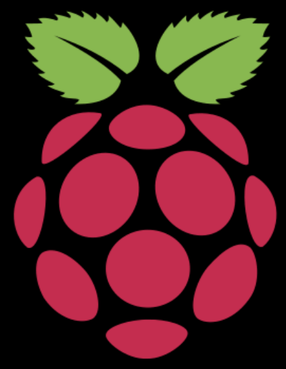


Projeto 04

Controle por Notificação

Jan K. S. – janks@puc-rio.br

ENG1419 – Programação de Microcontroladores

Hardware

Webcam

man.cx/fswebcam

Manpages

Manpage: go

Available in

(1)

Contents

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

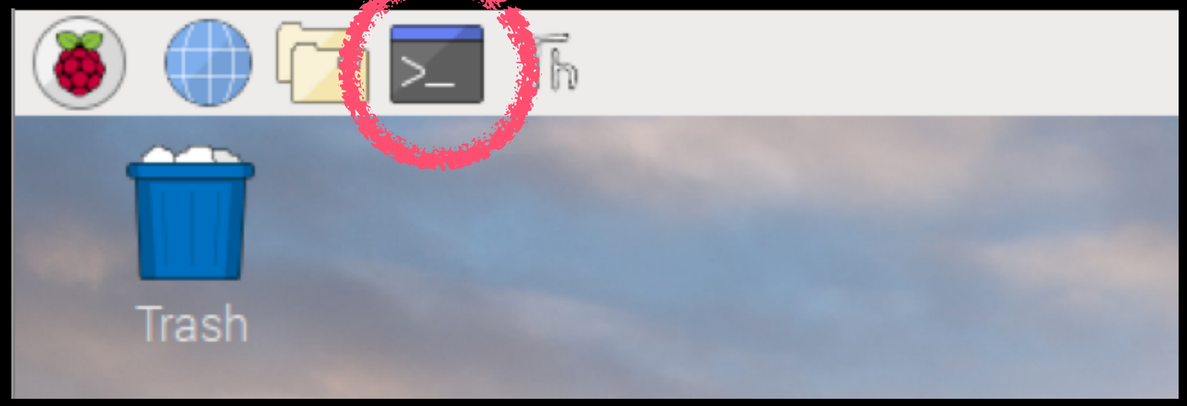
–?, ––help

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 FSWebcam

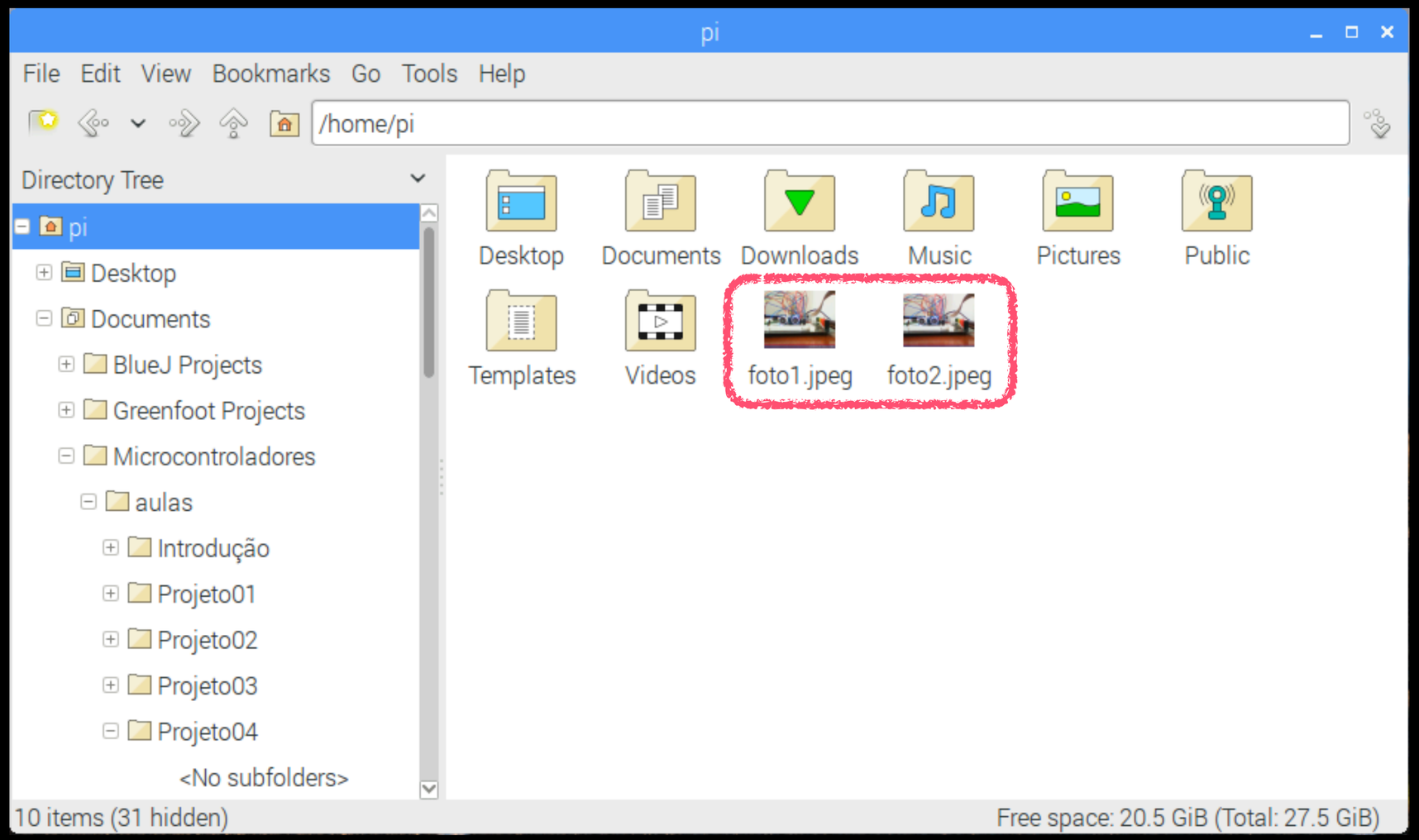


Aplicativo Terminal

```
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 'foto.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'.
```

