

# Projeto 05

## Controle Automático

Jan K. S. – janks@puc-rio.br

ENG1419 – Programação de Microcontroladores

# Revisão do Projeto 04

```
aula@raspberrypi ~ $ fswebcam foto1.jpeg
--- Opening /dev/video0...
Trying source module v4l2...
/dev/video0 opened.
No input was specified, using the first.
Adjusting resolution from 384x288 to 352x288.
--- Capturing frame...
Captured frame in 0.00 seconds.
--- Processing captured image...
Writing JPEG image to 'foto1.jpeg'.
```

```
aula@raspberrypi ~ $ fswebcam --resolution 640x480 foto2.jpeg
--- Opening /dev/video0...
Trying source module v4l2...
/dev/video0 opened.
No input was specified, using the first.
--- Capturing frame...
Captured frame in 0.00 seconds.
--- Processing captured image...
Writing JPEG image to 'foto2.jpeg'.
```

As imagens são geradas na mesma pasta  
do arquivo com o código Python.



```
>>> from os import system  
>>> comando = "fswebcam foto1.jpeg"  
>>> system(comando)  
>>> system("fswebcam --resolution 640x480 foto2.jpeg")
```

Chamada de Comandos para Tirar uma Foto



audio1.wav  
(baixa qualidade)



audio2.wav  
(qualidade de CD)

```
>>> from os import system  
  
>>> system("arecord --duration 3 audio1.wav")  
  
>>> system("arecord --duration 3 --format cd audio2.wav")
```

Gravação de 3 Segundos de Áudio

```
>>> from subprocess import Popen  
  
>>> aplicativo = None  
  
>>> def iniciar_gravacao():  
...     global aplicativo  
...     comando = ["arecord", "--duration", "30", "audio.wav"]  
...     aplicativo = Popen(comando) ← executa arecord em plano de fundo  
...  
>>> def parar_gravacao():  
...     if aplicativo != None:  
...         aplicativo.terminate()  
...  
>>> iniciar_gravacao()  
>>> sleep(4)  
>>> parar_gravacao()
```

```
>>> from os import system  
>>> system("lame audio.wav audio.mp3")  
>>> system("opusenc audio.wav audio.ogg")
```

Conversão de WAV para MP3 e para OGG em Python

`api.telegram.org/bot<SUA_CHAVE_SECRETA>/<comando>`

+

dados extras

Comando	Dados Obrigatórios
sendMessage	id do chat, texto
sendPhoto	id do chat, arquivo
sendVoice	id do chat, arquivo
getUpdates	(nenhum)
getFile	id do arquivo

Exemplos de Comandos para Bots do Telegram

```
>>> chave = "COLOQUE A SUA CHAVE AQUI!"  
>>> id_da_conversa = "COLOQUE O ID DA SUA CONVERSA AQUI!"  
>>> endereco_base = "https://api.telegram.org/bot" + chave
```

Configuração Inicial para Usar o Bot

```
>>> from requests import post  
>>> endereco = endereco_base + "/sendMessage"  
>>> dados = {"chat_id": id_da_conversa, "text": "Oi!"}  
>>> resposta = post(endereco, json=dados)
```

```
>>> from requests import post  
>>> endereco = endereco_base + "/sendPhoto"  
>>> dados = {"chat_id": id_da_conversa}  
>>> arquivo = {"photo": open("foto.jpeg", "rb")}  
>>> resposta = post(endereco, data=dados, files=arquivo)
```

```
>>> from requests import post  
>>> endereco = endereco_base + "/sendVoice"  
>>> dados = {"chat_id": id_da_conversa}  
>>> arquivo = {"voice": open("audio.ogg", "rb")}  
>>> resposta = post(endereco, data=dados, files=arquivo)
```

```
{  
    "ok":true,  
    "result": [  
        {  
            "update_id":564714813,  
            "message": {  
                "voice": {  
                    "file_id": "AwADAQADJAADQzTwRQGVIkswNsWHAg",  
                    "...  
                },  
                "...  
            }  
        },  
        {  
            "update_id":564714814,  
            "message": {  
                "photo": [  
                    {  
                        "file_id": "AgADAQAD1qcxG0M08EXPZxFkkC7VB-drDDAABHNCNifaQdcHHhEBAEAC",  
                        "file_size": 1438,  
                        "...  
                    },  
                    {  
                        "file_id": "AgADAQAD1qcxG0M08EXPZxFkkC7VB-drDDAABAuIfKssEffgIBEBAEAC",  
                        "file_size": 117149,  
                        "...  
                    },  
                    "...  
                ]  
            }  
        },  
        {  
            "update_id":564714815,  
            "message": {  
                "text": "Texto!",  
                "...  
            }  
        }  
    ]  
}
```

Dicionário (JSON) com as Atualizações de Mensagens

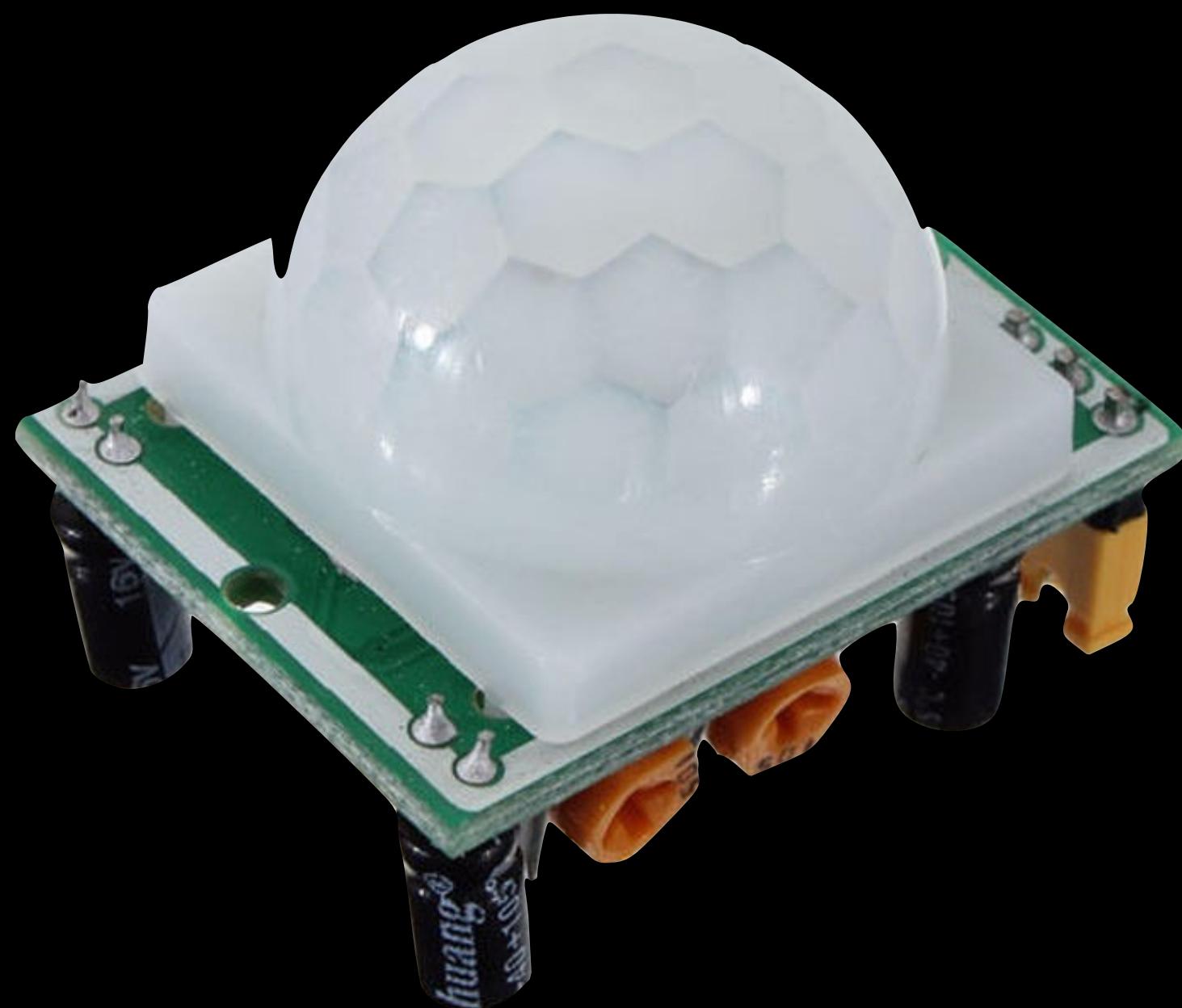
```
>>> from requests import get
>>> endereco = endereco_base + "/getUpdates"
>>> resposta = get(endereco)
>>> dicionario_da_resposta = resposta.json()
>>> for resultado in dicionario_da_resposta["result"]:
...     mensagem = resultado["message"]
...     if "text" in mensagem:
...         texto = mensagem["text"]
...     elif "voice" in mensagem:
...         id_do_arquivo = mensagem["voice"]["file_id"]
...     elif "photo" in mensagem:
...         foto_mais_resolucao = mensagem["photo"][-1]
...         id_do_arquivo = foto_mais_resolucao["file_id"]
```

```
>>> endereco = endereco_base + "/getFile"
>>> dados = {"file_id": id_do_arquivo}
>>> resultado = get(endereco, json=dados)
>>> dicionario = resultado.json()
>>> caminho_do_arquivo = dicionario["result"]["file_path"]

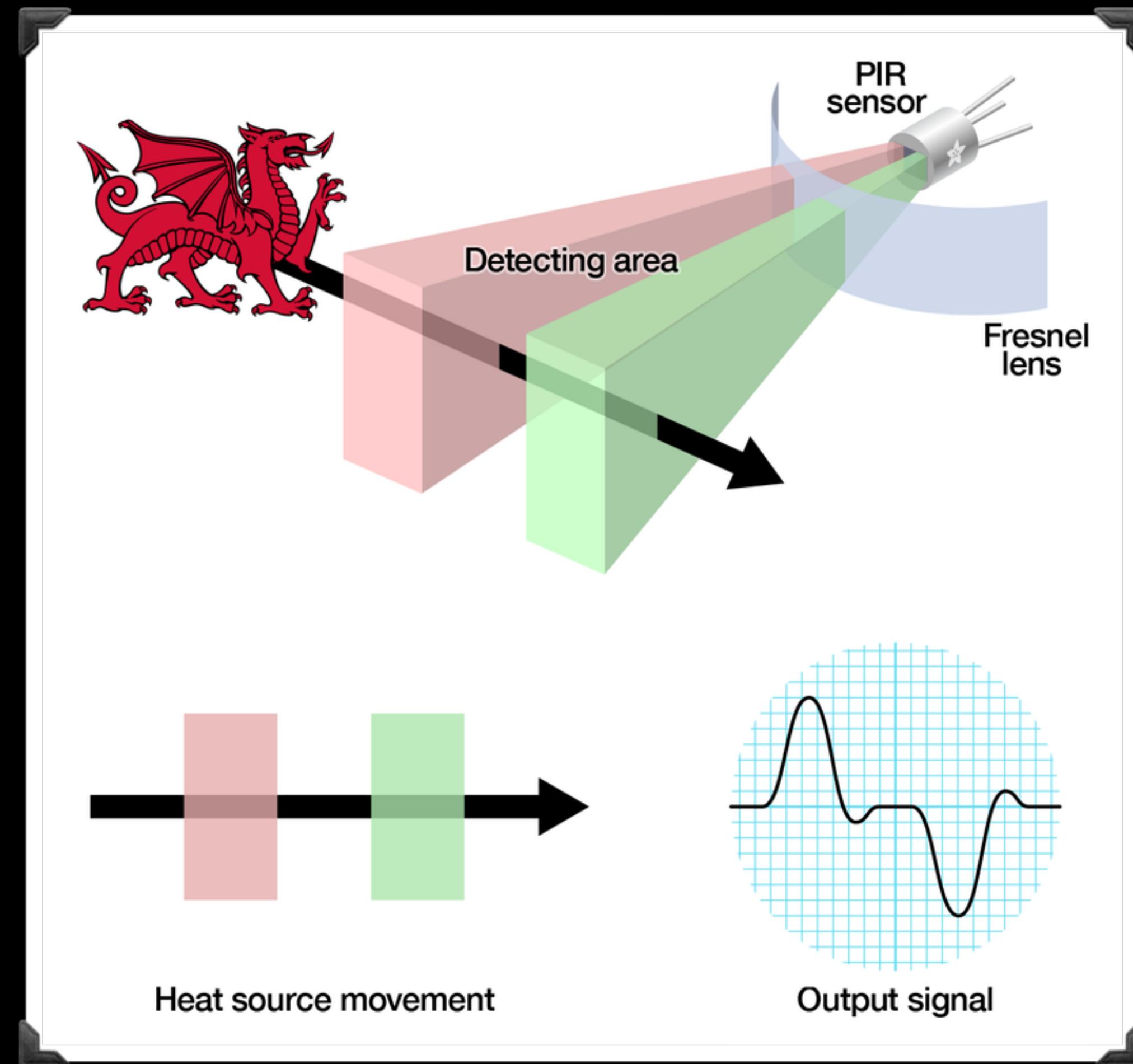
>>> from urllib.request import urlretrieve
>>> endereco_do_arquivo = "https://api.telegram.org/file/bot"
+ chave + "/" + caminho_do_arquivo
>>> arquivo_de_destino = "meu_arquivo.ogg"
>>> urlretrieve(endereco_do_arquivo, arquivo_de_destino)
```

```
>>> proximo_id_de_update = 0
>>> while True:
...
    endereco = endereco_base + "/getUpdates"
...
    dados = {"offset": proximo_id_de_update}
...
    resposta = get(endereco, json=dados)
...
    dicionario_da_resposta = resposta.json()
...
    for resultado in dicionario_da_resposta["result"]:
...
        ...
...
    proximo_id_de_update = int(resultado["update_id"]) + 1
```

# Hardware

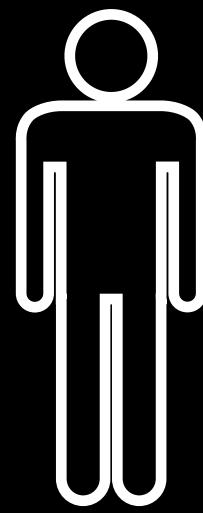
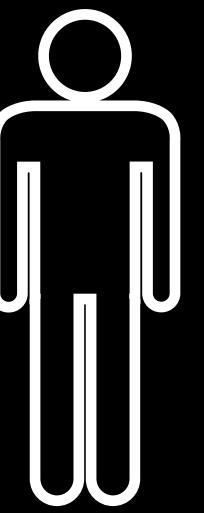


Sensor de Movimento



Funcionamento de um Infra-Vermelho Passivo (PIR)

```
>>> from gpiozero import MotionSensor  
>>> sensor_de_movimento = MotionSensor(27)  
>>> sensor_de_movimento.wait_for_motion()  
>>> sensor_de_movimento.wait_for_no_motion()  
>>> def movimento_detectado():  
...     print("Haja luz!")  
...  
>>> sensor_de_movimento.when_motion = movimento_detectado  
>>> def inercia_detectada():  
...     print("Stop! Hammer time!")  
...  
>>> sensor_de_movimento.when_no_motion = inercia_detectada
```



movimento!



ainda em  
movimento!

$\approx 2\text{s}$



inércia!

