

Projeto 04

Controle por Notificação – Teoria

Jan K. S. – janks@puc-rio.br

ENG1419 – Programação de Microcontroladores

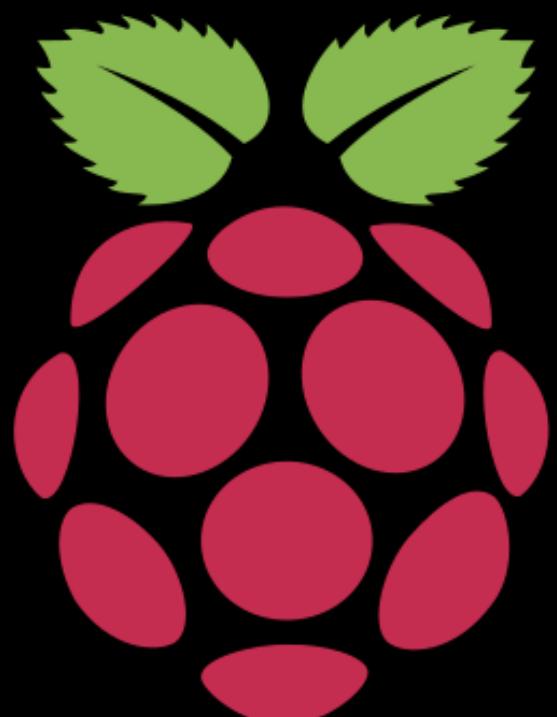
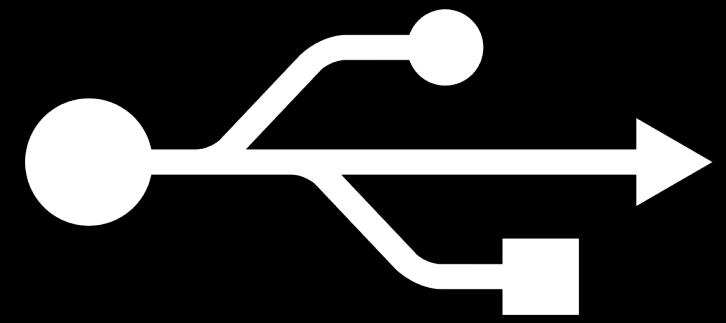
Hardware



Webcam



USB



Comunicação via USB com o Sistema Operacional do Raspberry



Envio de Comandos para Programas que Controlam a Webcam

The screenshot shows a web browser window with the URL `man.cx/fswebcam` in the address bar. The page content is a manpage for `fswebcam`. On the left, there's a sidebar with a "Available in" section containing a link to "(1)". Below it is a "Contents" section with links to various sections: NAME, SYNOPSIS, DESCRIPTION, CONFIGURATION, SIGNALS, KNOWN BUGS, REPORTING BUGS, SEE ALSO, AUTHOR, and COMMENTS. The main content area starts with a "NAME" section, followed by a "SYNOPSIS" section containing the command `fswebcam [<options>] <filename> [<options>] <filename> ...]`. The "DESCRIPTION" section explains that `fswebcam` is a small and simple webcam app for *nix, capable of capturing images from multiple sources and performing simple manipulation. It notes that output can be sent to stdio using "-". The "CONFIGURATION" section includes a "Configuration File" subsection about config files and comments, and a "General Options" subsection with entries for `-?`, `--help`, `-c`, and `--config`. The `-c` entry describes loading options from a file.

Available in

(1)

Contents

[NAME](#)
[SYNOPSIS](#)
[DESCRIPTION](#)
[CONFIGURATION](#)
[SIGNALS](#)
[KNOWN BUGS](#)
[REPORTING BUGS](#)
[SEE ALSO](#)
[AUTHOR](#)
[COMMENTS](#)

NAME

`fswebcam – Small and simple webcam for *nix.`

SYNOPSIS

`fswebcam [<options>] <filename> [<options>] <filename> ...]`

DESCRIPTION

`fswebcam` is a small and simple webcam app for *nix. It can capture images from a number of different sources and perform simple manipulation on the captured image. The image can be saved as one or more PNG or JPEG files. The PNG or JPEG image can be sent to stdio using the filename "-". The output filename is formatted by `strftime`.

CONFIGURATION

Configuration File
Config files use the long version of options without the "--" prefix.
Comments start with a # symbol at the beginning of the line.

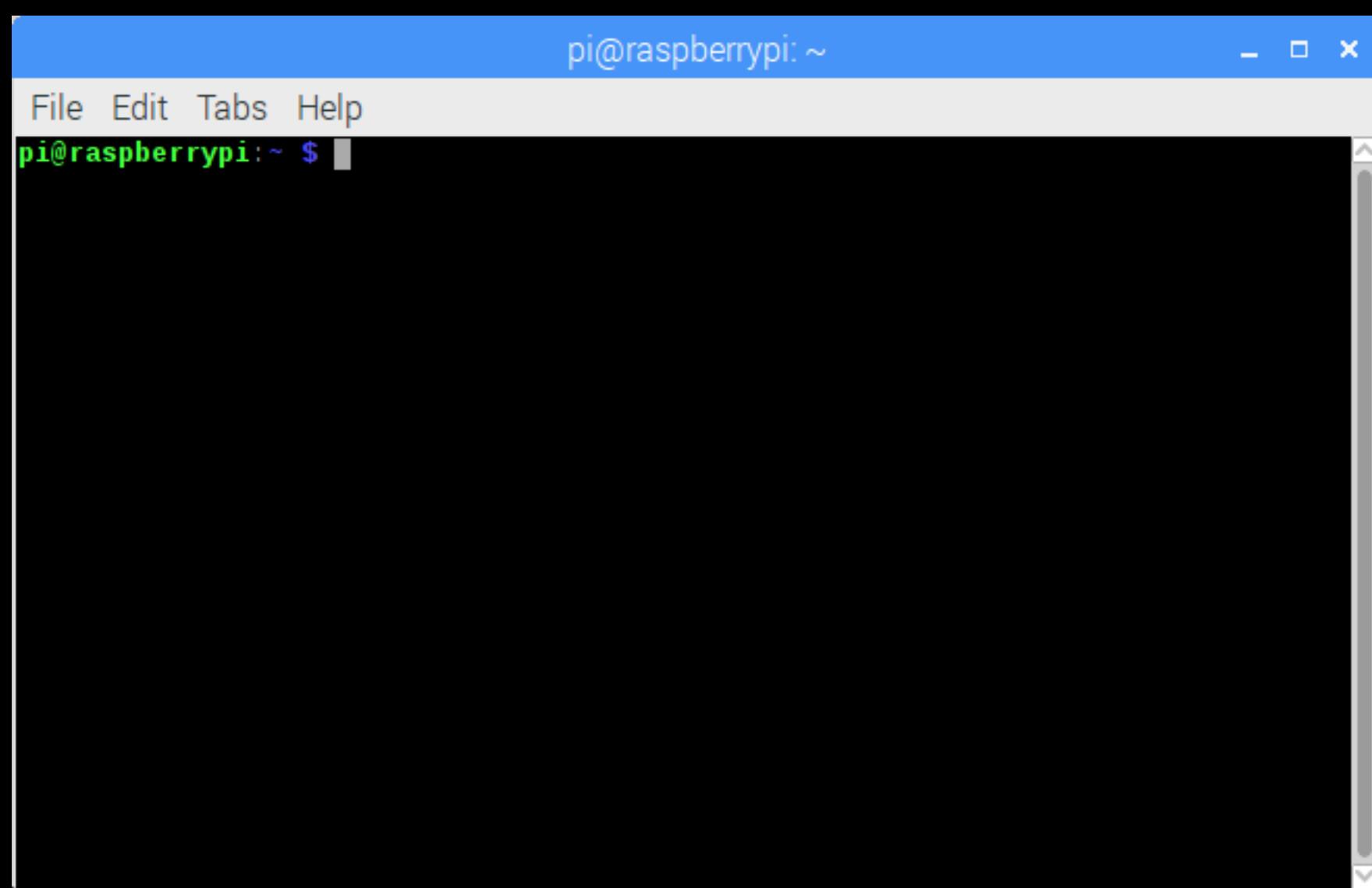
General Options

`-?`, `--help`

Show a usage summary.

`-c`, `--config`

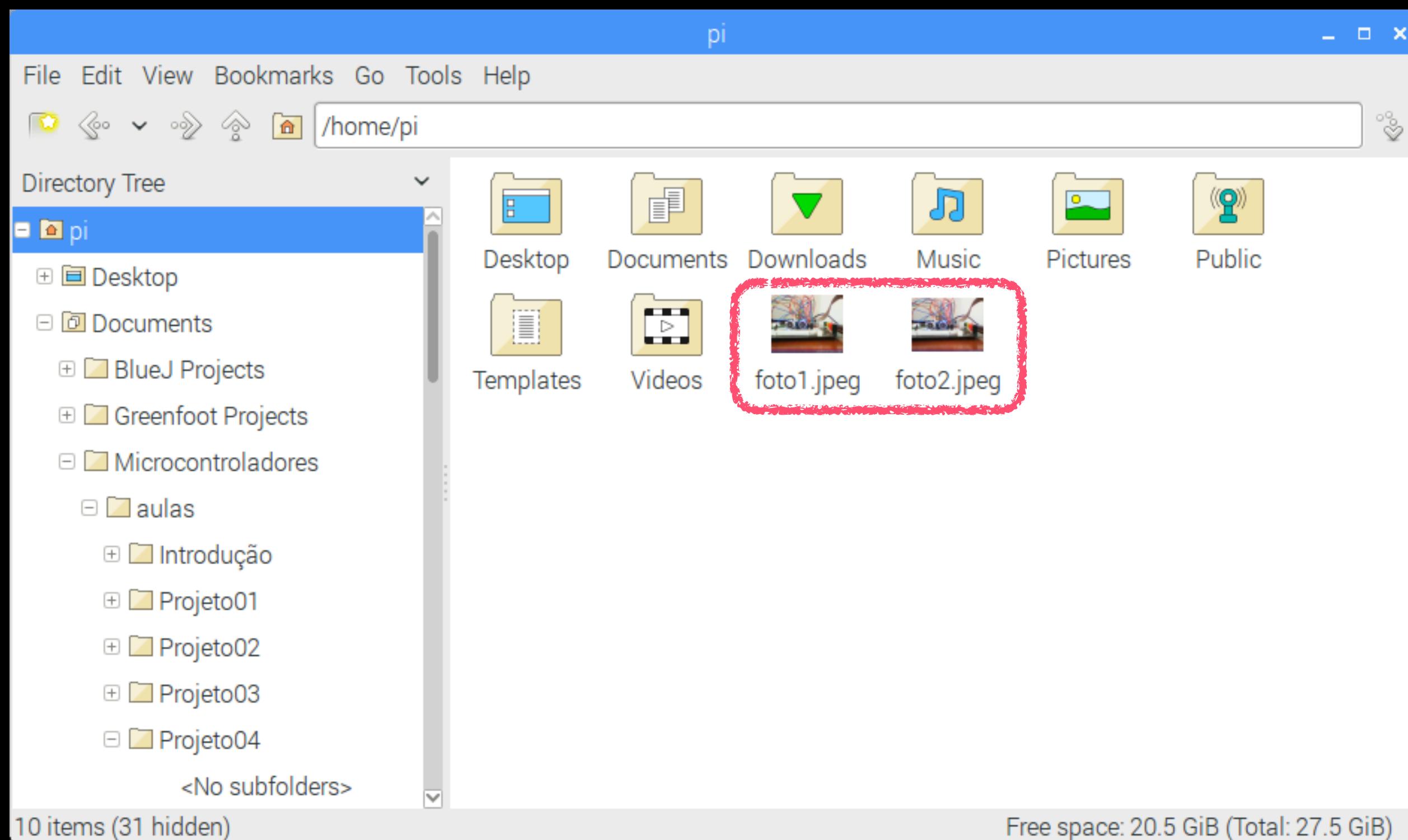
Load options from a file. You can load more than one config file, and can mix them with command-line arguments.



