

Projeto 03

Controle de Acesso

Jan K. S. – janks@puc-rio.br

ENG1419 – Programação de Microcontroladores

Ferramentas do Python



Datas e Horas

```
>>> from datetime import date
>>> data = date(2015, 10, 21)
>>> hoje = date.today()
>>> hoje
datetime.date(2018, 3, 26)

>>>
>>> from datetime import datetime
>>> data_e_hora = datetime(2015, 10, 21, 16, 29, 0)
>>> agora = datetime.now()
>>> agora
datetime.datetime(2018, 3, 26, 15, 11, 41, 111740)
```

```
>>> from datetime import datetime, timedelta
>>> agora = datetime.now()
>>> agora
datetime.datetime(2018, 3, 26, 15, 12, 35, 111740)
>>> daqui_a_uma_hora = agora + timedelta(hours=1)
>>> daqui_a_uma_hora
datetime.datetime(2018, 3, 26, 16, 12, 35, 111740)
>>> agora - timedelta(days=300)
datetime.datetime(2017, 5, 29, 15, 12, 35, 111740)
>>> daqui_a_uma_hora - agora
datetime.timedelta(0, 3600)
```

```
>>> from datetime import datetime,  
>>> agora = datetime.now()  
>>> str(agora.day) + "/" + str(agora.month)  
'26/3'  
  
>>> from time import strftime  
>>> 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¹
한국어²
中文³

7 Referências
8 Bibliografia
9 Ligações externas

Mecanismo [editar | editar código-fonte]

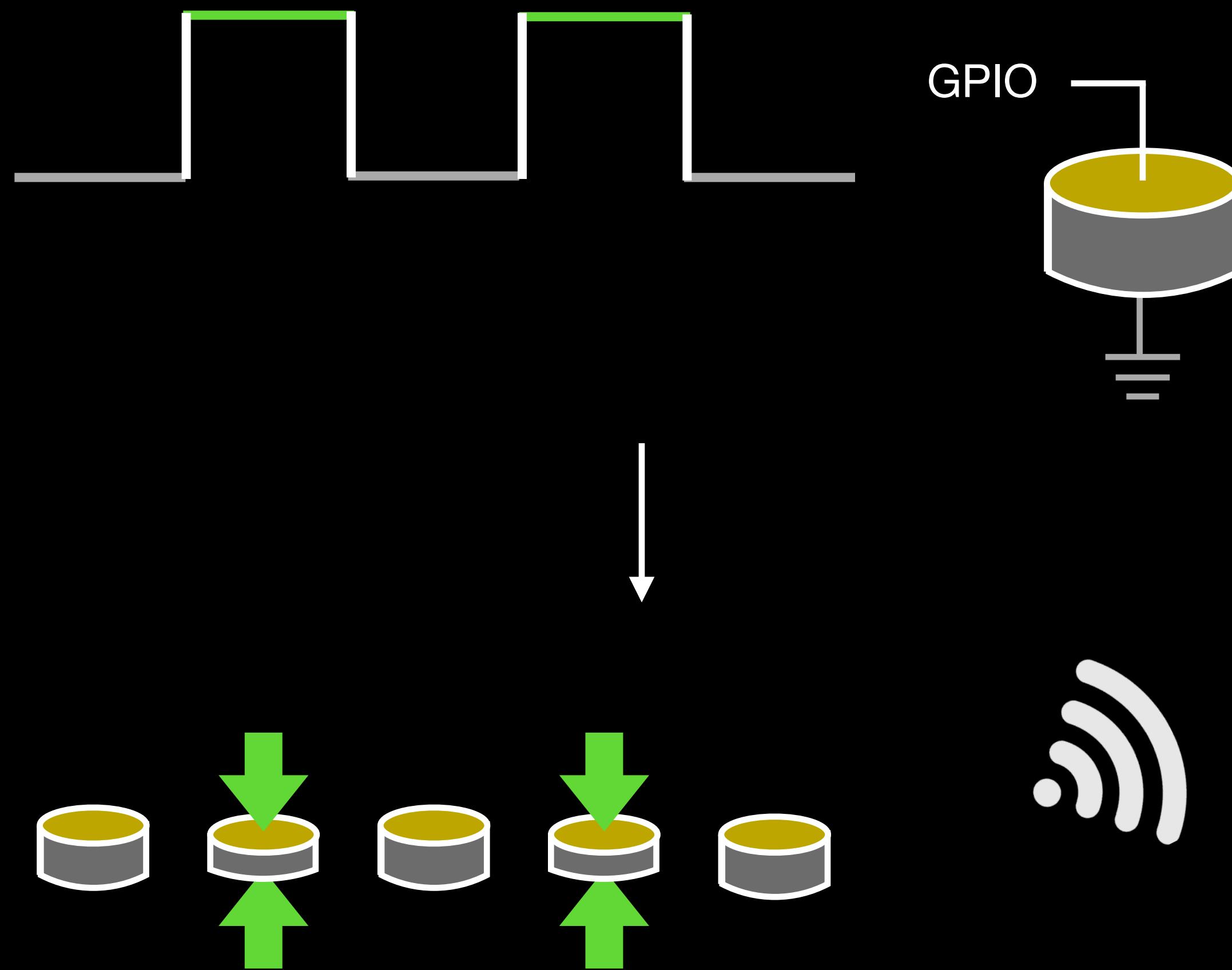
O efeito piezoelétrico é entendido como a interação eletromecânica linear entre a força mecânica e o estado elétrico ([forças de Coulomb](#)) em materiais cristalinos ([cerâmicos](#), [polímeros](#)).