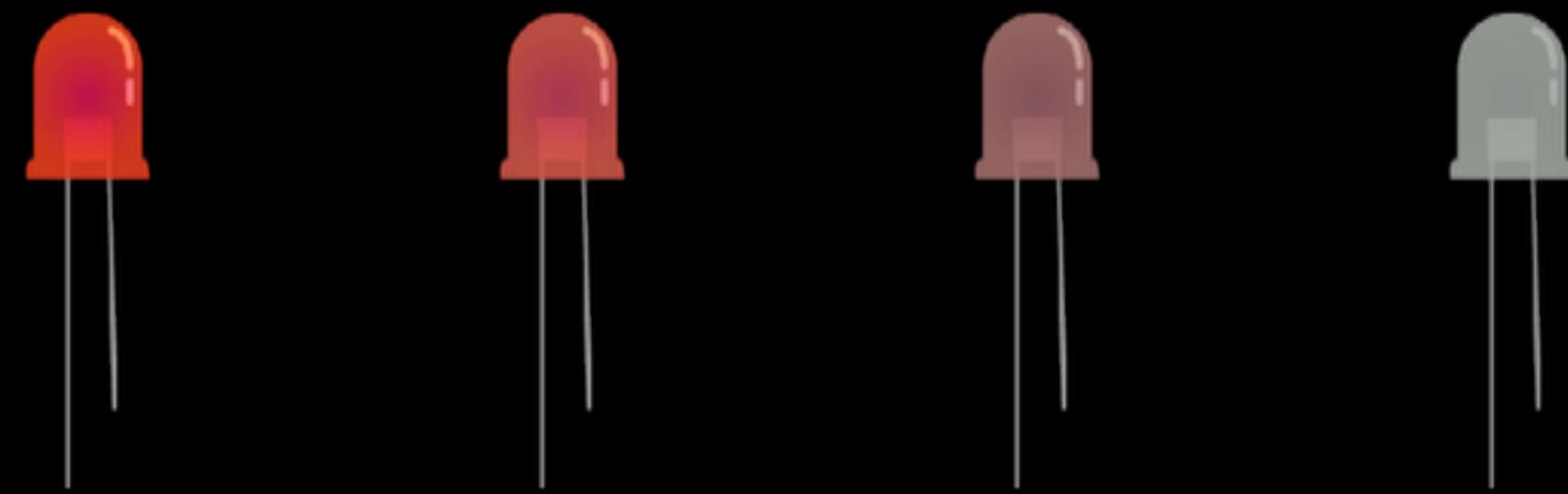
Sensor de Movimento

```
>>> from threading import Timer  
>>> def timer_acabou():  
...     print("Acabouuuoo, acabou!")  
  
...  
>>> timer = Timer(3.5, timer_acabou)  
>>> timer.start()
```

*3 segundos e meio depois...*

Acabouuuoo, acabou!

```
>>> from threading import Timer  
>>> timer = None  
>>> def timer_recorrente():  
...     print("Hey, listen!")  
...     global timer  
...     timer = Timer(1.0, timer_recorrente)  
...     timer.start()  
...  
>>> timer_recorrente()  
>>> def parar_timer():  
...     timer.cancel()  
...
```



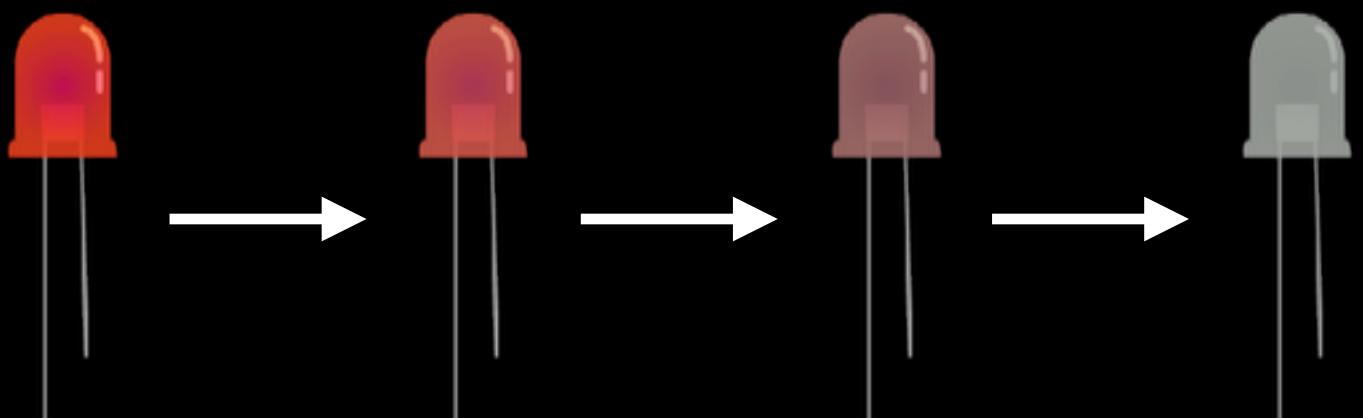
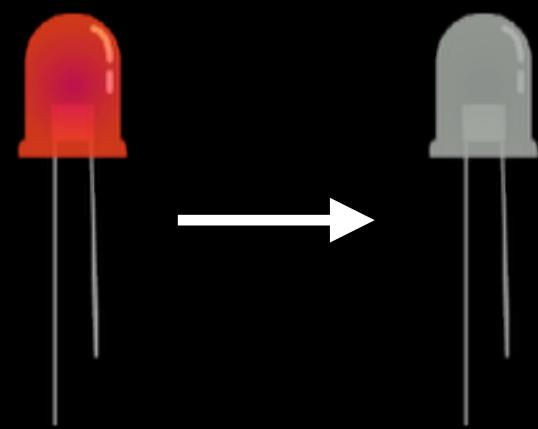
Níveis de Brilho de um LED

```
>>> from gpiozero import PWMLED  
>>> ledPWM = PWMLED(21)  
  
>>> ledPWM.on()  
  
>>> ledPWM.value  
1.0  
  
>>> ledPWM.off()  
  
>>> ledPWM.value  
0.0  
  
>>> ledPWM.value = 0.5
```

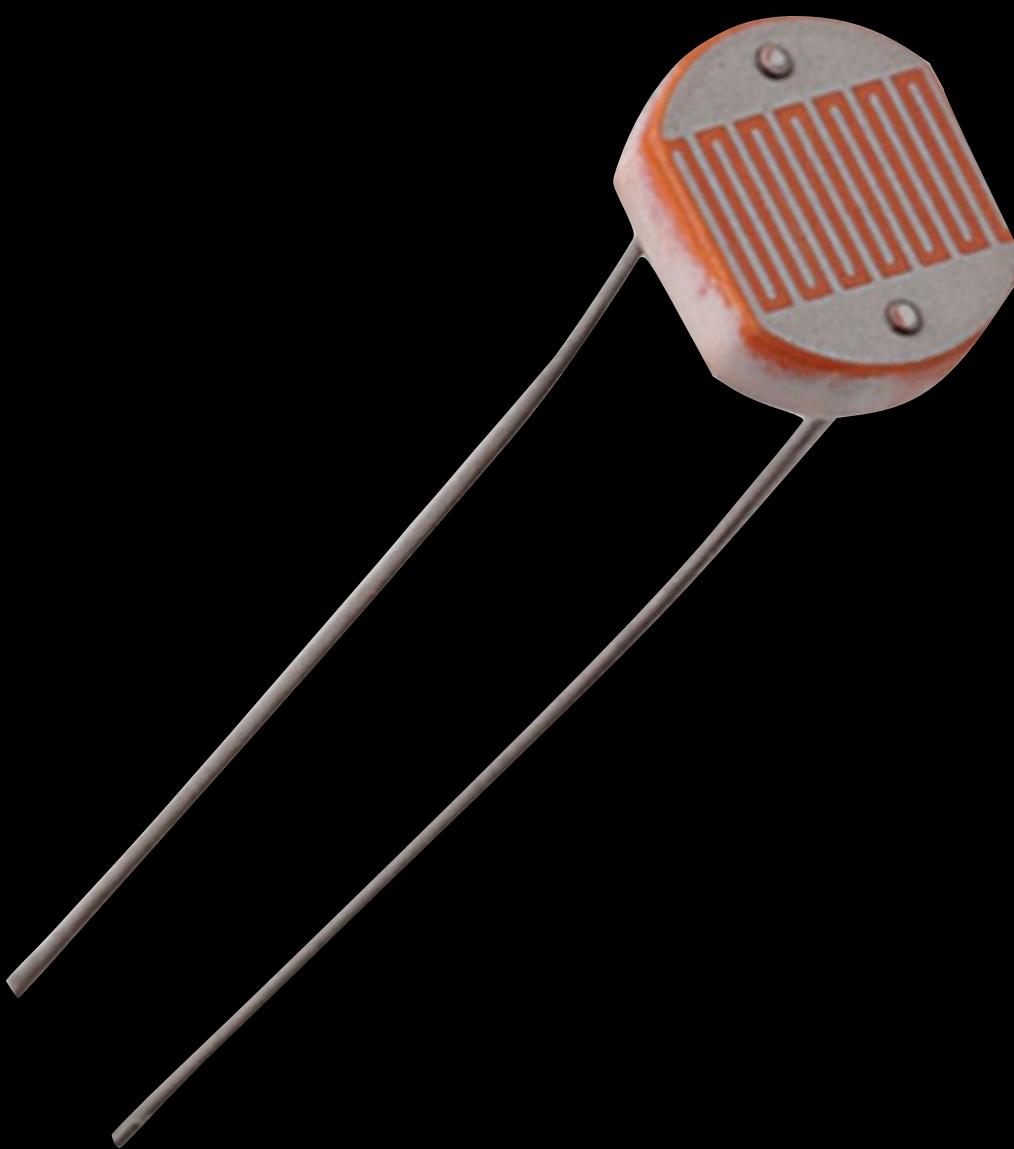


Valor de um LED PWM

```
>>> from gpiozero import PWMLED  
>>> ledPWM = PWMLED(21)  
>>> ledPWM.blink()  
>>> ledPWM.blink(n=3, on_time=1, off_time=0.5)  
>>> ledPWM.pulse()  
>>> ledPWM.pulse(n=3, fade_in_time=2, fade_out_time=1)
```

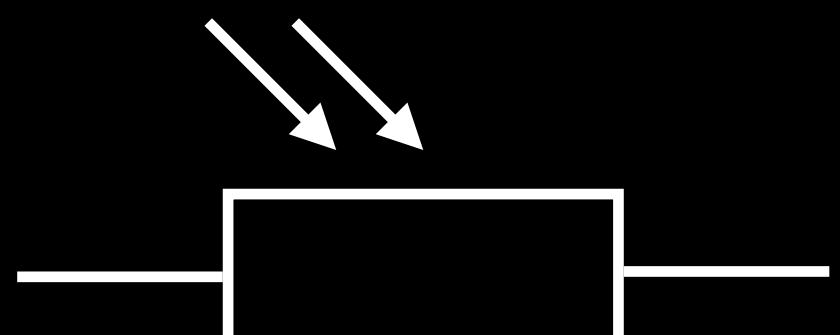
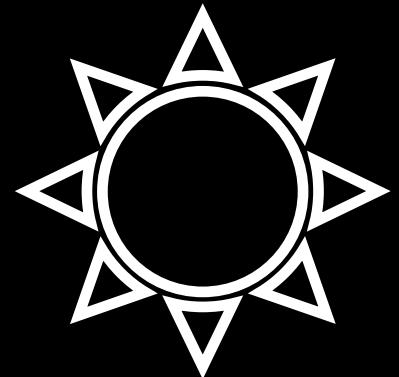
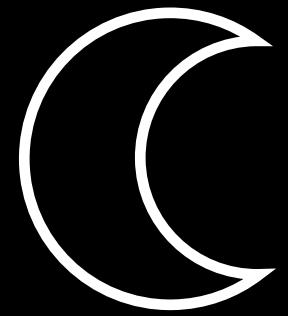
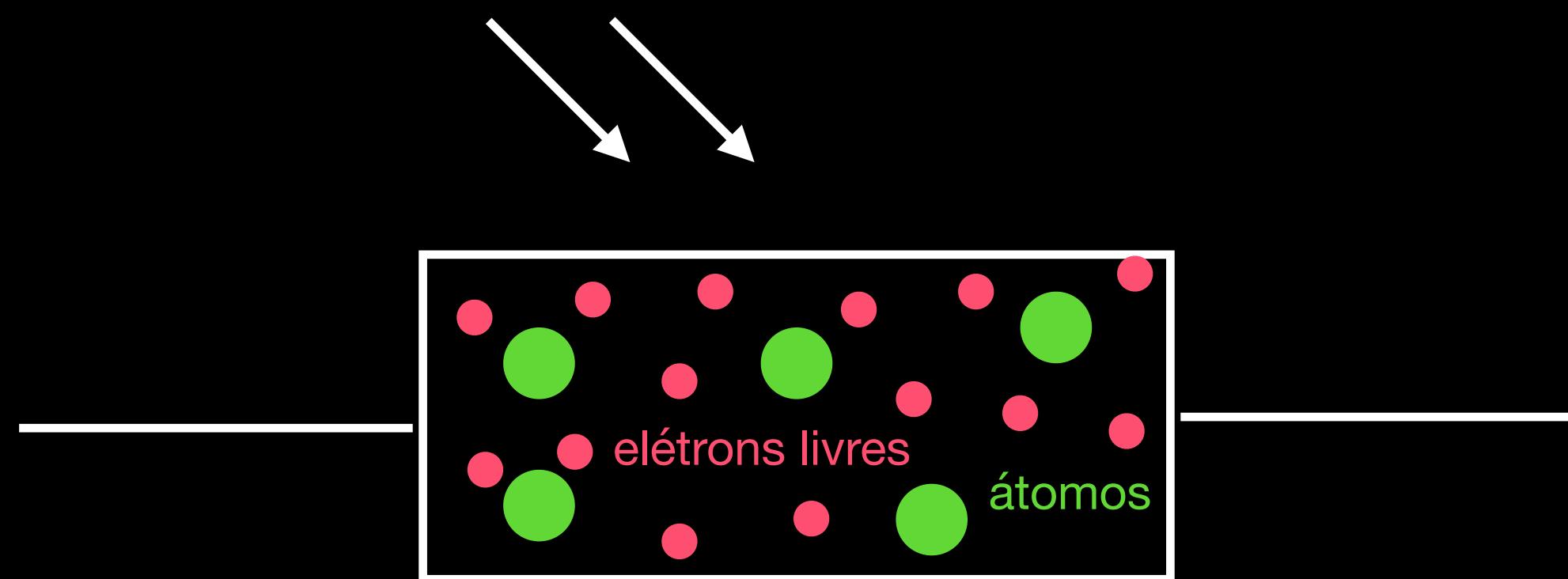


