

PROTOCOLE MQTT – Client MQTT sur STM32

Les modules terminaux de l'internet des objets sont généralement alimentés par batterie. Dans ce cas l'utilisation d'un nano-ordinateur est impossible, l'autonomie serait trop faible. (Un nano-ordinateur Raspberry Pi alimenté par 3 piles de 1,5v a une autonomie de 3h à 6h)

L'utilisation d'un microcontrôleur à faible consommation est généralement la solution utilisée dans l'IOT embarqué.

Les microcontrôleurs STM32 sont parmi les plus économes en énergie et sont particulièrement adaptés à l'IOT.

Objectifs de formation :

Ce TP propose la réalisation d'une application terminale autonome communiquant par WIFI avec le protocole MQTT vers un broker.

À la fin du TP vous serez capable de créer une application sur STM32 transmettant et recevant des données avec le protocole MQTT.

Matériels :

Une carte NUCLEO L073RZ, mais la plupart des cartes NUCLEO conviendront

Un module WIFI ESP01 (ESP8266 avec module WIFI)

Les logiciels sont développés sur mbed.com, un compte gratuit est nécessaire

Objectifs techniques du TP

Lors de l'appui sur le bouton bleu de la carte NUCLEO, un message est envoyé au broker sur le topic0

Lors de la réception d'un message commençant par 'l' la LED verte (LED1) change d'état

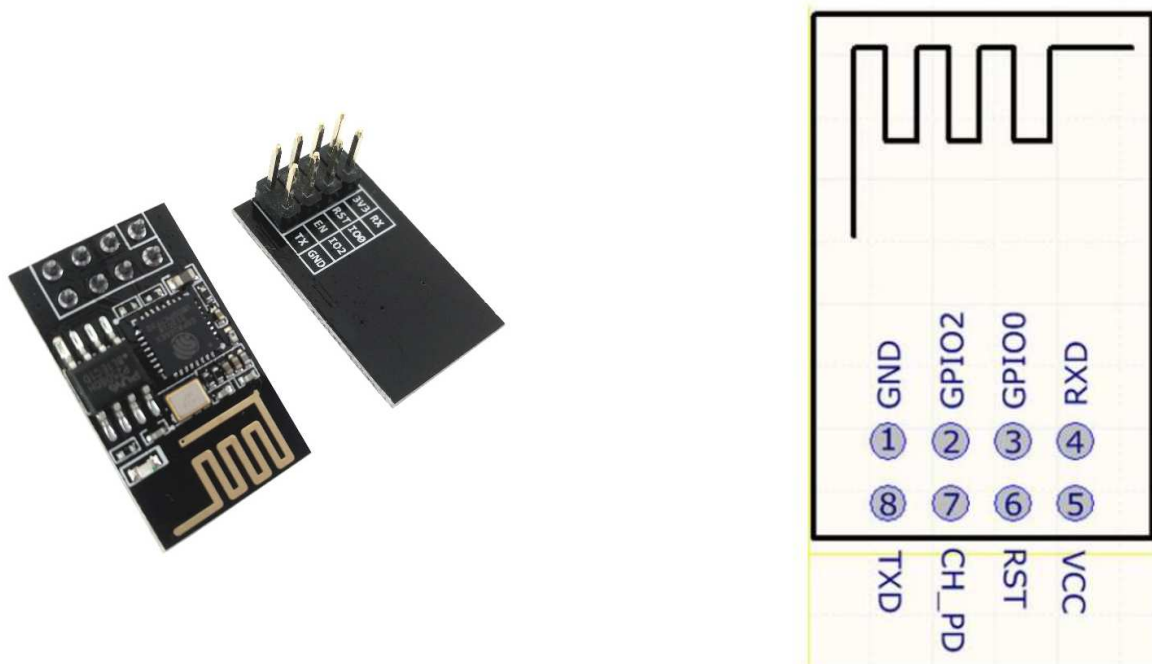
Lors de la réception d'un message commençant par 'q' le programme se termine.