Comandos no Terminal para Captura de Fotos



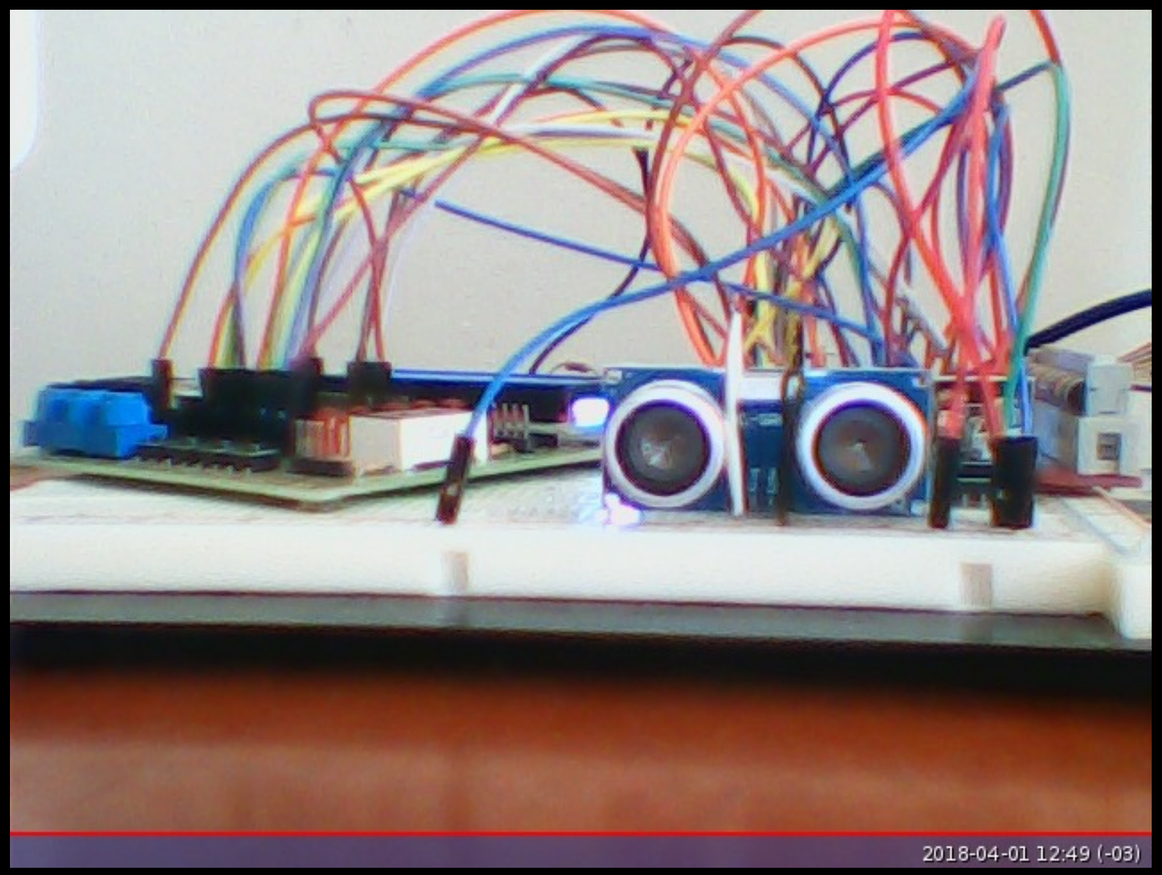
Arquivos Gerados no File Manager

foto1.jpeg

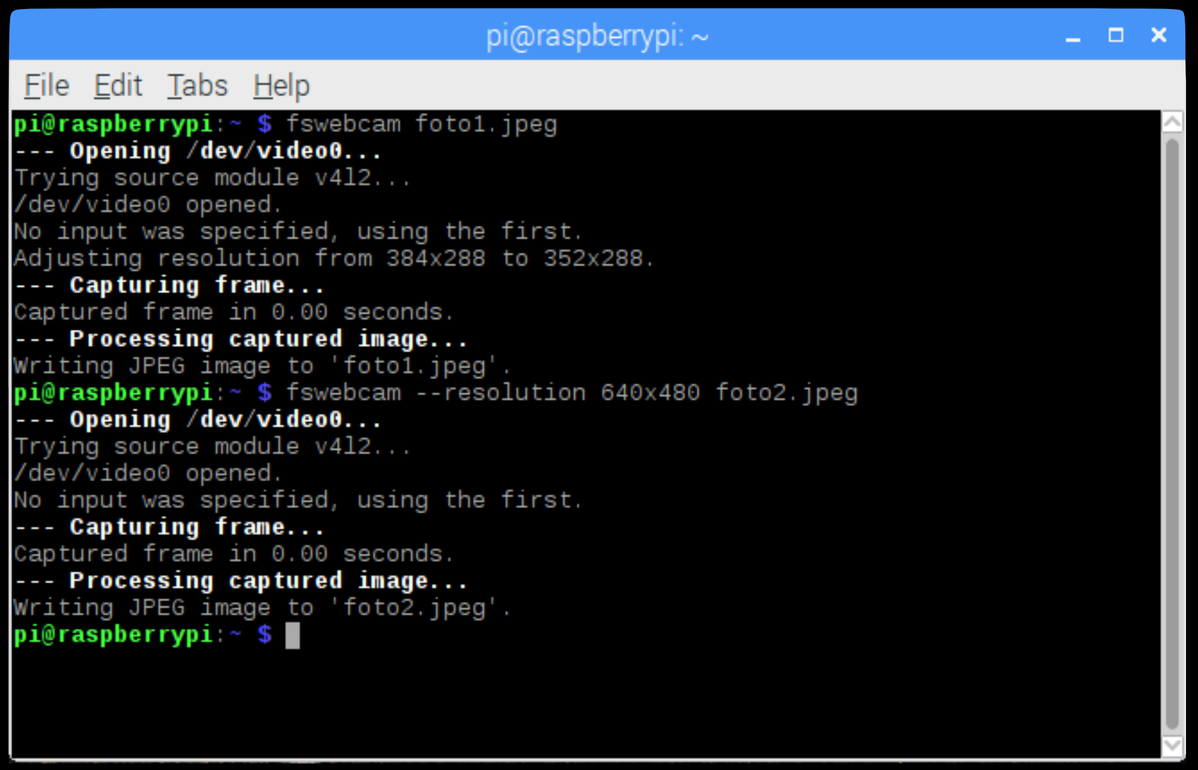
resolução padrão (352 × 288)

foto2.jpeg

resolução 640x480



Comparação entre Diferentes Resoluções



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

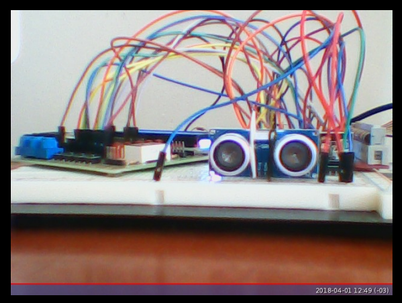


foto-2018-04-02 15:25:26.752455.jpeg

```
>>> from os import system
>>> from datetime import datetime
>>> agora = datetime.now()
>>> nome_do_arquivo = "foto-" + str(agora) + ".jpeg"
>>> comando = "fswebcam " + nome_do_arquivo
>>> system(comando)
```

Nome da Foto com Data e Hora

=

+

Microfone da Câmera Web

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.

Aplicativo ARecord

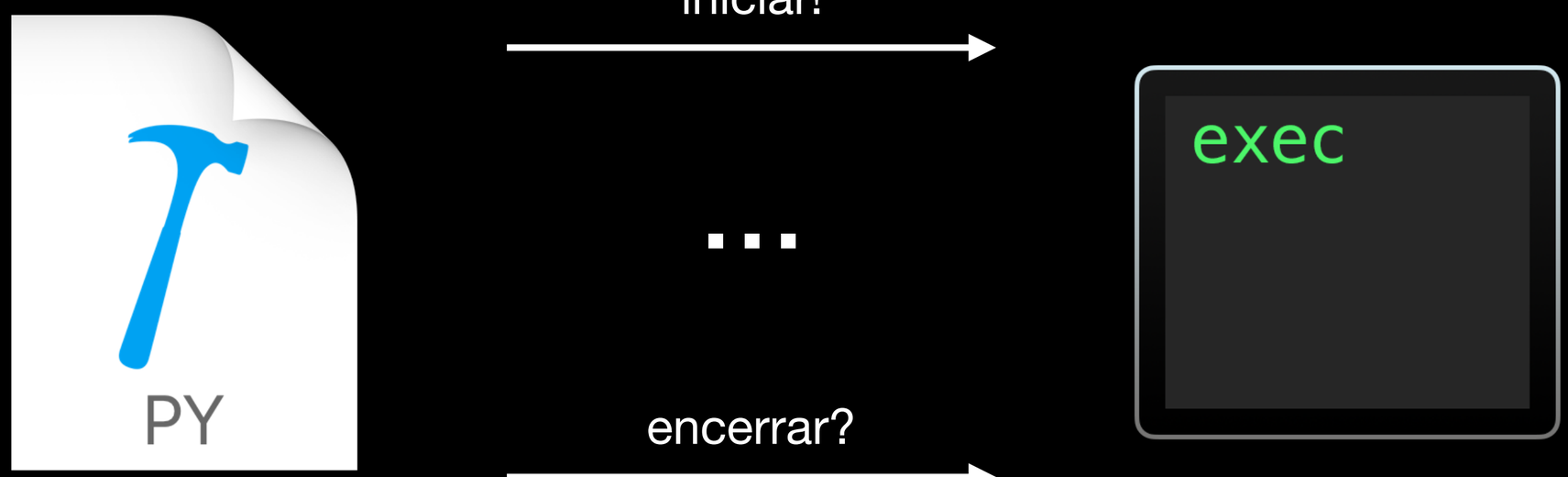
audio1.wav
(baixa qualidade)

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



Interrupção da Gravação de Áudio

```
>>> from subprocess import Popen
>>> aplicativo = None
>>> def iniciar_gravacao():
...     global aplicativo
...     comando = ["arecord", "--duration", "30", "audio.wav"]
...     aplicativo = Popen(comando)
...
```

executa arecord em plano de fundo

```
>>> def parar_gravacao():
...     if aplicativo != None:
...         aplicativo.terminate()
...
>>> iniciar_gravacao()
>>> sleep(4)
>>> parar_gravacao()
```

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

man.cx/lame

Manpages

Manpage: go

NAME

lame – create mp3 audio files

SYNOPSIS

lame [options] <infile> <outfile>

DESCRIPTION

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>.

OPTIONS

Input options:

–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 *fseek()'s* on the input file looking for WAV and AIFF headers.

Might not be available on your release.