Piscamento vs Pulsação

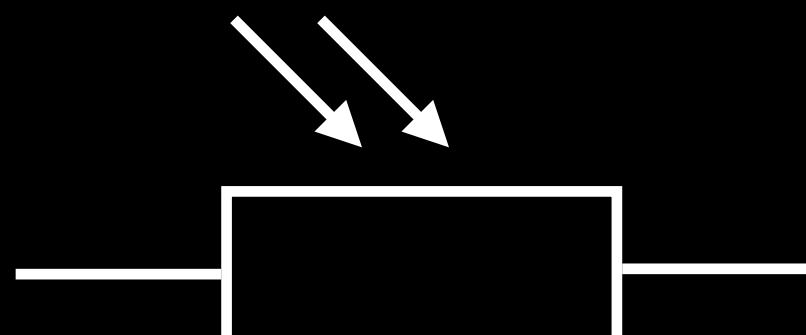


Sensor de Luz

fótons da luz



$50 \text{ k}\Omega$

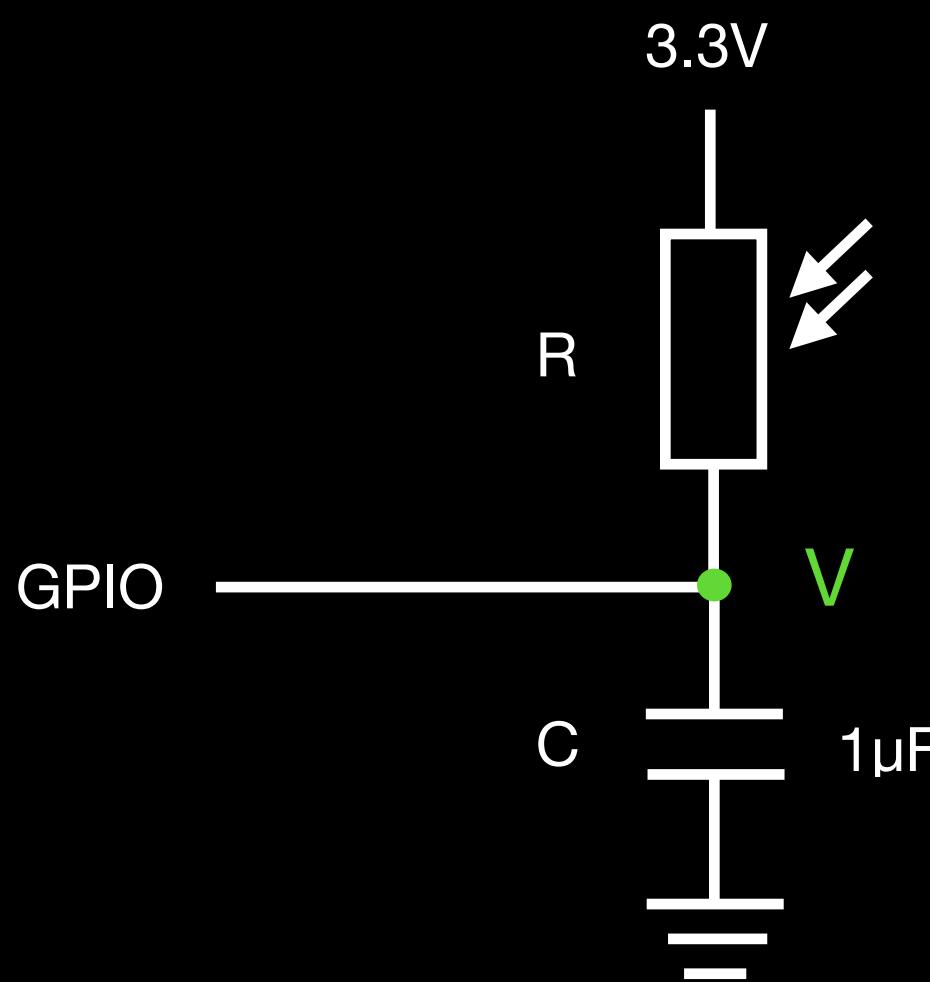


$3 \text{ k}\Omega$

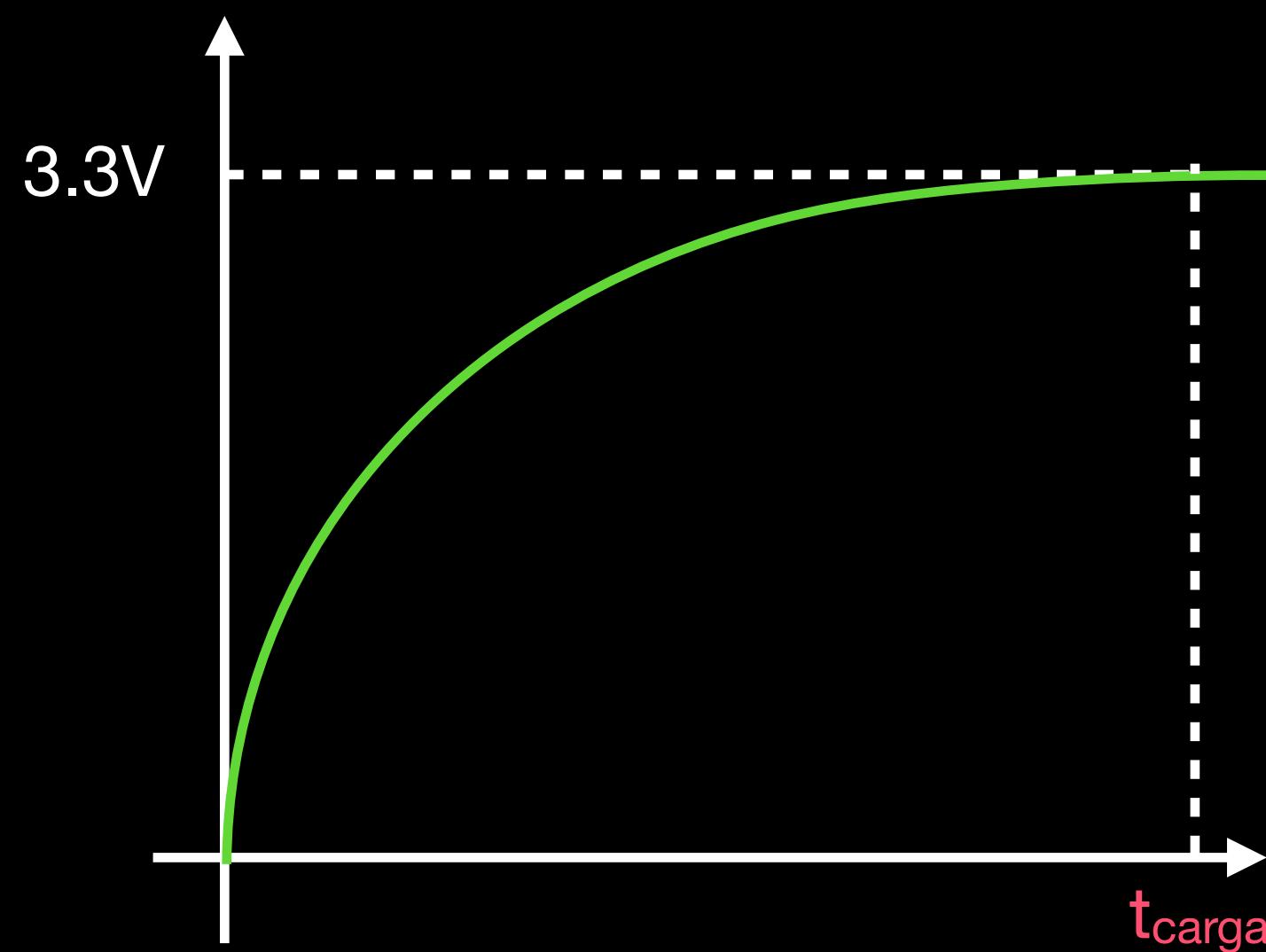
Resistência Dependente de Luz (LDR)



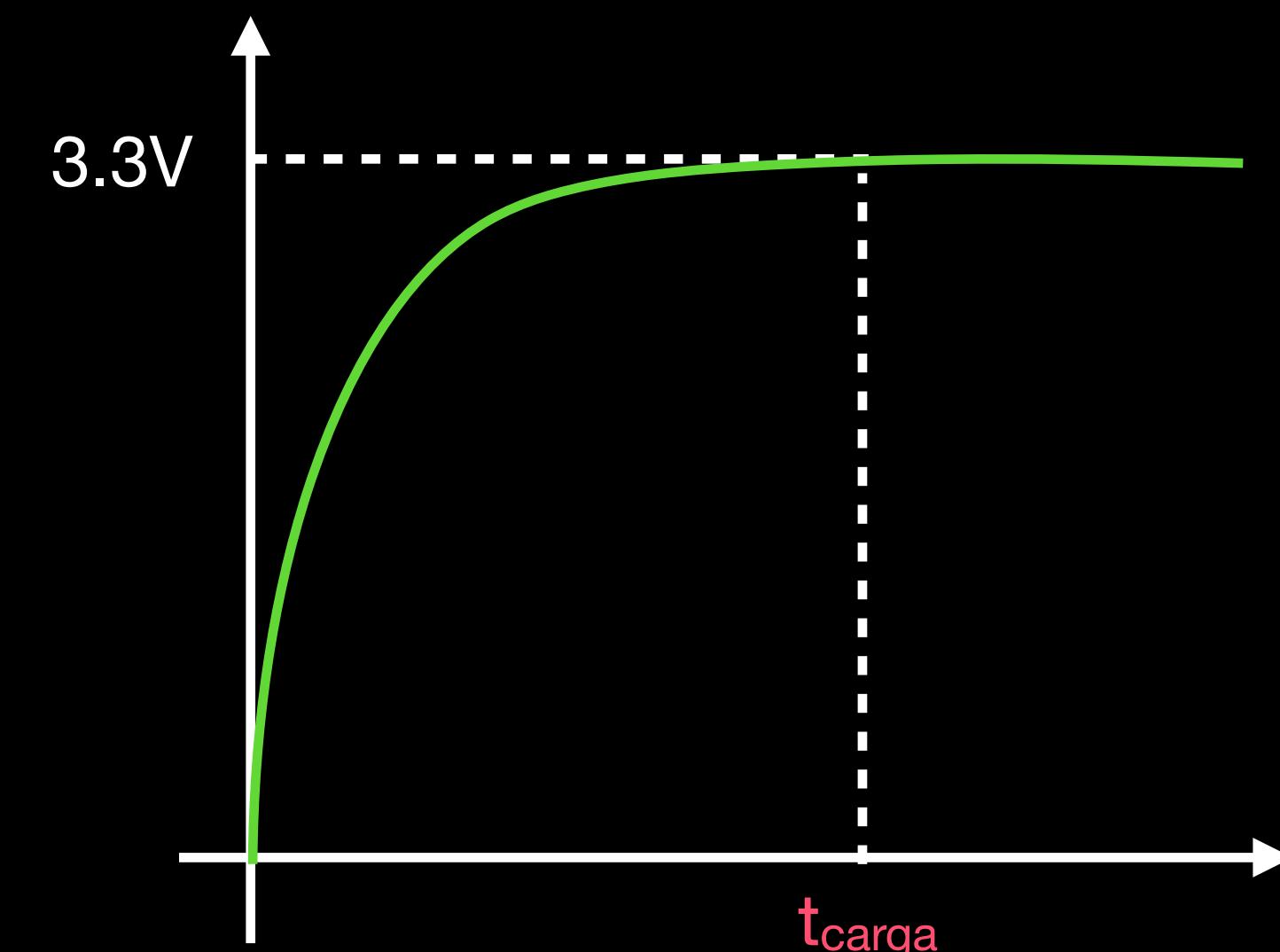
Círculo com Capacitor para Detecção de Luz



Carregamento sem Luz (R Maior)

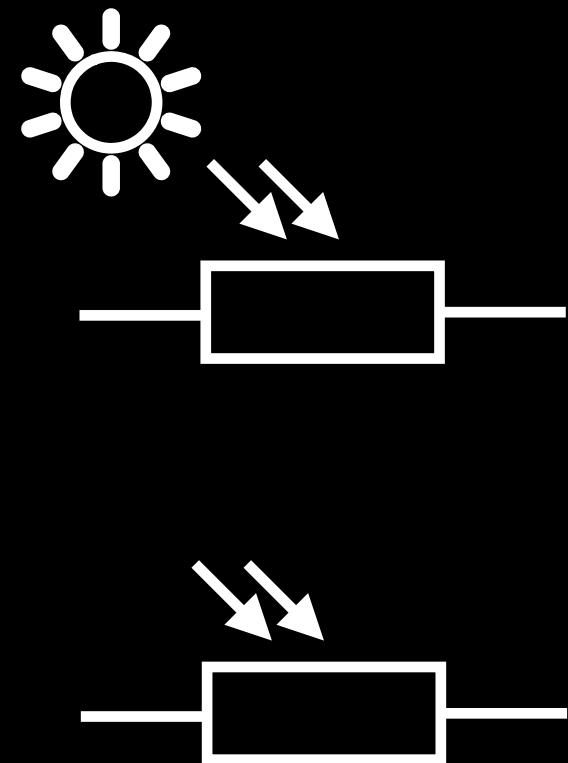


Carregamento com Luz (R Menor)



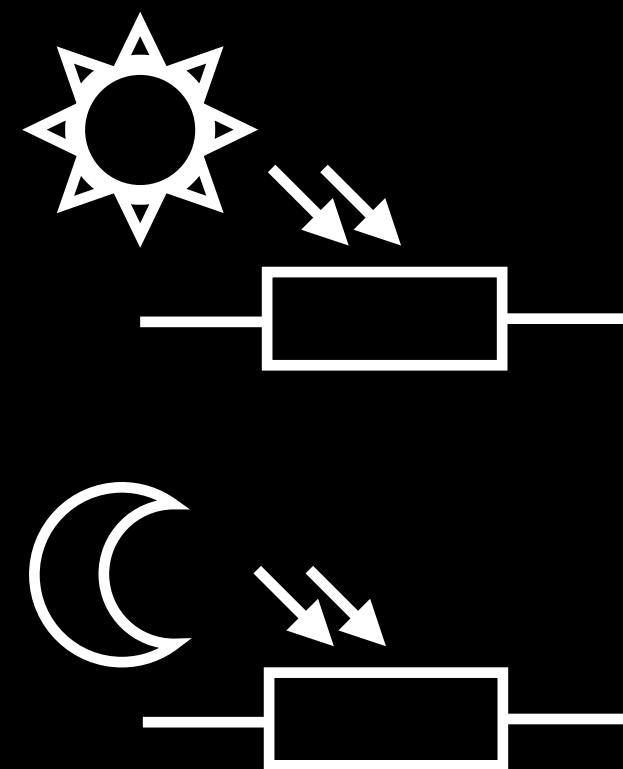
Tempo de Carregamento

```
>>> from gpiozero import LightSensor  
>>> sensor_de_luz = LightSensor(8)  
>>> sensor_de_luz.value  
0.854363  
  
>>> sensor_de_luz.value  
0.347932
```



Valor de um Sensor de Luz

```
>>> from gpiozero import LightSensor  
>>> from time import sleep  
>>> sensor_de_luz = LightSensor(8)  
>>> for i in range(0, 3):  
...     print(sensor_de_luz.value)  
...     sleep(1)  
...  
0.854363  
0.854363 ← alteração imediata do valor  
0.347932  
0.347932
```



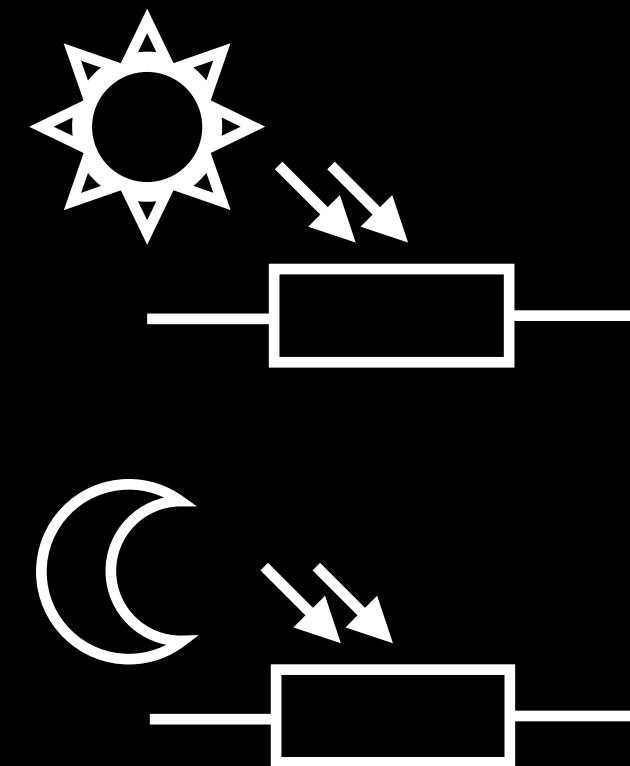
Captação Gradual de um Sensor de Luz

```
>>> from gpiozero import LightSensor  
>>> from time import sleep  
>>> sensor_de_luz = LightSensor(8, queue_len=20)  
>>> for i in range(0, 3):  
...     print(sensor_de_luz.value)  
...     sleep(1)
```

```
...
```

```
0.854363  
0.653456  
0.498232  
0.345463
```

← alteração gradual do valor

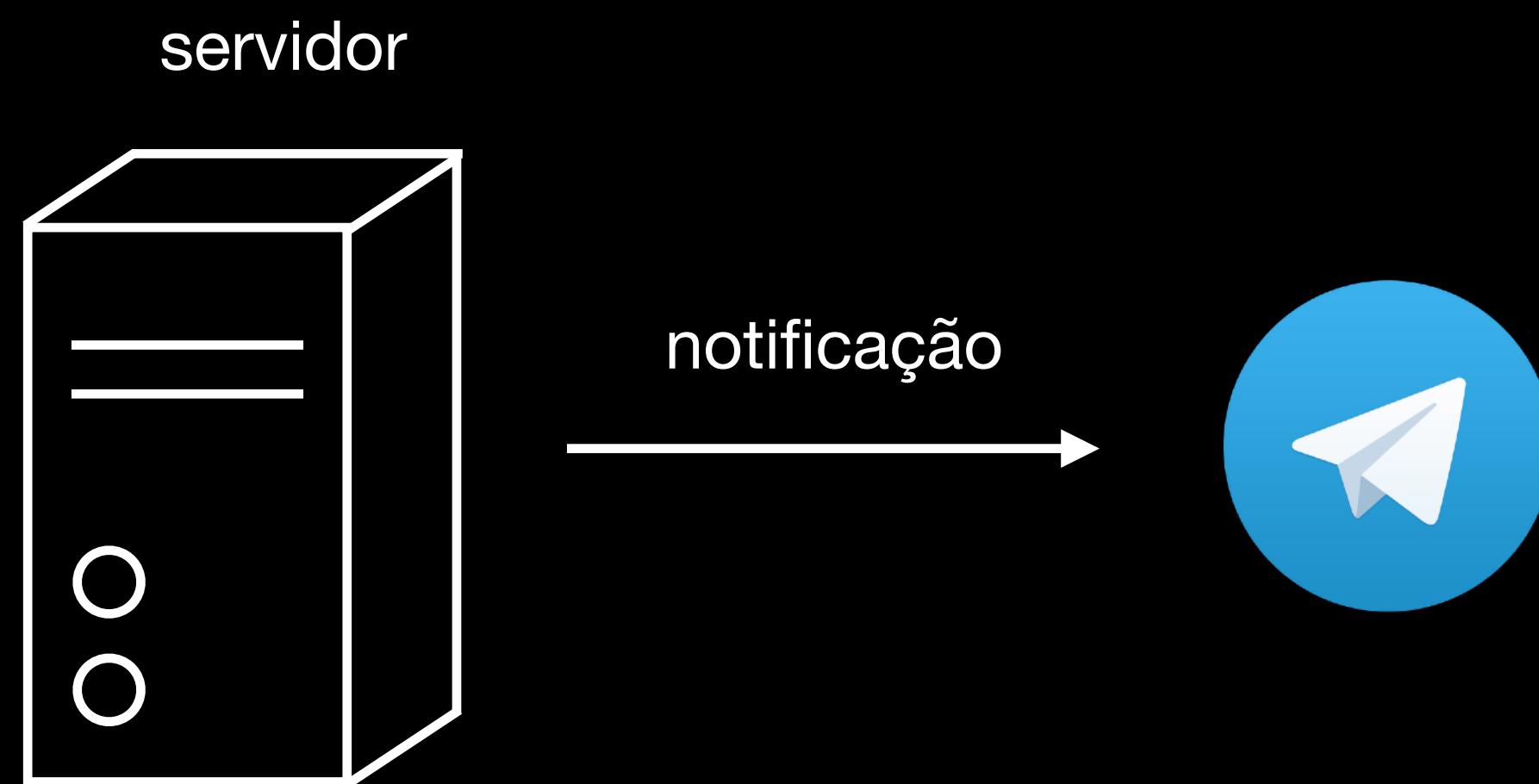
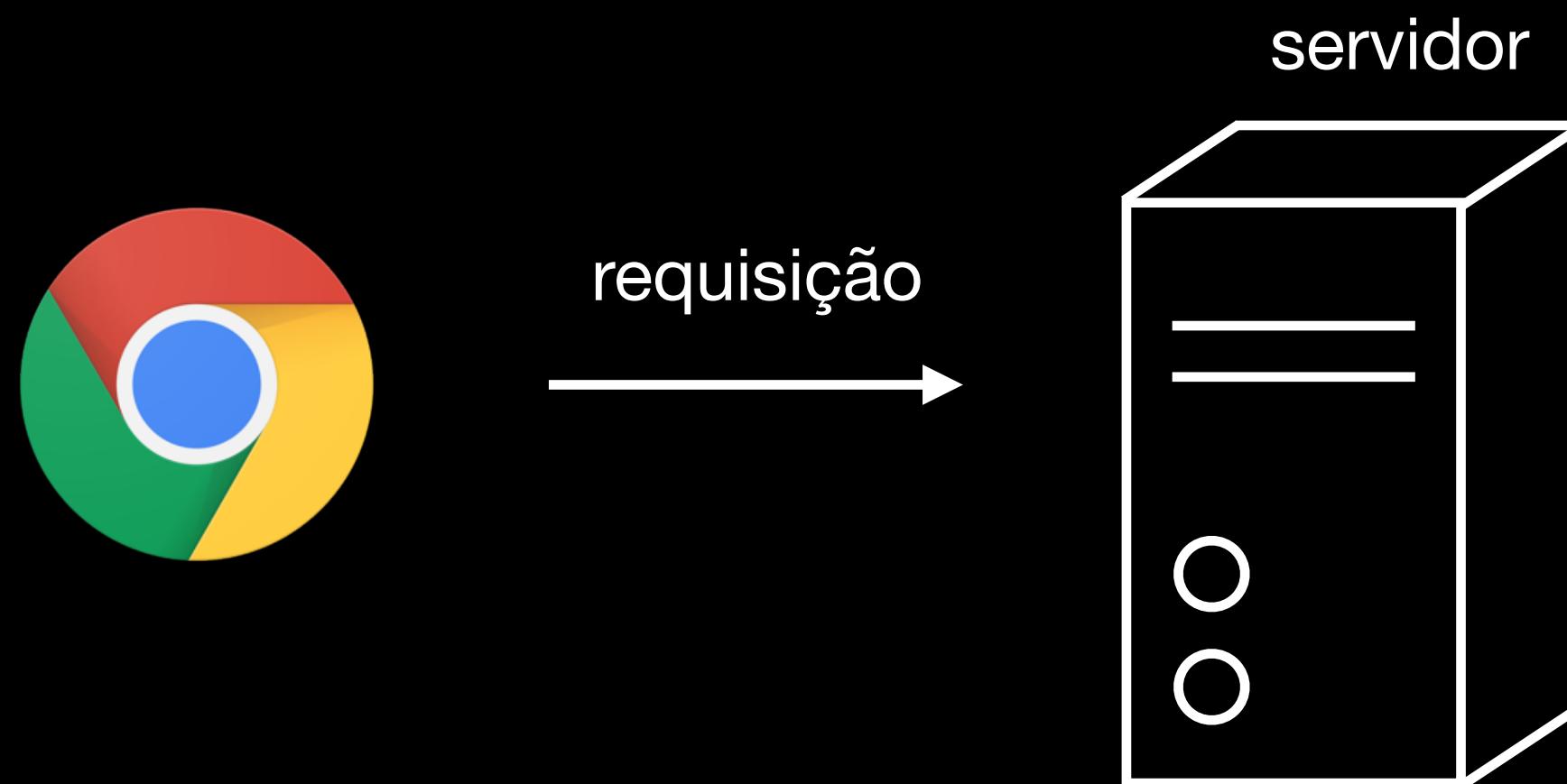