La liaison UART over USB de la carte NUCLEO est utilisée pour visualiser l'état du STM32 et des messages circulant par WIFI

1) Donner à la carte NUCLEO des capacités de communications WIFI avec le module ESP01

Le module ESP01 comporte un connecteur 8 broches, un émetteur/récepteur WIFI et un microcontrôleur ESP8266 disposant d'un programme de communication par commandes AT.

La documentation du module est ici http://wiki.ai-thinker.com/media/esp8266/esp8266_series_modules_user_manual_v1.1.pdf



Vue de dessus

PROTOCOLE MQTT – Client MQTT sur STM32

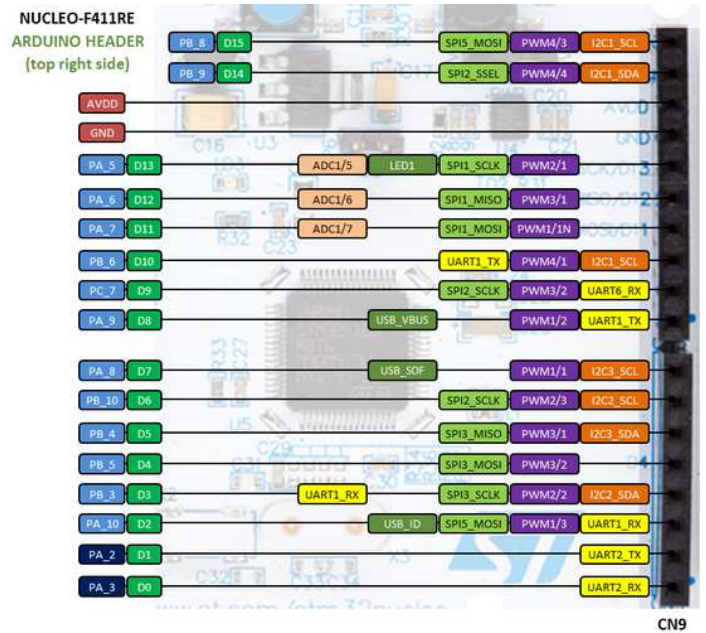
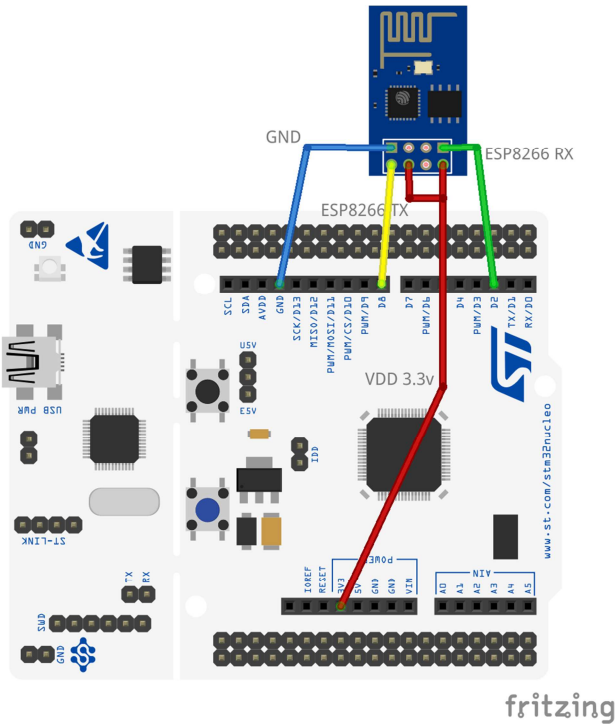
2) Câblage à réaliser

L'ESP8266 sera alimenté par la tension de VDD 3.3v disponible sur les connecteurs Arduino de la carte NUCLEO

La broche CH_PD (chip select) doit être reliée à VDD.

La sortie TX est reliée à l'entrée RX de la carte Nucleo (broche D2, UART1 RX)

L'entrée RX est reliée à la sortie TX de la carte Nucleo (broche D8, UART1 TX)



3) Mise en œuvre de l'ESP8266

L'ESP8266 est piloté par un protocole AT.

