

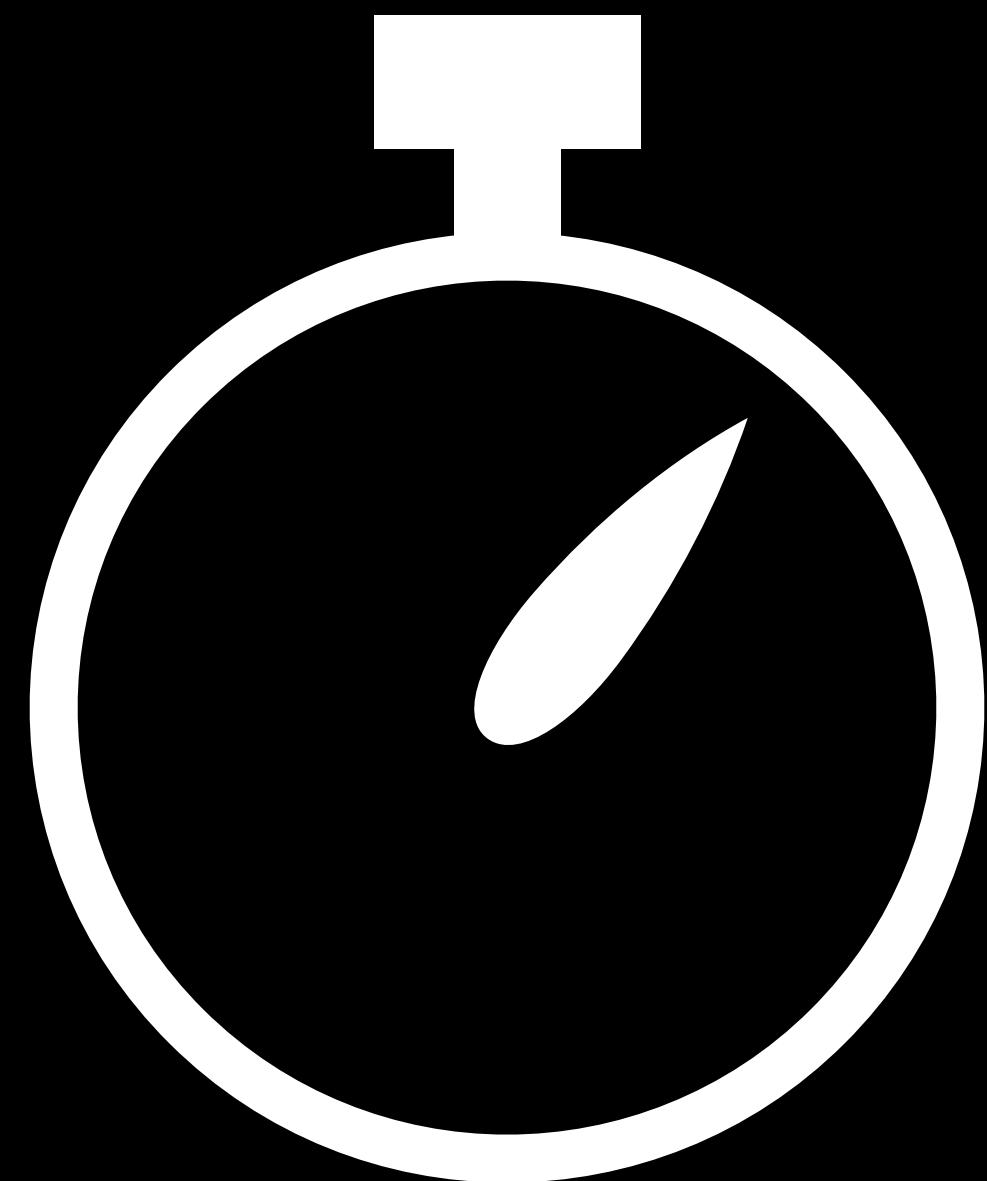
# Projeto 05

## Controle Automático – Teoria

Jan K. S. – janks@puc-rio.br

ENG1419 – Programação de Microcontroladores

# Ferramentas do Python



Timer

início do timer

após um tempo  
----->

execução da  
função desejada

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



3.5 segundos depois

Acabouuuu, acabou!

Ao contrário da sleep,  
Timer não trava a execução!



```
>>> timer = Timer(3.5, timer_acabou)  
>>> timer.start()  
>>> timer.cancel()
```

Cancelamento de Timer

```
>>> timer = Timer(3.5, timer_acabou)
>>> timer.start()
>>> timer.cancel()
>>> timer.start()
Traceback (most recent call last):
...
RuntimeError: threads can only be started once
```

```
# precisamos criar o timer novamente
>>> timer = Timer(3.5, timer_acabou)
>>> timer.start()
```

## Timer Único

daqui a 3.5 segundos, 3.5 segundos -----> Acabou!

## Timer Recorrente

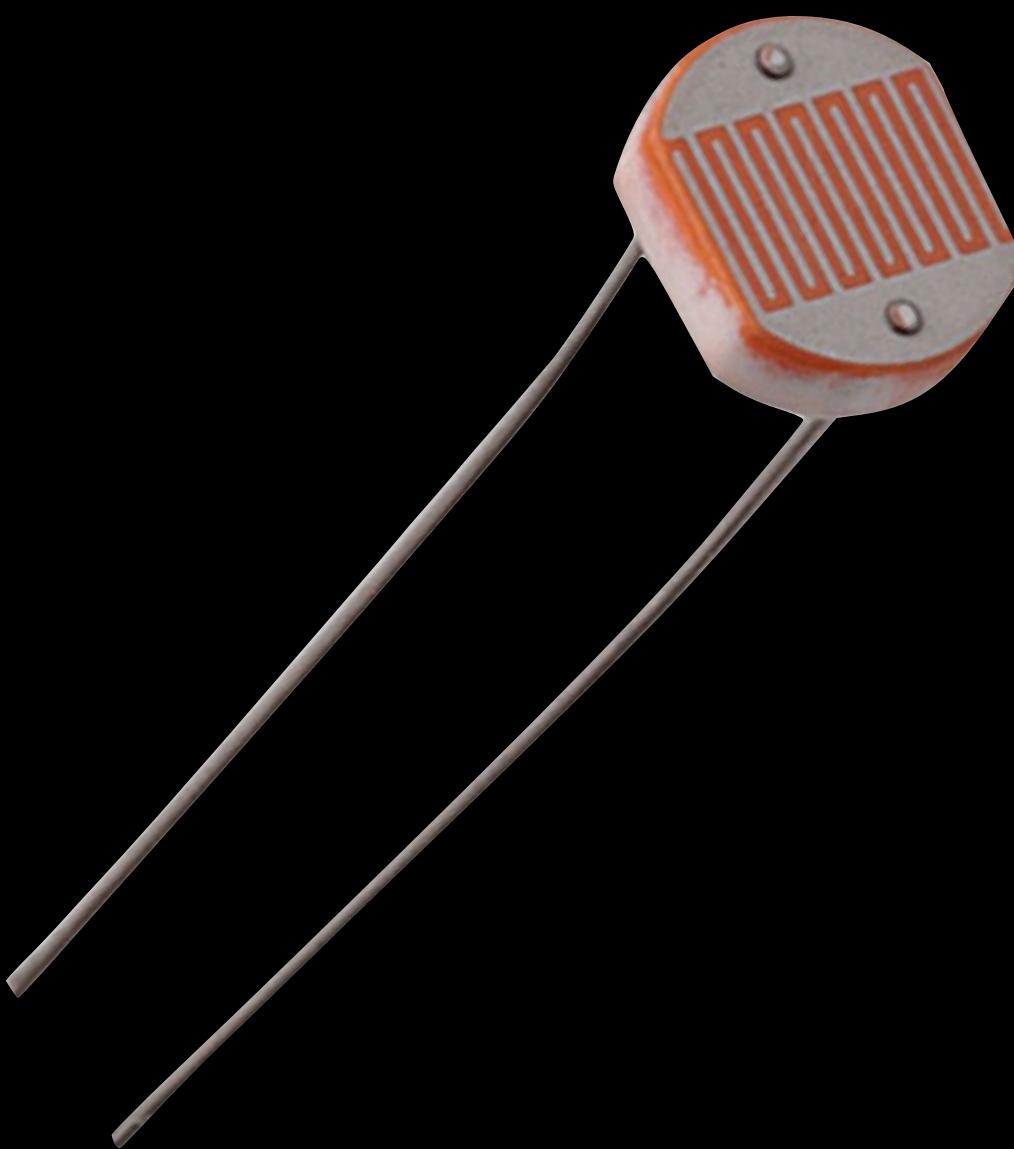
1 segundo

```
imprima "Repetição"
---> daqui a 1 segundo, -----
      repita o processo
      ↑
      cancele o timer!
```

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

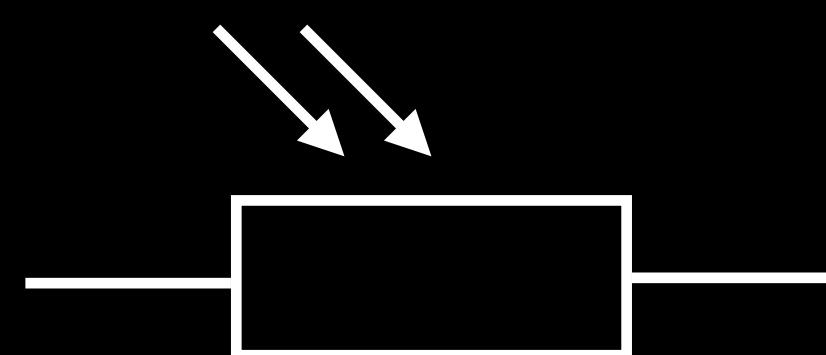
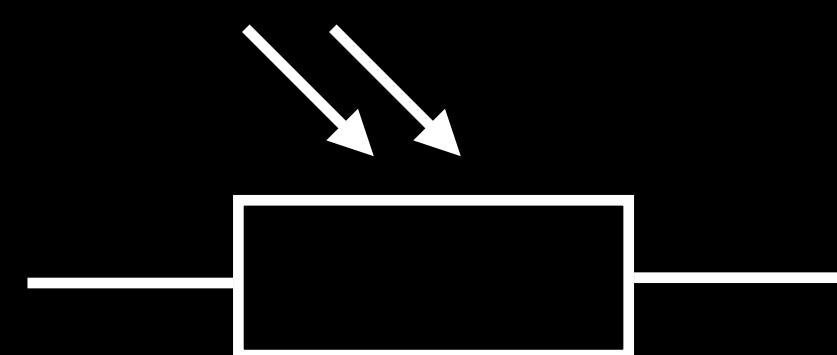
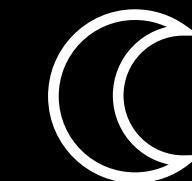
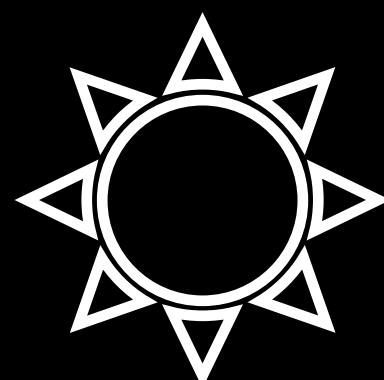
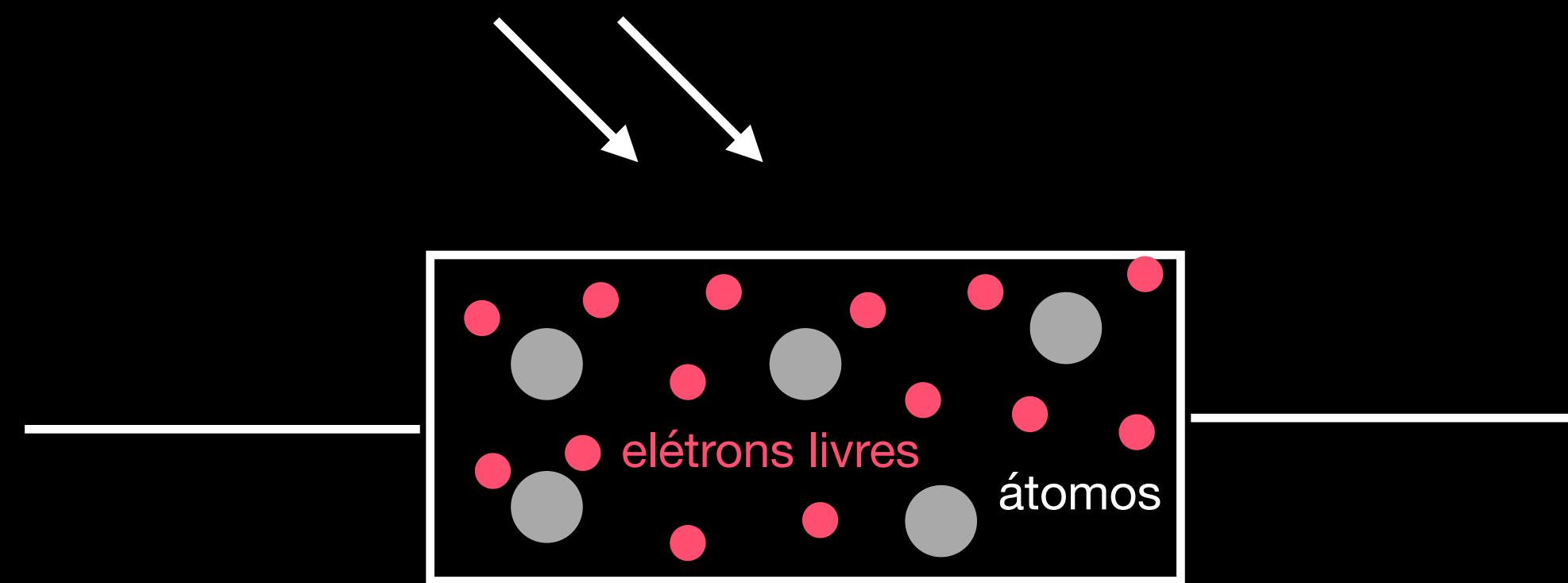
chame depois a própria função!

