

Pin pong Telegram

Como paso previo a enviar y recibir mensajes, vamos a realizar los pasos de este vídeo

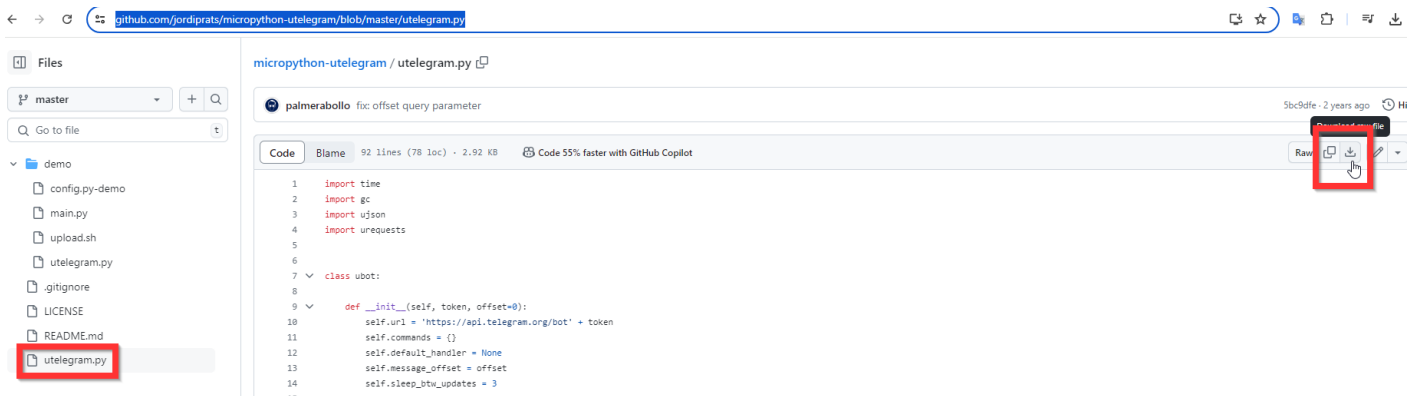
<https://www.youtube.com/watch?v=eZkb9omr-sA>