Pour tester le bon fonctionnement du montage, réaliser dans MBED le programme "SerialPassthrough" qui établit une liaison l'UART over USB vers le port série de l'ESP8266.

(Tout caractère émis ou reçu sur le terminal ASCII est retransmis vers l'ESP8266).

```
#include "mbed.h"

#define bauds 115200

RawSerial pc(USBTX, USBRX);
RawSerial dev(D8,D2); //TX,RX PA_11,PA_12 ou PA_9,PA_10

DigitalOut led1(LED1);
DigitalOut led4(LED4);

void dev_recv()
{
    led1 = !led1;
    while(dev.readable()) {
        pc.putc(dev.getc());
    }
}

void pc_recv()
{
    led4 = !led4;
    while(pc.readable()) {
        dev.putc(pc.getc());
    }
}

int main()
{
    pc.baud(bauds);
    dev.baud(bauds);
    pc.printf("PASSTHROUGH ACTIF VITESSE -> %d BAUDS\n",bauds);

    pc.attach(&pc_recv, Serial::RxIrq);
    dev.attach(&dev_recv, Serial::RxIrq);
}
```

La commande AT+GMR renvoie la configuration de l'ESP8266 :



The screenshot shows a terminal window titled 'COM7 - Tera Term VT'. The menu bar includes 'Fichier', 'Edition', 'Configuration', 'Contrôle', 'Fenêtre(W)', and 'Aide'. The terminal output displays the following text:

```
PASSTHROUGH ACTIF VITESSE -> 115200 BAUDS
AT+GMR
AT version:1.3.0.0(Jul 14 2016 18:54:01)
SDK version:2.0.0(656edbf)
compile time:Jul 19 2016 18:44:44
OK
```

La documentation des commandes AT est ici : https://www.espressif.com/sites/default/files/documentation/4a-esp8266_at_instruction_set_en.pdf.

4) TP MQTT sur STM32

Le driver ESP8266 pour STM32 contrôle ce dernier par des commandes AT.

La bibliothèque MQTT pour STM32 dispose de la classe MQTT et des méthodes :

MQTT::Client qui crée un client MQTT ;

MQTT::Message qui permet l'émission et la réception des messages MQTT.

Le logiciel de démonstration est disponible ici :

https://os.mbed.com/users/cdupaty/code/ESP8266_HelloMQTT_Mosquitto_STM32/

Le STM 32 envoie et reçoit des messages suivant le protocole MQTT vers le broker test.mosquitto.org
Ce broker est public et ne nécessite pas d'identification ni mot de passe.

Configuration :

Éditer le fichier mbed_app.json et compléter avec les ID et pass du WIFI.

```
{
  "config": {
    "network-interface": {
      "help": "options are ETHERNET, WIFI_ESP8266, WIFI_ODIN, WIFI_RTW, MESH_LOWPAN_ND, MESH_THREAD, CELLULAR_ONBOARD",
      "value": "WIFI_ESP8266"
    },
    "mesh_radio_type": {
      "help": "options are ATMEL, MCR20",
      "value": "ATMEL"
    },
    "esp8266-tx": {
      "help": "Pin used as TX (connects to ESP8266 RX)",
      "value": "D8"
    },
    "esp8266-rx": {
      "help": "Pin used as RX (connects to ESP8266 TX)",
      "value": "D2"
    },
    "esp8266-ssid": {
      "value": "\"ssIDwifi\""
    },
    "esp8266-password": {
      "value": "\"motdepassewifi\""
    },
    "esp8266-debug": {
      "value": false
    }
  },
  "target_overrides": {
    "**": {
      "target.features_add": ["NANOSTACK", "LOWPAN_ROUTER", "COMMON_PAL"],
      "mbed-mesh-api.6lowpan-nd-channel-page": 0,
      "mbed-mesh-api.6lowpan-nd-channel": 12,
      "mbed-trace.enable": 0
    },
    "HEXIWEAR": {
      "esp8266-tx": "PTD3",
      "esp8266-rx": "PTD2"
    },
    "NUCLEO_L073RZ": {
      "esp8266-tx": "D8",
      "esp8266-rx": "D2"
    },
    "NUCLEO_F411RE": {
      "esp8266-tx": "D8",
      "esp8266-rx": "D2"
    }
  }
}
```

PROTOCOLE MQTT – Client MQTT sur STM32

Configurer le fichier main.c pour une connexion sur test.mosquitto.org.

