



Les objets connectés

(Partie 3 : MQTT)



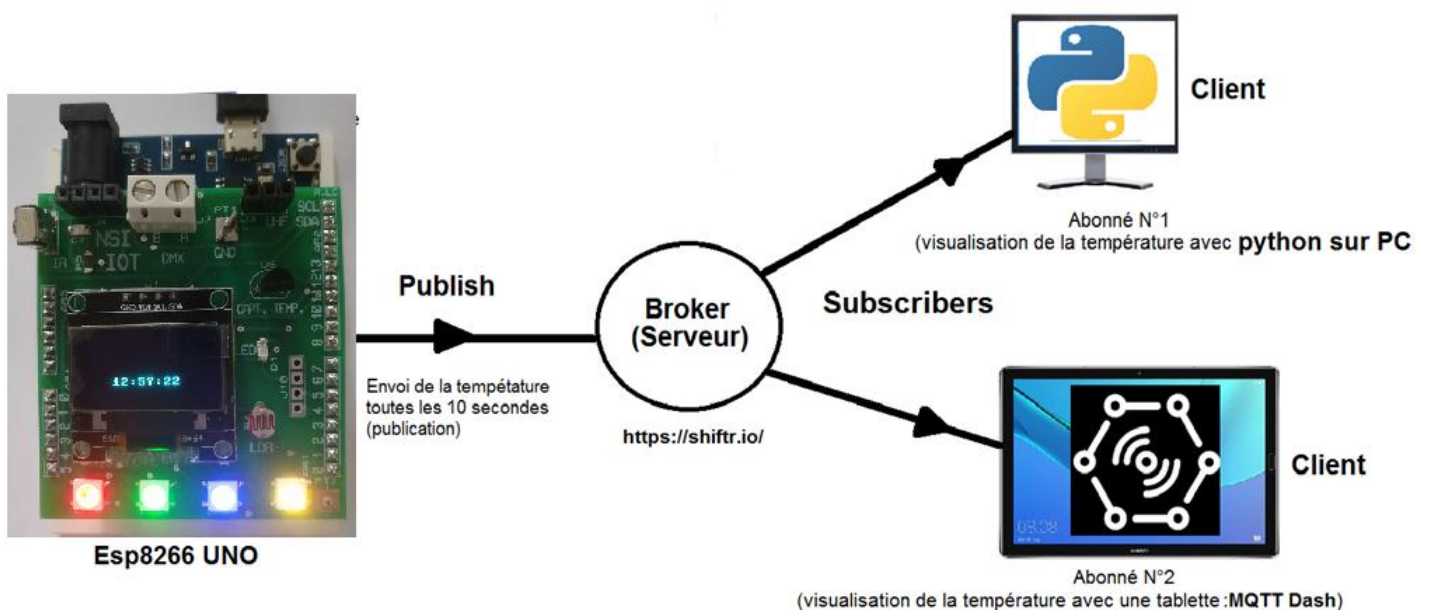
LE CREN Anthony

1 Qu'est-ce que MQTT ?

MQTT (Message Queuing Telemetry Transport) est une messagerie publish-subscribe basé sur le protocole TCP/IP. MQTT est utilisé pour les IOT (Internet of Things / objets connectés). Il est conçu comme un transport de messagerie de publication / abonnement extrêmement léger en termes de ressources.

Mise en situation :

Vous avez réalisé un système à base d'ESP8266 permettant de mesurer la température d'une pièce. Vous voulez connaître cette température quand vous êtes à l'extérieur de votre maison. A première vue, une solution serait de concevoir une page WEB afin de pouvoir y accéder depuis un navigateur. MQTT va vous permettre de remplir cet objectif plus rapidement en utilisant une bande passante très réduite. Il est aussi possible de déclencher un mécanisme à distance comme une des volets roulants etc... La communication peut ainsi être bidirectionnelle.



1 Publication de température

Q1 A l'aide de l'annexe 1, configurer votre broker sur le site <https://shiftr.io/>

Noter la clé et le mot de passe dans le tableau ci-dessous :

Adresse du serveur	broker.shiftr.io
Client id	arduino
Key	weatherSensors
Secret	bme280Sensors
Température Topic	/sensors/temperature

