

# MicroPython con IoT

- [¿Qué es Internet de las cosas IoT?](#)
- [Conectar a Wifi](#)
- [Crear bot en Telegram](#)
- [Encontrar tu ID en Telegram](#)
- [Mensajes a Telegram](#)
- [Pin pong Telegram](#)
- [Recepción mensajes Telegram](#)



Estamos hablando de dispositivos que se conectan a internet de forma desatendida, por vía hardware (o mejor dicho firmware) a diferencia de un ordenador, tablet o móvil, donde tienes que configurar por software el dispositivo y hay un diálogo entre usuario y dispositivo sobre el uso de Internet (el software solicita tal página web, tales datos etc por voluntad del usuario o por diálogo con el usuario) Aquí los dispositivos están ya configurados de los datos que se comunican. Es decir "conectar y olvidar".

Piensa en la diferencia entre un enchufe inteligente y un ordenador, el primero es lo que se considera dentro de IoT

Las formas "desatendidas" son un avance en la sociedad pero también puede generar problemas muy serios a nivel mundial, [ver el caso Mirai](#)

### Las cosas claras. ¿asíncrono o síncrono?

Hay muchas herramientas IoT

- **Blynk:** lo que nos gusta de esta herramienta es que es casi "instantánea" o "síncrona". Esto es imprescindible con ciertos robots como el **Rover Marciano con Arduino**. Necesitamos que "gire" para evitar un obstáculo, no podemos esperar !!! Veremos con **BLYNK** un protocolo que entre el dispositivo electrónico (nuestro robot) y nosotros (en ordenador, en una APP en el móvil) la comunicación es instantánea, gracias a un servidor que hará de intermedio, que puede ser local (BLYNK LEGACY) o en Internet (BLYNK IoT).
  - **Blynk legacy** es la que se va a trabajar en
    - [Rover Marciano con Arduino](#)
    - [Arduinoblocks en el aula](#)
    - [ESP32 en el aula](#)
  - **Blynk IoT** es la que se va a trabajar con
    - [En ESP32 en el aula](#)
- **MQTT** El emisor envía datos, se almacenan en un servidor, y cuando puede, lo vuelca al cliente. Cliente y emisor pueden ser el dispositivo electrónico y nosotros o viceversa. Veremos que esto es lo que hace el protocolo **MQTT** y está tremendamente extendido por lo barato y fácil que es. Hace que los servidores no estén tan ocupados, por lo tanto hay varios proveedores que ofrecen este servicio gratuitamente. Hay robots como los que tienen la placa **TDR STEAM IMAGINA** que envía datos de temperatura, humedad, .. y pueden recibir datos pero no precisan de esta exigencia instantánea como un rover.
  - [ESP32 EN EL AULA](#)
- **Arduino cloud IoT**
  - [Arduino Alvik](#)



- **Cyberpi y mBot2**
  - [lot con Cyberpi](#)

# Conectar a Wifi

Para ello necesitamos importar la librería network, crear un objeto network que se conecta a la wifi :

```
import network
import urequests

WIFI_NETWORK='NOMBREREDWIFI'  ## tu red wifi
WIFI_PASSWORD='CONTRASENA'  ## la contraseña de la red wifi

wlan = network.WLAN(network.STA_IF)
wlan.active(True)
wlan.connect(WIFI_NETWORK, WIFI_PASSWORD)

print("\nconectando.....")
if wlan.isconnected():
    print("Alvik se conectó a",WIFI_NETWORK)
else :
    print ("Alvik NO se conectó a",WIFI_NETWORK)
```

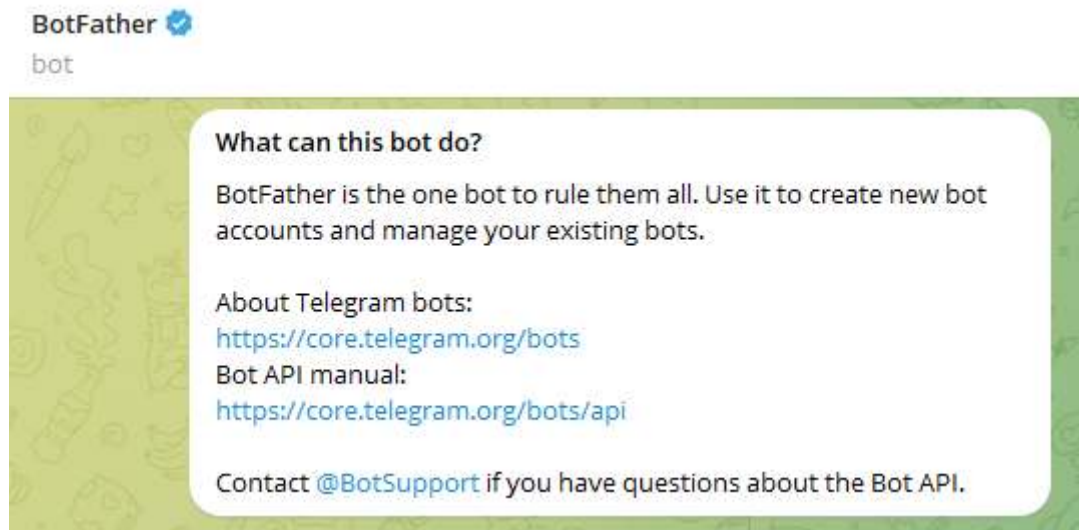
Tiene que salir este mensaje:

```
>>>
raw REPL; CTRL-B to exit
>OK
conectando.....
Alvik se conectó a NOMBREREDWIFI
>
```