# Hardware

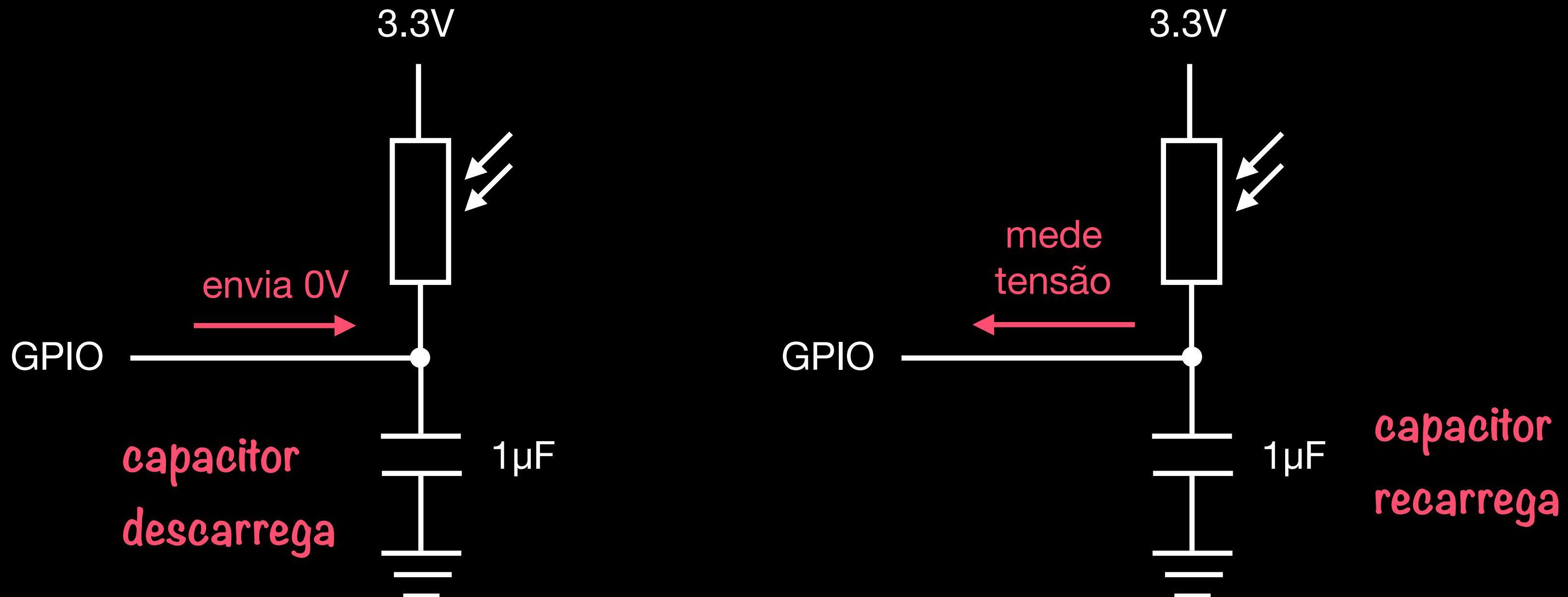


Sensor de Luz

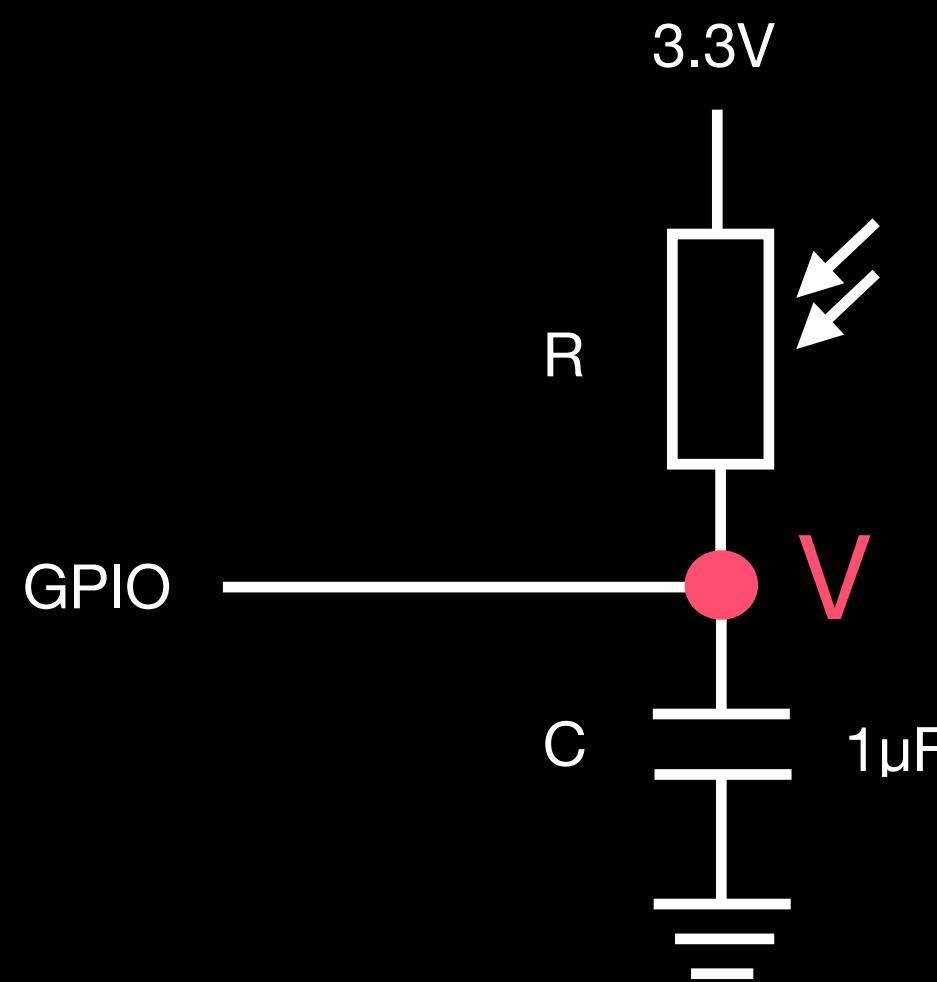
fótons da luz



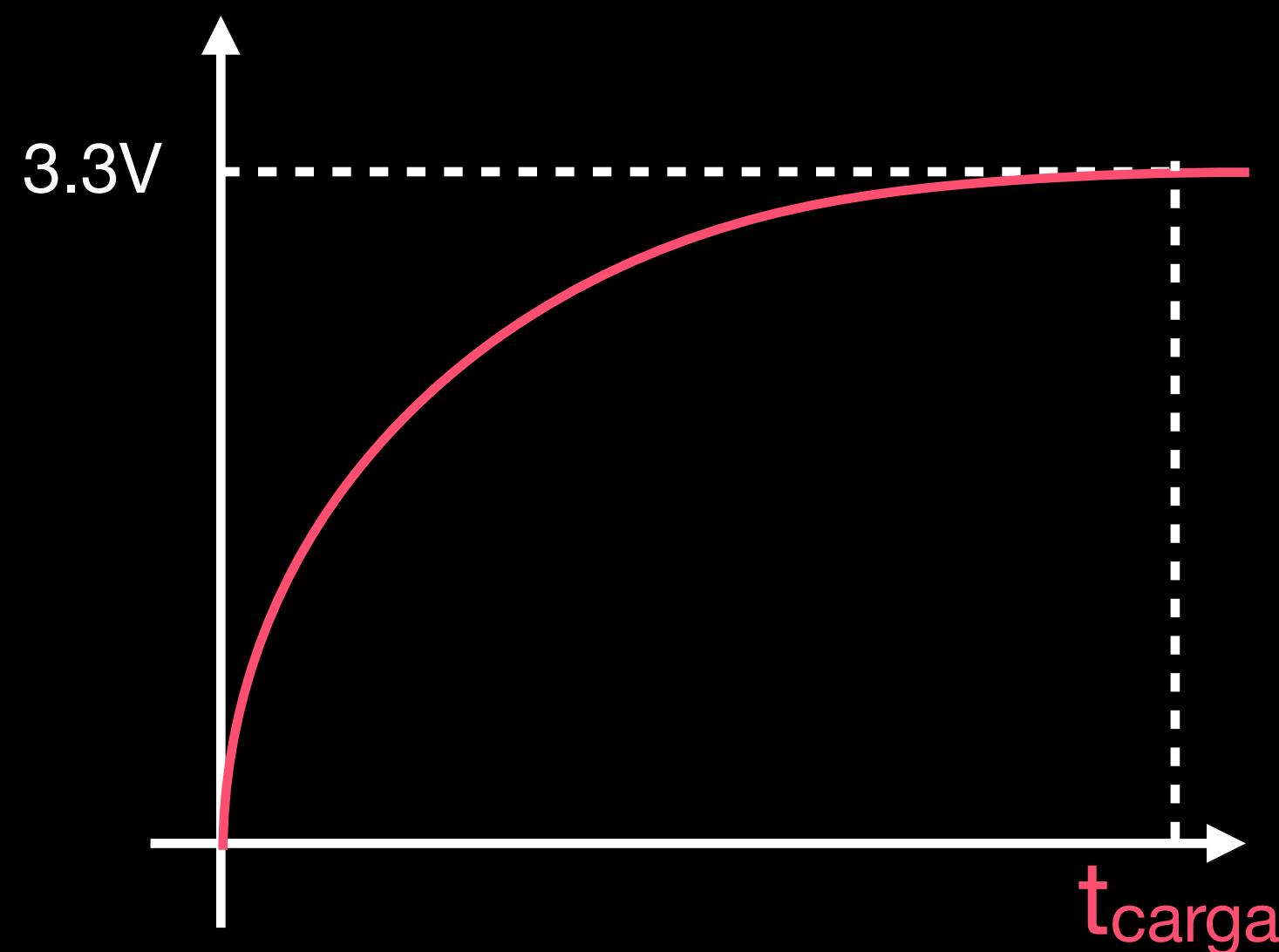
Resistência Dependente de Luz (LDR) ou Fotoresistor



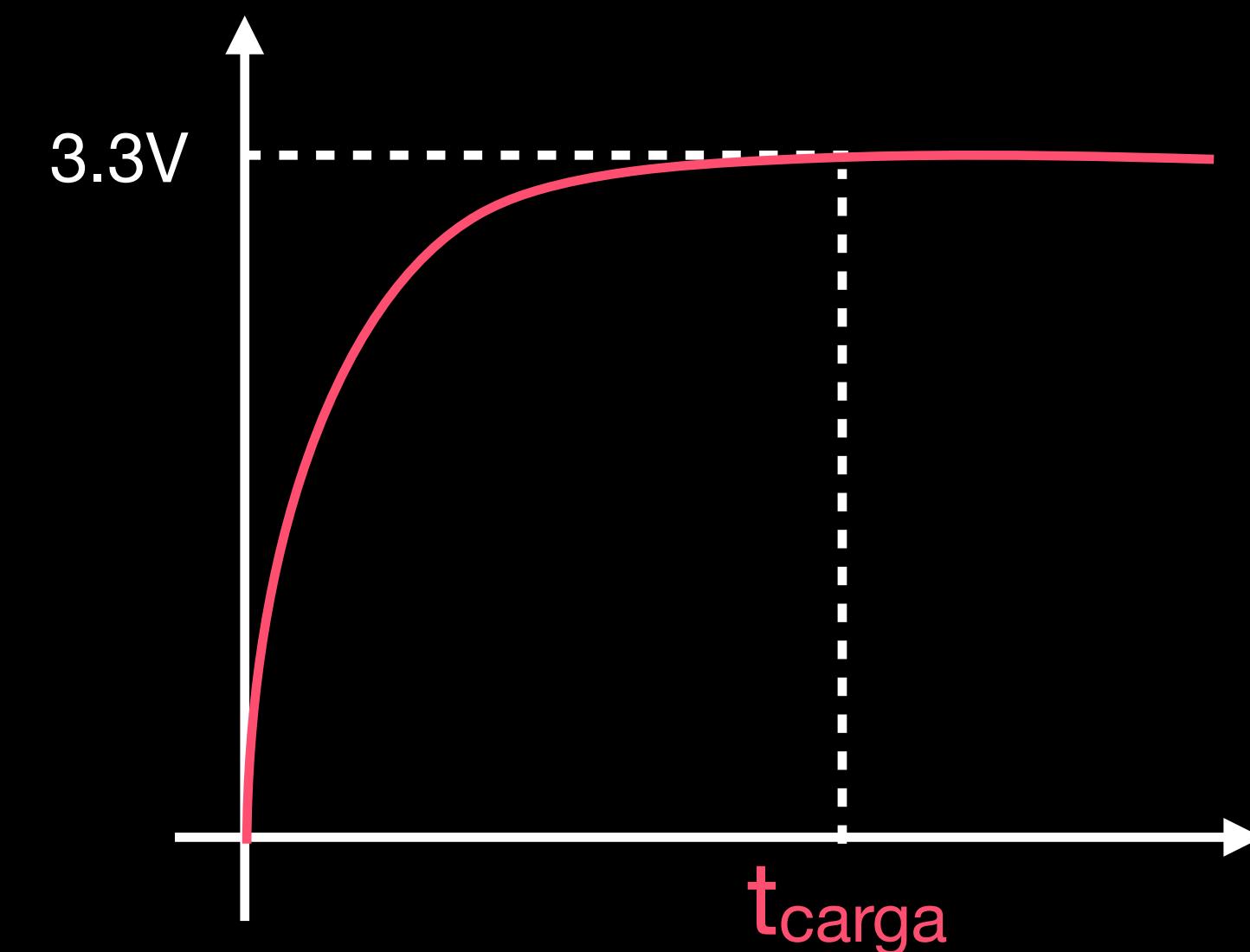
Círculo com Capacitor para Detecção de Luz



Carregamento sem Luz (R Maior)

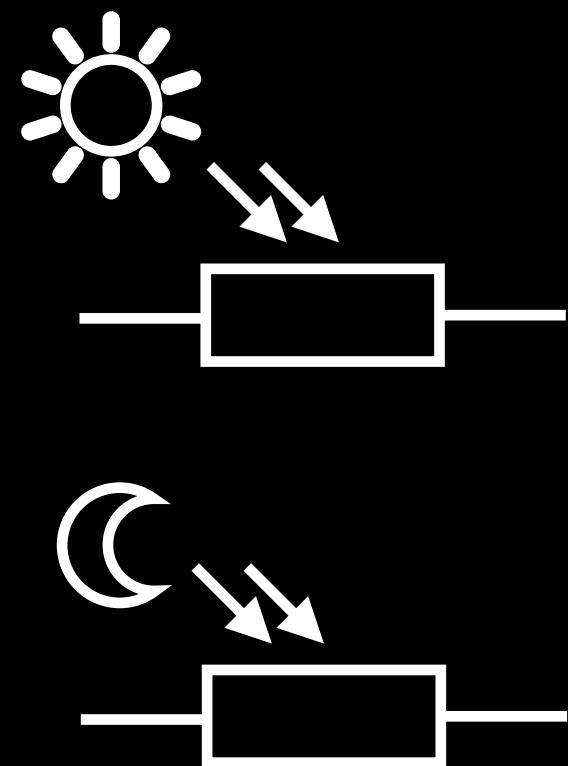


Carregamento com Luz (R Menor)



