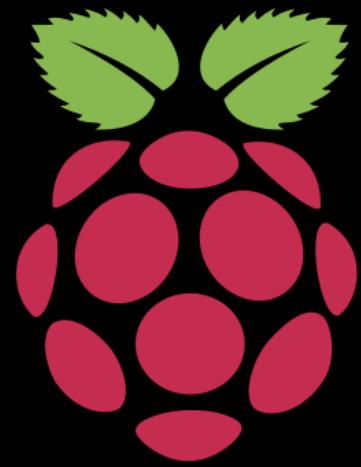
```
>>> colecao.find_one(busca)
{'_id': ..., 'nome': 'Sherlock', 'episódios': 13,
'estréia': datetime.datetime(2010, 7, 25, 0, 0),
'encerrado': False}
```

```
>>> busca2 = {"episódios": {"$gt": 100, "$lt": 300}}
```

```
>>> series = list( colecao.find(busca2) )
>>> for serie in series:
...     print(serie["nome"])

...
'Friends'
'Scandal'
```

```
>>> dez_anos_atras = datetime.now() - timedelta(years=10)
>>> busca = {"estreia": {"$lt": dez_anos_atras}}
>>> series = list( colecao.find(busca) )
>>> for serie in series:
...     print(serie["nome"])
...
'Friends'
```



buscar todos os seriados  
com encerrado = False  
em **ordem crescente** de estréia



buscar todos os seriados  
com encerrado = False  
em **ordem decrescente** de estréia



Busca de Dados com Ordenação

```
>>> from pymongo import ASCENDING, DESCENDING  
  
>>> busca = {"encerrado": False}  
  
>>> ordenacao = [ ["estréia", ASCENDING] ]  
  
>>> list( colecao.find(busca, sort=ordenacao) )  
[..., 'nome': 'Sherlock', 'episódios': 13, 'estréia':  
datetime.datetime(2010, 7, 25, 0, 0), 'encerrado': False},  
..., 'nome': 'Black Mirror', 'episódios': 19, 'estréia':  
datetime.datetime(2011, 12, 4, 0, 0), 'encerrado': False}]  
  
>>> ordenacao = [ ["estréia", DESCENDING] ]  
  
>>> list( colecao.find(busca, sort=ordenacao) )  
[..., 'nome': 'Black Mirror', 'episódios': 19, 'estréia':  
datetime.datetime(2011, 12, 4, 0, 0), 'encerrado': False},  
..., 'nome': 'Sherlock', 'episódios': 13, 'estréia':  
datetime.datetime(2010, 7, 25, 0, 0), 'encerrado': False}]
```

```
>>> from pymongo import ASCENDING, DESCENDING
>>> busca = {"encerrado": False}
>>> ordenacao = [ ["estreia", ASCENDING] ]
>>> colecao.find_one(busca, sort=ordenacao)
{..., 'nome': 'Sherlock', 'episódios': 13, 'estreia':
datetime.datetime(2010, 7, 25, 0, 0), 'encerrado': False}

>>> ordenacao = [ ["estreia", DESCENDING] ]
>>> colecao.find_one(busca, sort=ordenacao)
{..., 'nome': 'Black Mirror', 'episódios': 19, 'estreia':
datetime.datetime(2011, 12, 4, 0, 0), 'encerrado': False}
```

# Resumo da Ópera

## Funcionalidade

### Datas e Horários

[acessar documentação](#)

## Comandos

```
from datetime import datetime, timedelta
tempo = datetime(2018, 3, 28, 15, 35, 12) • datetime.now()
intervalo = timedelta(months=4) • tempo2 = tempo + intervalo
tempo2 > tempo • intervalo.seconds • tempo.strftime("%H:%M")
```

### Campainha

[acessar documentação](#)

```
from gpiozero import Buzzer • buzzer = Buzzer(16)
buzzer.on() • buzzer.off() • buzzer.toggle()
buzzer.is_active • buzzer.beep()
buzzer.beep(n=4, on_time=0.5, off_time=2)
```

### Sensor de Distância

[acessar documentação](#)

```
from gpiozero import DistanceSensor
sensor = DistanceSensor(trigger=17, echo=18)
sensor.distance • sensor.threshold_distance
s.when_in_range = funcao • s.when_out_of_range = funcao
```

