

Installer le serveur MQTT mosquitto

[Installation](#)[Test de Mosquitto](#)[Déclaration dans Domoticz et sur un périphérique](#)[Contrôle des messages](#)

Installation

Comme indiqué sur projetsdiy.fr, MQTT (Message Queuing Telemetry Transport) est un protocole de messagerie qui fonctionne sur le principe de souscription / publication qui a été développé à la base pour simplifier la communication entre les machines et qui est maintenant un standard international pour la communication entre machines (M2M) et les objets (IoT). Pour économiser au maximum la batterie des appareils mobiles. MQTT consomme 11 fois moins d'énergie pour envoyer des messages et 170 fois moins pour en recevoir que le protocole HTTP. MQTT est également 93 fois plus rapide que le protocole HTTP. Pour cela il nécessite un serveur/broker. C'est ce que l'on va faire ici en installant Mosquitto sur un Raspberry.

La commande pour installer Mosquitto sous Linux est

```
sudo apt install mosquitto
```

[Lien vers le fichier : cliquez ici](#)

L'installation d'un client peut très utile pour faire des tests par la suite. Ces tests se feront avec la commande `mosquitto_sub` ou `mosquitto_pub`

```
sudo apt-get install mosquitto-clients
```

[Lien vers le fichier : cliquez ici](#)

Vérifier qu'il répond bien avec la commande

```
systemctl status mosquitto
```

[Lien vers le fichier : cliquez ici](#)

Le service peut être redémarré avec la commande

```
sudo systemctl restart mosquitto
```

[Lien vers le fichier : cliquez ici](#)

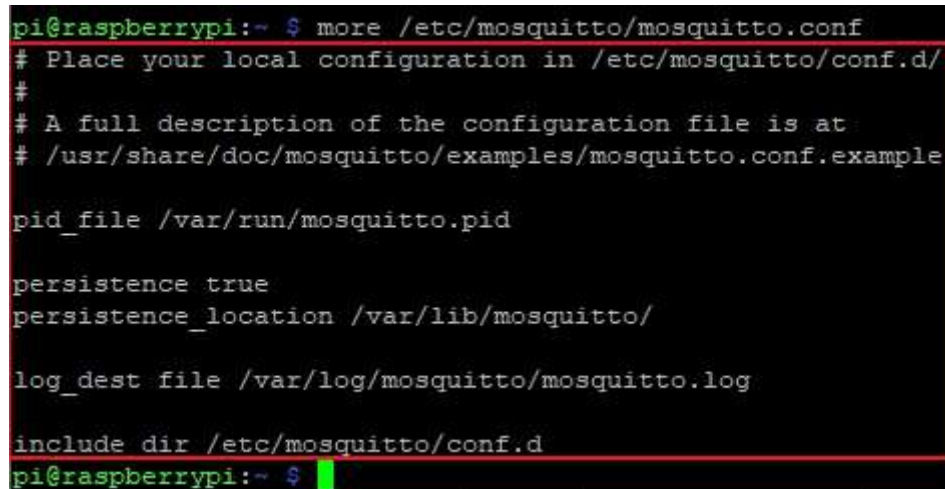
On peut visualiser le fichier de configuration avec

```
more /etc/mosquitto/mosquitto.conf
```

[Lien vers le fichier : cliquez ici](#)

Vous remarquerez qu'on demande de placer ses fichiers de configuration dans

/etc/mosquitto/conf.d/



```
pi@raspberrypi:~ $ more /etc/mosquitto/mosquitto.conf
# Place your local configuration in /etc/mosquitto/conf.d/
#
# A full description of the configuration file is at
# /usr/share/doc/mosquitto/examples/mosquitto.conf.example

pid_file /var/run/mosquitto.pid

persistence true
persistence_location /var/lib/mosquitto/

log_dest file /var/log/mosquitto/mosquitto.log

include_dir /etc/mosquitto/conf.d
pi@raspberrypi:~ $
```

On peut donc en créer un avec nano. On le nommera par exemple default.conf avec la commande

```
sudo nano /etc/mosquitto/conf.d/default.conf
```

[Lien vers le fichier : cliquez ici](#)

En option on peut faire en sorte qu'il faille un login/mot de passe pour pouvoir se connecter à Mosquitto.

Ce dernier n'est pas utilisé dans ce tuto.