–x Swap bytes in the input file (or output file when using **––decode).**

Available in

(1)

Contents

man.cx/opusenc

Manpages

Manpage: lame go

Available in

(1)

Contents

NAME

opusenc – encode audio into the Opus format

SYNOPSIS

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** *filename|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*

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.

Unless quieted **opusenc** displays fancy statistics about the encoding progress.

OPTIONS

General options

-h, --help

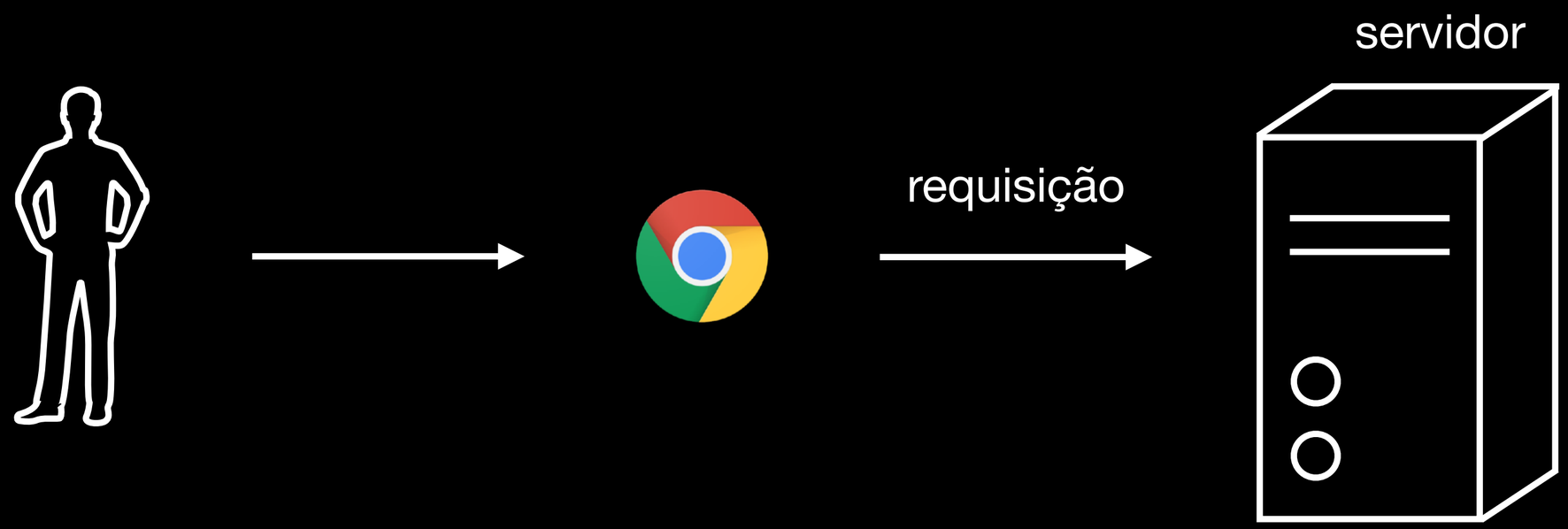
Show command help

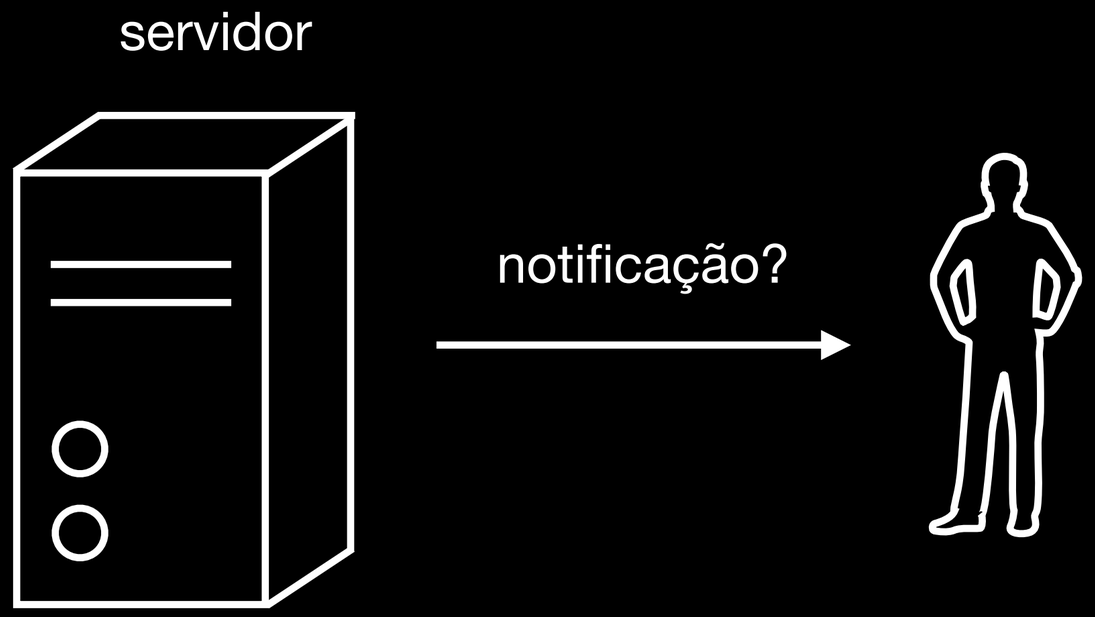
Aplicativos Lame e OpusEnc

