

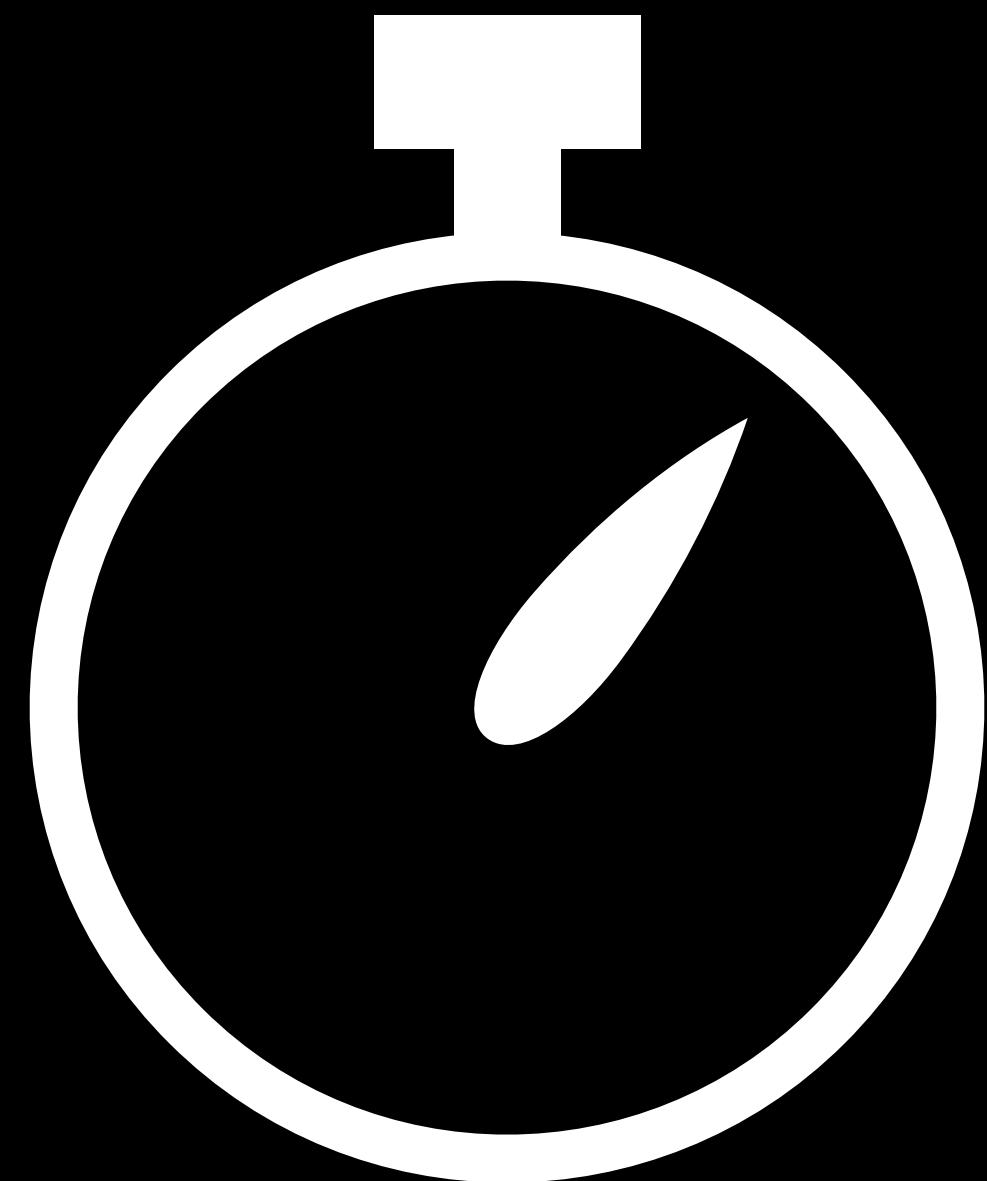
Projeto 05

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

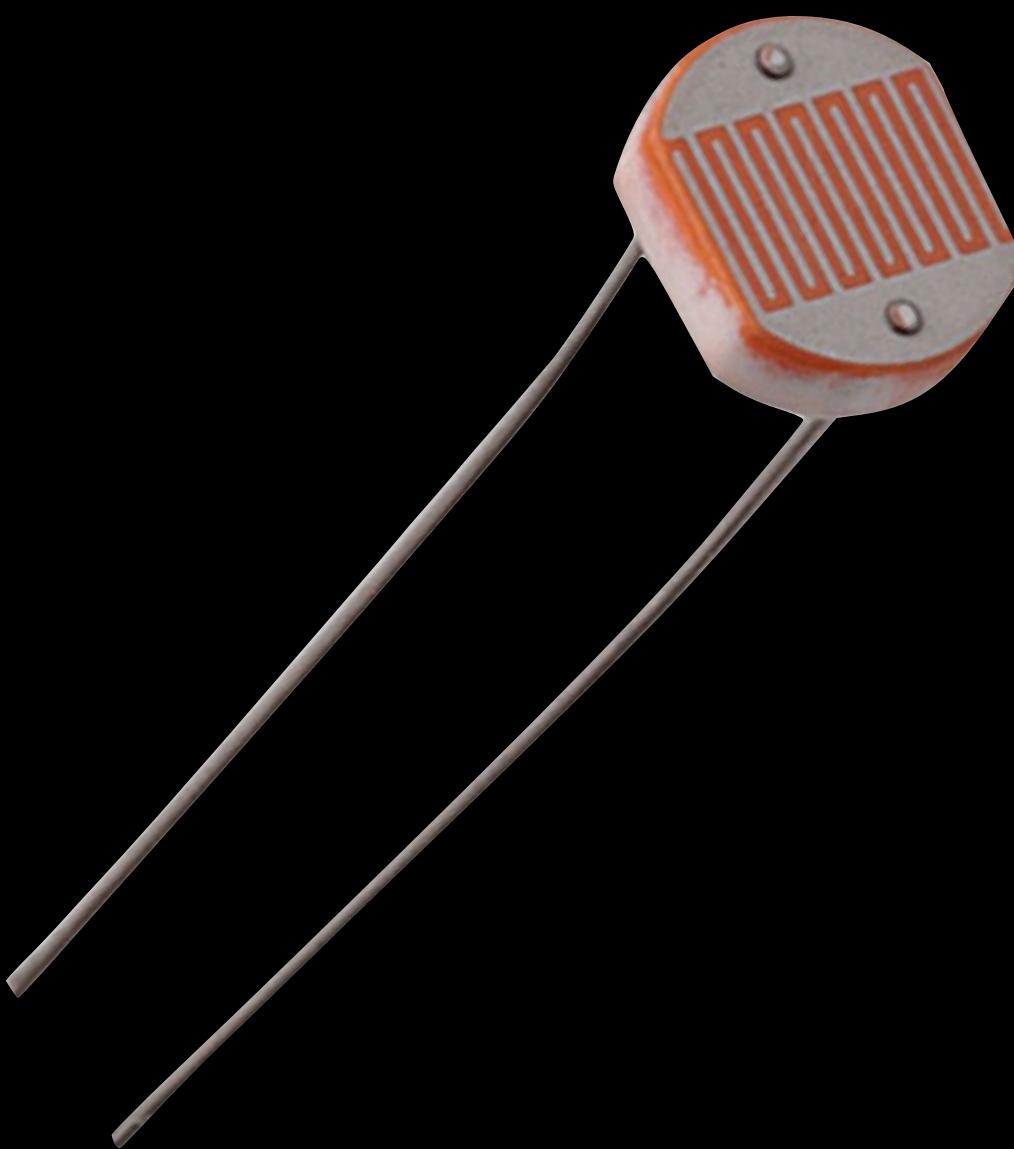
1 segundo

```
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  
...  
...
```

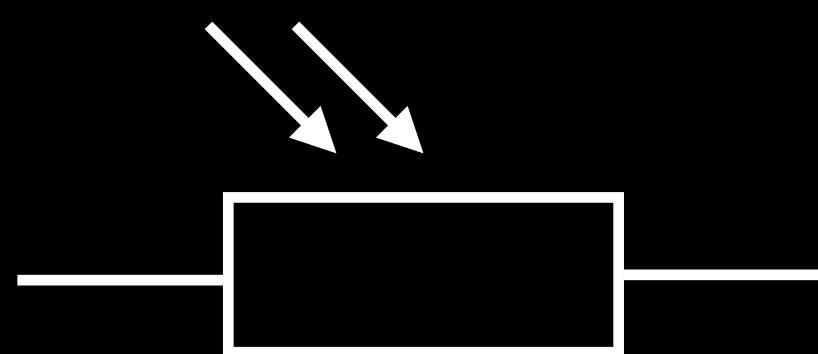
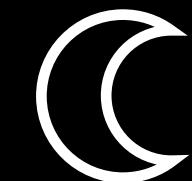
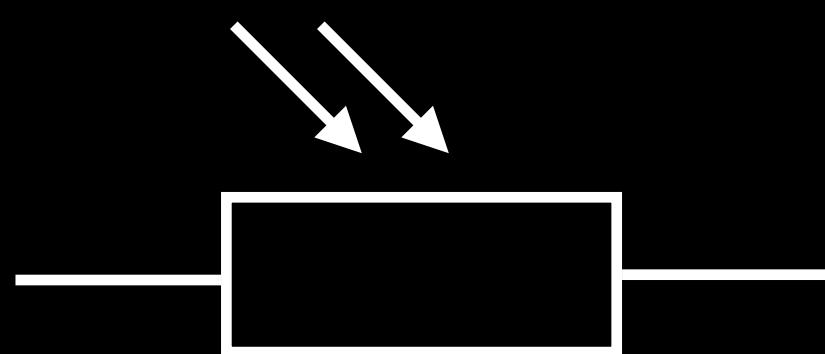
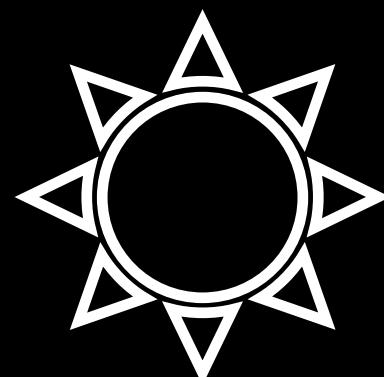
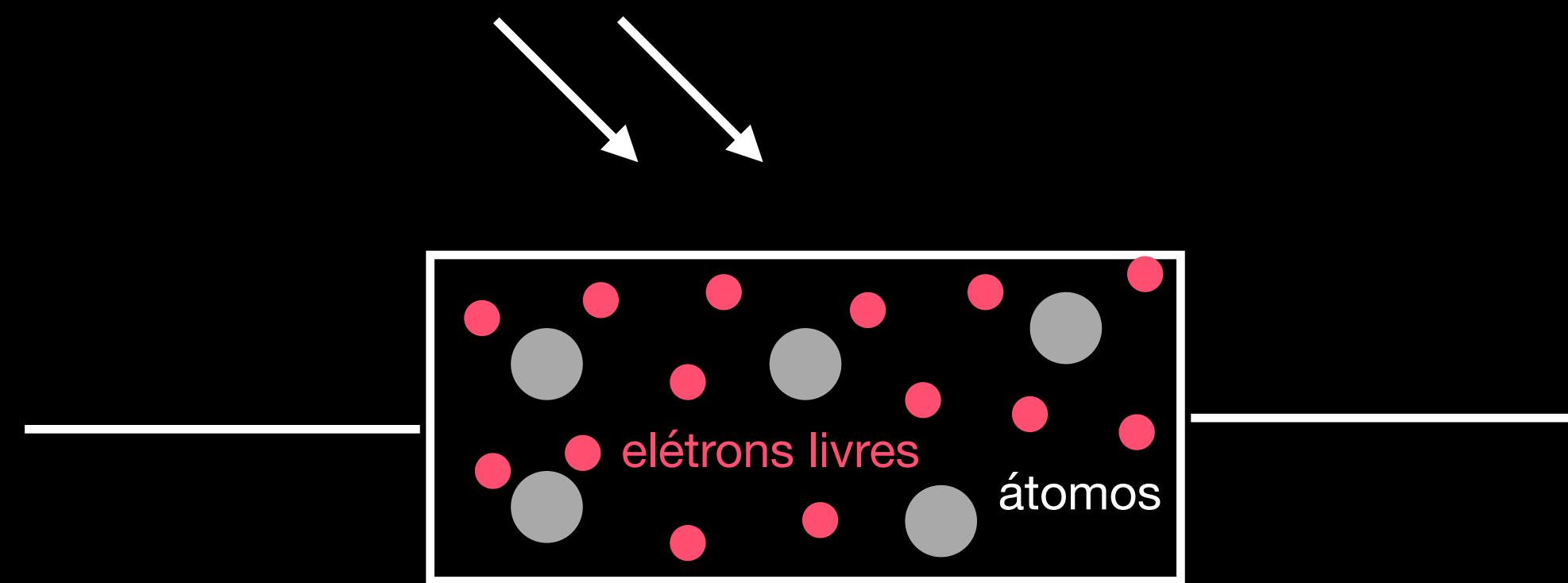
chame depois a própria função!