Si vous voulez en créer un, taper

```
sudo mosquitto_passwd -c /etc/mosquitto/passwd NOM_UTILISATEUR
```

[Lien vers le fichier : cliquez ici](#)

Et déclarez le dans le fichier de conf perso, cad ici dans /etc/mosquitto/conf.d/default.conf avec ces lignes

```
allow_anonymous false  
password_file /etc/mosquitto/passwd
```

[Lien vers le fichier : cliquez ici](#)

Vous pouvez voir les logs avec la commande

```
tail -f /var/log/mosquitto/mosquitto.log
```

[Lien vers le fichier : cliquez ici](#)

Si vous avez activé le firewall sur votre Linux, il faut penser à autoriser le port 1883 avec une commande du type

```
#Autoriser le port 1883 en entrée pour Mosquitto  
iptables -t filter -A INPUT -p tcp --dport 1883 -j LOGACCEPT  
echo "Mosquitto ok"
```

[Lien vers le fichier : cliquez ici](#)

Test de Mosquitto

Cette étape est facultative. Je vous conseille, une fois vos éléments en place, de contrôler avec MQTT.FX qui est abordé plus loin.

Comme expliqué sur <https://youtu.be/FsSlgAWtGio?t=443>

On peut tester que l'envoi des messages fonctionne bien en lançant 2 terminaux ssh.

Sur le premier, créer un subscriber avec une commande comme ci-dessous.

Si vous n'avez pas créé d'utilisateur sécuriser l'accès à Mosquitto, il n'est pas nécessaire de renseigner l'option -u et -P

```
mosquitto_sub -h localhost -u votreuser -P lepassword -t topic
```

[Lien vers le fichier : cliquez ici](#)

Dans une autre session ssh, créer un publisher et un message test avec une commande du type

```
mosquitto_pub -h localhost -u votreuser -P lepassword -t topic
```

[Lien vers le fichier : cliquez ici](#)

La première session avec le subscriber doit recevoir le message de la seconde



```
pi@raspberrypi: ~  
pi@raspberrypi:~$ mosquitto pub -h localhost -t topic/test -m "Voici un test"  
pi@raspberrypi:~$  
pi@raspberrypi:~$ mosquitto_sub -h localhost -t topic/test  
Voici un test
```

Déclaration dans Domoticz et sur un périphérique

Dans Domoticz, la déclaration de la passerelle MQTT se fait comme ci-dessous.

Dans adresse distante, c'est localhost qui est ici indiqué, car mosquitto est installé sur la même machine que Domoticz.

1883 est le port par défaut de Mosquitto.

Vous remarquerez que rien n'a été mis dans identifiant et mot de passe. Cela sera une amélioration de la configuration à faire plus tard. On a vu par exemple comment créer ce login/mot de passe au début.

Activé: ☒

Nom: MQTT

Type: MQTT Client Gateway with LAN interface ▼

Délai d'inactivité: Désactivé ▼
Le périphérique sera redémarré si aucune donnée n'est reçue pendant ce délai.
N'activez pas cette option pour les périphériques ne recevant pas de donnée