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](#).^[1]

Um disco piezoelétrico gera uma diferença de potencial quando deformado.

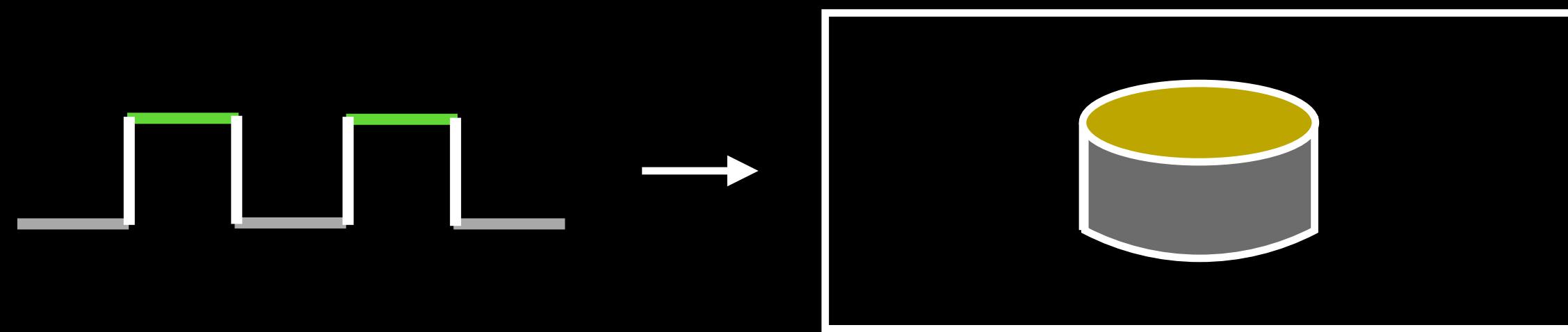
Cristais [editar | editar código-fonte]

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.

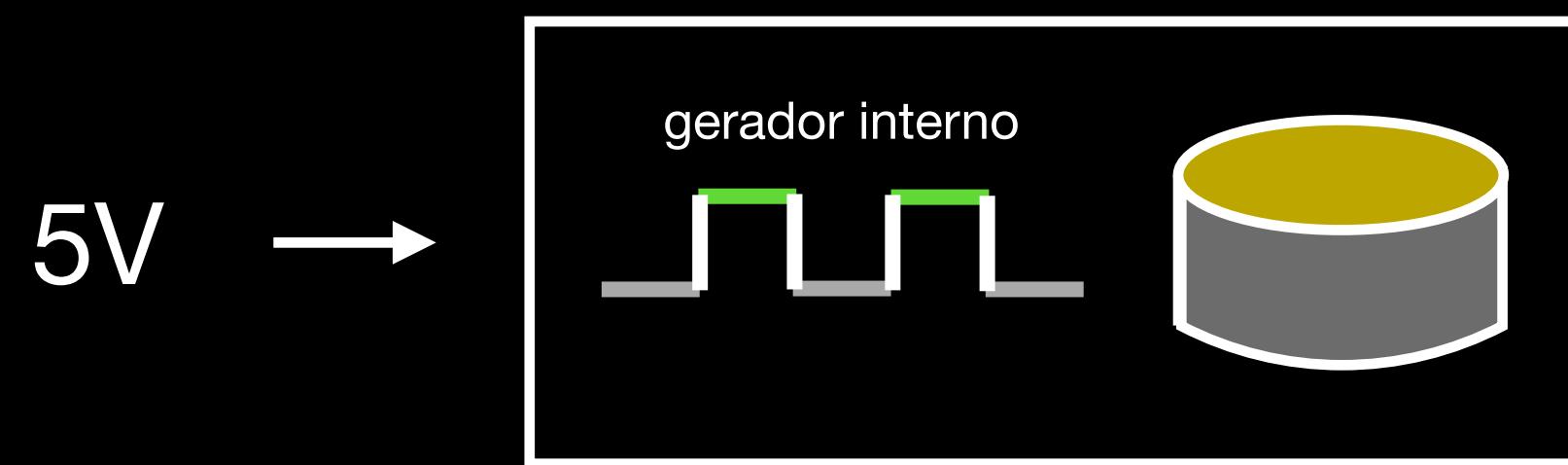


Geração de Som por Pulso

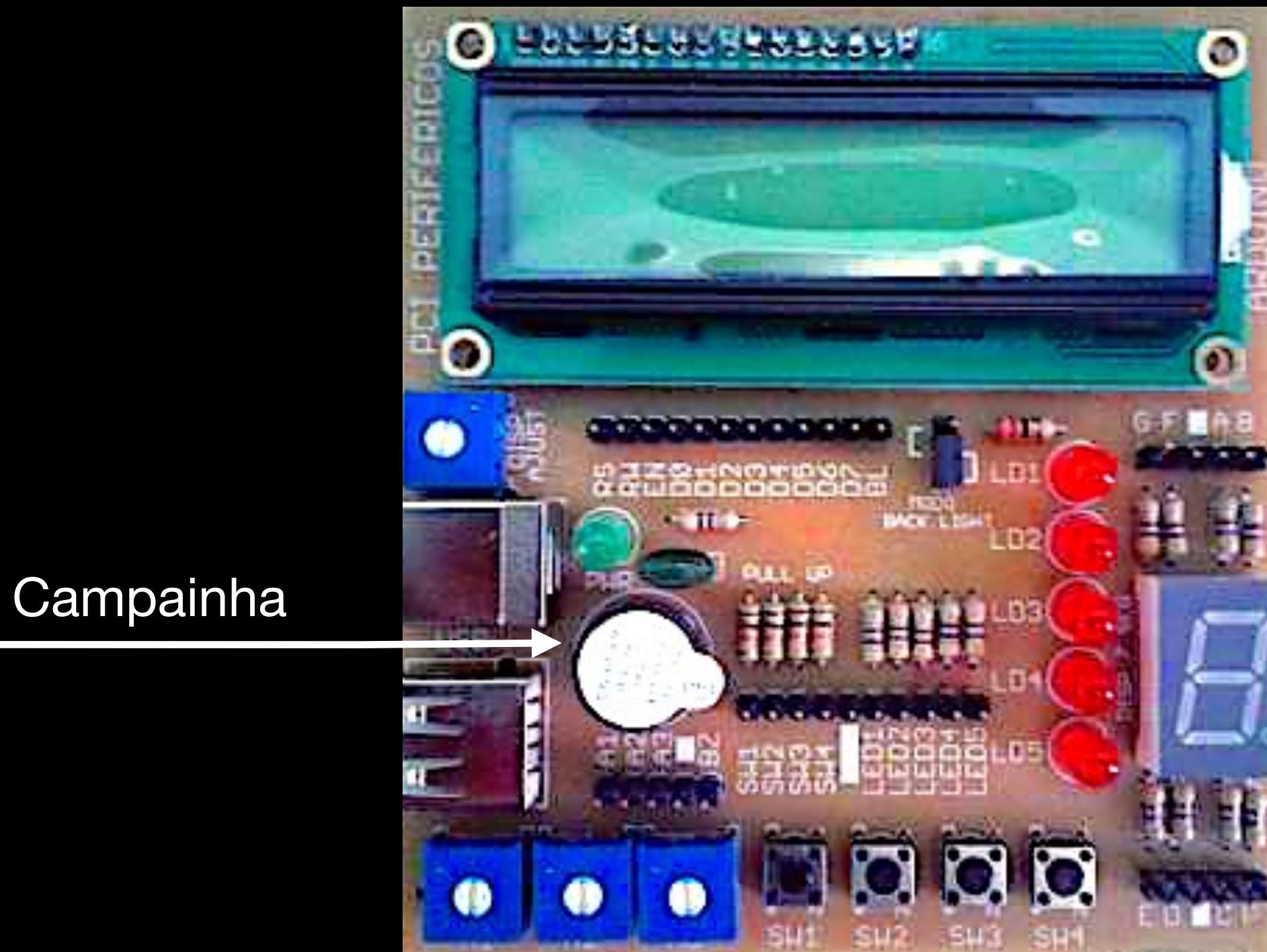
campainha passiva



campainha ativa



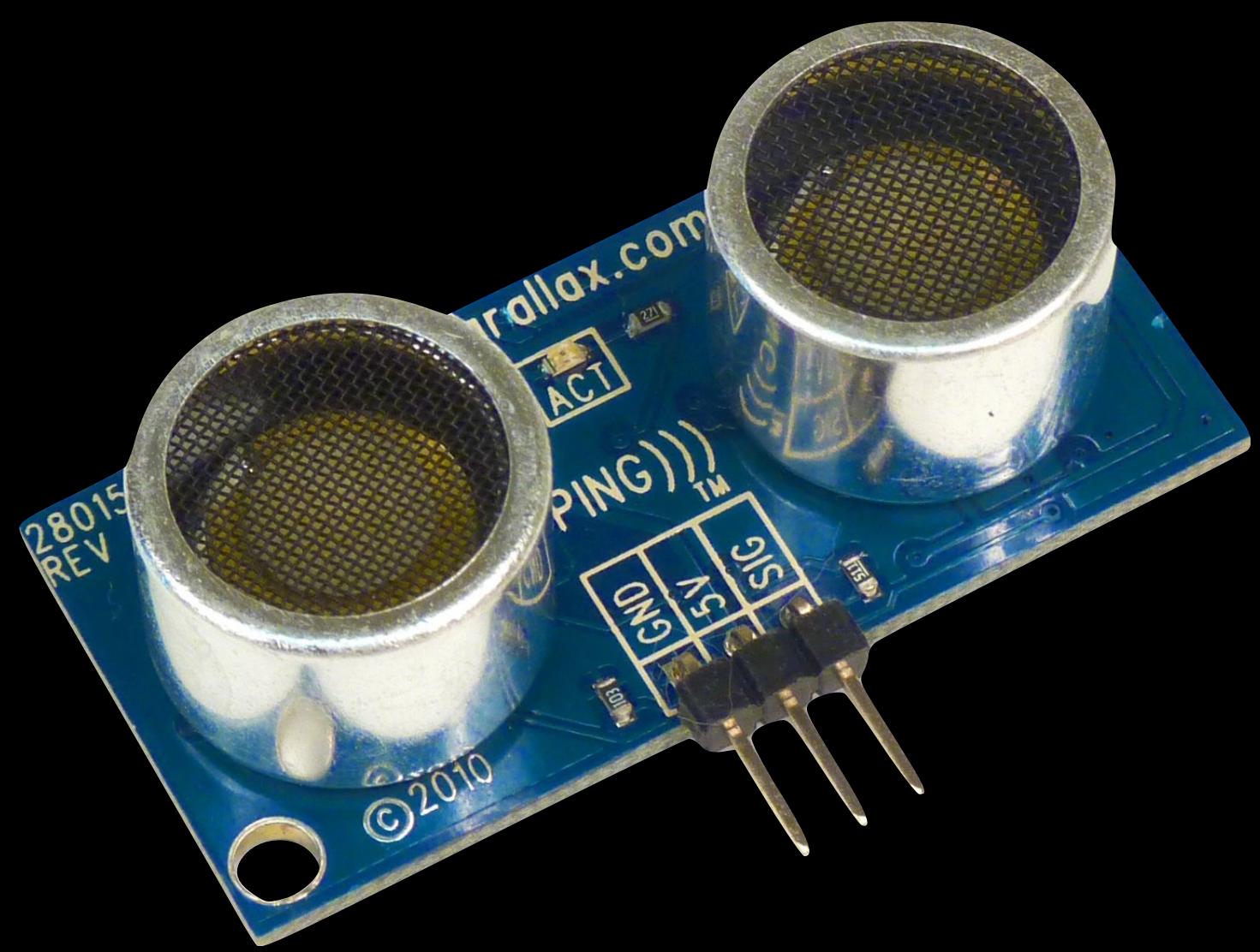
Campainhas Ativas e Passivas



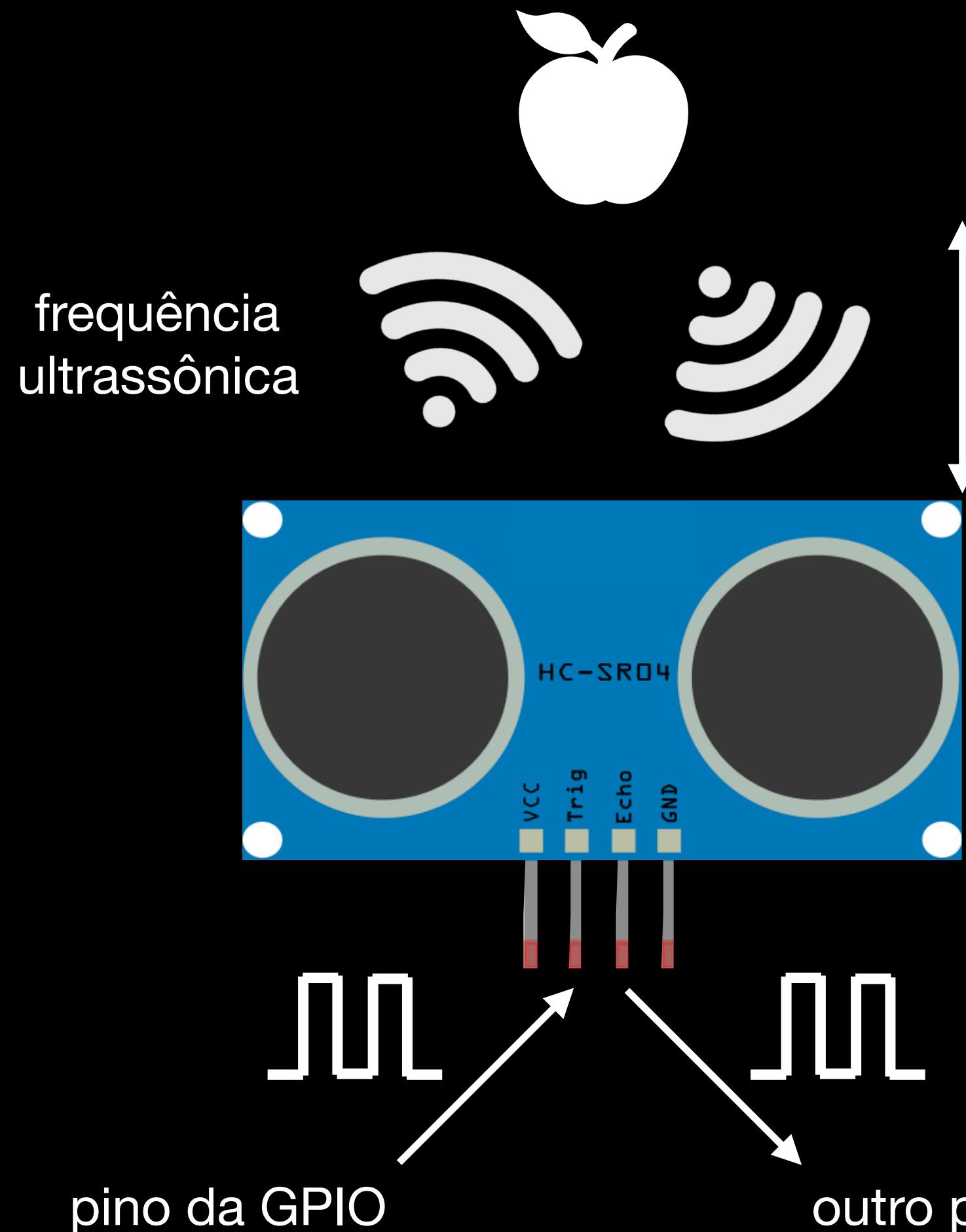
Campainha Ativa na Placa de Periféricos Básicos

