Telegram Bot

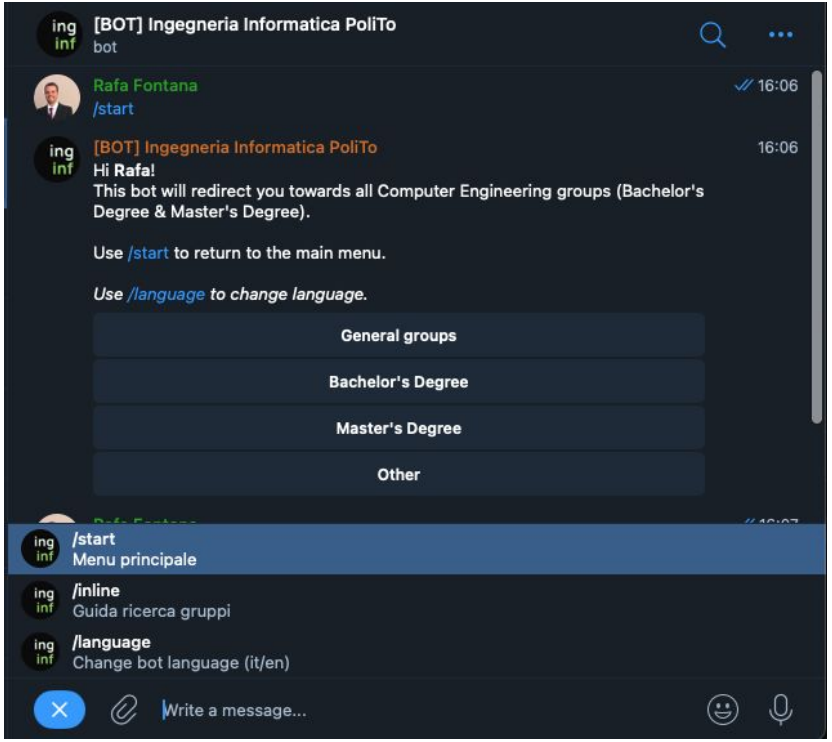
Telegram Bot

Bots are third-party applications that run inside Telegram. Users can interact with bots sending them messages, commands and inline requests. You control the bots sending HTTPS requests to our Bot API.

What can be done with bots?

Telegram Bot and IoT

For this course, Telegram-bots will essentially take the place of an app that is usually developed for a commercial project. We will see the basics steps to create a bot, communicate with it and use it to retrieve information and interact with our systems.



[BOT] Ingegneria Informatica PoliTo
bot

Rafa Fontana 16:06
/start

[BOT] Ingegneria Informatica PoliTo 16:06
Hi Rafa!
This bot will redirect you towards all Computer Engineering groups (Bachelor's Degree & Master's Degree).

Use /start to return to the main menu.

Use /language to change language.

/start
Menu principale

/inline
Guida ricerca gruppi

/language
Change bot language (it/en)

Requirements

Before we start to code, we need to do two things:

- Install the Python library called `telepot`

Terminal

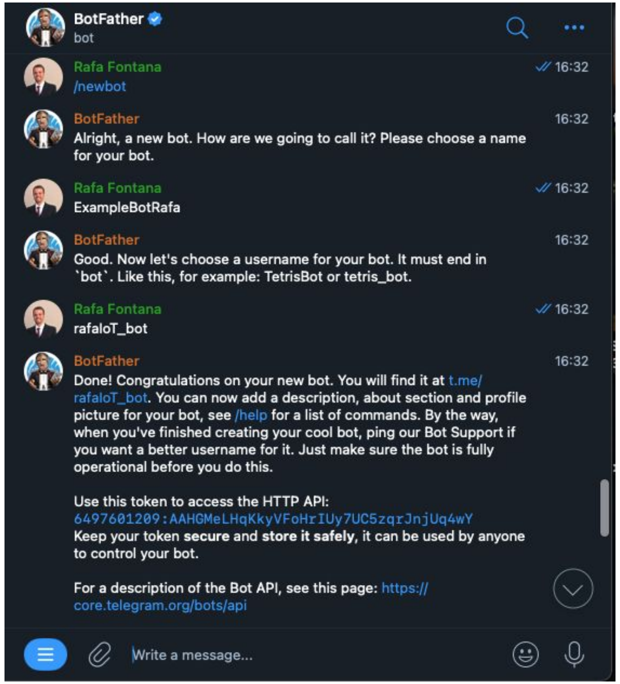
```
pip3 install telepot
```

- Contact Botfather to obtain a token for our bot

Once we have the token we can save it in a json file

⚠️⚠️⚠️ **For the project, we can store the token on the Catalog** ⚠️⚠️⚠️

BotFather



BotFather
bot

Rafa Fontana 16:32
/newbot

BotFather 16:32
Alright, a new bot. How are we going to call it? Please choose a name for your bot.