```
>>> 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

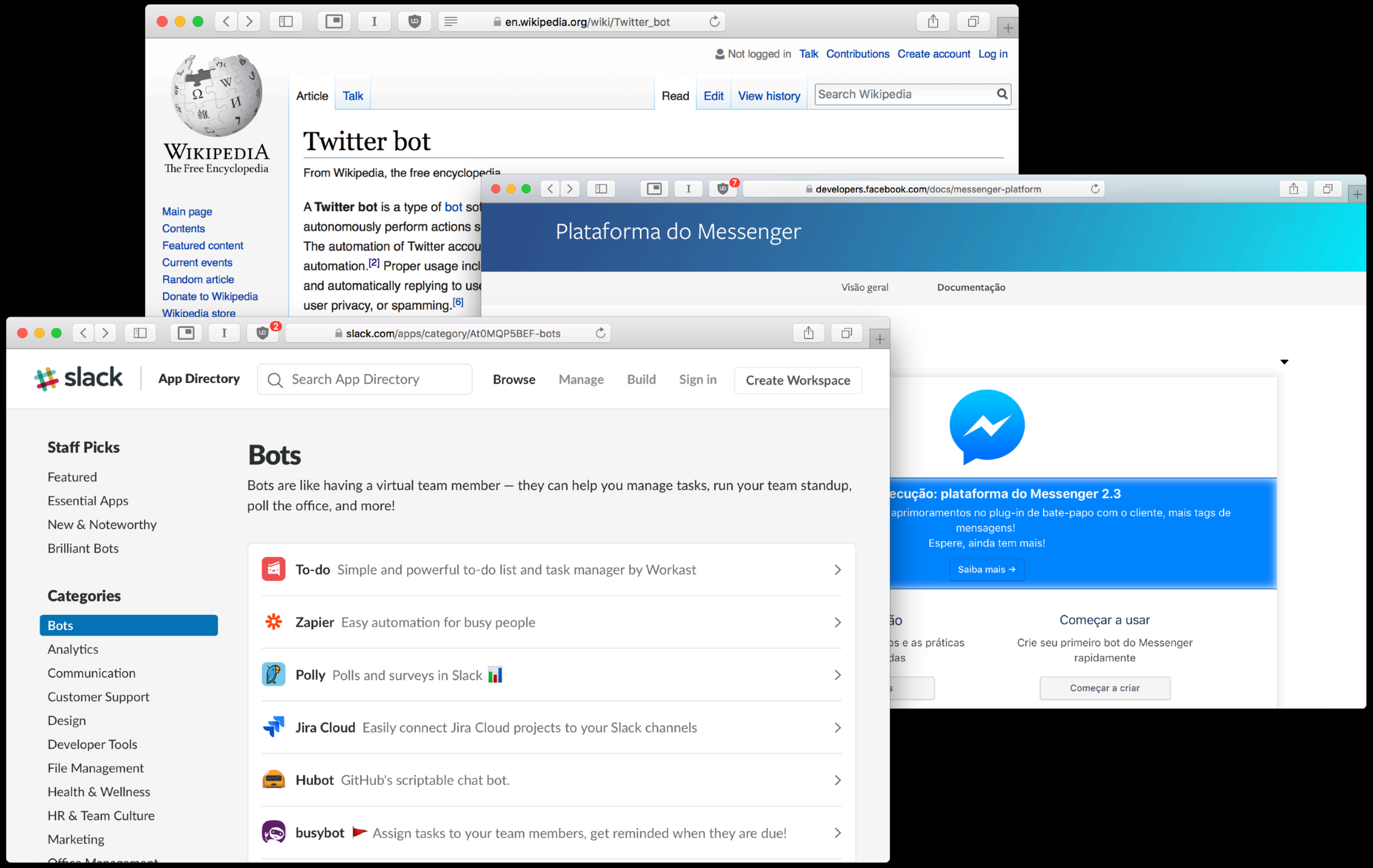




Requisição x Notificação

Notificação em Smartphones via Apps

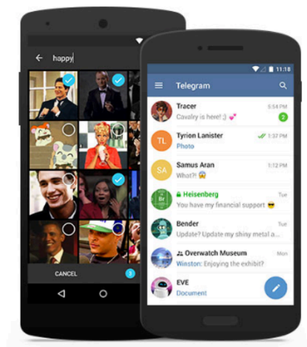
Notificações via Bots



Exemplos de Comunicação entre Bots e Apps

Telegram

telegram.org



Telegram for **Android**

Telegram for **iPhone / iPad**

Telegram for **WP**

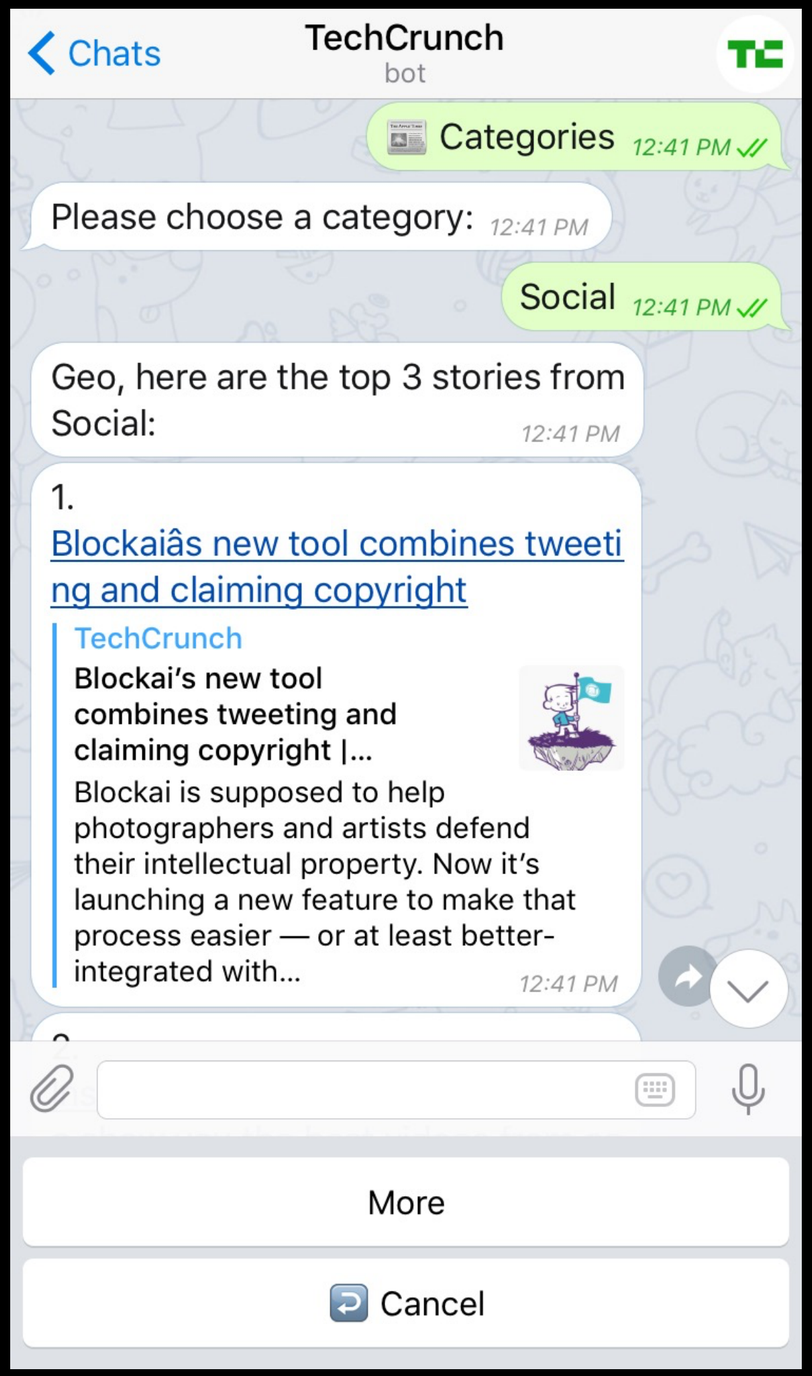
A native app for every platform

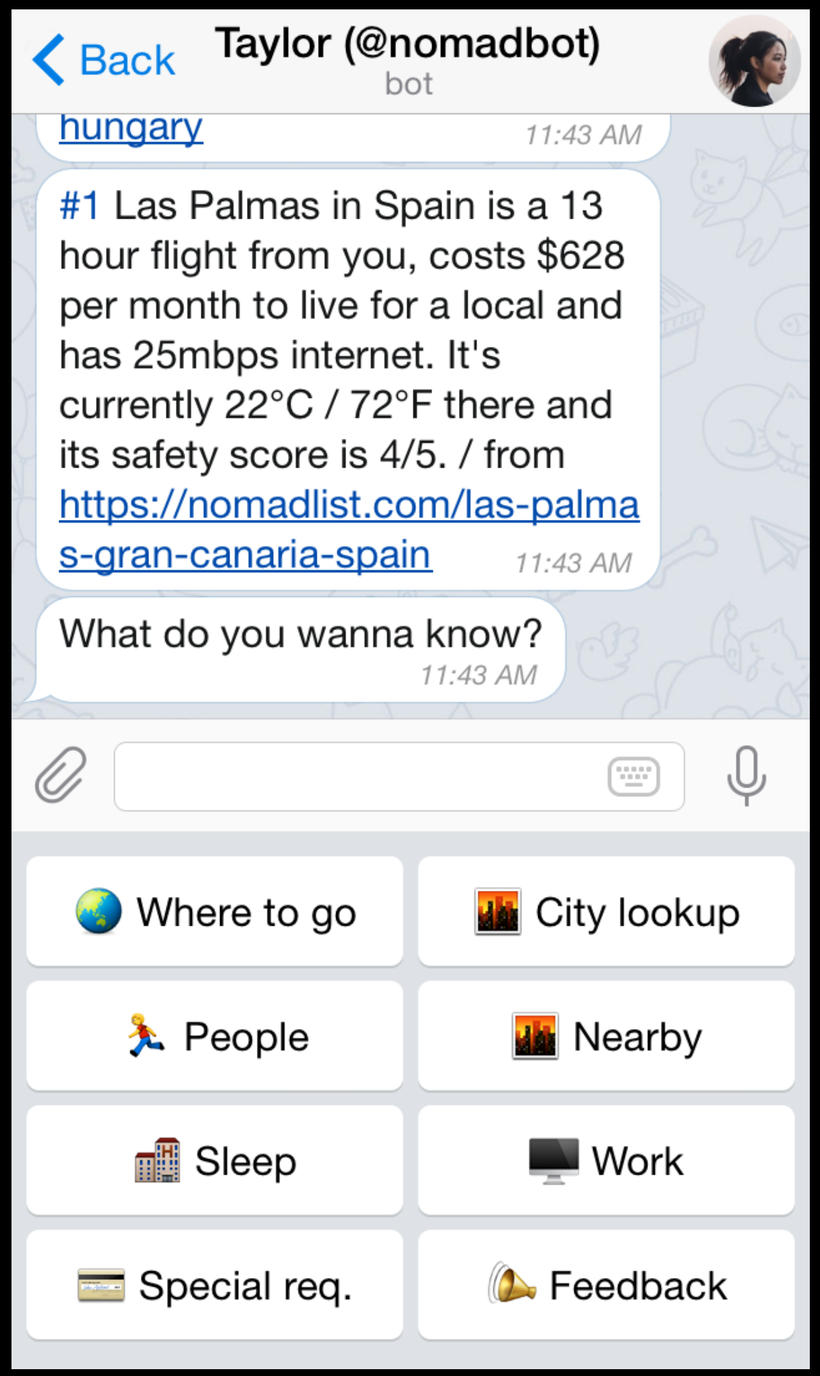
Telegram **Web-version**

Telegram for **macOS**

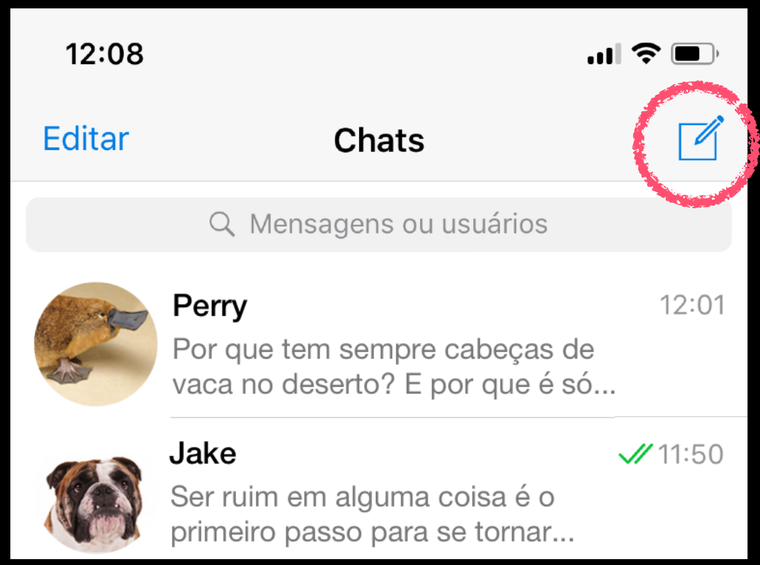
Telegram for **PC/Mac/Linux**

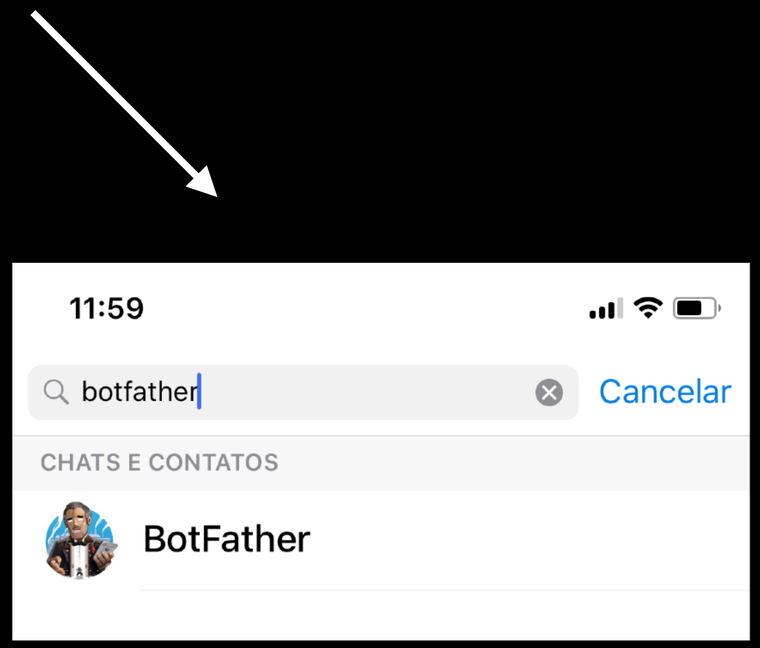