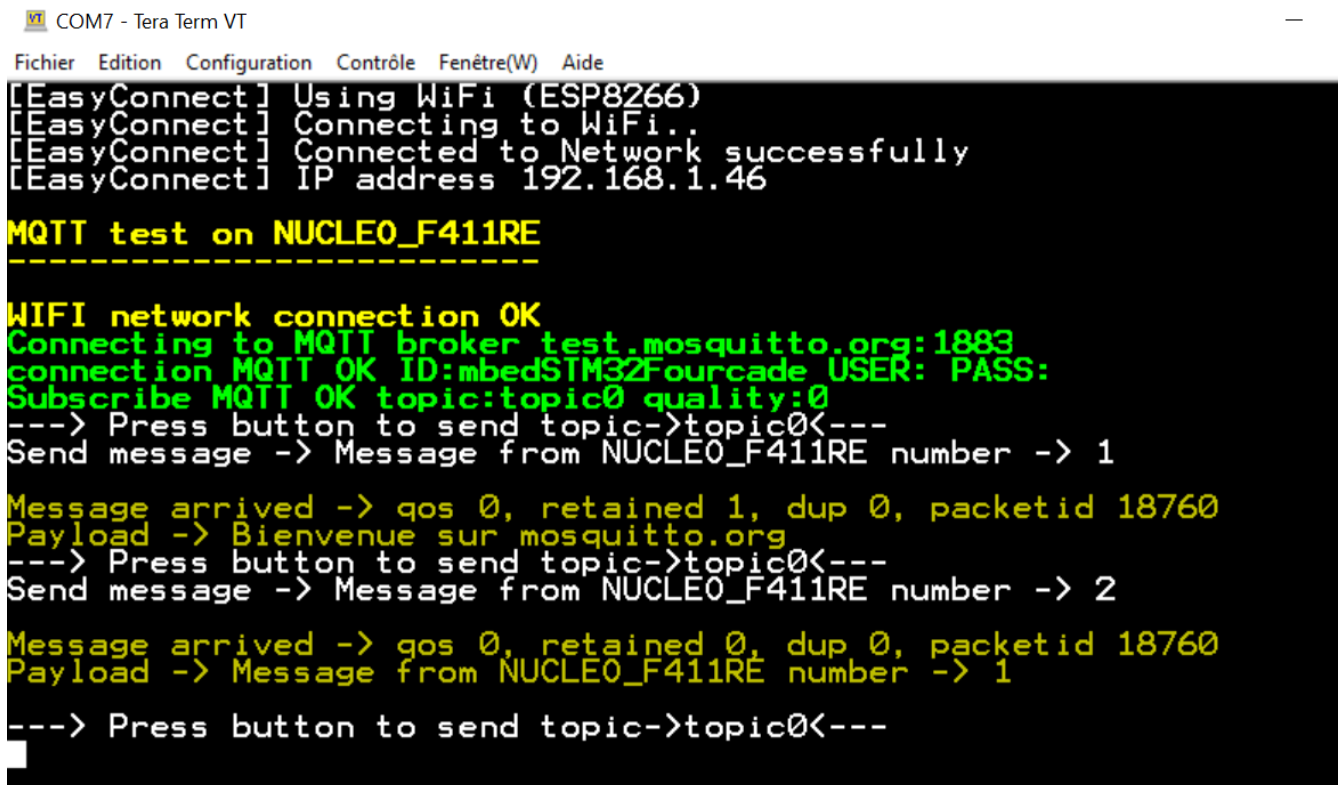
```
DigitalOut led(LED1);    // LED verte (user)
Serial pc(USBTX, USBRX);
//DigitalIn btn(USER_BUTTON); //PC13 sur F411
DigitalIn btn(PA_8,PullUp); // button connected with internal pullup

#define board "NUCLEO_F411RE"
/*
MQTT QoS,  There are 3 QoS levels in MQTT
At most once (MQTT::QOS0)
At least once (MQTT::QOS1)
Exactly once (MQTT::QOS2).
*/
#define quality MQTT::QOS0
/*
retained flag : The broker stores the last retained message and the corresponding QoS for that topic.
Each client that subscribes to a topic pattern that matches the topic of the retained
message receives the retained message immediately after they subscribe.
The broker stores only one retained message per topic.
*/
#define ret false
/*
dup flag : The flag indicates that the message is a duplicate
and was resent because the intended recipient (client or broker) did not acknowledge the original
message.
*/
#define dupli false

const char* hostname = "test.mosquitto.org";
const int port = 1883;
char* ID ="mbedSTM32Fourcade";
char* user =NULL;      // user & pass = NULL for broker with no credential like mosquitto.
char* pass =NULL;
```