Q2 Configurer et tester le programme ci-dessous avec vos propres identifiants conformément à la question **Q1**. (Q2-mqtt-publish.py)

```
from umqtt.robust import MQTTClient
import network
import sys
import time
from time import sleep_ms
from machine import Pin
from onewire import OneWire
from ds18x20 import DS18X20

sta_if = network.WLAN(network.STA_IF)
print(sta_if.active())
print(sta_if.ifconfig())

myMqttClient = b"micropython"

THINGSPEAK_URL = b"broker.shiftr.io"
THINGSPEAK_USER_ID = b'weatherSensors'
THINGSPEAK_MQTT_API_KEY = b'bme280Sensors'
client = MQTTClient(client_id=myMqttClient,
                    server=THINGSPEAK_URL,
                    user=THINGSPEAK_USER_ID,
                    password=THINGSPEAK_MQTT_API_KEY,
                    ssl=False)

try:
    client.connect()
    print("connection ok");
except Exception as e:
    print('could not connect to MQTT server {}'.format(type(e).__name__, e))
    sys.exit()

bus = OneWire(Pin(12))
ds = DS18X20(bus)
capteur_temperature = ds.scan()

PUBLISH_PERIOD_IN_SEC = 10

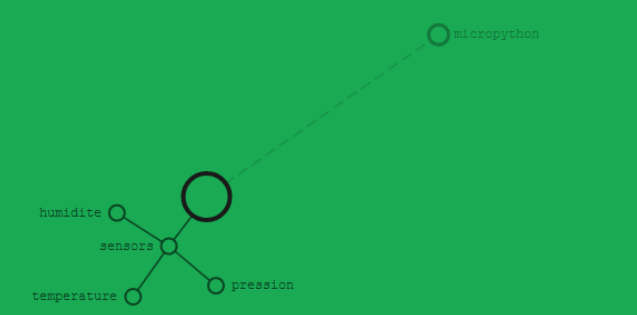
while True:
    try:
        ds.convert_temp()
        sleep_ms( 750 )
        temp_celsius = ds.read_temp(capteur_temperature[0])
        print("Température : ",temp_celsius )
        client.publish("/sensors/temperature", str(int(temp_celsius)))
        print("publish ok");
        time.sleep(PUBLISH_PERIOD_IN_SEC)
    except KeyboardInterrupt:
        print('Ctrl-C pressed...exiting')
        client.disconnect()
        sys.exit()
        print("exit")
```

Q3 Vérifier l'envoi de données au broker.

f4goh/test

sensors 1 Connection

```
sensors/temperature
26.625
micropython - 15:45:47 - Q0 NR
```



mqtt://f4goh-test@broker.shiftr.io - Public (read-only)

True

('192.168.1.104', '255.255.255.0', '192.168.1.1', '192.168.1.1')

connection ok

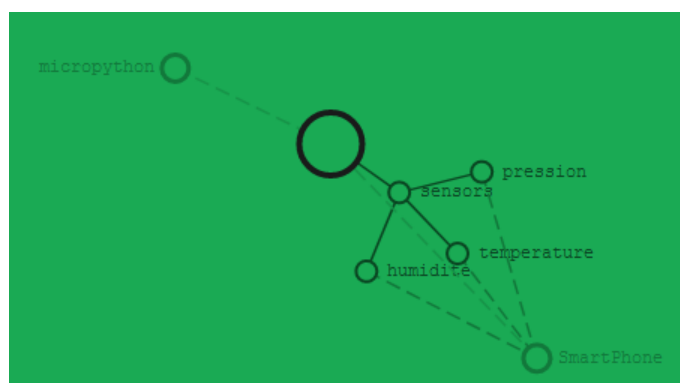
Température : 26.625

publish ok

Q4 A l'aide de l'annexe 2 ou 3, configurer le client MQTT sur un smartphone. Préférence pour **MQTT panel** : annexe 3)

Noter à nouveau la clé et le mot de passe en lien avec le broker (**Q1**) dans le tableau ci-dessous :

Name	sensorsTest
Adresse du serveur	broker.shiftr.io
Client id	Mqttdash-xxxxx (choisi par le logiciel MQTT dash)
User name (key)	weatherSensors
User password (password)	bme280Sensors
Topic name	sensors
Temperature Topic	/sensors/temperature

Q5 Vérifier l'envoi de données du broker vers le smartphone.**Q6** Identifier le subscriber et le publisher en complétant le tableau ci-dessous par oui ou non

	Subscriber	Publisher
L'esp8266 est le	Non	Oui
Le smartphone est le	Oui	Non

La qualité de service (QoS) ou quality of service (QoS) est la capacité à véhiculer dans de bonnes conditions un type de trafic donné.

<p>Other settings</p> <p><input checked="" type="radio"/> QoS(0)</p> <p><input type="radio"/> QoS(1)</p> <p><input type="radio"/> QoS(2)</p>	<ul style="list-style-type: none"> - QoS0. Le message envoyé n'est pas stocké par le Broker. Il n'y a pas d'accusé de réception. Le message sera perdu en cas d'arrêt du serveur ou du client. C'est le mode par défaut - QoS1. Le message sera livré au moins une fois. Le client renvoie le message jusqu'à ce que le broker envoie en retour un accusé de réception. - QoS2. Le broker sauvegarde le message et le transmettra jusqu'à ce qu'il ait été réceptionné par tous les souscripteurs connectés
--	--