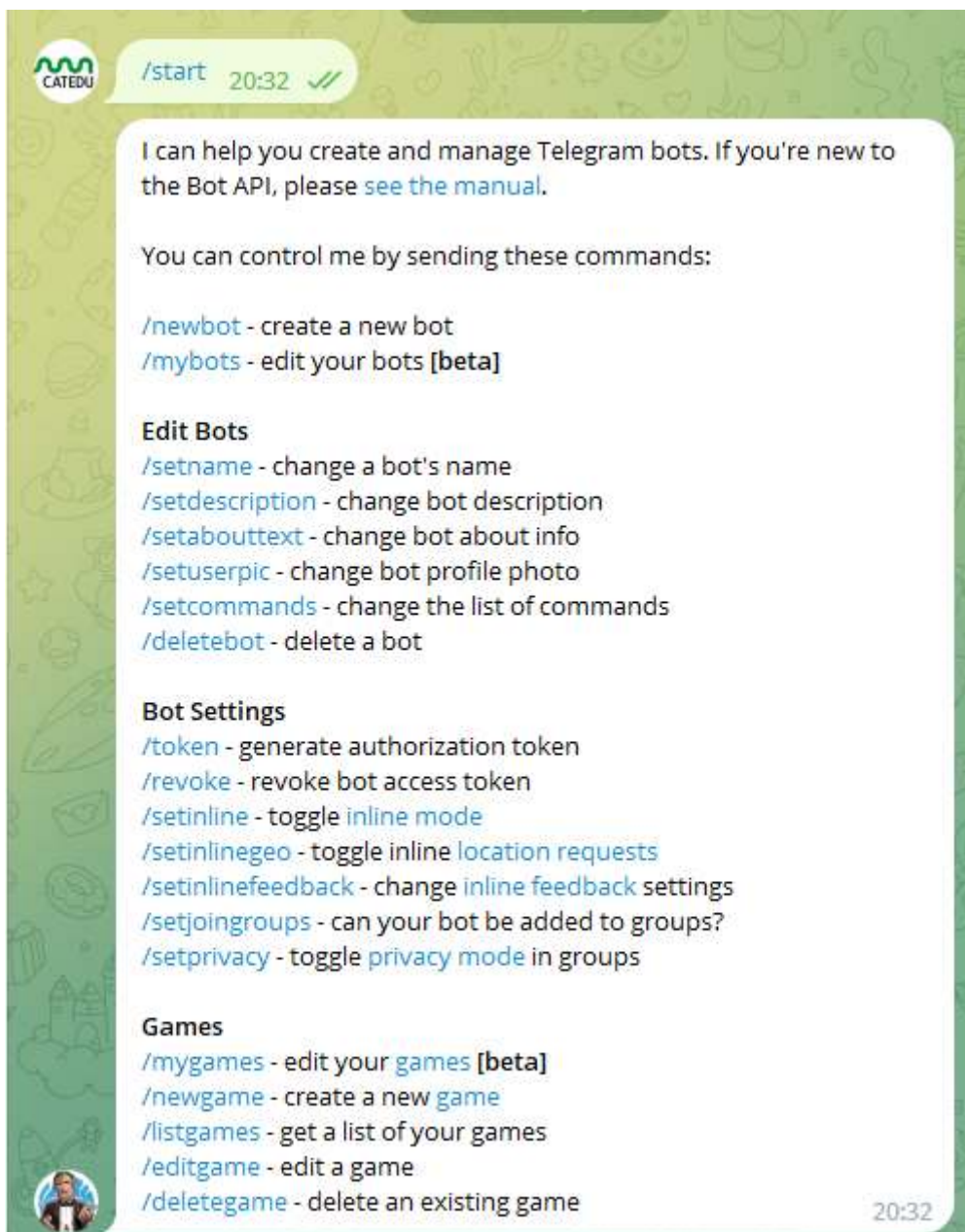
# Crear bot en Telegram



Entramos en nuestro Telegram y chateamos con el creador de los bots: **@BotFather** y nos saldrá esta pantalla:



Si tecleamos **/start** nos sale las diferentes opciones

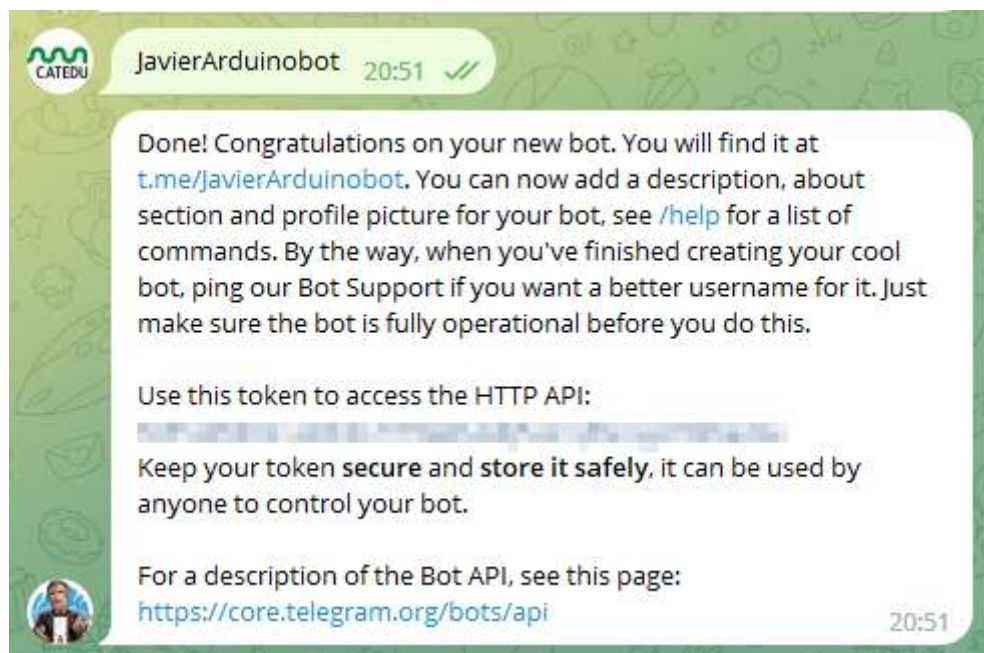


Para crear un nuevo bot, tecleamos **/newbot** y nos preguntará el nombre del bot

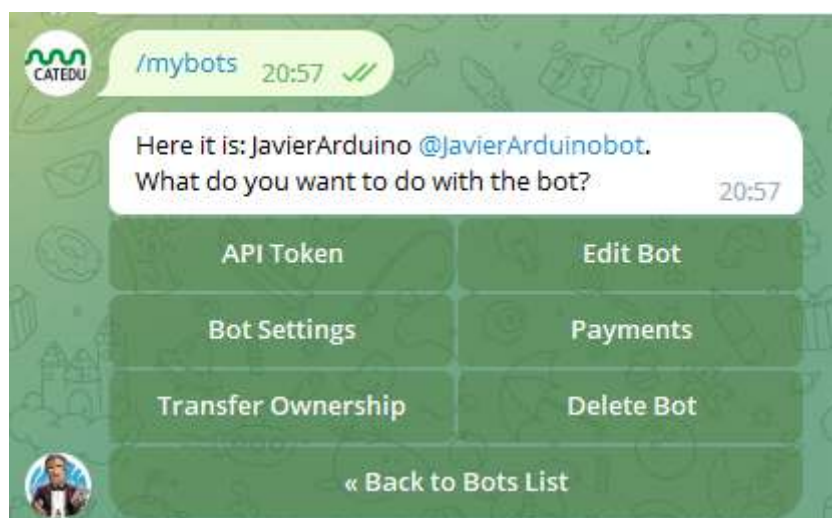
Por cierto, el nombre tiene que acabar con las letras **bot**,

NOS PROPORCIONARÁ EL **TOKEN** DEL ROBOT, QUE TOMAREMOS NOTA





Si tecleamos **/mybots** nos sale los diferentes bots creados y al pulsar en uno de ellos nos salen sus opciones