Captação Gradual de um Sensor de Luz

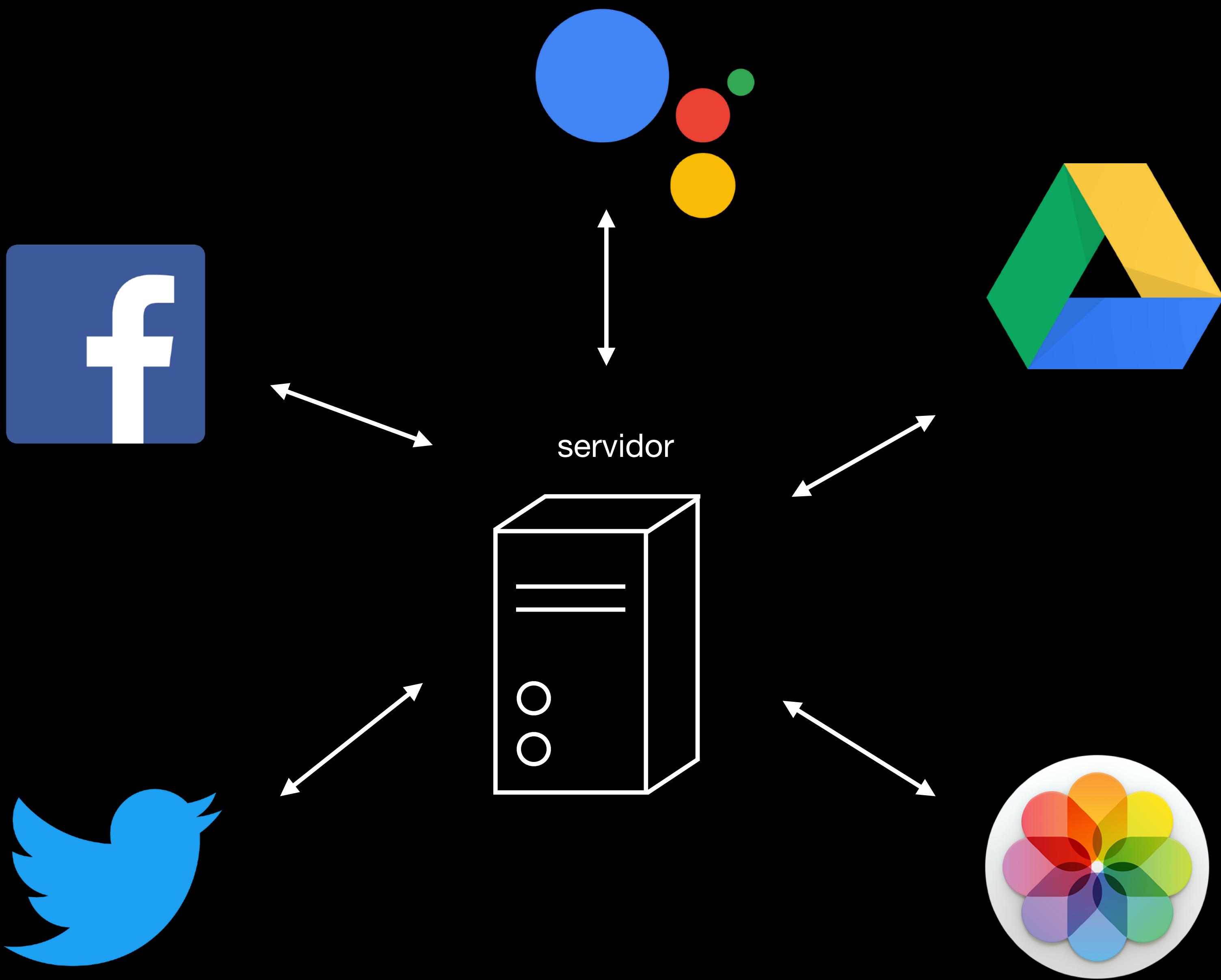
```
>>> from gpiozero import LightSensor  
>>> sensor_de_luz = LightSensor(8)  
>>> sensor_de_luz.value  
0.56743  
  
>>> sensor_de_luz.threshold  
0.5  
  
>>> sensor_de_luz.light_detected  
True  
  
>>> sensor_de_luz.threshold = 0.6  
>>> sensor_de_luz.light_detected  
False
```

```
>>> from gpiozero import LightSensor  
>>> sensor_de_luz = LightSensor(8)  
>>> sensor_de_luz.wait_for_dark()  
>>> sensor_de_luz.wait_for_light()  
>>>  
>>> def luz_detectada():  
...     print("Haja luz!")  
>>> sensor_de_luz.when_light = luz_detectada  
>>>  
>>> def escuridao_detectada():  
...     print("Hello darkness my old friend...")  
>>> sensor_de_luz.when_dark = escuridao_detectada
```

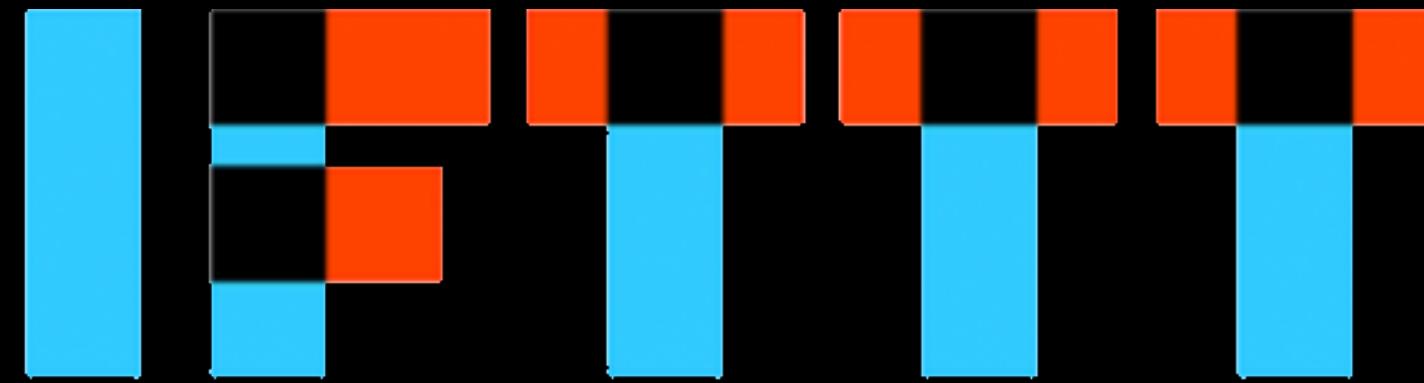
# Software



Requisições e Notificações (Vistas em Projetos Anteriores)



Dificuldade em Programar Várias Comunicações



Recipe

**if this then that**

Trigger

Action

IFTTT: If This Than That

The image shows the homepage of IFTTT (If This Then That) at ifttt.com. The main headline is "A world that works for you". Below it, a paragraph explains that IFTTT is the free way to get all your apps and devices talking to each other. A call-to-action button says "Get started". On the left, there are sign-in options for email, Google, and Facebook. To the right, a central graphic illustrates how various devices and services can be interconnected. It features a lightbulb, a smartphone with a dog icon, a calendar showing "DEC 14", an email invitation for a "Saturday Party!", a smart speaker, a smartwatch with a heart rate graph, a cloud icon, and a speaker. Arrows show the flow of data between these components, such as from the calendar to the email invitation or from the smartwatch to the cloud.

Integração entre Serviços com IFTTT

A screenshot of a web browser window showing the ifttt.com/create page. The browser interface includes standard controls like back, forward, and search at the top. The main content area has a blue header with the text "New Applet". Below this, the central text reads "if +this then that" where the "plus" sign is replaced by a blue square containing a white plus sign. At the bottom of the page, there is a green button icon with a smiley face and the text "Want to build your own service? Build on the platform ↗". The footer contains links for "About", "Blog", "Help", "Jobs", "Terms", and "Privacy". It also features two download buttons: "Download on the App Store" (with the Apple logo) and "GET IT ON Google play" (with the Google Play logo).

New Applet

if +this then that

Want to build your own service? Build on the platform ↗

About    Blog    Help    Jobs    Terms    Privacy

Download on the App Store    GET IT ON Google play

Escolha do Serviço "This"

[i](#) [o](#) [ifttt.com/create/if?sid=2](#) [↻](#) [+/-](#)

[Back](#)

# Choose a service

Step 1 of 6

Search services

 Twitter	 Date & Time	 RSS Feed	 SMS	 Email	 Weather Underground
 Phone Call (US only)	 Delicious	 Facebook	 Classifieds	 Tumblr	 Flickr

Lista de Serviços Compatíveis

A screenshot of a web browser window showing the IFTTT trigger configuration interface. The URL in the address bar is `ifttt.com/create/if-new-tweet-by-a-specific-user?sid=6`. The page title is "Complete trigger fields". A back button is visible on the left. The main content area features a Twitter logo icon and the title "Complete trigger fields". Below the title, it says "Step 2 of 6". A large blue callout box contains the trigger details: "New tweet by a specific user" (description), "This Trigger fires every time the Twitter user you specify tweets." (trigger condition), "Username to watch" (input field containing "dog\_rates"), and a "Create trigger" button.

IFTTT

ifttt.com/create/if-new-tweet-by-a-specific-user?sid=6

Back

Complete trigger fields

Step 2 of 6

New tweet by a specific user

This Trigger fires every time the Twitter user you specify tweets.

Username to watch

dog\_rates

Create trigger

Configuração do Serviço "This"

  ifttt.com/create/if-new-tweet-by-a-specific-user-then? 

 [Back](#)

# if then that

[About](#) [Blog](#) [Help](#) [Jobs](#) [Terms](#) [Privacy](#)

 Download on the  
[App Store](#)  GET IT ON  
[Google play](#)

Add your service and become a partner

 [IFTTT Platform](#)

Escolha do Serviço "That"

A screenshot of a web browser window showing the IFTTT platform. The URL in the address bar is [ifttt.com/create/if-new-tweet-by-a-specific-user-then-send](https://ifttt.com/create/if-new-tweet-by-a-specific-user-then-send). The window title is "Complete action fields" and it is Step 5 of 6. The main content area has a green background and displays the "Send me an SMS" action. It includes a description: "This Action will send an SMS to your mobile phone.", a "Message" input field containing the template "@[ UserName ] : [ Text ] (via Twitter LinkToTweet)", an "Add ingredient" button, and a large "Create action" button.

IFTTT

ifttt.com/create/if-new-tweet-by-a-specific-user-then-send

Back

Complete action fields

Step 5 of 6

**Send me an SMS**

This Action will send an SMS to your mobile phone.

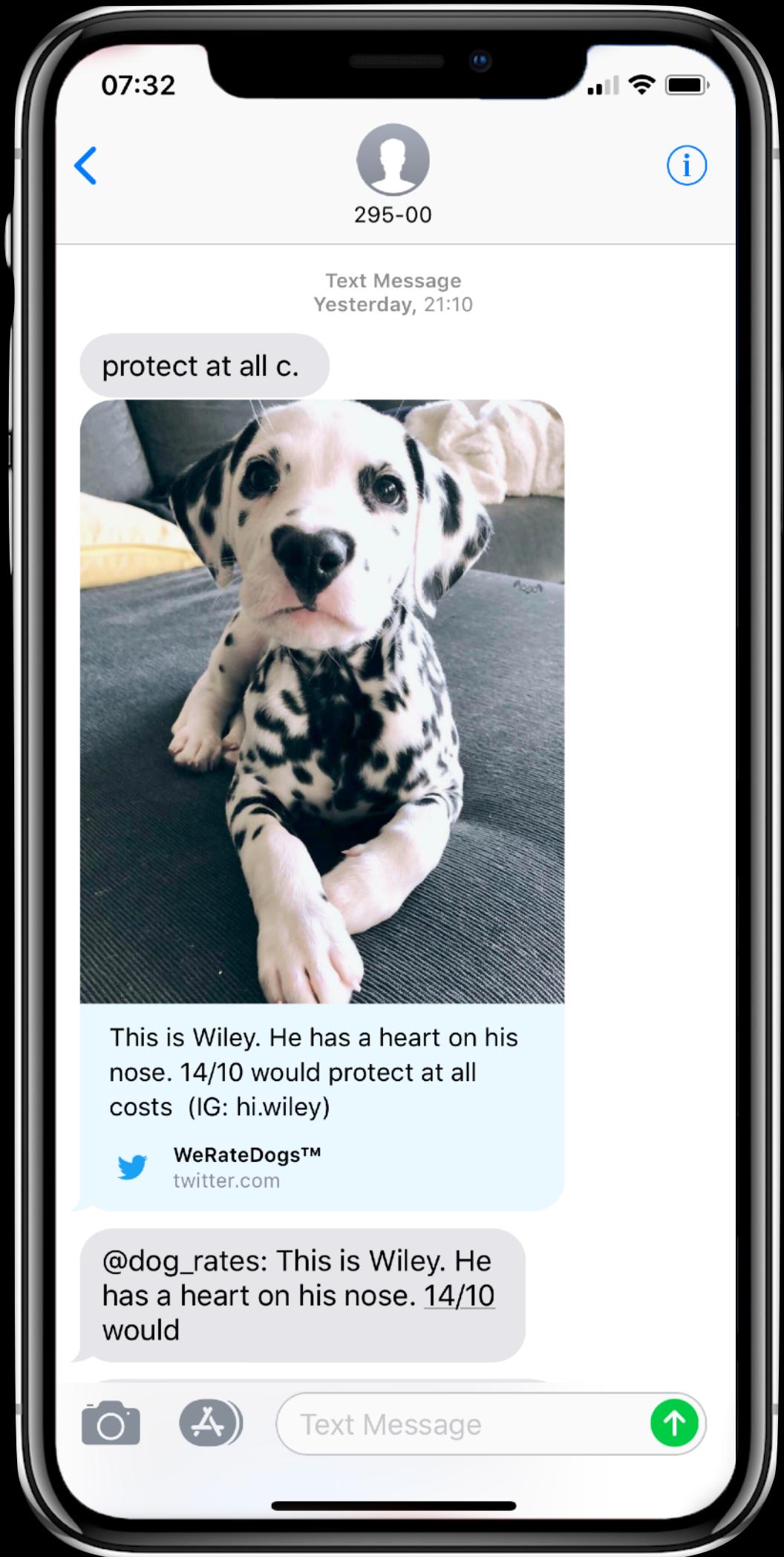
**Message**

@[ UserName ] : [ Text ] (via Twitter  
LinkToTweet)