Hardware

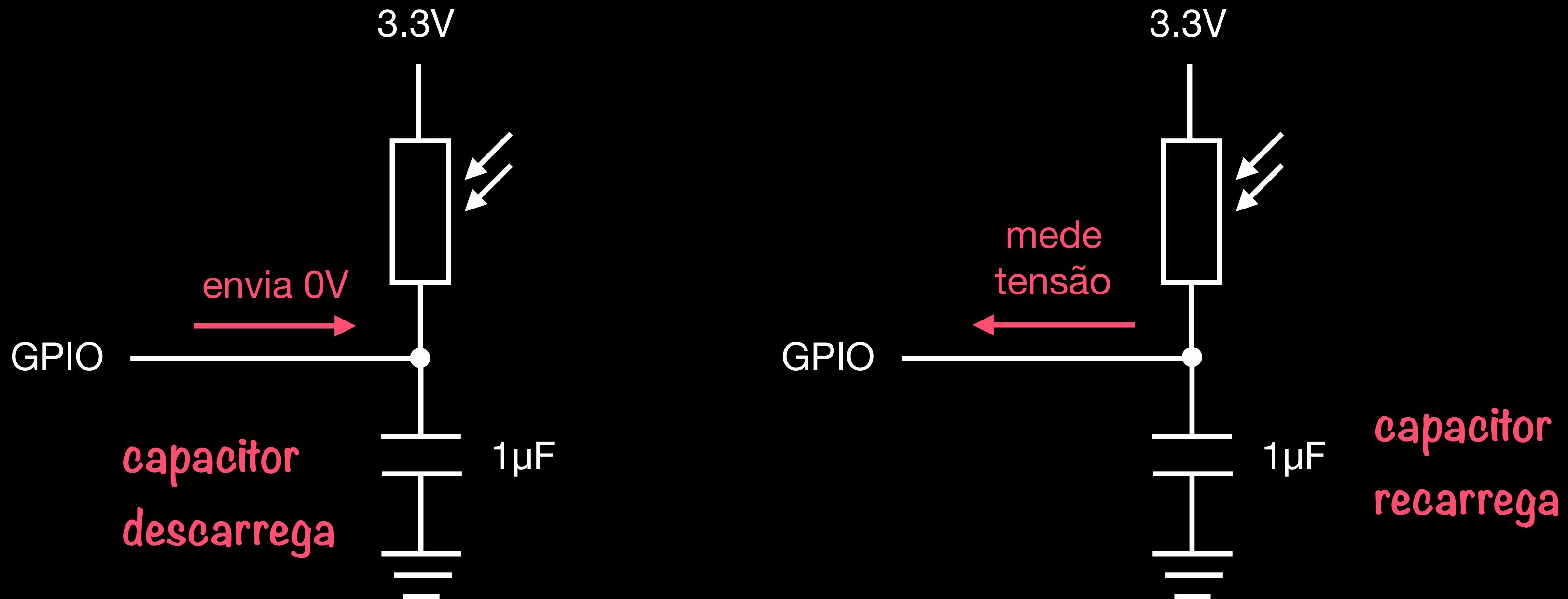


Sensor de Luz

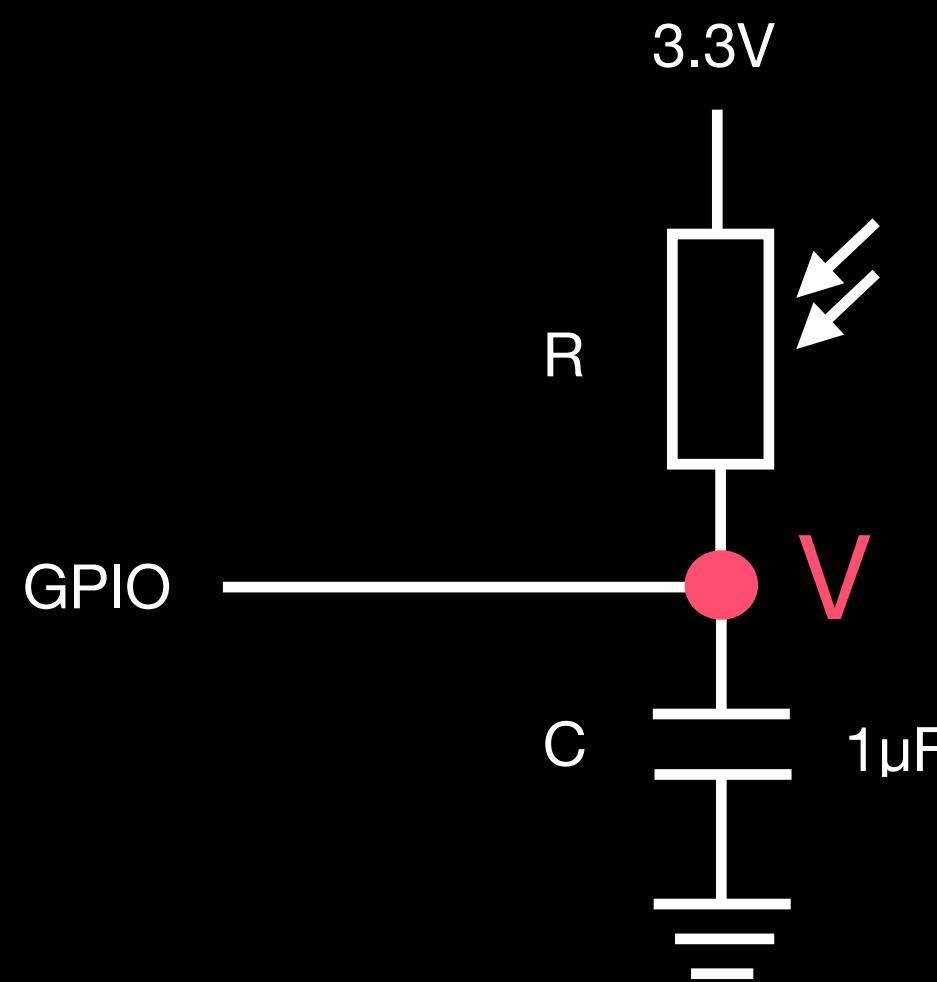
fótons da luz



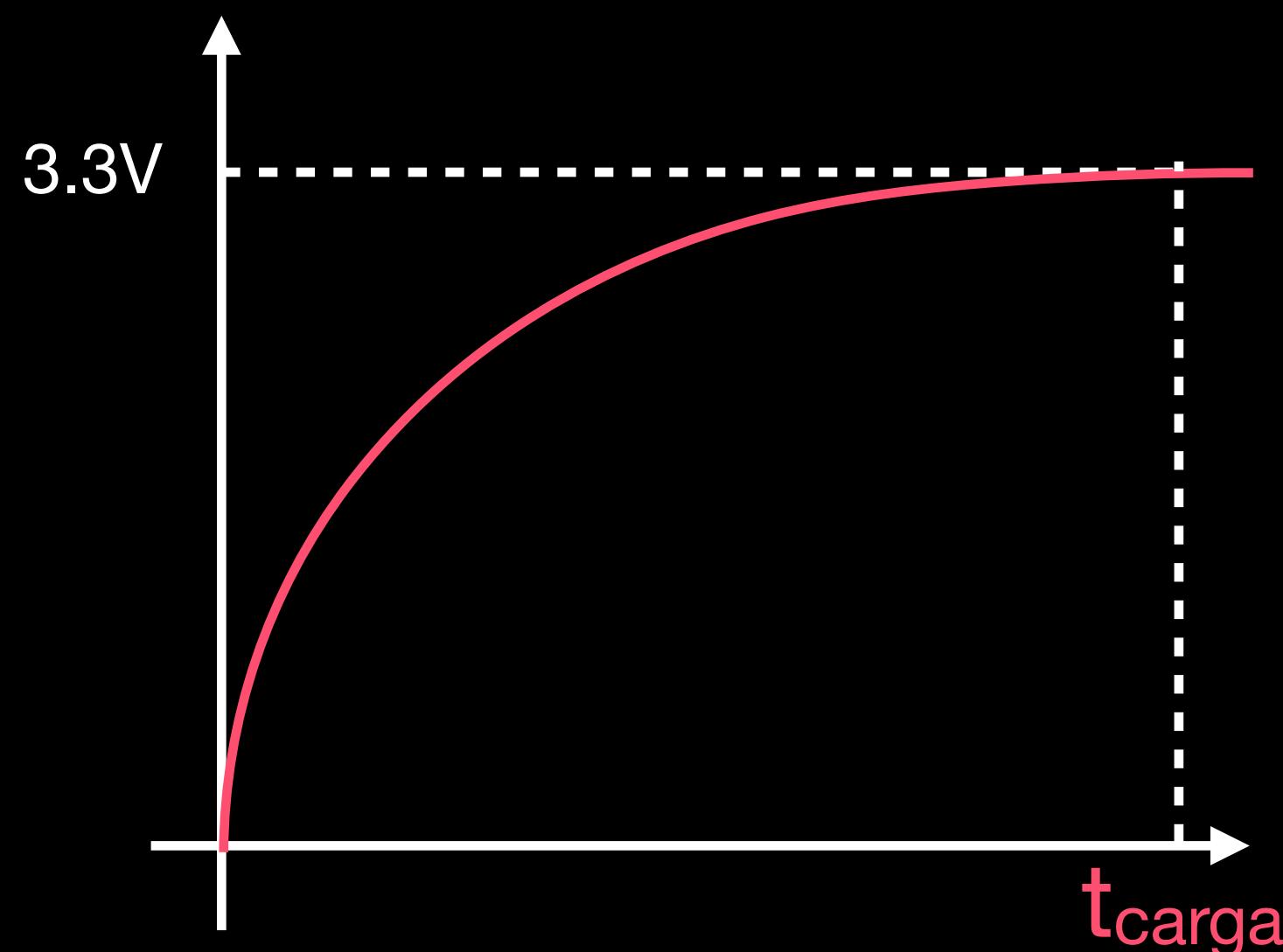
Resistência Dependente de Luz (LDR) ou Fotoresistor



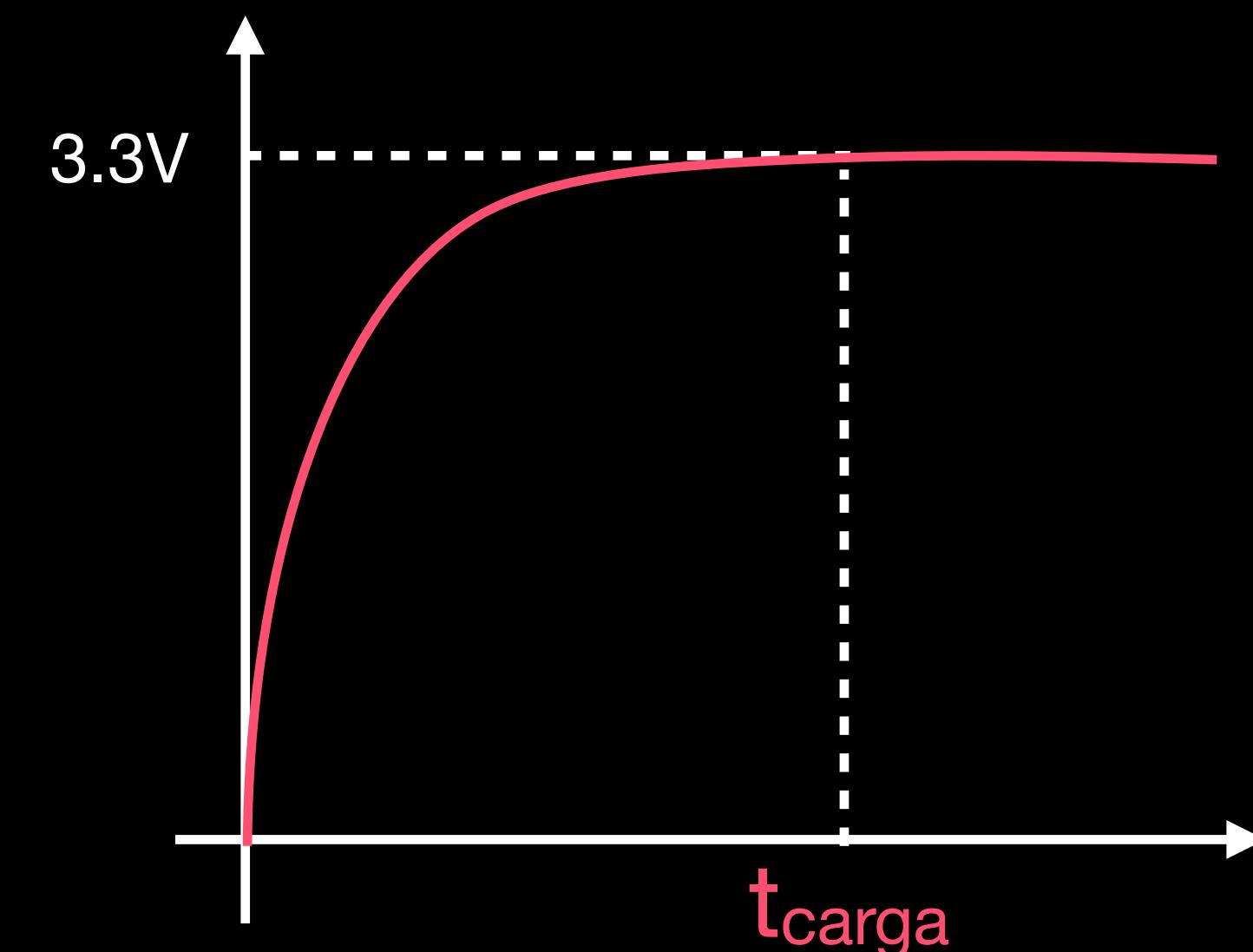
Círculo com Capacitor para Detecção de Luz



Carregamento sem Luz (R Maior)

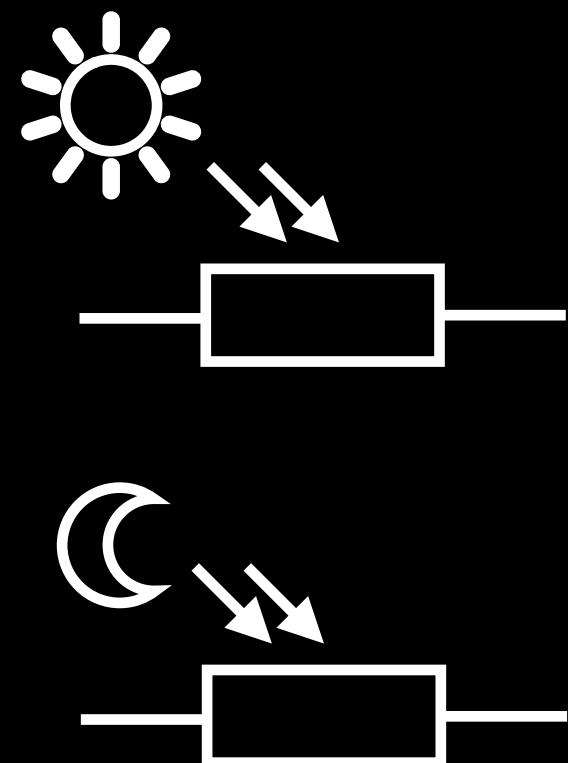


Carregamento com Luz (R Menor)