Compiler, télécharger le .bin dans la carte Nucleo.

Ouvrir un terminal (ex teraterm) sur le port série over USB (ici COM7) à 9600,n,8,1 :



```
COM7 - Tera Term VT

Fichier Edition Configuration Contrôle Fenêtre(W) Aide

[EasyConnect] Using WiFi (ESP8266)
[EasyConnect] Connecting to WiFi..
[EasyConnect] Connected to Network successfully
[EasyConnect] IP address 192.168.1.46

MQTT test on NUCLEO_F411RE
-----

WIFI network connection OK
Connecting to MQTT broker test.mosquitto.org:1883
connection MQTT OK ID:mbedSTM32Fourcade USER: PASS:
Subscribe MQTT OK topic:topic0 quality:0
---> Press button to send topic->topic0<---
Send message -> Message from NUCLEO_F411RE number -> 1

Message arrived -> qos 0, retained 1, dup 0, packetid 18760
Payload -> Bienvenue sur mosquitto.org
---> Press button to send topic->topic0<---
Send message -> Message from NUCLEO_F411RE number -> 2

Message arrived -> qos 0, retained 0, dup 0, packetid 18760
Payload -> Message from NUCLEO_F411RE number -> 1

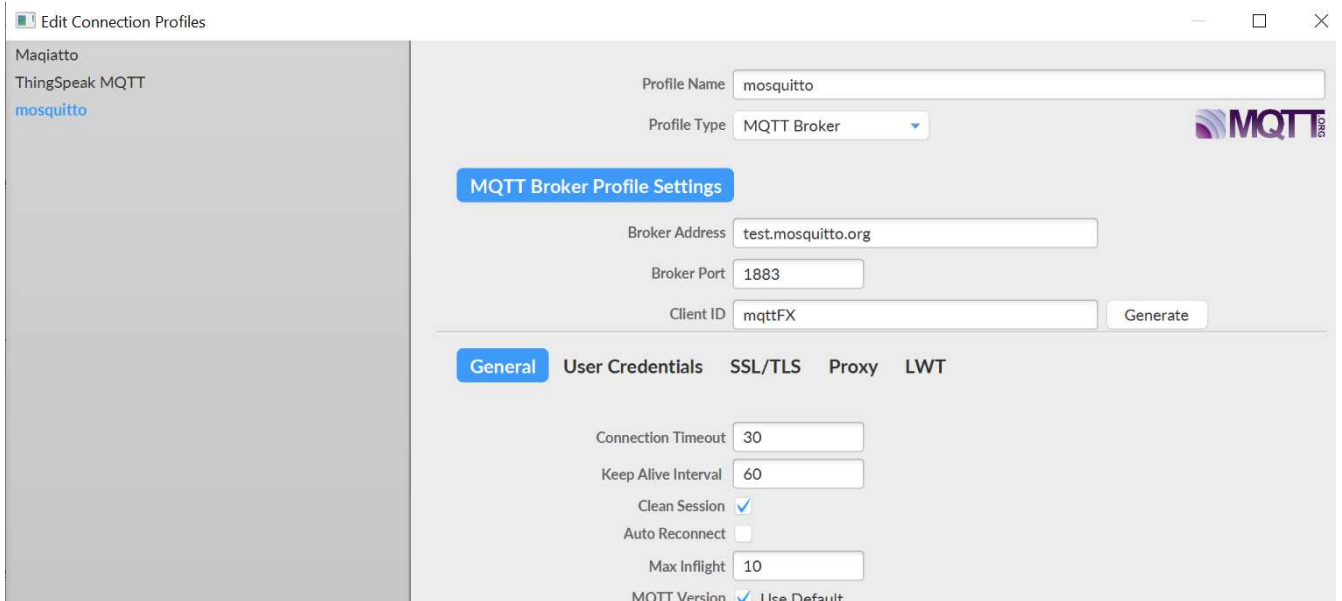
---> Press button to send topic->topic0<---
```