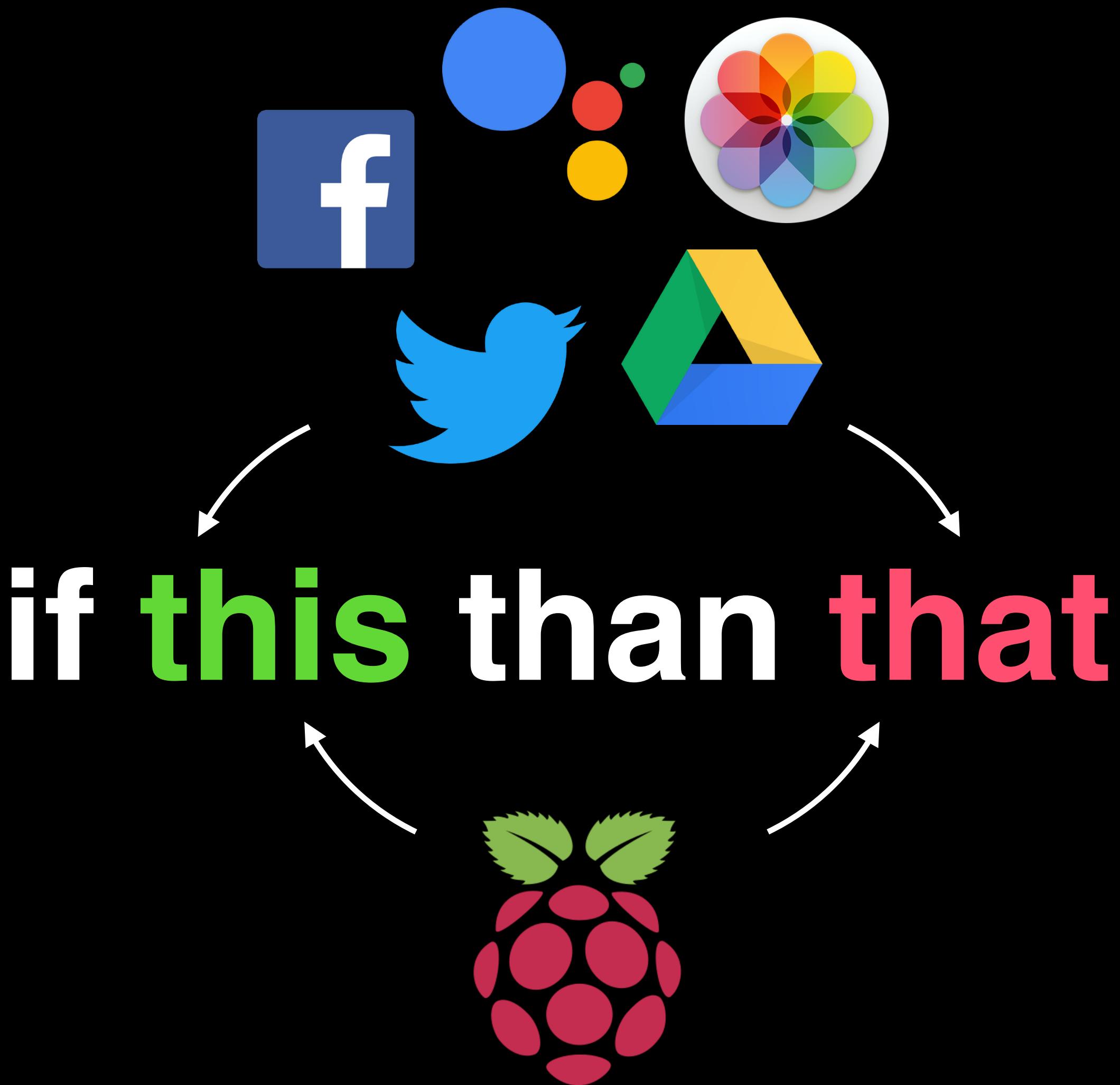
Add ingredient

Create action

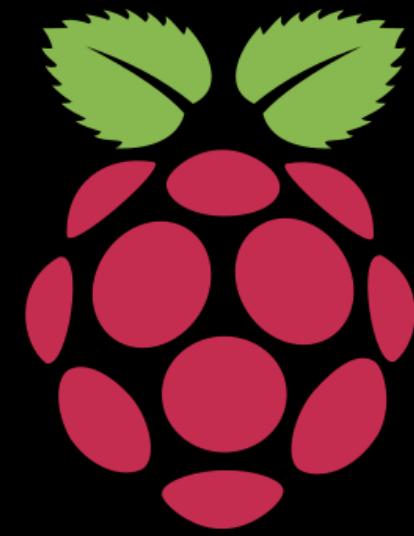
Configuração do Serviço "That"



SMS com Tweets de um Perfil

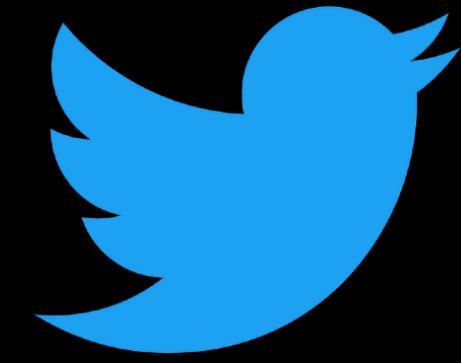


Integração entre Raspberry Pi e Serviços



se botão for  
pressionado

então



poste um  
tweet

Exemplo de Integração entre Raspberry Pi e Twitter

ifttt.com/create/if?sid=0

[Back](#)

# Choose a service

Step 1 of 6

[About](#) [Blog](#) [Help](#) [Jobs](#) [Terms](#) [Privacy](#)

Download on the App Store GET IT ON Google play

Serviço Webhooks como "This"

The screenshot shows a web browser window with the URL [ifttt.com/create/if-receive-a-web-request?sid=2](https://ifttt.com/create/if-receive-a-web-request?sid=2). The title bar includes standard OS X icons for window control and a back button. The main content area has a blue header with the text "Complete trigger fields" and a "Back" link. Below this, it says "Step 2 of 6". A blue box contains the "Receive a web request" trigger details. It describes the trigger as firing every time the Maker service receives a web request. It provides instructions to go to service settings or tap the URL/mobile username. An "Event Name" input field contains the value "botao\_pressionado". A note below the input field specifies that the name should be like "button\_pressed" or "front\_door\_opened". A large blue "Create trigger" button is at the bottom of the box.

**Receive a web request**

This trigger fires every time the Maker service receives a web request to notify it of an event. For information on triggering events, go to your Maker service settings and then the listed URL (web) or tap your username (mobile)

**Event Name**

botao\_pressionado

The name of the event, like "button\_pressed" or "front\_door\_opened"

**Create trigger**

Configuração de um Evento do Webhook

The screenshot shows a web browser window for IFTTT, specifically the 'Complete action fields' step of a recipe. The URL in the address bar is [ifttt.com/create/if-receive-a-web-request-then-post-a-tweet](https://ifttt.com/create/if-receive-a-web-request-then-post-a-tweet). The main title is 'Complete action fields' with a Twitter icon. Below it, it says 'Step 5 of 6'. The central action is 'Post a tweet', which will post a new tweet to the user's Twitter account. A note states: 'This Action will post a new tweet to your Twitter account. NOTE: Please adhere to Twitter's Rules and Terms of Service.' The 'Tweet text' field contains the placeholder text: 'O valor medido pelo sensor de luz é {{Value1}}.' The 'Add ingredient' button is visible. Below the tweet text, there is a card for 'Receive a web request' with three parameters: EventName (Value1), Value2, and Value3.

Post a tweet

This Action will post a new tweet to your Twitter account. NOTE: Please adhere to Twitter's Rules and Terms of Service.

Tweet text

O valor medido pelo sensor de luz é  
{{Value1}}.

Add ingredient

Receive a web request

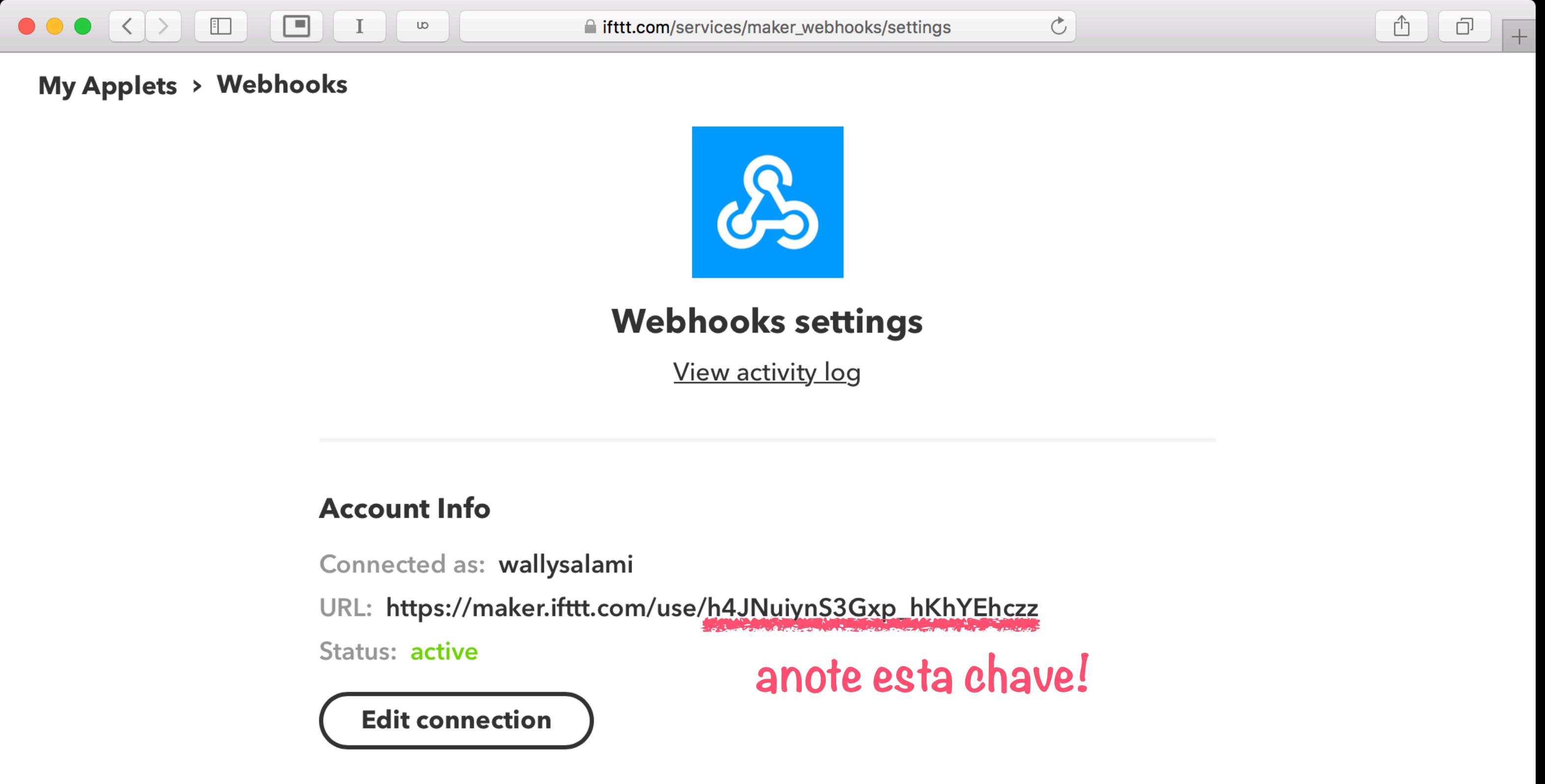
EventName  
Value1

Value2

Value3

Configuração do "That" como Tweet com Parâmetros

[https://ifttt.com/services/maker\\_webhooks/settings](https://ifttt.com/services/maker_webhooks/settings)



The screenshot shows a web browser window with the URL [https://ifttt.com/services/maker\\_webhooks/settings](https://ifttt.com/services/maker_webhooks/settings) in the address bar. The page title is "My Applets > Webhooks". A blue icon representing a webhook is displayed. The main heading is "Webhooks settings". Below it is a link "View activity log". A horizontal line separates this from the "Account Info" section. In the "Account Info" section, it says "Connected as: wallysalami", "URL: [https://maker.ifttt.com/use/h4JNuiynS3Gxp\\_hKhYEhczz](https://maker.ifttt.com/use/h4JNuiynS3Gxp_hKhYEhczz)" (the URL is redacted), and "Status: active". There is a button labeled "Edit connection". To the right of the URL, there is a large red text overlay that reads "anote esta chave!" (note this key!).

Chave Secreta do IFTTT

```
>>> chave = "COLQUE SUA CHAVE AQUI"  
>>> evento = "botao_pressionado"  
>>> endereco = "https://maker.ifttt.com/trigger/" +  
    evento + "/with/key/" + chave
```

Configuração do Endereço do IFTTT

```
>>> from gpiozero import LightSensor, Button  
>>> from requests import post  
>>> sensor_de_luz = LightSensor(8)  
>>> def botao_pressionado():  
...     dados = {"value1": sensor_de_luz.value}  
...     post(endereco, json=dados)  
...  
>>> botao = Button(11)  
>>> botao.when_pressed = botao_pressionado
```

Configuração do Sensor de Luz e do Botão

A screenshot of a Twitter page in a web browser. The tweet is from the account **ENG1419 – Programação de Micro...** (@eng1419). The tweet content is: "O valor medido pelo sensor de luz é 0.7632026672363281." The tweet was posted at 09:33 - 9 de abr de 2018. Below the tweet is a reply input field with the placeholder "Tweete sua resposta". At the bottom of the page, there is a sidebar with the heading "Assuntos para você" and a list of hashtags: #BamBamBlackcard, #TheVoiceKids, #Quantum18, #cblol, #UKSG18, #abc730, #4Corners, #MondayMotivaton, #wrestlemania, #LTSIG. The footer of the page includes links for © 2018 Twitter, Sobre, Central de Ajuda, Termos, Política de privacidade, Cookies, and Informações de anúncios.

Página Inicial    Moments    Notificações    Mensagens    Buscar no Twitter    Tweetar

ENG1419 – Programação de Micro...  
@eng1419

O valor medido pelo sensor de luz é  
0.7632026672363281.

09:33 - 9 de abr de 2018

Tweete sua resposta

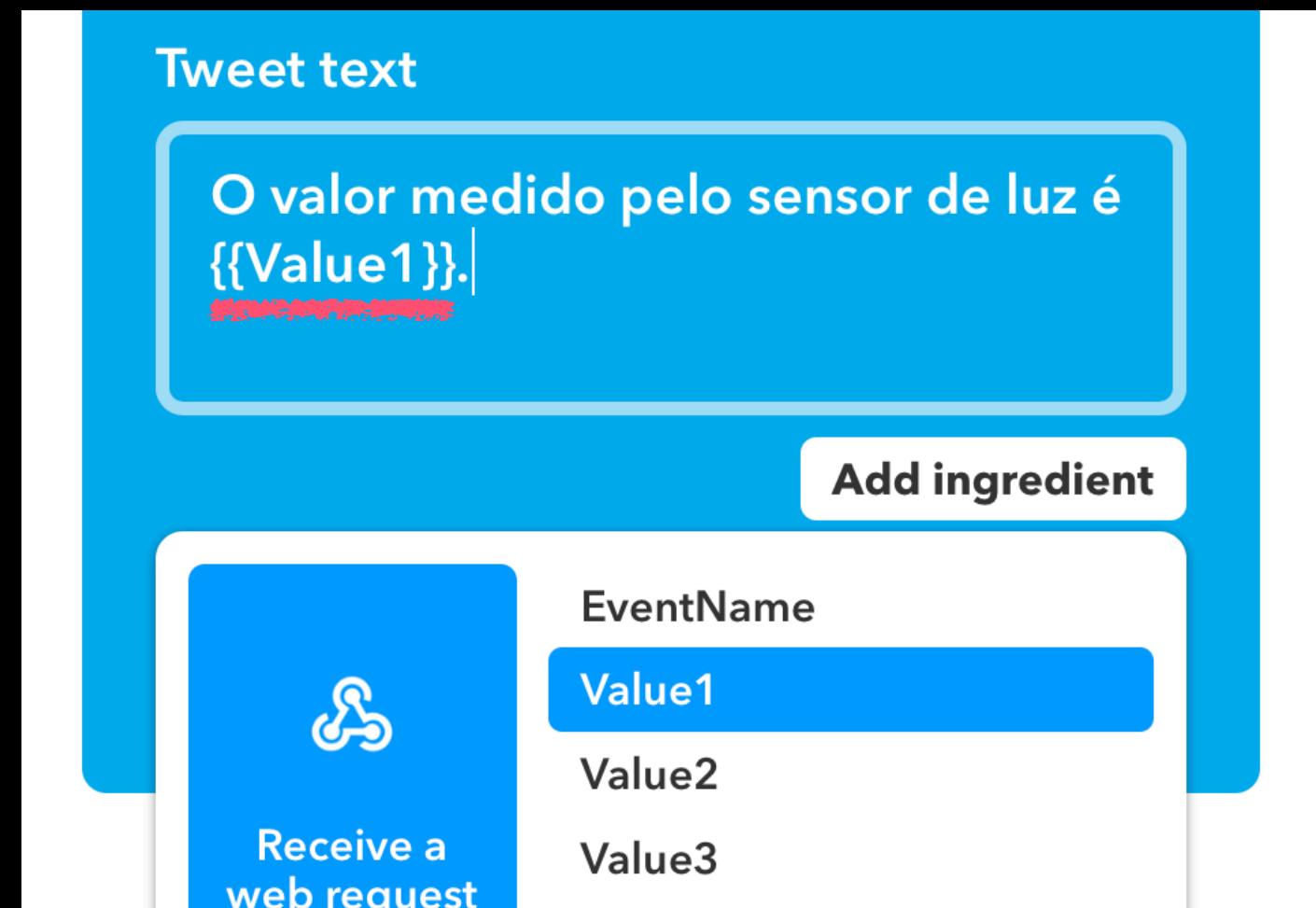
Assuntos para você

#BamBamBlackcard #TheVoiceKids #Quantum18 #cblol #UKSG18 #abc730 #4Corners  
#MondayMotivaton #wrestlemania #LTSIG