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

```

sensor_de_luz = LightSensor(8)

while True:
    print(sensor_de_luz.value)
    sleep(1)

```

**alteração  
imediata** →

0.857614

0.854363

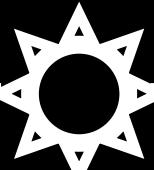
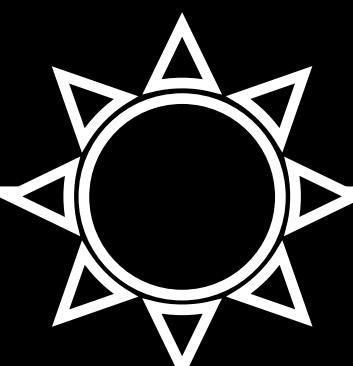
0.862208

0.347932

0.349850

0.345466

0.342849



```

sensor = LightSensor(8, queue_len=20)

while True:
    print(sensor.value)
    sleep(1)

```

**alteração gradual**

0.857614

0.854363

0.862208

0.653456

0.498232

0.345466

0.342849

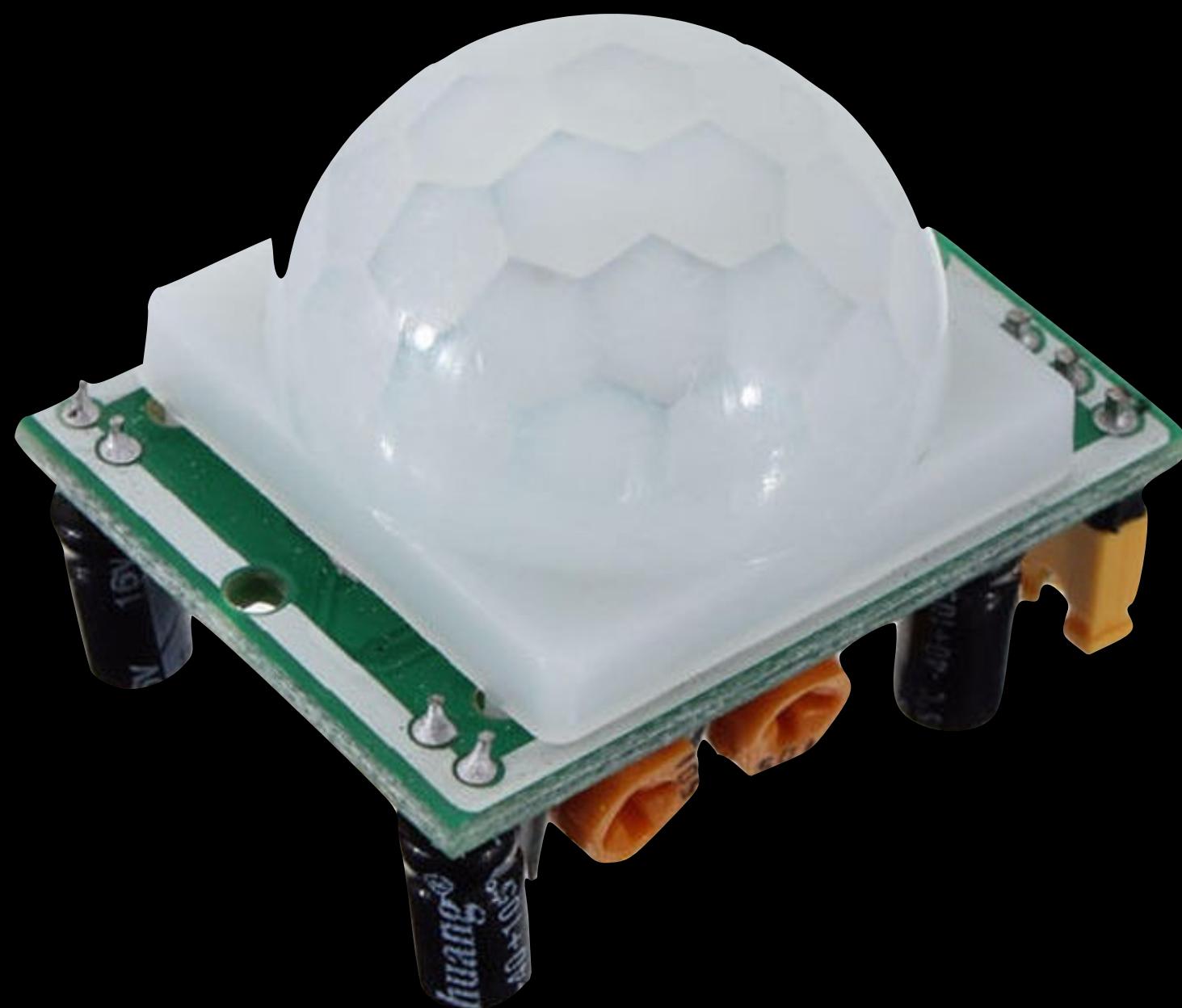


Sensibilidade a Mudanças de um Sensor de Luz

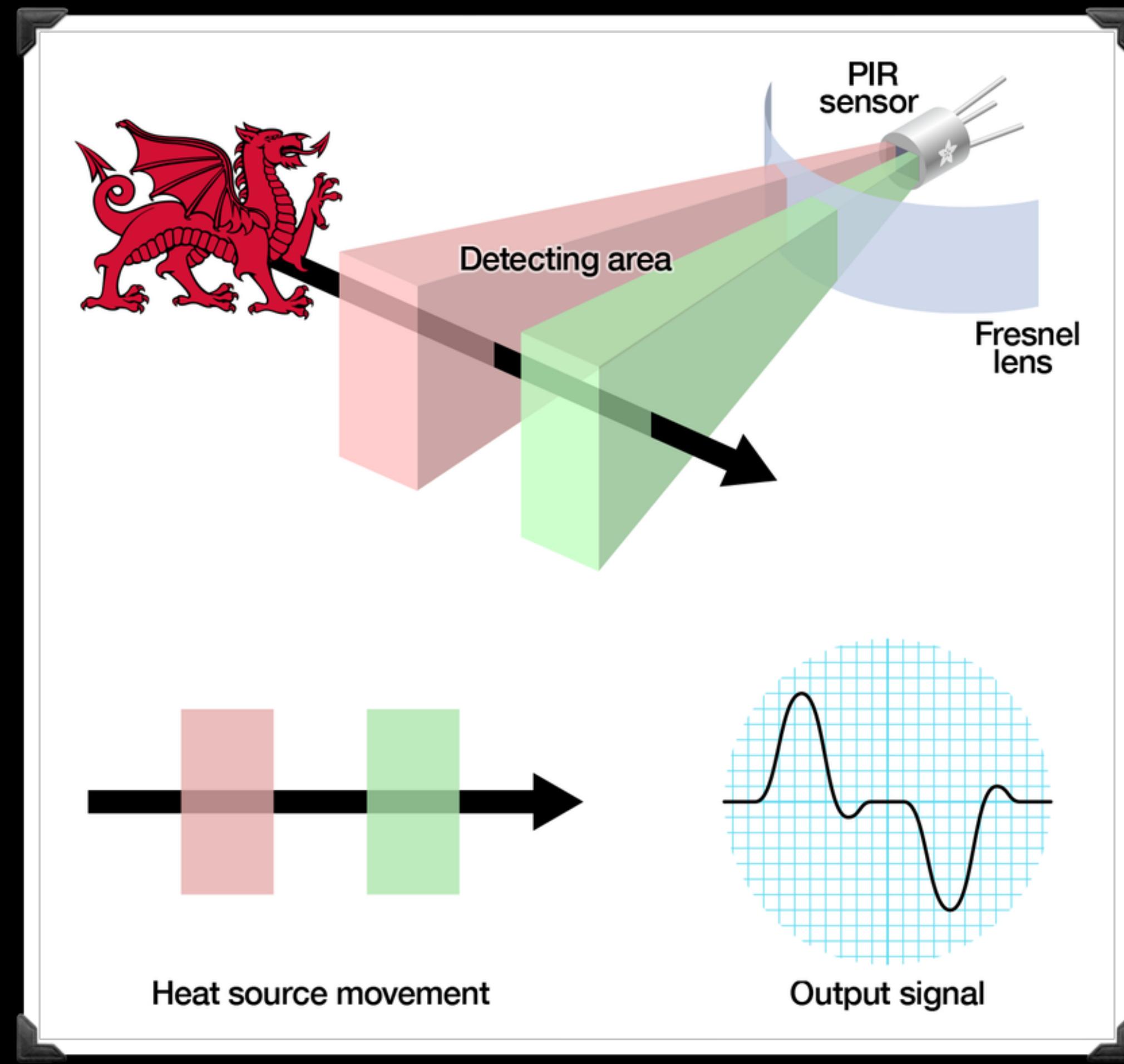
```
>>> from gpiozero import LightSensor  
>>> sensor_de_luz = LightSensor(8)  
>>> sensor_de_luz.threshold  
0.5  
>>> sensor_de_luz.value  
0.56743  
>>> sensor_de_luz.light_detected  
True  
  
>>> sensor_de_luz.threshold = 0.6  
>>> sensor_de_luz.value  
0.55891  
>>> sensor_de_luz.light_detected  
False
```

Limiar para Detecção de Luz

```
>>> from gpiozero import LightSensor  
>>> sensor_de_luz = LightSensor(8)  
>>> sensor_de_luz.threshold = 0.6  
  
>>> 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
```



Sensor de Movimento

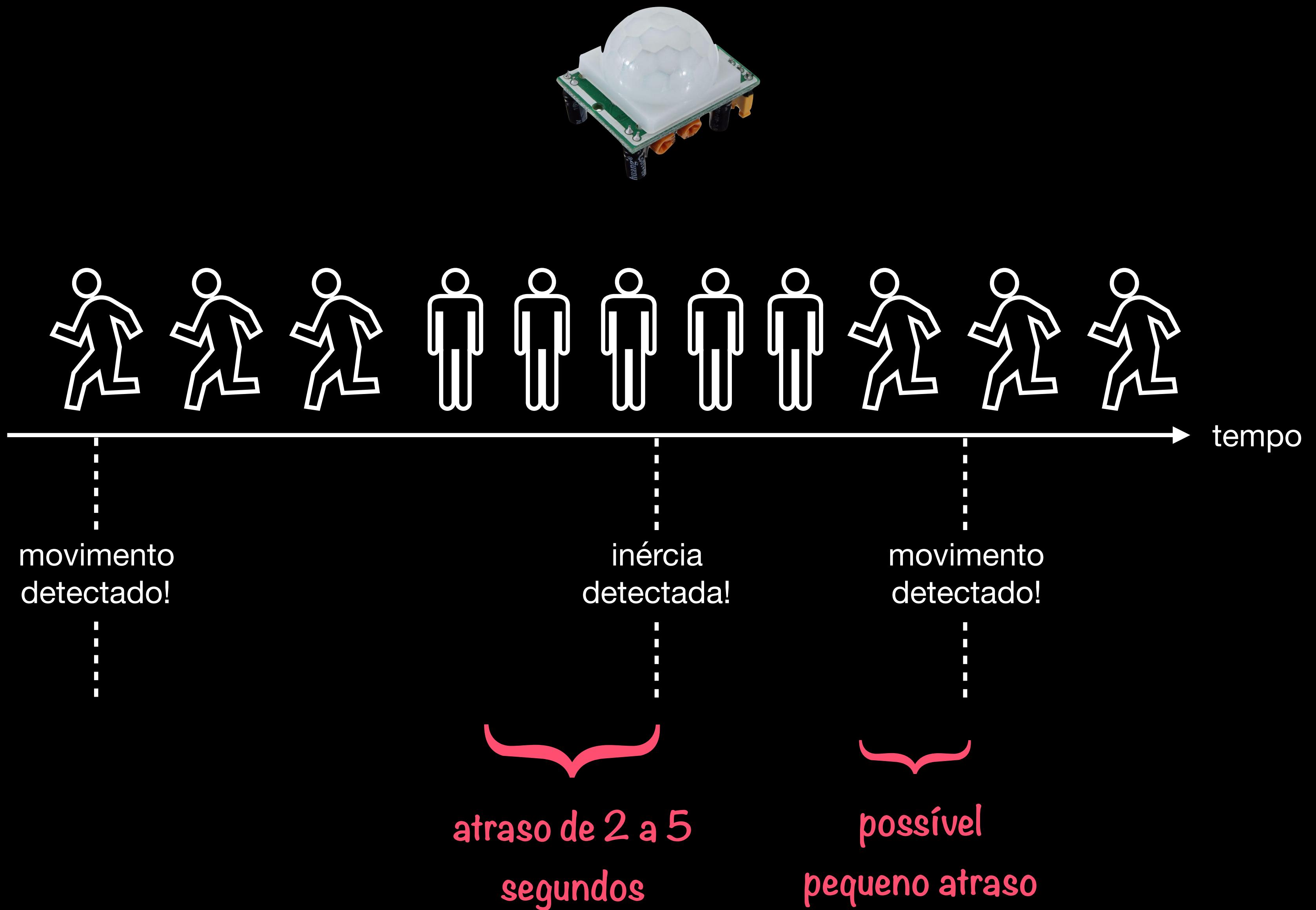


Funcionamento de um Infravermelho Passivo (PIR)

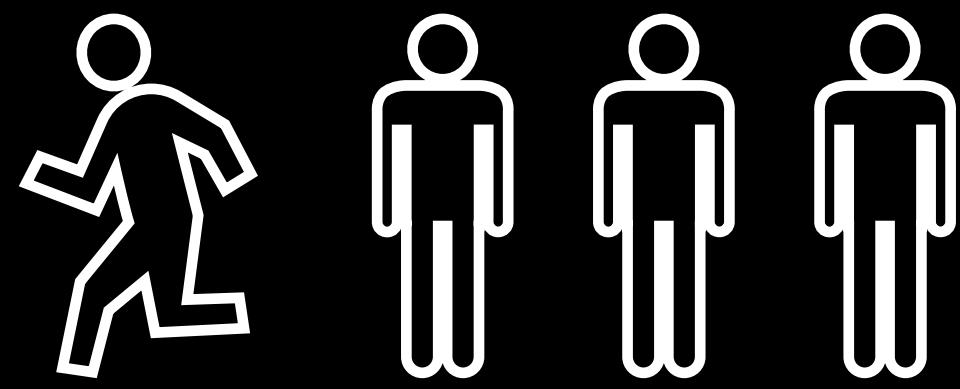
```
>>> from gpiozero import MotionSensor  
>>> sensor_de_movimento = MotionSensor(27)  
>>> sensor.motion_detected  
False  
>>> def movimento_detectado():  
...     print("Mexe a cadeira, hey!")  
...  
>>> sensor_de_movimento.when_motion = movimento_detectado  
>>> def inercia_detectada():  
...     print("Stop! Hammer time!")  
...  
>>> sensor_de_movimento.when_no_motion = inercia_detectada
```



Atraso de Detecção nos Sensores de Movimento



Atraso de Detecção nos Sensores de Movimento



Como faz para  
esperar mais tempo?



