

# Projeto 04

## Controle por Notificação – Teoria

Jan K. S. – janks@puc-rio.br

ENG1419 – Programação de Microcontroladores

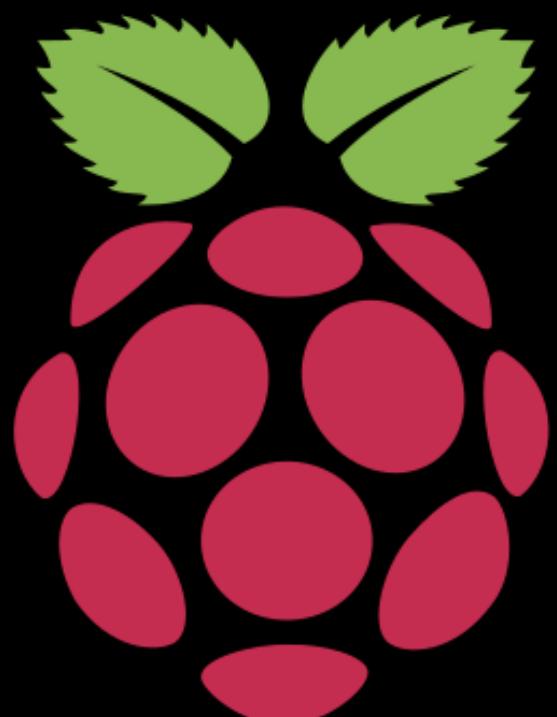
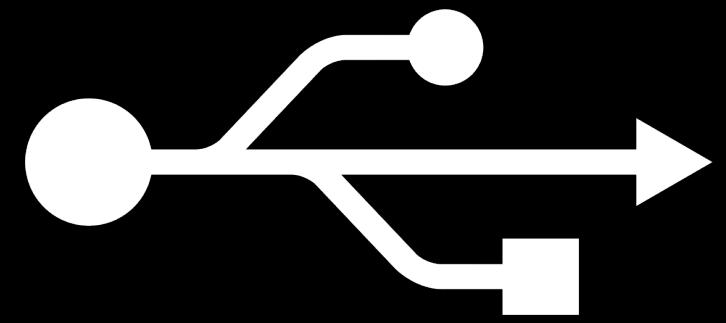
# Hardware



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-line syntax `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 file syntax and comments, and a "General Options" subsection with entries for `-?`, `--help`, `-c`, and `--config`.

**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`.

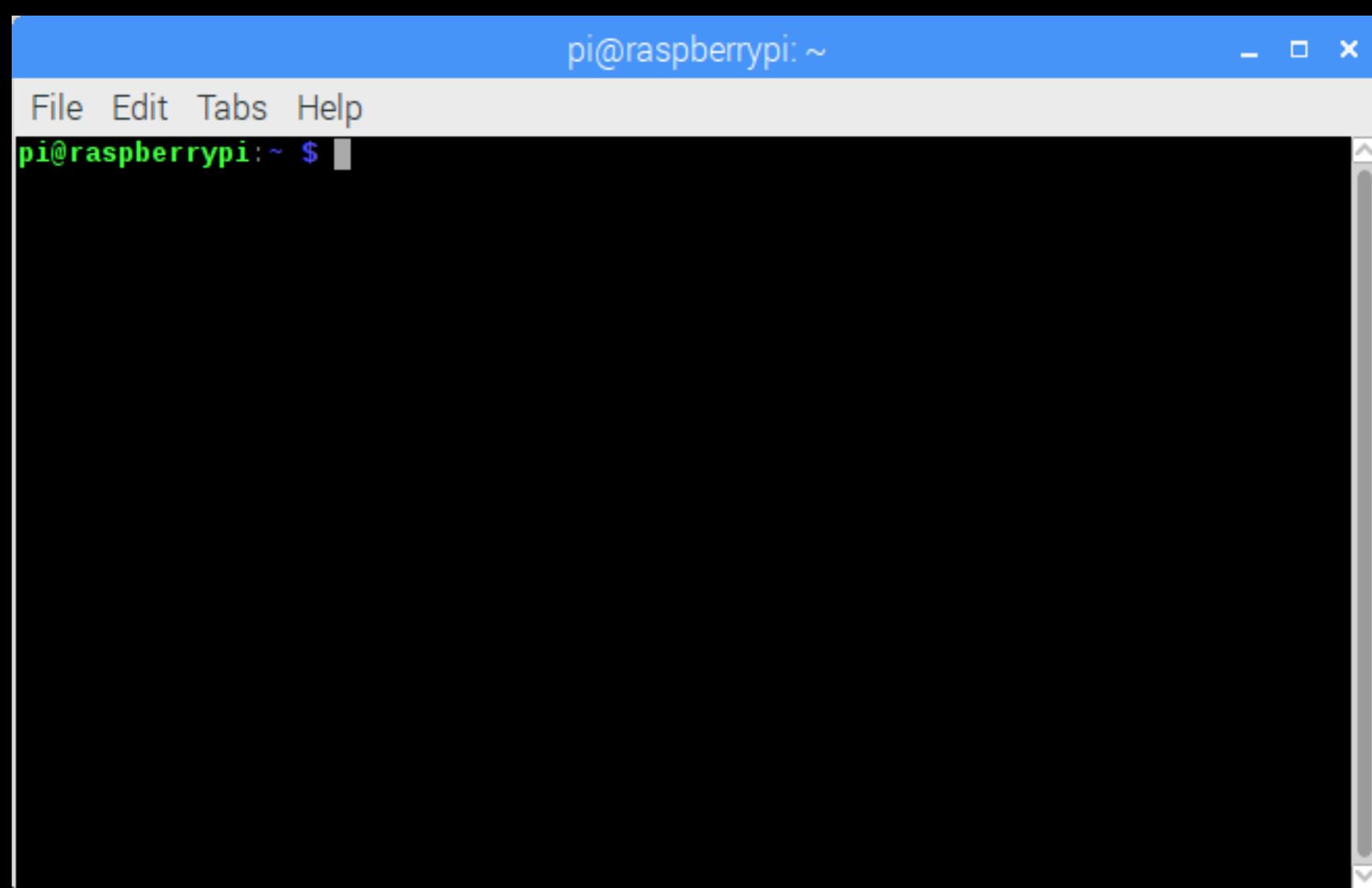
**CONFIGURATION**

**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.

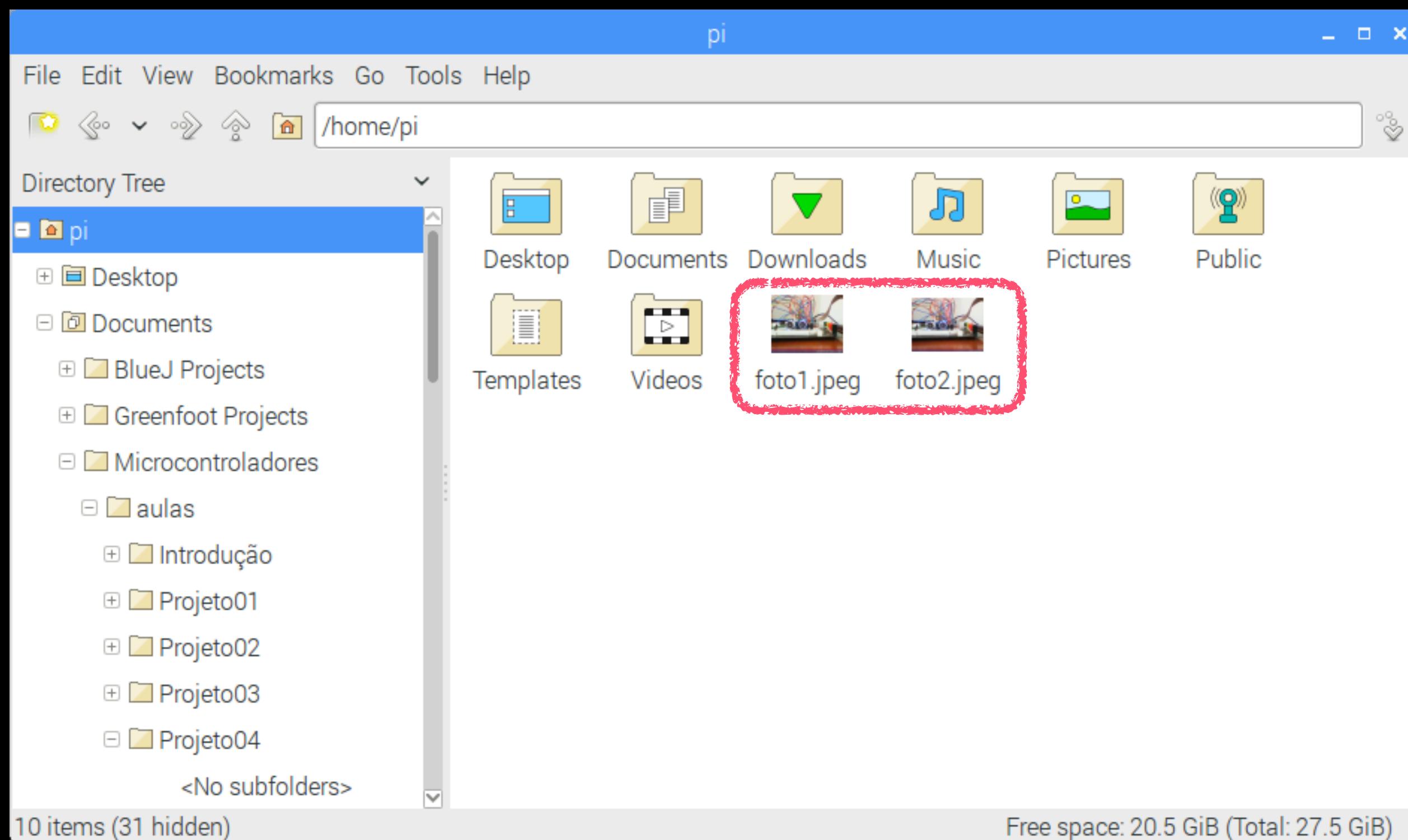
`-c`, `--config`  
Load options from a file. You can load more than one config file, and can mix them with command-line arguments.