### MongoDB

[acessar documentação](#)

```
from pymongo import MongoClient, ASCENDING, DESCENDING
cliente = MongoClient("localhost", 27017)
banco = cliente["nome"] • colecao = banco["nome"]
dados = {"nome": "Jan K. S.", "idade": 32}
colecao.insert(dados) • colecao.insert([dados1, dados2])
busca = {"chave1": valor1, "chave2": {"$gt": valor2}}
ordenacao = [ ["idade", DESCENDING] ]
documento = colecao.find_one(busca, sort=ordenacao)
documentos = list( colecao.find(busca, sort=ordenacao) )
```

## Funcionalidade

## Comandos

Emissor  
Infravermelho

```
from py_irsend.irsend import *
controles = list_remotes() • codigos = list_codes("mini")
send_once("mini", ["KEY_1", "KEY_2"] )
```

Receptor  
Infravermelho

```
from lirc import init, nextcode
init("aula", blocking=False)
codigo = nextcode() • codigo == ["KEY_1"]
```

Servidor Flask  
acessar documentação

```
from flask import Flask
app = Flask(__name__)

@app.route("/")
def mostrar_inicio():
    return "Bem-vindo!"

@app.route("/ contato")
def mostrar_contato():
    return "janks@puc-rio.br"

@app.route("/numero/<int:x>")
def mostrar_numero(x):
    return "x = " + str(x)

return
redirect("/outrapagina")

return
render_template("index.html")

app.run(port=5000, debug=False)
```

HTML

```
<p>Parágrafo</p>

<a href="/pagina">Link</a>
```

```
<ul>
    <li>Item 1 da Lista</li>
    <li>Item 2 da Lista</li>
</ul>
```

Ngrok

abrir Terminal → ngrok http 5000

## Funcionalidade

## Comandos

LED

[acessar documentação](#)

```
from gpiozero import LED • led = LED(21)
led.on() • led.off() • led.toggle() • led.is_lit
led.blink() • led.blink(n=4, on_time=0.5, off_time=2)
```

Botão

[acessar documentação](#)

```
from gpiozero import Button • botao = Button(11)
botao.is_pressed • botao.wait_for_press()
botao.when_pressed = funcao
botao.when_released = funcao
botao.when_held = funcao
```

LCD

[acessar documentação](#)

```
from Adafruit_CharLCD import Adafruit_CharLCD
lcd = Adafruit_CharLCD(2, 3, 4, 5, 6, 7, 16, 2)
lcd.message("Texto 1\nTexto 2")
lcd.clear()
```

MPlayer

```
from mplayer import Player • player = Player()
player.loadfile("Musica.mp3") • player.loadlist("lista.txt")
player.pause() • player.paused • player.quit()
player.time_pos = 2 • player.length • player.pt_step(-1)
player.metadata["Title"] • player.metadata["Artist"]
player.volume = 70 • player.speed = 2
```

Funcionalidade	Comandos
Funções	<pre>x = input("Digite um número: ") • print("Resultado: ", x) from time import sleep • sleep(0.5)</pre>
Listas <a href="#">acessar documentação</a>	<pre>lista = [1, 2, 3] • lista2 = ["texto", [0, 0], 5] lista[0] • total_de_elementos = len(lista) lista.append(novo_elemento) • lista.remove(indice)</pre>
Dicionários <a href="#">acessar documentação</a>	<pre>dicionario = {"chave 1": 42, "chave 2": [1, 2, 3]} dicionario["chave 1"] • dicionario["chave 3"] = "Olá!"</pre>
Textos (Strings) <a href="#">acessar documentação</a>	<pre>texto = "olá!" • len(texto) • caractere = texto[0] texto + "\n" • texto + str(numero) • "x = %.2f" % numero</pre>
Condicionais	<pre>if x != 0:     if x not in [1, 2]:         y = 4     elif x &gt;= 0:         y = 3 else:     y = 0</pre>
Repetições	<pre>for elem in lista:     ... for i in range(1, 4):     ... while x &gt; 1:     ...</pre>
Criação de Funções	<pre>def funcao1(x):     return x + 2 def funcao2(x, y, z):     ... def funcao3():     global x</pre>