Plataformas do Telegram



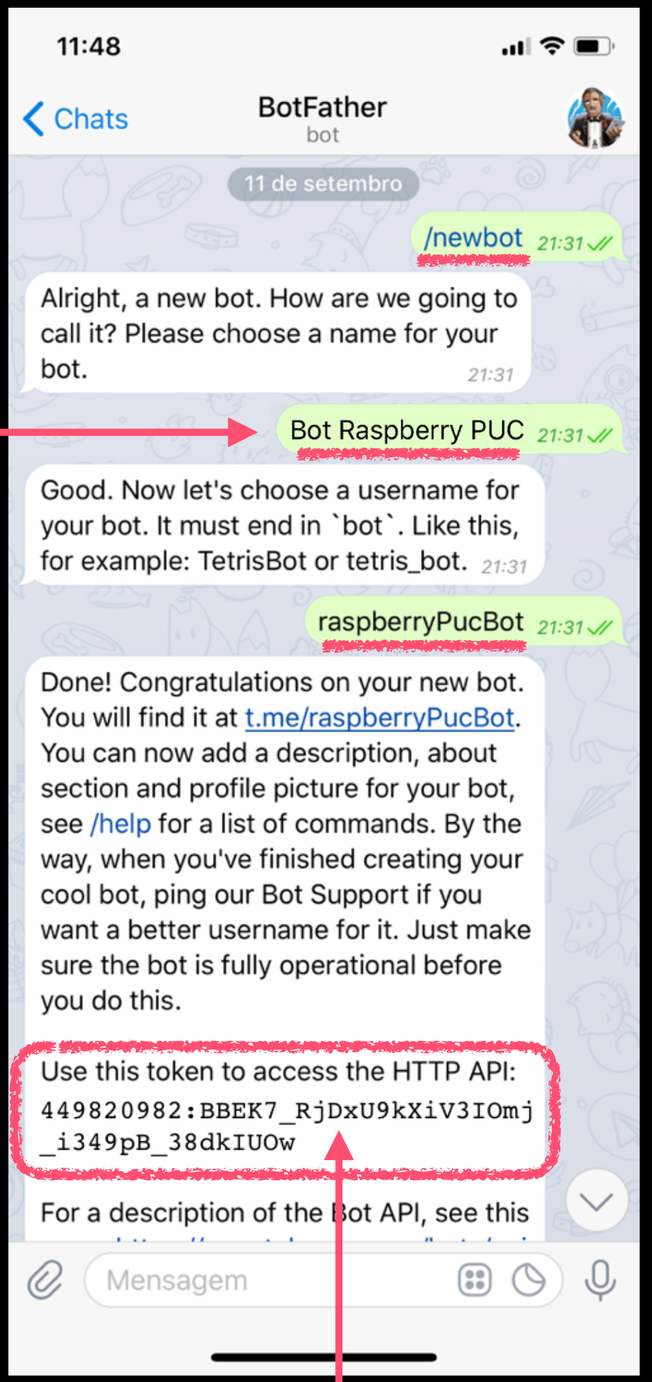


Exemplos de Conversas com Bots





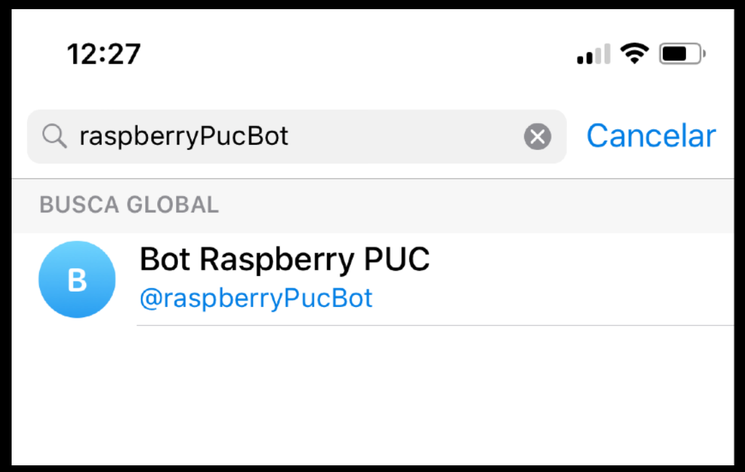
invente um nome seu!

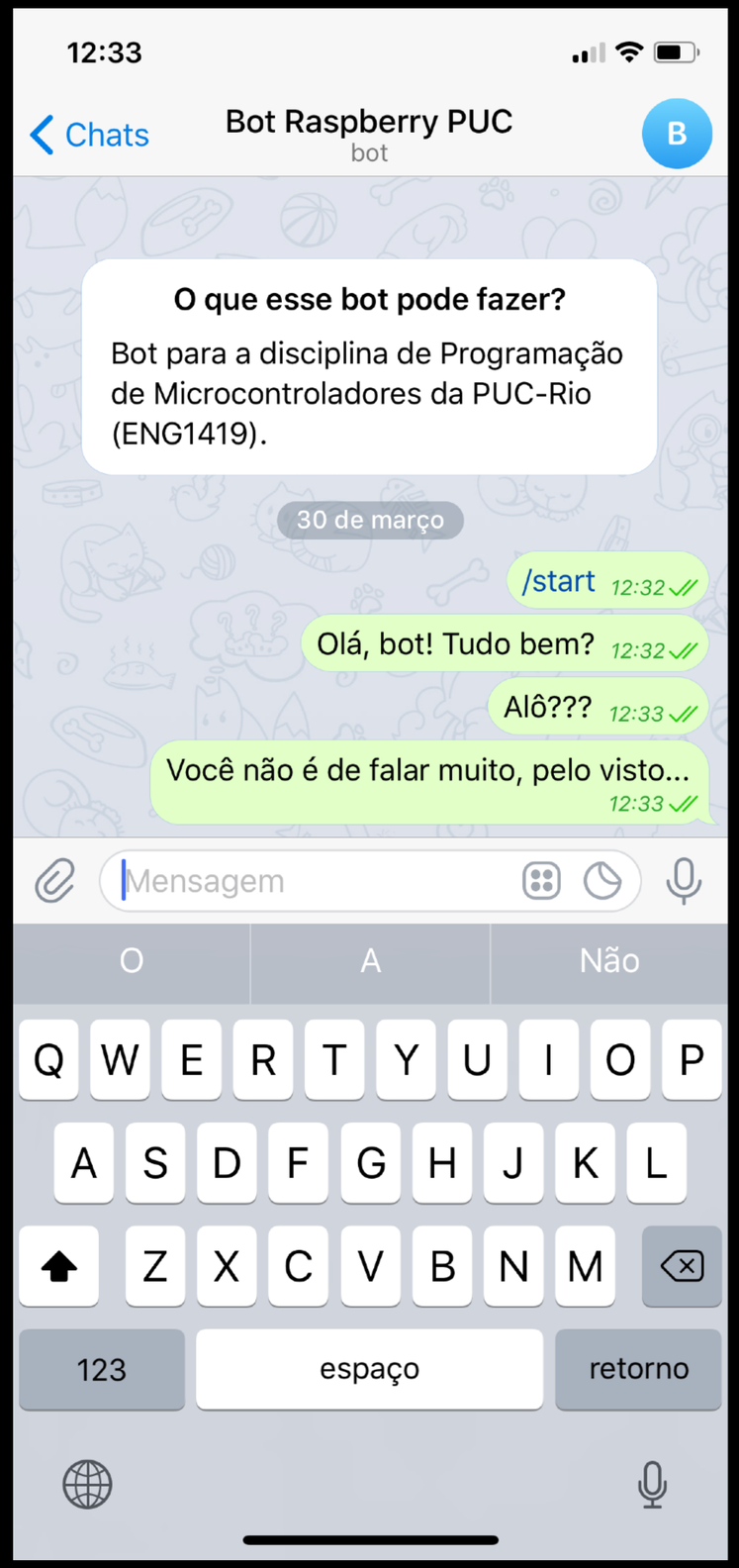


anote esta chave!

Criação de Bots no Telegram via BotFather

pesquise o nome
do SEU bot!





Conversa com um Bot

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

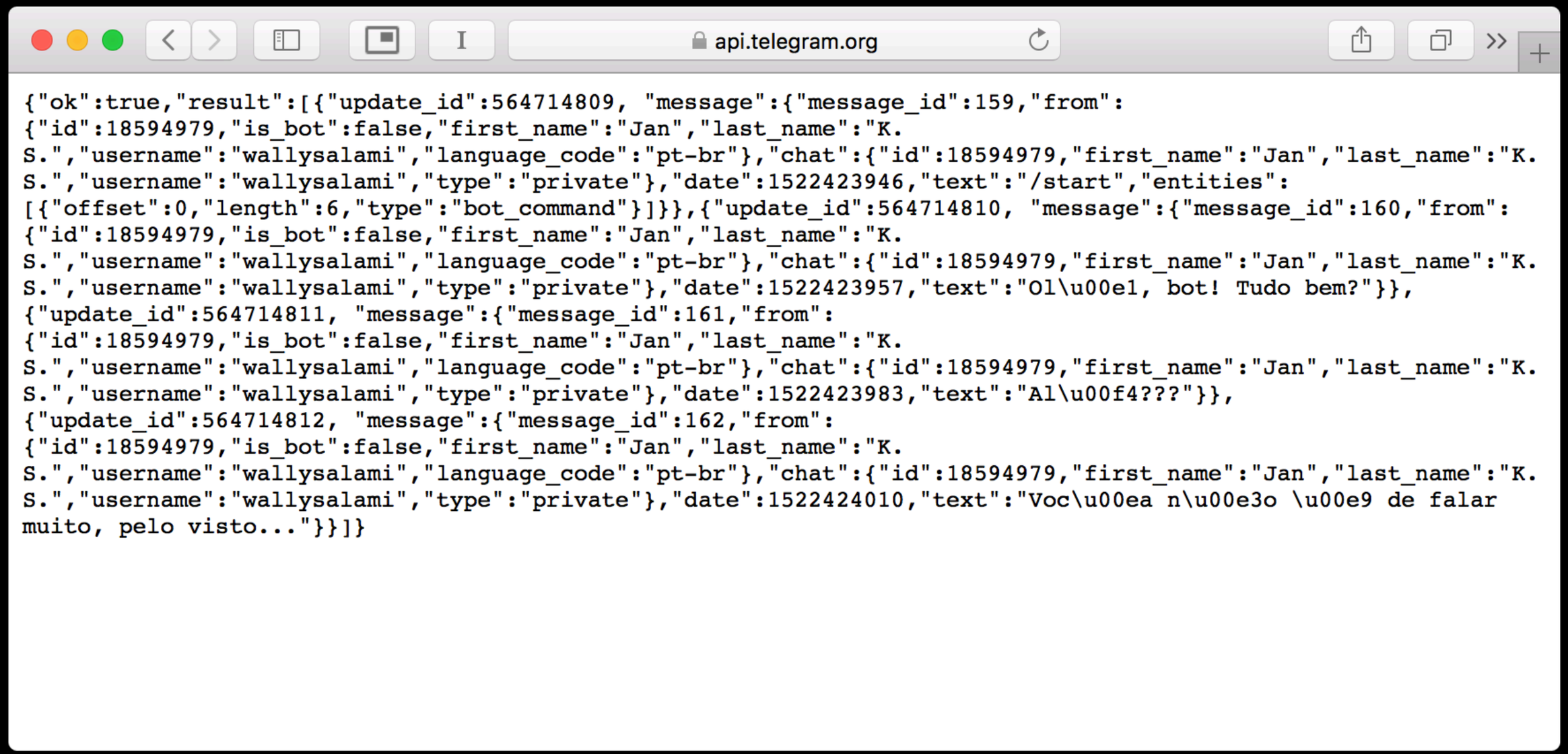
+

`dados extras`

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

Exemplos de Comandos para Bots do Telegram

api.telegram.org/bot<SUA_CHAVE_SECRETA>/getUpdates



Endereço para Obter Atualizações das Conversas com o Bot