# Encontrar tu ID en Telegram

## Buscar tu ID : chat privado

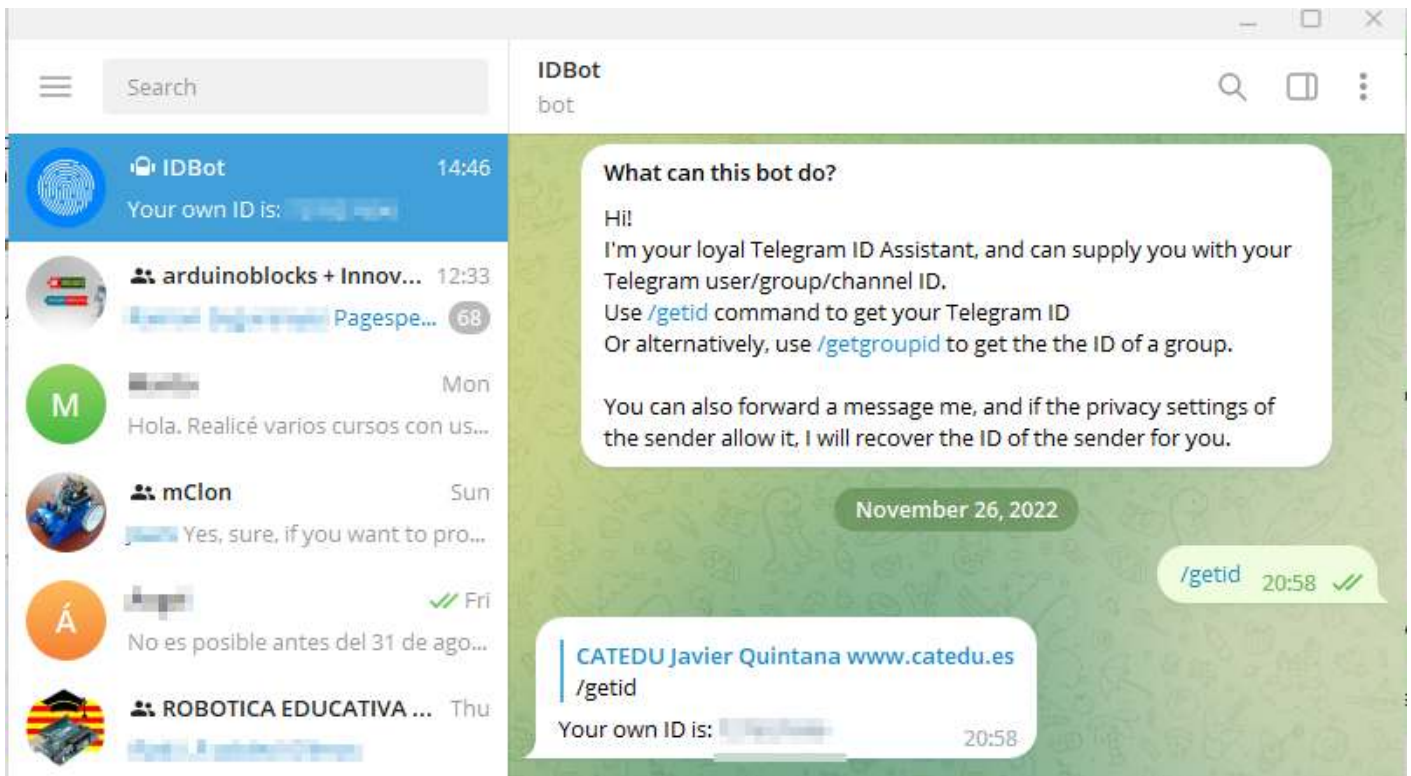
En este caso para que mi Bot me envié mensajes a mi usuario de Telegram directamente busco mi ID.

Vamos a chatear con **@myidbot**



y le preguntamos por nuestro identificador con **/getid**

TOMAMOS NOTA DE NUESTRO IDENTIFICADOR **ID**



## Buscar ID de un grupo

En este caso tendríamos que añadir a **@myidbot** al grupo y ejecutar el comando en el chat del grupo **/getgroupid** saldrá un identificador **negativo**

Una vez conseguido el ID podemos eliminar **@myidbot** del grupo

# Mensajes a Telegram

Una vez que tengamos el *TOKEN* y el *ID* lo ponemos en las líneas

telegramBot="MI\_TOKEN" ## el Token que sale de @BotFather

telegramChatId="MI\_ID" ## El ID del usuario de Telegram destinatario lo da @myidbot

y entonces si ejecutamos la URL

[https://api.telegram.org/botMI\\_TOKEN/sendMessage?chat\\_id=MI\\_ID&text=MENSAJE\\_QUE QUIERA ENVIAR](https://api.telegram.org/botMI_TOKEN/sendMessage?chat_id=MI_ID&text=MENSAJE_QUE QUIERA ENVIAR)

Entonces aparece en mi Telegram desde mi bot el mensaje

Con la librería *urequest* nos permite con la instrucción *urequest.get(url)* nos permite ejecutar la llamada url