```
>>> 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)  
>>> sirene.beep(n=2, background=False)
```

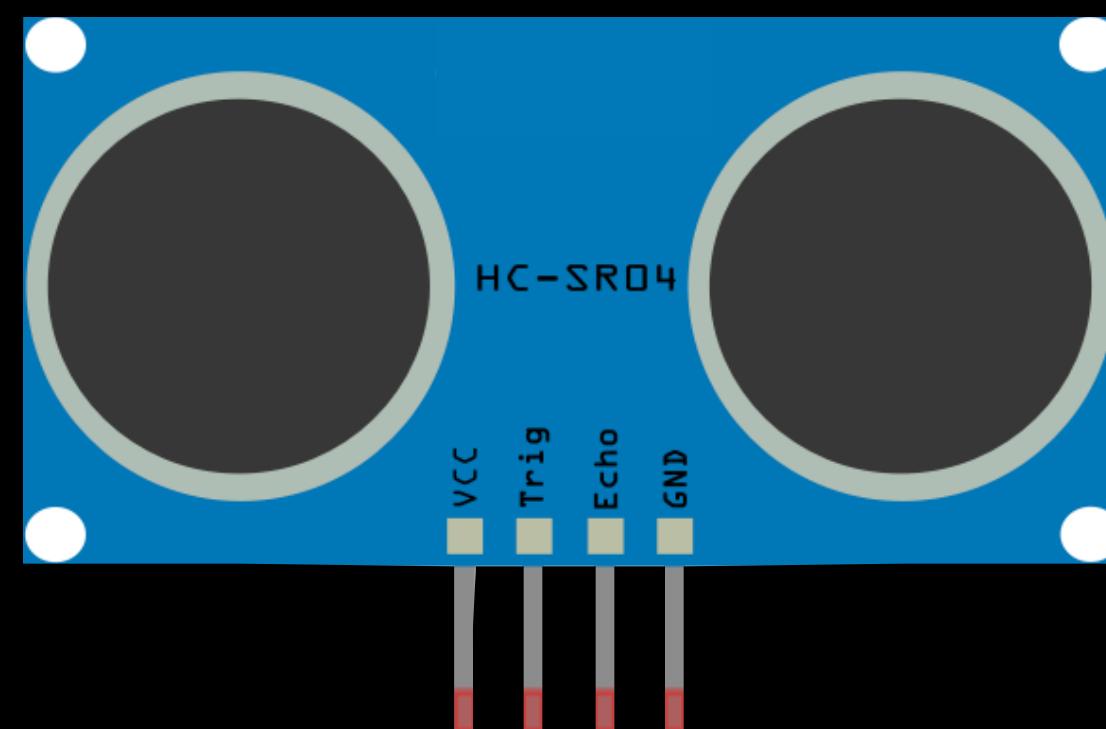


Sensor de Distância Ultrassônico



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

Cálculo da Distância pelo Tempo Entre Emissão e Eco

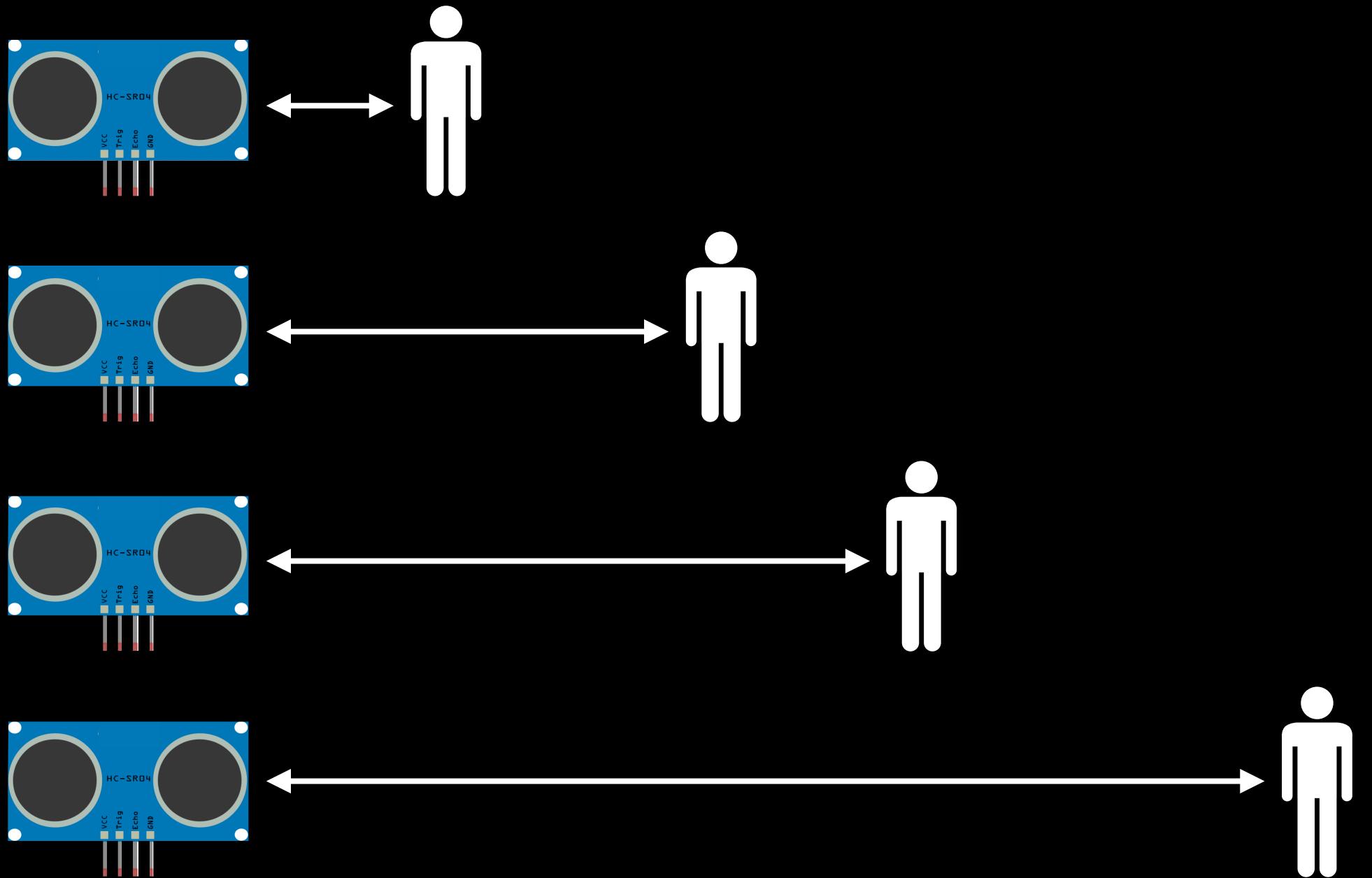


GPIO 17

GPIO 18

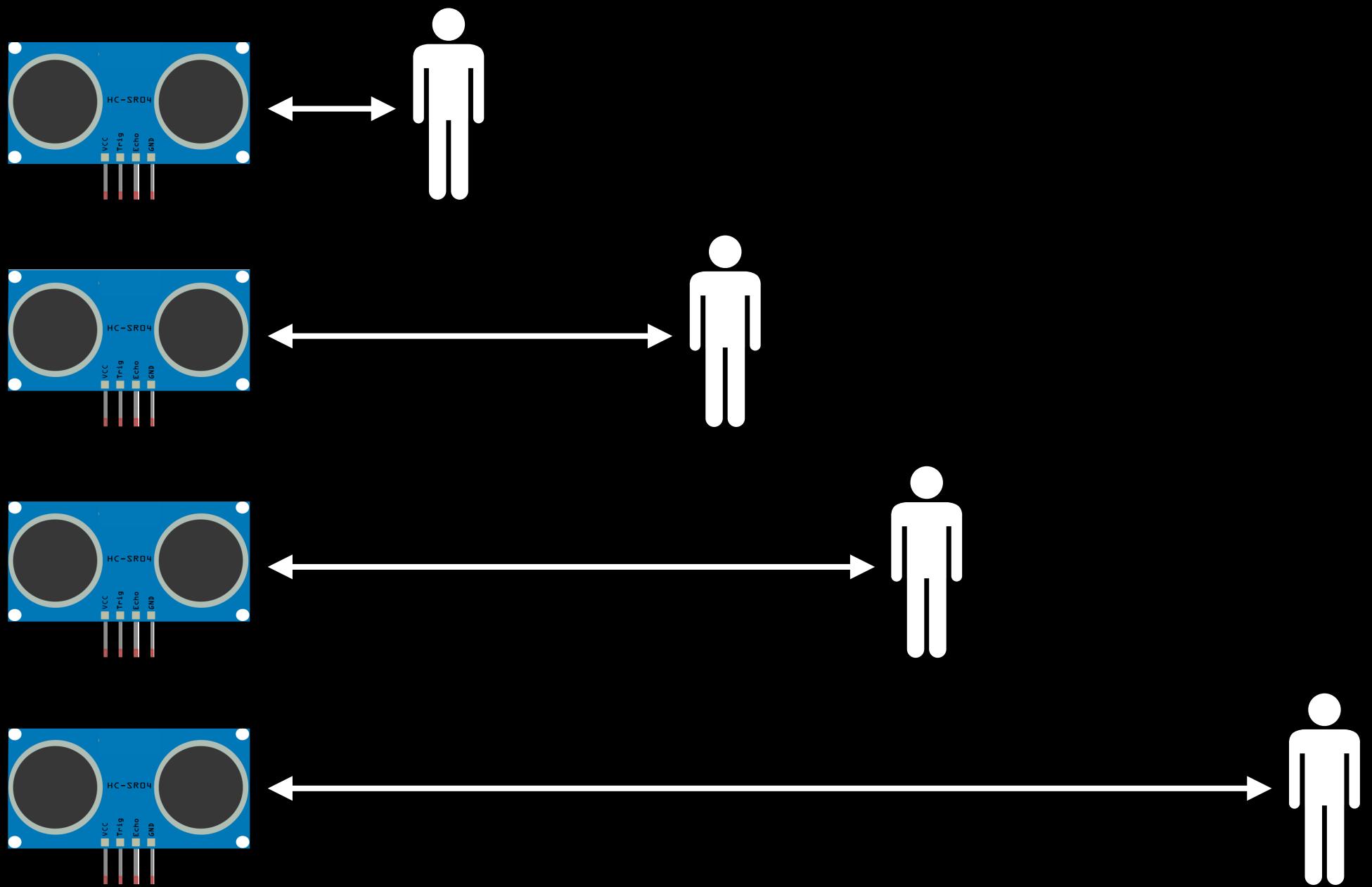
Pinos Usados pela Nossa Placa