Aplicativo Terminal

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

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



Arquivos Gerados no File Manager

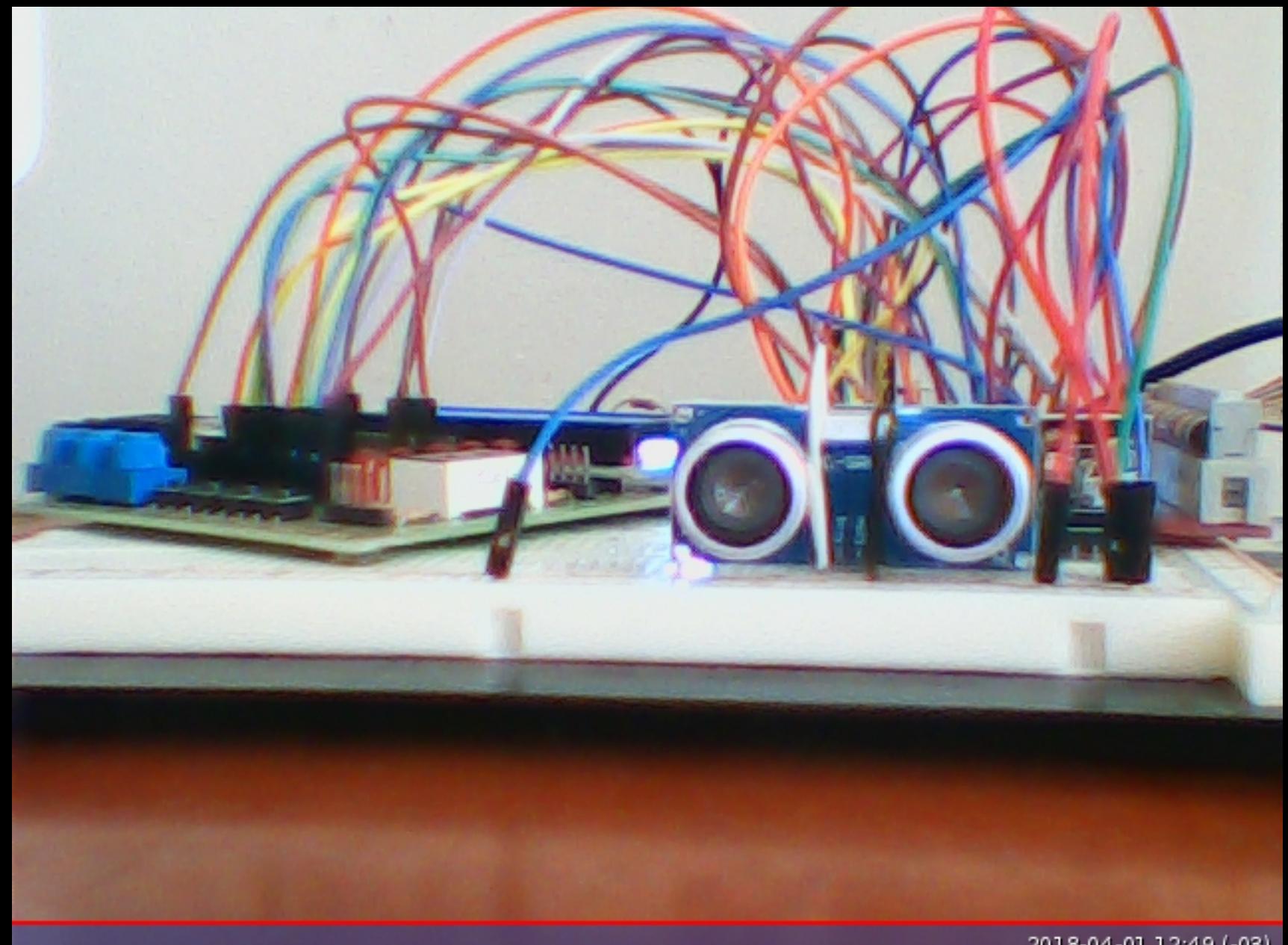
**foto1.jpeg**

resolução padrão (352 × 288)

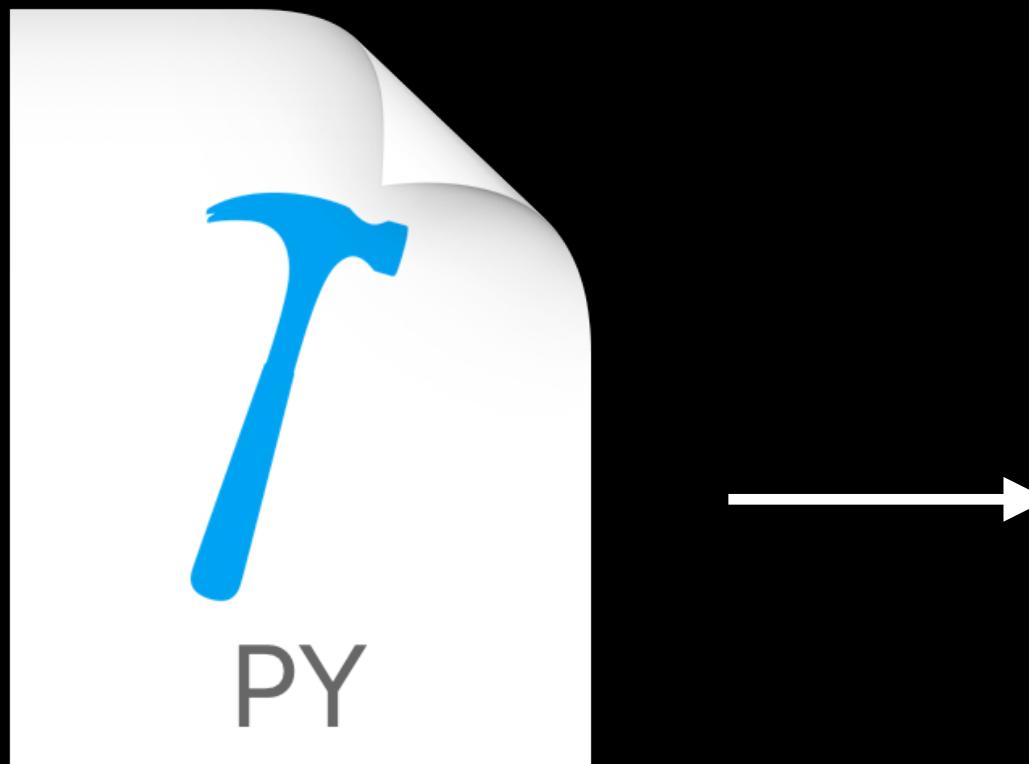


**foto2.jpeg**

resolução 640x480



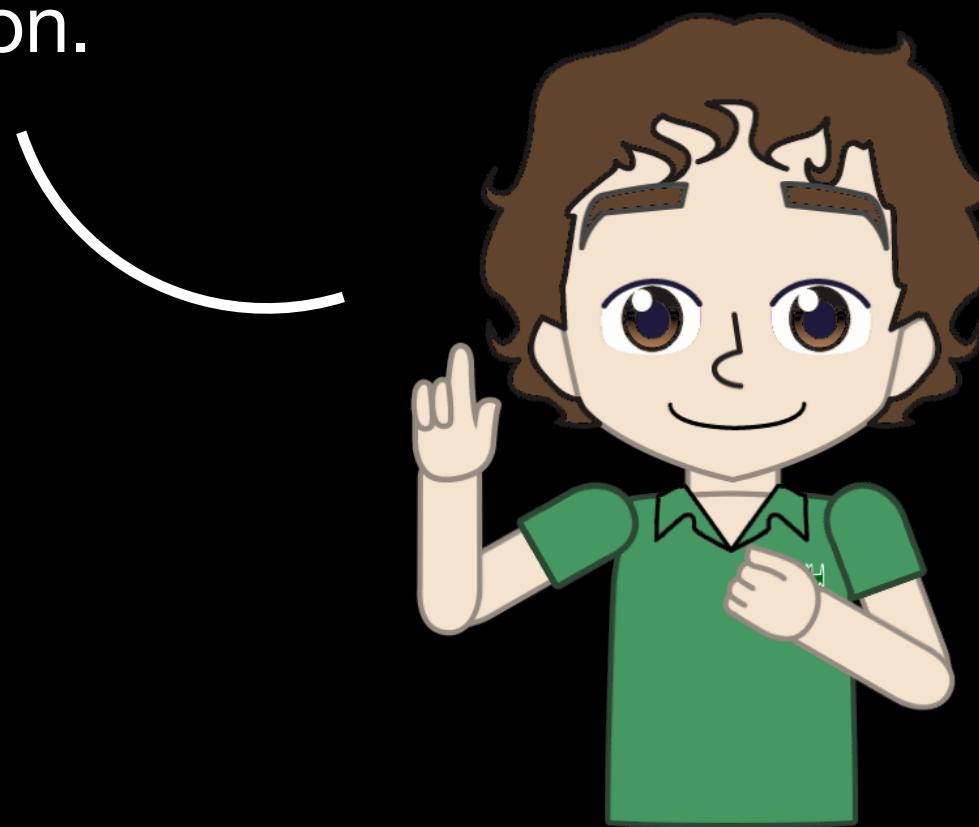
Comparação entre Diferentes Resoluções



```
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.jpeg"  
>>> system(comando)  
>>> system("fswebcam --resolution 640x480 foto2.jpeg")
```