Rafa Fontana 16:32
ExampleBotRafa

BotFather 16:32
Good. Now let's choose a username for your bot. It must end in `bot`. Like this, for example: TetrisBot or tetris_bot.

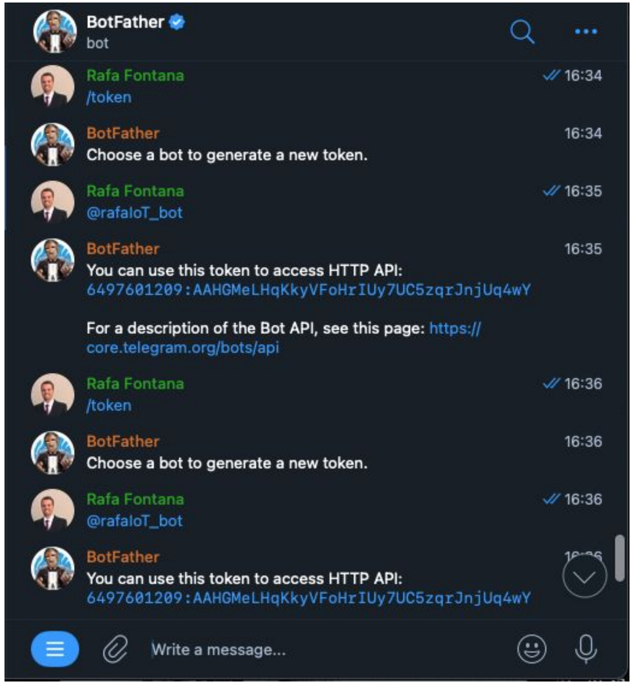
Rafa Fontana 16:32
rafaloT_bot

BotFather 16:32
Done! Congratulations on your new bot. You will find it at t.me/rafaloT_bot. You can now add a description, about section and profile picture for your bot, see /help for a list of commands. By the way, when you've finished creating your cool bot, ping our Bot Support if you want a better username for it. Just make sure the bot is fully operational before you do this.

Use this token to access the HTTP API:
`6497601209:AAHGMeLHqKkyVFoHrIUy7UC5zqrJnjUq4wY`
Keep your token **secure** and **store it safely**, it can be used by anyone to control your bot.

For a description of the Bot API, see this page: https://core.telegram.org/bots/api

Write a message...



BotFather
bot

Rafa Fontana 16:34
/token

BotFather 16:34
Choose a bot to generate a new token.

Rafa Fontana 16:35
@rafaloT_bot

BotFather 16:35
You can use this token to access HTTP API:
`6497601209:AAHGMeLHqKkyVFoHrIUy7UC5zqrJnjUq4wY`

For a description of the Bot API, see this page: https://core.telegram.org/bots/api

Rafa Fontana 16:36
/token

BotFather 16:36
Choose a bot to generate a new token.

Rafa Fontana 16:36
@rafaloT_bot

BotFather
You can use this token to access HTTP API:
`6497601209:AAHGMeLHqKkyVFoHrIUy7UC5zqrJnjUq4wY`

Write a message...

Simple Echo Bot

Bot Initialization

```python
import telepot
from telepot.loop import MessageLoop
import json
import requests

class MyBot:
        def __init__(self,token):
                # Local token
                self.tokenBot=token
                # Catalog token
                #self.tokenBot=requests.get("http://catalogIP/telegram_token").json()["telegramToken"]
                self.bot=telepot.Bot(self.tokenBot)
                MessageLoop(self.bot,{'chat': self.on_chat_message}).run_as_thread()
```