```
{
   "ok":true,
   "result":[
      ...
      {
         "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":314253469,
               "first_name":"Jan",
               "last_name":"K. S.",
               "username":"wallysalami",
               "type":"private"
            },
            "date":1522423957,
            "text":"Ol\u00e1, bot! Tudo bem?"
         }
      },
      ...
   ]
}
```


anote este número!

Dicionário (JSON) de Resposta com as Atualizações de Mensagens

core.telegram.org/bots/api

Go up

sendMessage

Use this method to send text messages. On success, the sent Message is returned.

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 `@channelusername`)
text	String	Yes	Text of the message to be sent
parse_mode	String	Optional	Send *Markdown* or *HTML*, if you want Telegram apps to show bold, italic, fixed-width 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 silently. 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	InlineKeyboardMarkup or ReplyKeyboardMarkup or ReplyKeyboardRemove or ForceReply	Optional	Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove reply keyboard or to force a reply from the user.

Documentação de Comandos e Dados sobre Bots

docs.python-requests.org

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

Fork me on GitHub

Star 31,429

Requests is an elegant and simple HTTP library for Python, built for human beings.

Sponsored by **Linode** and other wonderful organizations.

Requests Stickers!

Stay Informed

Receive updates on new releases and upcoming projects.

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

Biblioteca para Requisições HTTP

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

Configuração Inicial para Usar o Bot

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

Envio de Mensagens de Texto

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)
```

Envio de Fotos

Foto Enviada pelo Bot na Conversa

```
>>> 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)
```

Envio de Áudio

Áudio Enviado pelo Bot na Conversa

Recebimento de Mensagens

```
>>> from requests import get
>>> endereco = endereco_base + "/getUpdates"
>>> resposta = get(endereco)
>>> dicionario_da_resposta = resposta.json()
>>> dicionario_da_resposta
{'ok': True, 'result': [{'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': 'Olá, bot! Tudo
bem?'}}, ...
```

Busca de Mensagens

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


Dicionário (JSON) com as Atualizações de Mensagens

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

Verificação dos Tipos de Mensagens

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

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

Download de um Arquivo de Chat

Busca Atualizações

```
update_id: 123000
text: "Olá!"
```

um tempo depois...