Chamada de Comandos para Tirar uma Foto

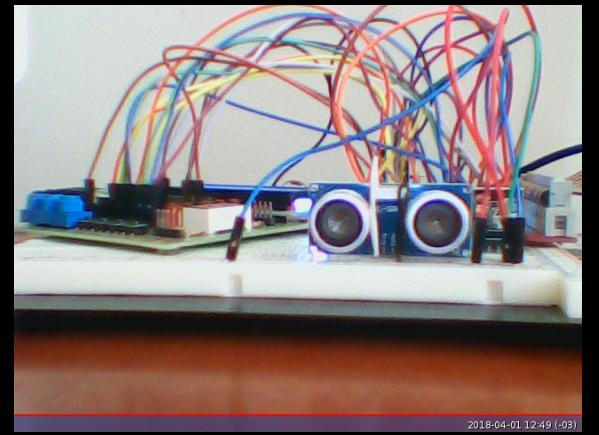


foto-15:25:37.jpeg

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

**NAME**

**SYNOPSIS**

**DESCRIPTION**

**OPTIONS**

**SIGNALS**

**EXAMPLES**

**SEE ALSO**

**BUGS**

**AUTHOR**

**COMMENTS**

**NAME**

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

**SYNOPSIS**

**arecord** [*flags*] [*filename*]  
**aplay** [*flags*] [*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!



arecord

Interrupção da Gravação de Áudio

```
>>> from subprocess import Popen  
>>> 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()  
>>> ...  
>>> parar_gravacao() →  audio.wav
```

Interrupção da Gravação de Áudio



audio.wav

516 kB

conversão →



audio.mp3

47.8 kB



audio.ogg

28.9 kB

Conversão do Formato WAV para MP3 e para OGG

**Manpages**

Manpage:  go

<b>NAME</b>	<b>Available in</b>
<a href="#">lame – create mp3 audio files</a>	(1)
<b>SYNOPSIS</b>	<a href="#">Contents</a>
<a href="#">lame [options] &lt;infile&gt; &lt;outfile&gt;</a>	<a href="#">NAME</a>
<b>DESCRIPTION</b>	<a href="#">SYNOPSIS</a>
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>.	<a href="#">DESCRIPTION</a>
<b>OPTIONS</b>	<a href="#">OPTIONS</a>
Input options:	<a href="#">ID3 TAGS</a>
<b>r</b> 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 <b>r</b> , LAME will perform several <code>fseek()</code> 's on the input file looking for WAV and AIFF headers.	<a href="#">ENCODING MODES</a>
Might not be available on your release.	<a href="#">PRESETS</a>
<b>x</b> Swap bytes in the input file (or output file when using <b>–decode</b> ).	<a href="#">EXAMPLES</a>
	<a href="#">BUGS</a>
	<a href="#">SEE ALSO</a>
	<a href="#">AUTHORS</a>
	<a href="#">COMMENTS</a>

**Manpages**

Manpage:  lame go

<b>Available in</b>	<b>NAME</b>
(1)	<a href="#">opusenc – encode audio into the Opus format</a>
<b>Contents</b>	<b>SYNOPSIS</b>
<a href="#">NAME</a>	<a href="#">opusenc [ -h ] [ -V ] [ --help-picture ] [ --quiet ] [ --bitrate <i>kbit/sec</i> ] [ --vbr ] [ --cvbr ] [ --hard-cbr ] [ --comp <i>complexity</i> ] [ --framesize 2.5, 5, 10, 20, 40, 60 ] [ --expect-loss <i>pct</i> ] [ --downmix-mono ] [ --downmix-stereo ] [ --max-delay <i>ms</i> ] [ --title '<i>track title</i>' ] [ --artist <i>author</i> ] [ --album '<i>album title</i>' ] [ --genre <i>genre</i> ] [ --date YYYY-MM-DD ] [ --comment <i>tag=value</i> ] [ --picture <i>filename specification</i> ] [ --padding <i>n</i> ] [ --discard-comments ] [ --discard-pictures ] [ --raw ] [ --raw-bits <i>bits/sample</i> ] [ --raw-rate <i>Hz</i> ] [ --raw-chan <i>N</i> ] [ --raw-endianness <i>flag</i> ] [ --ignorelength ] [ --serial <i>serial number</i> ] [ --save-range <i>file</i> ] [ --set-ctl-int <i>ctl=value</i> ] <i>input.wav output.opus</i></a>
<a href="#">SYNOPSIS</a>	<b>DESCRIPTION</b>
<a href="#">DESCRIPTION</a>	<b>opusenc</b> 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.
<a href="#">OPTIONS</a>	Unless quieted <b>opusenc</b> displays fancy statistics about the encoding progress.
<a href="#">EXAMPLES</a>	<b>OPTIONS</b>
<a href="#">BUGS</a>	<b>General options</b>
<a href="#">SEE ALSO</a>	-h, --help
<a href="#">AUTHORS</a>	Show command help
<a href="#">COMMENTS</a>	