<https://www.youtube.com/embed/eZkb9omr-sA>

Paso 1: Librería uTelegram.py

Del repositorio de Jordi Prats

<https://github.com/jordiprats/micropython-utelegram/blob/master/utelegram.py>



```
import time
import gc
import ujson
import urequests
```

```
class ubot:
```

```
    def __init__(self, token, offset=0):
        self.url = 'https://api.telegram.org/bot' + token
```

```
self.commands = {}
self.default_handler = None
self.message_offset = offset
self.sleep_btw_updates = 3

messages = self.read_messages()
if messages:
    if self.message_offset==0:
        self.message_offset = messages[-1]['update_id']
    else:
        for message in messages:
            if message['update_id'] >= self.message_offset:
                self.message_offset = message['update_id']
                break

def send(self, chat_id, text):
    data = {'chat_id': chat_id, 'text': text}
    try:
        headers = {'Content-type': 'application/json', 'Accept': 'text/plain'}
        response = urequests.post(self.url + '/sendMessage', json=data, headers=headers)
        response.close()
        return True
    except:
        return False

def read_messages(self):
    result = []
    self.query_updates = {
        'offset': self.message_offset + 1,
        'limit': 1,
        'timeout': 30,
        'allowed_updates': ['message']}

    try:
        update_messages = urequests.post(self.url + '/getUpdates', json=self.query_updates).json()
```

```
if 'result' in update_messages:
    for item in update_messages['result']:
        result.append(item)
    return result
except (ValueError):
    return None
except (OSError):
    print("OSError: request timed out")
    return None
```

```
def listen(self):
    while True:
        self.read_once()
        time.sleep(self.sleep_bt看_updates)
        gc.collect()

def read_once(self):
    messages = self.read_messages()
    if messages:
        if self.message_offset==0:
            self.message_offset = messages[-1]['update_id']
            self.message_handler(messages[-1])
        else:
            for message in messages:
                if message['update_id'] >= self.message_offset:
                    self.message_offset = message['update_id']
                    self.message_handler(message)
                    break

def register(self, command, handler):
    self.commands[command] = handler

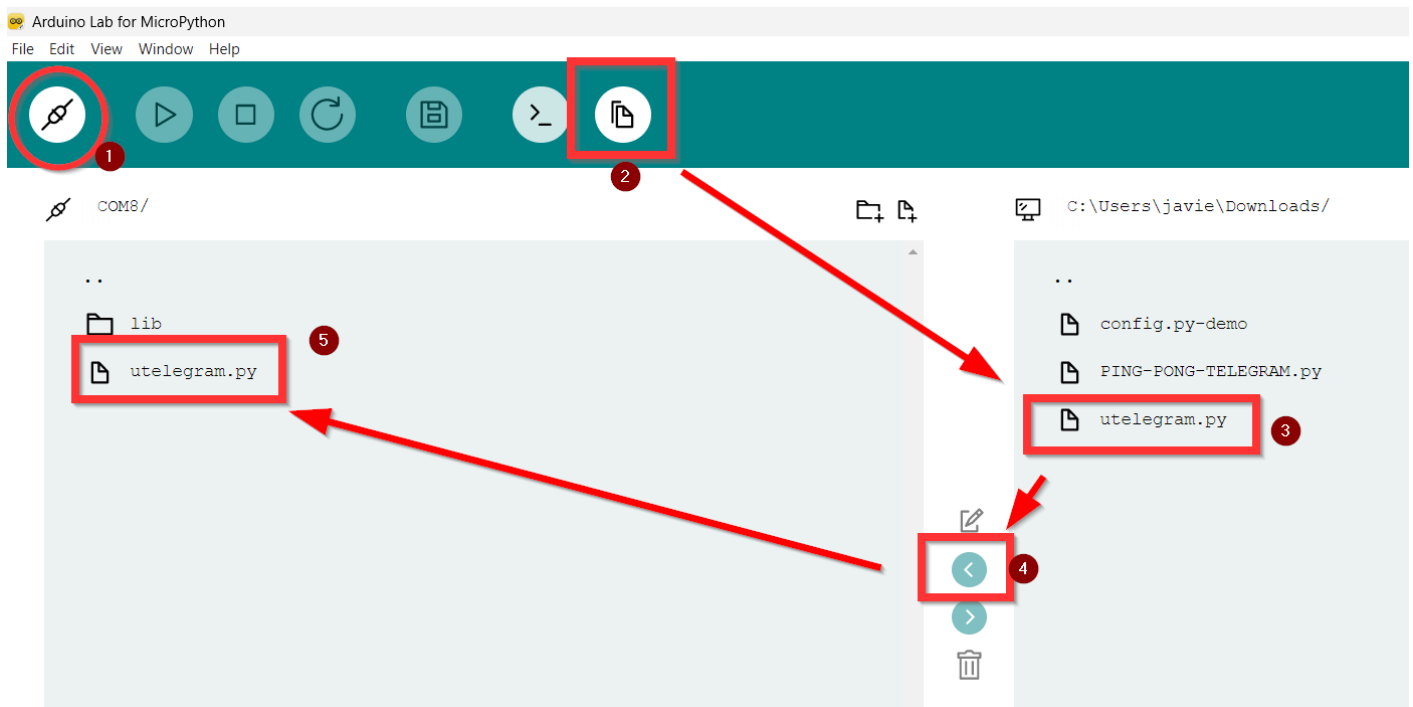
def set_default_handler(self, handler):
    self.default_handler = handler

def set_sleep_bt看_updates(self, sleep_time):
```

```
self.sleep_btw_updates = sleep_time
```

```
def message_handler(self, message):  
    if 'text' in message['message']:  
        parts = message['message']['text'].split(' ')  
        if parts[0] in self.commands:  
            self.commands[parts[0]](message)  
        else:  
            if self.default_handler:  
                self.default_handler(message)
```

Y la cargamos dentro de nuestro ESP32, ejecutamos Arduino Lab for MicroPython, conectamos, vamos al gestor de archivos y lo llevamos dentro del ESP32 Alvik