:  
inéria  
detectada!

atraso de 2 a 5  
segundos

atraso de mais alguns segundos

ao detectar **inéria**

ao detectar **movimento**



inicie um Timer  
para esperar mais tempo

cancele o Timer

Acréscimo no Tempo de Espera da Inéria

# Software



# Flask

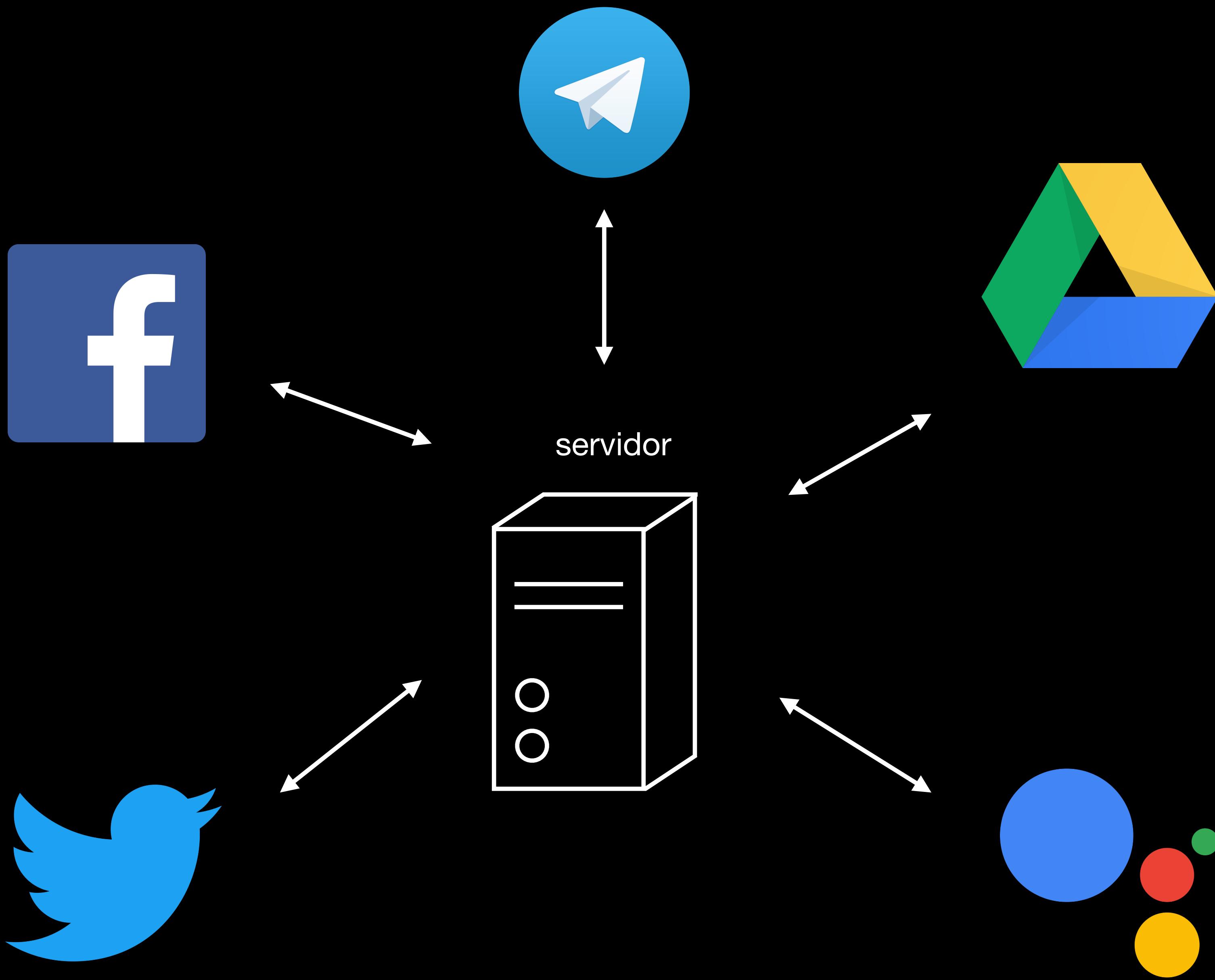
web development,  
one drop at a time



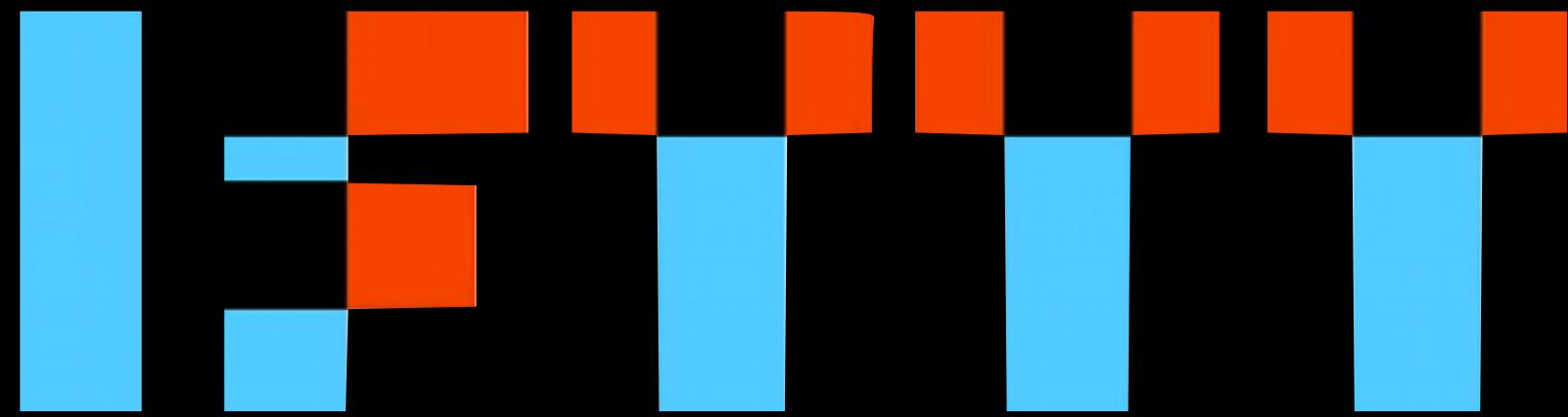
# Requests

*http for humans*

Uso de Softwares dos Projetos Passados

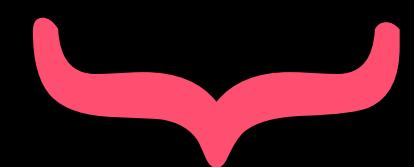


Dificuldade em Programar Várias Comunicações

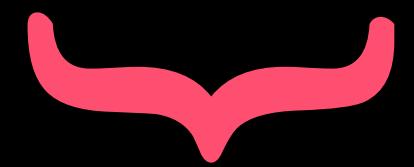


"applet" →

**if this than that**



gatilho



ação

IFTTT: If This Than That

The image shows the homepage of IFTTT (If This Then That) at ifttt.com. The main headline reads "A world that works for you". Below it, a subtext explains: "IFTTT is the free way to get all your apps and devices talking to each other. Not everything on the internet plays nice, so we're on a mission to build a more connected world." On the left, there's a sign-up form with fields for "Enter your email" and "Get started" button, along with "Continue with Google" and "Continue with Facebook" options. To the right, a large graphic illustrates how various devices and services can be interconnected. It features a lightbulb, a smartphone with a dog icon, a tablet, 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 with a download arrow, and a speaker. Arrows show the flow of data between these components, such as from the calendar to the email invite or from the smartwatch to the cloud.

Integração entre Serviços com IFTTT

[ifttt.com/create/if?sid=2](#)