2 Commande d'une Led

Q7 Configurer et tester le programme ci-dessous avec vos propres identifiants conformément à la question **Q1** (Q7-mqtt-subscribe-led.py)

```
from umqtt.robust import MQTTClient
import network
import sys
from machine import Pin

led = Pin(0, Pin.OUT)

def cb(topic, msg):
    print((topic, msg))
    valeur=int(msg.decode("ascii"))
    if valeur==1:
        led.on()
    elif valeur==0:
        led.off()

sta_if = network.WLAN(network.STA_IF)
print(sta_if.active())
print(sta_if.ifconfig())

myMqttClient = b"micropython"

THINGSPEAK_URL = b"broker.shiftr.io"
THINGSPEAK_USER_ID = b'weatherSensors'
THINGSPEAK_MQTT_API_KEY = b'bme280Sensors'
client = MQTTClient(client_id=myMqttClient,
                    server=THINGSPEAK_URL,
                    user=THINGSPEAK_USER_ID,
                    password=THINGSPEAK_MQTT_API_KEY,
                    ssl=False)
client.set_callback(cb)

try:
    client.connect()
    print("connection ok");
    client.subscribe("/actionneurs/led")
except Exception as e:
    print('could not connect to MQTT server {}'.format(type(e).__name__, e))
    sys.exit()

while True:
    try:
        client.wait_msg()
    except KeyboardInterrupt:
        print('Ctrl-C pressed...exiting')
        client.disconnect()
        sys.exit()
    print("exit")
```

Q8 Ajouter un panel de commande comme le montre l'exemple ci-dessous, puis valider la commande sur le site shiftr.io

Free 55% 19:32

Edit panel

Panel name*
led

Topic*
/actionneurs/led

Subscribe Topic

Payload on*
1

Payload off*
0

☒ Use icon switch

Choose on icon

☐ Choose off icon

Icon color
#27933c

Icon color
#c21818

```
sensors/temperature
26
micropython - 19:18:46 - Q0 NR
actionneurs/led
1
SmartPhone - 19:27:17 - Q0 NR
```

matt://f4goh-test@brokershiftr.io - Public (read-only)

Q9 Identifier le subscriber et le publisher en complétant le tableau ci-dessous par oui ou non

	Subscriber	Publisher
L'esp8266 est le	Oui	Non
Le smartphone est le	Non	Oui

3 Utilisation de MQTT avec un IDE Python (en remplacement du smartphone)

Prérequis : Installer paho-mqtt sur votre PC.

paho-mqtt 1.4.0

Q10 Reprendre le programme à la question **Q2 dans l'esp8266**, puis tester le programme suivant dans **un IDE python sur PC** (Q10-mqtt_subscribe_PC.py)

```
import paho.mqtt.client as mqtt

# The callback for when a PUBLISH message is received from the server.
def on_message(client, userdata, msg):
    print(msg.topic+" "+str(msg.payload))

client = mqtt.Client("python") #id must be unique
client.username_pw_set("weatherSensors", "bme280Sensors") #login, password
client.on_message = on_message

client.connect("broker.shiftr.io", 1883, 60)

client.subscribe("/sensors/temperature")

while True:
    client.loop()
```

```
/sensors/temperature b'25.6875'
/sensors/temperature b'25.6875'
/sensors/temperature b'25.6875'
```

L'information de température s'affiche sur le PC et non sur le smartphone

Q11 Identifier le subscriber et le publisher en complétant le tableau ci-dessous par oui ou non

	Subscriber	Publisher
L'esp8266 est le	Non	Oui
Le PC est le	Oui	Non

Q12 Reprendre le programme à la question **Q7** dans l'**esp8266**, puis tester le programme suivant dans un **IDE python sur PC**. (Q12-mqtt_publish_PC.py)

```
import paho.mqtt.client as mqtt

client = mqtt.Client("python") #id must be unique
client.username_pw_set("weatherSensors", "bme280Sensors") #login, password

client.connect("broker.shiftr.io", 1883, 60)

while True:
    #client.loop()
    valeur=input("saisir l'etat de la led 0/1 ?")
    client.publish("/actionneurs/led", valeur)
```

saisir l'etat de la led 0/1 ?1

saisir l'etat de la led 0/1 ?0

La Led câblée sur l'ESP8266 doit s'allumer ou s'éteindre en fonction de la commande 0 ou 1

Q13 Identifier le subscriber et le publisher en complétant le tableau ci-dessous par oui ou non

	Subscriber	Publisher
L'esp8266 est le	Oui	Non
Le PC est le	Non	Oui

Mini projets


Q14 Utiliser le smartphone pour commander le ruban de Led avec des couleurs préprogrammées.