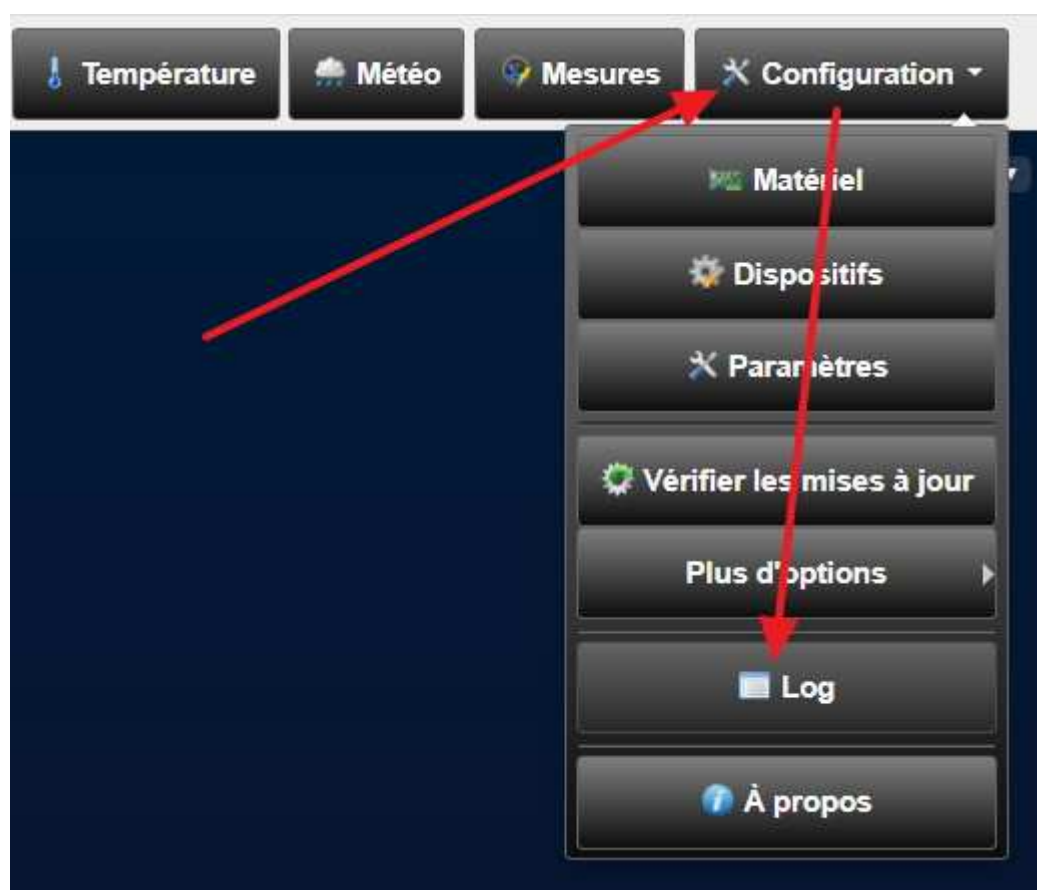
Adresse distante: localhost

Port: 1883

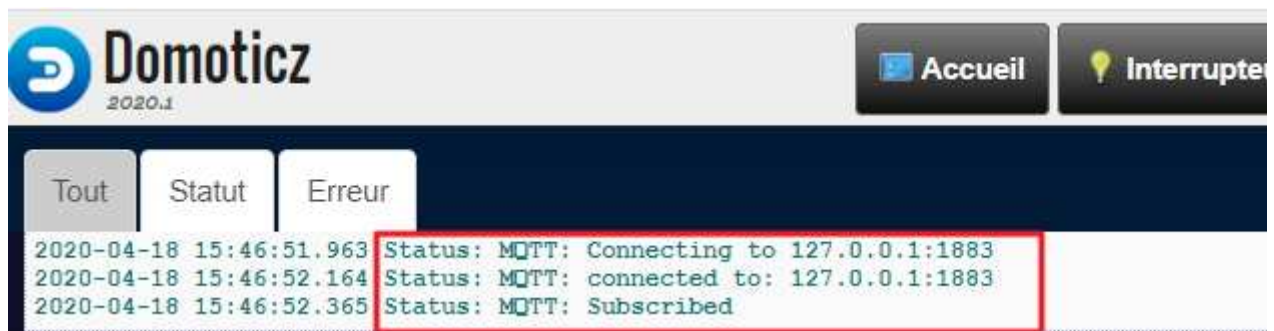
Identifiant:

Mot de passe:

Ensuite aller dans les logs

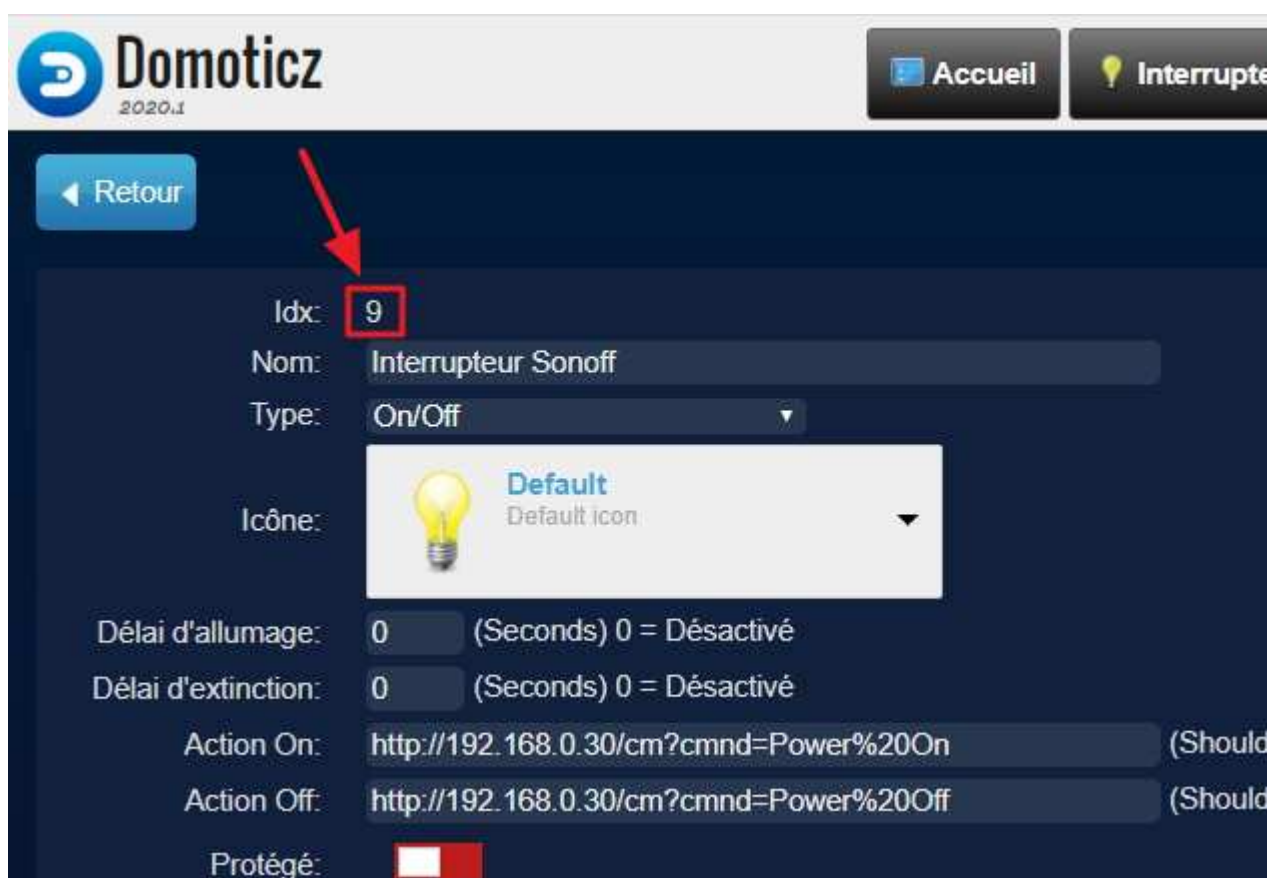


Vous devez voir que Domoticz se connecte sur Mosquitto et qu'il se déclare en tant que Subscriber



Maintenant que l'on a un serveur MQTT avec Domoticz en écoute dessus, on peut par exemple simuler l'envoi d'un ordre à Domoticz.

Pour cela je vais voir dans mes Interrupteurs, j'en prends un au hasard et je note son idx. Ici c'est 9.



On va lui envoyer un ordre avec la commande `mosquitto_pub`.

Là où avant on avait mis "Voici un test" dans l'option -m, ici on va mettre un ordre qui prend la forme d'une requête json. Dans cette requête on précise l'idx, ici 9, et l'ordre, ici nvalue à 1

Dans la commande ci-dessous, vous remarquerez que le canal est ici `domoticz/in` et non plus `topic/test`

```
mosquitto_pub -h localhost -u votreuser -P lepassword -t domot
```

[Lien vers le fichier : cliquez ici](#)

Et là BAM ! Cela s'allume bien



Et les ordres sont bien visibles dans le log de Domoticz

```
2020-04-18 16:04:02.297 MQTT: Topic: domoticz/in, Message: { "idx" : 9, "nvalue" : 1}
2020-04-18 16:04:12.319 MQTT: Topic: domoticz/in, Message: { "idx" : 9, "nvalue" : 0}
2020-04-18 16:04:16.315 MQTT: Topic: domoticz/in, Message: { "idx" : 9, "nvalue" : 1}
2020-04-18 16:17:59.330 (Dummy) Light/Switch (Interrupteur Sonoff)
```

On peut aussi prendre un périphérique où il y a ESP Easy et y déclarer le serveur MQTT en allant sur Controllers puis Add

Si vous vous demandez comment on peut faire un périphérique où il y a ESP Easy, vous pouvez regarder

<https://www.tutos.eu/3098>

ESP Easy Mega: ESP_Easy

△Main ⚙️Config **Controllers** 📌Hardware 🖨️Devices

	Nr	Enabled	
Add	1		
Add	2		
Add	3		

Powered by **Let's Control It** community

ESP Easy Mega: ESP_Easy

△Main ⚙️Config **Controllers** 📌Hardware 🖨️Devices

Controller Settings

Protocol:

Domoticz MQTT

- Standalone -

Domoticz HTTP

Domoticz MQTT

ThingSpeak

Home Assistant (openHAB) MQTT

PiDome MQTT

Emoncms

?

Powered by **Let's Control It** commu

△Main ⚙️Config **Controllers** 📌Hardware 🖨️Devices ✉️

Controller Settings

Protocol:

Domoticz MQTT

?

Close **Submit**

Indiquer l'adresse ip de mosquitto, activer et cliquer sur Submit

ESP Easy Mega: ESP_Easy

△ Main ⊗ Config 💬 Controllers 📌 Hardware 🔧 Devices 📧 Notific

Controller Settings

Protocol: Domoticz MQTT ?

Locate Controller: Use IP address

Controller IP: 192.168.0.23

Controller Port: 1883

Controller Queue

Minimum Send Interval: 100 [ms]

MQTT

Controller Subscribe: domoticz/out

Controller Publish: domoticz/in

Controller LWT Topic:

LWT Connect Message:

LWT Disconnect Message:

Send LWT to broker: ☒

Will Retain: ☒

Clean Session: ☐

Enabled: ☒

Close Submit

Le serveur MQTT déclaré, il faut indiquer quel type de device l'ESP représente.