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

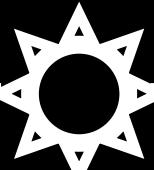
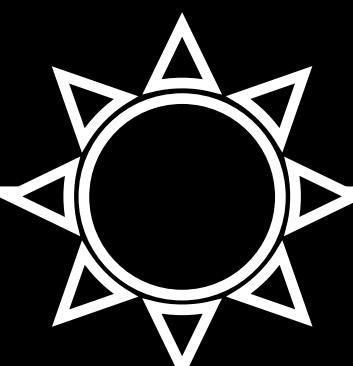
0.862208

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

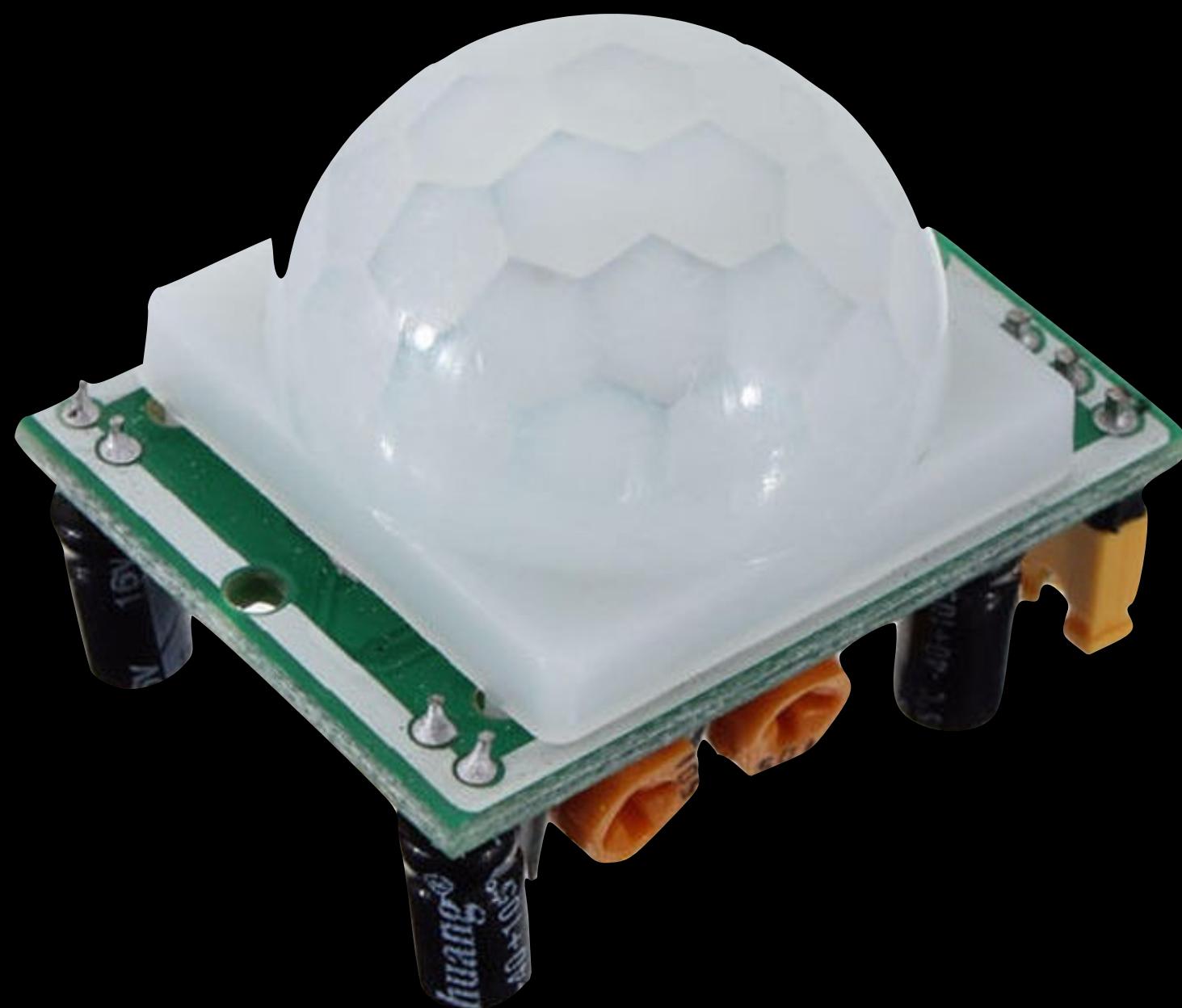


Sensibilidade a Mudanças de um Sensor de Luz

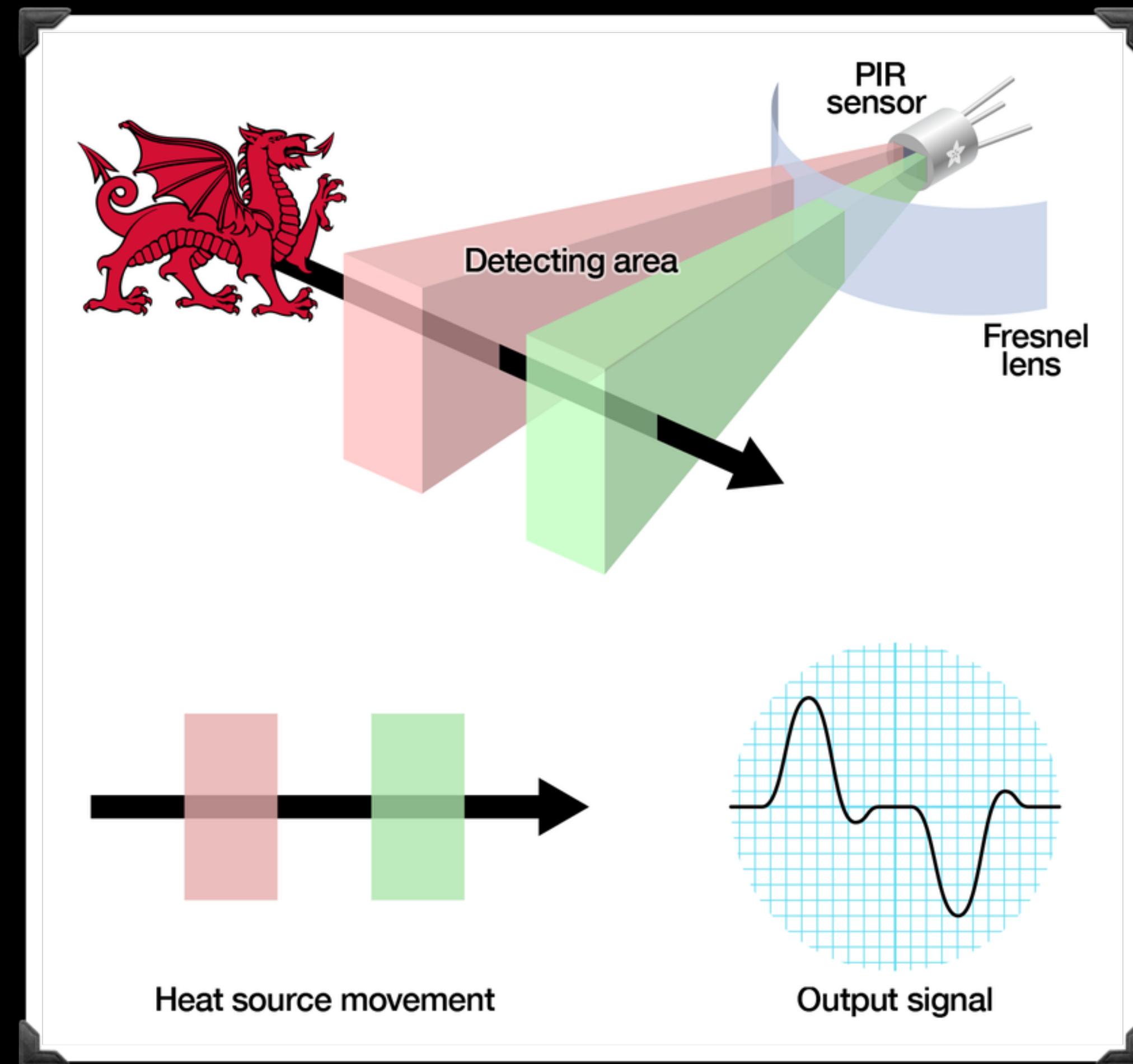
```
>>> from gpiozero import LightSensor  
>>> sensor_de_luz = LightSensor(8)  
>>> sensor_de_luz.threshold  
0.5  
>>> sensor_de_luz.value  
0.56743  
>>> sensor_de_luz.light_detected  
True  
  
>>> sensor_de_luz.threshold = 0.6  
>>> sensor_de_luz.value  
0.55891  
>>> sensor_de_luz.light_detected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

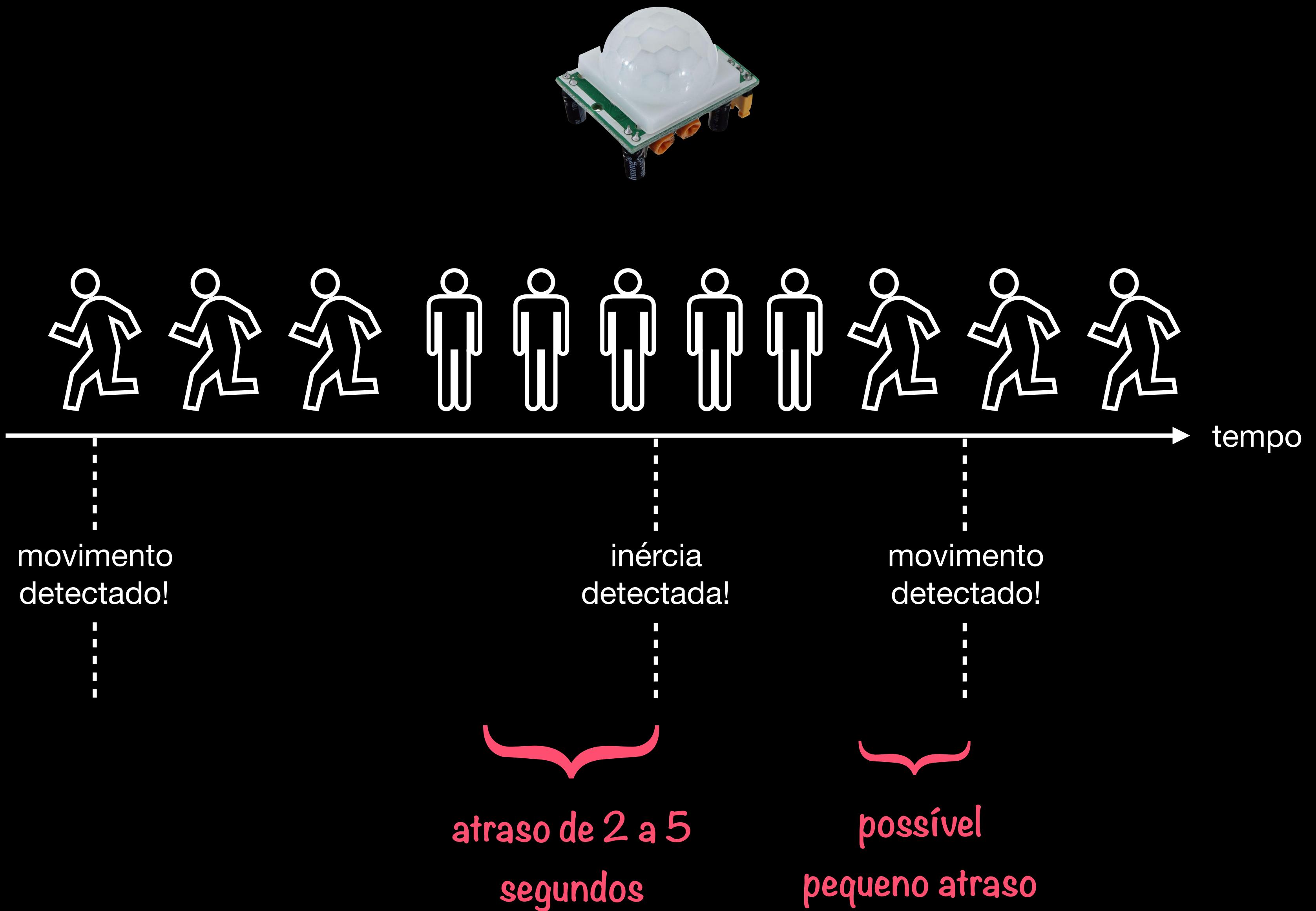


Funcionamento de um Infravermelho Passivo (PIR)

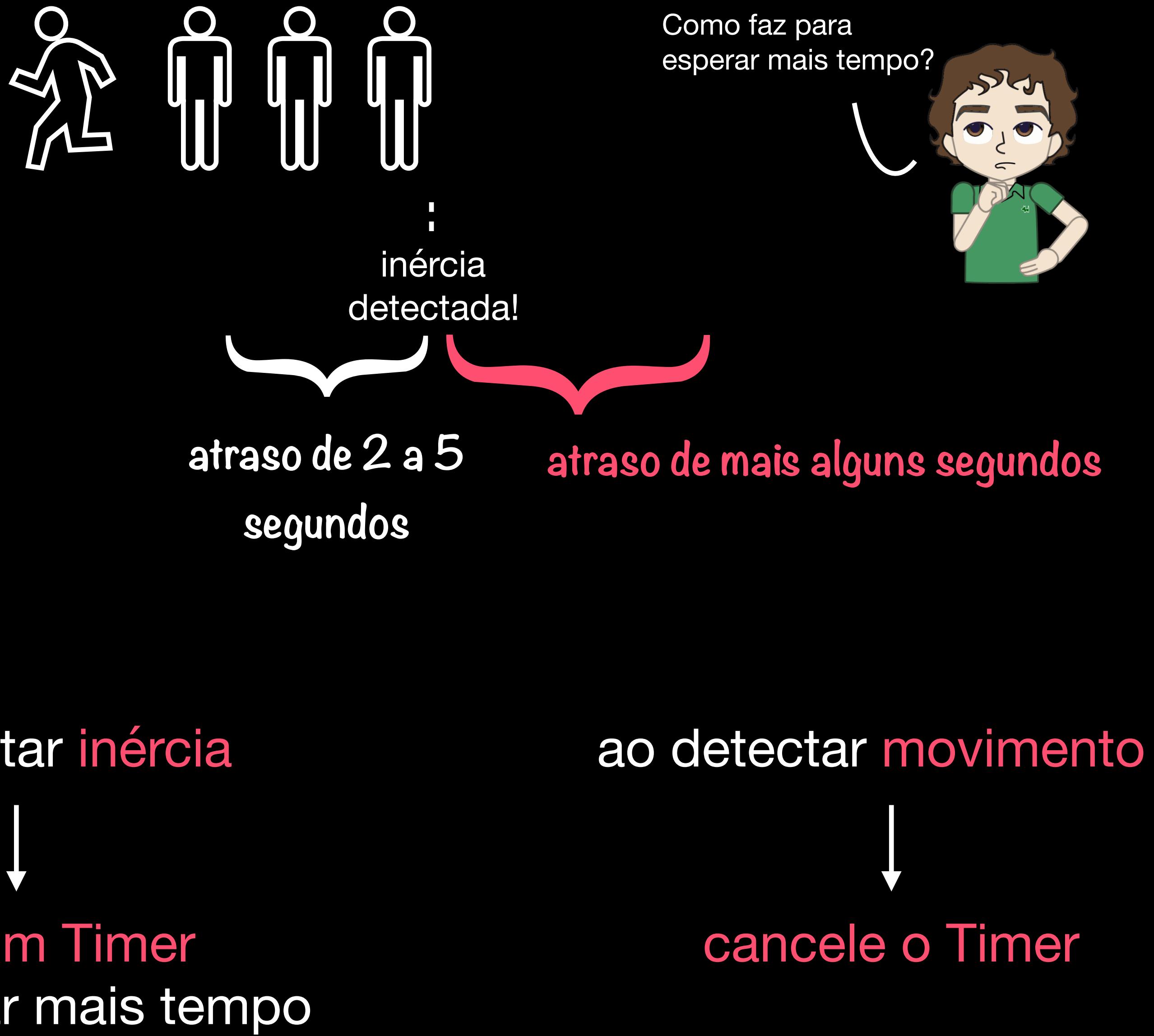
```
>>> from gpiozero import MotionSensor  
>>> sensor_de_movimento = MotionSensor(27)  
>>> sensor.motion_detected  
False  
>>> def movimento_detectado():  
...     print("Mexe a cadeira, hey!")  
...  
>>> sensor_de_movimento.when_motion = movimento_detectado  
>>> def inercia_detectada():  
... 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