audio.wav



audio.mp3

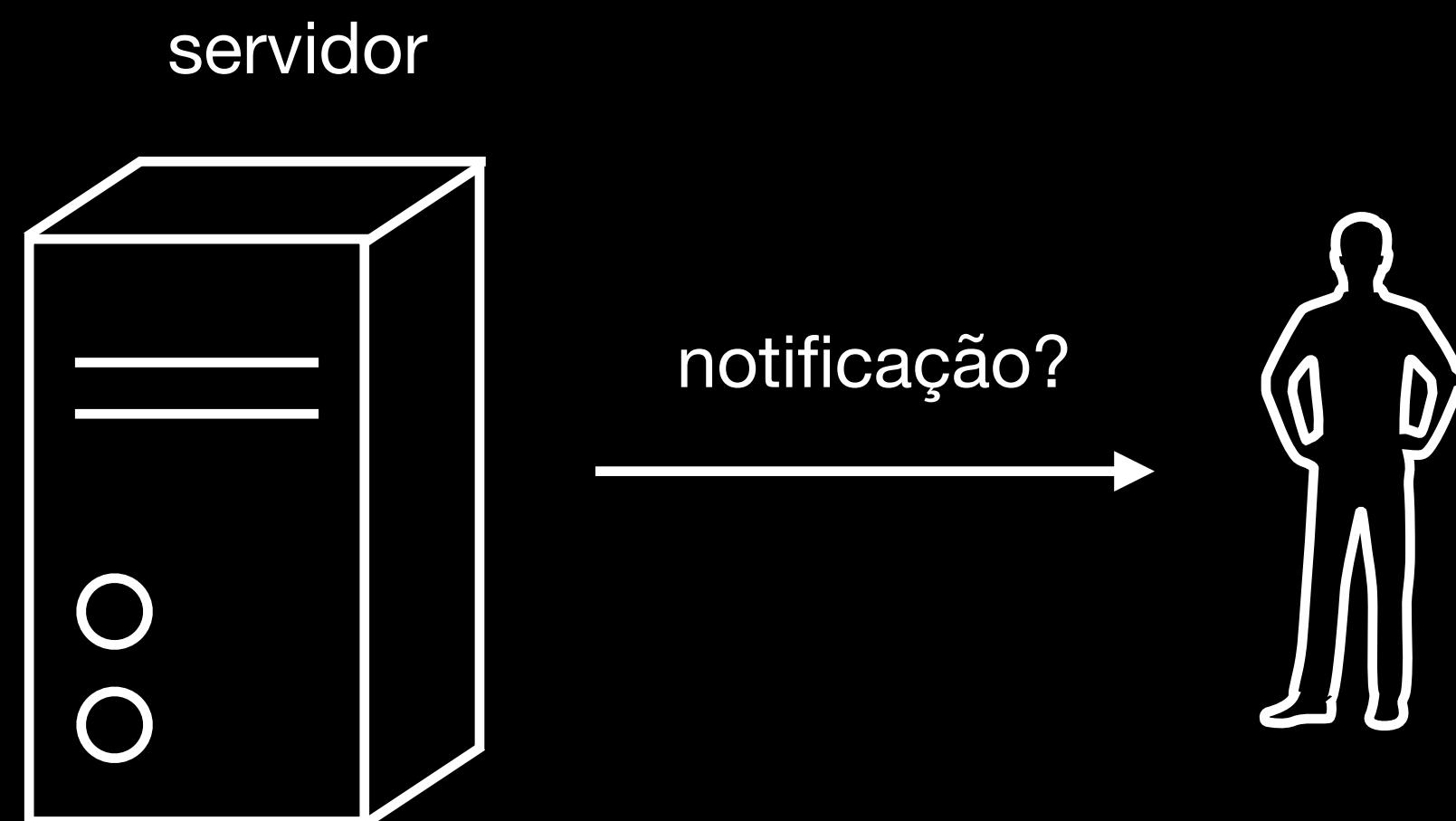
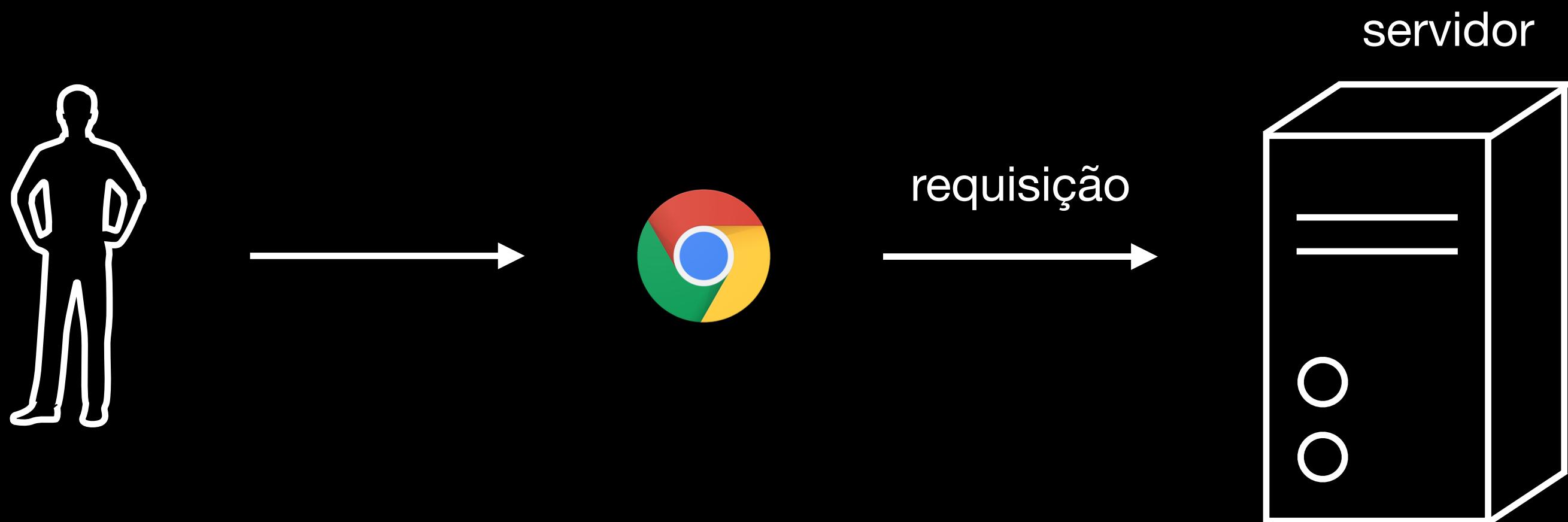