Pour l'exemple on va dire qu'on est un interrupteur. L'ESP va donc indiquer quand on actionne un périphérique.

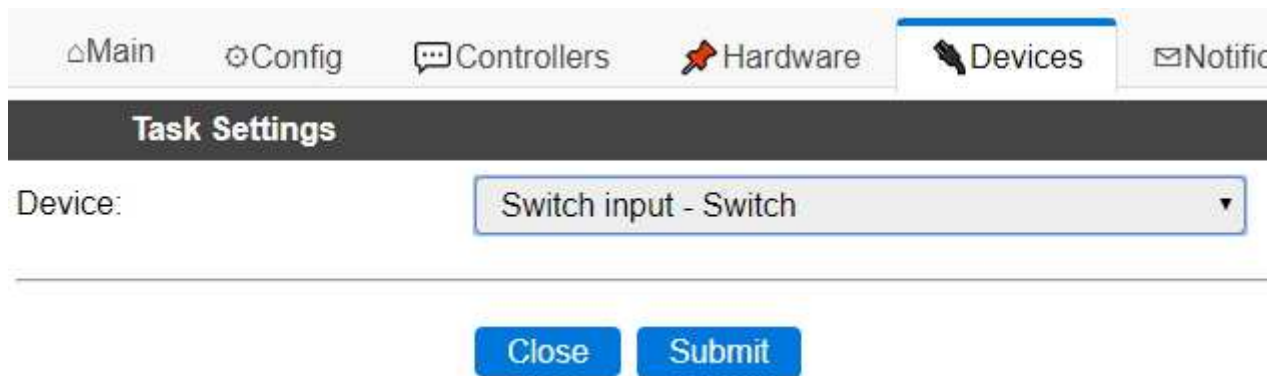
Charge après à Mosquitto de relayer l'information à Domoticz qui devra se

charger ensuite d'envoyer un ordre à une lampe à allumer par exemple.

Pour cela, toujours sous ESP Easy, cliquer sur Devices puis Add



Si c'est un interrupteur, prendre switch



Nommer le device, indiquer la broche avec lequel il est reliée à votre interrupteur et renseignez le type de switch.

Task Settings

Device: Switch input - Switch ? i

Name: Interrupteur

Enabled: ☒

Sensor

Internal PullUp: ☐

Inversed Logic: ☐

Note: Will go into effect on next input change.

GPIO: GPIO-0 (D3) ▲ ▼

Switch Type: Switch ▼

Switch Button Type: Normal Switch ▼

Send Boot state: ☐

Ensuite il faut faire le lien dans Domoticz qui joue le rôle de tour de contrôle.
A ce titre Domoticz référence tous les objets avec un idx.
Il faut trouver celui qui correspond à l'ESP avec lequel on est en train de jouer.

Donc sous Domoticz, il vous faut déclarer un interrupteur, si ce n'est pas déjà fait.

Si vous vous demandez comment, jeter un oeil à l'article

<https://www.tutos.eu/6215>

Noter son idx et le reporter sous ESP Easy.

Ici j'ai pris l'interrupteur 'Dummy' qui pilote la lampe de mon Gecko.

Cocher aussi send to controller

Advanced event managementDe-bounce (ms): Doubleclick event: Doubleclick max. interval (ms): Longpress event: Longpress min. interval (ms): Use Safe Button (slower): ☐**Data Acquisition**Send to Controller
1 ☒IDX: Interval: [sec] (Optional for this Device)

Cliquer sur Submit, suite à quoi si vous activez l'interrupteur, Domoticz recevra l'ordre via MQTT que vous devez visualiser dans ses logs.