Acréscimo no Tempo de Espera da Inéria

Software

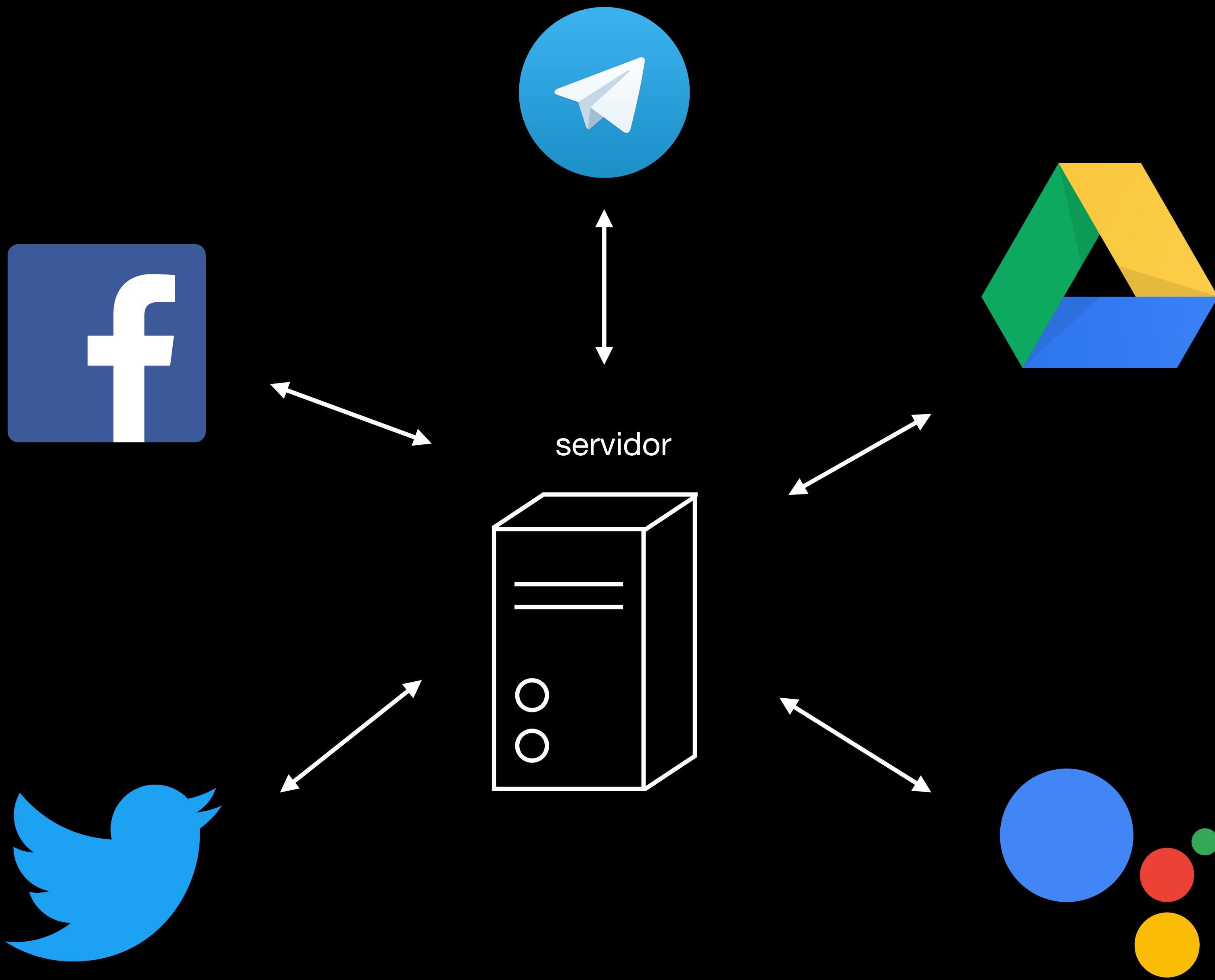


Flask

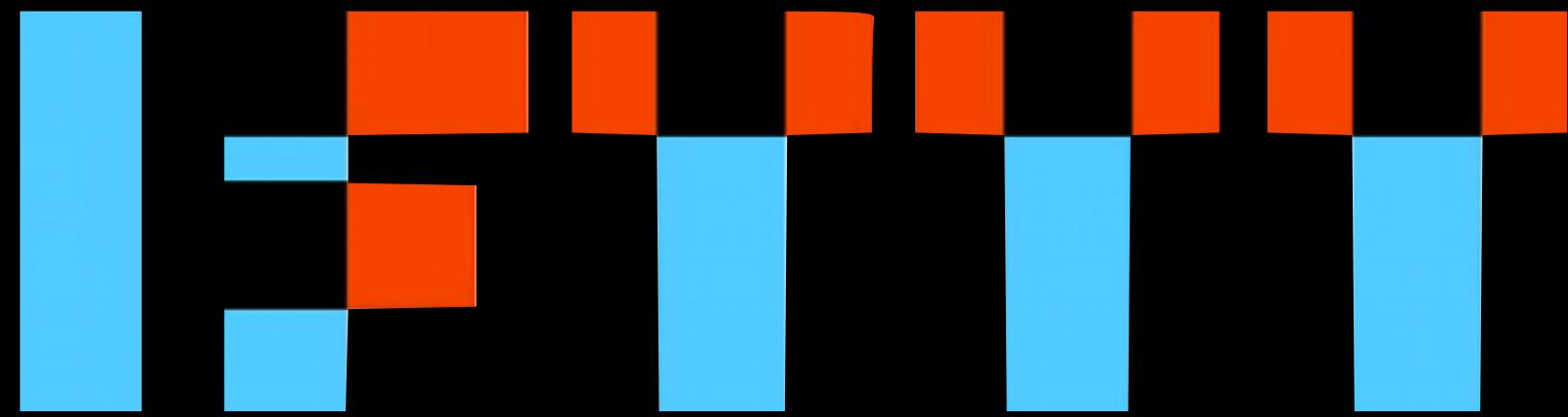
web development,
one drop at a time



Uso de Softwares dos Projetos Passados

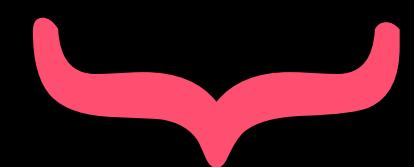


Dificuldade em Programar Várias Comunicações

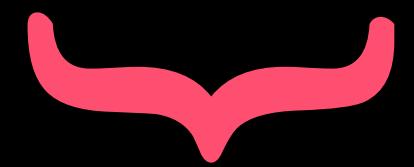


"applet" →

if this than that



gatilho



ação

IFTTT: If This Than That

The image shows the homepage of IFTTT (If This Then That) on a Mac OS X browser. The title "IFTTT" is at the top left, followed by a large, bold headline: "A world that works for you". Below the headline is a paragraph explaining IFTTT's mission: "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 is a central graphic illustrating how various devices and services interact. 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