```
>>> 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  
  
>>> 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

```
>>> from gpiozero import DistanceSensor  
>>> sensor = DistanceSensor(trigger=17, echo=18)  
>>> sensor.wait_for_in_range()  
>>> sensor.wait_for_out_of_range()  
>>> sensor.threshold_distance = 0.5 ← distância limite para proximidade
```

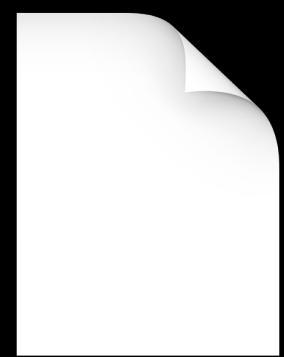
```
>>> from gpiozero import DistanceSensor  
>>> sensor = DistanceSensor(trigger=17, echo=18)  
>>> def alertar_proximidade():  
...     print("Objeto próximo!")  
...  
>>> sensor.when_in_range = alertar_proximidade  
>>> led = LED(21)  
>>> sensor.when_out_of_range = led.toggle
```

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

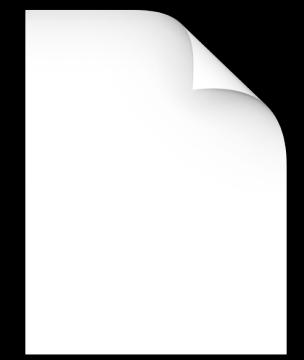
100 kB

com o tempo



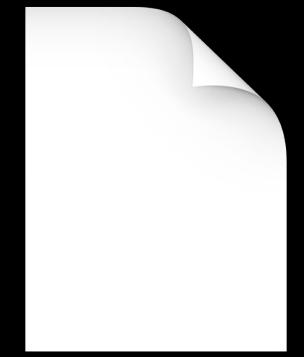
meus_dados

1.2 GB



meus_dados_2017

20 MB



meus_dados_2018

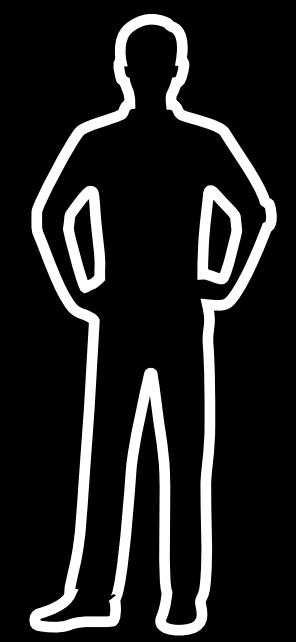
12 MB

...

Complicado de
gerenciar...



Armazenamento de Dados em Arquivos



comandos

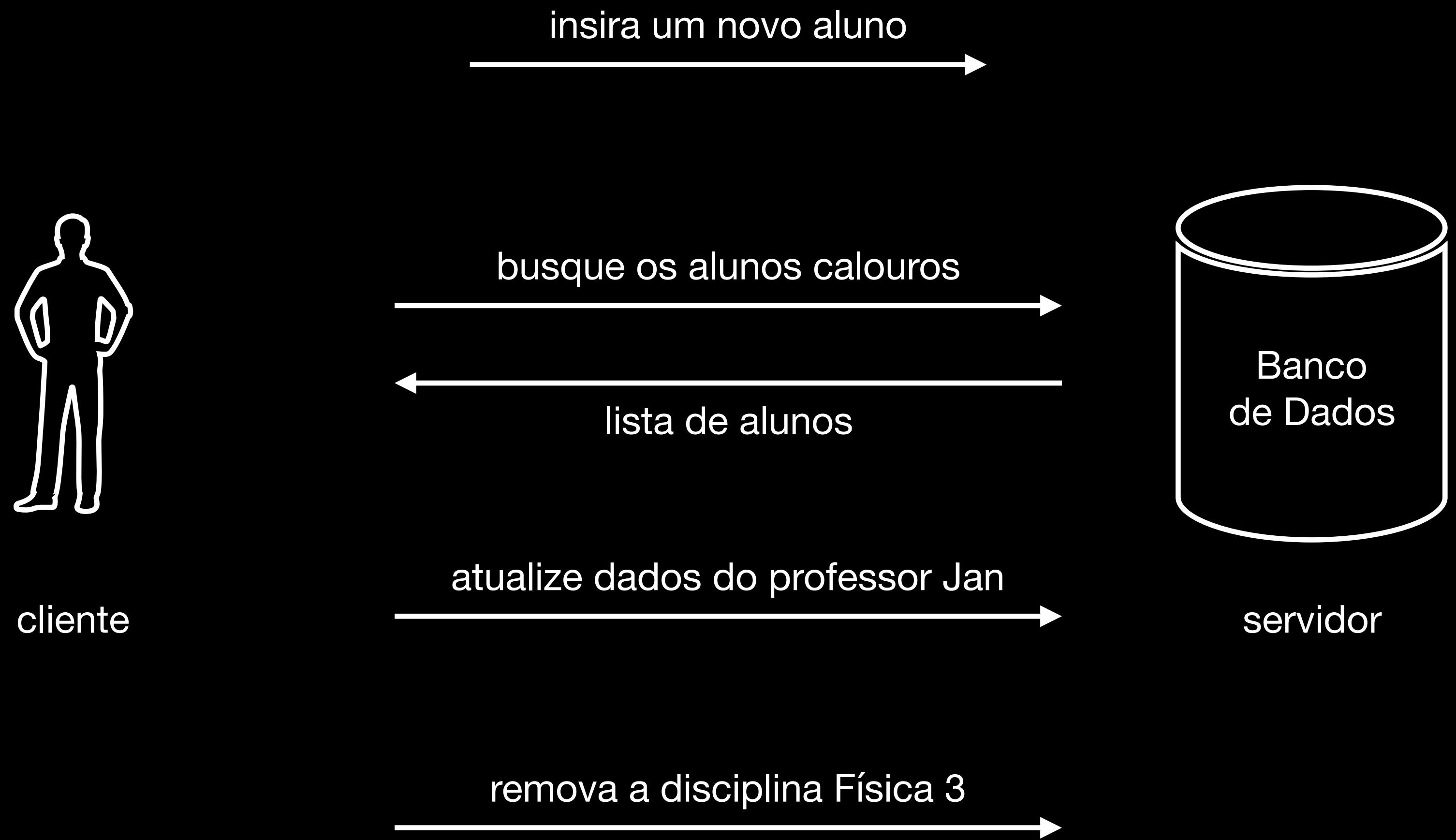
A simple black arrow pointing from the person icon towards the central cylinder.