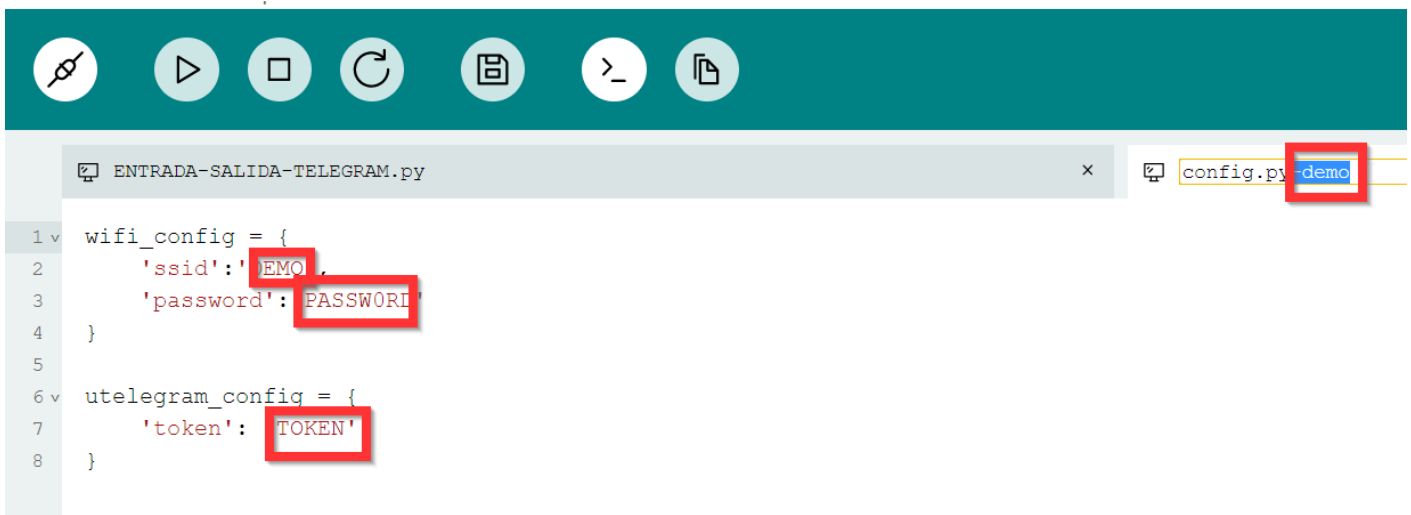


Paso 2 Archivo config.py

El archivo config.py no es más que el archivo que contiene la wifi y el token, se puede descargar de <https://github.com/jordiprats/micropython-utelegram/blob/master/demo/config.py-demo> o también se puede copiar y pegar de aquí mismo

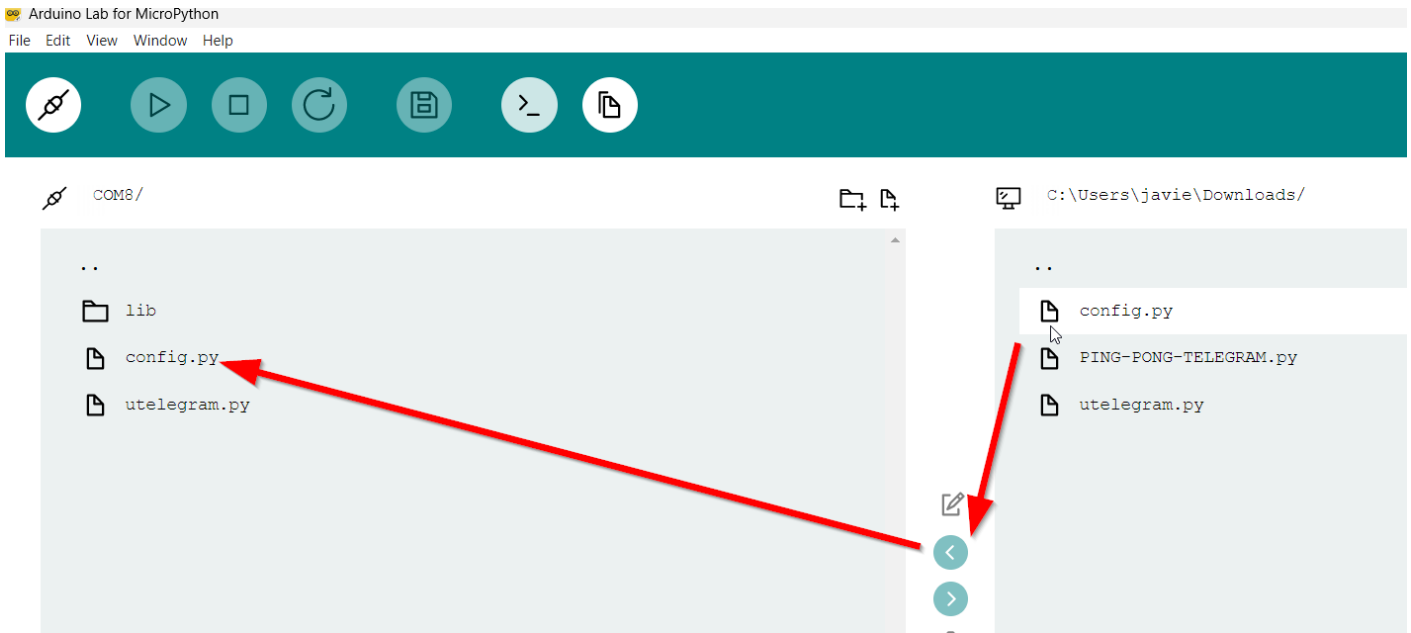
```
wifi_config = {  
    'ssid':'DEMO',  
    'password':'PASSWORD'  
}  
  
utelegram_config = {  
    'token': 'TOKEN'  
}
```