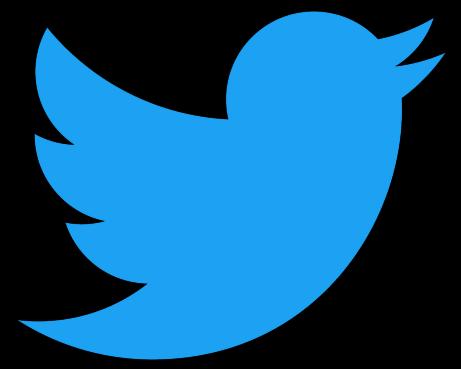
Choose a service

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



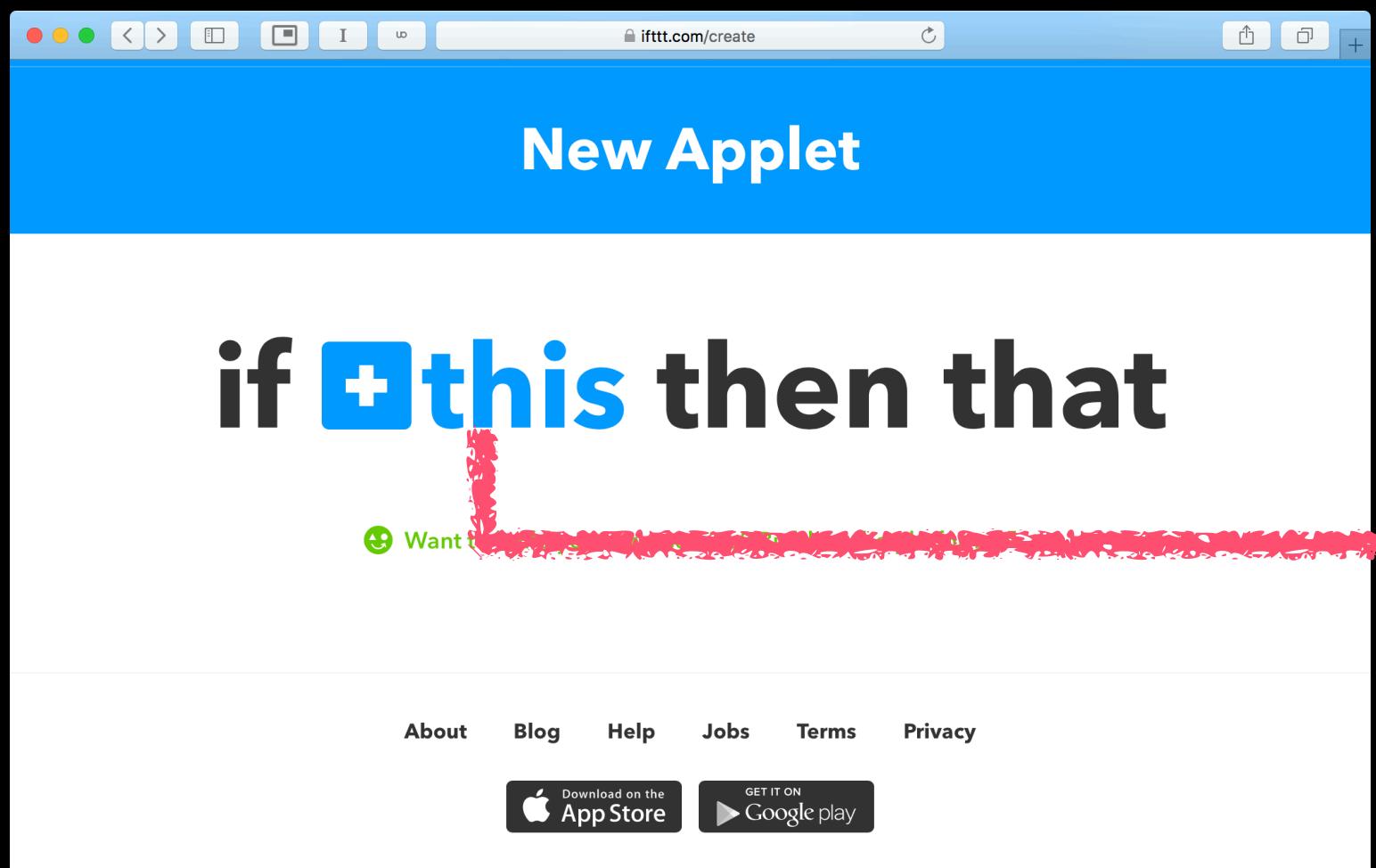
então



se um usuário tuitar

mande-me um SMS

Exemplo de Integração entre Twitter e SMS

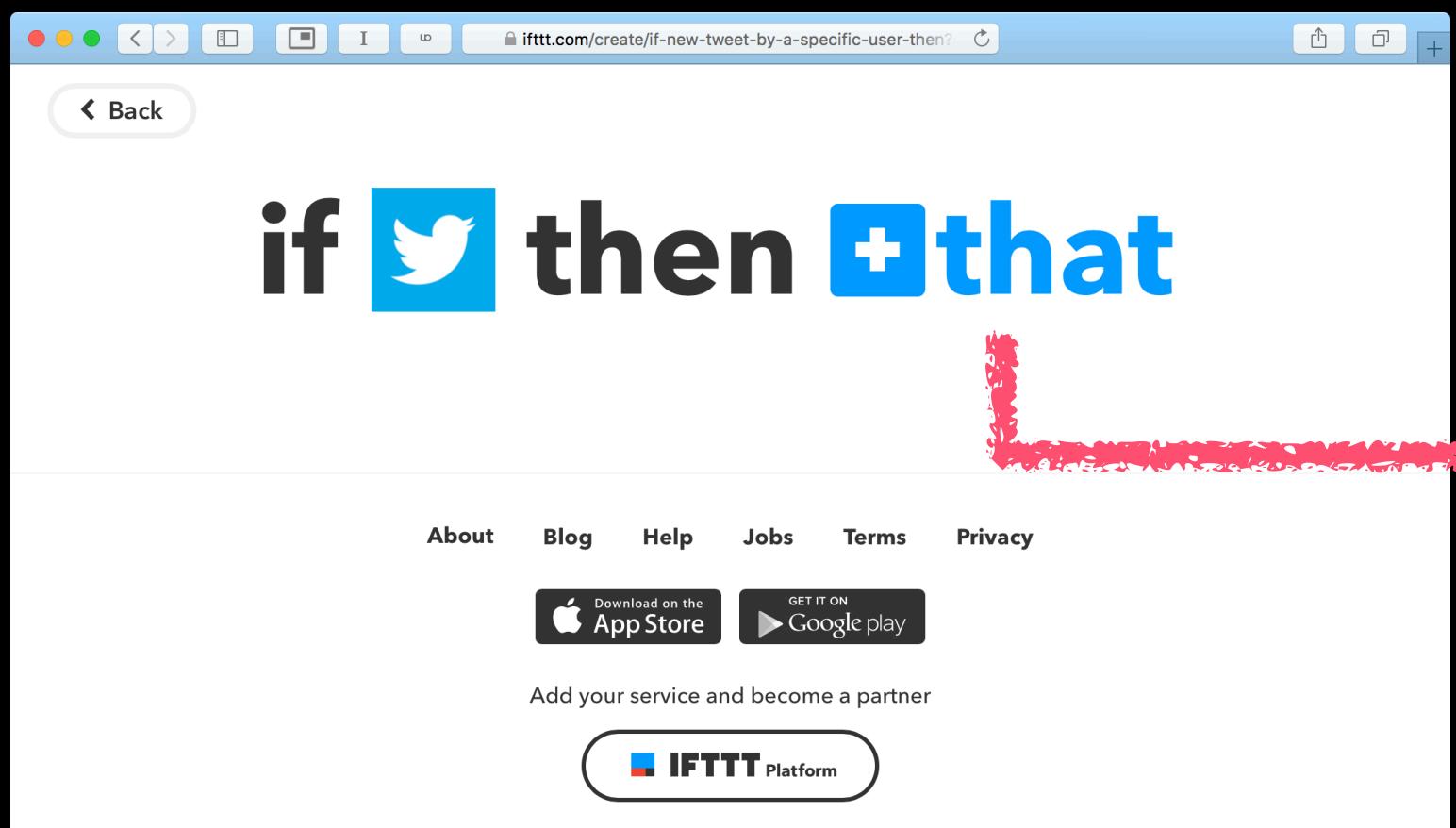


New tweet by a specific user

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

Username to watch

Create trigger



Send me an SMS

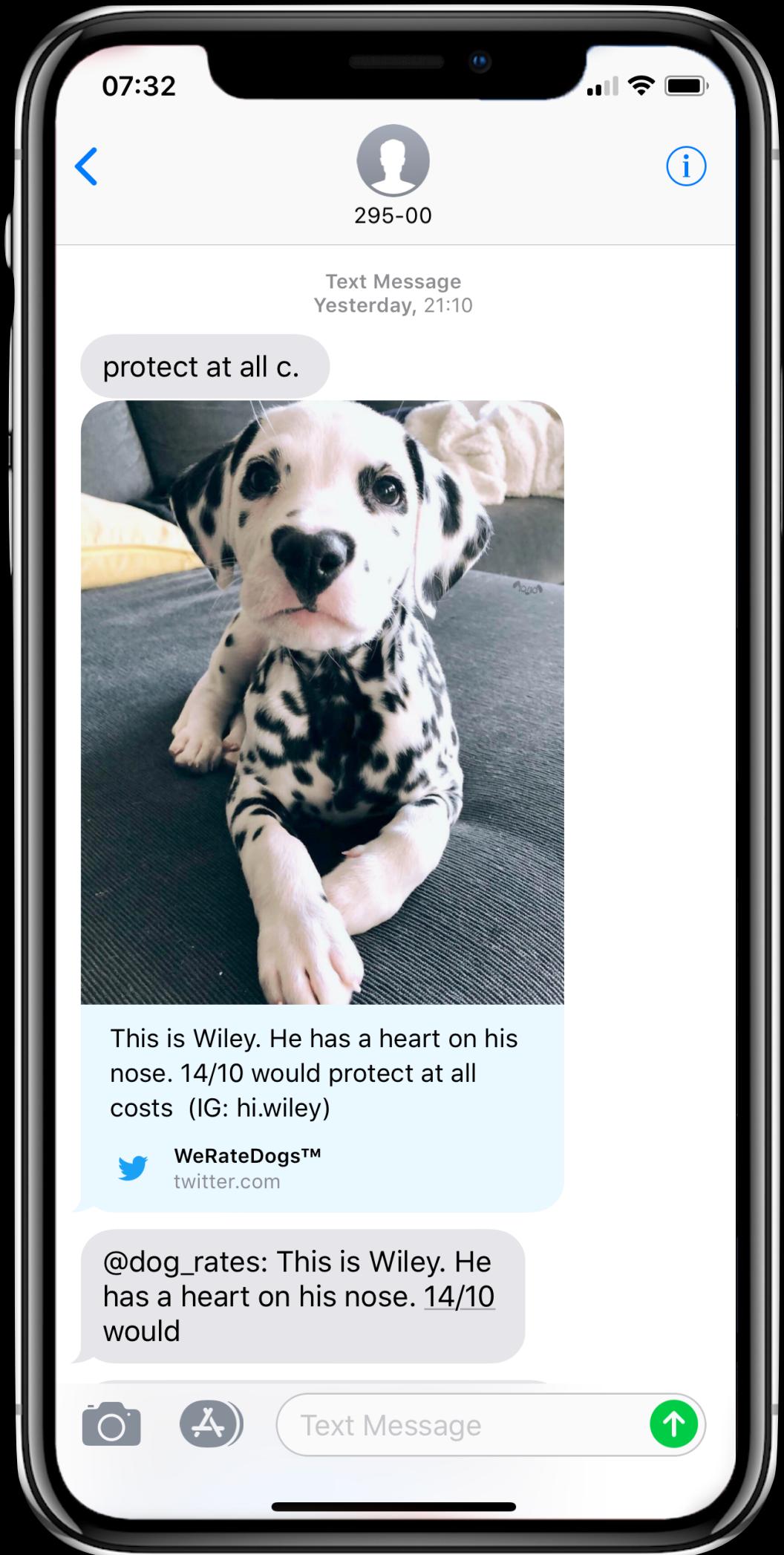
This Action will send an SMS to your mobile phone.

Message

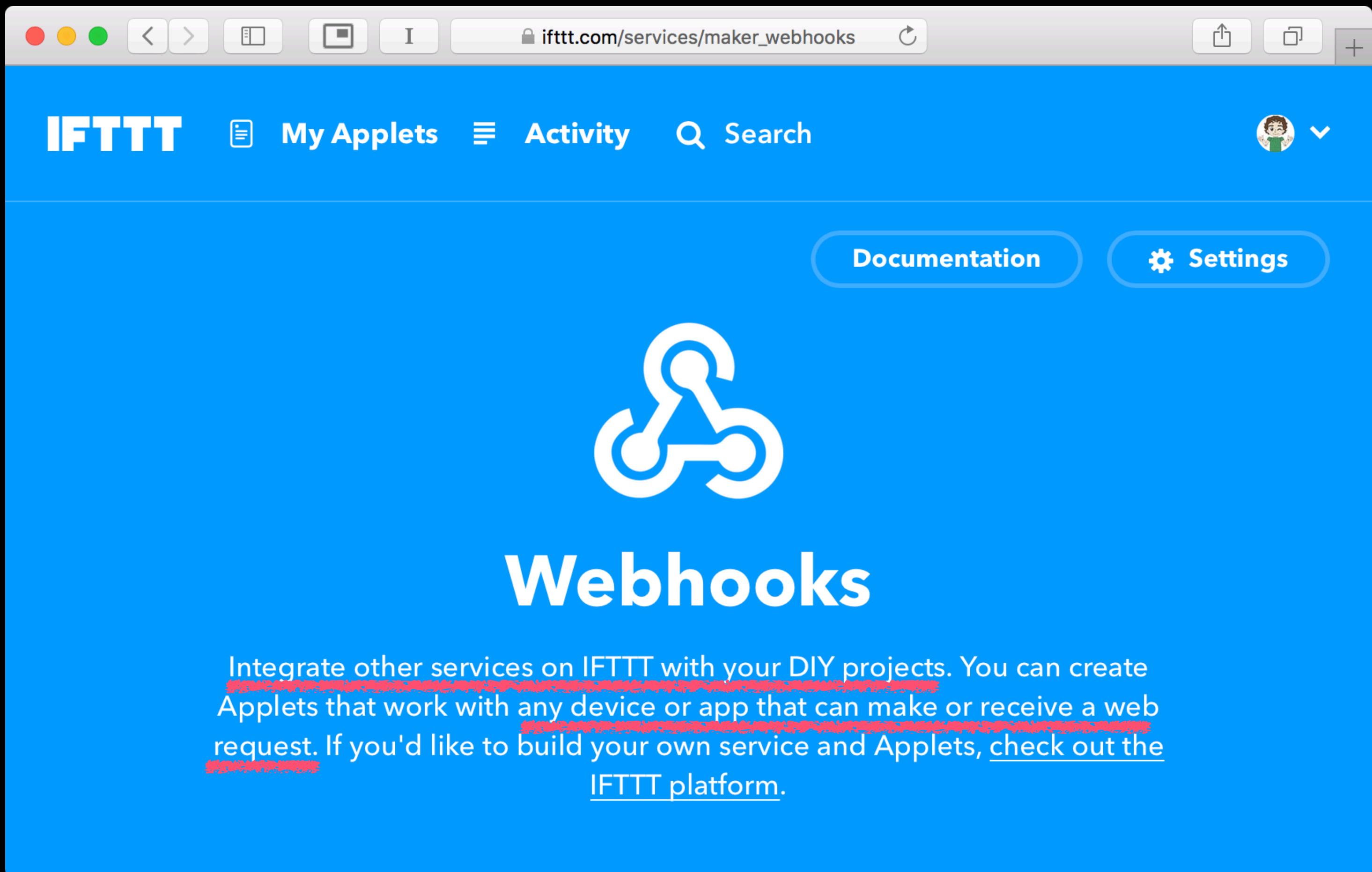
Add ingredient

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's address bar displays "ifttt.com/services/maker_webhooks". The main content area has a blue header with the IFTTT logo, navigation links for "My Applets", "Activity", and "Search", and a user profile icon. Below the header is a large white "Webhooks" logo consisting of three interconnected circles. The main text on the page 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." A red horizontal line highlights the first sentence, and another red horizontal line highlights the last sentence.