Simple Echo Bot

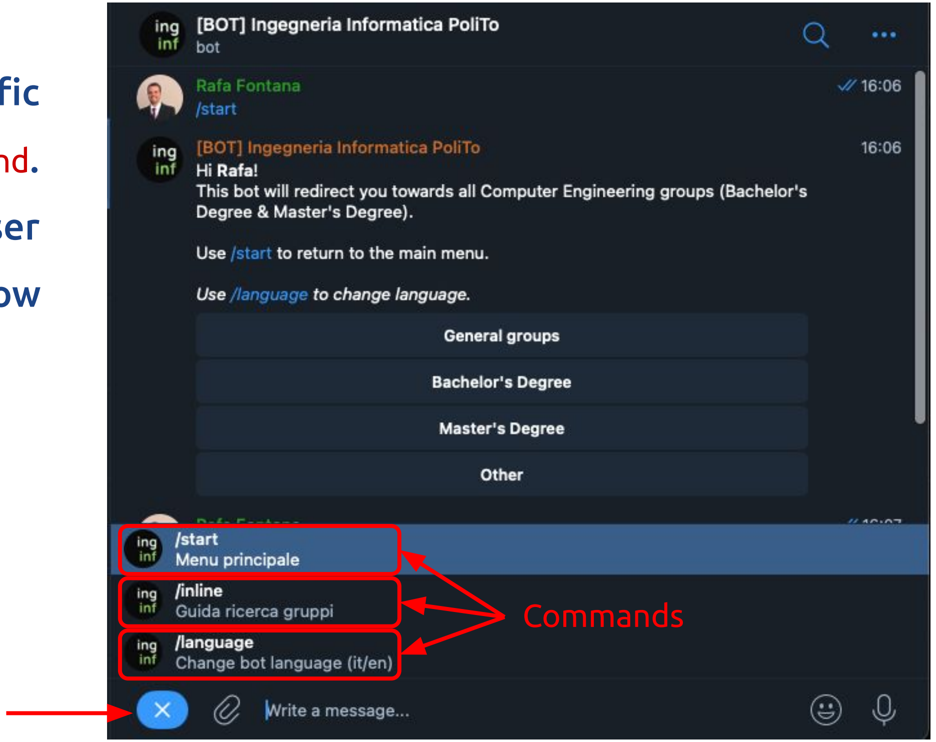
On-chat_message callback

```
def on_chat_message(self,msg):
    content_type, chat_type ,chat_ID = telepot.glance(msg)
    message=msg['text']

    self.bot.sendMessage(chat_ID,text="You sent:\n"+message)
```

Commands

Usually Telegram bot accept some specific commands in the chat given in the format `/command`. With BotFather we can add some hints for the user but then we need to specify inside the code how this command will be handled



[BOT] Ingegneria Informatica PoliTo
bot

Rafa Fontana 16:06
/start

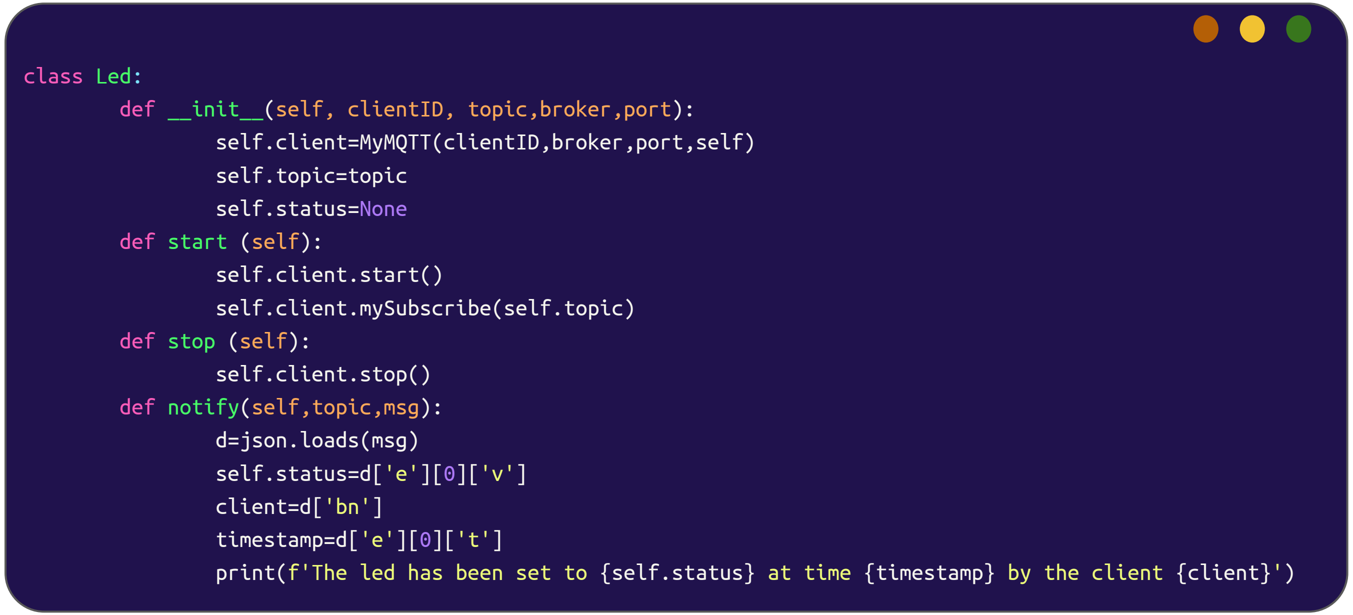
[BOT] Ingegneria Informatica PoliTo 16:06
Hi Rafa!
This bot will redirect you towards all Computer Engineering groups (Bachelor's Degree & Master's Degree).