Banco de Dados

CRUD

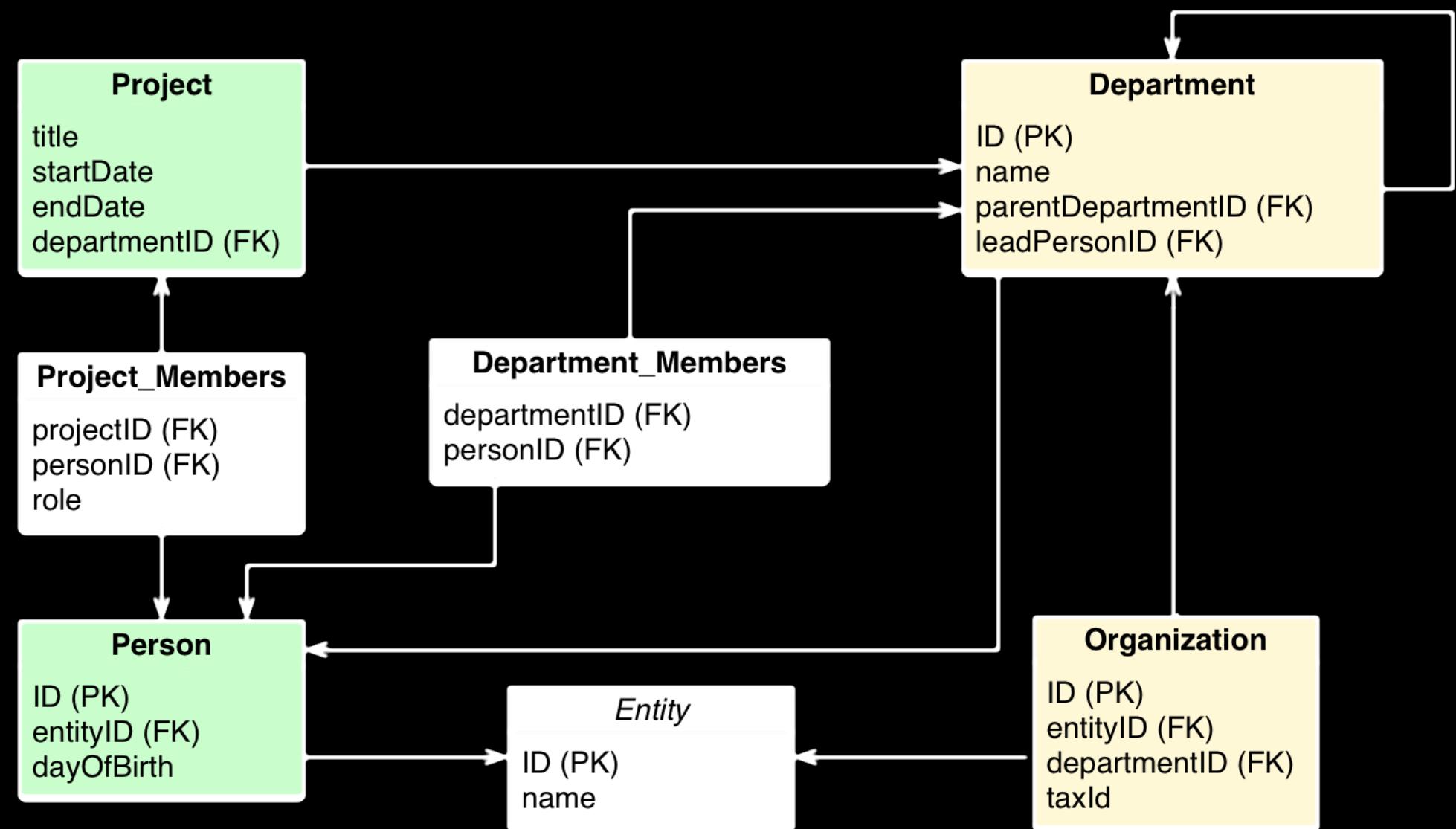
Create, Read, Update, Delete

Comandos Básicos em um Banco de Dados



Exemplos de Comandos entre Cliente e Servidor de 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",  
      }  
    ]  
  }  
}
```

Exemplos de Tipos de Bancos de Dados



Banco de Dados Não-Relacional MongoDB

```
"nome": "Paul McCartney"  
"nascimento": date(1942, 6, 18)  
"vivo": True  
"filhos": ["Mary", "Stela", "James"]  
"Grammys": 18
```

Dados na Forma de Chave → Valor

```
{  
  "título": "Master Maqui",  
  "artista": "Rodrigo y Gabriela",  
  "duração": 306,  
  "álbum": "11:11"  
}
```

documento

"músicas"

```
{  
  "título": "Master Maqui",  
  "artista": "Rodrigo y Gabriela",  
  "duração": 306,  
  "álbum": "11:11"  
}
```

coleção

"mídias"

"músicas"

"filmes"

"séries"

banco

Organização de Dados dentro do MongoDB

The screenshot shows a web browser window with the URL api.mongodb.com/python/3.0.3/ in the address bar. The page title is "PyMongo 3.0.3 Documentation". On the left, 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 it are sections for "Next topic" (Installing / Upgrading) and "This Page" (Show Source). A "Quick search" input field is also present. The main content area features a large heading "PyMongo 3.0.3 Documentation" followed by "Overview". It includes a paragraph about PyMongo being a Python distribution for MongoDB, and several links to other sections: "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), 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  
>>> cliente = MongoClient("localhost", 27017)  
>>> banco = cliente["mídias"] ← cria um banco novo se não existir  
>>> colecao = banco["músicas"] ← cria uma coleção nova se não existir
```

```
>>> documento1 = {"título": "Master Maqui", "artista":  
"Rodrigo y Gabriela", "álbum": "11:11"}  
  
>>> colecao.insert_one(documento1)  
  
>>> documento2 = {"título": "Summertime", "artista":  
"Chet Baker", "álbum": "My Favourite Songs"}  
  
>>> documento3 = {"título": "11:11", "artista":  
"Rodrigo y Gabriela", "álbum": "11:11"}  
  
>>> colecao.insert_many([documento2, documento3])
```

```
>>> documento4_incompleto = {"título": "Godzila"}  
>>> colecao.insert_one(documento4_incompleto)  
>>> documento5 = {"título": "Chat Perché", "artista":  
"Bombes 2 Bal", "álbum": "Danse Avec Ta Grand-Mère"}  
>>> colecao.insert_one(documento5)  
>>> colecao.insert_one(documento5)  
Traceback (most recent call last): ← erro ao tentar inserir o  
  File "<stdin>", line 1, in <module> mesmo documento de novo  
...  
pymongo.errors.DuplicateKeyError: E11000 duplicate key  
error collection: mídias.músicas index: _id_ dup key:  
{ : ObjectId('5ab7c0ef530a6913628f0852') }
```

```
>>> busca = {"título": "Summertime"}  
>>> documento_da_busca = colecao.find_one(busca)  
  
>>> documento_da_busca  
{'_id': ObjectId('5ab7ba0e530a6913628f084d'),  
'título': 'Summertime', 'artista': 'Chet Baker',  
'álbum': 'My Favourite Songs'}  
  
>>> documento_da_busca["artista"]  
'Chet Baker'  
  
>>>不存在 = colecao.find_one({"título": "Abc"})  
>>>不存在 == None  
True
```

```
>>> busca = {"artista": "Rodrigo y Gabriela"}  
>>> documentos_da_busca = colecao.find(busca)  
>>> documentos_da_busca  
<pymongo.cursor.Cursor object at 0x10bba80f0>  
>>> list(documentos_da_busca)  
[{..., "título": "11:11", "artista": "Rodrigo y  
Gabriela", "álbum": "11:11"},  
{..., "título": "Master Maqui", "artista": "Rodrigo y  
Gabriela", "álbum": "11:11"}]  
>>> for documento in documentos_da_busca:  
...     print(documento["título"] )  
  
...  
Master Maqui  
11:11
```

```
>>> colecao.count()
5

>>> busca = {"artista": "Rodrigo y Gabriela"}
>>> colecao.count(busca)
2
```

```
>>> busca = {"título": "Godzila"}  
>>> novo_titulo = {"título": "Godzilla"}  
>>> colecao.update_one(busca, {"$set": novo_titulo})  
>>> novo_campo = {"artista": "Blue Öyster Cult"}  
>>> colecao.update_one(busca, {"$set": novo_campo})  
>>> colecao.find_one(busca)  
{'_id': ObjectId('5ab9199979e38e0e1e9c1970'),  
'título': 'Godzilla', 'artista': 'Blue Öyster Cult'}  
>>> busca = {"artista": "Rodrigo y Gabriela"}  
>>> genero = {"gênero": "Flamenco / Rock"}  
>>> colecao.update_all(busca, {"$set": genero})
```

```
>>> busca = {"título": "Godzilla"}  
  
>>> colecao.delete_one(busca)  
{'n': 1, 'ok': 1.0} ← 1 documento removido  
  
>>> busca = {"artista": "Rodrigo y Gabriela"}  
  
>>> colecao.delete_many(busca)  
{'n': 2, 'ok': 1.0} ← 2 documentos removidos  
  
>>> colecao.drop() ← remove a coleção inteira  
  
>>> list( colecao.find() )  
[]
```

```
>>> from datetime import datetime, timedelta  
>>> colecao = banco["testes"]  
>>> doc1 = {"data": datetime.now(), "valor": 4}  
>>> futuro = datetime.now() + timedelta(days=10000)  
>>> doc2 = {"data": futuro, "valor": 10}  
>>> doc3 = {"data": None, "valor": 0}  
>>> colecao.insert_many([doc1, doc2, doc3])
```