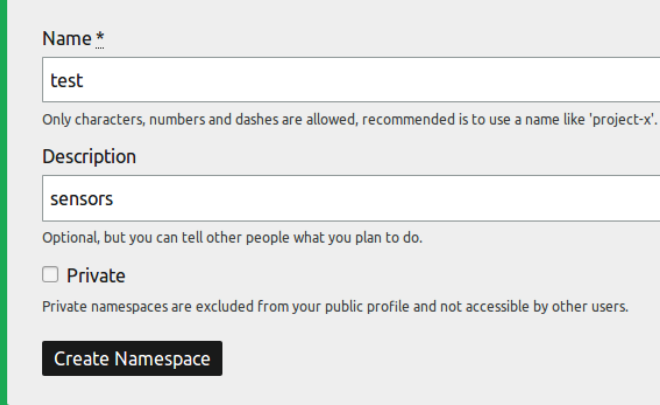
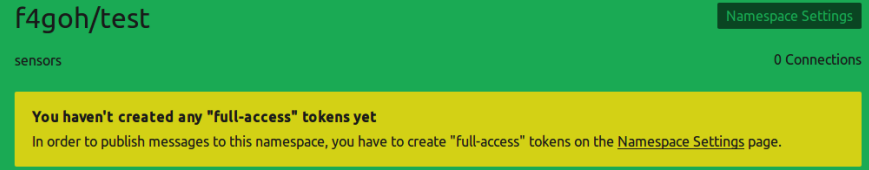
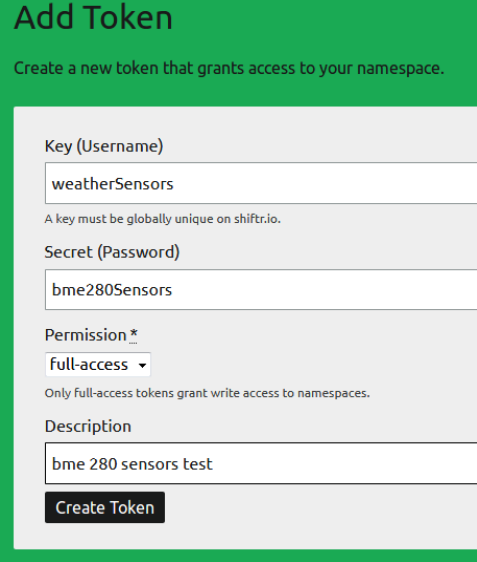
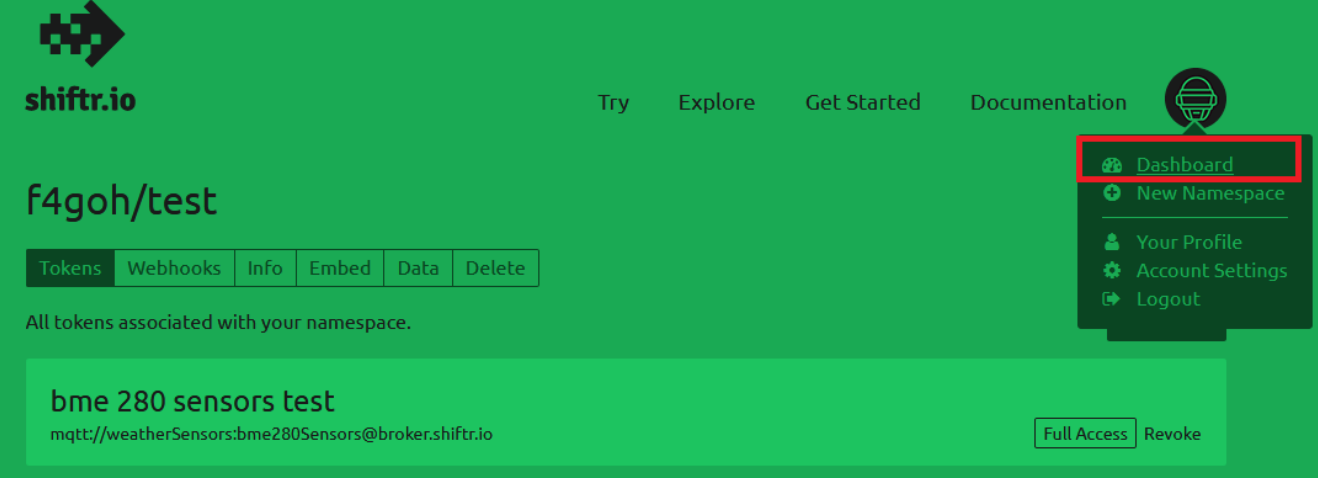
Q15 Le smartphone envoie un texte sur l'afficheur OLED.

Q16 Le smartphone reçoit une alerte d'intrusion détectée par le capteur de lumière.