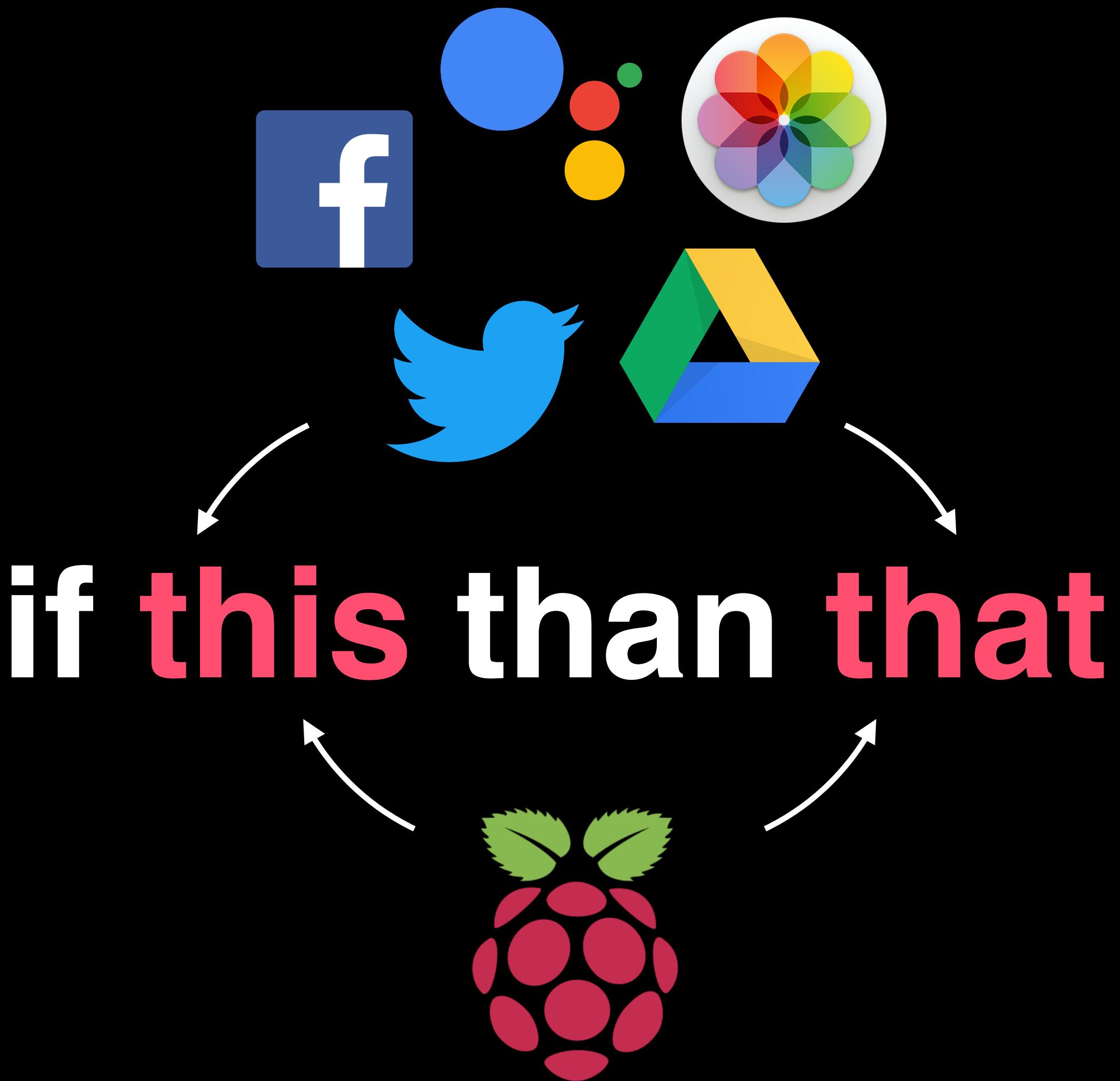
IFTTT My Applets Activity Search

Documentation Settings

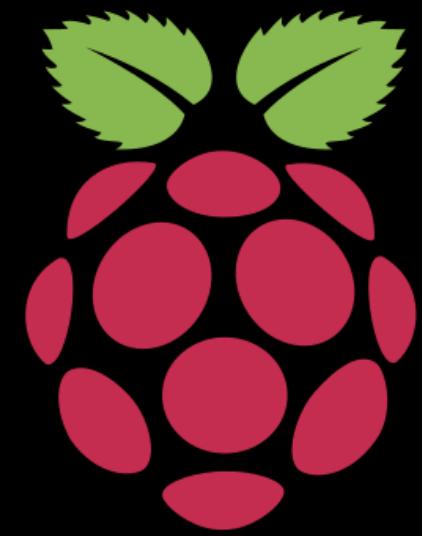
Webhooks

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

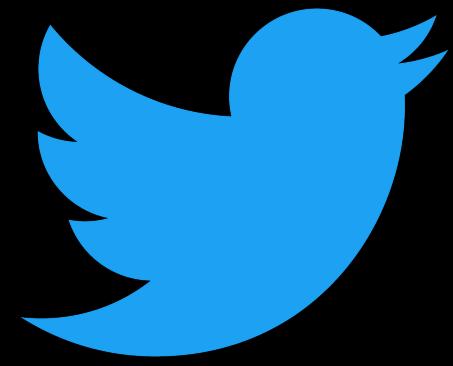


Integração entre Raspberry Pi e Serviços



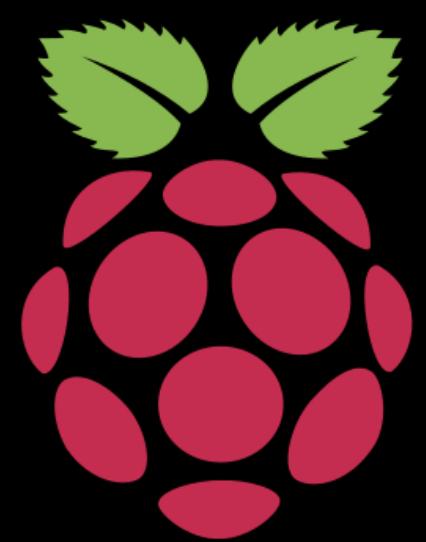
se botão for
pressionado

então

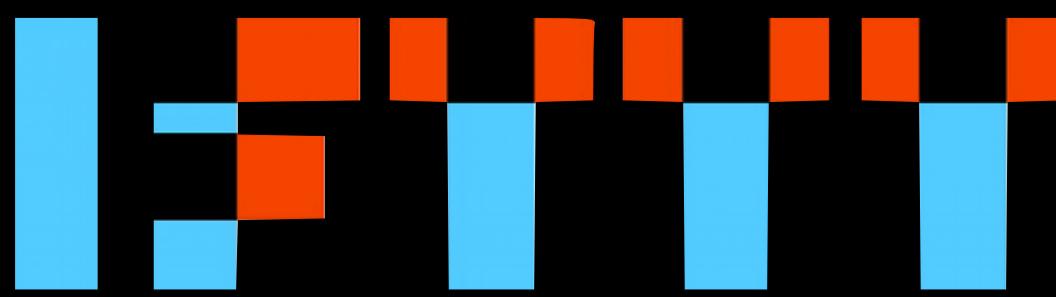


poste um
tweet

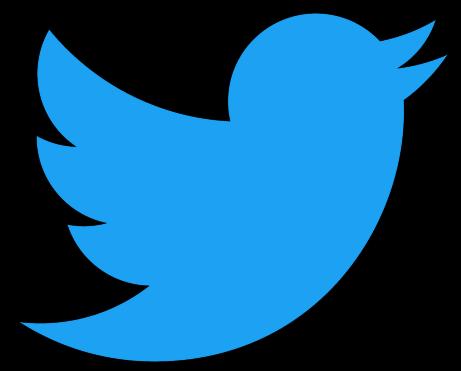
Exemplo de Integração entre Raspberry Pi e Twitter



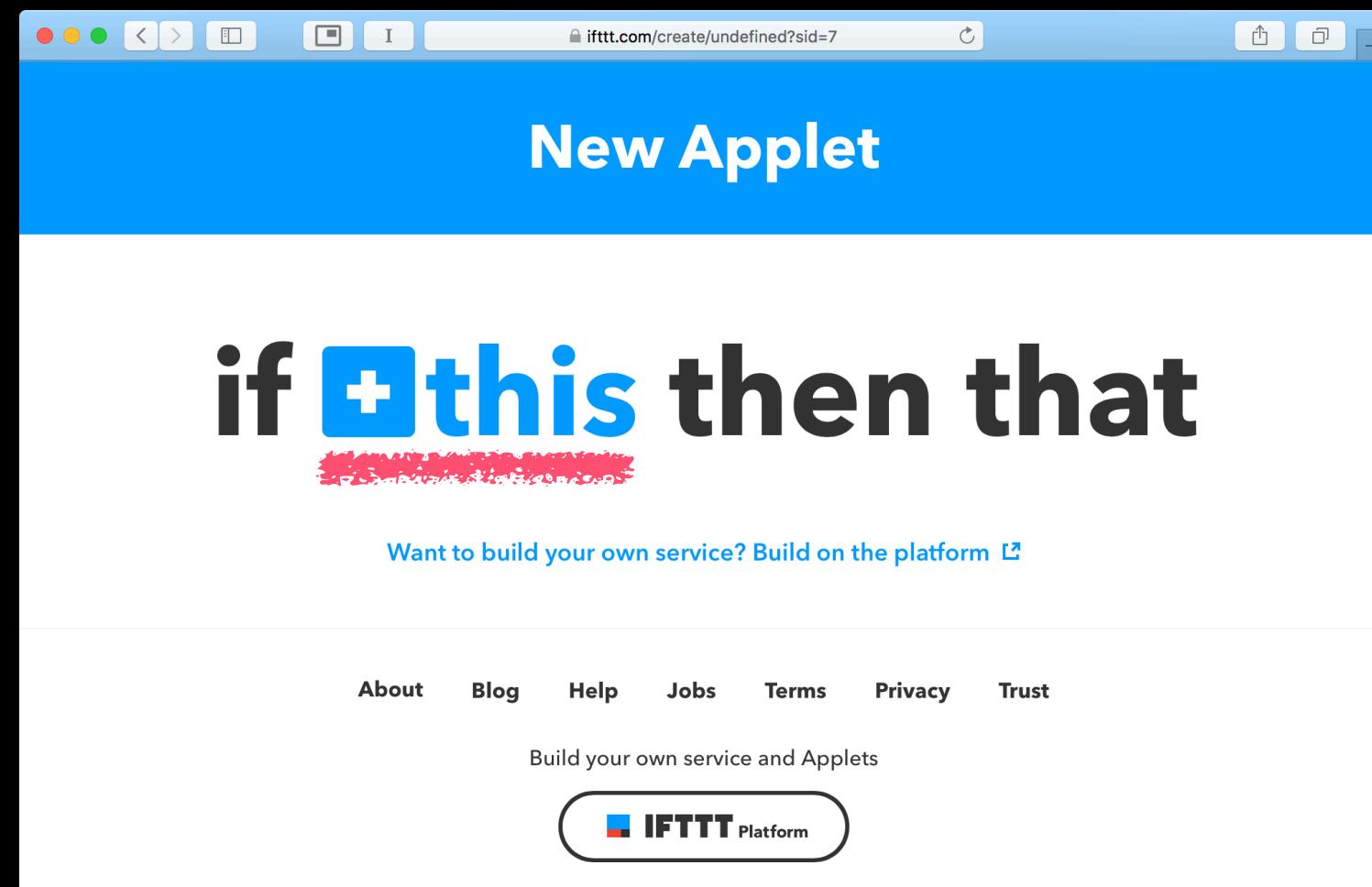
THIS
evento
"botao_pressionado"



THAT
tweet



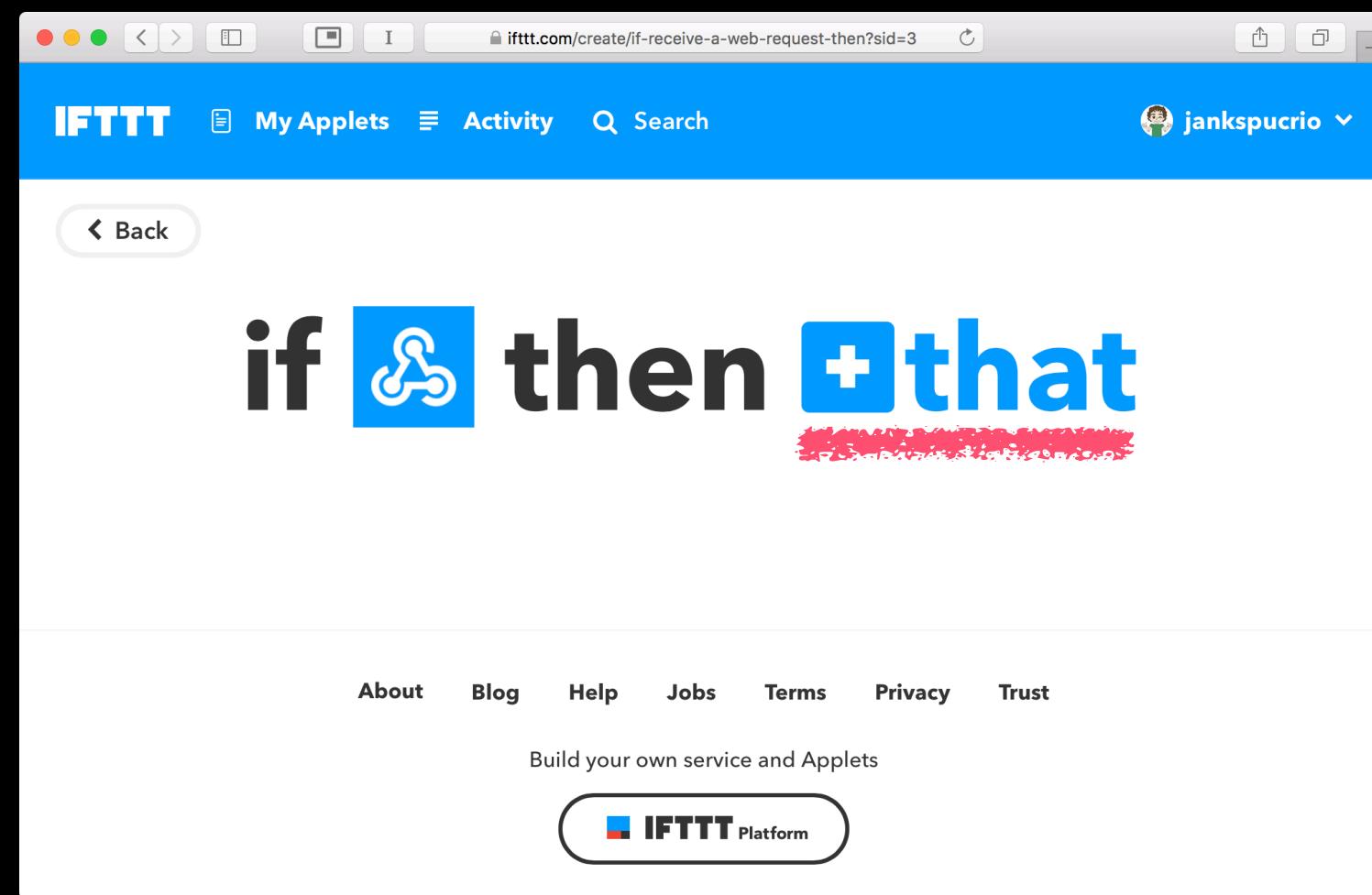
Envio de Eventos ao IFTTT



This screenshot shows the 'Choose a service' step of a new applet creation. The URL in the address bar is 'ifttt.com/create/if?sid=0'. The title 'Choose a service' is at the top, followed by 'Step 1 of 6'. A search bar contains the text 'webhooks'. Below the search bar is a blue square icon with a white 'Webhooks' logo. To the right of the search bar, there's descriptive text about the 'Maker service' trigger, which fires every time it receives a web request. It includes instructions to go to the Maker service settings and provides examples for the event name. An input field shows the event name 'botao_pressionado'. A large blue button at the bottom right says 'Create trigger'. Navigation links for 'About', 'Blog', 'Help', 'Jobs', 'Terms', and 'Privacy' are at the bottom, along with download links for the App Store and Google Play.