PROTOCOLE MQTT – Client MQTT sur STM32

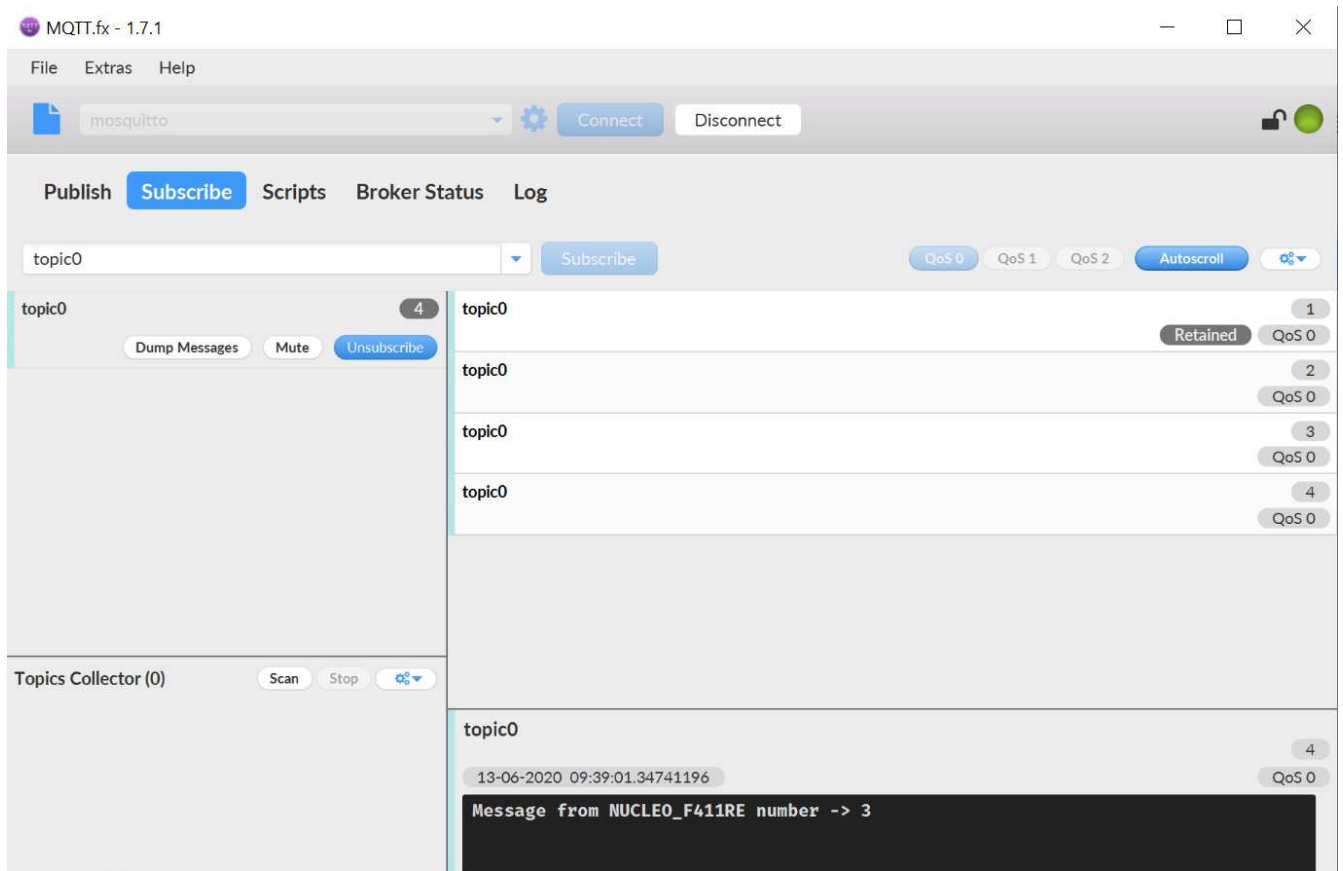
Sur un PC utiliser MQTT.fx, sur un smartphone, utiliser MyMQTT pour tester l'ensemble.