audio.ogg

```
>>> 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

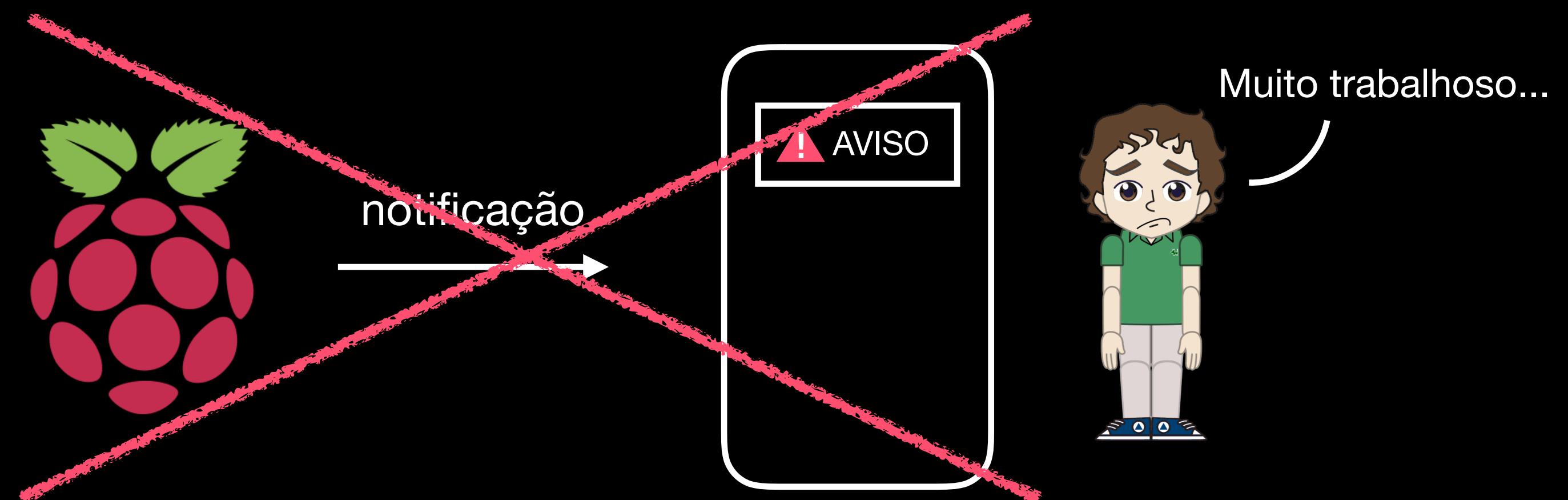
# Software



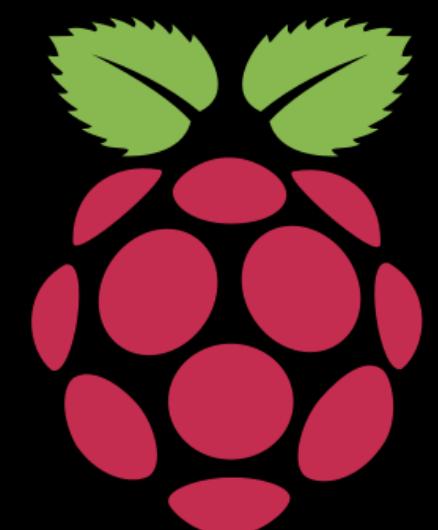
Requisição x Notificação



Notificação em Smartphones via Apps



Notificações com um Servidor Intermediário



"bot"

requisição

Servidor de um App



objetivo do Projeto 04

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 responding to users without violating privacy or spamming.
- Middle Window:** Facebook Developers documentation for the 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 App Directory. The sidebar shows "Staff Picks" with categories like Featured, Essential Apps, New & Noteworthy, and Brilliant Bots. The main content area is titled "Bots" and lists several apps:
  - To-do (Workast)
  - Zapier
  - Polly
  - Jira Cloud
  - Hubot
  - busybot

Exemplos de Comunicação entre Bots e Apps



Telegram

The screenshot shows the official Telegram website ([telegram.org](https://telegram.org)) displayed in a web browser. The page features three main sections illustrating the platform's cross-device integration:

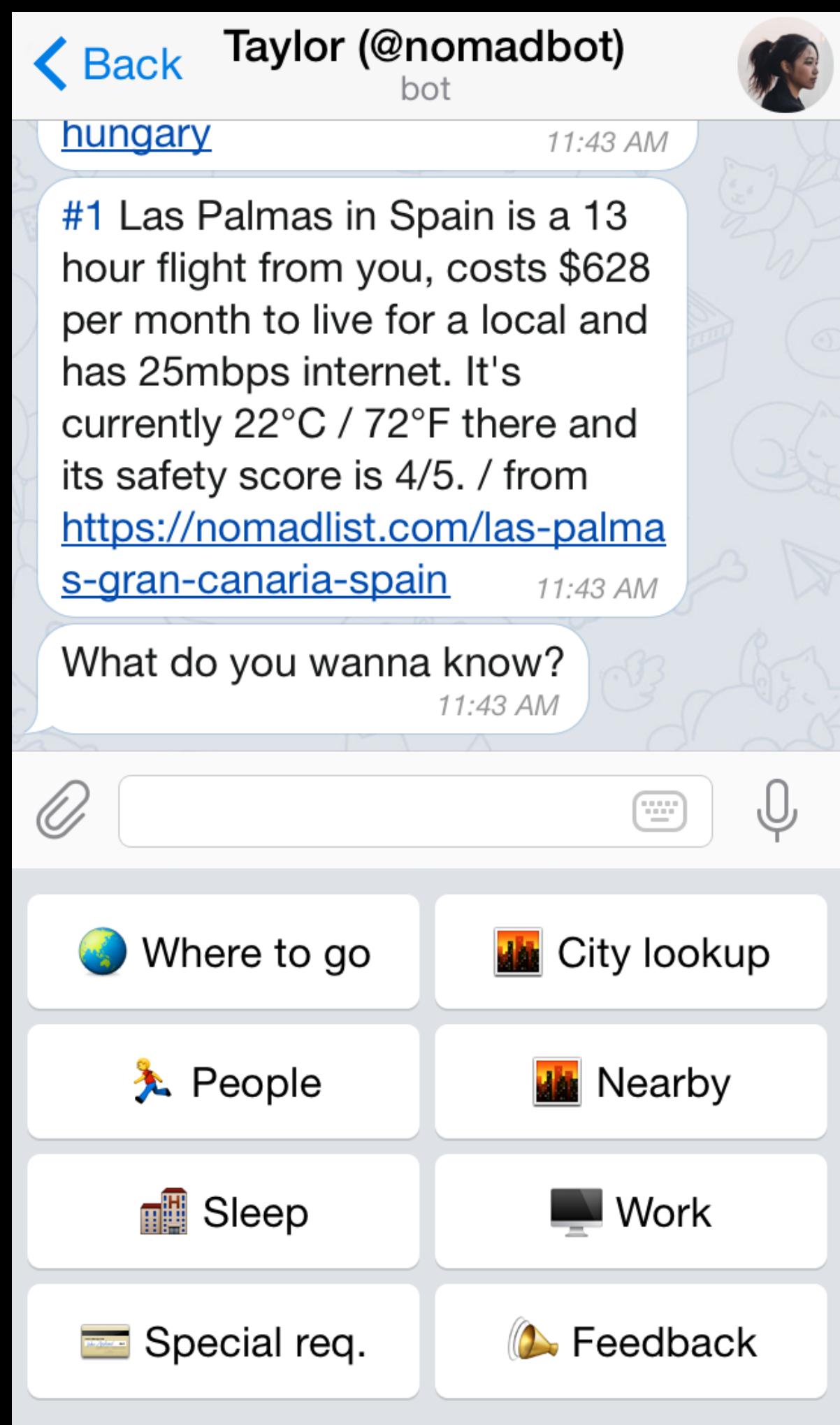
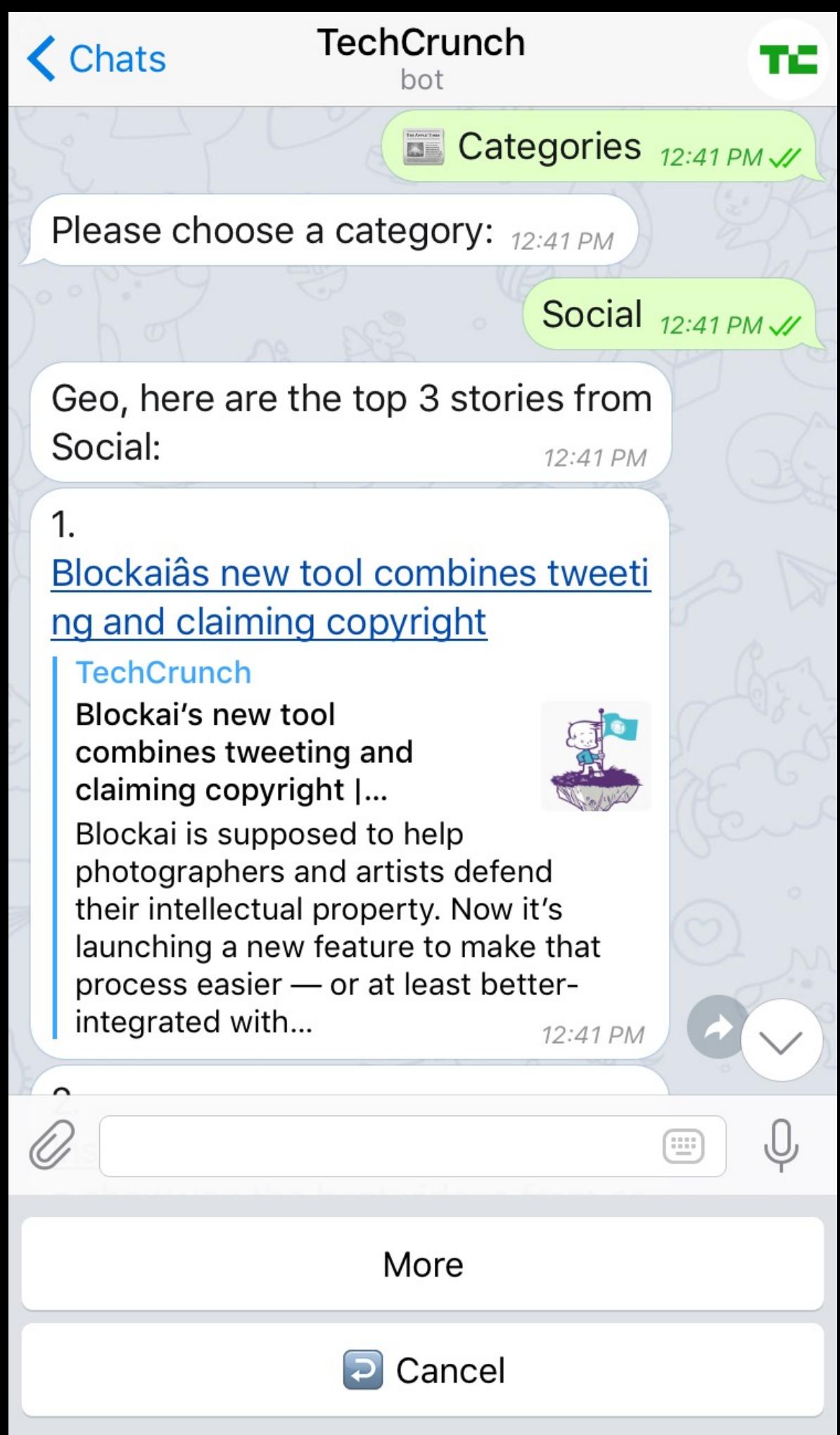
- Mobile Devices:** Displays screenshots of the Telegram app running on an Android smartphone, an iPhone, and a Windows Phone. The Android and iPhone screens show a messaging interface with multiple conversations and media files. The Windows Phone screen shows a similar interface with a focus on shared media files.
- Desktop/Laptop:** Shows a laptop displaying the Telegram Web-version, which has a clean, modern design with a sidebar for contacts and a main area for messages and media.
- Mac OS X:** Shows a Mac OS X desktop with the Telegram for macOS application open. The interface is identical to the mobile versions, providing a native experience on the desktop.

**A native app for every platform**

**Telegram for Android** **Telegram for iPhone / iPad** **Telegram for WP**

**Telegram Web-version** **Telegram for macOS** **Telegram for PC/Mac/Linux**

Plataformas do Telegram

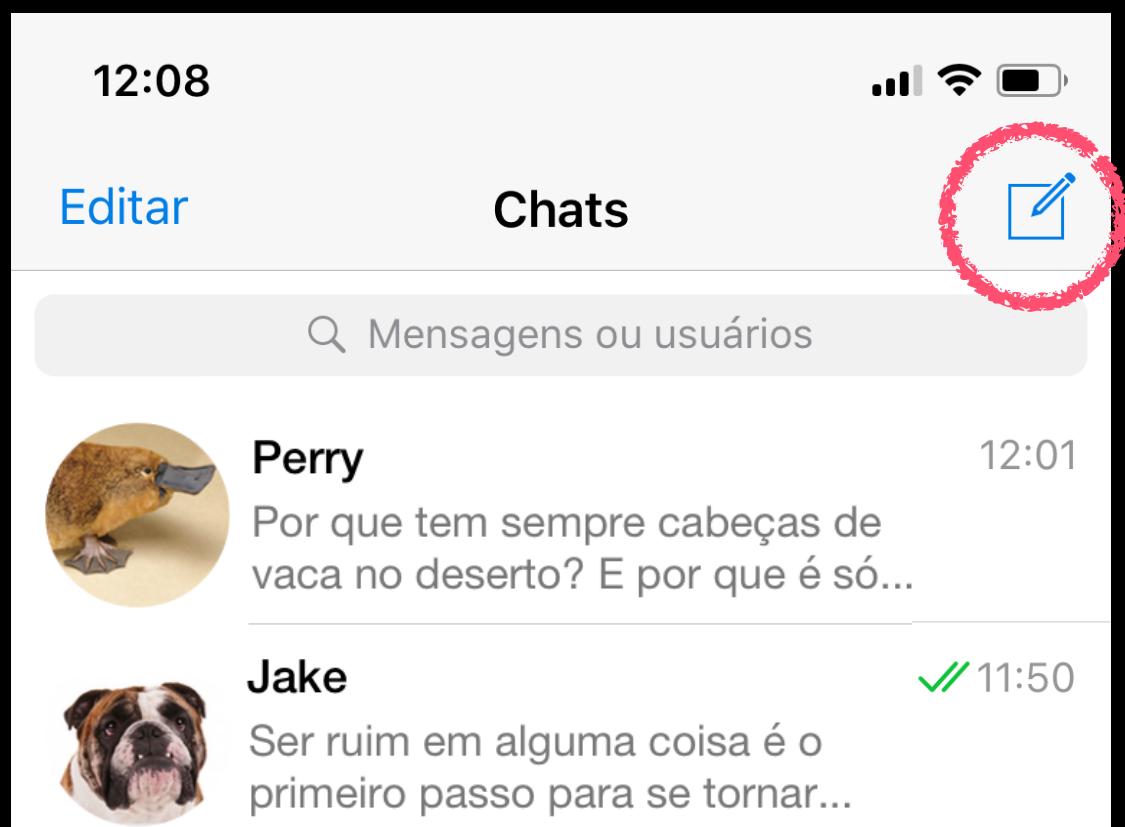


Exemplos de Conversas com Bots

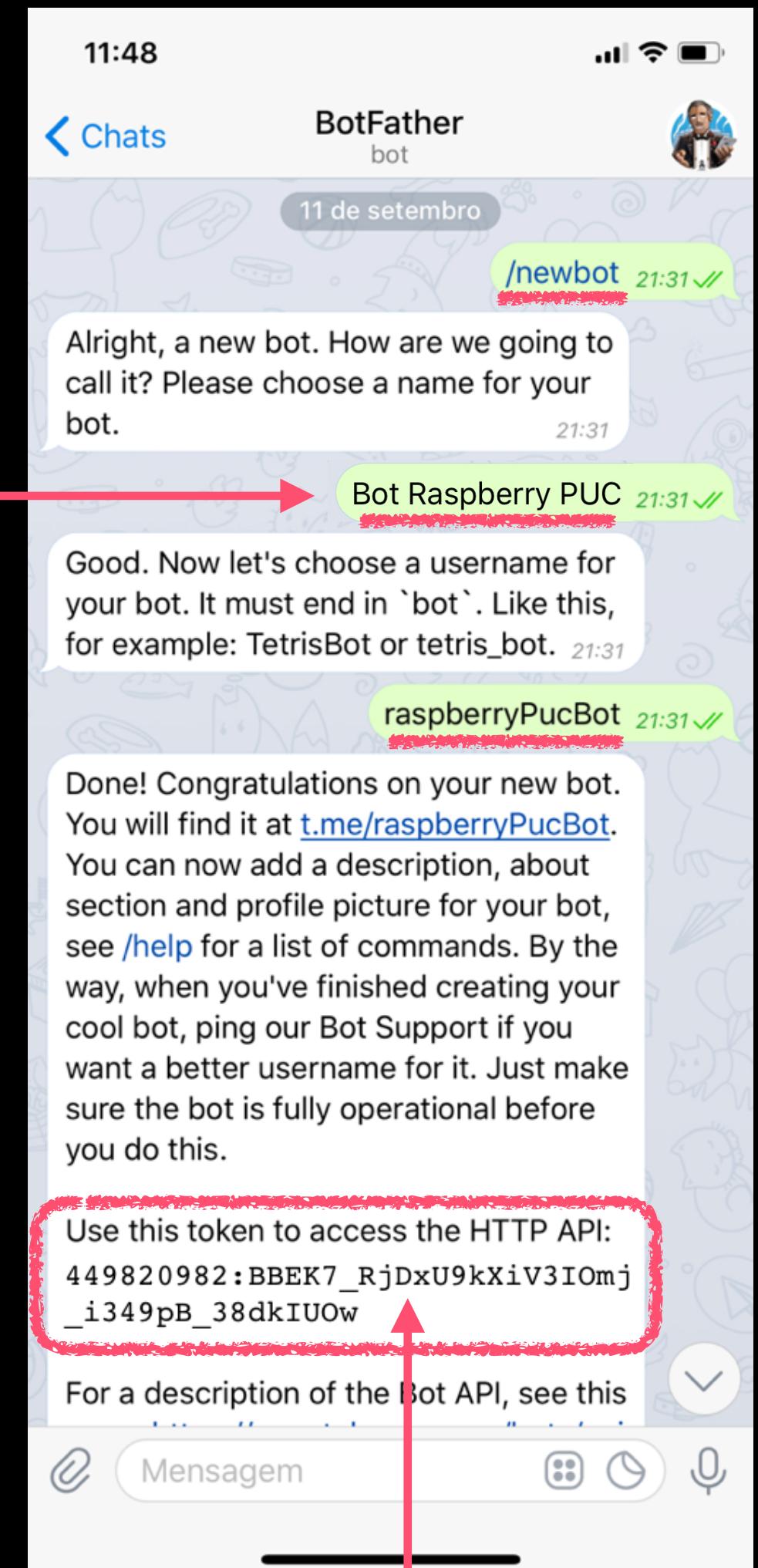
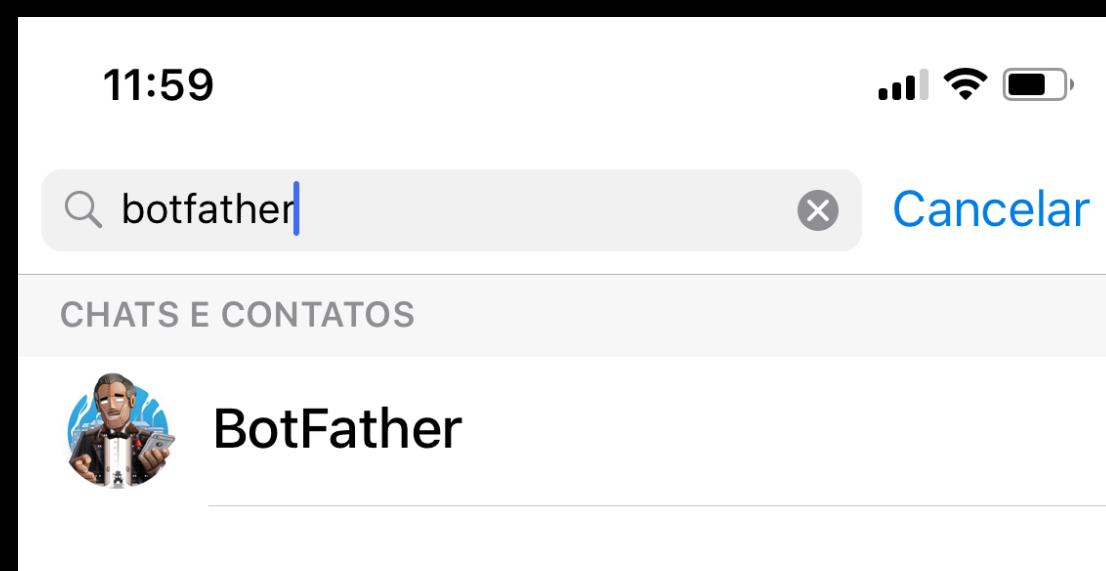


BotFather is the one bot to rule them all.  
Use it to create new bot accounts and manage your existing bots.

Criação de Bots no Telegram via BotFather



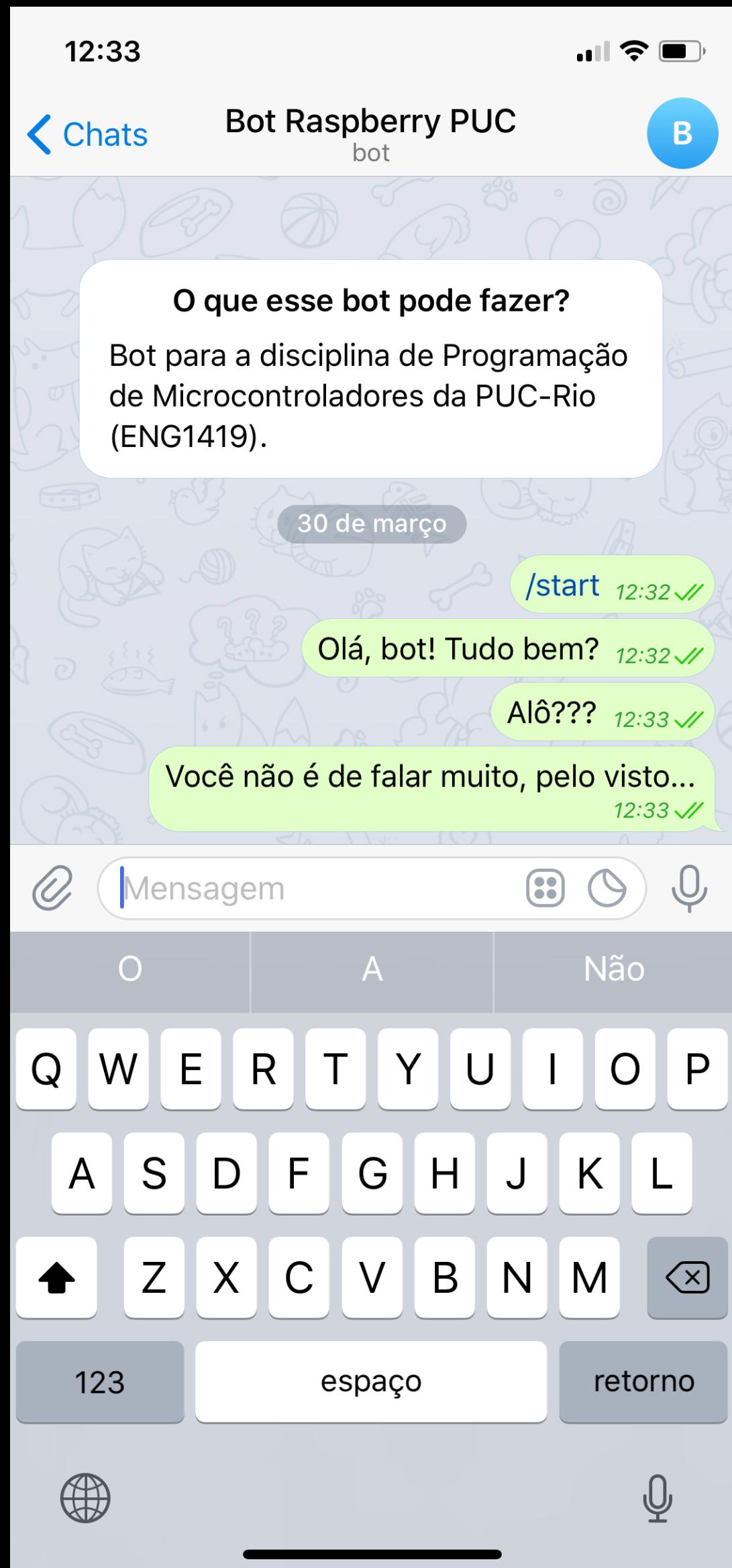