El siguiente programa envía por Telegram el botón que estemos pulsando en el Arduino Alvik :

```
from arduino_alvik import ArduinoAlvik
from time import sleep
import random
import sys
import network
import urequests
import time

alvik = ArduinoAlvik()
alvik.begin()

def enviarmensaje(mensaje):

url="https://api.telegram.org/bot"+telegramBot+"/sendMessage?chat_id="+telegramChatId+"&text="+mensaje
    respuesta = urequests.get(url)
```

```
#print (type(respuesta))

WIFI_NETWORK='' ## tu red wifi
WIFI_PASSWORD='' ## la contraseña de la red wifi
telegramBot="" ## el Token que sale de @BotFather
telegramChatId="" ## El ID del usuario de Telegram destinatario lo da @myidbot

wlan = network.WLAN(network.STA_IF)
wlan.active(True)
wlan.connect(WIFI_NETWORK, WIFI_PASSWORD)

if wlan.isconnected():
    print("Alvik se conectó a",WIFI_NETWORK)
else :
    print ("Alvik NO se conectó a",WIFI_NETWORK)

while True:

    if alvik.get_touch_any():
        alvik.left_led.set_color(1, 0, 0)
        alvik.right_led.set_color(1, 0, 0)
    else:
        alvik.left_led.set_color(0, 1, 0)
        alvik.right_led.set_color(0, 1, 0)

    if alvik.get_touch_up() :
        enviarmensaje("arriba")
    if alvik.get_touch_down():
        enviarmensaje("abajo")
    if alvik.get_touch_left() :
```



```
    enviarmensaje("izquierda")  
if alvik.get_touch_right() :  
    enviarmensaje("derecha")  
  
time.sleep(1)
```

## Resultado

<https://www.youtube.com/embed/8jN3Ns8uW7g>

## Para saber más...

- Si en vez de hacerlo con Micropython lo quieres hacer con ArduinoIDE [te recomiendo esta página](#)
- [Un ejemplo](#) de como el Alvik va a la plaza del parking y cuando lo consigue envía un mensaje a Telegram

# Pin pong Telegram

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

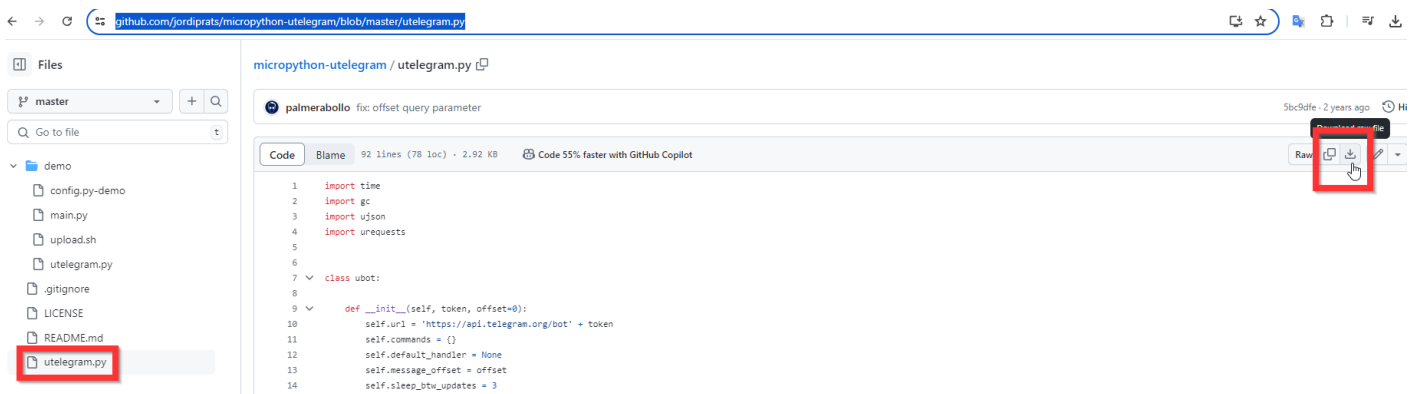
<https://www.youtube.com/watch?v=eZkb9omr-sA>

<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_btwn_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