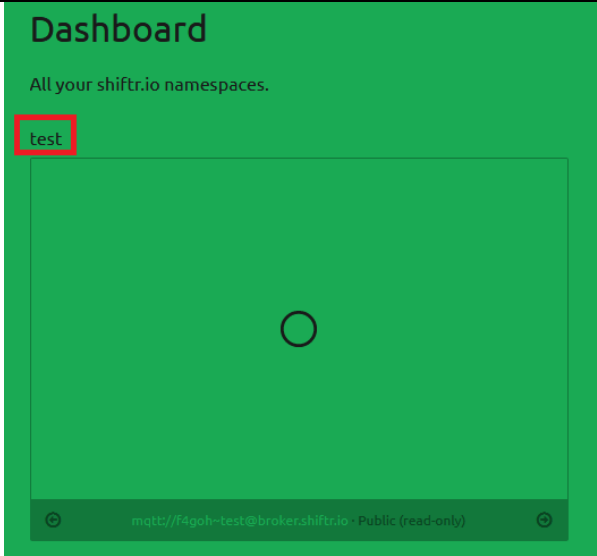
Q17 Votre projet personnel (le programme de votre choix)

Annexe 1 : Configuration d'un broker (serveur MQTT) en ligne

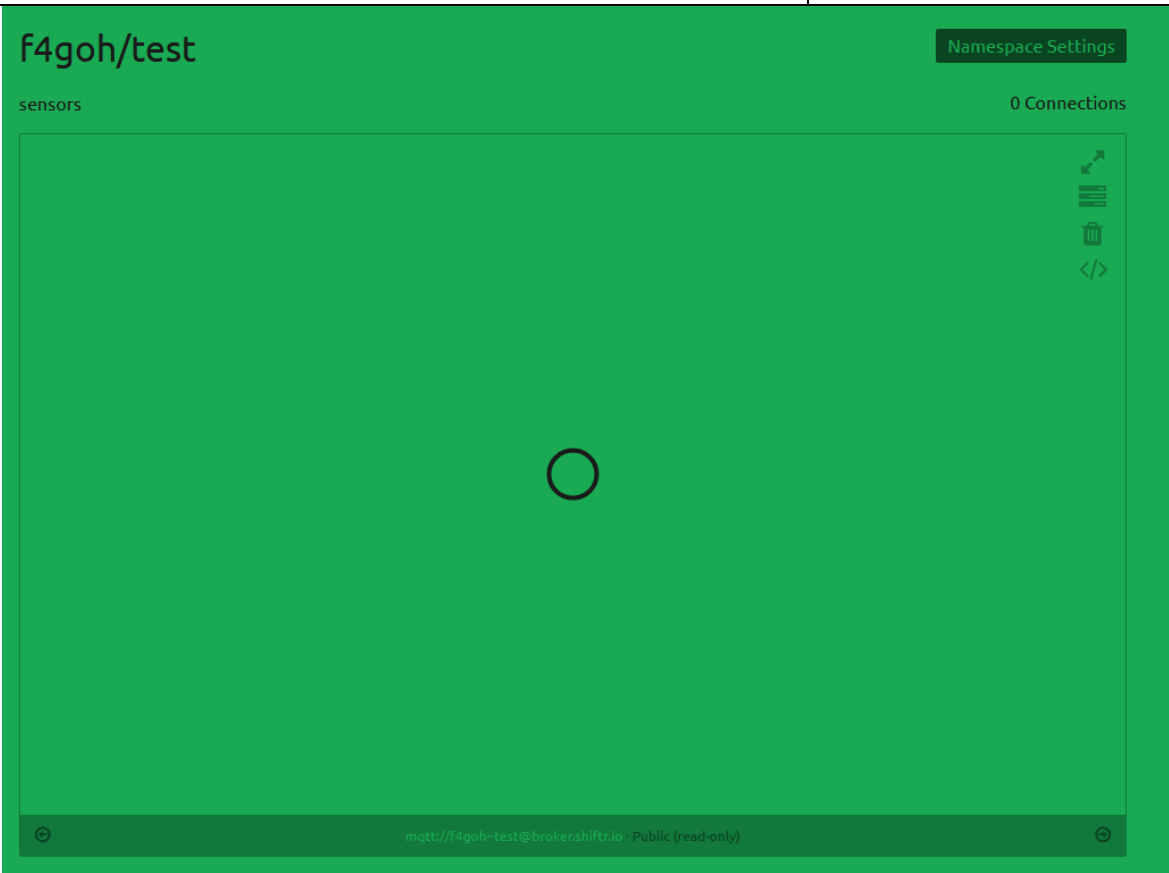
 <p>shiftr.io</p> <p>You have to confirm your email address before continuing.</p> <p>Sign in</p> <p>user@example.com</p> <p>****</p> <p>Sign in</p>	<p>Créer un compte sur le site https://shiftr.io/</p>
<p>Welcome f4goh!</p> <p>You can confirm your account email through the link below:</p> <p>Confirm my account</p>	<p>Confirmer le compte à partir de votre adresse mail</p>
<p>Signed in successfully.</p> <p>Welcome to shiftr.io!</p> <p>The following three steps let you get started with the platform and learn all the necessary things.</p> <p>1. Namespace</p> <p>In order to connect things to shiftr.io you first need to create a namespace.</p> <p>Namespaces act as isolated virtual brokers that have their own topic hierarchy. You can create any amount of namespaces, recommended is to create at least one for each of your projects.</p> <p>Try Explore Get Started Documentation</p> <p>Dashboard New Namespace</p>	<p>Créer un « namespace » pour chaque projets</p>