Send to Controller
1 ☒IDX: Interval: [sec] (Optional for this Device)**Values**

#	
1	State

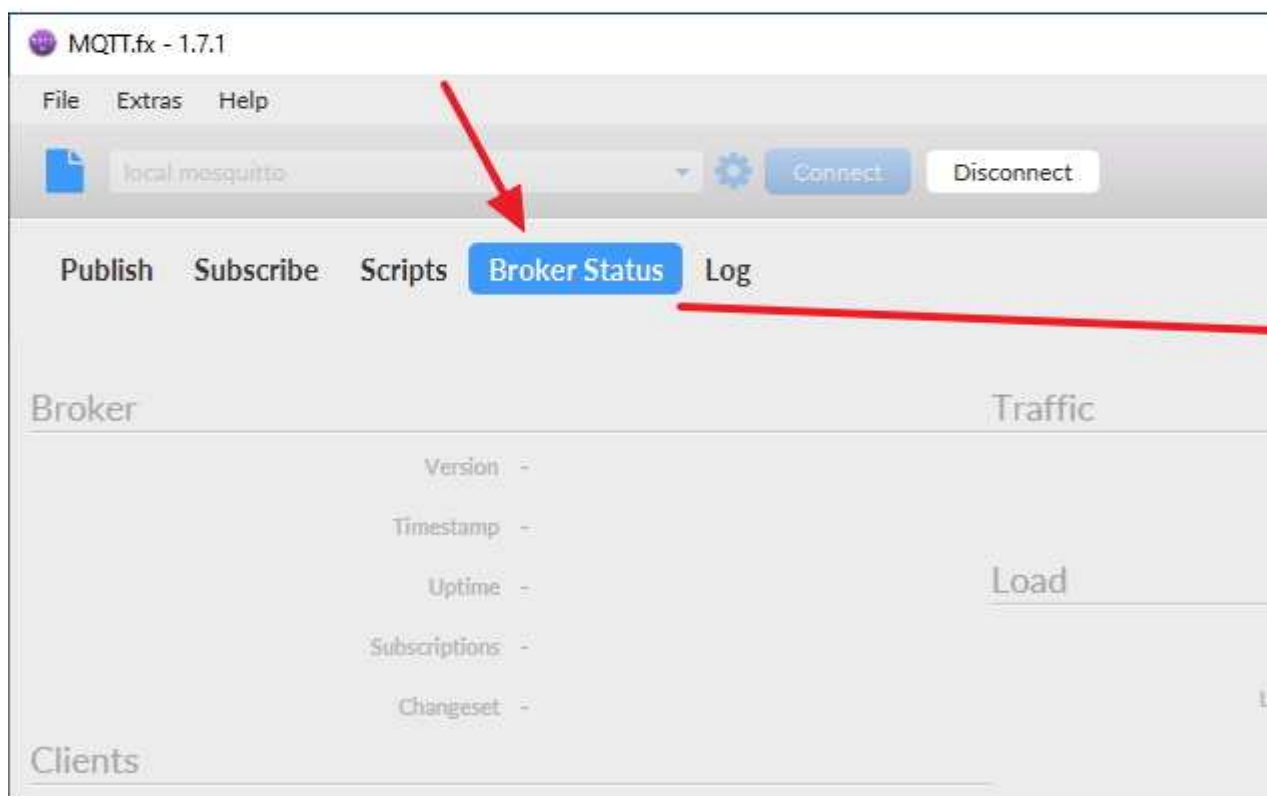
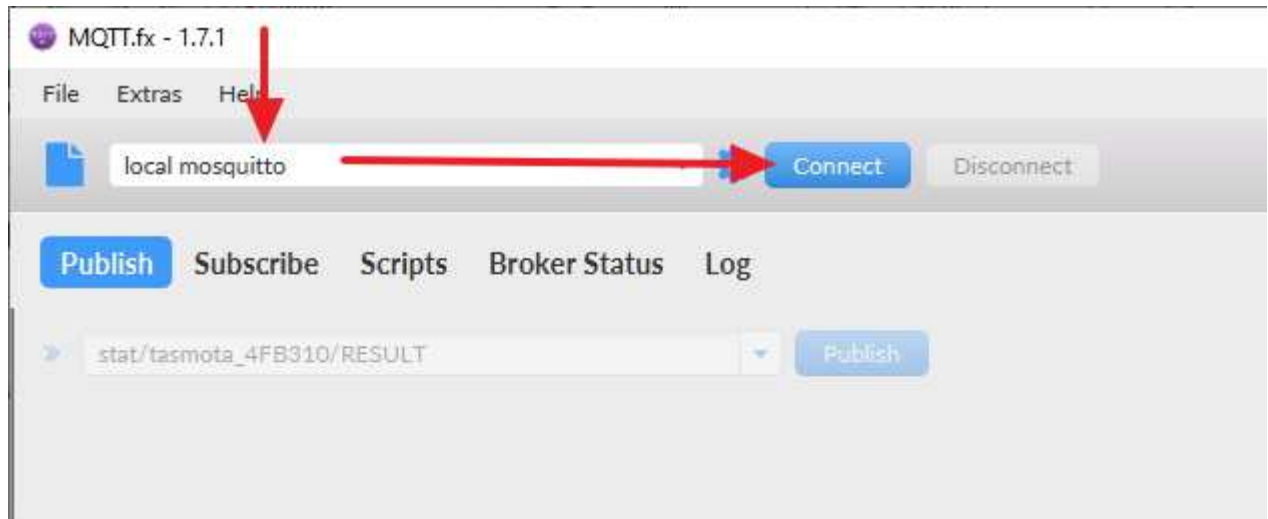
Contrôle des messages