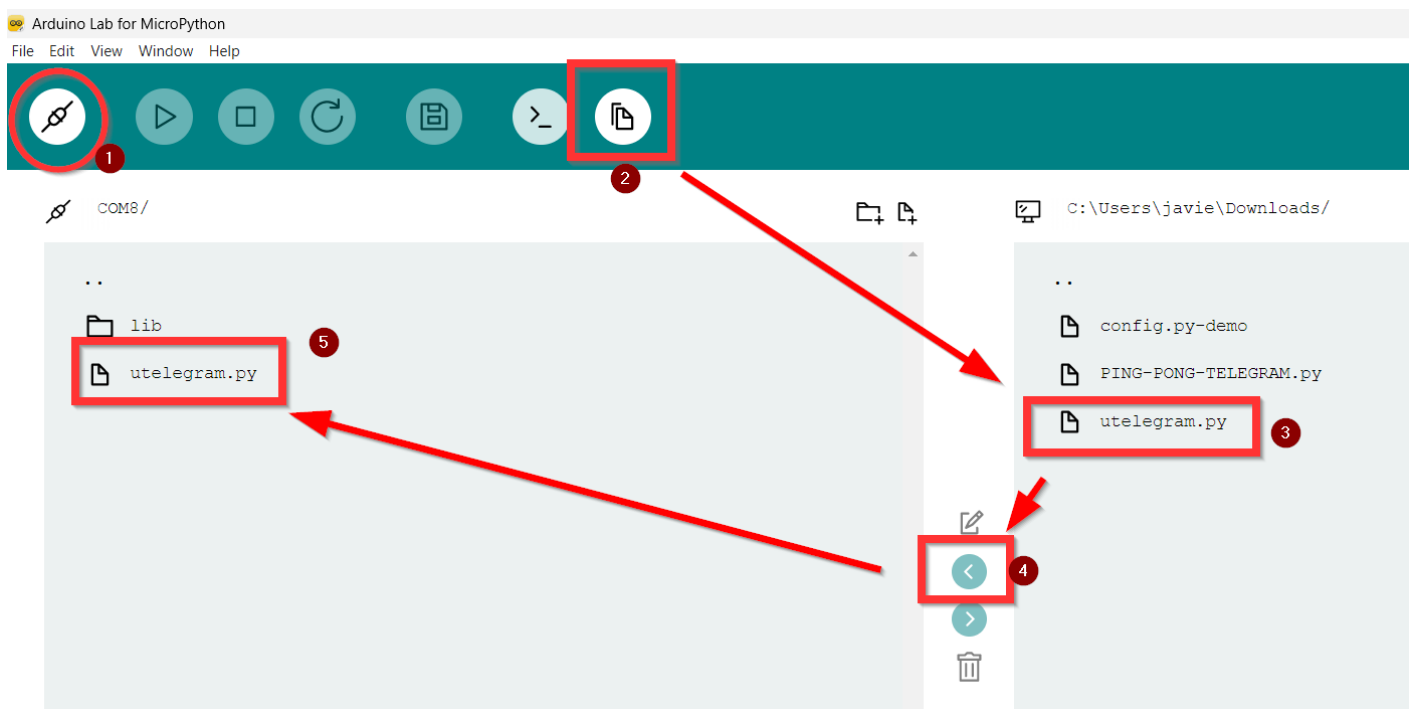
def set_default_handler(self, handler):
    self.default_handler = handler