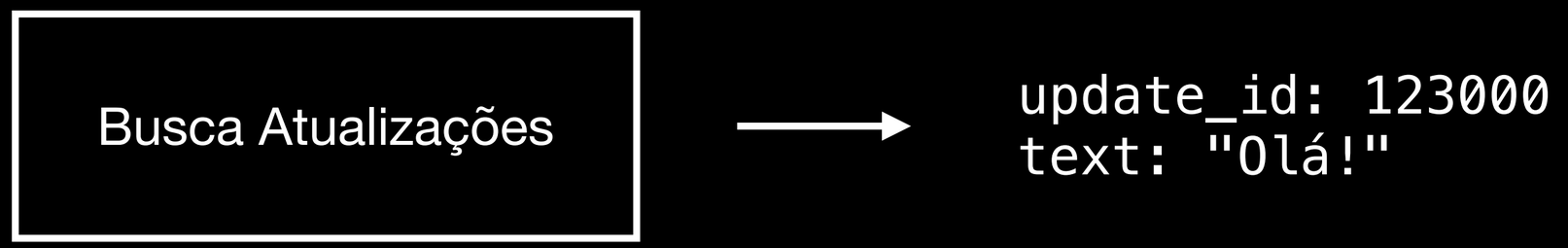
Busca Atualizações

```
update_id: 123000
text: "Olá!"
```

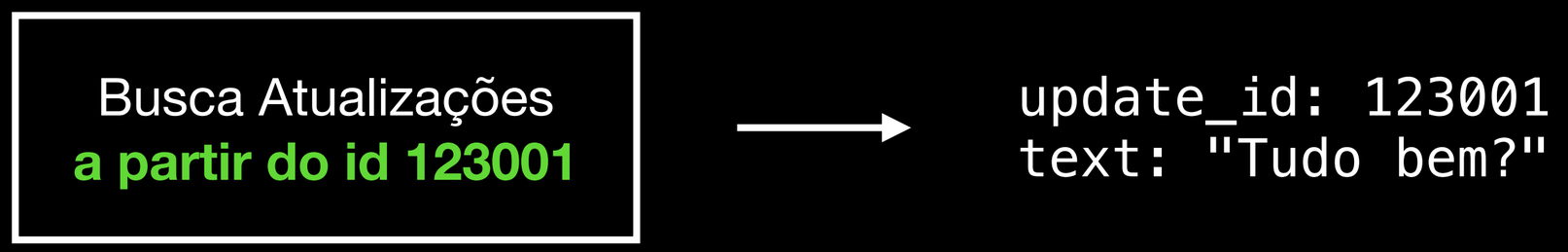
```
update_id: 123001
text: "Tudo bem?"
```

Como fazer para não vir a
mesma mensagem de novo?

Atualizações com Repetição



um tempo depois...



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 = int(resultado["update_id"]) + 1
```

Busca Contínua de Atualizações de Mensagens

Resumo da Opera

Funcionalidade	Comandos

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

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

Sensor de Distância
(documentação)
	`from gpiozero import DistanceSensor`
	`sensor = DistanceSensor(trigger=17, echo=18)`
	`sensor.distance • sensor.threshold_distance`
	`sensor.wait_for_in_range() • sensor.wait_for_out_of_range()`
	`s.when_in_range = funcao • s.when_out_of_range = funcao`

MongoDB
(documentação)
	`from pymongo import MongoClient, ASCENDING, DESCENDING`
	`cliente = MongoClient("localhost", 27017)`
	`banco = cliente["nome"] • colecao = banco["nome"]`
	`dados = {"nome": "Jan K. S.", "idade": 32}`
	`colecao.insert_one(dados) • colecao.insert_many(lista_de_dados)`
	`busca = {"chave": valor} • documento = colecao.find_one(busca)`
	`documentos = colecao.find_many(busca) • list(documentos)`
	`colecao.find_many(busca, sort=ASCENDING)`

Resumo para o Projeto 03

<table>
<tr><th>Funcionalidade</th><th>Comandos</th></tr>
<tr><td>Emissor
Infravermelho</td><td>

```
from py_irsend.irsend import *
nomes_dos_controles = list_remotes()
codigos = list_codes("mini") • send_once("mini", ["KEY_1"])
```

</td></tr>
<tr><td>Receptor
Infravermelho</td><td>

```
from lirc import init, nextcode
receptor = init("aula", blocking=False)
codigo = nextcode() • codigo == ["KEY_1"]
```

</td></tr>
<tr><td>Servidor</td><td>

```
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=True)
```

</td></tr>
<tr><td>HTML</td><td>

```
<p>Parágrafo</p>
<img src="site.com/imagem">
<a href="/pagina">Link</a>
```

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

</td></tr>
<tr><td>Ngrok</td><td>

```
abrir Terminal → ngrok http 5000
```

</td></tr>
</table>

Resumo para o Projeto 02

Funcionalidade	Comandos

LED	`from gpiozero import LED • led = LED(porta_da_GPIO)`
	`led.on() • led.off() • led.toggle() • led.is_lit`
	`led.blink() • led.blink(n=4, on_time=0.5, off_time=2)`
	mais funções na documentação oficial

Botão	`from gpiozero import Button • botao = Button(porta_da_GPIO)`
	`botao.is_pressed • led.wait_for_press()`
	`botao.when_pressed = funcao`
	`botao.when_held = funcao • botao.when_released = funcao`
	mais funções na documentação oficial

LCD	`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()`
	mais funções no exemplo do repositório oficial

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.duration • player.pt_step(-1)`
	`player.metadata["Title"] • player.metadata["Artist"]`
	`player.volume = 70 • player.speed = 2`

Resumo para o Projeto 01

Funcionalidade	Comandos
Entrada / Saída	`x = input("Digite um número: ") • print("Resultado: ", x)`

Listas	`lista = [1, 2, 3] • lista2 = ["texto", [0, 0], 5];`
	`lista[0] • lista.append(elemento) • lista.remove(indice)`
	mais funções na documentação oficial

Dicionários	`dicionario = {"chave 1": 42, "chave 2": [1, 2, 3]}`
	`dicionario["chave 1"] • dicionario["chave 3"] = "Olá!"`
	mais funções na documentação oficial

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

Condicionais

```
if x == 0:
    y = 4
```

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