Un outil très utile c'est MQTT.FX que l'on peut télécharger sur

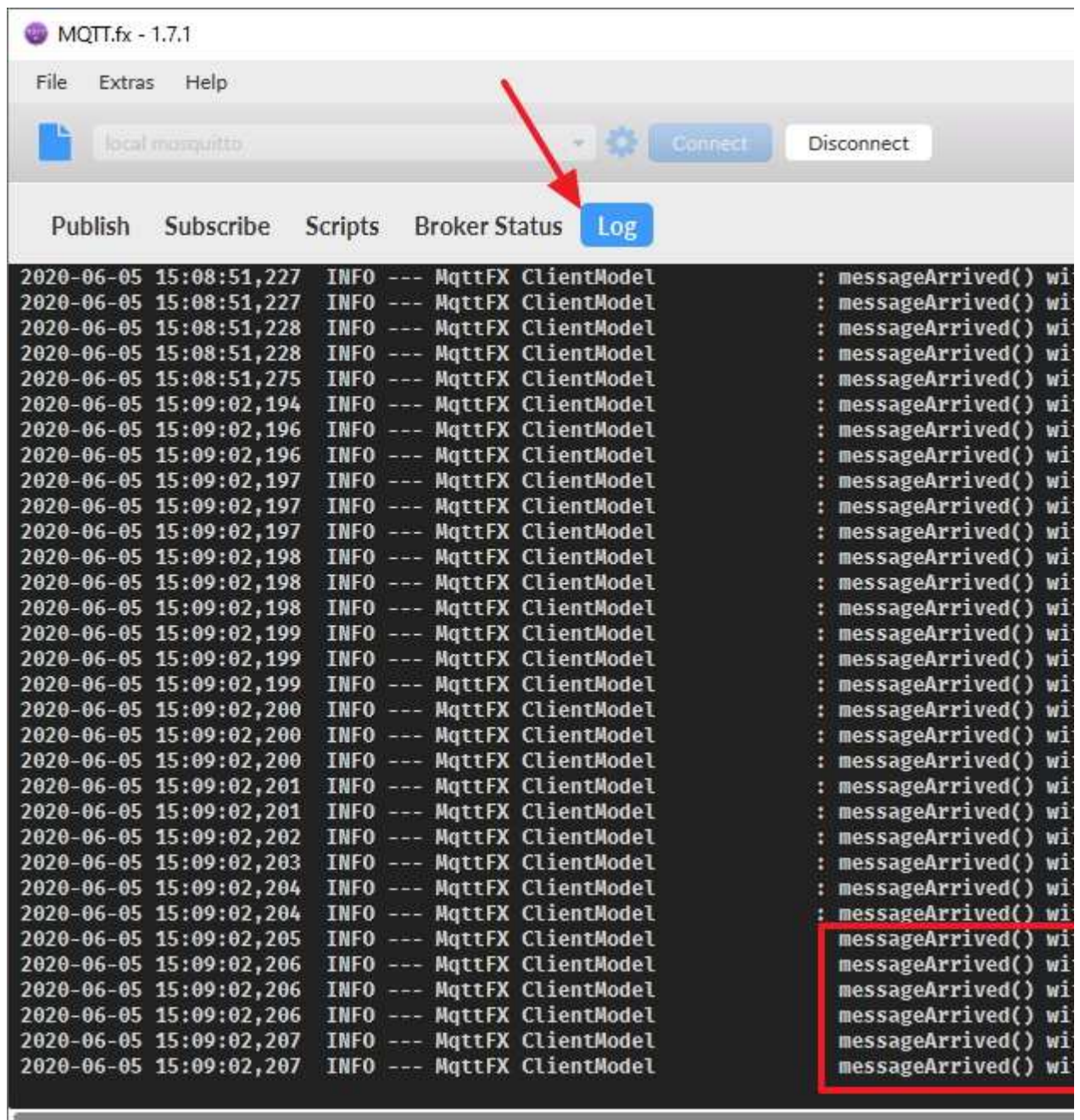
<https://mqttfx.jensd.de/index.php/download>

Je l'ai trouvé en lisant <https://projetsdiy.fr/mosquitto-broker-mqtt-raspberry-pi/>

Il permet de se connecter sur votre broker MQTT comme montré ci-dessous



Du coup ensuite on peut voir les logs du serveur et voir en temps réel si un message part ou arrive, et ça c'est top pour contrôler si tout va bien



On peut trouver les différents channels, s'y abonner, voir les messages passer et leur contenu

MQTT.fx - 1.7.1

File Extras Help

local mosquitto [Connect] [Disconnect]

Publish **Subscribe** Scripts Broker Status Log

tele/tasmota_4FB310/SENSOR [Subscribe]