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

```
self.sleep_bt看_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

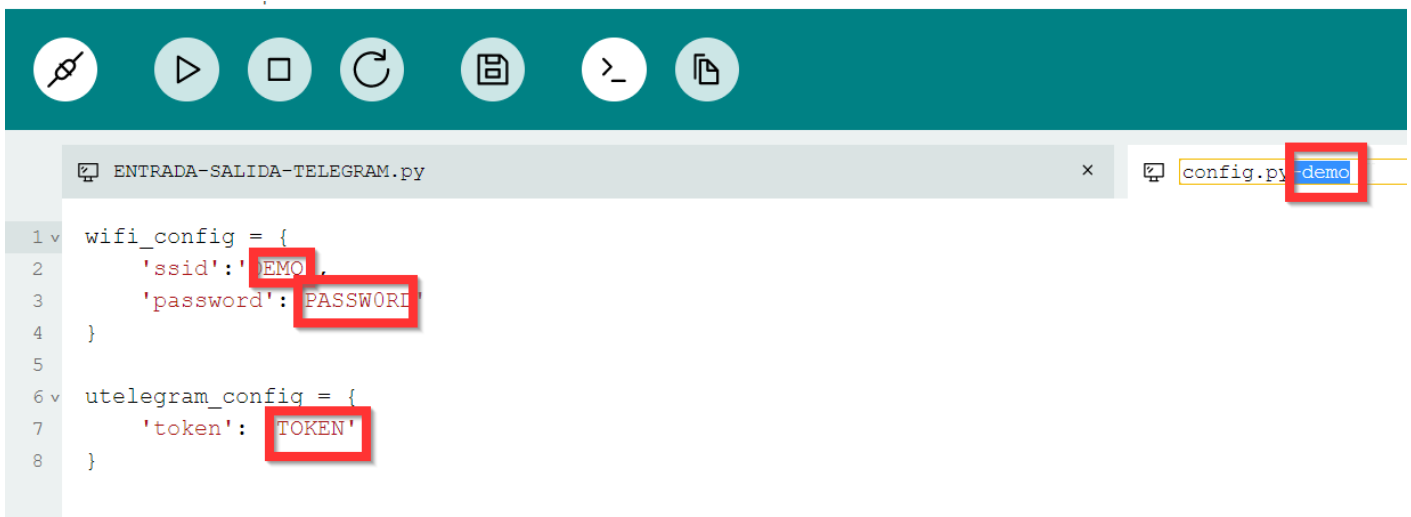


## Paso 2 Archivo config.py

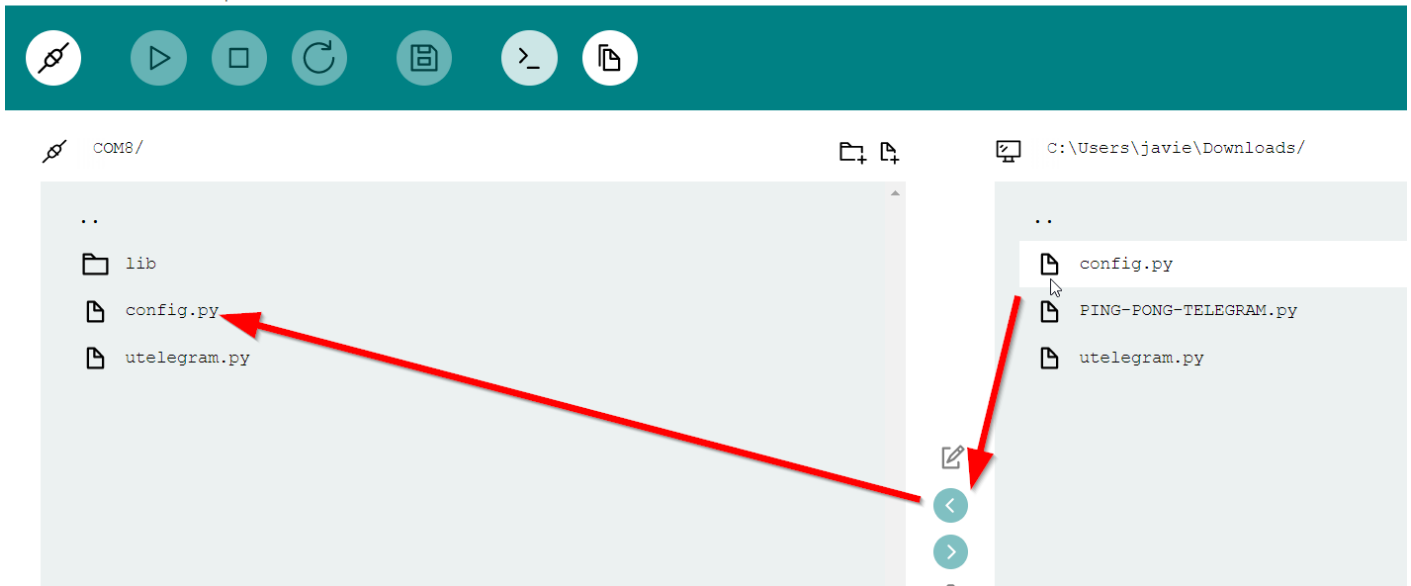
El archivo config.py no es más que el archivo que contiene la wifi y el token, se puede descargar de <https://github.com/jordiprats/micropython-utelegram/blob/master/demo/config.py-demo> o también se puede copiar y pegar de aquí mismo