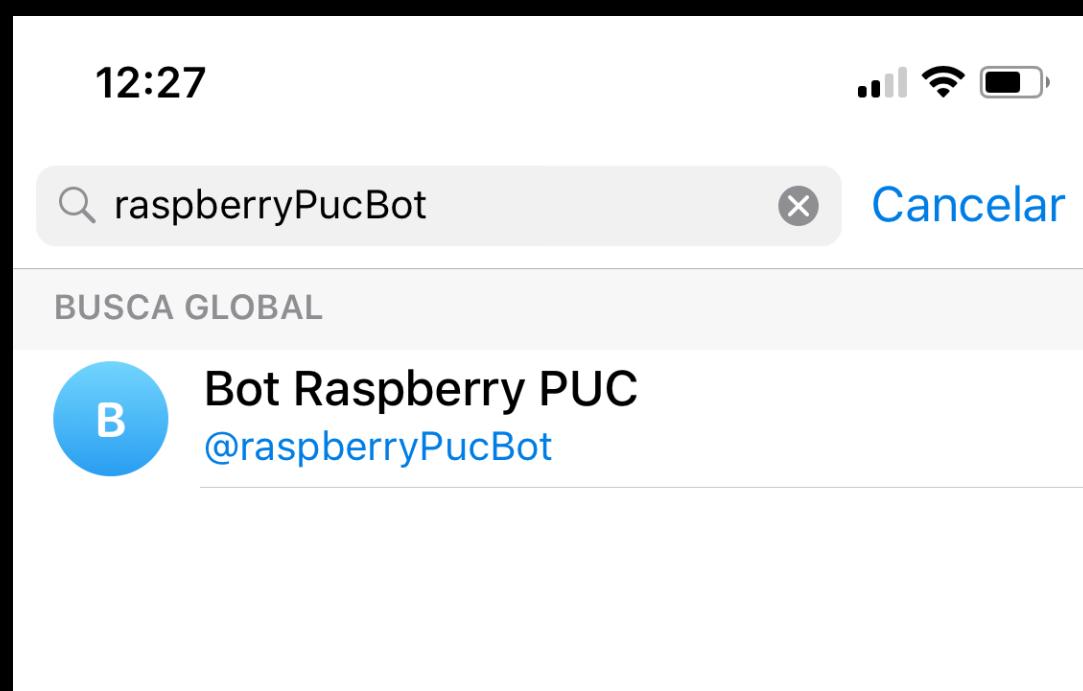
invente um nome seu!



anote esta chave!

Criação de Bots no Telegram via BotFather

**pesquise o nome  
do SEU bot!**



Conversa com o Seu Bot

core.telegram.org/bots/api

Recent changes  
Authorizing your bot  
Making requests  
Getting updates  
Available types  
**Available methods**

Parameters	Type	Required	Description
chat_id	Integer or String	Yes	Unique identifier for the target chat or username of the target channel (in the format <code>@channelusername</code> )
text	String	Yes	Text of the message to be sent
parse_mode	String	Optional	Send <a href="#">Markdown</a> or <a href="#">HTML</a> , if you want Telegram apps to show <b>bold</b> , <i>italic</i> , <code>fixed-width</code> text or inline URLs in your bot's message.
disable_web_page_preview	Boolean	Optional	Disables link previews for links in this message
disable_notification	Boolean	Optional	Sends the message <a href="#">silently</a> . Users will receive a notification with no sound.
reply_to_message_id	Integer	Optional	If the message is a reply, ID of the original message
reply_markup	<a href="#">InlineKeyboardMarkup</a> or <a href="#">ReplyKeyboardMarkup</a> or <a href="#">ReplyKeyboardRemove</a> or <a href="#">ForceReply</a>	Optional	Additional interface options. A JSON-serialized object for an <a href="#">inline keyboard</a> , <a href="#">custom reply keyboard</a> , instructions to remove reply keyboard or to force a reply from the user.

Muita coisa para entender!



Métodos Cheios de Detalhes do Telegram

The screenshot shows a Mac OS X browser window with the URL [docs.python-requests.org](https://docs.python-requests.org) in the address bar. The page content is as follows:

**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:**

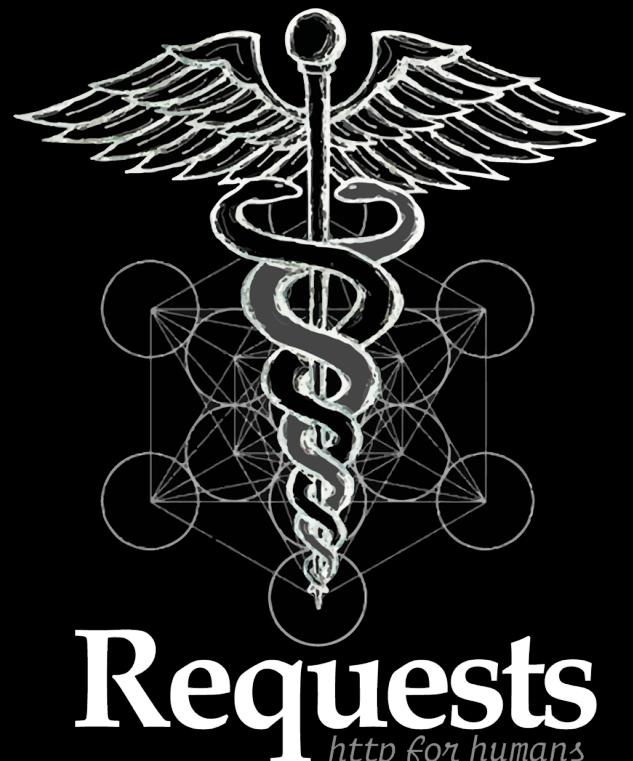
The use of **Python 3** is *highly* preferred over Python 2. Consider upgrading your applications and infrastructure if you find yourself *still* using Python 2 in production today. If you are using Python 3, congratulations — you are indeed a person of excellent taste.  
—Kenneth Reitz

---

**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 ▾



endereço web do comando  
+ dados opcionais

resposta

Servidor de um App

Requisição Web a um Servidor

[api.telegram.org/botSUA\\_CHAVE\\_SECRETA/getMe](https://api.telegram.org/botSUA_CHAVE_SECRETA/getMe)



Endereço para Informações sobre o Bot do Telegram