Use /start to return to the main menu.

Use /language to change language.

Bot + Actuation via MQTT 1/4

Now we will see how to send commands using MQTT to switch on/off the simulated led (`lightActuator.py`)we used in the previous exercises (MQTT Exercises - 03_MQTT).



```
class Led:
        def __init__(self, clientID, topic,broker,port):
                self.client=MyMQTT(clientID,broker,port,self)
                self.topic=topic
                self.status=None
        def start (self):
                self.client.start()
                self.client.mySubscribe(self.topic)
        def stop (self):
                self.client.stop()
        def notify(self,topic,msg):
                d=json.loads(msg)
                self.status=d['e'][0]['v']
                client=d['bn']
                timestamp=d['e'][0]['t']
                print(f'The led has been set to {self.status} at time {timestamp} by the client {client}')
```

Bot + Actuation via MQTT 2/4

We need to do two steps:

1. Adding an instance of the class `MyMQTT` to enable our bot to be a publisher

2. Handle the command sent from the user to switch the led on or off

Bot + Actuation via MQTT 3/4

Enable MQTT

```
class SimpleSwitchBot:
       #....
       self.client=MyMQTT("telegramBot",broker,port,None)
       self.client.start()
       self.topic=topic
       self.__message={
                          'bn':"telegramBot",
                          'e':[
                                {'n':'switch','v':'', 't':'','u':'bool'},
                          ]
                   }
       #....
```

Bot + Actuation via MQTT 4/4

Manage Commands

```
def on_chat_message(self,msg):
        content_type, chat_type ,chat_ID = telepot.glance(msg)
        message=msg['text']
        if message=="/switchon":
                payload=self.__message.copy()
                payload['e'][0]['v']="on"
                payload['e'][0]['t']=time.time()
                self.client.myPublish(self.topic,payload)
                self.bot.sendMessage(chat_ID,text="Led switched on")
        elif message=="/switchoff":
                payload=self.__message.copy()
                payload['e'][0]['v']="off"
                payload['e'][0]['t']=time.time()
                self.client.myPublish(self.topic,payload)
                self.bot.sendMessage(chat_ID,text="Led switched off")
        else:
                self.bot.sendMessage(chat_ID,text="Command not supported")
```

Bot callback and Keyboards 1/2

Keyboard tools

```
from telepot.namedtuple import InlineKeyboardMarkup, InlineKeyboardButton
```

Query callback

```
MessageLoop(self.bot, {'chat': self.on_chat_message,
                       'callback_query': self.on_callback_query}).run_as_thread()
```

Bot callback and Keyboards 2/2

In the callback query we will define the behaviour to follow according to the value of **callback_data**

```
def on_callback_query(self,msg):
      query_ID , chat_ID , query_data = telepot.glance(msg,flavor='callback_query')

      payload = self.__message.copy()
      payload['e'][0]['v'] = query_data
      payload['e'][0]['t'] = time.time()
      self.client.myPublish(self.topic, payload)
      self.bot.sendMessage(chat_ID, text=f"Led switched {query_data}")
```

Exercise 1

Create a telegram bot with a REST interface that is able to receive POST requests with a body like `{"alert":"...","action":"..."}` and sends a message to an user.

Tip

In your project, the body should contain also something that identifies the user. In this way, the bot can ask to the catalog which is the **`chat_id`** corresponding to that user. Just for this exercise, you can obtain the **`chat_id`** from the `/start` message

Exercise 2

Create a telegram bot with an MQTT interface. The bot should be subscribed to a topic like “`MyProjectID/alert/#`” and should receive messages like `{"alert":"...","action":"..."}` and sends a message to an user.

Tip

In your project, the body should contain also something that identifies the user. In this way, the bot can ask to the catalog which is the **`chat_id`** corresponding to that user. Just for this exercise, you can obtain the **`chat_id`** from the `/start` message