Aplicativo Terminal

```
aula@raspberrypi ~ $ fswebcam foto1.jpg
--- 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.jpg'.
```

```
aula@raspberrypi ~ $ fswebcam --resolution 640x480 foto2.jpg
--- 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.jpg'.
```



Arquivos Gerados no File Manager

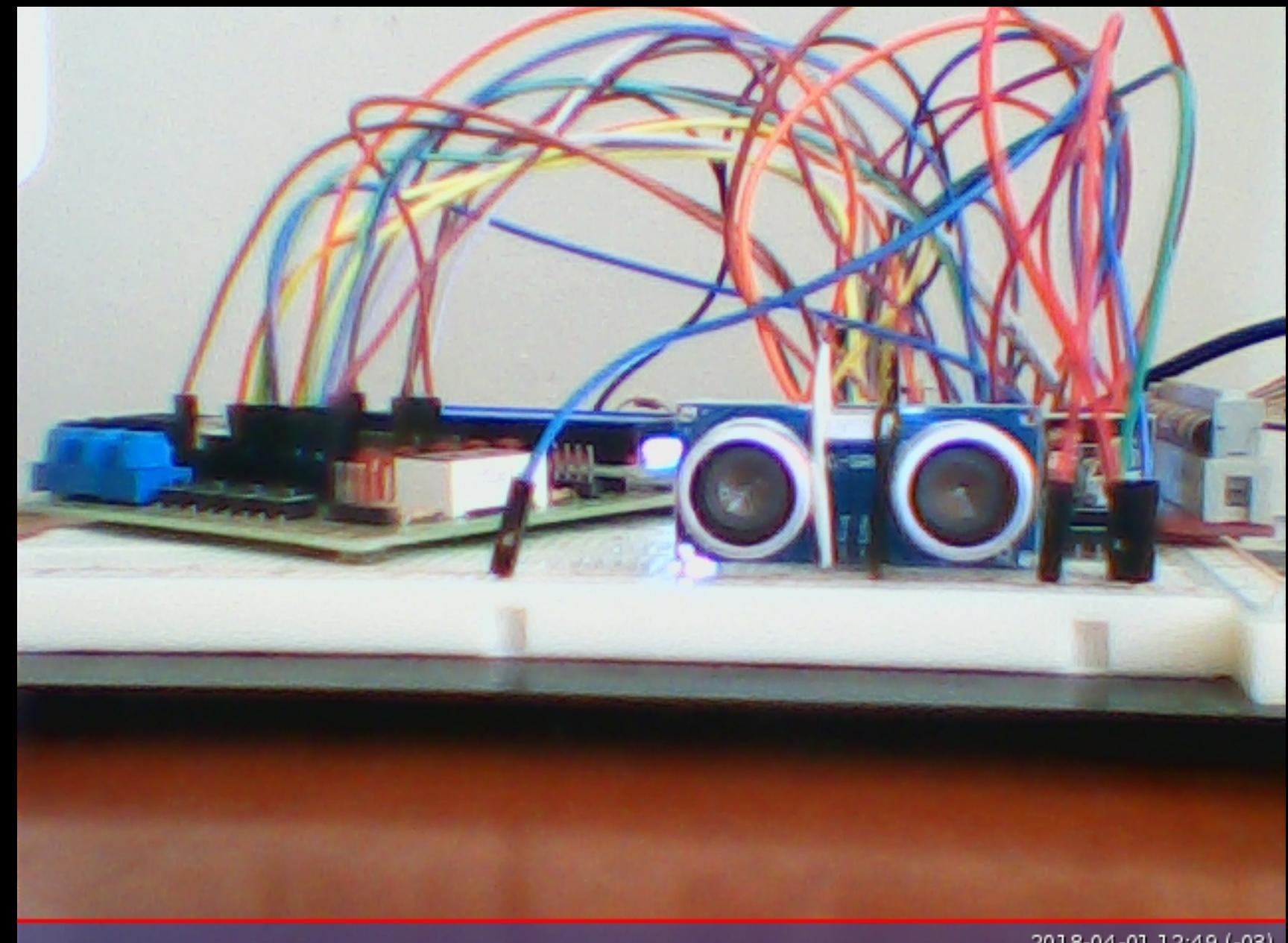
foto1.jpg

resolução padrão (352 × 288)

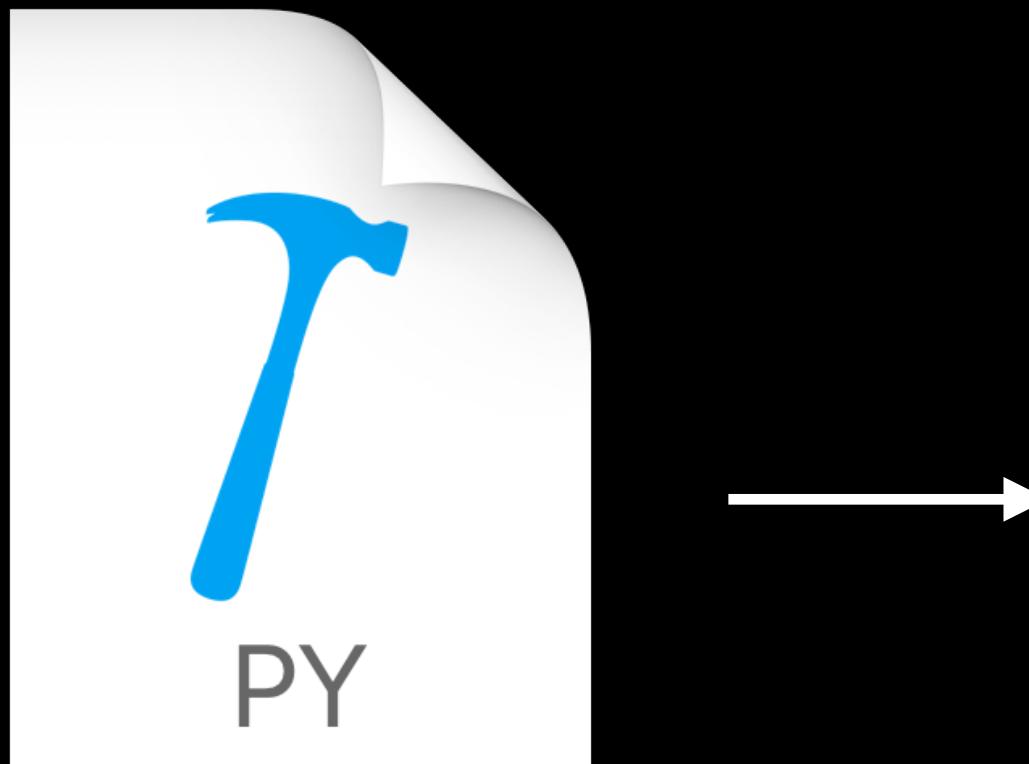


foto2.jpg

resolução 640x480



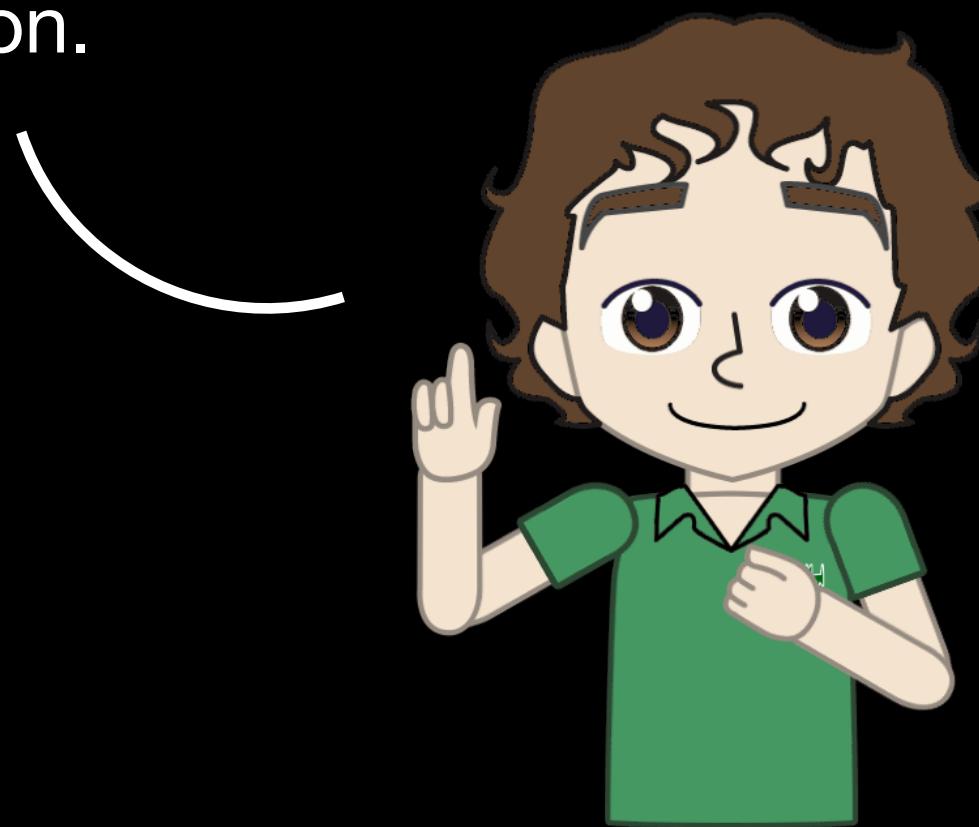
Comparação entre Diferentes Resoluções



```
pi@raspberrypi: ~
File Edit Tabs Help
pi@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'.
pi@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'.
pi@raspberrypi: ~ $
```

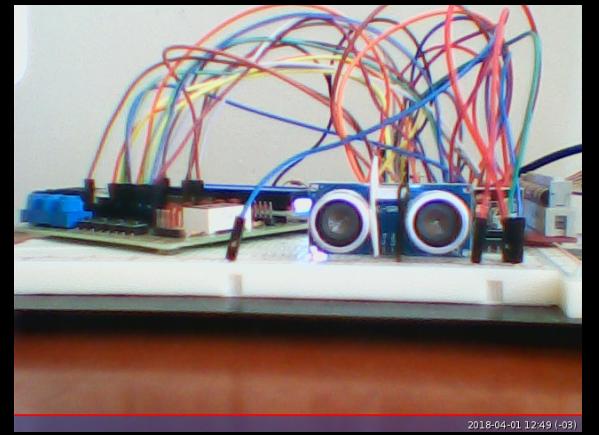
Chamada de Comandos do Terminal em Python

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



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

Chamada de Comandos para Tirar uma Foto



```
>>> from os import system  
>>> from datetime import datetime  
>>> agora = datetime.now()  
>>> hora = agora.strftime("%H:%M:%S")  
>>> nome_do_arquivo = "foto_" + hora + ".jpg"  
>>> comando = "fswebcam " + nome_do_arquivo  
>>> comando  
'fswebcam foto-15:25:37.jpg'  
>>> system(comando)
```



=



+



Microfone da Câmera Web

The screenshot shows a web browser window with the URL `man.cx/arecord` in the address bar. The page content is for the `arecord` manpage. On the left, there is a sidebar with links to other sections: Available in (1), Contents, NAME, SYNOPSIS, DESCRIPTION, OPTIONS, SIGNALS, EXAMPLES, SEE ALSO, BUGS, AUTHOR, and COMMENTS. The main content area has sections for NAME, SYNOPSIS, DESCRIPTION, OPTIONS, and a detailed description of the `arecord` command.