Les messages émis par le STM32 sont retransmis sur ces applications.

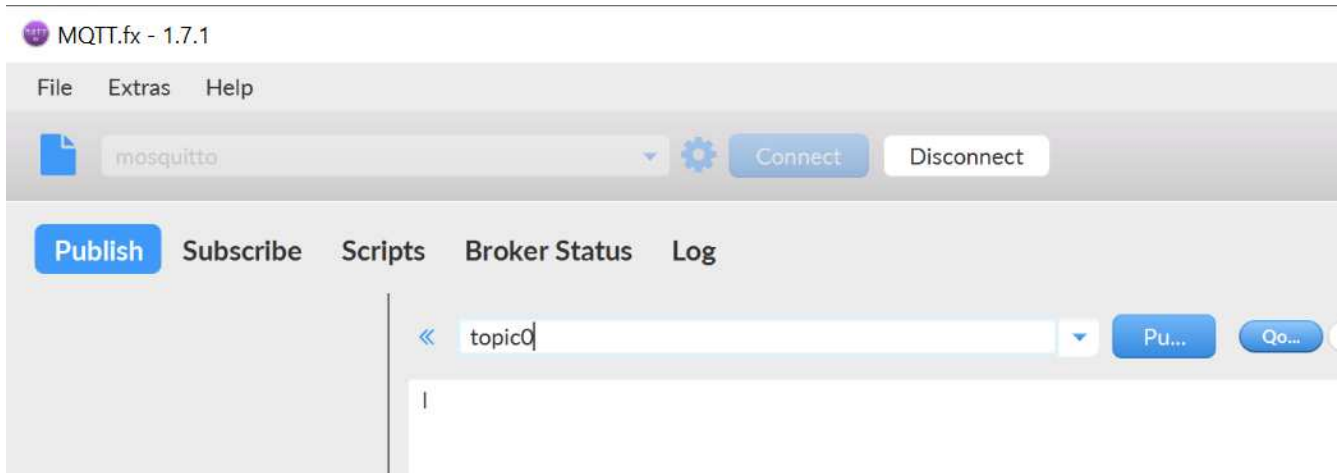
La publication sur le topic0 d'un 'l' depuis une ces applications entraîne le basculement de la LED verte de la carte Nucleo :



Après souscription au topic0, réception des messages en provenance du STM32 :



Publication du message 'l' sur le topic0 :



Réception du message et basculement de la LED :

```
Message arrived -> qos 0, retained 0, dup 0,
60
Payload -> l
toggle LED
---> Press button to send : topic0<---
```

5) Exercice :

Interfacer un capteur de température TMP102 (ex breakout <https://www.sparkfun.com/products/13314>).

Un driver MBED est disponible ici : <https://os.mbed.com/components/TMP102-Temperature-Sensor/>.

Adapter le programme de manière à publier la température sur topic0/temp lors de la réception du caractère 't' ou lors de l'appui sur le bouton bleu.