operador "maior que"

(Greater Than)

operador "menor que"

(Less Than)

```
>>> colecao.find_one({"valor": {"$gt": 8}})  
{'_id': ..., 'valor': 10}
```

```
>>> colecao.find_one({"valor": {"$gt": 0, "$lt": 7}})  
{'_id': ..., 'valor': 4}
```

```
>>> colecao.find_one({"data": {"$gt": datetime.now()}})  
{'_id': ..., 'data': datetime(2045, 8, 11,  
13, 15, 25, 275000 )}
```

```
>>> from pymongo import ASCENDING, DESCENDING
>>> busca = {"valor": {"$gt": 0}}
>>> ordenacao = [ ("data", ASCENDING) ]
>>> list( colecao.find(busca, sort=ordenacao) )
[{'_id': ..., 'data': datetime.datetime(2018, 3, 26,
13, 15, 48, 836000)},
 {'_id': ..., 'data': datetime.datetime(2045, 8, 11,
13, 15, 25, 275000)}]
>>> ordenacao = [ ("data", DESCENDING) ]
>>> list( colecao.find(busca, sort=ordenacao) )
[{'_id': ..., 'data': datetime.datetime(2045, 8, 11,
13, 15, 25, 275000)},
 {'_id': ..., 'data': datetime.datetime(2018, 3, 26,
13, 15, 48, 836000)}]
```