Available in

(1)

Contents

NAME

`arecord`, `aplay` – command-line sound recorder and player for ALSA soundcard driver

SYNOPSIS

`arecord` [*flags*] [*filename*]
`aplay` [*flags*] [*filename*] [*filename*] ...

DESCRIPTION

`arecord` is a command-line soundfile recorder for the ALSA soundcard driver. It supports several file formats and multiple soundcards with multiple devices. If recording with interleaved mode samples the file is automatically split before the 2GB filesize.

`aplay` is much the same, only it plays instead of recording. For supported soundfile formats, the sampling rate, bit depth, and so forth can be automatically determined from the soundfile header.

If filename is not specified, the standard output or input is used. The `aplay` utility accepts multiple filenames.

OPTIONS

`-h`, `--help`
Help: show syntax.
`--version`
Print current version.



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



iniciar!

...

encerrar!



Interrupção da Gravação de Áudio

```
>>> from subprocess import Popen  
>>> global aplicativo  
>>> aplicativo = None  
>>> def iniciar_gravacao():  
...     global aplicativo  
...     # limite máximo de 30 segundos, caso não parem a gravação  
...     comando = ["arecord", "--duration", "30", "audio.wav"]  
...     aplicativo = Popen(comando) # executa em plano de fundo  
...  
>>> def parar_gravacao():  
...     global aplicativo  
...     if aplicativo != None:  
...         aplicativo.terminate()  
...     aplicativo = None  
...  
>>> iniciar_gravacao()  
>>> # faz outras coisas no código...  
>>> parar_gravacao() →  audio.wav
```

Interrupção da Gravação de Áudio



audio.wav

516 kB



audio.mp3

47.8 kB



audio.ogg

28.9 kB

Conversão do Formato WAV para MP3 e para OGG

Manpages

Manpage: go

NAME	Available in
lame – create mp3 audio files	(1)
SYNOPSIS	Contents
lame [options] <infile> <outfile>	NAME
DESCRIPTION	SYNOPSIS
LAME is a program which can be used to create compressed audio files. (Lame ain't an MP3 encoder). These audio files can be played back by popular MP3 players such as mpg123 or madplay. To read from stdin, use "--" for <infile>. To write to stdout, use "--" for <outfile>.	DESCRIPTION
OPTIONS	OPTIONS
Input options:	ID3 TAGS
r Assume the input file is raw pcm. Sampling rate and mono/stereo/jstereo must be specified on the command line. For each stereo sample, LAME expects the input data to be ordered left channel first, then right channel. By default, LAME expects them to be signed integers with a bitwidth of 16 and stored in little-endian. Without r , LAME will perform several <code>fseek()</code> 's on the input file looking for WAV and AIFF headers.	ENCODING MODES
Might not be available on your release.	PRESETS
x Swap bytes in the input file (or output file when using –decode).	EXAMPLES
	BUGS
	SEE ALSO
	AUTHORS
	COMMENTS

Manpages

Manpage: lame go

Available in	NAME
(1)	opusenc – encode audio into the Opus format
Contents	SYNOPSIS
NAME	opusenc [-h] [-V] [--help-picture] [--quiet] [--bitrate kbit/sec] [--vbr] [--cvbr] [--hard-cbr] [--comp complexity] [--framesize 2.5, 5, 10, 20, 40, 60] [--expect-loss pct] [--downmix-mono] [--downmix-stereo] [--max-delay ms] [--title 'track title'] [--artist author] [--album 'album title'] [--genre genre] [--date YYYY-MM-DD] [--comment tag=value] [--picture filenam specification] [--padding n] [--discard-comments] [--discard-pictures] [--raw] [--raw-bits bits/sample] [--raw-rate Hz] [--raw-chan N] [--raw-endianness flag] [--ignorelength] [--serial serial number] [--save-range file] [--set-ctl-int ctl=value] input.wav output.opus
SYNOPSIS	DESCRIPTION
DESCRIPTION	opusenc reads audio data in Wave, AIFF, FLAC, Ogg/FLAC, or raw PCM format and encodes it into an Ogg Opus stream. If the input file is "-" audio data is read from stdin. Likewise, if the output file is "-" the Ogg Opus stream is written to stdout.
OPTIONS	Unless quieted opusenc displays fancy statistics about the encoding progress.
EXAMPLES	OPTIONS
BUGS	General options
SEE ALSO	-h, --help
AUTHORS	Show command help
COMMENTS	



audio.wav



audio.mp3



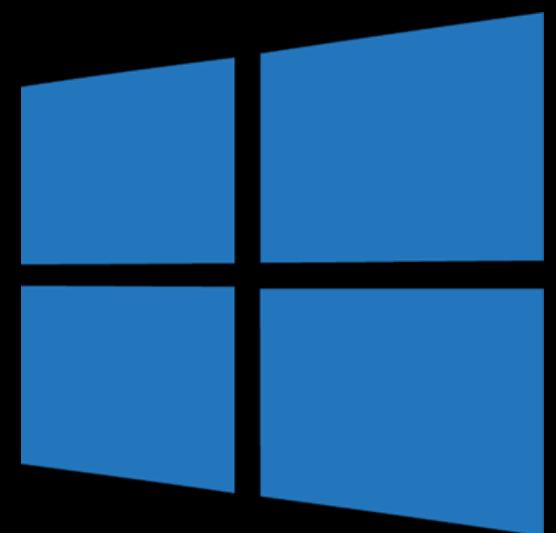
audio.ogg

```
>>> 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



```
fswebcam foto1.jpg  
arecord --duration 3 audio1.wav  
opusenc audio.wav audio.ogg  
lame audio.wav audio.mp3
```

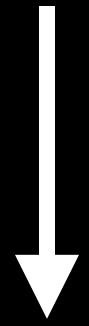


?

Alternativas para Windows e Mac?

LINUX

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



WINDOWS

```
system("CommandCam /filename foto.jpg /delay 500")
```

MAC

```
system("ffmpeg -y -pix_fmt uyvy422 -ss 0.5 -f avfoundation -r 30 -i 0  
foto.jpg")
```

provavelmente O ou I



LINUX

```
system("arecord --duration 3 audio.wav")
```



WINDOWS

```
system('ffmpeg -y -f dshow -i audio="COLOQUE AQUI O SEU  
MICROFONE" -t 00:03 audio.wav')
```

MAC

```
system("ffmpeg -y -f avfoundation -i :1 -t 3 audio.wav")
```

provavelmente :0 ou :1



LINUX

```
comando = ["arecord", "--duration", "30", "audio.wav"]  
aplicativo = Popen(comando)
```



WINDOWS

SEM ASPAS!

```
comando = ["ffmpeg", "-y", "-f", "dshow", "-i", "audio=COLOQUE  
AQUI O SEU MICROFONE", "-t", "00:30", "audio.wav"]
```

MAC

provavelmente :O ou :l