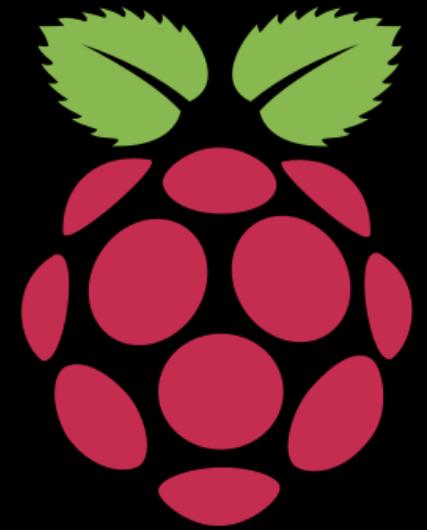
nome do
"evento"

Configuração do "This" como Serviço Webhook



This screenshot shows the fourth step of creating an IFTTT applet. The title 'Choose action' is at the top, with a back arrow on the left. Below it, there are two options: 'Post a tweet' and 'Post a tweet with image'. The 'Post a tweet' option is selected and its details are shown: 'This Action will post a new tweet to your Twitter account. NOTE: Please adhere to Twitter's Rules and Terms of Service.' To the right, a large blue box titled 'Post a tweet' contains the tweet text 'O valor medido pelo sensor de luz é {{Value1}}.' Below this, there is an 'Add ingredient' button and a list of parameters: EventName (Value1, Value2, Value3), and OccurredAt.

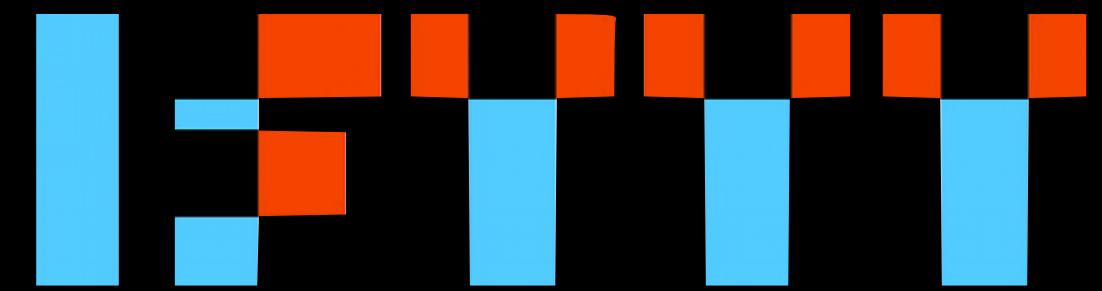
Configuração do "That" como Tweet com Parâmetros



post com dados



post com dados



Envio de Requisições ao IFTTT

https://ifttt.com/services/maker_webhooks/settings

The screenshot shows a web browser window with the URL 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" (the URL is redacted), and "Status: active". There is a button labeled "Edit connection". To the right of the redacted URL, the text "anote esta chave!" is written in red.

My Applets > Webhooks

Webhooks settings

[View activity log](#)

Account Info

Connected as: wallysalami

URL: 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 browser window. The tweet is from the account **ENG1419 – Programação de Micro...** (@eng1419). The message reads: "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, and #LTSIG.

Twitter, Inc. twitter.com/eng1419/status/983321886523297794

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

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}

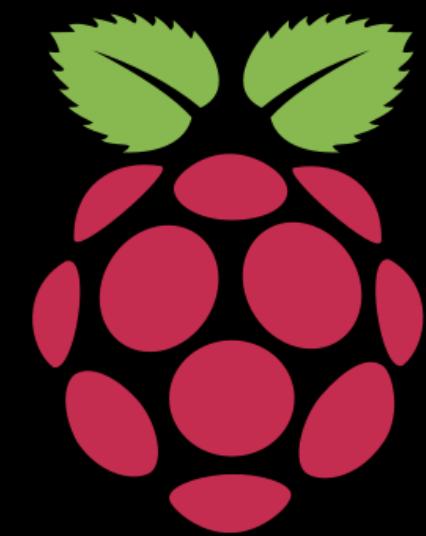
"v" minúsculo



Atenção à Variação da Ortografia de Value1



então



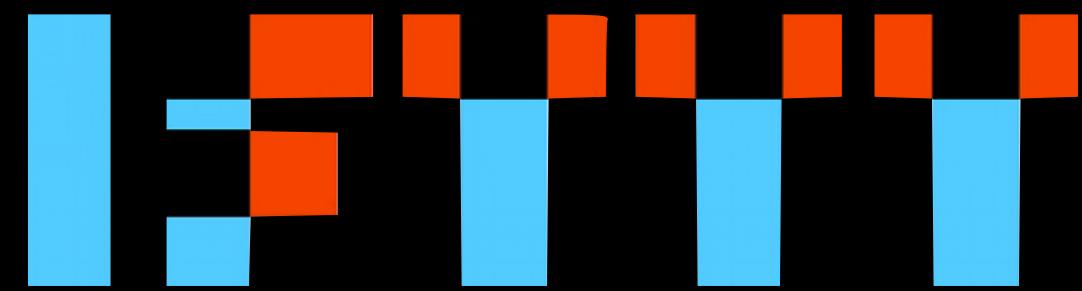
se houver
previsão do tempo

mostre a previsão
no LCD

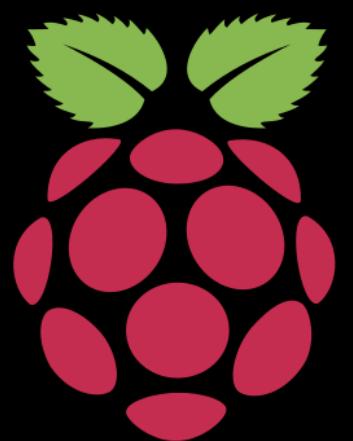
Exemplo de Integração entre Weather Underground e Raspberry Pi



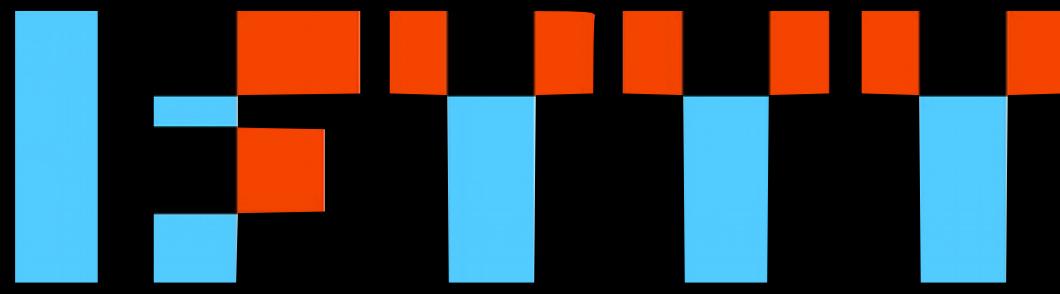
THIS
mudança
no tempo



THAT
acessa endereço

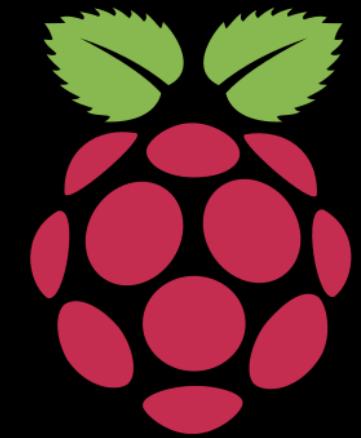


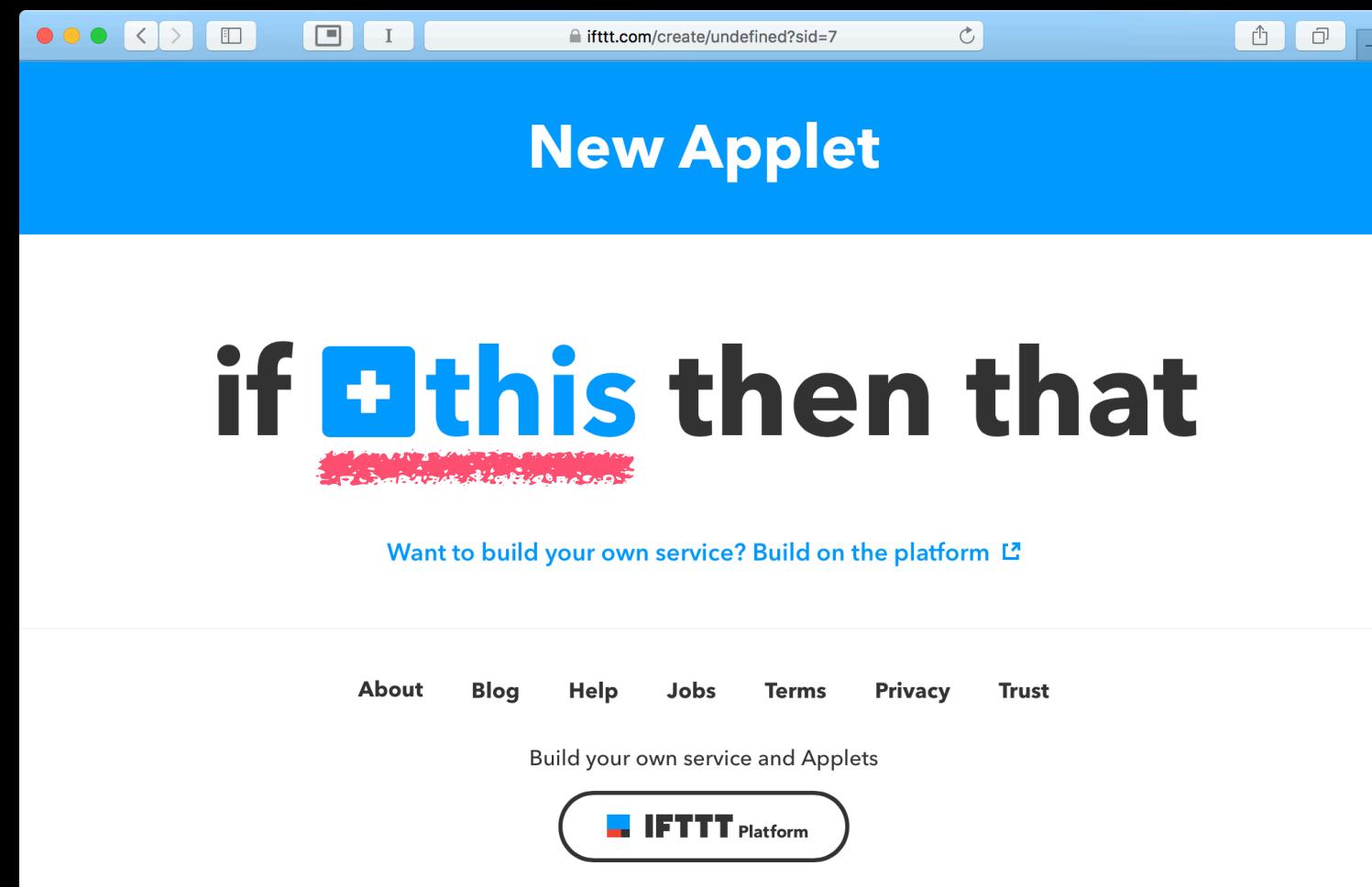
Recebimento de Eventos do IFTTT



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

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





A screenshot of the IFTTT "Choose trigger" screen. The title is "Choose trigger" and it's step 2 of 6. It shows two options: "Today's weather report" and "Tomorrow's weather report". Both descriptions mention that the trigger retrieves current weather reports at specified times, noting that pollen count is available only in the USA. The "Today's weather report" option is selected.

A screenshot of the trigger configuration screen for "Today's weather report". The title is "Today's weather report". It states that the trigger retrieves today's current weather report at the specified time. It includes fields for "Time of day" (set to 08 AM) and "00 Minutes". A large "Create trigger" button is at the bottom.

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 central text area says "Build your own service and Applets" and features a "IFTTT Platform" button.

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 large blue button labeled "Webhooks" with a corresponding icon. At the bottom of the page are standard IFTTT navigation links: "About", "Blog", "Help", "Jobs", "Terms", "Privacy", and "Trust".