Resumo da Ópera

Funcionalidade

Comandos

Emissor
Infravermelho

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

Receptor
Infravermelho

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

Servidor

```
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=True)
```

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

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

mais funções na documentação oficial

Botão

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

mais funções na documentação oficial

LCD

```
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()
```

mais funções no exemplo do repositório oficial

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.duration • player.pt_step(-1)
    player.metadata["Title"] • player.metadata["Artist"]
        player.volume = 70 • player.speed = 2
```

Funcionalidade	Comandos
Entrada / Saída	<pre>x = input("Digite um número: ") • print("Resultado: ", x)</pre>
Listas	<pre>lista = [1, 2, 3] • lista2 = ["texto", [0, 0], 5]; lista[0] • lista.append(elemento) • lista.remove(indice)</pre> <p><u>mais funções na documentação oficial</u></p>
Dicionários	<pre>dicionario = {"chave 1": 42, "chave 2": [1, 2, 3]} dicionario["chave 1"] • dicionario["chave 3"] = "Olá!"</pre> <p><u>mais funções na documentação oficial</u></p>
Textos	<pre>x = "aspas duplas" • y = 'aspas simples' • x + "\n" + y</pre>
Condicionais	<pre>if x == 0: y = 4 else: if x not in [1, 2]: y = 4 else: y = 0</pre> <p style="text-align: right;"> <code>if x != 0 y = 4 elif x >= 0: y = 3 else: y = 0</code> </p>
Repetições	<pre>for i in [1, 2, 3]: for i in range(1, 4): while x > 1: def funcao1(x): def funcao2(x, y, z): def funcao3(): return x + 2 </pre>
Criação de Funções	

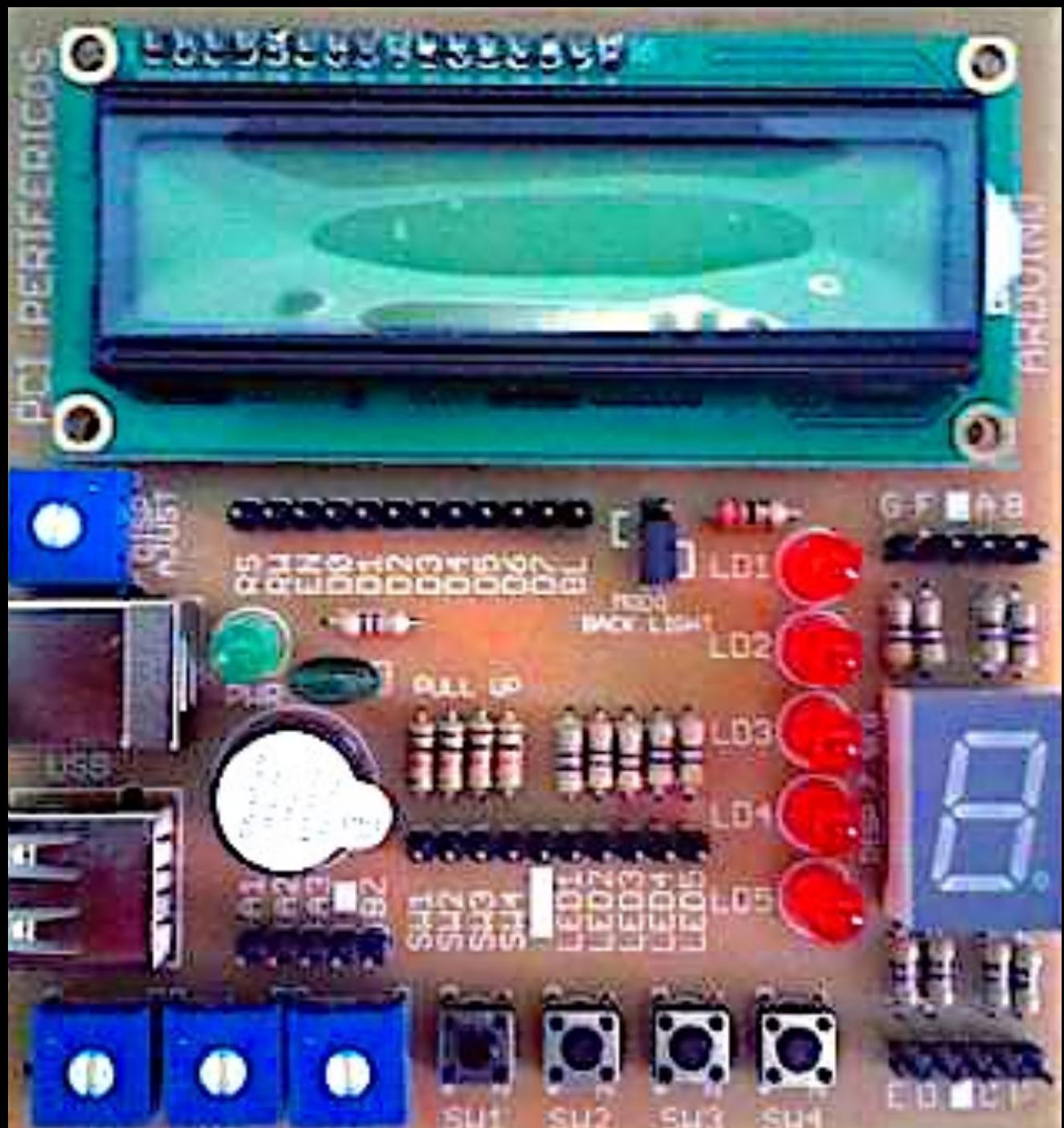
Prática



janks.link/micro/projeto03.zip

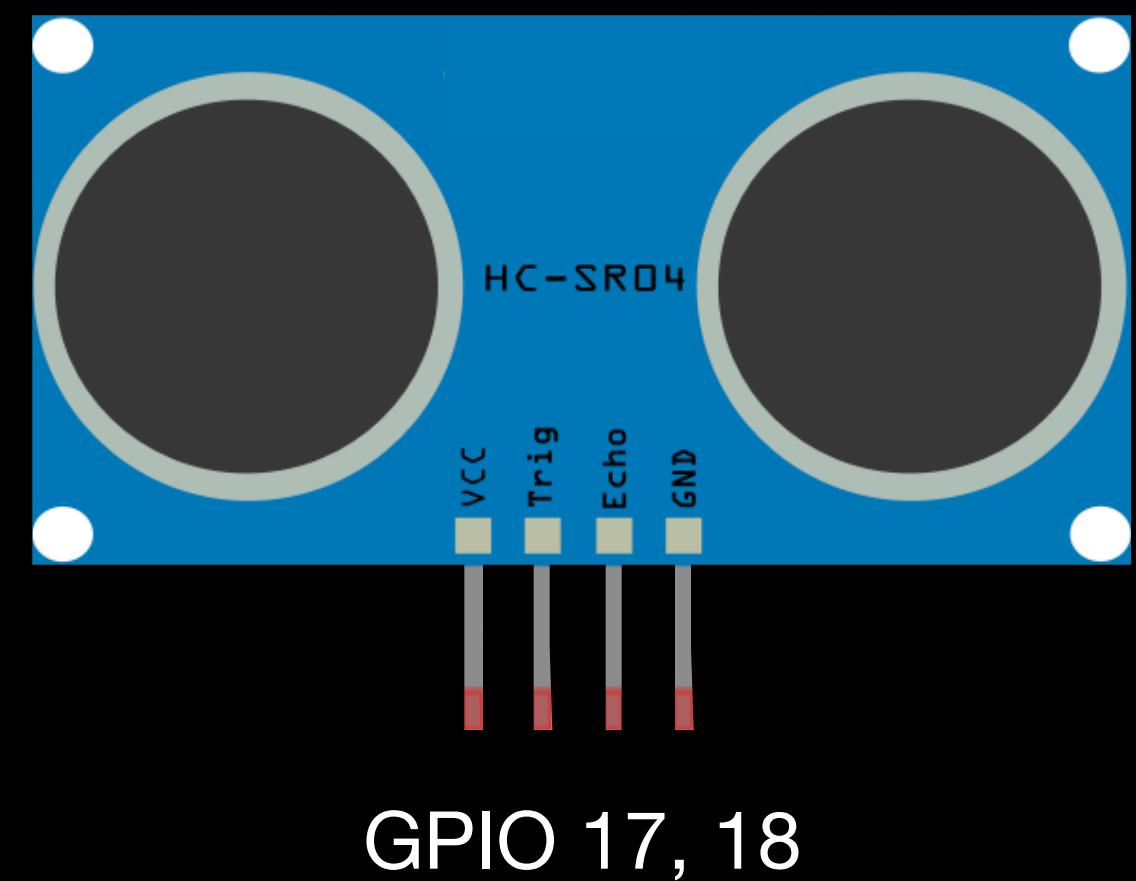
Testes Iniciais

GPIO 2, 3, 4, 5, 6, 7



GPIO 11, 12, 13, 14

GPIO 21
GPIO 22
GPIO 23
GPIO 24
GPIO 25



GPIO 17, 18

Conexões com as Portas da GPIO



Testes Iniciais

Ao pressionar o botão 1, toque a campainha 1 vez durante meio segundo.

Ao pressionar o botão 2, meça a distância (em centímetros) e exiba-a no LCD de caracteres.

Também ao pressionar o botão 2, salve a distância e a data/hora em uma coleção do banco de dados.

Ao pressionar o botão 3, busque a distância menor que 20 cm mais recente e exiba a distância e a hora (ex: 16:04:32) no LCD de caracteres.

↪ DICA: busque em ordem descrescente de hora.

Implementação



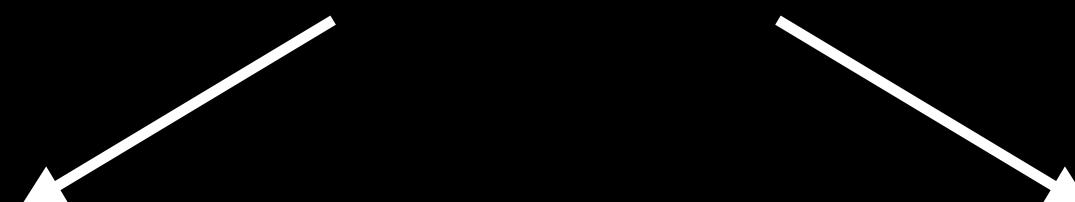
Controle de Acesso de Moradores



Digite o apto:



Digite a senha:



Bem-vindo
Jan K. S.

Acesso negado

Controle de Acesso de Moradores

banco "projeto03"
coleção "apartamentos"

```
[  
  {  
    "numero": "101",  
    "moradores": [  
      {  
        "nome": "Rodrigo",  
        "senha": "101001"  
      },  
      {  
        "nome": "Gabriela",  
        "senha": "101002"  
      }  
    ],  
    {  
      "numero": "102",  
      "moradores": [  
        {  
          "nome": "Paul McCartney",  
          "senha": "102001"  
        },  
        {  
          "nome": "John Lennon",  
          "senha": "102002"  
        },  
        {  
          "nome": "George Harrison",  
          "senha": "102003"  
        },  
        {  
          "nome": "Ringo Star",  
          "senha": "102004"  
        }  
      ]  
    },  
    ...  
  ]
```



		KEY_UP
KEY_LEFT	KEY_OK	KEY_RIGHT
	KEY_DOWN	
	KEY_1	KEY_2
	KEY_4	KEY_5
KEY_7	KEY_8	KEY_9
	KEY_0	

Códigos dos Botões do Mini Controle Remoto



Implementação

Crie uma função que receba uma mensagem, peça os dígitos do controle remoto (seguindo o layout sugerido) e retorne o texto com os dígitos.

↪ DICA: use um while True e o acesso a um caracter da string com texto [posicao].

Crie uma função que receba o nome do morador e a senha. Caso seja encontrado no banco, exiba o nome do morador no LCD; caso contrário, exibe uma mensagem de erro.

↪ DICA: busque o apartamento e percorra seus moradores comparando a senha.

Use as duas funções acima para solicitar o número do apartamento, solicitar a senha e validar o acesso.

↪ DICA: use um outro while True.

Aperfeiçoamento



Aperfeiçoamento

Ao pressionar as teclas do controle remoto, emita um "beep" rápido. Ao final de um acesso inválido, emita um outro efeito sonoro.

Peça o número do apartamento apenas se houver alguém a pelo menos 20 cm do sensor.

A cada acesso válido, salve em uma outra coleção o número do apartamento, o nome do morador e a data/hora de seu acesso.

Ao apertar o botão 1, peça o número do apartamento. Em seguida, imprima **no terminal** o nome e a data/hora de cada acesso a ele, em ordem crescente de data/hora.
↪ SUGESTÃO: imprima de um jeito bonitinho.