© 2018 Twitter Sobre Central de Ajuda Termos Política de privacidade Cookies  
Informações de anúncios

Tweet Gerado pelo Raspberry Pi

"V" MAIÚSCULO →



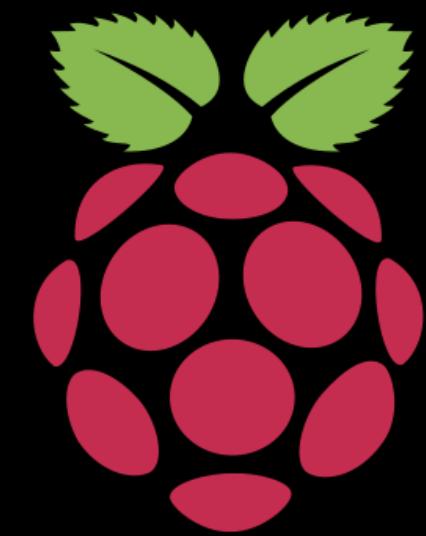
← "v" minúsculo

```
dados = {"value1": sensor_de_luz.value}
```

Atenção à Variação da Ortografia de Value1



então



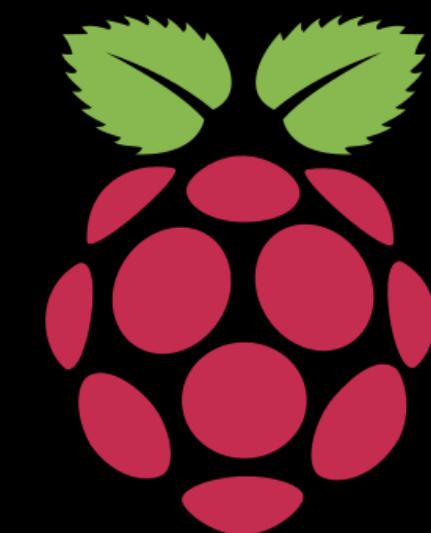
se houver  
previsão do tempo

mostre a previsão  
no LCD

Exemplo de Integração entre Weather Underground e Raspberry Pi



`https://.../tempo/rain`



`https://.../tempo/sunny`



Integração entre Weather Underground e Raspberry Pi via Requisições Web

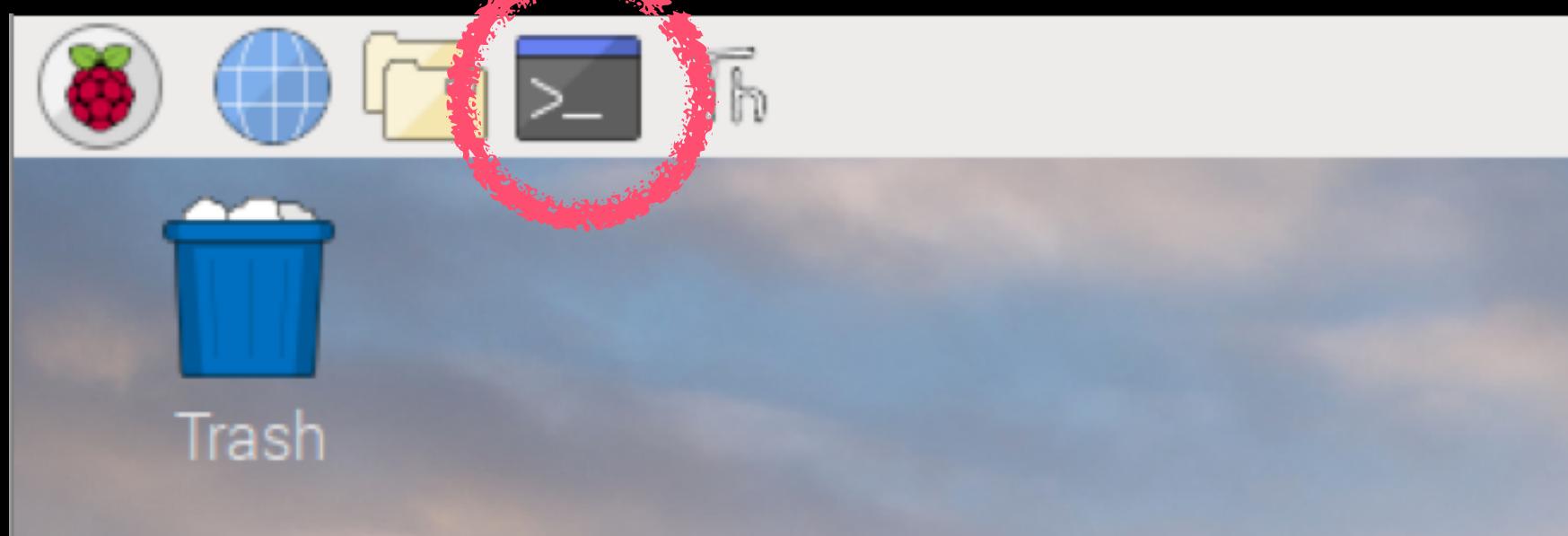
```
from flask import Flask
from Adafruit_CharLCD import Adafruit_CharLCD
from gpiozero import Buzzer

app = Flask(__name__)
lcd = Adafruit_CharLCD(2, 3, 4, 5, 6, 7, 16, 2)
campainha = Buzzer(16)

@app.route("/tempo/<string:previsao>")
def tempo(previsao):
    lcd.message("Tempo hoje:\n" + previsao)
    if previsao == "Rain":
        campainha.beep(n=10)
    return "A previsão do tempo é: " + previsao

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

Configuração do Servidor com Página para Previsão do Tempo



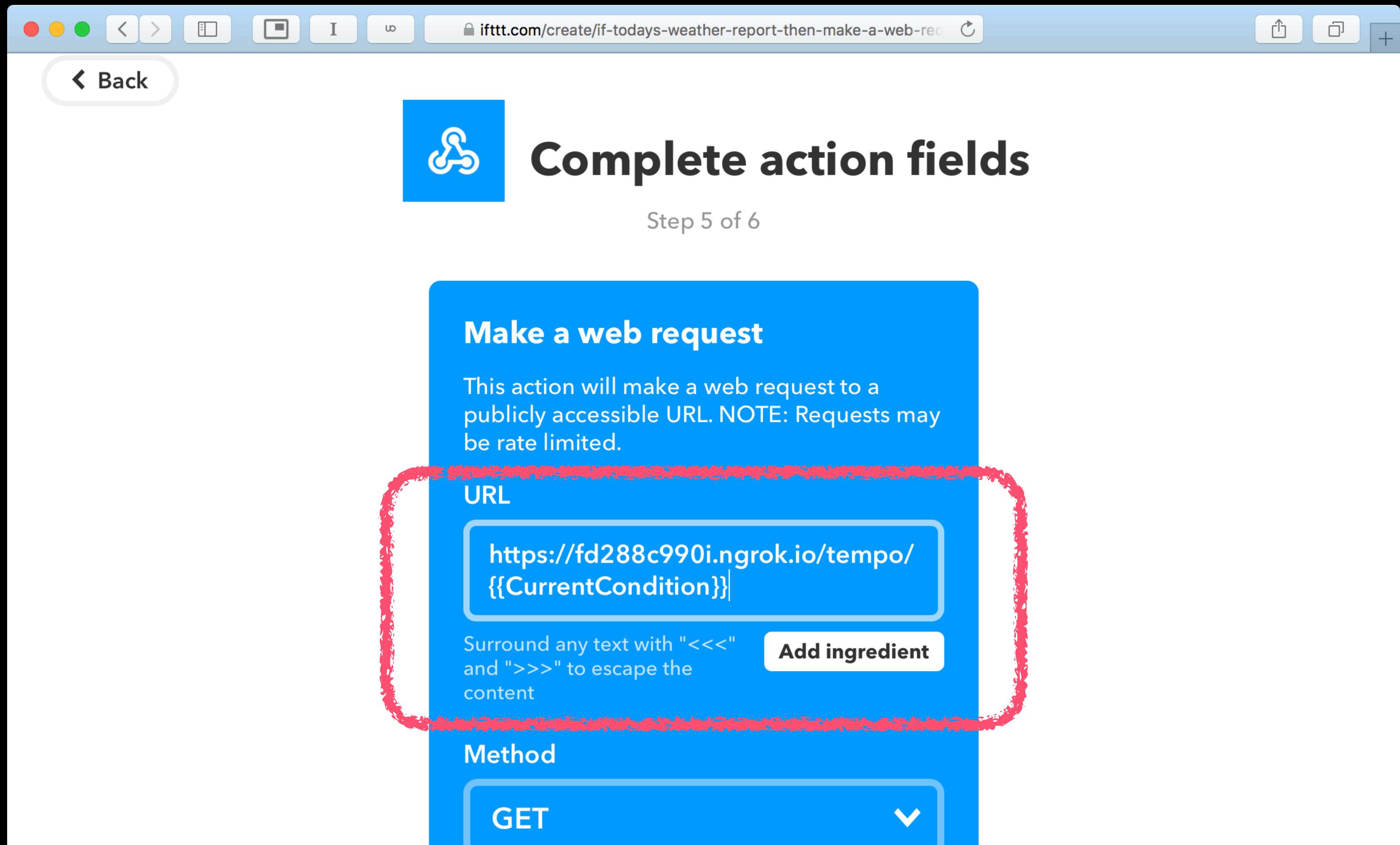
```
aula@raspberrypi ~ $ ngrok http 5000
```

```
ngrok by @inconshreveable  
(Ctrl+C to quit)
```

Session Status	online
Session Expires	7 hours, 59 minutes
Version	2.2.8
Region	United States (us)
Web Interface	<a href="http://127.0.0.1:4040">http://127.0.0.1:4040</a>
Forwarding	<a href="http://fd288c990i.ngrok.io">http://fd288c990i.ngrok.io</a> -> localhost:5000
Forwarding	<a href="https://fd288c990i.ngrok.io">https://fd288c990i.ngrok.io</a> -> localhost:5000
Connections	ttl      opn      rt1      rt5      p50      p90
	0          0          0.00    0.00    0.00    0.00

A screenshot of a web browser window showing the IFTTT platform. The URL in the address bar is `ifttt.com/create/if-todays-weather-report?sid=6`. The main content area is titled "Complete trigger fields" with a "Step 2 of 6" subtitle. On the left, there is a small "w" logo icon. The central part of the screen displays a "Today's weather report" trigger configuration. It includes a description: "This Trigger retrieves today's current weather report at the time you specify. NOTE: Pollen count available only in the USA." Below this, there are two dropdown menus for "Time of day": one set to "08 AM" and another set to "00 Minutes". At the bottom is a large, rounded rectangular button labeled "Create trigger".

Configuração do Weather Underground como "This"



[ifttt.com/create/if-todays-weather-report-then-make-a-web-rec](#)

< Back

## Complete action fields

Step 5 of 6

**Make a web request**

This action will make a web request to a publicly accessible URL. NOTE: Requests may be rate limited.

**URL**

`https://fd288c990i.ngrok.io/tempo/{{CurrentCondition}}`

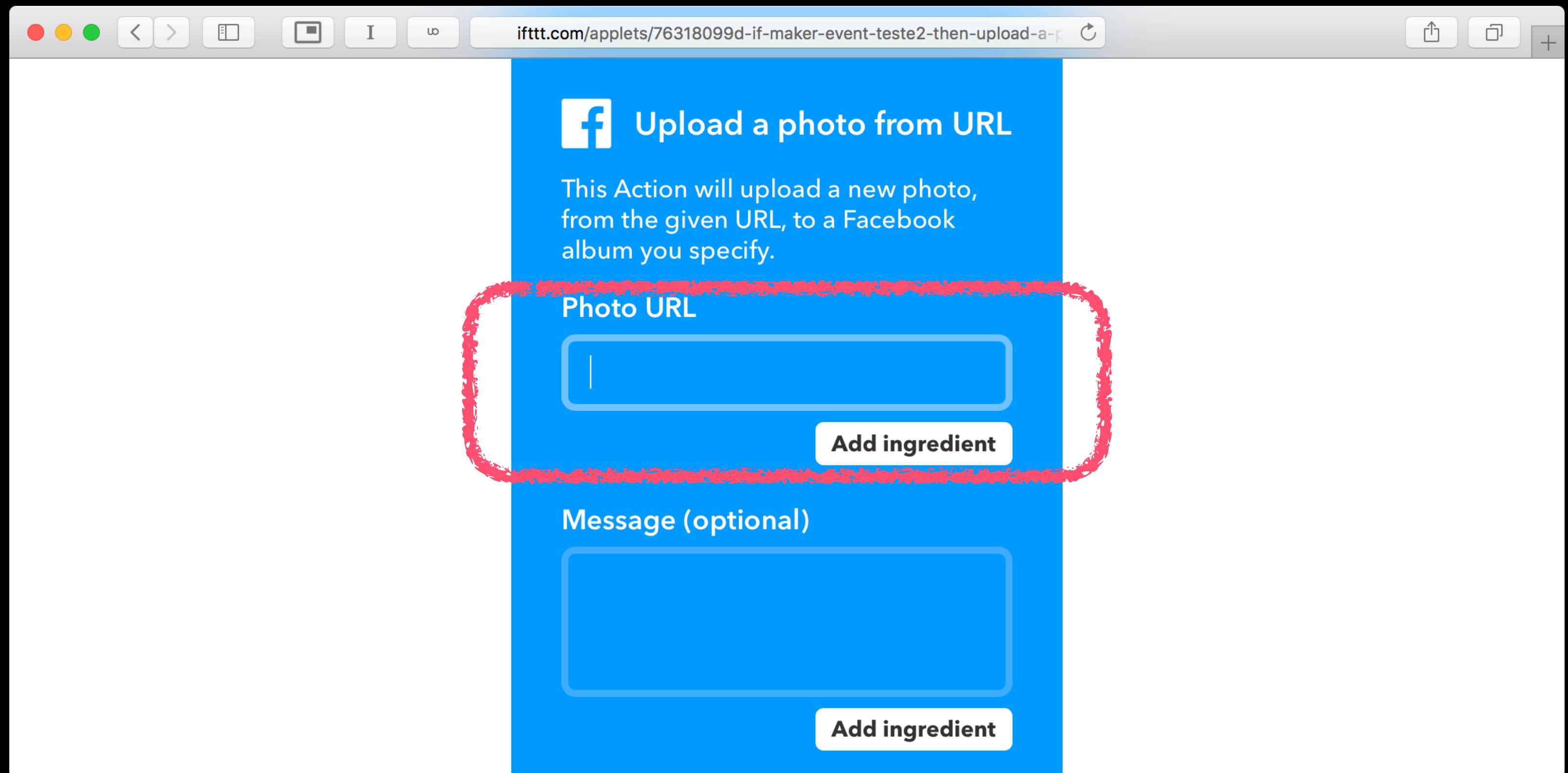
Surround any text with "<<<" and ">>>" to escape the content

**Add ingredient**

**Method**

GET

Configuração do Webhook como "That"



URL para Envio de Arquivos

localhost:5000/foto1234.jpeg

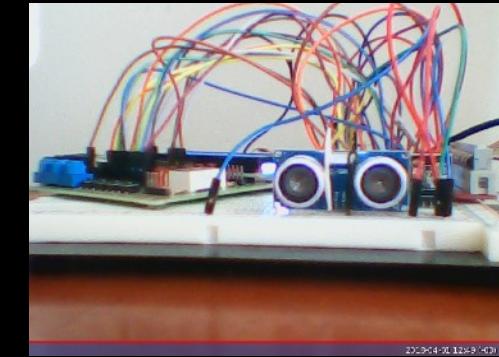


foto1234.jpeg

```
from flask import Flask, send_from_directory  
  
app = Flask(__name__)  
  
@app.route("/<string:nome_da_foto>")  
def foto(nome_da_foto):  
    return send_from_directory("pasta", nome_da_foto)  
  
app.runserver(port=5000, debug=False)
```

# Resumo da Ópera

## Funcionalidade

FSWebcam

[acessar documentação](#)

ARecord

[acessar documentação](#)

Lame

[acessar documentação](#)

OpusEnc

[acessar documentação](#)

Telegram

[acessar documentação](#)

## Comandos

```
from os import system  
system("fswebcam foto.jpg")  
system("fswebcam --resolution 640x480 foto.jpg")
```

```
from os import system  
system("arecord --duration 5 --format cd audio.wav")  
from subprocess import POpen  
aplicativo = POpen("arecord audio.wav") • aplicativo.terminate()
```