```
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



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')
```

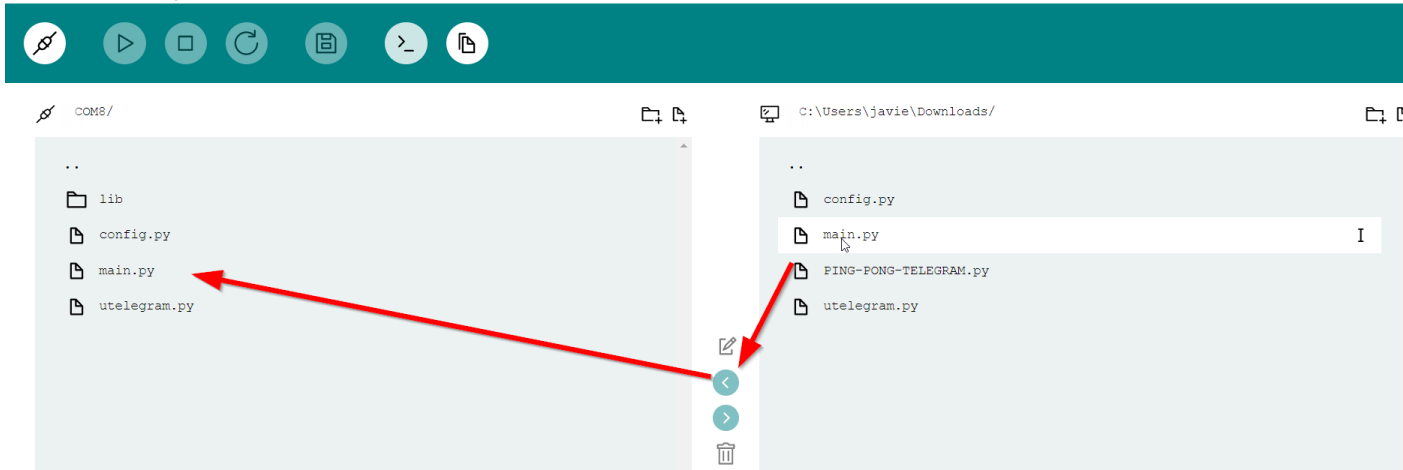
❏ No sé por qué hay que esperar 20 segundos en `utime.sleep(20)` ❏ ¿sospecho que necesita tiempo para estar preparado para "escuchar"

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

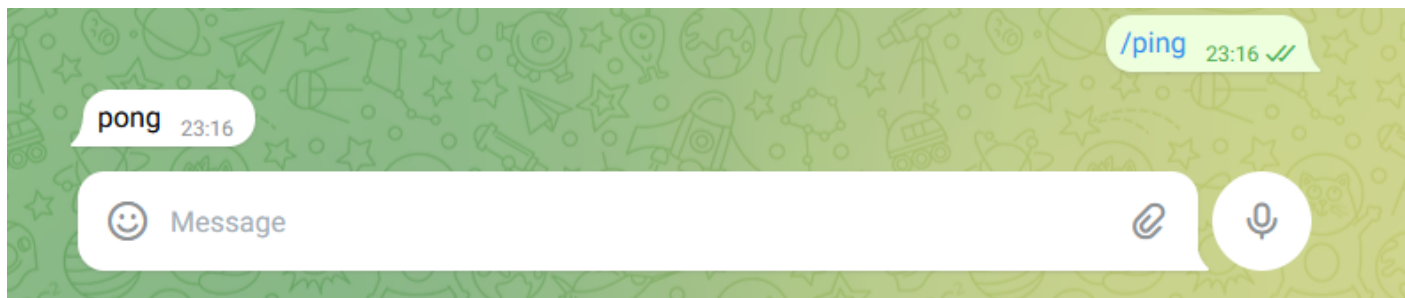
Run

```
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>

# Recepción mensajes Telegram

Podemos ahora enviar un mensaje a ArduinoAlvik y que ejecute un programa por ejemplo el evita obstáculos:

- Envía un mensaje por Telegram que está preparado
- Si le enviamos /go el contesta voy
- Ejecuta la rutina de evita obstáculos

```
from arduino_alvik import ArduinoAlvik  ##### IMPORTAMOS LAS FUNCIONES DE ALVIK

from config import utelegram_config
from config import wifi_config
##### AÑADIMOS LIBRERÍAS PARA LOS OBSTÁCULOS
from time import sleep_ms
import sys

import utelegram
import network
import utime
##### añadimos urequest para enviar mensajes con url
import urequests

alvik = ArduinoAlvik()  ##### CREAMOS UN OBJETO ALVIK
alvik.begin()          ##### LO INICIALIZAMOS
##### VARIABLES PARA EVITAR OBSTÁCULOS
distance = 10
degrees = 45.00
speed = 50.00
#####
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 enviarmensaje(mensaje):                                     ###FUNCION ENVIAR MENSAJE CON
URL ojo cambiar PONAQUITUID

url="https://api.telegram.org/bot"+utelegram_config['token']+"/sendMessage?chat_id=PONAQUITUID
&text="+mensaje
    respuesta = urequests.get(url)
#####
def reply_ping(message):
    print(message)
    bot.send(message['message']['chat']['id'], 'voy')          ### CAMBIAMOS EL MENSAJE
    evitamosobstaculos()                                       ### y EVITAMOS OBSTÁCULOS
#####
def evitamosobstaculos():                                     ### FUNCIÓN EVITAR OBSTÁCULOS
    while (True):

        distance_l, distance_cl, distance_c, distance_r, distance_cr = alvik.get_distance()
        sleep_ms(50)
        print(distance_c)

        if distance_c < distance:
            alvik.rotate(degrees, 'deg')
        elif distance_cl < distance:
            alvik.rotate(degrees, 'deg')
        elif distance_cr < distance:
            alvik.rotate(degrees, 'deg')
        elif distance_l < distance:
            alvik.rotate(degrees, 'deg')
        elif distance_r < distance:

```



```

    alvik.rotate(degrees, 'deg')
else:
    alvik.drive(speed, 0.0, linear_unit='cm/s')

#####

if sta_if.isconnected():
    bot = utelegram.ubot(utelegram_config['token'])
    bot.register('/go', reply_ping)          ### LA CONSIGNA SERÁ GO
    bot.set_default_handler(get_message)

    print('BOT LISTENING')
    enviarmensaje("preparado")
    bot.listen()
else:
    print('NOT CONNECTED - aborting')

```

[https://www.youtube.com/embed/jwe\\_PTmp7g](https://www.youtube.com/embed/jwe_PTmp7g)