 <p>Name *</p> <p>test</p> <p>Only characters, numbers and dashes are allowed, recommended is to use a name like 'project-x'.</p> <p>Description</p> <p>sensors</p> <p>Optional, but you can tell other people what you plan to do.</p> <p><input type="checkbox"/> Private</p> <p>Private namespaces are excluded from your public profile and not accessible by other users.</p> <p>Create Namespace</p>	<p>Exemple :</p> <p>test</p> <p>sensors</p>
 <p>f4goh/test</p> <p>sensors</p> <p>0 Connections</p> <p>You haven't created any "full-access" tokens yet</p> <p>In order to publish messages to this namespace, you have to create "full-access" tokens on the Namespace Settings page.</p>	<p>Ajouter un token (jeton)</p> <p>Namespace Settings, Add token</p>
 <p>Add Token</p> <p>Create a new token that grants access to your namespace.</p> <p>Key (Username)</p> <p>weatherSensors</p> <p>A key must be globally unique on shiftr.io.</p> <p>Secret (Password)</p> <p>bme280Sensors</p> <p>Permission *</p> <p>full-access</p> <p>Only full-access tokens grant write access to namespaces.</p> <p>Description</p> <p>bme 280 sensors test</p> <p>Create Token</p>	<p>Key et Secret 8 caractères minimum</p>
 <p>shiftr.io</p> <p>Try Explore Get Started Documentation</p> <p>f4goh/test</p> <p>Tokens Webhooks Info Embed Data Delete</p> <p>All tokens associated with your namespace.</p> <p>bme 280 sensors test</p> <p>mqtt://weatherSensors:bme280Sensors@broker.shiftr.io</p> <p>Full Access Revoke</p> <p>Dashboard</p> <p>New Namespace</p> <p>Your Profile</p> <p>Account Settings</p> <p>Logout</p>	

Aller sur le Dashboard (Tableau de bord)



Agrandir le Dashboard



La configuration du Broker (Serveur MQTT) est terminée.

Exemple de configuration

Adresse du serveur	broker.shifttr.io
Client id	arduino
Key	weatherSensors
Secret	bme280Sensors
Temperature Topic	/sensors/temperature
Pression Topic	/sensors/pression
Humidité Topic	/sensors/humidite

Annexe 2 : Configuration du client sur Android (MQTT Dash)

Installer le programme MQTT dash



MQTT Dash (IoT, Smart Home)

Routix software Communication

★★★★★ 1 911

3 PEGI 3

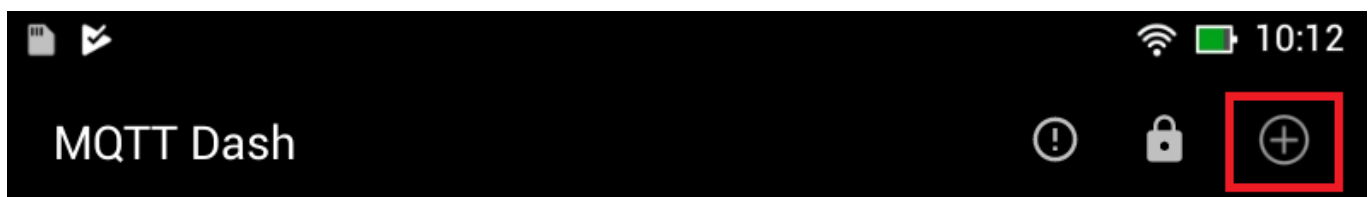
Cette application est compatible avec vos appareils.