```
comando = ["ffmpeg", "-y", "-f", "avfoundation", "-i", ":1", "-t", "30",  
"audio.wav"]
```

LINUX

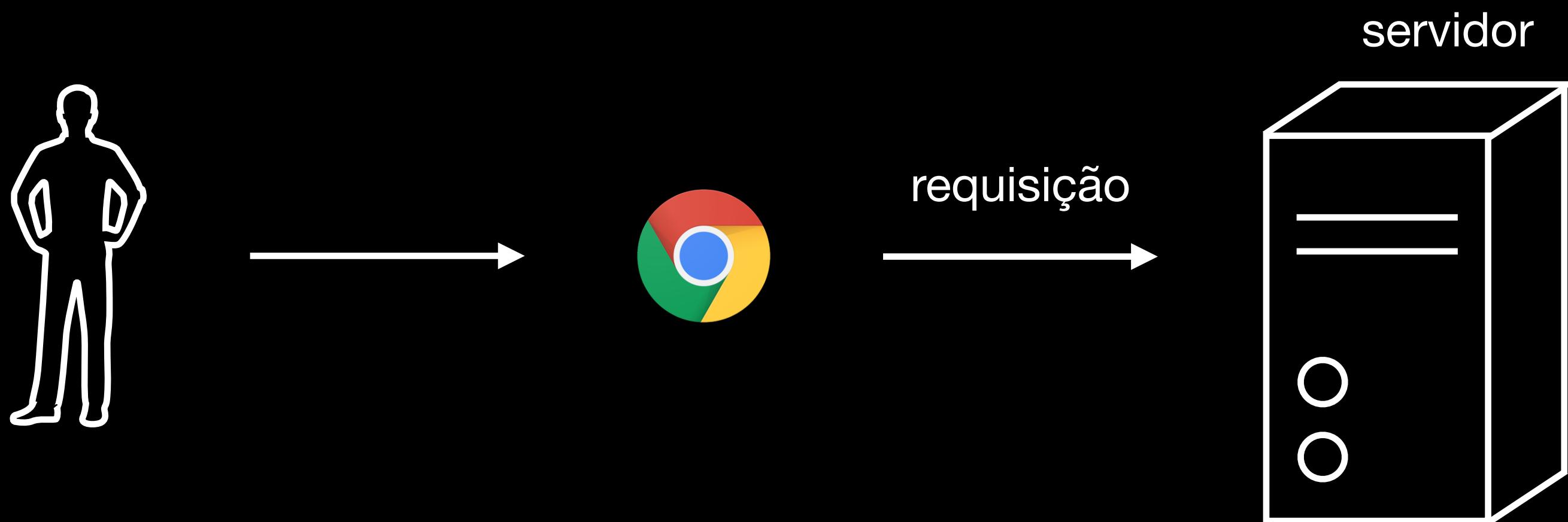
```
system("lame audio.wav audio.mp3")
system("opusenc audio.wav audio.ogg")
```



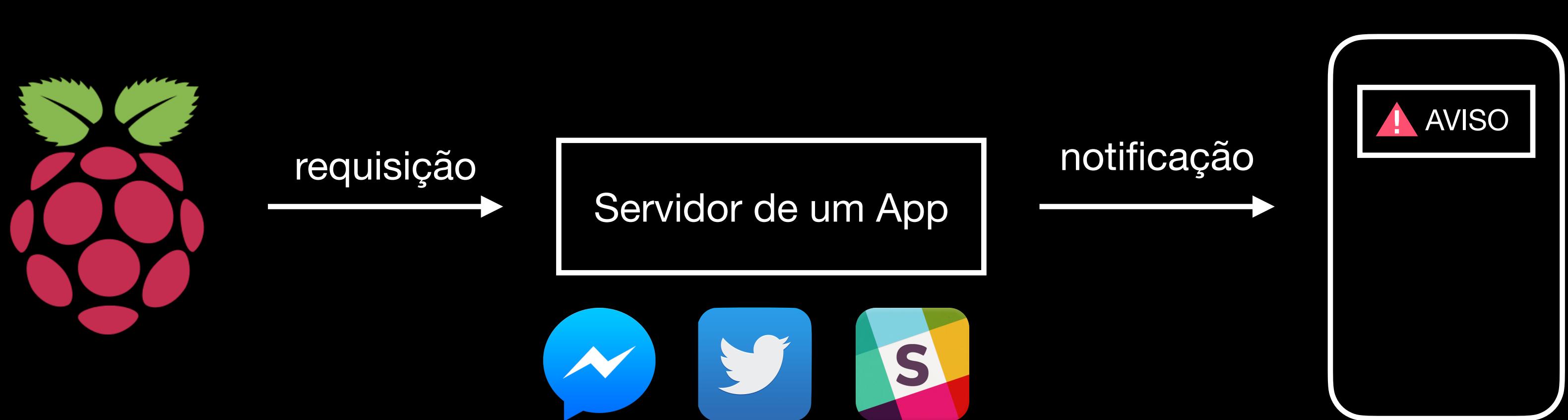
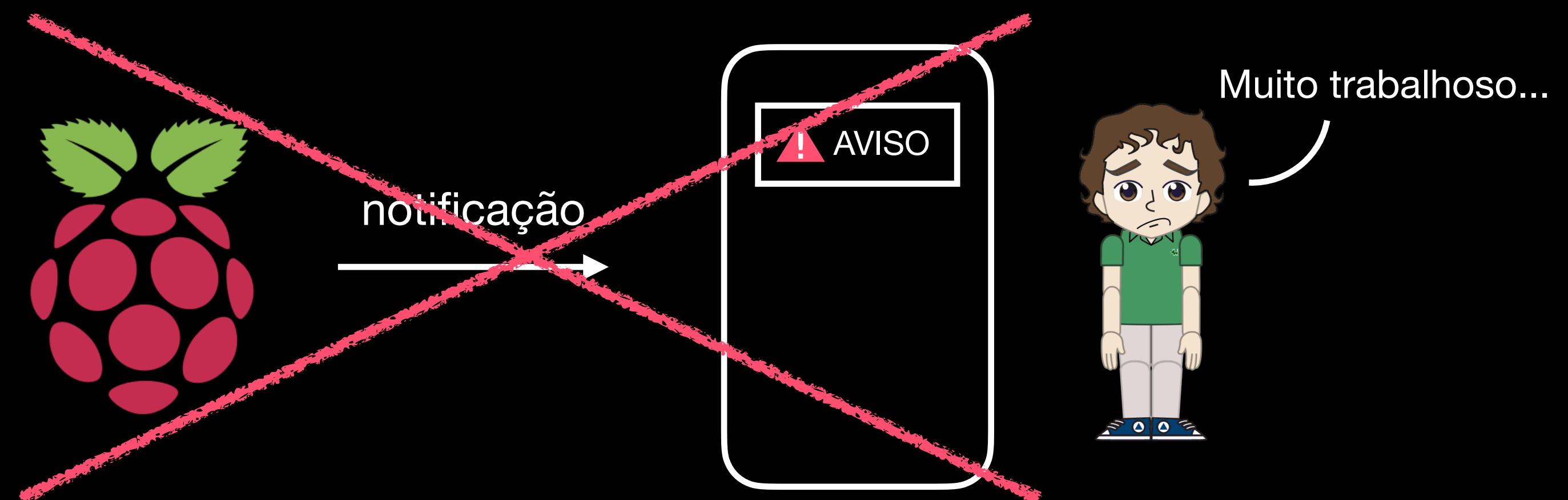
WINDOWS e MAC