Ponemos los valores de nuestra wifi SSID, PASSWORD y TOKEN y borramos del nombre el -demo y lo dejamos como config.py



```
ENTRADA-SALIDA-TELEGRAM.py x config.py-demo  
1 v wifi_config = {  
2     'ssid':'DEMO',  
3     'password':'PASSWORD'  
4 }  
5  
6 v utelegram_config = {  
7     'token': 'TOKEN'  
8 }
```

y como antes, lo pasamos al ESP32 Alvik



Se podría poner esa información en el código del programa principal main.py tal y como el programa de la página <https://libros.catedu.es/books/arduino-alvik/page/mensajes-a-telegram>

Paso 3 Programa principal main.py

El programa lo podemos descargar de <https://github.com/jordiprats/micropython-utelegram/blob/master/demo/main.py>

o de aquí mismo

tal cual, no hay que poner nuestro ssid, ni password ni token pues lo "lee" de config.py

```
from config import utelegram_config
from config import wifi_config

import utelegram
import network
import utime
```

```
debug = True

sta_if = network.WLAN(network.STA_IF)
sta_if.active(True)
sta_if.scan()
sta_if.connect(wifi_config['ssid'], wifi_config['password'])

if debug: print('WAITING FOR NETWORK - sleep 20')
utime.sleep(20)

def get_message(message):
    bot.send(message['message']['chat']['id'], message['message']['text'].upper())

def reply_ping(message):
    print(message)
    bot.send(message['message']['chat']['id'], 'pong')

if sta_if.isconnected():
    bot = utelegram.ubot(utelegram_config['token'])
    bot.register('/ping', reply_ping)
    bot.set_default_handler(get_message)

    print('BOT LISTENING')
    bot.listen()
else:
    print('NOT CONNECTED - aborting')
```