< Back

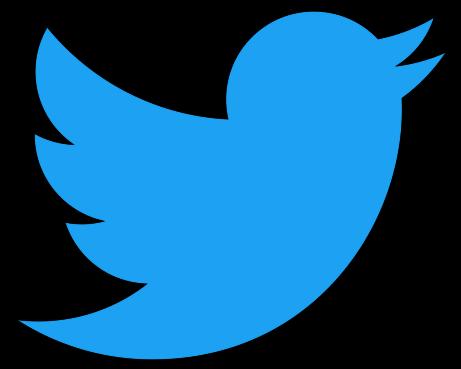
## Choose a service

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



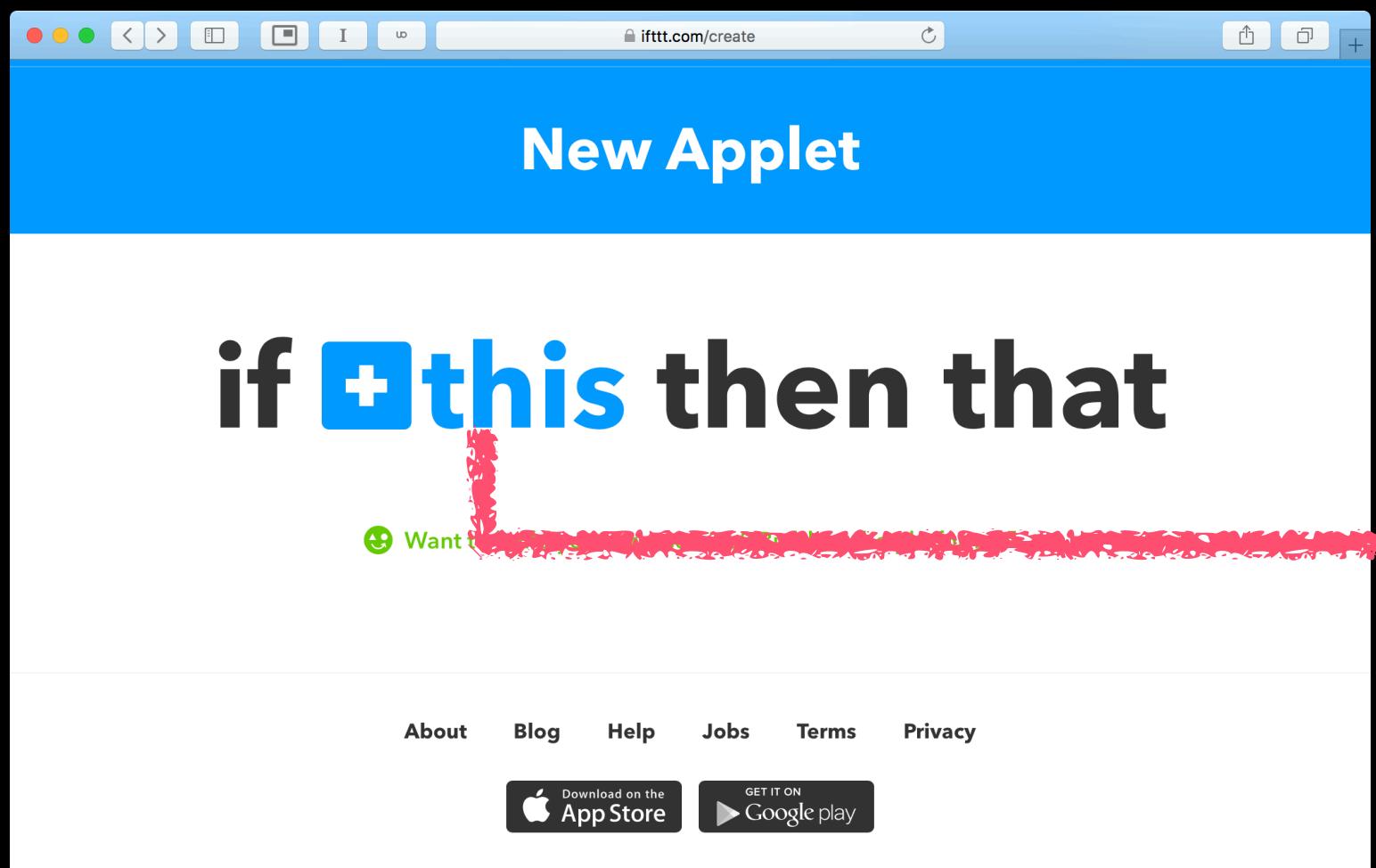
então



se um usuário tuitar

mande-me um SMS

Exemplo de Integração entre Twitter e SMS

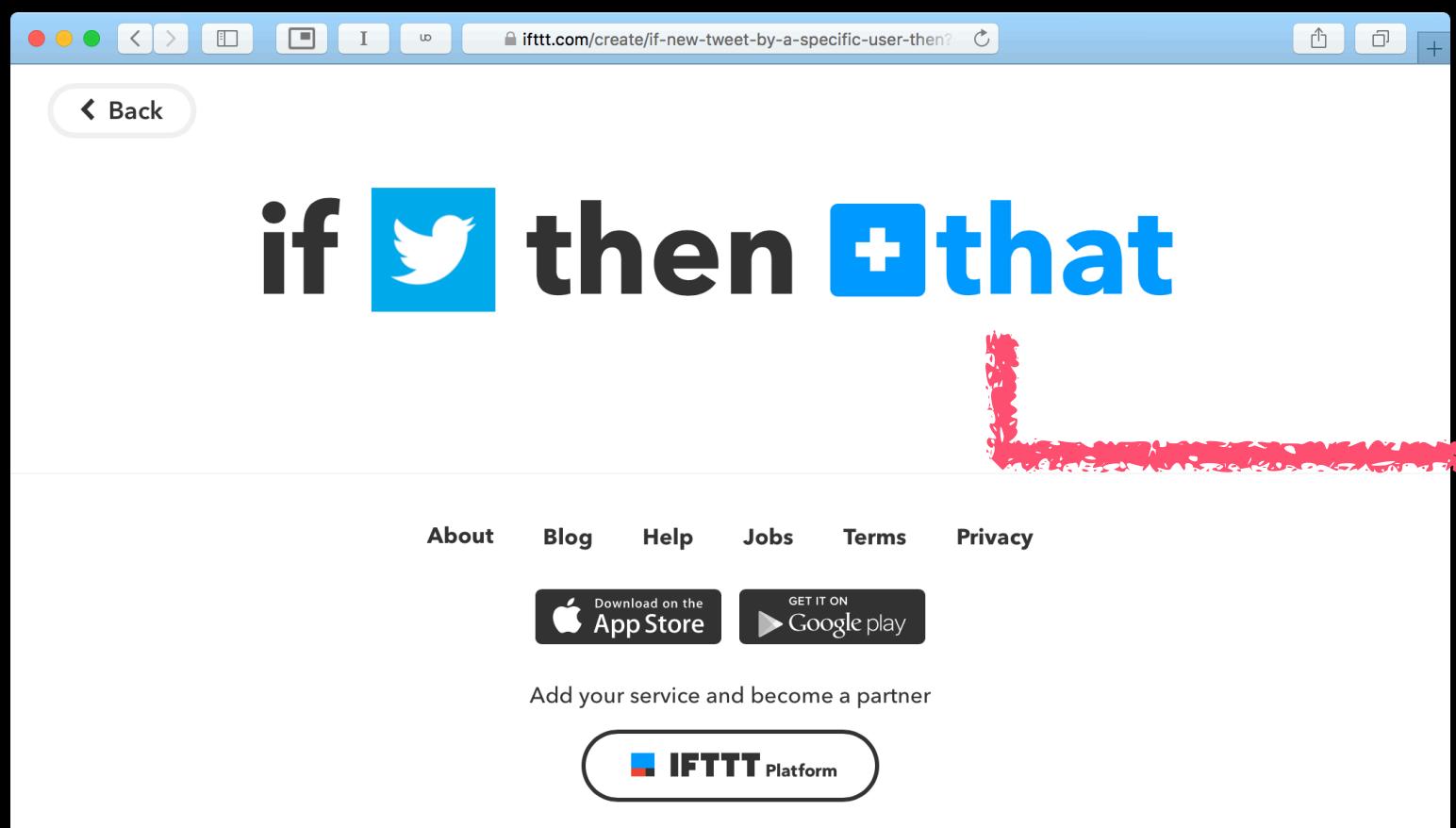


**New tweet by a specific user**

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

Username to watch

**Create trigger**



**Send me an SMS**

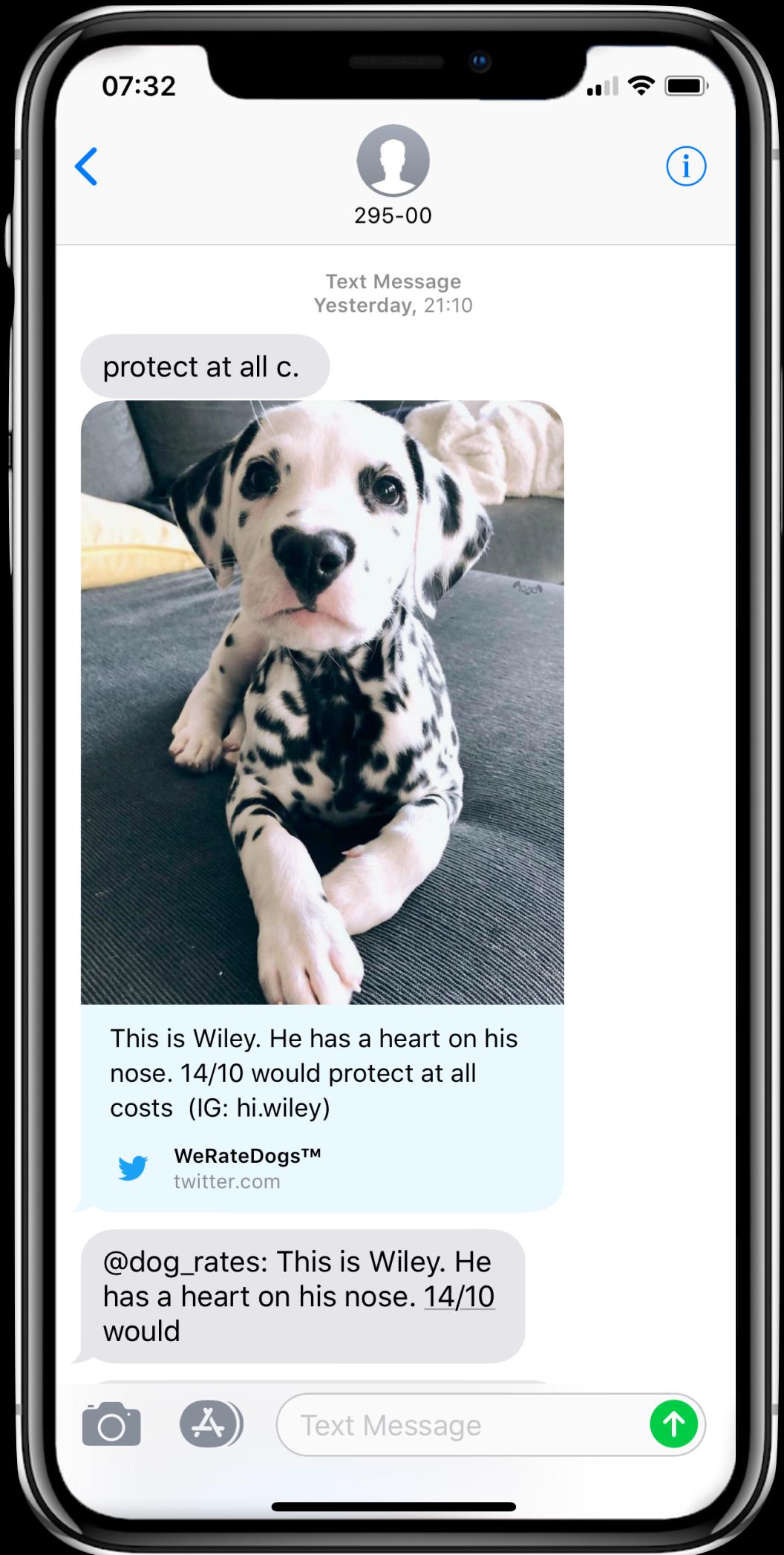
This Action will send an SMS to your mobile phone.

Message

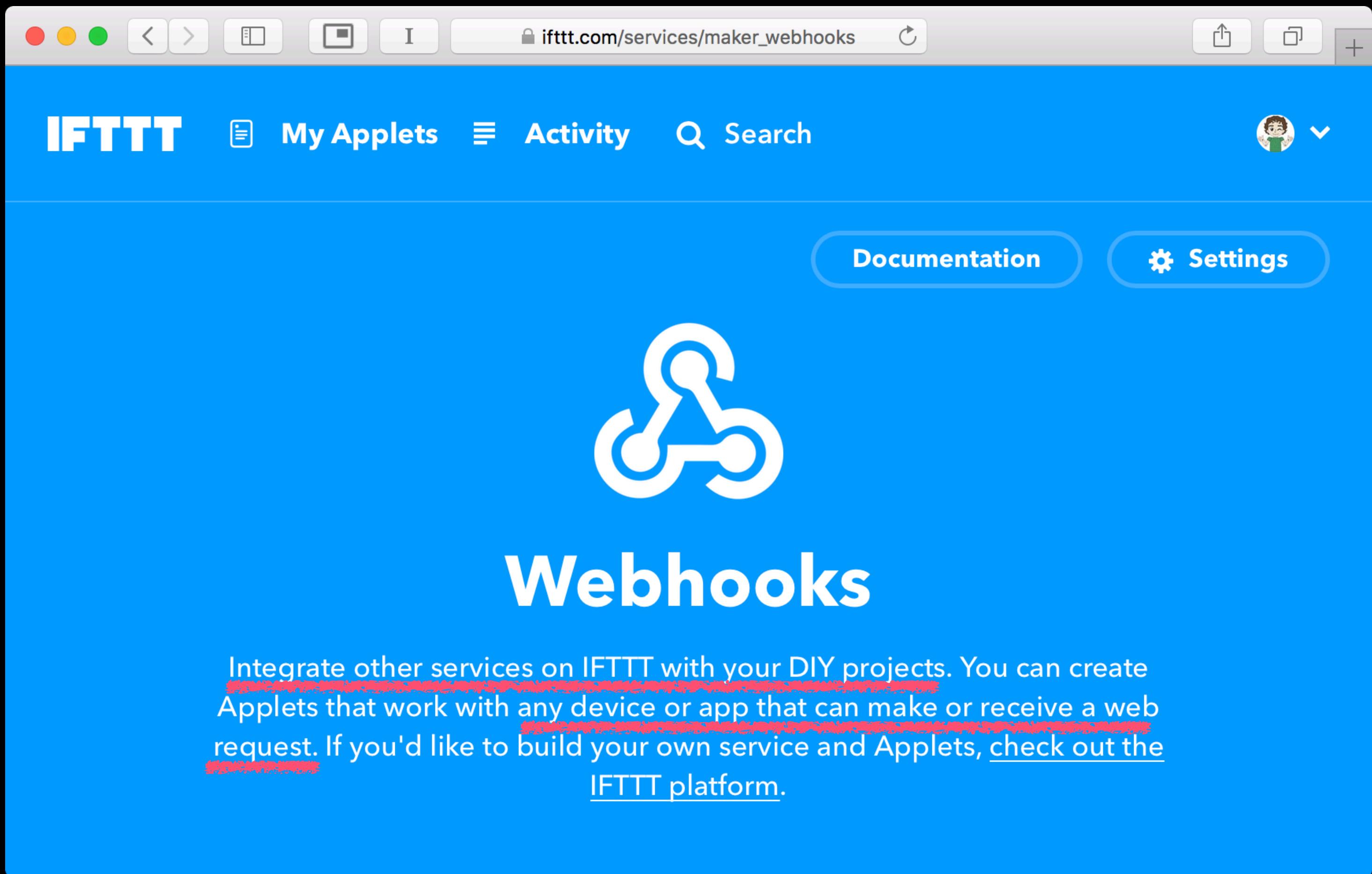
**Add ingredient**

**Create action**

Configuração do "This" e do "That"

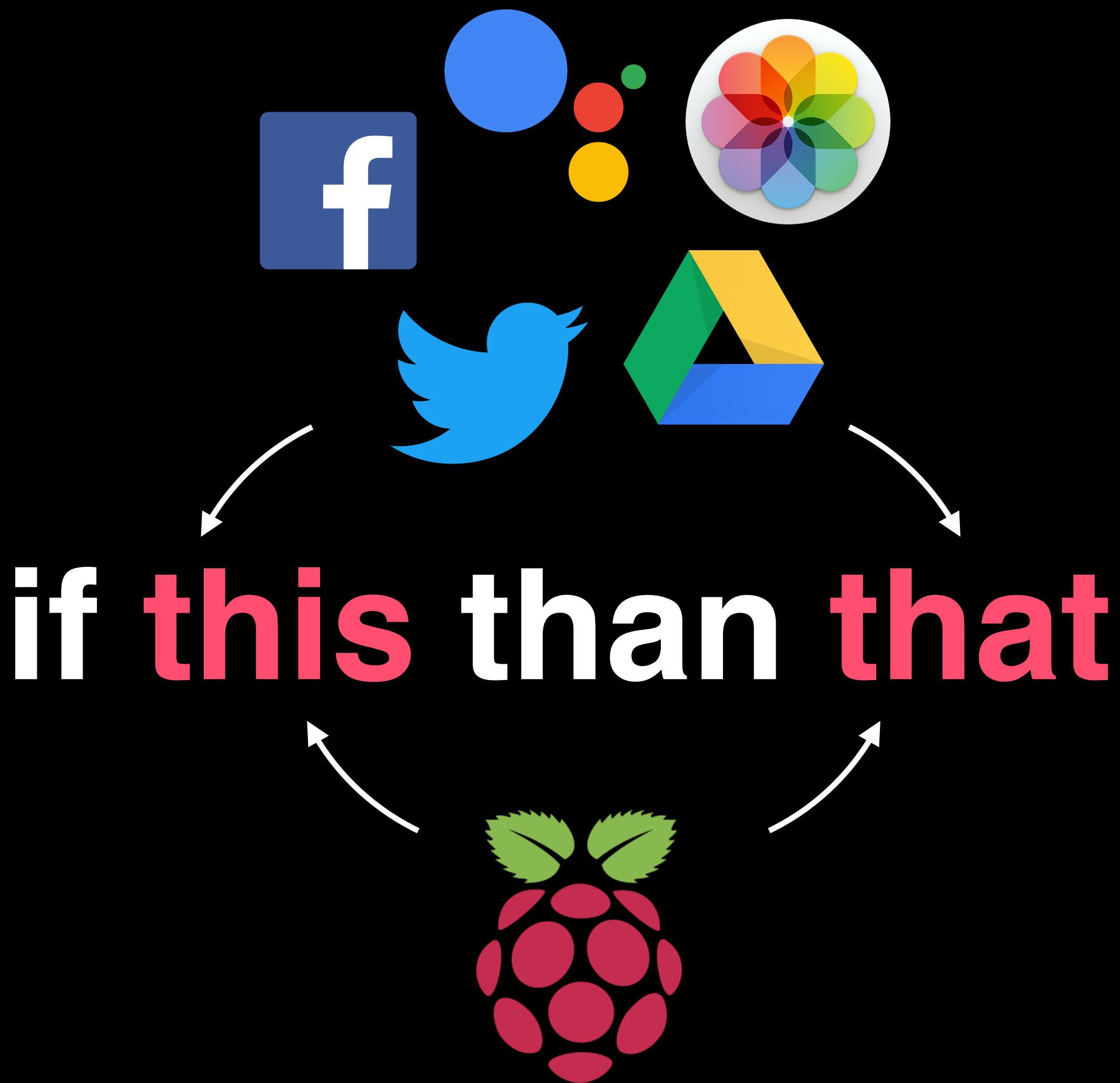


SMS com Tweets de um Perfil

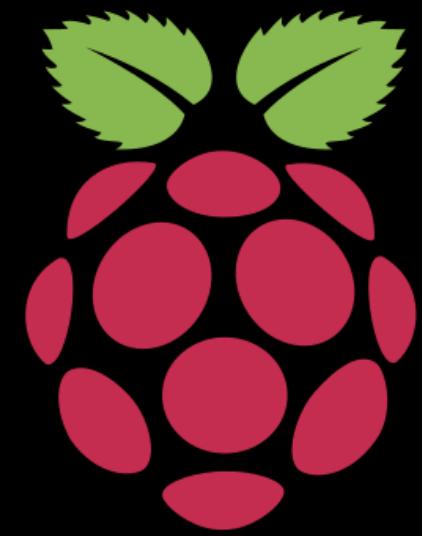
A screenshot of a web browser window showing the IFTTT Webhooks page. The browser has a light gray header with standard OS X-style buttons and a URL bar showing "ifttt.com/services/maker\_webhooks". Below the header is a blue navigation bar with the IFTTT logo, "My Applets", "Activity", "Search", and a user profile icon. On the right side of the blue bar are three buttons: "Documentation", "Settings" (which is highlighted with a blue border), and another unlabeled button. The main content area has a white background. In the center is a large white icon consisting of three interconnected circles. Below the icon is the word "Webhooks" in a large, bold, white sans-serif font. Underneath "Webhooks" is a paragraph of text in a smaller white sans-serif font. The text reads: "Integrate other services on IFTTT with your DIY projects. You can create Applets that work with any device or app that can make or receive a web request. If you'd like to build your own service and Applets, check out the IFTTT platform." The word "check out the" and "IFTTT platform." are underlined and have a red underline underneath them.