```
>>> chave = "COLOQUE A SUA CHAVE AQUI!"  
>>> endereco_base = "https://api.telegram.org/bot" + chave  
                  "  
Atenção ao "s"!
```

```
>>> from requests import get  
  
>>> endereco = endereco_base + "/getMe"  
  
>>> resposta = get(endereco)  
  
  
>>> resposta  
<Response [200]>  
  
>>> resposta.text  
'{"ok": true, "result": {"id":449820982, "is_bot": true,  
"first_name": "Bot Raspberry PUC", "username": "raspberryPucBot"}}'  
  
>>> dicionario_da_resposta = resposta.json()  
  
>>> dicionario_da_resposta["ok"]  
True  
  
>>> dicionario_da_resposta["result"]  
{'id': 449129546, 'is_bot': True, 'first_name': 'Bot Raspberry  
PUC', 'username': 'raspberryPucBot'}  
  
>>> dicionario_da_resposta["result"]["username"]  
'raspberryPucBot'
```

Chamada GET para Informações sobre o Bot

[api.telegram.org/botSUA\\_CHAVE\\_SECRETA/getUpdates](https://api.telegram.org/botSUA_CHAVE_SECRETA/getUpdates)

A screenshot of a Mac OS X desktop environment showing a web browser window. The address bar contains the URL "api.telegram.org". The main content area of the browser displays a JSON object representing a series of messages from a conversation. The JSON structure includes fields for update\_id, message (with message\_id, from, chat), and entities (for bot commands). The messages are in Portuguese and involve a user named "Jan" and a bot named "wallysalami".