```
from os import system  
system("lame audio.wav audio.mp3")
```

```
from os import system  
system("opusenc audio.wav audio.ogg")
```

```
from requests import post, get  
base = "https://api.telegram.org/bot" + chave  
endereco = base + "/sendMessage"  
dados = {"chat_id": id_da_conversa, "text": "Olá!"}  
post(endereco, json=dados)  
endereco = base + "/sendPhoto" • endereco = base + "/sendVoice"  
arquivos = {"photo": open("foto.jpg", "rb")}  
post(endereco, data=data, files=arquivos)  
resultado = get(base + "/getUpdates", json=dados)
```

## Funcionalidade

### Datas e Horas

[acessar documentação](#)

## Comandos

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

### Campainha

[acessar documentação](#)

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

### Sensor de Distância

[acessar documentação](#)

```
from gpiozero import DistanceSensor
sensor = DistanceSensor(trigger=17, echo=18)
sensor.distance • sensor.threshold_distance
sensor.wait_for_in_range() • sensor.wait_for_out_of_range()
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_one(dados) • colecao.insert_many(lista_de_dados)
busca = {"chave": valor} • documento = colecao.find_one(busca)
documentos = colecao.find_many(busca) • list(documentos)
colecao.find_many(busca, sort=ASCENDING)
```

## 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 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"
```

HTML

```
<p>Parágrafo</p>

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

Ngrok

abrir Terminal → ngrok http 5000

## Funcionalidade

LED

[acessar documentação](#)

Botão

[acessar documentação](#)

LCD

[acessar documentação](#)

MPlayer

## Comandos

```
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)
```

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

```
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()
```

```
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

### Entrada / Saída

```
x = input("Digite um número: ") • print("Resultado: ", x)
```

### Listas acessar documentação

```
lista = [1, 2, 3] • lista2 = ["texto", [0, 0], 5];  
lista[0] • lista.append(elemento) • lista.remove(indice)
```

### Dicionários acessar documentação

```
dicionario = {"chave 1": 42, "chave 2": [1, 2, 3]}  
dicionario["chave 1"] • dicionario["chave 3"] = "Olá!"
```

### Textos

```
x = "aspas duplas" • y = 'aspas simples' • x + "\n" + y
```

### Condicionais

```
if x == 0:  
    y = 4  
else:  
    if x != 0:  
        if x not in [1, 2]:  
            y = 4  
        else:  
            y = 0  
    elif x >= 0:  
        y = 3  
    else:  
        y = 0
```

### Repetições

```
for i in [1, 2, 3]:  
    ...  
for i in range(1, 4):  
    ...  
while x > 1:  
    ...
```

### Criação de Funções

```
def funcao1(x):  
    return x + 2  
def funcao2(x, y, z):  
    ...  
def funcao3():  
    ...
```

## Comandos

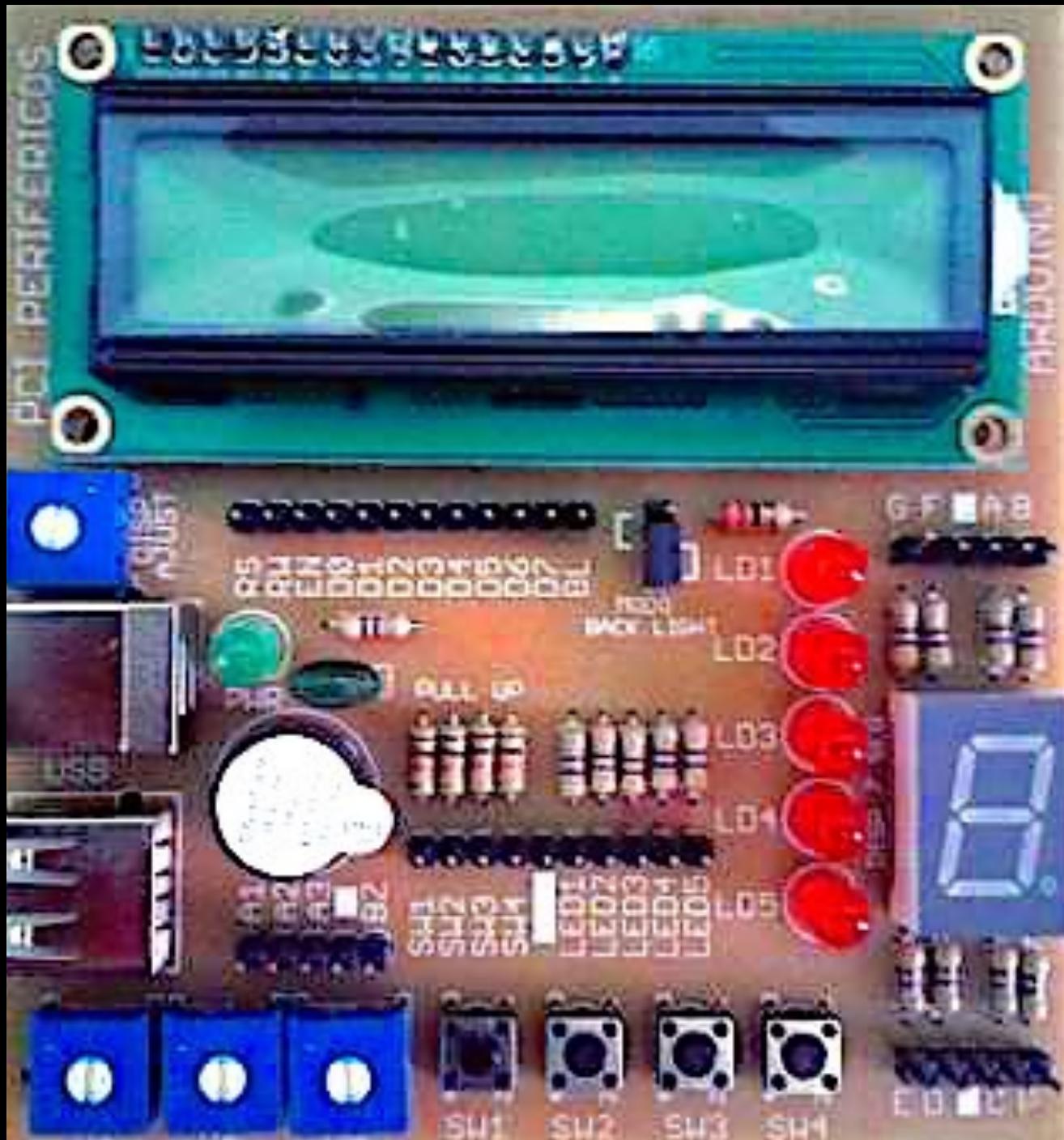
# Prática



[janks.link/micro/projeto05.zip](https://janks.link/micro/projeto05.zip)

# Testes Iniciais

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

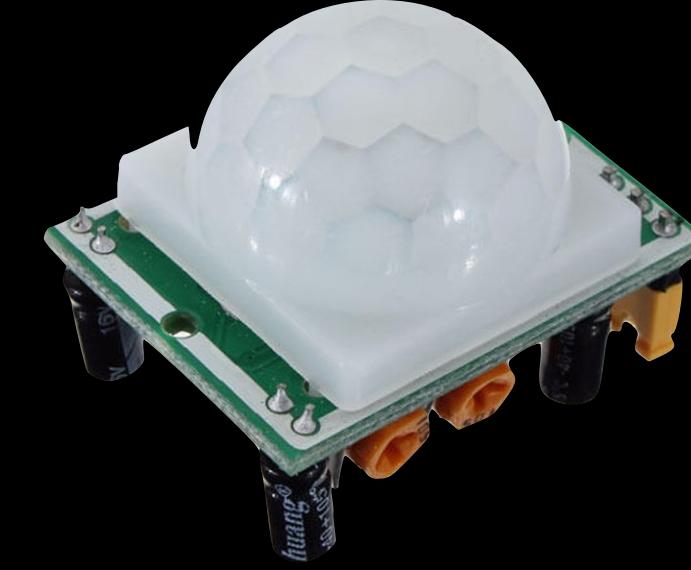


GPIO 16

GPIO 11, 12, 13, 14

GPIO 21  
GPIO 22  
GPIO 23  
GPIO 24  
GPIO 25

GPIO 8



GPIO 27

Conexões com as Portas da GPIO



## Testes Iniciais

Faça com que a intensidade de brilho do LED 1 varie de acordo com a luz captada no sensor.

↪ DICA: use um **While True**.

Acenda o LED 2 ao detectar um movimento, e apague-o ao detectar a inércia.

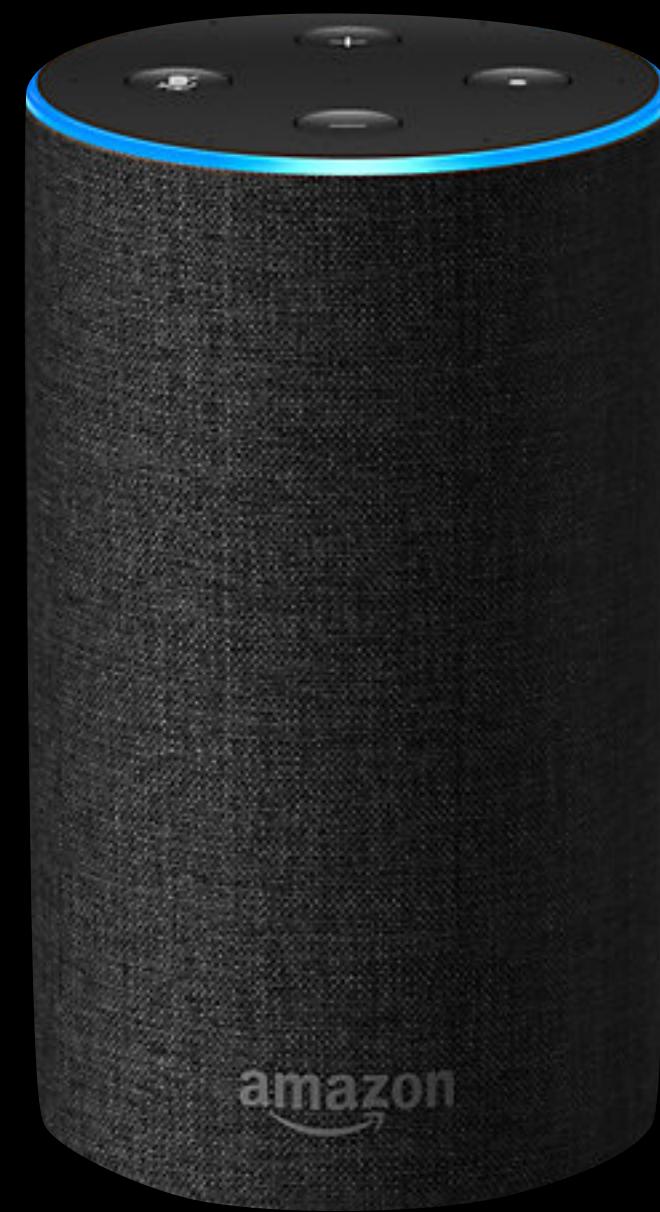
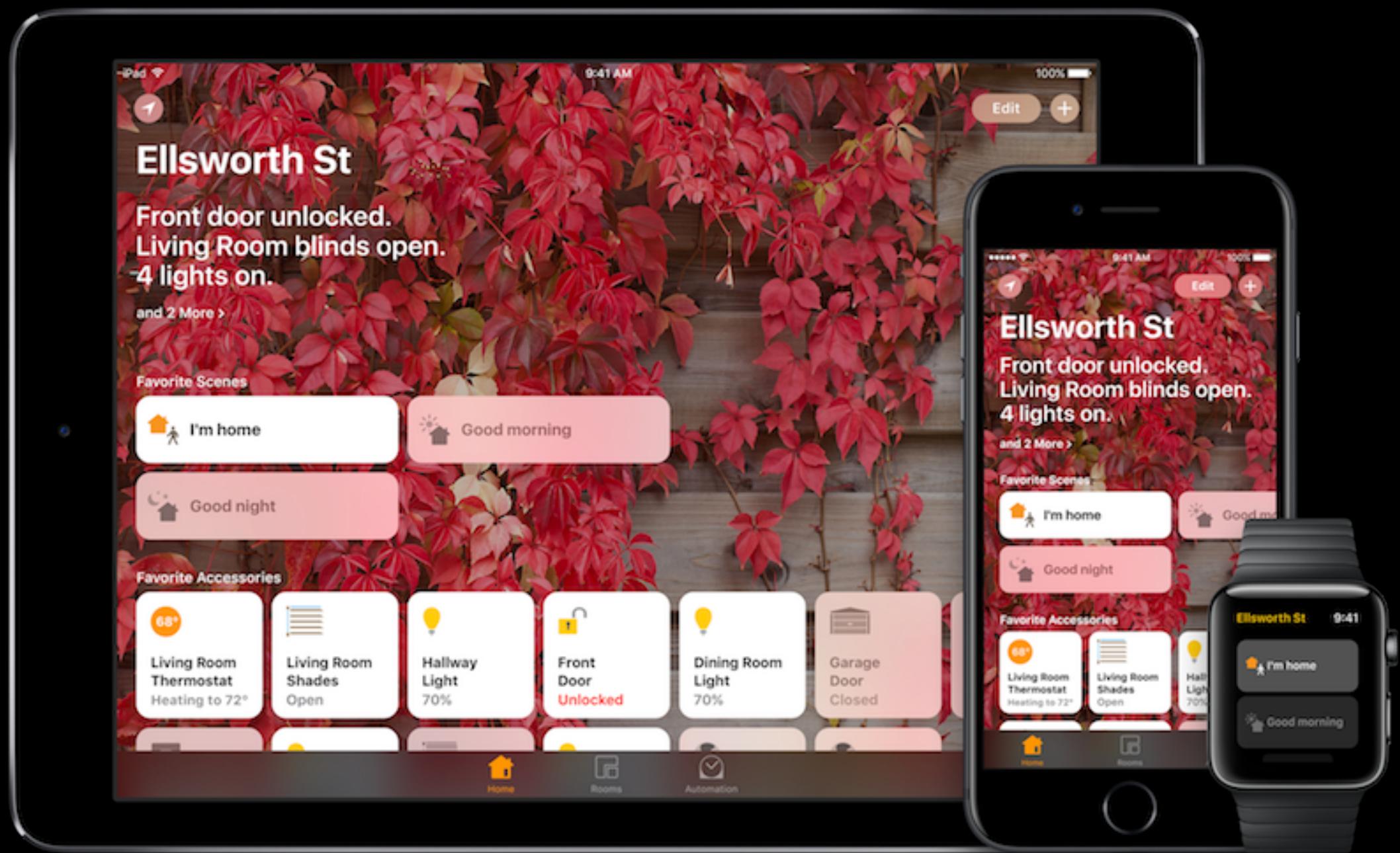
Acenda o LED 3 ao detectar um movimento e apague-o somente se não houver movimento por 8 segundos.

↪ DICA: use um **Timer** em caso de inércia e cancele-o se houver movimento.

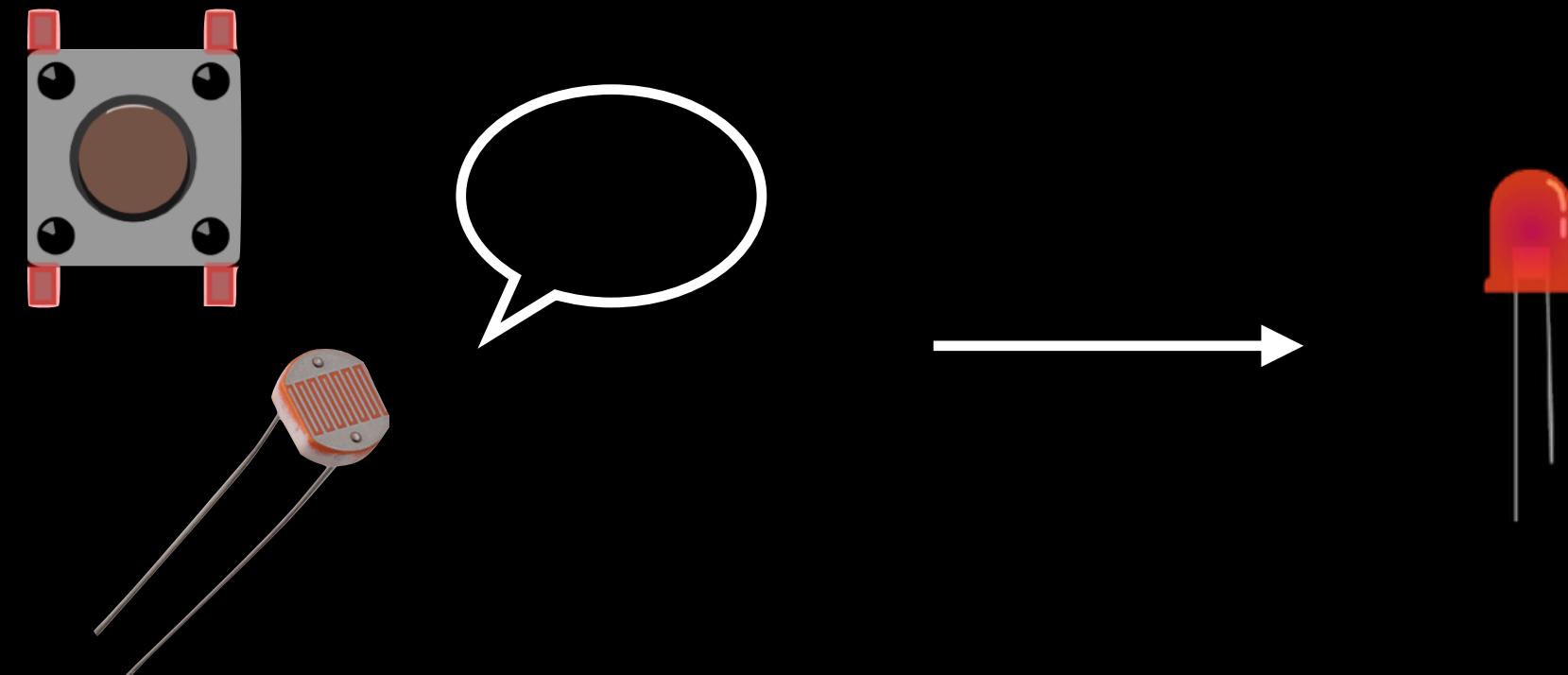
Configure a integração do IFTTT com o Google Sheets.