Integrate other services on IFTTT with your DIY projects. You can create Applets that work with any device or app that can make or receive a web request. If you'd like to build your own service and Applets, check out the IFTTT platform.

Opção Webhook do IFTTT

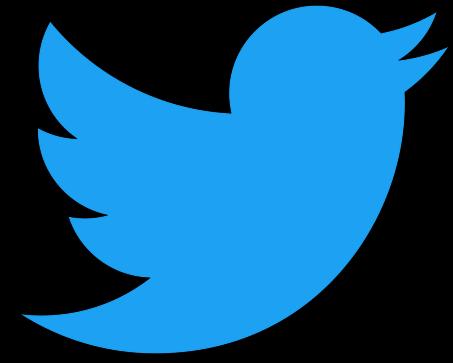


Integração entre Raspberry Pi e Serviços



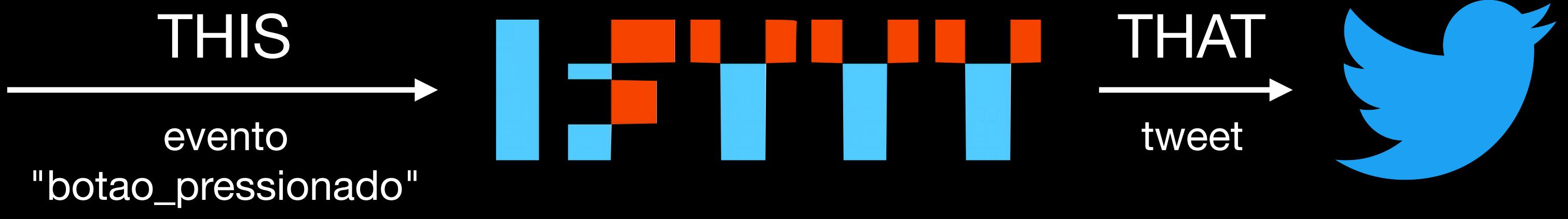
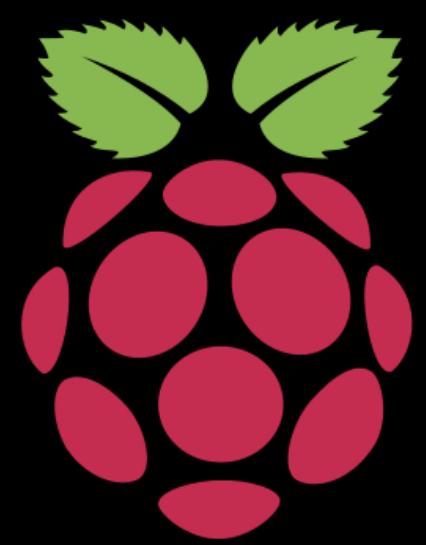
se botão for  
pressionado

então

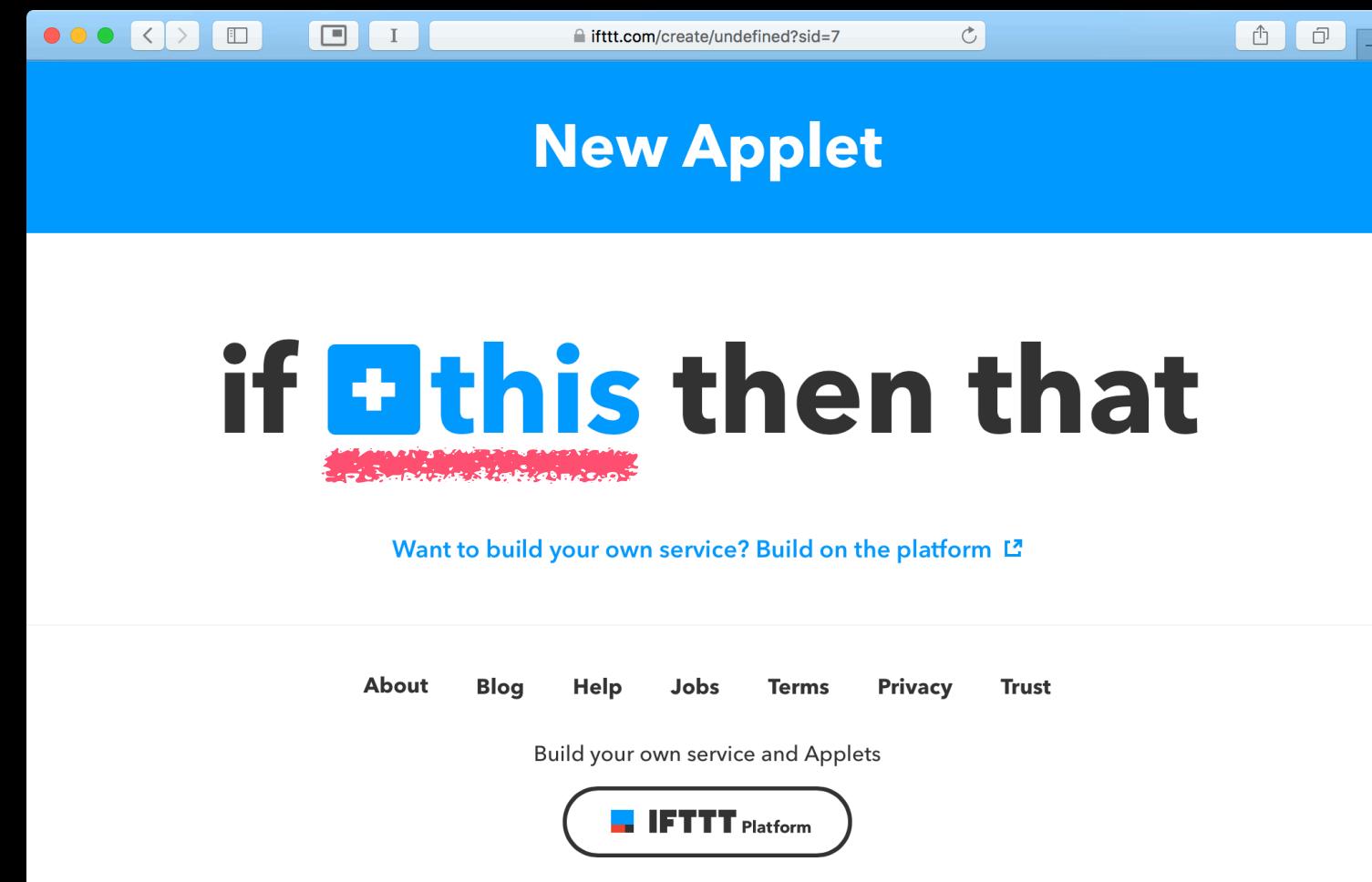


poste um  
tweet

Exemplo de Integração entre Raspberry Pi e Twitter



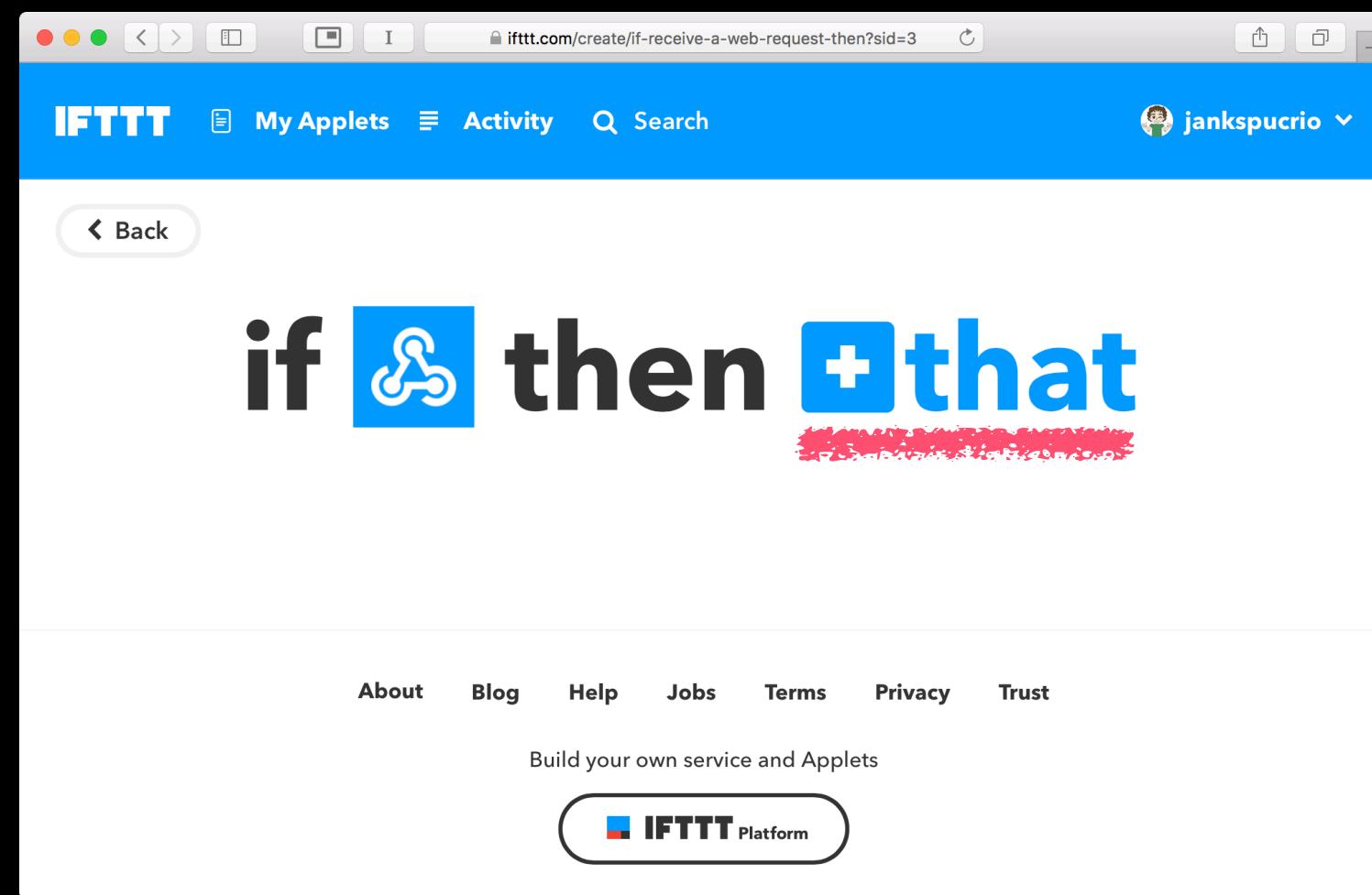
Envio de Eventos ao IFTTT



A screenshot of the IFTTT website showing the "Choose a service" step. The search bar contains "webhooks". A blue box highlights the "Webhooks" service icon, which features a white gear-like shape. The page indicates "Step 1 of 6". The navigation bar at the bottom includes links for About, Blog, Help, Jobs, Terms, and Privacy, along with download links for the App Store and Google Play.

A screenshot of the IFTTT website showing the "Receive a web request" configuration step. It explains that this trigger fires every time the Maker service receives a web request. An input field shows the event name "botao\_pressionado". A note below states that the name can be like "button\_pressed" or "front\_door\_opened". A large "Create trigger" button is at the bottom right. To the right of the window, the text "nome do \"evento\"".

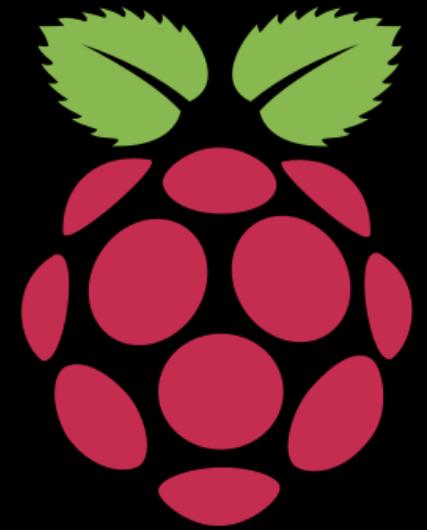
Configuração do "This" como Serviço Webhook



This screenshot shows the "Choose action" step of a workflow creation process. The title "Choose action" is at the top, followed by "Step 4 of 6". On the left, there are two blue rounded rectangular boxes: one for "Post a tweet" and another for "Post a tweet with image". Both boxes contain a brief description of the action and a note about adhering to Twitter's Rules and Terms of Service. To the right, a large white area displays the configuration for the "Post a tweet" action. It shows a text input field containing the Portuguese phrase "O valor medido pelo sensor de luz é {{Value1}}." Below this, there are several input fields for parameters: "EventName" (with "Value1" entered), "Value2", "Value3", and "OccurredAt". There is also a "Receive a web request" button with a blue icon.

This screenshot shows the configuration for the "Post a tweet" action. The title "Post a tweet" is at the top. 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." Below this, there is a "Tweet text" input field containing the text "O valor medido pelo sensor de luz é {{Value1}}.". To the right, there is a "Add ingredient" button. Below the tweet text, there are input fields for "EventName" (containing "Value1"), "Value2", "Value3", and "OccurredAt".

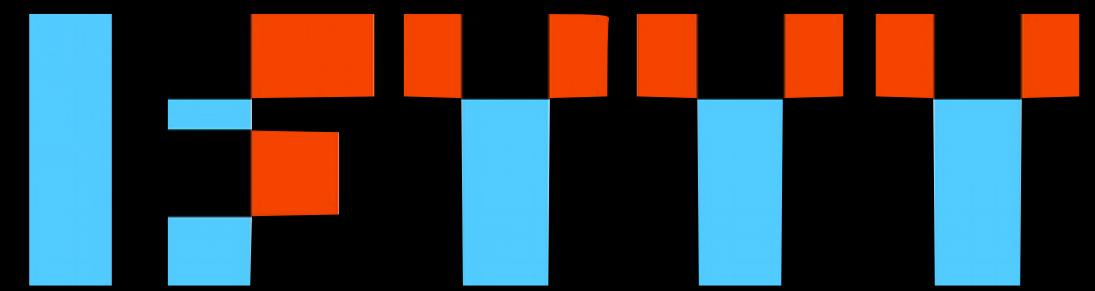
Configuração do "That" como Tweet com Parâmetros



post com dados



post com dados



Envio de Requisições ao IFTTT

[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 section is titled "Webhooks settings" with a link to "View activity log". Below this, the "Account Info" section shows the user is connected as "wallysalami" and provides a URL: [https://maker.ifttt.com/use/h4JNuiynS3Gxp\\_hKhYEhczz](https://maker.ifttt.com/use/h4JNuiynS3Gxp_hKhYEhczz). The URL is partially redacted with a red marker. The status is listed as "active". A red annotation in Spanish, "anote esta chave!", is overlaid on the redacted URL area. A button labeled "Edit connection" is visible at the bottom left.