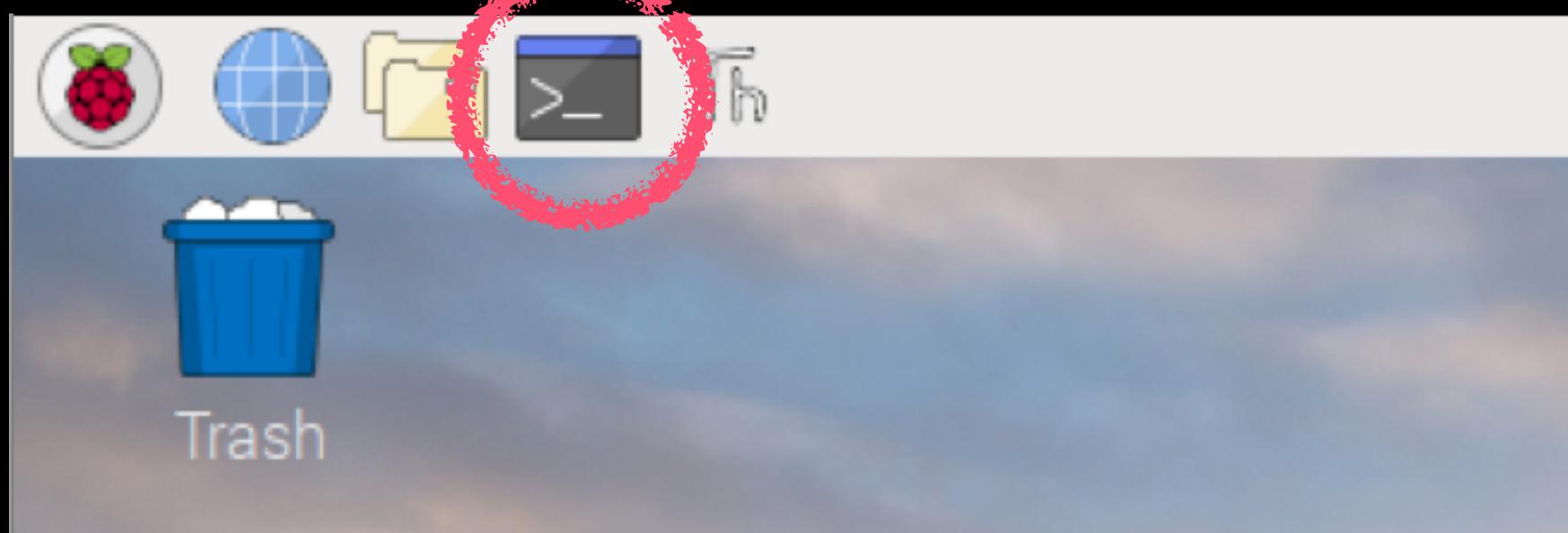
This screenshot shows the configuration for the "Make a web request" action, which is highlighted with a red border. The title "Make a web request" is at the top. A note states: "This action will make a web request to a publicly accessible URL. NOTE: Requests may be rate limited." To the right is a cartoon character of a boy pointing upwards. The configuration fields include:

- URL:** `https://fd288c990i.ngrok.io/tempo/{{CurrentCondition}}`
- Method:** `GET`

Below the URL field is a note: "Surround any text with '<>>' to escape the content". There is also a "Add ingredient" button.

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	http://127.0.0.1:4040
Forwarding	http://fd288c990i.ngrok.io -> localhost:5000
Forwarding	https://fd288c990i.ngrok.io -> 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)
```

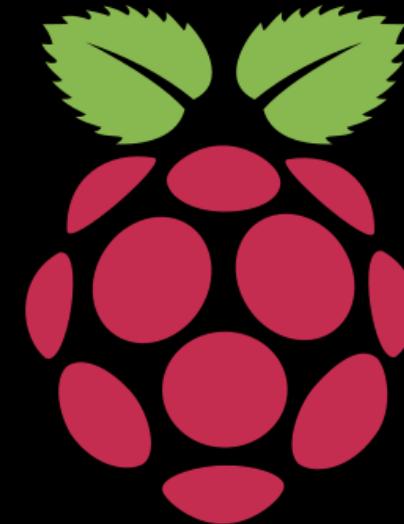
Configuração do Servidor com Página para Previsão do Tempo



Previsão de Tempo no LCD

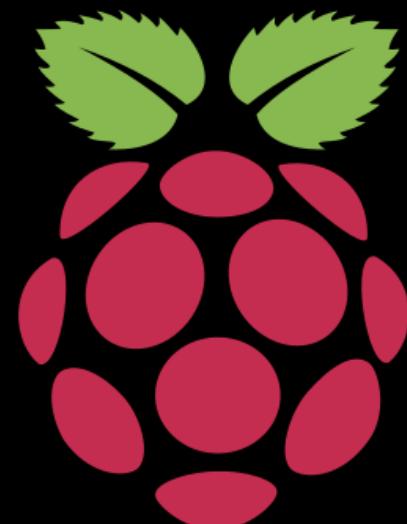
Uber

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/ 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
```

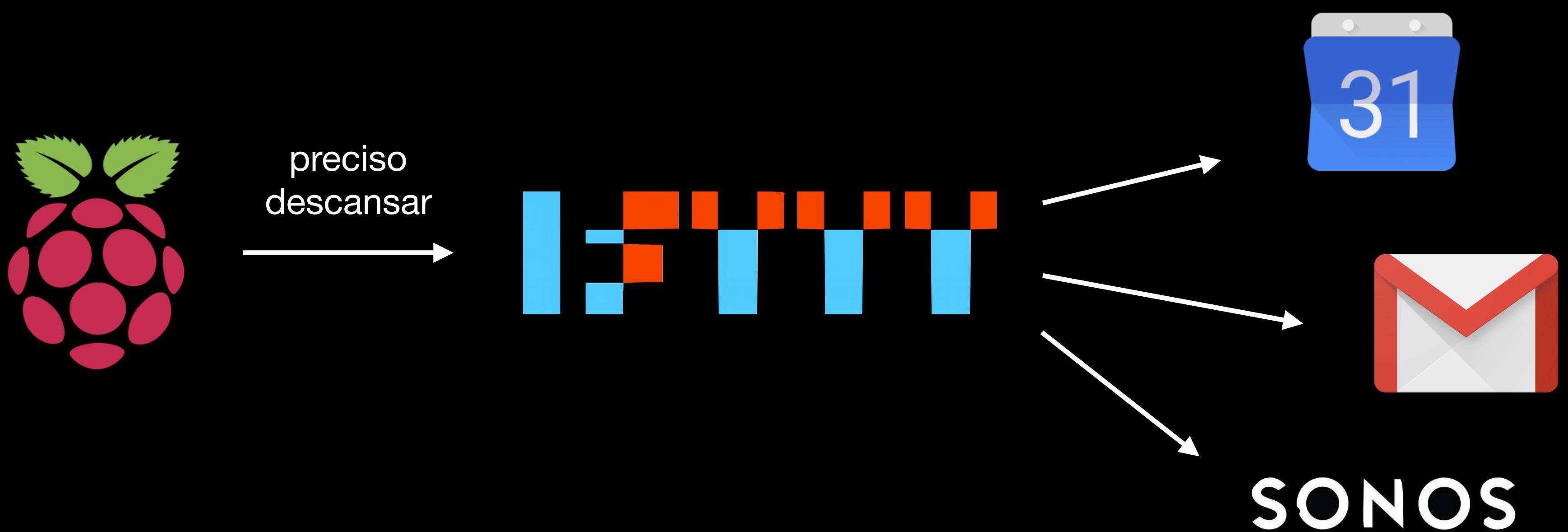
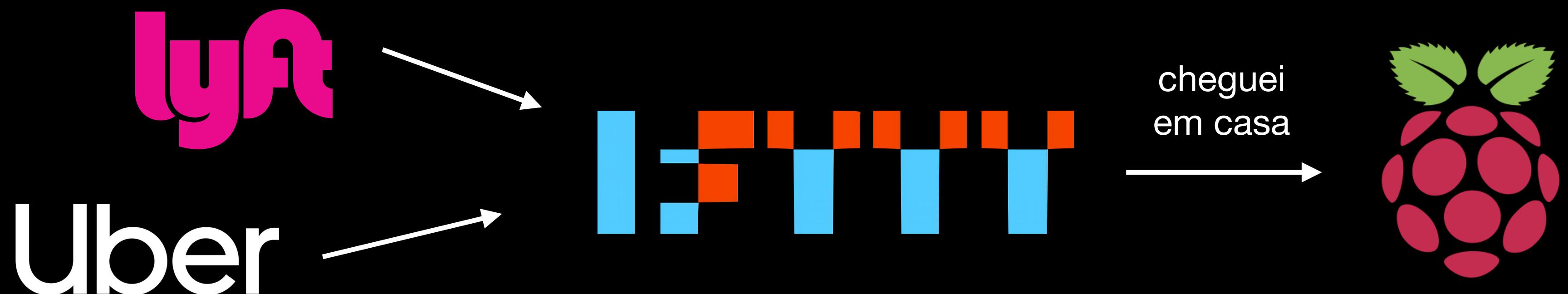
Função para Buscar Dados sobre as Viagens

```
from requests import post

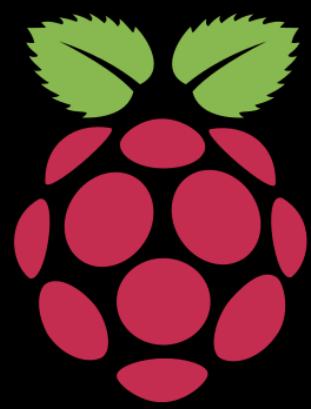
chave = "COLOQUE SUA CHAVE AQUI"
evento = "preciso_descansar"
endereco = "https://maker.ifttt.com/trigger/" + evento
+ "/with/key/" + chave

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

Função para Enviar Pedido ao IFTTT



Novas Integrações Aproveitando o Mesmo Código

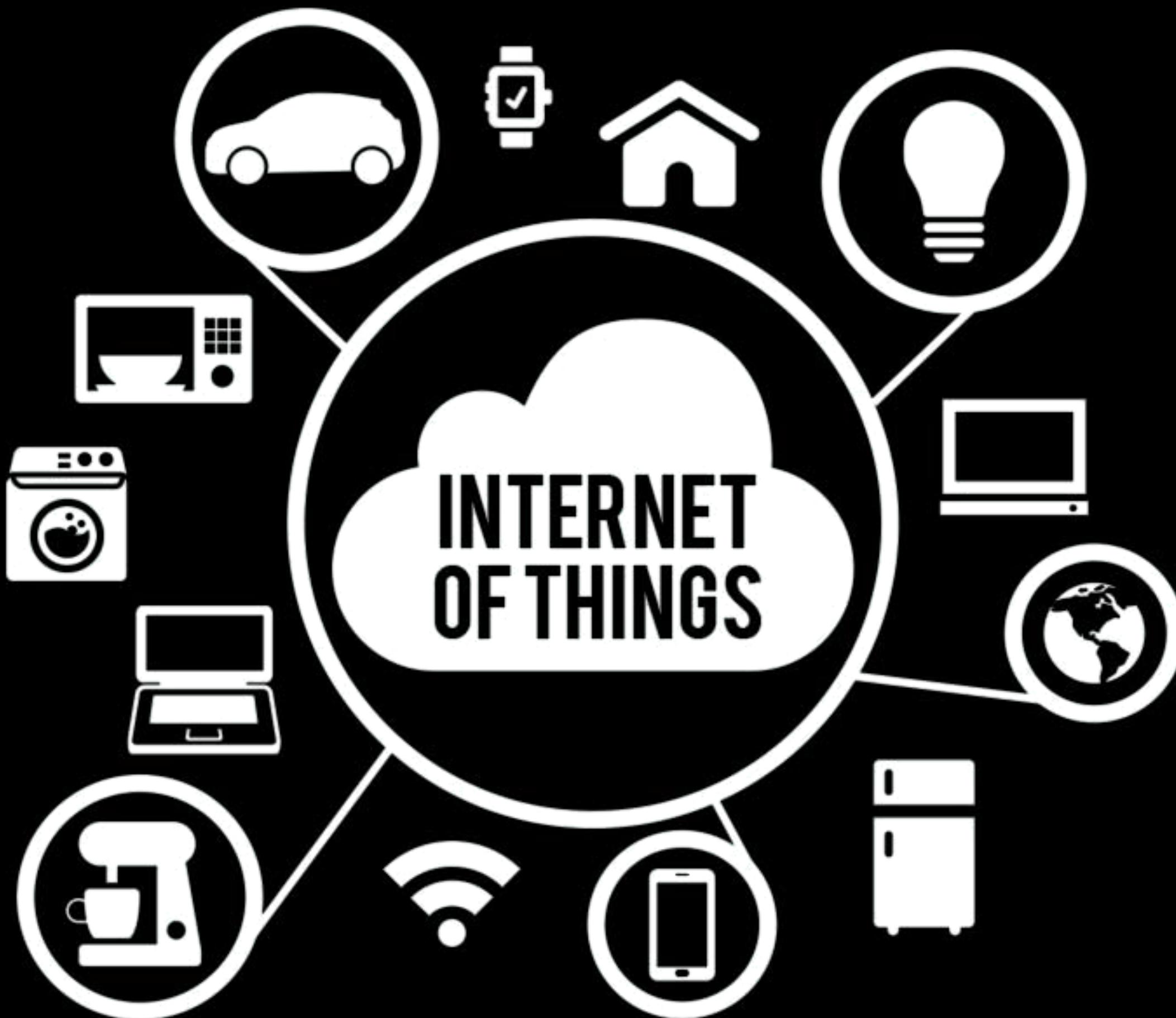


led.on()

↓
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](#)

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 --skip 10 foto.jpg")
```

```
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()
```

```
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!"}  
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=data, files=arquivo)  
dados = {"offset": proximo_id_de_update}  
resposta = get(base + "/getUpdates", json=dados)
```

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}}
documento = colecao.find_one(busca)
ordenacao = [ ["idade", DESCENDING] ]
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 acessar documentação	<pre>lista = [1, 2, 3] • lista2 = ["texto", [0, 0], 5] lista[0] • total_de_elementos = len(lista) lista.append(novo_elemento) • lista.remove(indice)</pre>
Dicionários acessar documentação	<pre>dicionario = {"chave 1": 42, "chave 2": [1, 2, 3]} dicionario["chave 1"] • dicionario["chave 3"] = "Olá!"</pre>
Textos (Strings) acessar documentação	<pre>texto = "olá!" • len(texto) • caractere = texto[0] texto + "\n" • texto + str(numero) • "x = %.2f" % numero</pre>
Condicionais	<pre>if x != 0: if x not in [1, 2]: y = 4 elif x >= 0: y = 3 else: y = 0</pre>
Repetições	<pre>for elem in lista: ... for i in range(1, 4): ... while x > 1: ... def funcao1(x): return x + 2 def funcao2(x, y, z): ... def funcao3(): global x</pre>
Criação de Funções	