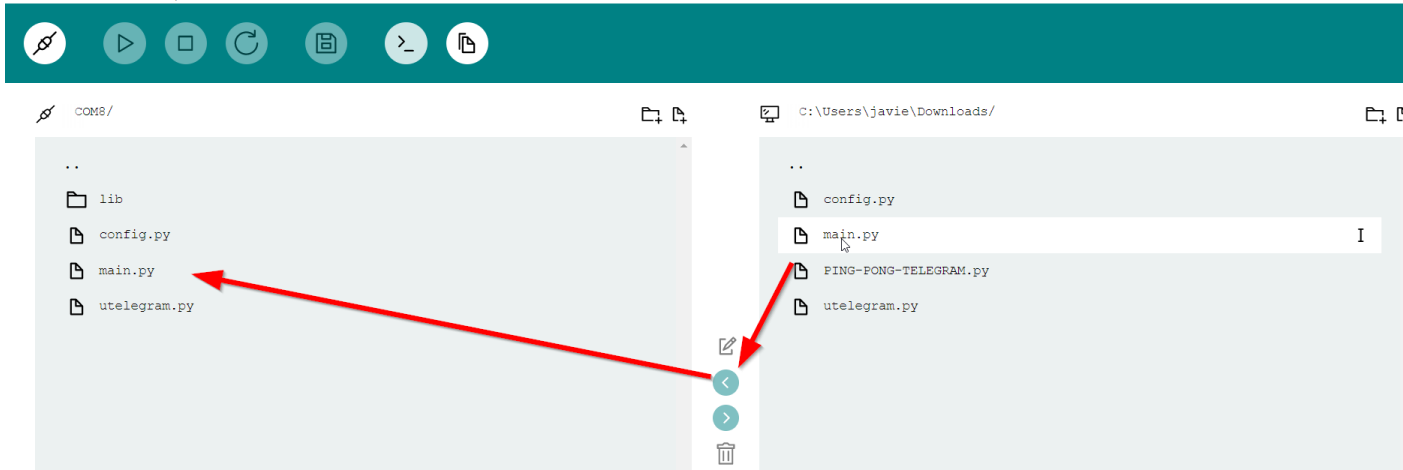
❏ No sé por qué hay que esperar 20 segundos en `utime.sleep(20)` ❏ ¿sospecho que necesita tiempo para estar preparado para "escuchar"

Y lo llevamos al ESP32



Arquino Lab for micro:python

File Edit View Window Help



Ejecución

Pulsamos el main.py del ESP32 (no hace falta encender Alvik pues todas las instrucciones son sólo del ESP32), ESPERAR 20 SEGUNDOS hasta que aparezca BOT LISTENING



Arduino Lab for MicroPython

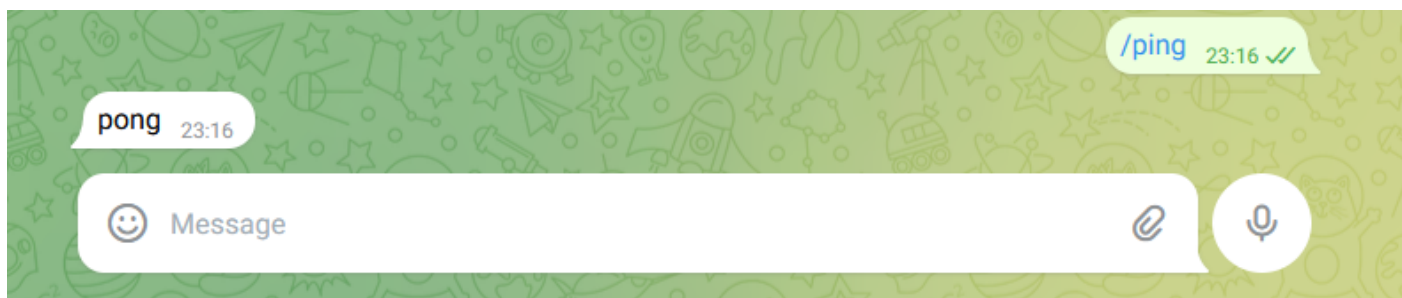
File Edit View Window Help

Run

```
main.py
1
2 from config import utelegram_config
3 from config import wifi_config
4
5 import utelegram
6 import network
7 import utime
8
9 debug = True
10
11 sta_if = network.WLAN(network.STA_IF)
12 sta_if.active(True)
13 sta_if.scan()
14 sta_if.connect(wifi_config['ssid'], wifi_config['password'])
```

```
Type "help()" for more information.
>>>
>>>
>>>
>>>
MicroPython v1.23.0 on 2024-06-02; Arduino Nano ESP32 with ESP32S3
Type "help()" for more information.
>>>
raw REPL: CTRL-B to exit
>OKWAITING FOR NETWORK - sleep 20
BOT LISTENING
```

Nos vamos a Telegram al usuario del bot que hemos creado, le tecleamos **/ping** y contesta el ESP32 **pong**



<https://www.youtube.com/embed/eZkb9omr-sA>

Revision #8

Created 6 July 2024 22:07:45 by Javier Quintana

Updated 6 September 2024 19:29:24 by Javier Quintana