My Applets > Webhooks

Webhooks settings

[View activity log](#)

---

Account Info

Connected as: wallysalami

URL: [https://maker.ifttt.com/use/h4JNuiynS3Gxp\\_hKhYEhczz](https://maker.ifttt.com/use/h4JNuiynS3Gxp_hKhYEhczz)

Status: active

[Edit connection](#)

anote esta chave!

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}  
...     resultado = post(endereco, json=dados)  
...     print(resultado.text)  
...  
>>> botao = Button(11)  
>>> botao.when_pressed = botao_pressionado
```

A screenshot of a Twitter post from the user @eng1419. The tweet content is: "O valor medido pelo sensor de luz é 0.7632026672363281." The post was made at 09:33 - 9 de abr de 2018. The background of the page features a large image of a Raspberry Pi logo.

Página Inicial    Moments    Notificações (11)    Mensagens    Buscar no Twitter    Tweetar    X

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

Tweet text

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

Add ingredient

EventName

Value1

Value2

Value3

Receive a web request

dados = {"value1": sensor\_de\_luz.value}

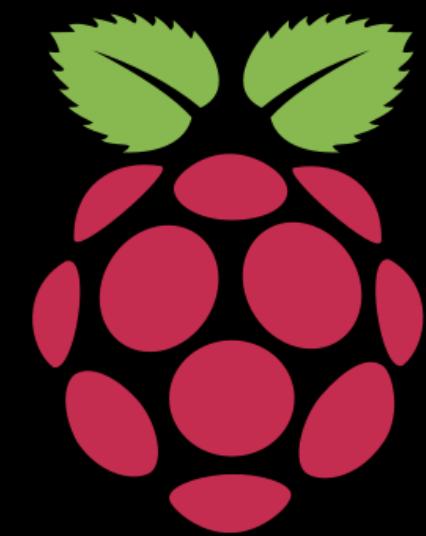
"v" minúsculo



Atenção à Variação da Ortografia de Value1



então



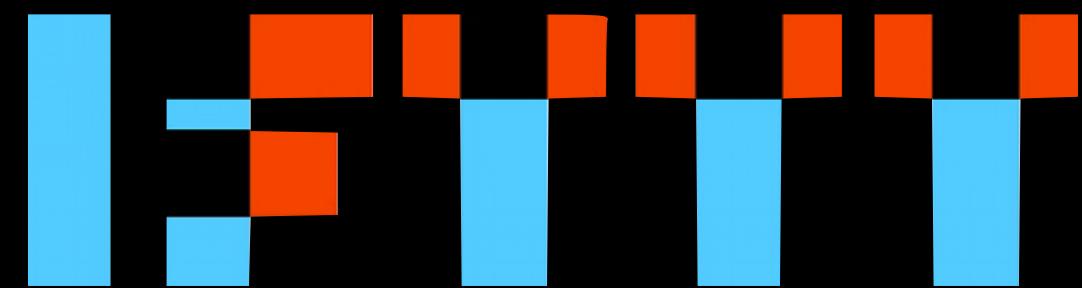
se houver  
previsão do tempo

mostre a previsão  
no LCD

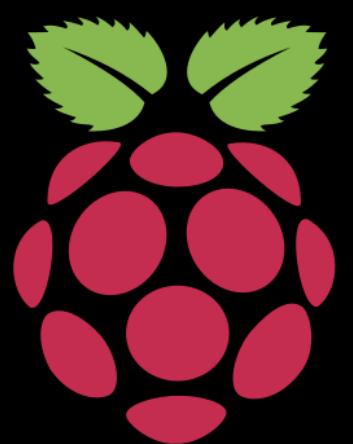
Exemplo de Integração entre Weather Underground e Raspberry Pi



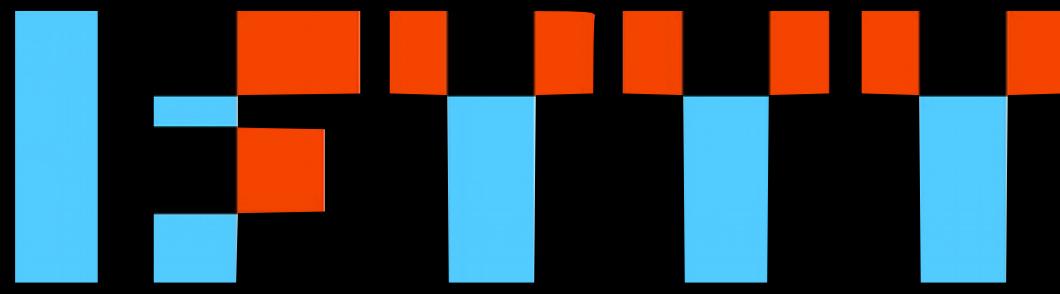
THIS  
mudança  
no tempo



THAT  
acessa endereço

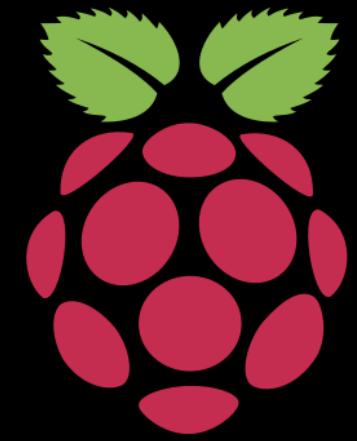


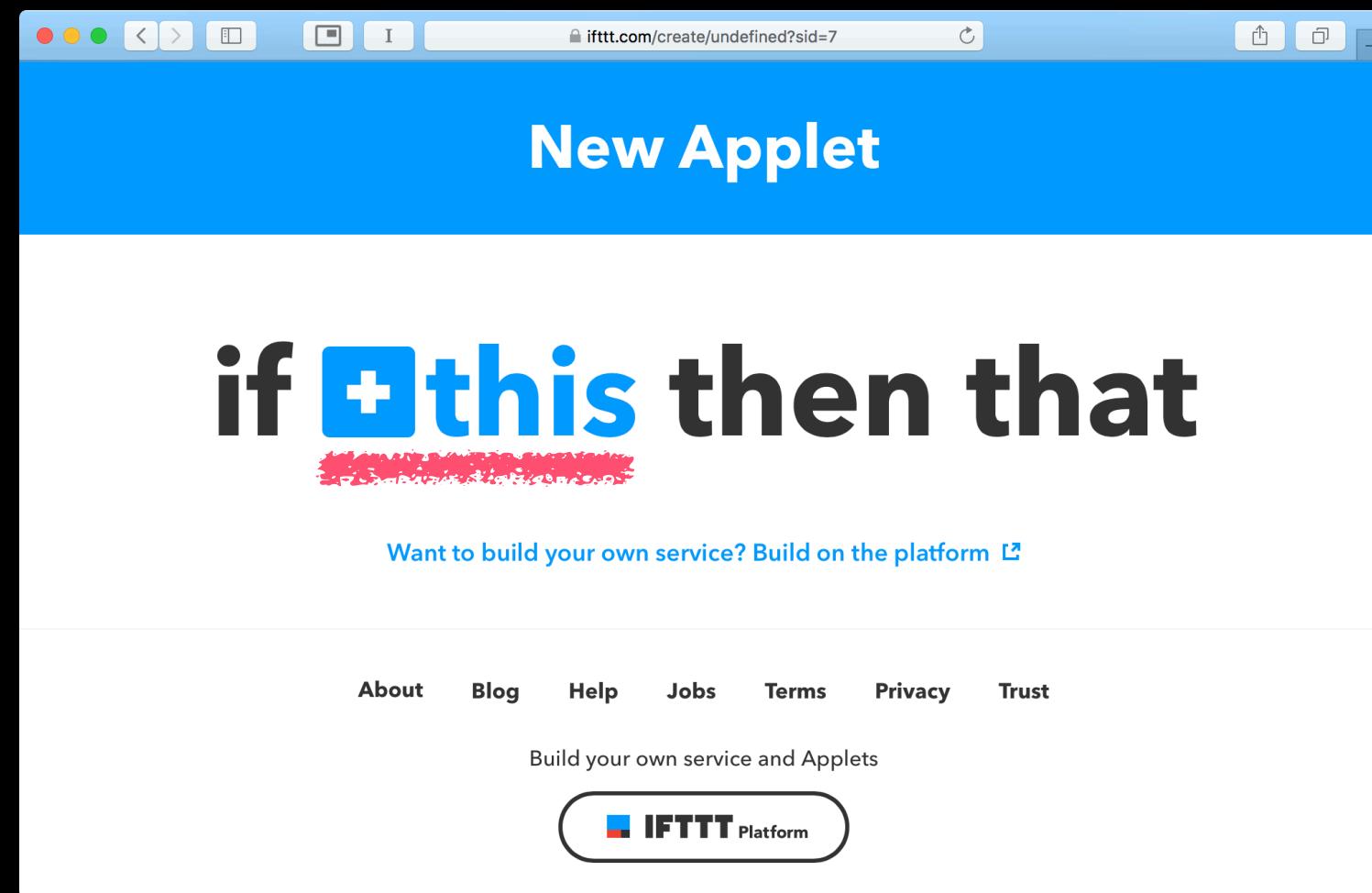
Recebimento de Eventos do IFTTT



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

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





This screenshot shows the 'Choose trigger' step of an applet creation. The title 'Choose trigger' is at the top, followed by 'Step 2 of 6'. There are two options: 'Today's weather report' and 'Tomorrow's weather report'. Both descriptions mention retrieving weather reports at specified times, noting that pollen count is available only in the USA. The 'Today's weather report' option is selected.

This screenshot shows the configuration of the 'Today's weather report' trigger. It displays the title 'Today's weather report' and a note about retrieving today's current weather report. It includes dropdown menus for 'Time of day' set to '08 AM' and '00 Minutes', and a large 'Create trigger' button.

Configuração do Weather Underground como "This"

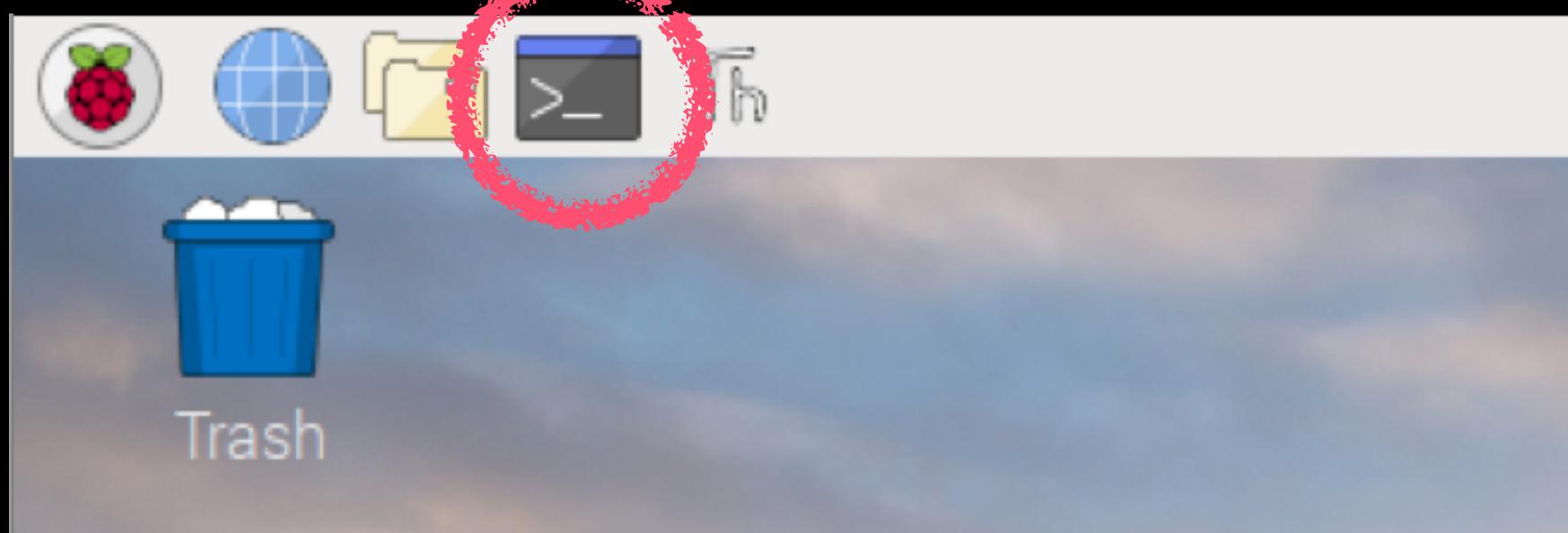
The screenshot shows the IFTTT web interface for creating a new applet. At the top, the URL is ifttt.com/create/if-todays-weather-report-then?sid=3. The main header reads "if [WU logo] then +that". Below the header are navigation links for "About", "Blog", "Help", "Jobs", "Terms", "Privacy", and "Trust". A sub-header says "Build your own service and Applets". At the bottom is a button labeled "IFTTT Platform".

This screenshot shows the "Choose action service" step of the IFTTT applet creation process, which is Step 3 of 6. The URL is ifttt.com/create/if-todays-weather-report-then?sid=4. The title "Choose action service" is displayed above a search bar containing the query "webhooks". Below the search bar is a blue button with a white icon and the text "Webhooks". The footer includes links for "About", "Blog", "Help", "Jobs", "Terms", "Privacy", and "Trust".

This screenshot shows the configuration for a "Make a web request" action. The title is "Make a web request". It states: "This action will make a web request to a publicly accessible URL. NOTE: Requests may be rate limited." A cartoon character is pointing towards the URL field. The URL is set to `https://fd288c990i.ngrok.io/tempo/{{CurrentCondition}}`. Below the URL is a note: "Surround any text with '<>>' to escape the content". There is a "Add ingredient" button. The method is set to "GET". A note at the bottom says: "The method of the request e.g. GET, POST, DELETE".

ATENÇÃO: não deixe  
espaços em branco no  
endereço!

Configuração do Webhook como "That"



```
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

```
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=5)

    return "A previsão do tempo é: " + previsao

app.run(port=5000)
```

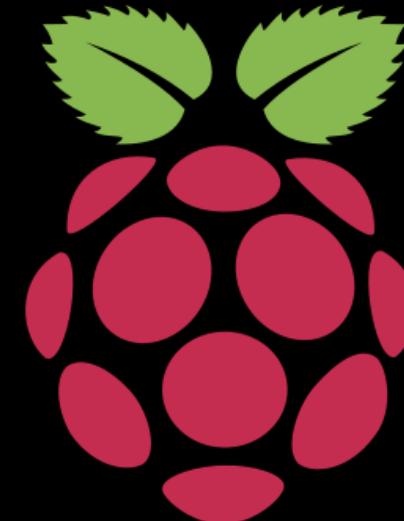
Configuração do Servidor com Página para Previsão do Tempo



Previsão de Tempo no LCD

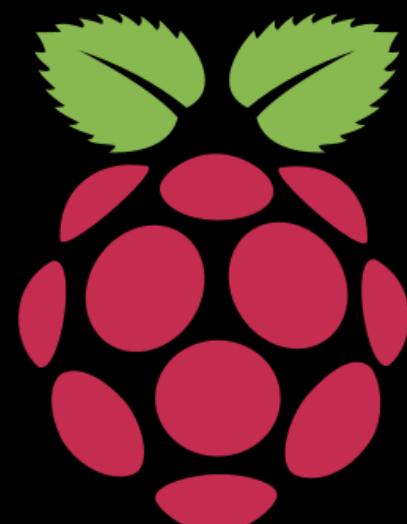
# Uber

então



se viagem para  
casa for concluída

ligue a luz e a TV  
e salve dados no banco



então



se eu ando chegando  
muito tarde em casa

agende um horário  
para meditar após almoço

# +this

## Ride completed

This trigger fires when your Uber ride has ended.