Ajouter à la liste de souhaits

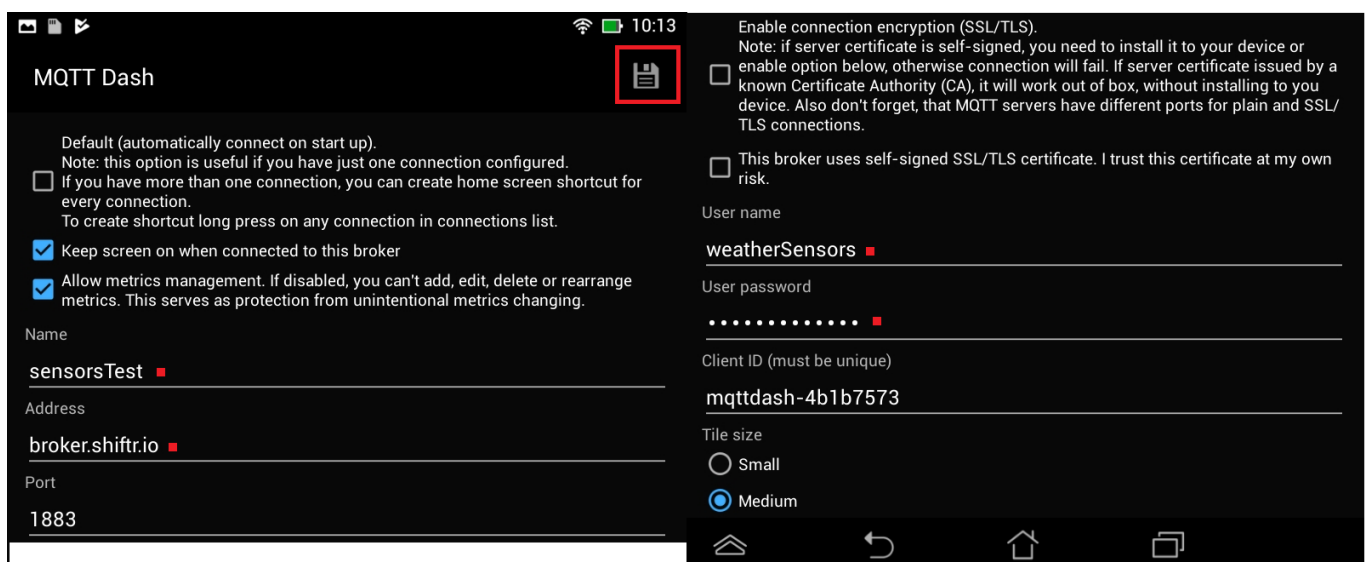
Installer

<https://play.google.com/store/apps/details?id=net.routix.mqttdash>

Configuration :



Ajouter la connexion au broker



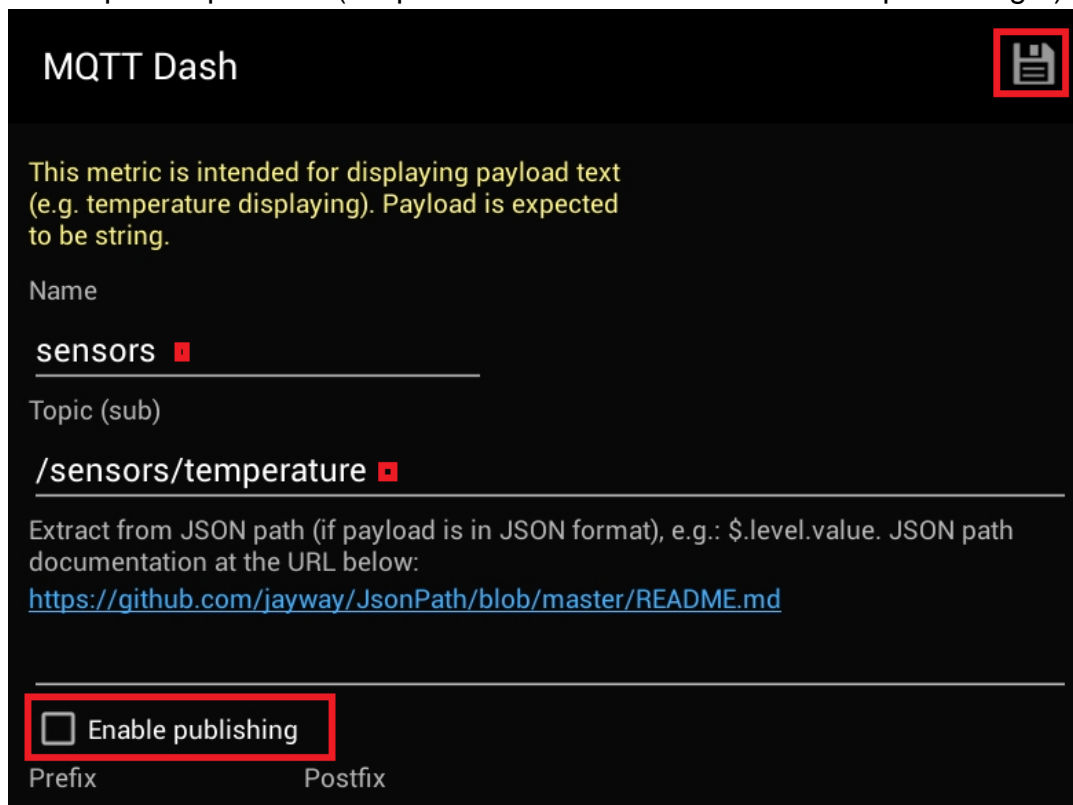
Cliquer sur sensorsTest



Puis ajouter les topics nécessaires.