```
system("ffmpeg -y -i audio.wav audio.mp3")
system("ffmpeg -y -i audio.wav -acodec libopus audio.ogg")
```

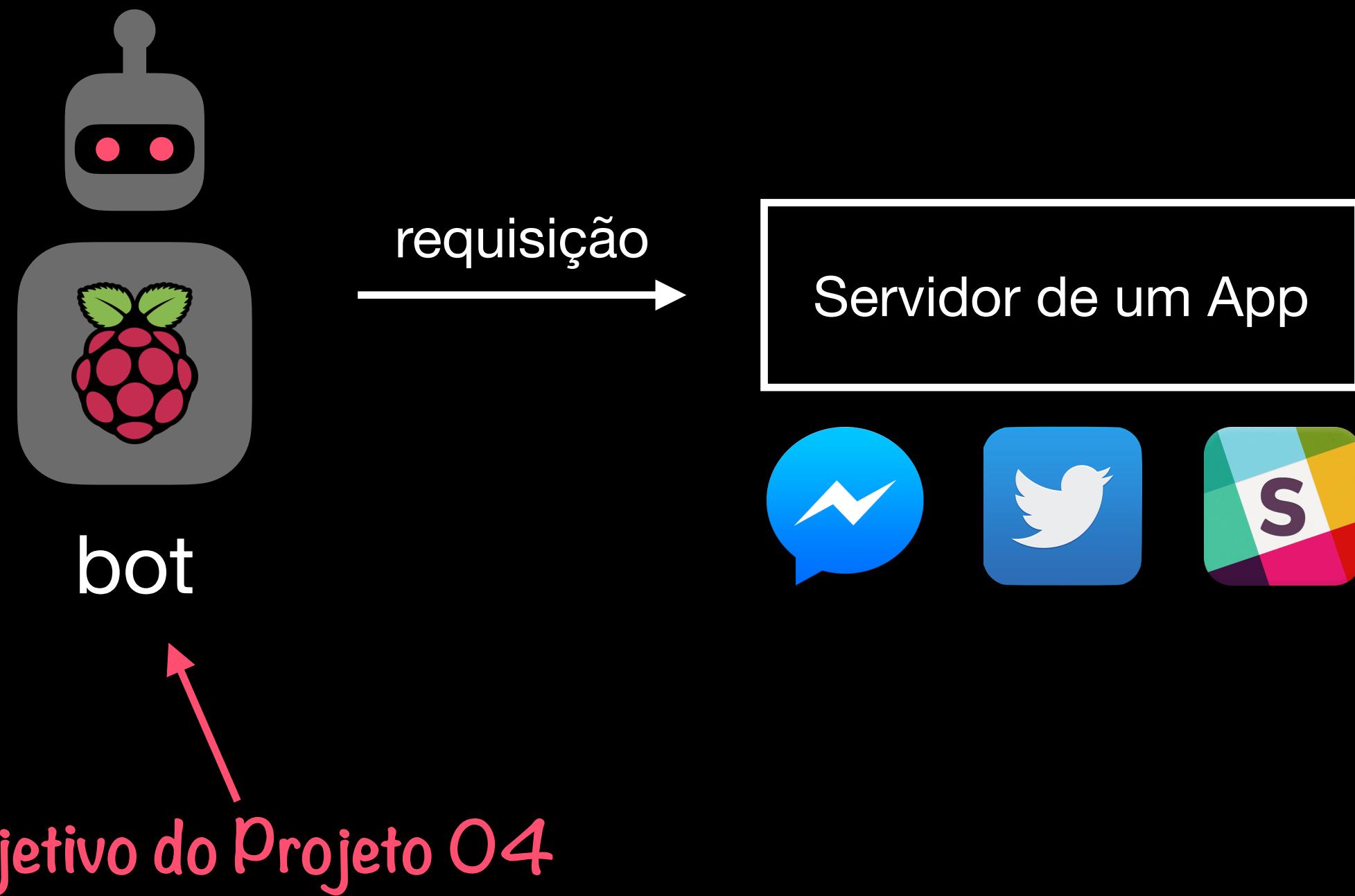
Software



Requisição x Notificação



Notificações com um Servidor Intermediário



Notificações via Bots

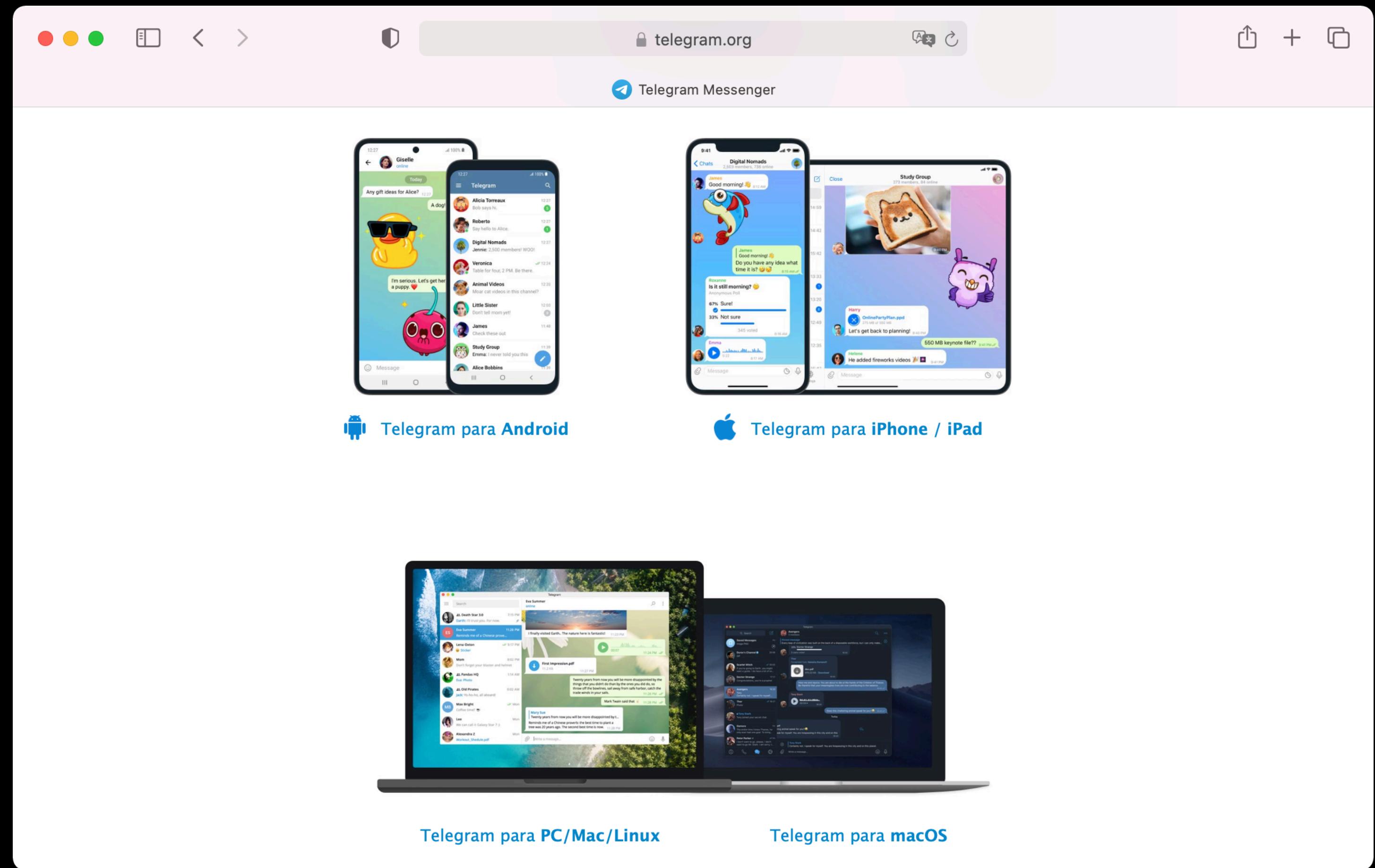
The image shows a Mac desktop with three browser windows open:

- Top Window:** Wikipedia article titled "Twitter bot". The page content describes a Twitter bot as a type of bot software that can perform actions autonomously. It mentions the automation of Twitter accounts and proper usage including replying to users without spamming.
- Middle Window:** developers.facebook.com/docs/messenger-platform. The title is "Plataforma do Messenger". It features a large blue header and a section titled "Introdução: plataforma do Messenger 2.3" with text about improvements to the messaging plugin.
- Bottom Window:** slack.com/apps/category/At0MQP5BEF-bots. The title is "Bots". It lists several Slack app categories: Staff Picks, Analytics, Communication, Customer Support, Design, Developer Tools, File Management, Health & Wellness, HR & Team Culture, Marketing, and Office Management. Under the "Bots" category, there is a list of apps: To-do (Workast), Zapier, Polly, Jira Cloud, Hubot, and busybot.

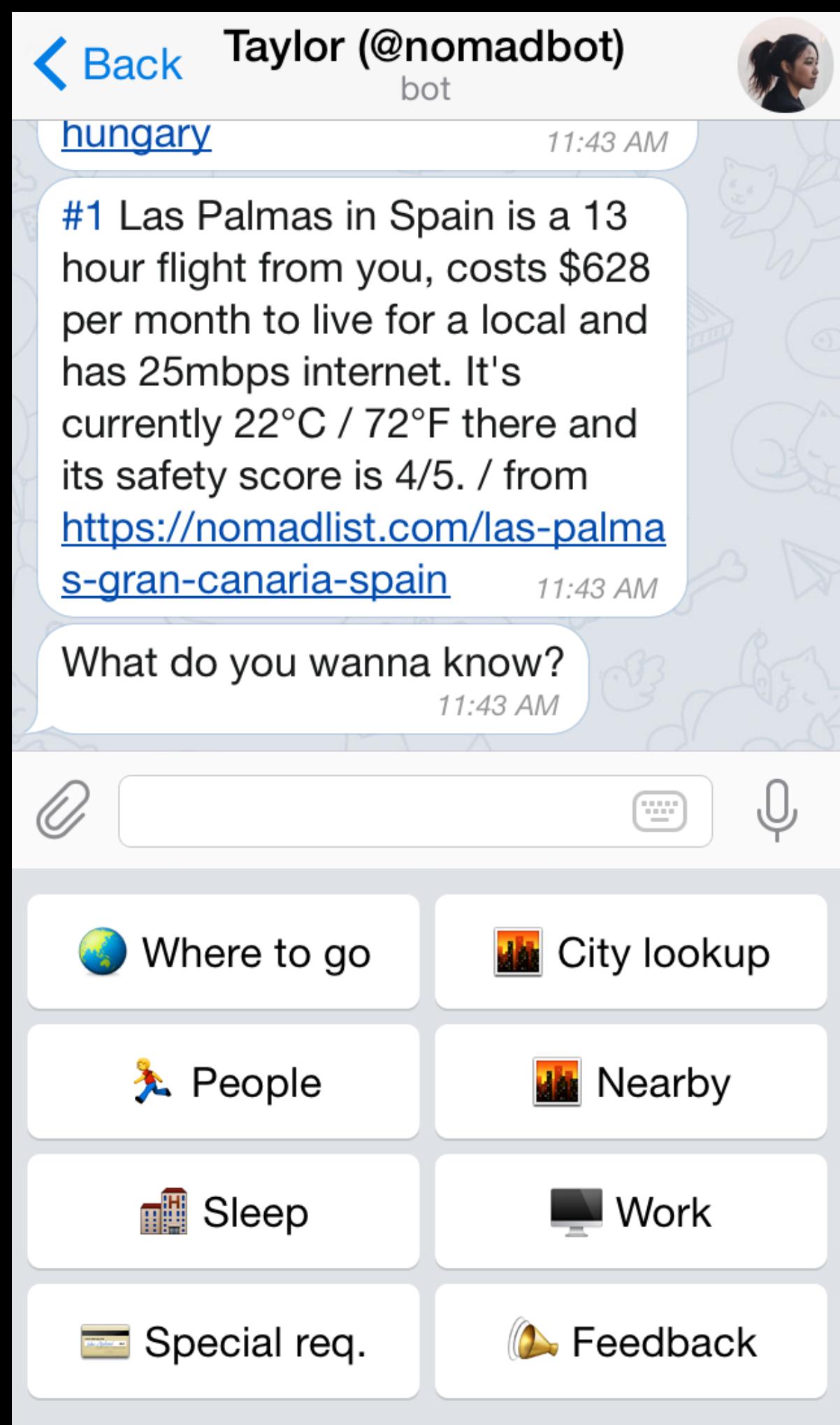
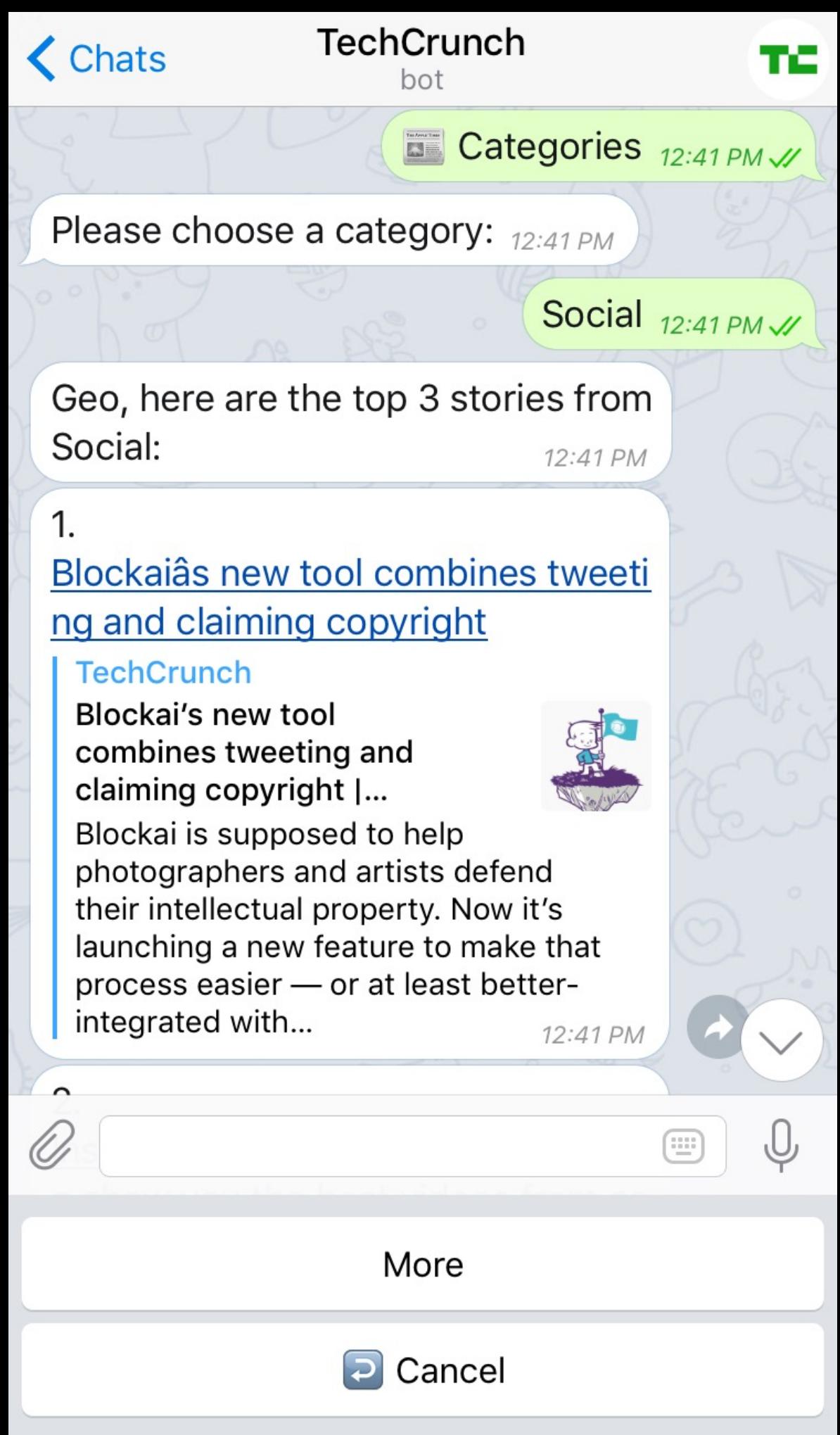
Exemplos de Comunicação entre Bots e Apps



Telegram



Plataformas do Telegram



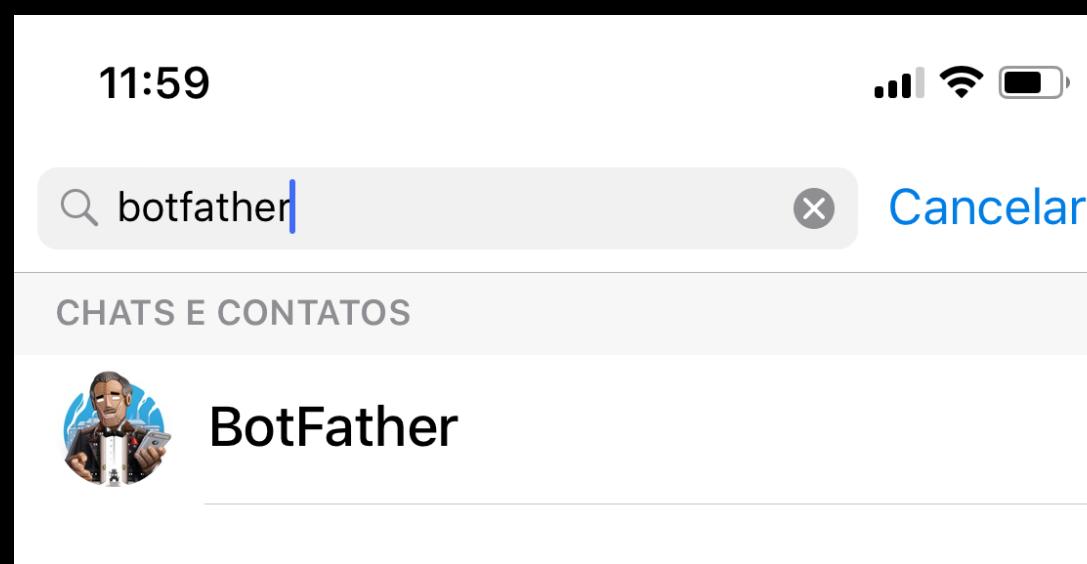
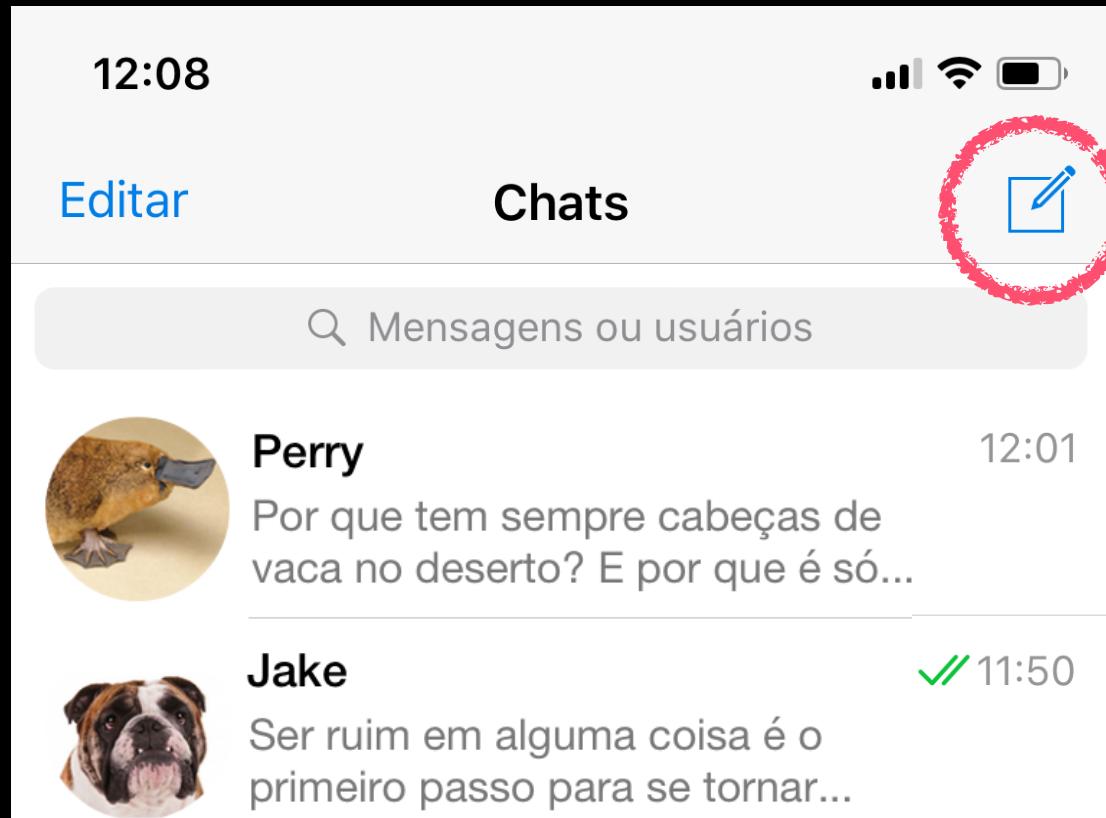
Exemplos de Conversas com Bots



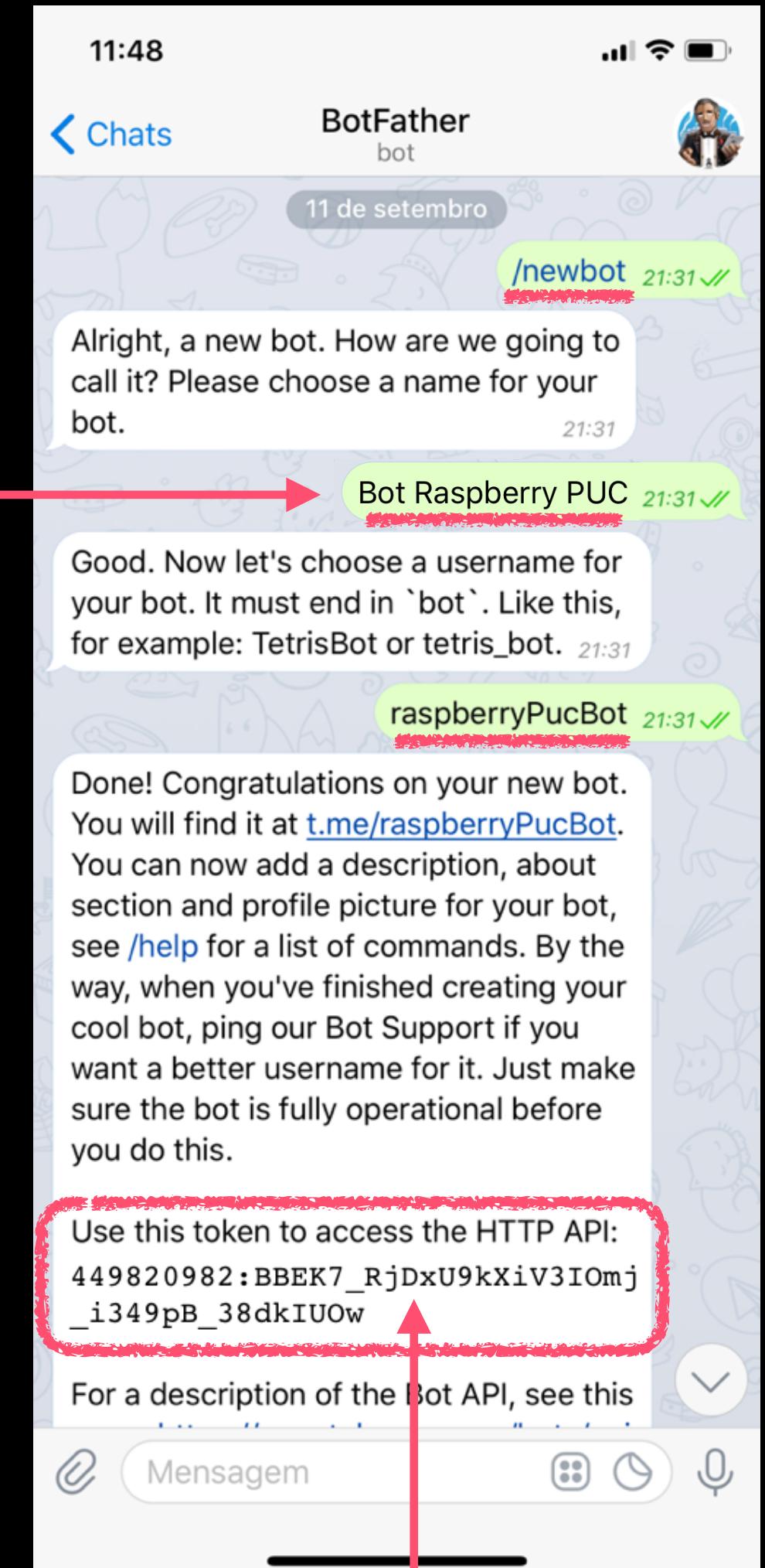
BotFather is the one bot to rule them all.
Use it to create new bot accounts and manage your existing bots.

Criação de Bots no Telegram via BotFather

1



invente um nome seu!

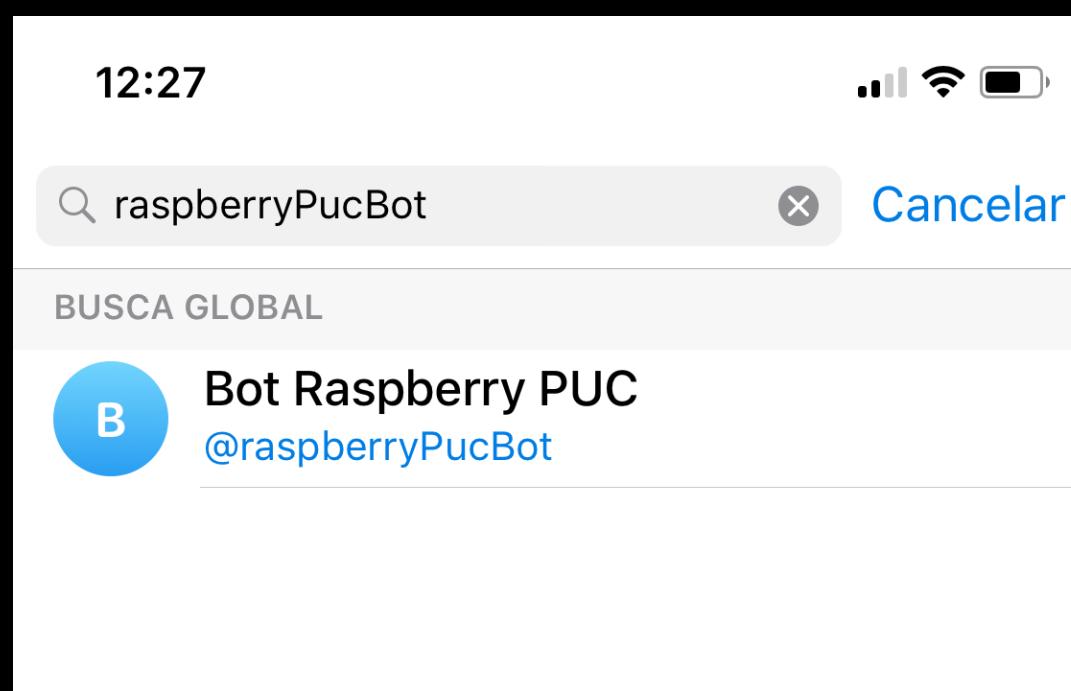


anote esta CHAVE!

Passo 1: Crie o Bot via BotFather e Anote a Chave

2

pesquise o nome
do SEU bot!



Passo 2: Converse com o Seu Bot

3

api.telegram.org/botSUA_CHAVE_SECRETA/getUpdates