On peut s'abonner

domoticz/in 698 [Dump Messages] [Mute] [Unsubscribe]

domoticz/out/LWT 1 [Dump Messages] [Mute] [Unsubscribe]

stat/tasmota_4FB310/POWER 22 [Dump Messages] [Mute] [Unsubscribe]

stat/tasmota_4FB310/RESULT 18 [Dump Messages] [Mute] [Unsubscribe]

tele/tasmota_4FB310/SENSOR 34 [Dump Messages] [Mute] [Unsubscribe]

Topics Collector (7) [Scan] [Stop] [Filter]

domoticz/in

domoticz/out/LWT

stat/tasmota_4FB310/POWER

stat/tasmota_4FB310/RESULT

tele/tasmota_4FB310/LWT

tele/tasmota_4FB310/SENSOR

tele/tasmota_4FB310/STATE

On peut scanner pour voir les topics

19-04-2020 20:48:40.749206

```
{
  "Time": "2020-04-19T19:48:40.749206Z",
  "day": 0.000,
  "Today": 1.000,
  "Voltage": 232,
  "Current": 0.000
}
```

Avant j'avais tenté d'accéder aux logs via `/var/log/mosquitto/mosquitto.log` mais c'est compliqué et peu pratique.
De base il n'y a rien d'intéressant dans les logs.

Cet article <http://www.steves-internet-guide.com/mosquitto-logging> nous apprend que si on veut avoir des logs, il faut lancer Mosquitto avec l'option

verbose, cad -v

Pour ajouter ce -v, il faut déjà savoir comment se lance Mosquitto de base. On trouve le fichier de configuration du service de Mosquitto dans le fichier /lib/systemd/system/mosquitto.service

Ce dernier nous apprend que la commande qui lance le service est /usr/sbin/mosquitto -c /etc/mosquitto/mosquitto.conf

Donc pour avoir des logs, il faut commencer par arrêter son service avec la commande

```
sudo systemctl stop mosquitto
```

[Lien vers le fichier : cliquez ici](#)

Ajouter le -v dans le paramètre ExecStart du service de mosquito ne fonctionne pas. Je ne sais pas pourquoi.

Il faut relancer le programme en direct, avec le -v, avec une commande comme ci-dessous.

```
cd /usr/sbin  
sudo ./mosquitto -v -c /etc/mosquitto/mosquitto.conf
```

[Lien vers le fichier : cliquez ici](#)

De là depuis une autre session ssh, si on regarde les logs de Mosquitto avec cette commande, là on verra tout ce qui se passe

```
sudo tail -f /var/log/mosquitto/mosquitto.log
```

[Lien vers le fichier : cliquez ici](#)

Exemple

```
1587231024: Received PINGREQ from ESPClient_5C:CF:7F:CB:A1:E8  
1587231024: Sending PINGRESP to ESPClient_5C:CF:7F:CB:A1:E8  
1587231034: Received PINGREQ from ESPClient_5C:CF:7F:CB:A1:E8  
1587231034: Sending PINGRESP to ESPClient_5C:CF:7F:CB:A1:E8  
1587231036: Received PINGREQ from mqttjs_18c6255b  
1587231036: Sending PINGRESP to mqttjs_18c6255b  
1587231044: Received PINGREQ from ESPClient_5C:CF:7F:CB:A1:E8  
1587231044: Sending PINGRESP to ESPClient_5C:CF:7F:CB:A1:E8  
1587231045: Received PINGREQ from Domoticz8f92af83-2c82-4dc7-a412-9258a9c3474d  
1587231045: Sending PINGRESP to Domoticz8f92af83-2c82-4dc7-a412-9258a9c3474d  
1587231055: Received PINGREQ from ESPClient_5C:CF:7F:CB:A1:E8
```

Du coup, quand on reçoit un ordre de Google Assistant qui passe par Homebridge, je vois passer des ordres qui commencent par Received PUBLISH from mqttjs_e0de85d7

```
1587244519: Received PUBLISH from mqttjs_e0de85d7 (d0, q0, r0, m0, 'domoticz/i
1587244519: Sending PUBLISH to Domoticz8f92af83-2c82-4dc7-a412-9258a9c3474d (d
1587244526: Received PINGREQ from ESPClient_5C:CF:7F:CB:A1:E8
1587244526: Sending PINGRESP to ESPClient_5C:CF:7F:CB:A1:E8
1587244530: Received PUBLISH from mqttjs_e0de85d7 (d0, q0, r0, m0, 'domoticz/i
1587244530: Sending PUBLISH to Domoticz8f92af83-2c82-4dc7-a412-9258a9c3474d (d
1587244536: Received PINGREQ from ESPClient_5C:CF:7F:CB:A1:E8
```

Article(s) suivant(s)

[Piloter Domoticz avec Google Assistant](#)

Article(s) en relation(s)

[Installer domoticz sur un raspberry Pi](#)

[Flasher un module sonoff avec Tasmota et le contrôler sous Domoticz](#)