Exemple pour le Topic température (Ne pas oublier de décocher « Enable publishing »)



Annexe 3 : Configuration du client sur Android (MQTT panel)

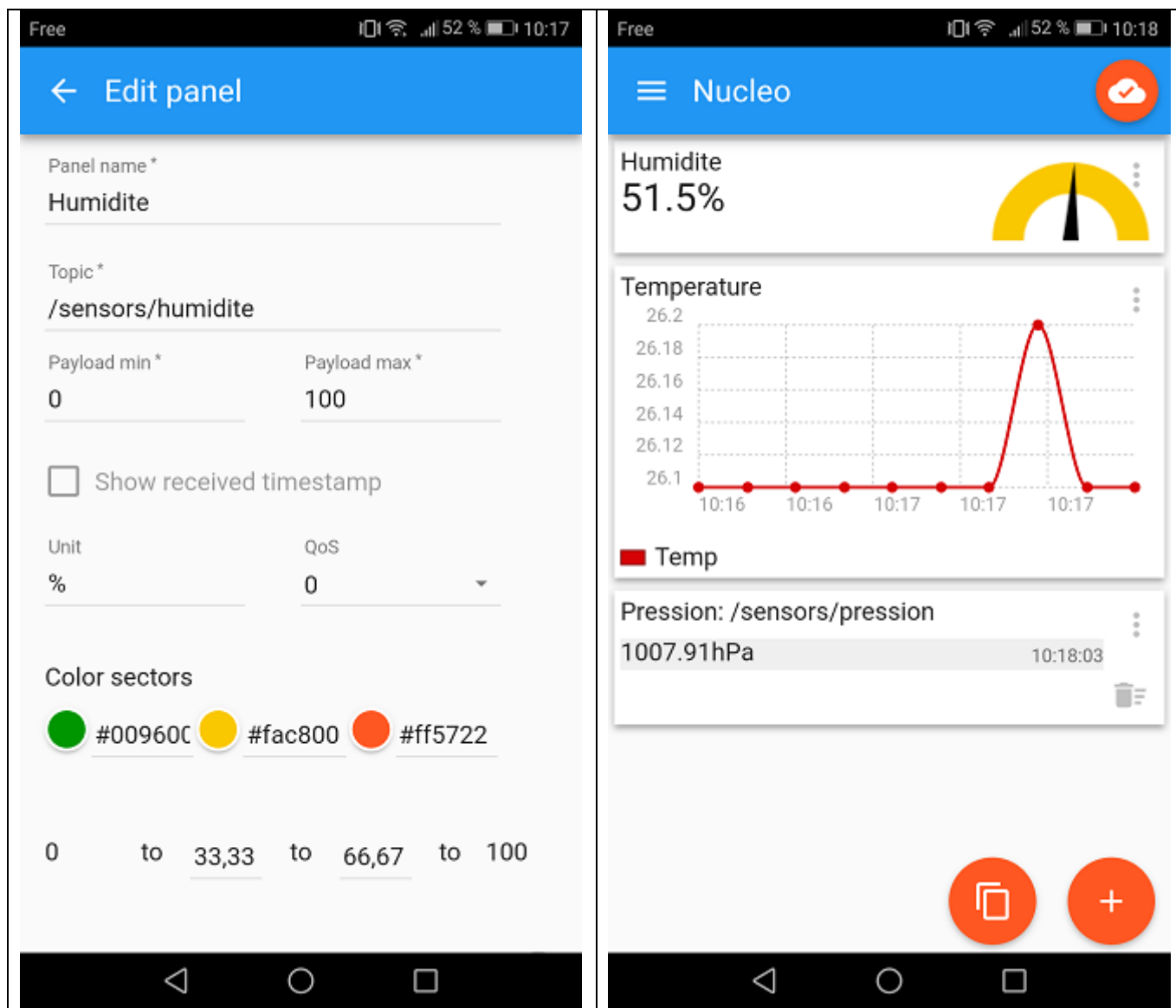
Installer le programme IoT MQTT Panel



<https://play.google.com/store/apps/details?id=snr.lab.iotmqttpanel.prod>

Ce programme un plus convivial que MQTT dash et possède des widgets. (Élément de base de l'interface graphique d'un logiciel : fenêtre, barre d'outils, par exemple).

Configuration de la connexion au Broker	Ajout des panels
Exemple pour l'humidité	Rendu final



Exemple de configuration en lien avec le broker :

Name	sensorsTest
Adresse du serveur	broker.shiftr.io
Client id	Mqttdash-xxxxx (choisi par le logiciel MQTT dash)
User name (key)	weatherSensors
User password (password)	bme280Sensors
Topic name	sensors
Temperature Topic	/sensors/temperature

Port 1881 : communication non sécurisée (données en clair sur le réseau) par défaut dans ce document.

Port 8881 : communication non sécurisé SSL (Secure Socket Layer) / TLS (Transport Layer Security)