Ao apertar o botão 1, adicione uma linha com um texto qualquer em uma planilha do Google Sheets.

# Implementação



Smart Home

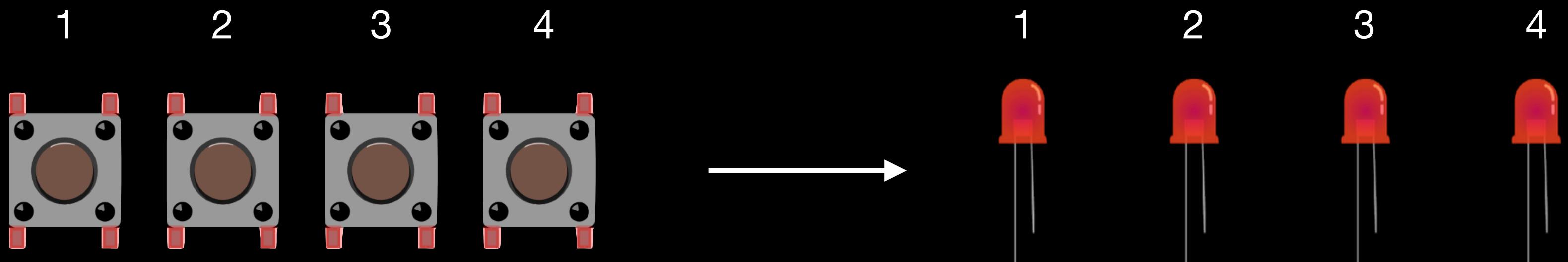


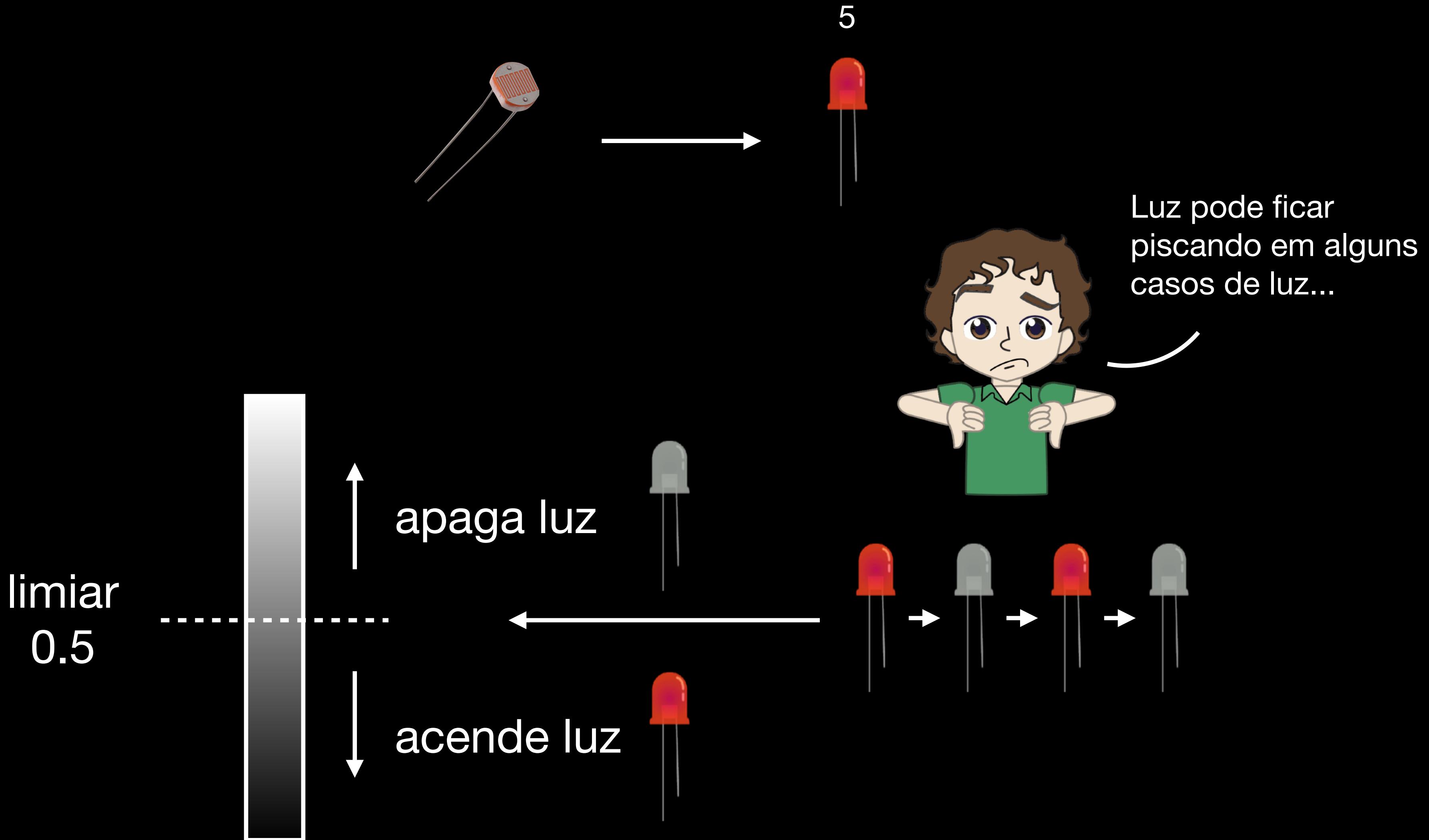
A screenshot of a Google Sheets document titled "Luzes". The spreadsheet contains a log of light status changes over time. The columns represent the row number, the light number, the status (aceso or apagado), and the timestamp. The data is as follows:

	A	B	C	D	E	F	G
1	Luz 1	aceso	April 8, 2018 at 06:30PM				
2	Luz 1	apagado	April 8, 2018 at 06:30PM				
3	Luz 2	aceso	April 8, 2018 at 06:31PM				
4	Luz 2	apagado	April 8, 2018 at 06:31PM				
5	Luz 3	aceso	April 8, 2018 at 06:31PM				
6	Luz 1	aceso	April 8, 2018 at 06:31PM				
7	Luz 2	aceso	April 8, 2018 at 06:31PM				
8	Luz 1	aceso	April 8, 2018 at 08:34PM				
9	Luz 2	aceso	April 8, 2018 at 08:34PM				
10	Luz 2	apagado	April 8, 2018 at 08:34PM				
11	Luz 1	apagado	April 8, 2018 at 08:34PM				
12	Luz 5	aceso	April 8, 2018 at 08:35PM				
13	Luz 5	apagado	April 8, 2018 at 08:36PM				
14	Luz 2	aceso	April 8, 2018 at 08:39PM				
15	Luz 2	apagado	April 8, 2018 at 08:39PM				

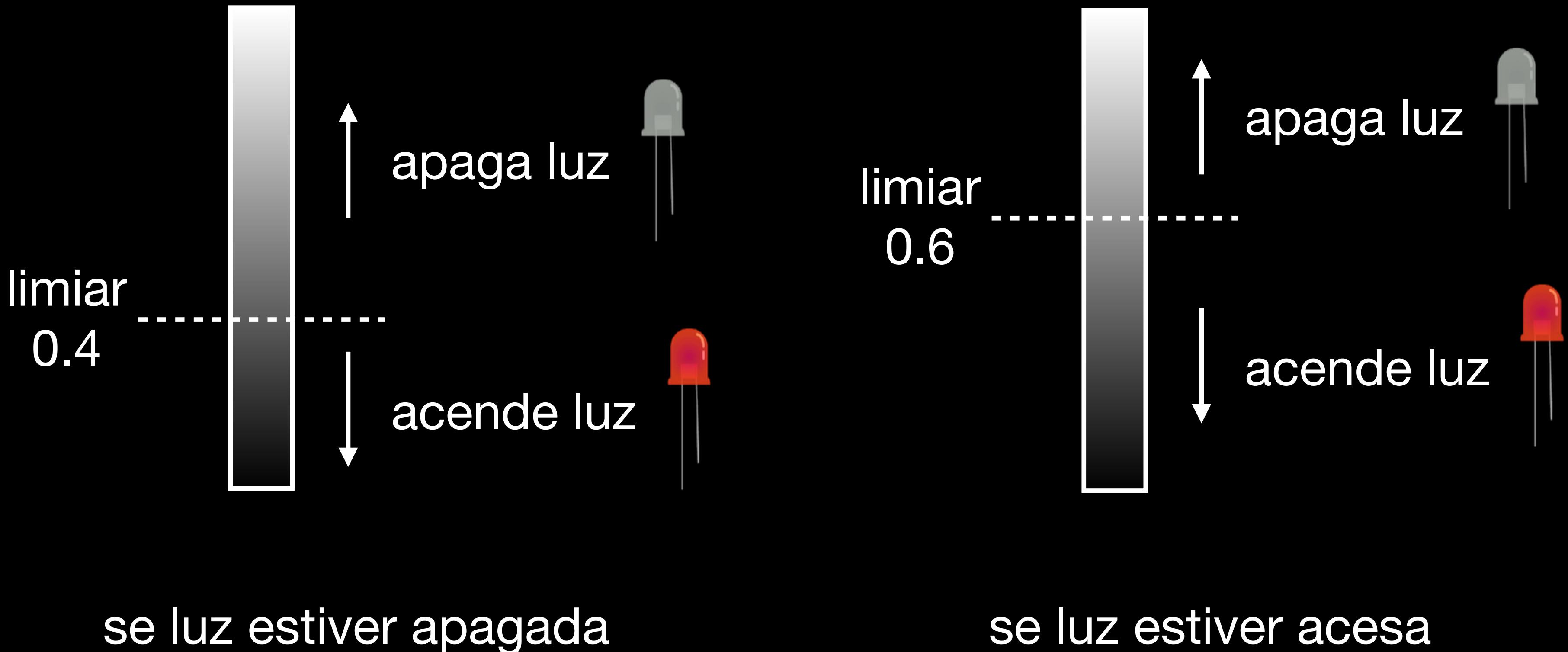
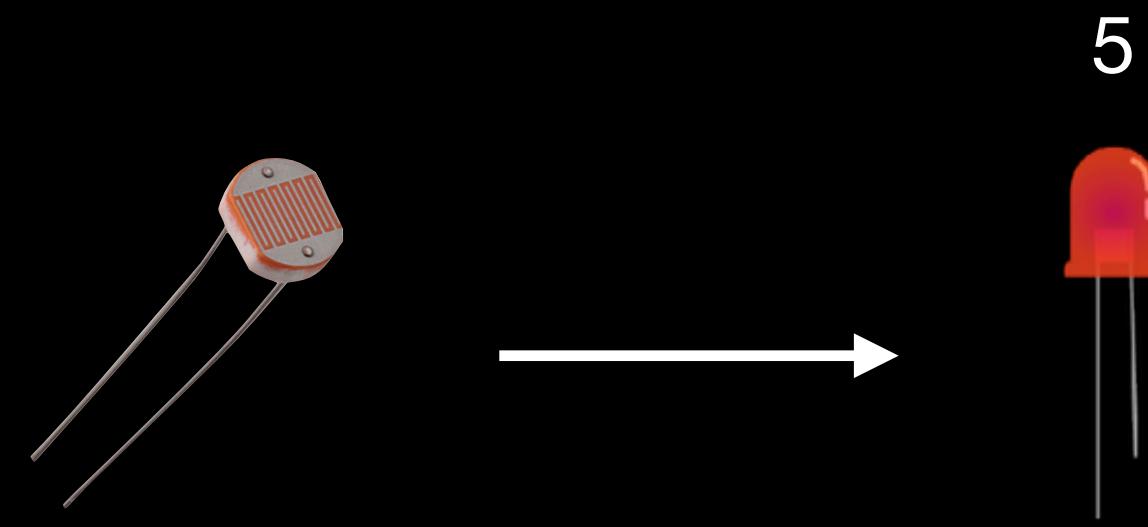
Smart Home Caseiro com Raspberry Pi

aperte para ligar, aperte de novo para desligar





Parte 2: Controle de Luzes por Sensor de Luz com 1 Limiar



turn **on** light number 3

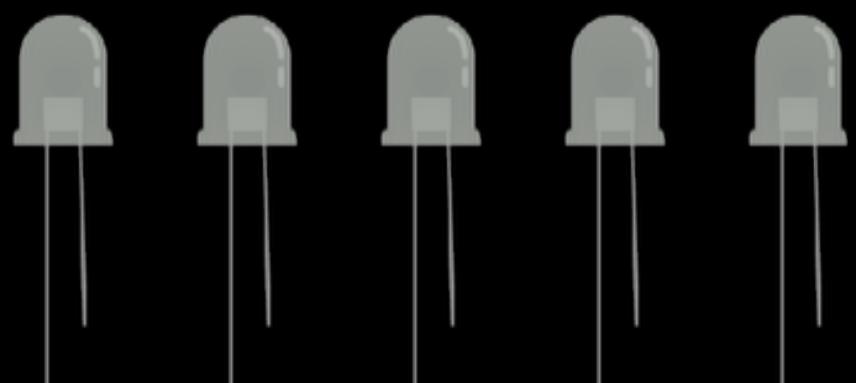
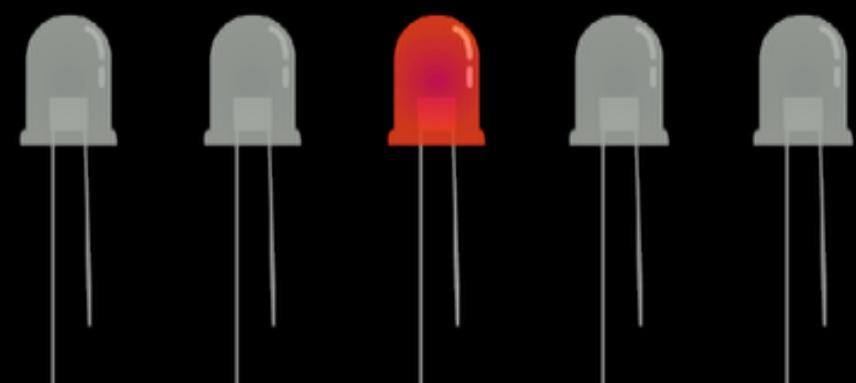
turn **off** light number 3



<http://.../luz/3/on>



<http://.../luz/3/off>





## Implementação

Implemente a função **salvar\_na\_planilha** que receba o número do LED e o seu estado, gerando uma nova linha na planilha (como foi ilustrado). Teste a função manualmente.  
↪ DICA: crie uma lista de LEDs no script.

Ao apertar os botões 1, 2, 3 e 4, alterne o estado dos LEDs 1, 2, 3 e 4, respectivamente. Depois, atualize a planilha.

Ao detectar escuridão, acenda o LED 5. Ao detectar luz, apague-o. Exiba o valor do sensor de luz no LCD. Por fim, atualize a planilha.  
↪ DICA: varie o threshold como explicado.

Crie um **servidor** com uma página que receba o número do LED e o estado desejado (on/off), e altere o LED solicitado. Acesse a página em localhost:5000.  
↪ DICA: use os parâmetros `int` e `string` na rota. Teste inicialmente com localhost:5000, depois use o Ngrok.

Integre o **Google Assistant** com o IFTTT para controlar as luzes por voz pelo app. E, mais uma vez, não esqueça de atualizar a planilha.

...

```
print("Enviando linha para planilha...")
```

...

```
print("Planilha enviada!")
```

...

```
resposta = post(endereco, json=dados)
print(resposta.text)
```

...

DICA: Acompanhamento das Etapas com Print

# Aperfeiçoamento



Alguém entrou na casa! Fotos:



Email

Alguém entrou na casa!<br>

<img src='https://.../foto15:25:37.jpeg'><br>

<img src='https://.../foto15:25:40.jpeg'><br>

<img src='https://.../foto15:25:43.jpeg'><br>

...

Email com Fotos ao Detectar Movimento

## Aperfeiçoamento



Ao detectar movimento, inicie um Timer que capture uma foto a cada 3 segundos; ao detectar inércia, cancele o timer. Salve as fotos com a hora no nome na pasta "fotos".

↪ DICA: veja o slide a seguir.

Crie um servidor com uma página que receba o nome da foto e a retorne. Teste a página no navegador com o localhost:5000.

Em caso de inércia, envie um email com as fotos caso haja pelo menos 3 delas.

↪ DICA: crie uma lista com o nome das fotos tiradas desde a última inércia. Gere o texto do email com tags HTML, conforme exemplificado.

```
>>> from datetime import datetime  
>>> agora = datetime.now()  
>>> hora = agora.strftime("%H:%M:%S")  
>>> nome_arquivo = "foto-" + hora + ".jpeg"  
>>> nome_arquivo  
foto-15:25:37.jpeg
```

DICA: Nome da Foto com Data e Hora