```
if x != 0
    y = 4
elif x >= 0:
    y = 3
else:
    y = 0
```

Repetições

```
for i in [1, 2, 3]:
    ...
```

```
for i in range(1, 4):
    ...
```

```
while x > 1:
    ...
```

Criação de Funções

```
def funcao1(x):
    return x + 2
```

```
def funcao2(x, y, z):
    ...
```

```
def funcao3():
    ...
```

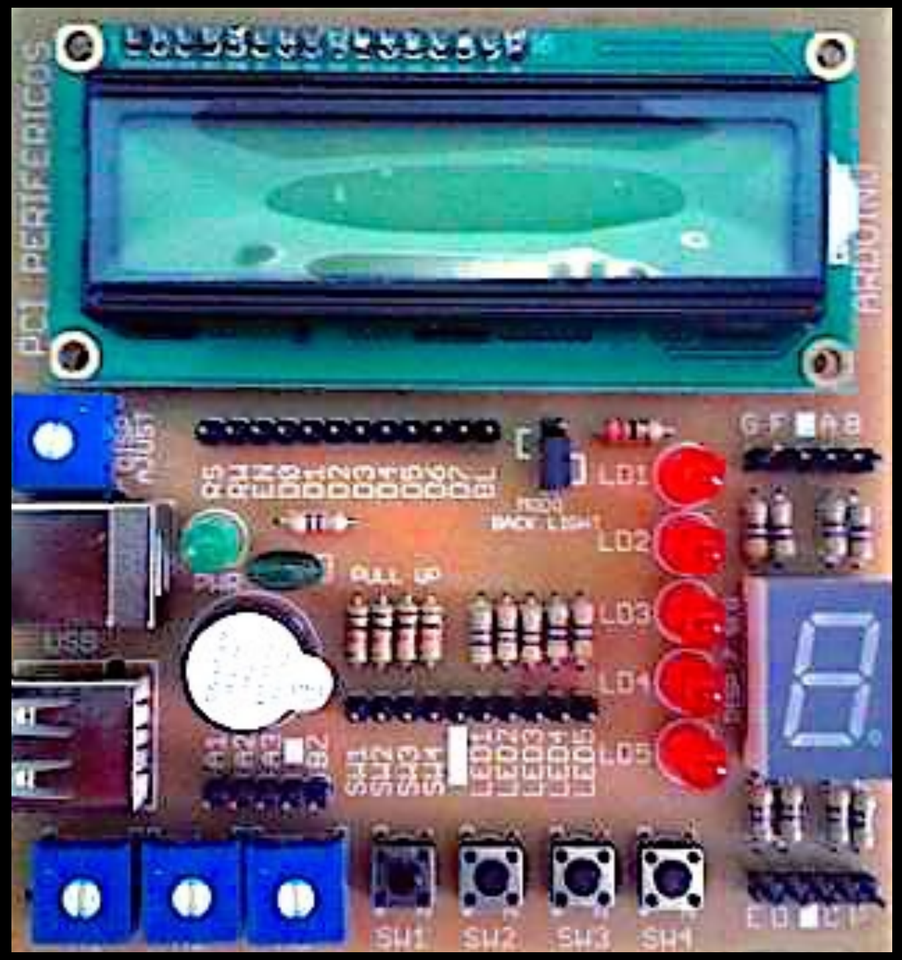
Prática

janks.link/micro/projeto04.zip

Material do Projeto 04

Testes Iniciais

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



GPIO 16

GPIO 21
GPIO 22
GPIO 23
GPIO 24
GPIO 25

GPIO 11, 12, 13, 14

Conexões com as Portas da GPIO

Testes Iniciais

Ao apertar o botão 1, exiba o texto "Gravando..." no LCD, grave 5 segundos de áudio e apague o texto. Abra o arquivo manualmente para ouvir a gravação.
↪ DICA: use a função **`system`**.

Ao apertar o botão 2, tire 5 fotos com a webcam, com 2 segundos de intervalo entre elas. Abra o 5 arquivos manualmente para visualizar as imagens.
↪ DICA: use um for, variando o nome do arquivo.

Pisque 1 vez rapidamente o LED 1 após capturar cada foto no item anterior.

Configure o seu bot no Telegram.
↪ DICA: você pode usar o Telegram pelo navegador em web.telegram.org.

Ao apertar o botão 3, envie uma mensagem de texto qualquer para a conversa com o bot.

```
>>> endereco = endereco_base + "/sendMessage"
>>> dados_incompletos = {"text": "Olá!"}
>>> resposta = post(endereco, data=dados_incompletos)
>>> print(resposta.text)
{"ok":false,"error_code":400,"description":"Bad
Request: chat_id is empty"}
```

DICA: Mensagens de Erro do Telegram (Caso Algo Não Esteja Funcionando)

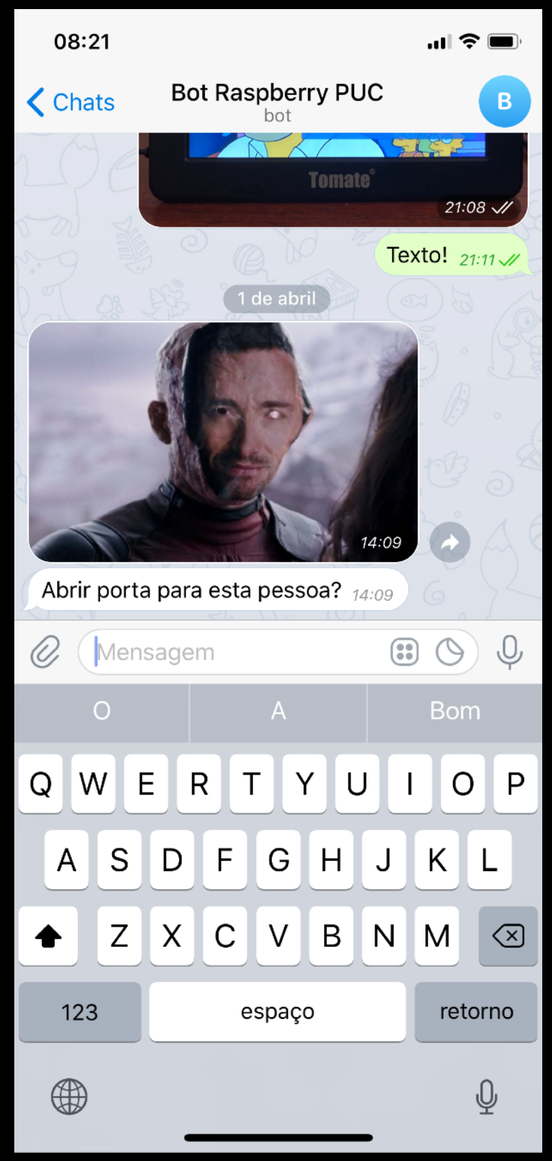
Implementação

Visitantes?

Controle de Acesso para Visitantes

notificação com foto

comando



Controle de Acesso por Notificação

Implementação

Ao apertar o botão 1, envie uma mensagem de alerta para o chat.

Após o envio do alerta, capture e envie uma foto tirada pela webcam.

Ao receber uma mensagem "Abrir", acenda o LED 1 (representando a abertura da porta). Ao receber "Alarme", toque a campainha 5 vezes rapidamente.

Ao apertar o botão 2, apague o LED 1 (representando o fechamento da porta).

```
...

print("Enviando foto...")

...

print("Foto enviada!")
```

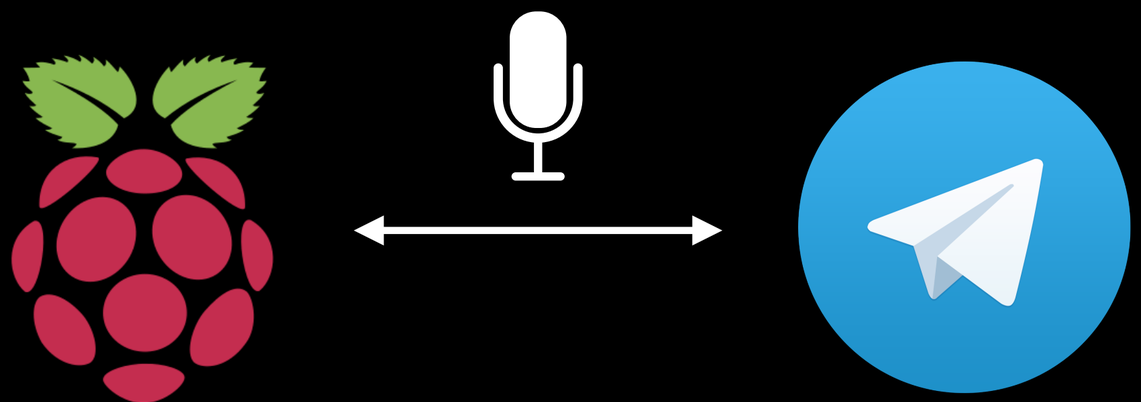
```
...

print("Enviando mensagem de texto...")

...
```

DICA: Acompanhamento das Etapas com Print

Aperfeiçoamento



Troca de Áudio e Teclado de Respostas Pré-Definidas

Aperfeiçoamento

Ao pressionar o botão 3, comece a gravar um áudio. Ao soltar, finalize a gravação, converta o arquivo para OGG e envie-o para o chat.
↳ DICA: use o **`POpen`**.

Ao receber um arquivo de áudio do chat, toque-o no Raspberry Pi.
↳ DICA: use o **`mplayer`**.

Após enviar a foto da webcam, configure opções pré-definidas de resposta para "Abrir", "Soar Alarme" e "Ignorar".
↳ DICA: envie um dicionário como parâmetro extra na `sendMessage`, conforme a documentação em core.telegram.org/bots/api#sendmessage.

```
dados = {"chat_id": ..., "text": ..., "reply_markup": ...}
```