```
{"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\u00ea n\u00e3o \u00e9 de falar muito, pelo visto..."}}]}]
```

Endereço para Obter Atualizações das Conversas com o Bot

```
{  
    "ok":true,  
    "result": [  
        {  
            "update_id":564714809,  
            "message":{  
                ...  
                "chat":{  
                    "id":314253469,  
                    "first_name":"Jan",  
                    "last_name":"K. S.",  
                    ...  
                },  
                "text":"/start"  
            }  
        },  
        {  
            "update_id":564714810,  
            "message":{  
                ...  
                "chat":{  
                    "id":314253469,  
                    "first_name":"Jan",  
                    "last_name":"K. S.",  
                    ...  
                },  
                "text":"Olá, bot! Tudo bem?"  
            }  
        },  
        ...  
    ]  
}
```

Campos Importantes nas Atualizações de Mensagens

{json}

KEEP  
CALM  
AND  
PERCORRE  
O JSON

```
>>> from requests import get
>>> endereco = endereco_base + "/getUpdates"
>>> resposta = get(endereco)
>>> dicionario_da_resposta = resposta.json()
>>> for atualizacao in dicionario_da_resposta["result"]:
...     print(atualizacao["message"]["text"])
...
'/start'
'Olá, bot! Tudo bem?'
'Alô?'
'Você não é de falar muito, não é?'
```

Tipo de Requisição	Modifica dados no servidor?
GET	Não
POST	Sim

Requisição GET x Requisição POST

api.telegram.org/bot**SUA\_CHAVE\_SECRETA**/**<comando>**

+

dados extras

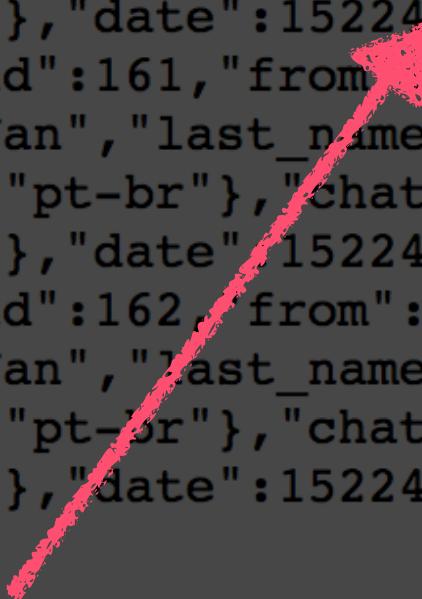
	<b>Comando</b>	<b>Dados Obrigatórios</b>	
GET {	getMe	(nenhum)	
	getUpdates	(nenhum)	
POST {	sendMessage	id do chat, texto	 Onde encontro o id do chat?
	sendPhoto	id do chat, arquivo	
	sendVoice	id do chat, arquivo	

Exemplos de Outros Comandos para Bots do Telegram

[api.telegram.org/botSUA\\_CHAVE\\_SECRETA/getUpdates](https://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 identificador!**

Endereço para Obter Atualizações das Conversas com o Bot

```
>>> chave = "COLOQUE A SUA CHAVE 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":  
1535815441,"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.jpeg", "rb")}  
>>> resposta = post(endereco, data=dados, files=arquivo)
```



Foto Enviada pelo Bot na Conversa

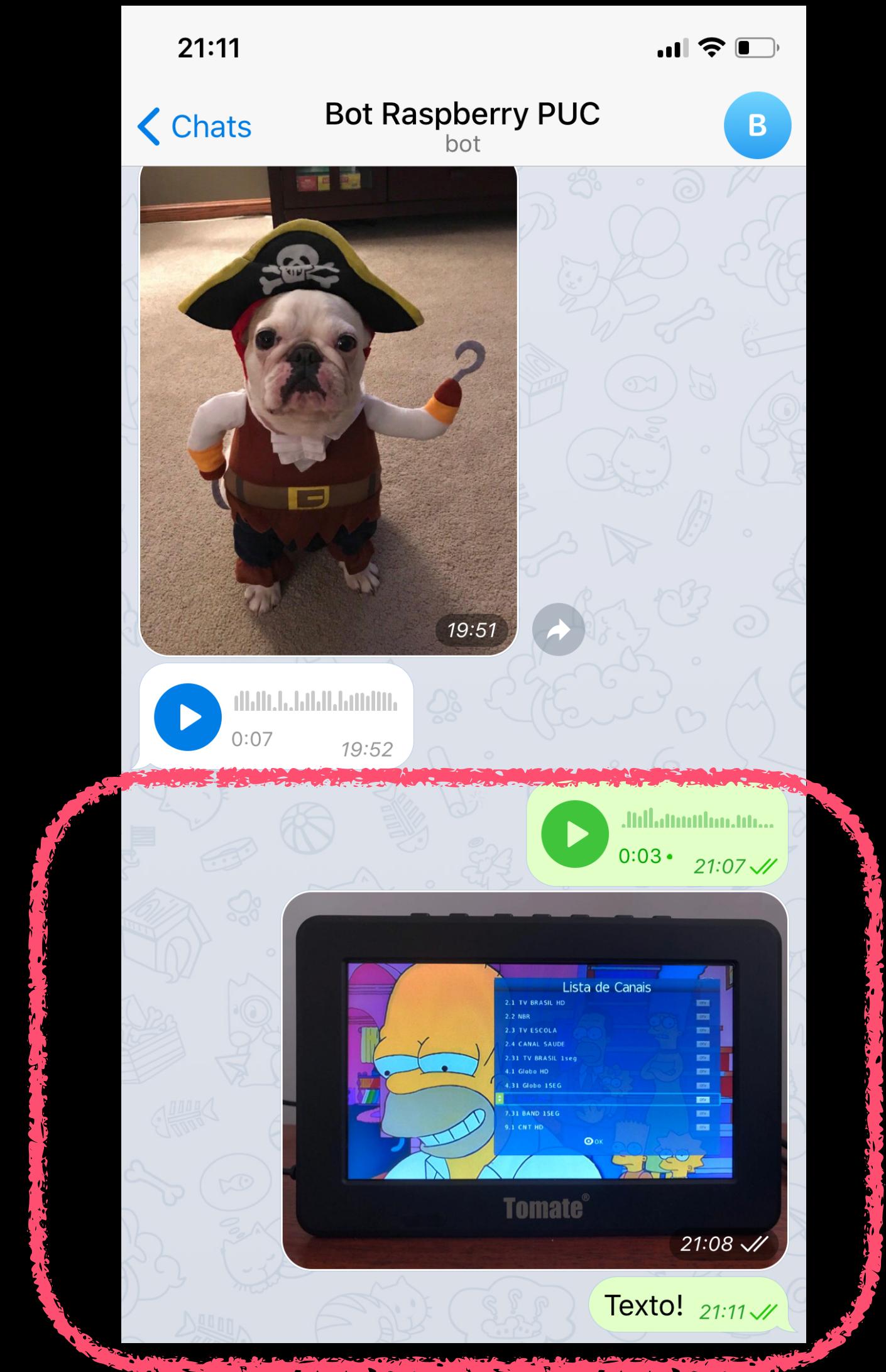
Use sempre o formato OGG para enviar áudio via Telegram.



```
>>> 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

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

Dicionário (JSON) com as Atualizações de Mensagens

{json}

KEEP  
CALM  
AND  
PERCORRE  
O JSON

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

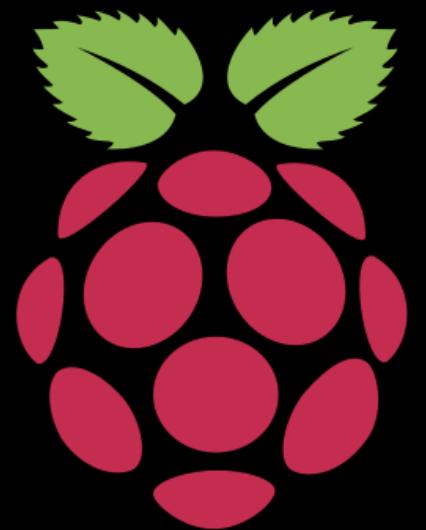
`id_do_arquivo`

download?



JPEG

Download de um Arquivo pelo Id



"bot"

link para arquivo  
com este id?



link do arquivo



baixar arquivo neste link



Download de um Arquivo pelo Id

```
>>> 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)
```

Busca Atualizações



update\_id: 123000  
text: "Olá!"

um tempo depois...

Busca Atualizações



update\_id: 123000  
text: "Olá!"

update\_id: 123001  
text: "Tudo bem?"

Como fazer para não vir a  
mesma mensagem de novo?



Atualizações com Repetição

Busca Atualizações



update\_id: 123000  
text: "Olá!"

um tempo depois...

Busca Atualizações  
**a partir do id 123001**



update\_id: 123001  
text: "Tudo bem?"

Atualizações a Partir de um Id

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



Muita coisa...



**É só seguir a receita**

Receitas para Telegram do Chef Jan K. S.



Download do Telegram

# Resumo da Ópera

## Funcionalidade

FSWebcam

[acessar documentação](#)

ARecord

[acessar documentação](#)

Lame

[acessar documentação](#)

OpusEnc

[acessar documentação](#)

Telegram

[acessar documentação](#)

## Comandos

```
from os import system  
system("fswebcam foto.jpg")  
system("fswebcam --resolution 640x480 --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 <a href="#">acessar documentação</a>	<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 <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!" • 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 &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>