```
{"ok":true,"result":[{"update_id":564714809, "message":{"message_id":159,"from":{"id":18594979,"is_bot":false,"first_name":"Jan","last_name":"K.S.","username":"wallysalami","language_code":"pt-br"}, "chat":{"id":18594979,"first_name":"Jan","last_name":"K.S.","username":"wallysalami","type":"private"}, "date":1522423946, "text":"/start", "entities":[{"offset":0,"length":6,"type":"bot_command"}]}}, {"update_id":564714810, "message":{"message_id":160,"from":{"id":18594979,"is_bot":false,"first_name":"Jan","last_name":"K.S.","username":"wallysalami","language_code":"pt-br"}, "chat":{"id":18594979,"first_name":"Jan","last_name":"K.S., "username":"wallysalami","type":"private"}, "date":1522423957, "text":"Ol\u00e1, bot! Tudo bem?"}}, {"update_id":564714811, "message":{"message_id":161,"from":{"id":18594979,"is_bot":false,"first_name":"Jan","last_name":"K.S., "username":"wallysalami","language_code":"pt-br"}, "chat":{"id":18594979,"first_name":"Jan","last_name":"K.S., "username":"wallysalami","type":"private"}, "date":1522423983, "text":"Al\u00f4????"}, {"update_id":564714812, "message":{"message_id":162,"from":{"id":18594979,"is_bot":false,"first_name":"Jan","last_name":"K.S., "username":"wallysalami","language_code":"pt-br"}, "chat":{"id":18594979,"first_name":"Jan","last_name":"K.S., "username":"wallysalami","type":"private"}, "date":1522424010, "text":"Voc\u00e1 n\u00e3o \u00e1 de falar muito, pelo visto..."}}]}
```

anote este ID DO CHAT!

Passo 3: Obtenha o Id do Seu Chat com o Bot

The screenshot shows a Mac OS X browser window displaying the Requests library documentation. The title bar reads "docs.python-requests.org". The main content area features a large logo on the left with the text "Requests http for humans". To the right, the title "Requests: HTTP for Humans" is displayed, followed by "Release v2.18.4. (Installation)". Below this are several badge links: "license Apache 2.0", "wheel yes", "python 2.6, 2.7, 3.4, 3.5, 3.6", "codecov 90%", and "Say Thanks! 🐧". A note from Kenneth Reitz encourages upgrading to Python 3. A code block shows a Python script for making a GitHub API request, and a status bar at the bottom right indicates "v: master".

Requests 3.0 development is underway, and your financial help is appreciated!

Fork me on GitHub

Requests: HTTP for Humans

Release v2.18.4. (Installation)

license Apache 2.0 wheel yes python 2.6, 2.7, 3.4, 3.5, 3.6 codecov 90% Say Thanks! 🐧

Requests is the only *Non-GMO* HTTP library for Python, safe for human consumption.

Note:

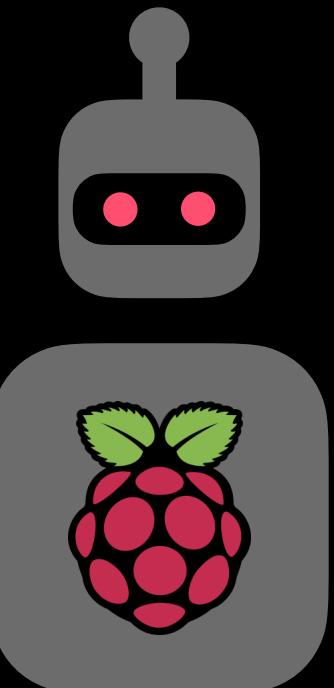
The use of **Python 3** is *highly* preferred over Python 2. Consider upgrading your applications and infrastructure if you find yourself *still* using Python 2 in production today. If you are using Python 3, congratulations — you are indeed a person of excellent taste.
—Kenneth Reitz

Behold, the power of Requests:

```
>>> r = requests.get('https://api.github.com/user', auth=('user', 'pass'))
>>> r.status_code
200
>>> r.headers['content-type']
'application/json; charset=utf8'
>>> r.encoding
'utf-8'
```

v: master ▾

Envie a mensagem "Olá!" para o chat



OK, enviei com sucesso!



Quais são as mensagens novas?

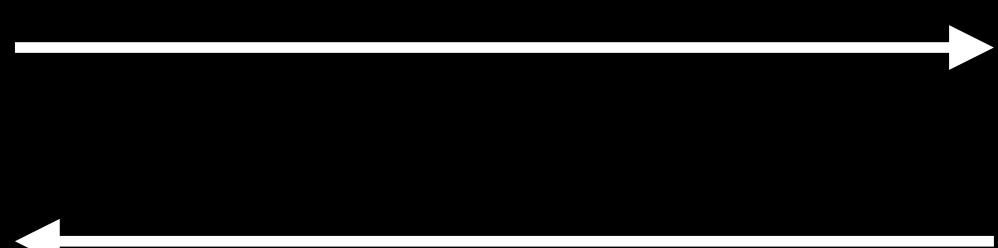


O usuário respondeu "Olá, sumido!"



Requests
http for humans

endereço web do comando
+ dados JSON



resposta JSON

Servidor de um App

Requisição Web a um Servidor

https://api.telegram.org/botSUA_CHAVE_SECRETA/<COMANDO>

+

dados extras

Ação	Comando	Dados Extras
Enviar Texto	sendMessage	id do chat + texto
Enviar Foto	sendPhoto	id do chat + arquivo
Enviar Áudio	sendVoice	id do chat + arquivo
Receber Mensagens	getUpdates	id da atualização
Obter Link do Arquivo	getFile	id do arquivo
Dados do Bot	getMe	(nenhum)
	...	

Exemplos de Comandos para Bots do Telegram

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

```
>>> from requests import post  
>>> endereco = endereco_base + "/sendMessage"  
>>> dados = {"chat_id": id_da_conversa, "text": "Oi!"}  
>>> resposta = post(endereco, json=dados)  
  
>>> resposta.text  
'{"ok":true,"result":{"message_id":291,"from":  
{"id":449820982,"is_bot":true,"first_name":"Bot Raspberry  
PUC","username":"raspberryPucBot"},"chat":  
{"id":18594979,"first_name":"Jan","last_name":"K.  
S.","username":"wallysalami","type":"private"},"date":153  
5815441,"text":"Oi!"}}'
```



Mensagem Enviada pelo Bot na Conversa

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



Foto Enviada pelo Bot na Conversa

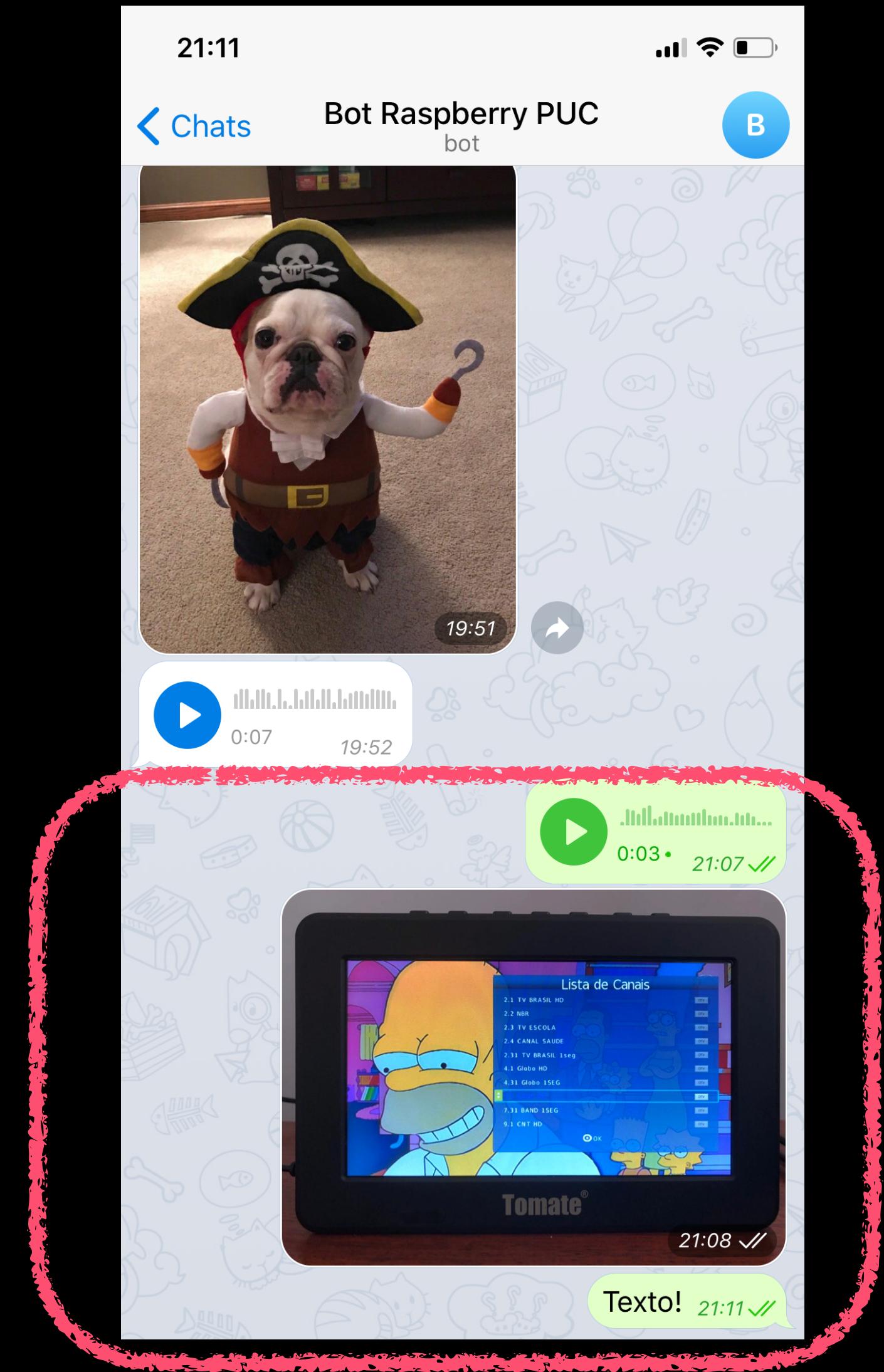
Use sempre o formato OGG para enviar áudio via Telegram.