Trigger when pickup location is

Anywhere



Be sure to set "Home" and "Work" addresses in the Uber app

and drop off location is

Home



Be sure to set "Home" and "Work" addresses in the Uber app

Create trigger

# +that

## 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/chequei\\_em\\_casa/](https://fd288c990i.ngrok.io/chequei_em_casa/) VehicleLicensePlate

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

Add ingredient

Method

GET



The method of the request e.g. GET, POST, DELETE

# +this

# +that

## 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

preciso\_descansar

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

Create trigger

## Create a detailed event

This action will create a detailed event in your Google Calendar.

Which calendar?

Teste



Start time

tomorrow at 13PM

Ex. 10AM, Next Monday at 3PM.

Add ingredient

End time

tomorrow at 14PM

Add ingredient

Segunda Parte: Integração com Google Calendar

```
from flask import Flask

app = Flask(__name__)

@app.route("/cheguei_em_casa/<string:placa_do_carro>")
def querida_cheguei(placa_do_carro):
    ligar_aparelhos()
    registrar_chegada(placa_do_carro)

    if trabalhando_muito():
        providenciar_descanso()

    return "Cheguei em casa!"

app.run(port=5000)
```



Agora temos que implementar essas funções...

```
from gpiozero import LED
from py_irsend.irsend import send_once

luz = LED(21)

def ligar_aparelhos():
    luz.on()
    send_once("TV", ["KEY_POWER"])
    sleep(6) # espera a TV ligar
    send_once("TV", ["KEY_5", "KEY_4", "KEY_0"])
```

Função para Ligar Luz e TV

```
from pymongo import MongoClient
from datetime import datetime

cliente = MongoClient("localhost", 27017)
banco = cliente["casa"]
colecao = banco["viagens"]

def registrar_chegada(placa_do_carro):
    agora = datetime.now()

    dados = {"chegada": agora, "placa": placa_do_carro}
    if agora.hour > 9 and agora.hour < 20:
        dados["tarde"] = False
    else:
        dados["tarde"] = True # chegou muito tarde

    colecao.insert(dados)
```

Função para Salvar Dados da Viagem no Banco de Dados

```
from pymongo import MongoClient, DESCENDING
from datetime import datetime

cliente = MongoClient("localhost", 27017)
banco = cliente["casa"]
colecao = banco["viagens"]

def trabalhando_muito():
    agora = datetime.now()
    ha_seis_dias = agora - timedelta(days=6)

    busca = {"tarde": True, "chegada": {"$gt": ha_seis_dias}}
    viagens_tardias = list( colecao.find(busca) )
    total_de_viagens_tardias = len(viagens_tardias)

    if total_de_viagens_tardias > 4:
        return True
    else:
        return False
```

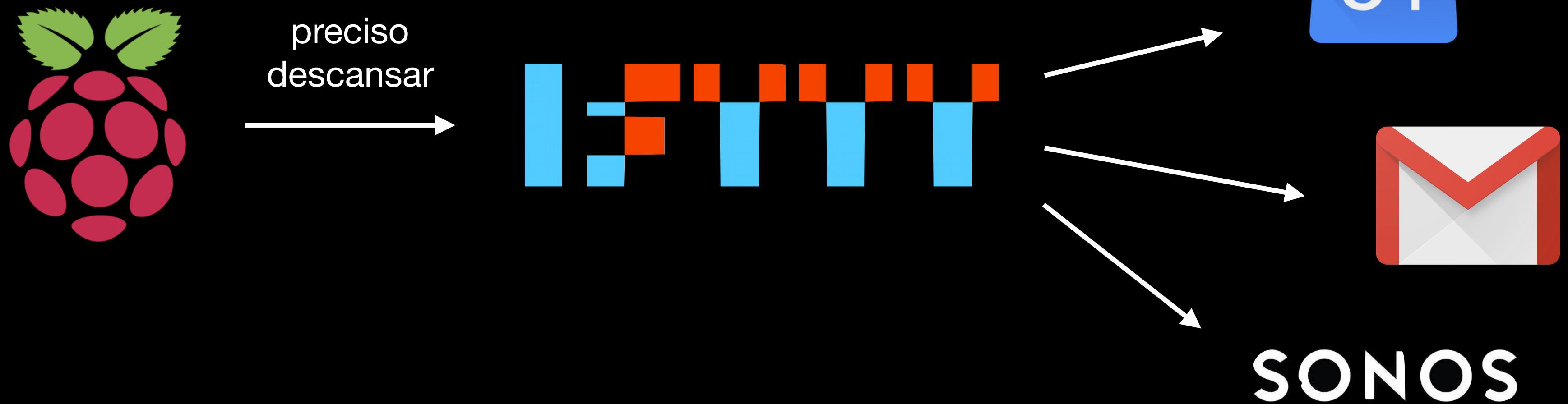
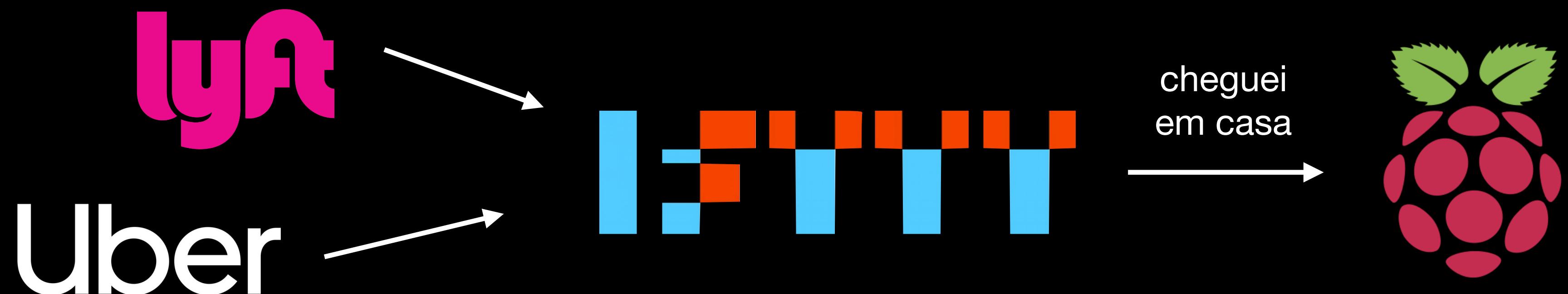
Função para Buscar Dados sobre as Viagens

```
from requests import post

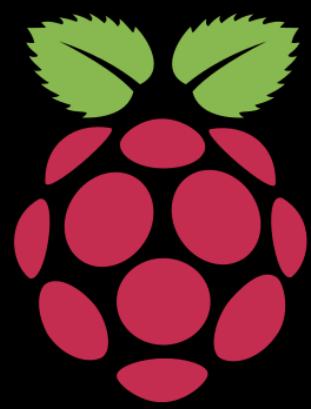
chave = "COLOQUE SUA CHAVE AQUI"
evento = "preciso_descansar"
endereco = "https://maker.ifttt.com/trigger/" + evento
+ "/with/key/" + chave

def providenciar_descanso():
    dados = {}
    resultado = post(endereco, json=dados)
```

Função para Enviar Pedido ao IFTTT



Novas Integrações Aproveitando o Mesmo Código

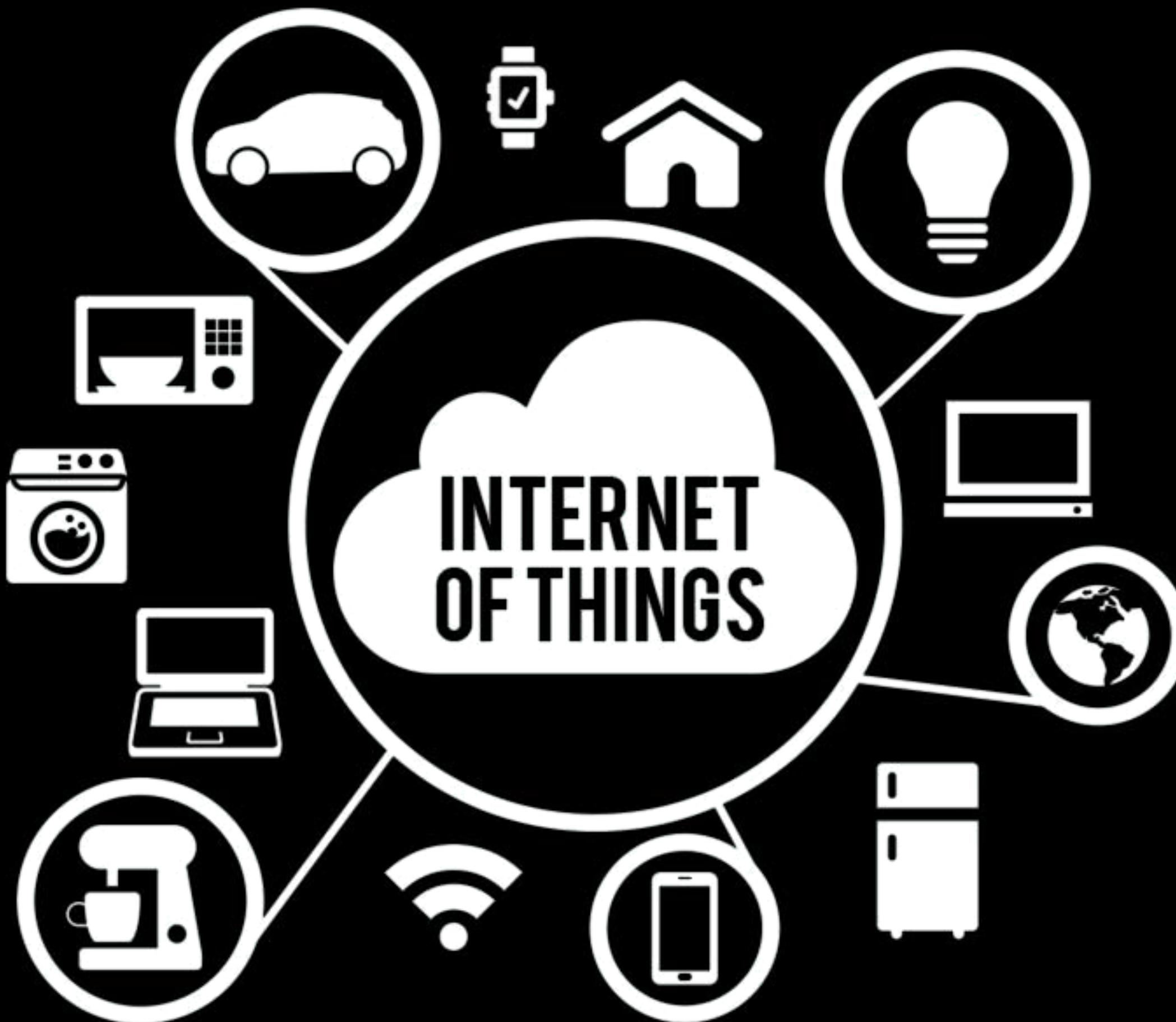


led.on()

↓  
led.on()



Controle de Liga/Desliga na Vida Real



Internet das Coisas (IoT)

# Resumo da Ópera

## Funcionalidade

## Comandos

### Timer

[acessar documentação](#)

```
from threading import Timer  
timer = Timer(3.5, funcao_para_quando_terminar)  
timer.start()  
timer.cancel()
```

### Sensor de Luz

[acessar documentação](#)

```
from gpiozero import LightSensor  
sensor = LightSensor(8)  
intensidade_de_luz = sensor.value  
sensor.threshold = 0.6 • sensor.light_detected  
sensor.when_dark = funcao • sensor.when_light = funcao
```

### Sensor de Movimento

[acessar documentação](#)

```
from gpiozero import MotionSensor  
sensor = MotionSensor(27)  
tem_movimento = sensor.motion_detected  
sensor.when_motion = funcao • sensor.when_no_motion = funcao
```

### IFTTT

[acessar documentação](#)

```
url = "https://maker.ifttt.com/trigger/evento/with/key/CHAVE"  
dados = {"value1": 2, "value2": -9.3, "value3": "olá"}  
resultado = post(url, json=dados)
```

## Funcionalidade

### FSWebcam

[acessar documentação](#)

```
from os import system
system("fswebcam --resolution 640x480 --skip 10 foto.jpg")
```

### ARecord

[acessar documentação](#)

```
from os import system • from subprocess import Popen
system("arecord --duration 3 --format cd audio.wav")
comando = ["arecord", "--duration", "30", "audio.wav"]
aplicativo = Popen(comando) • aplicativo.terminate()
```

### Lame

[acessar documentação](#)

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

### OpusEnc

[acessar documentação](#)

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

### Telegram

[acessar documentação](#)

```
from requests import post, get
base = "https://api.telegram.org/bot" + chave
endereco = base + "/sendMessage"
dados = {"chat_id": id_da_conversa, "text": "Olá!"}
resposta = post(endereco, json=dados)
print(resposta.text) • dicionario = resposta.json()
endereco = base + "/sendPhoto" • endereco = base + "/sendVoice"
arquivo = {"photo": open("foto.jpg", "rb")}
resposta = post(endereco, data=dados, files=arquivo)
dados = {"offset": proximo_id_de_update}
resposta = get(base + "/getUpdates", json=dados)
from urllib.request import urlretrieve
urlretrieve(endereco_do_arquivo, nome_do_arquivo_baixado)
```

## 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] • lista[1:3] • 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!" • texto[0] • texto[1:4] • len(texto) texto + "\n" • texto + str(numero) • "x = %.2f" % numero 2 + int("11") • 4 / float("23.5")</pre>
Condicionais	<pre>if x != 0:     if x not in [1, 2]:         y = 4     else:         y = 0 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>