```
>>> 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)
```



Áudio Enviado pelo Bot na Conversa

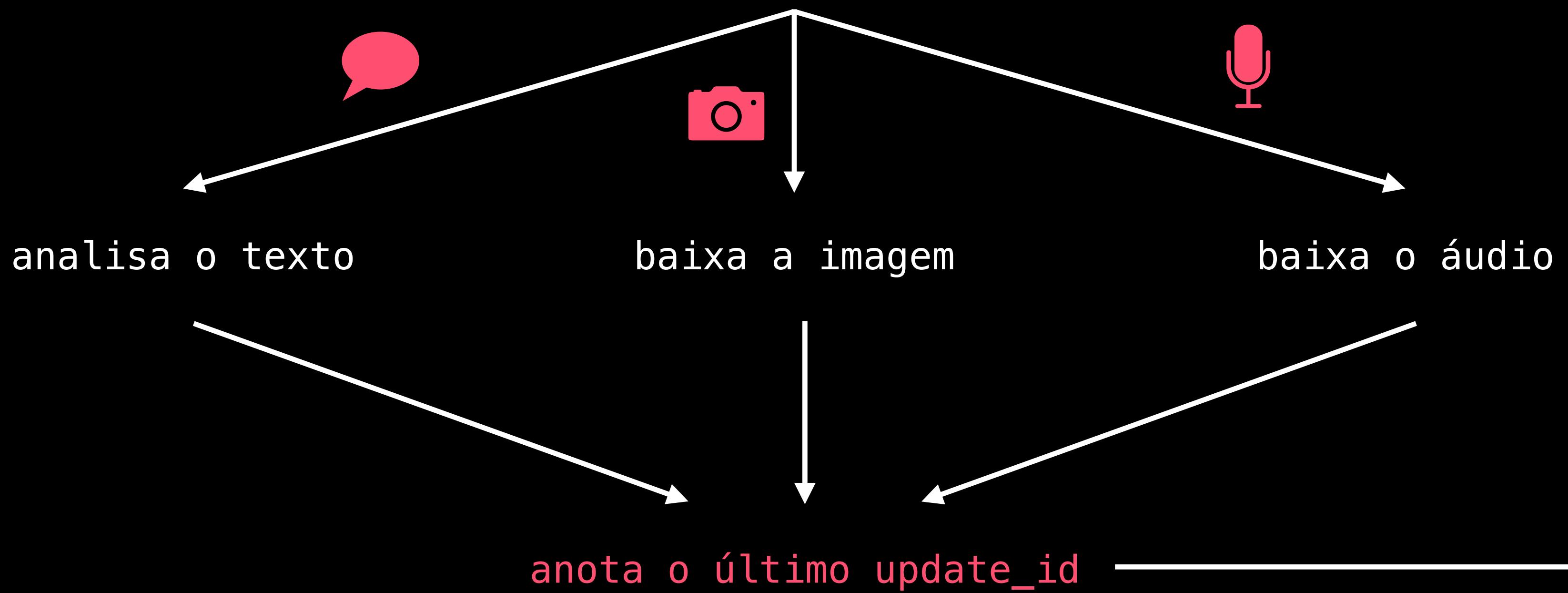


Recebimento de Outros Tipos de Mensagens

busca novas mensagens
(a partir de um certo **update_id**)



para cada nova mensagem:
verifica o **tipo**



/getUpdates



```
{  
    "ok":true,  
    "result": [  
        {  
            "update_id":564714813,  
            "message": {  
                "voice": {  
                    "file_id": "AwADAQADJAADQzTwRQGVIkswNsWHAg",  
                    ...  
                },  
                ...  
            }  
        },  
        {  
            "update_id":564714814,  
            "message": {  
                "photo": [  
                    {  
                        "file_id": "AgADAQAD1qcxGxFkkC7VB-drDDA",  
                        "file_size": 1438,  
                        ...  
                    },  
                    ...  
                ]  
            }  
        },  
        {  
            "update_id":564714815,  
            "message": {  
                "text": "Texto!",  
                ...  
            }  
        }  
    ]  
}
```

Dicionário (JSON) com as Atualizações de Mensagens

{json}

KEEP
CALM
AND
PERCORRE
O JSON

```
>>> 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"]:
...         mensagem = resultado["message"]
...         if "text" in mensagem:
...             texto = mensagem["text"]
...         elif "voice" in mensagem:
...             id_do_arquivo = mensagem["voice"]["file_id"]
...             # depois baixa o arquivo e faz algo com ele...
...         elif "photo" in mensagem:
...             foto_mais_resolucao = mensagem["photo"][-1]
...             id_do_arquivo = foto_mais_resolucao["file_id"]
...             # depois baixa o arquivo e faz algo com ele...
...
...             proximo_id_de_update = resultado["update_id"] + 1
...
...     sleep(1)
```

Código para Busca Contínua de Novas Mensagens

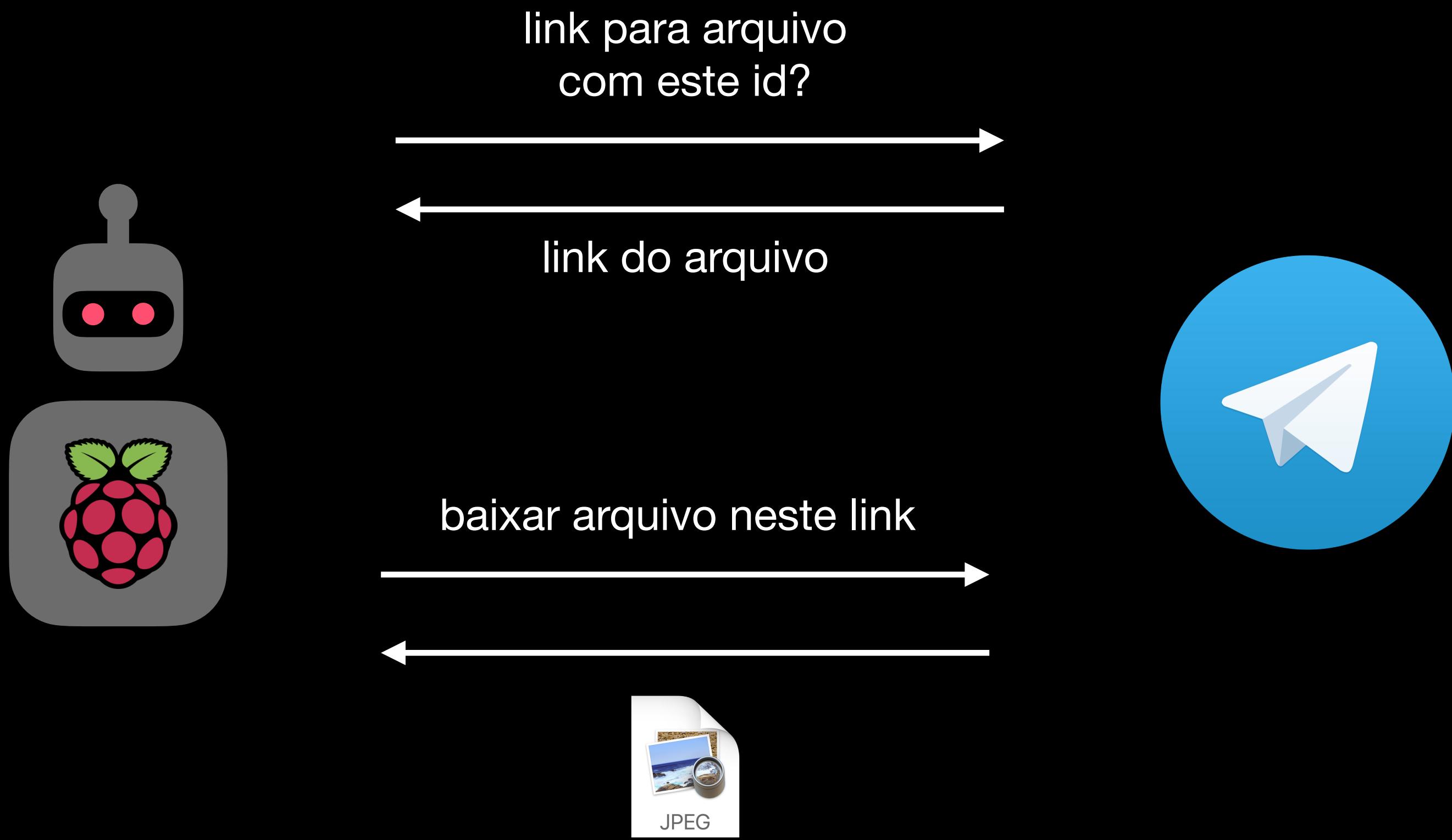
`id_do_arquivo`

download?



JPEG

Download de um Arquivo pelo Id



Download de um Arquivo pelo Id

```
>>> % depois de pegar o id_do_arquivo na getUpdate...
>>> endereco = endereco_base + "/getFile"
>>> dados = {"file_id": id_do_arquivo}
>>> resposta = get(endereco, json=dados)
>>> dicionario = resposta.json()
>>> final_do_link = dicionario["result"]["file_path"]
>>> final_do_link
'photos/file_117.jpg'
>>> link_do_arquivo = "https://api.telegram.org/file/bot" +
chave + "/" + final_do_link

>>> from urllib.request import urlretrieve
>>> arquivo_de_destino = "meu_arquivo.jpg"
>>> urlretrieve(link_do_arquivo, arquivo_de_destino)
```



Muita coisa...



É só seguir a receita

Receitas para Telegram do Chef Jan K. S.



Download do Telegram

Resumo da Ópera

Funcionalidade

Comandos

Capturar Foto

fswebcam

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

Capturar Áudio

arecord

```
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()
```

Conversão

lame / opusenc

```
from os import system • system("lame audio.wav audio.mp3")  
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=data, files=arquivo)  
dados = {"offset": proximo_id_de_update}  
resposta = get(base + "/getUpdates", json=dados)  
from urllib.request import urlretrieve  
resposta = get(base + "/getFile", json= {"file_id": id})  
final_do_endereco = resposta.json()["result"]["file_path"]  
urlretrieve(endereco_do_arquivo, nome_do_arquivo_baixado)
```

Funcionalidade

Datas e Horários
[acessar documentação](#)

Campainha
[acessar documentação](#)

Sensor de
Distância
[acessar documentação](#)

MongoDB
[acessar documentação](#)

Comandos

```
from datetime import datetime, timedelta
tempo = datetime(2018, 3, 28, 15, 35, 12) • datetime.now()
intervalo = timedelta(months=4) • intervalo.total_seconds()
tempo2 = tempo + intervalo • intervalo = tempo2 - tempo
tempo2 > tempo • texto = tempo.strftime("%d/%m/%Y %H:%M")
```

```
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)
```

```
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
```

```
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

Envio de
Infravermelho

Recebimento de
Infravermelho

Servidor Flask
[acessar documentação](#)

HTML

Ngrok

Comandos

```
from py_irsend.irsend import send_once
send_once("nome do controle", ["KEY_1", "KEY_2"])

from lirc import init, nextcode • init("aula", blocking=False)
while True:
    lista_com_codigo = nextcode()
    if lista_com_codigo != []:
        codigo = lista_com_codigo[0]
```

```
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"
```

```
<p>Parágrafo</p>

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

```
@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)
```

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

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.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.clear()
    lcd.message("Texto 1\nTexto 2")
```

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] • lista[1:3] • total_de_elementos = len(lista) lista.append(novo_elemento) • del lista[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!" • texto[0] • texto[1:4] • len(texto) texto + str(numero) • "x = %.2f, y = %d" % (num1, num2) 2 + int("11") • 4 / float("23.5")</pre>
Condicionais	<pre>if x != 0: 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): global x x = 2</pre>
Criação de Funções e Variáveis Globais	