

MQTT AVEC ARDUINIO ET PROCESSING

L'objectif est de mettre en œuvre un environnement MQTT complet que se soit avec un Arduino Ethernet ou avec le logiciel de programmation Processing.

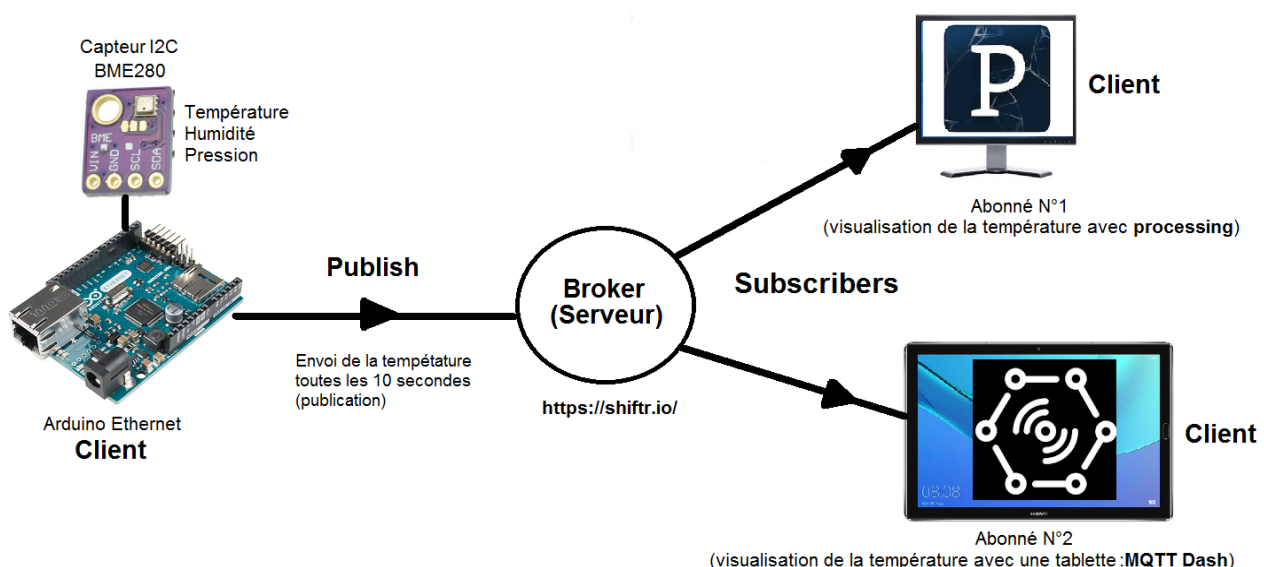
| | |
|---|---|
| 1 Qu'est-ce que MQTT ? | <ul style="list-style-type: none"> Mise en situation |
| 2 Configuration d'un broker en ligne | <ul style="list-style-type: none"> Inscription et configuration Bilan de la configuration |
| 3 Utilisation avec un Arduino Ethernet | <ul style="list-style-type: none"> Partie matérielle Installation de la librairie Utilisation du programme d'exemple |
| 4 Utilisation avec Processing | <ul style="list-style-type: none"> Installation de la librairie Analyse du programme d'exemple |
| 5 Utilisation avec une tablette Android | <ul style="list-style-type: none"> Configuration du logiciel MQTT dash |

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.

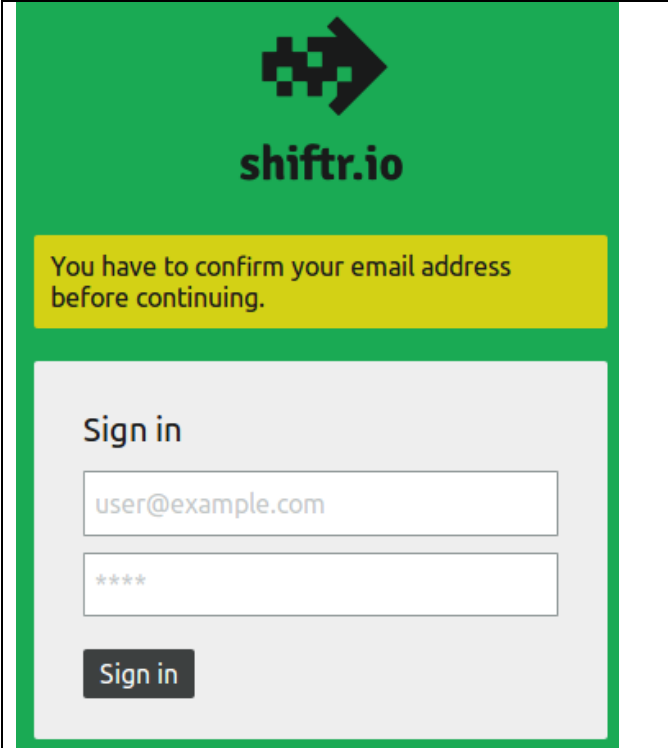
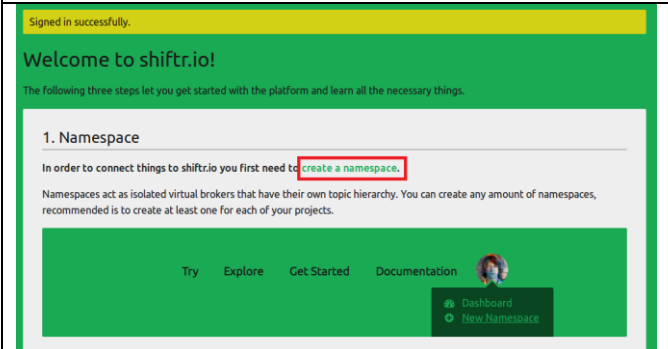
Mise en situation :


Vous avez réalisé un système à base d'Arduino Ethernet 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.



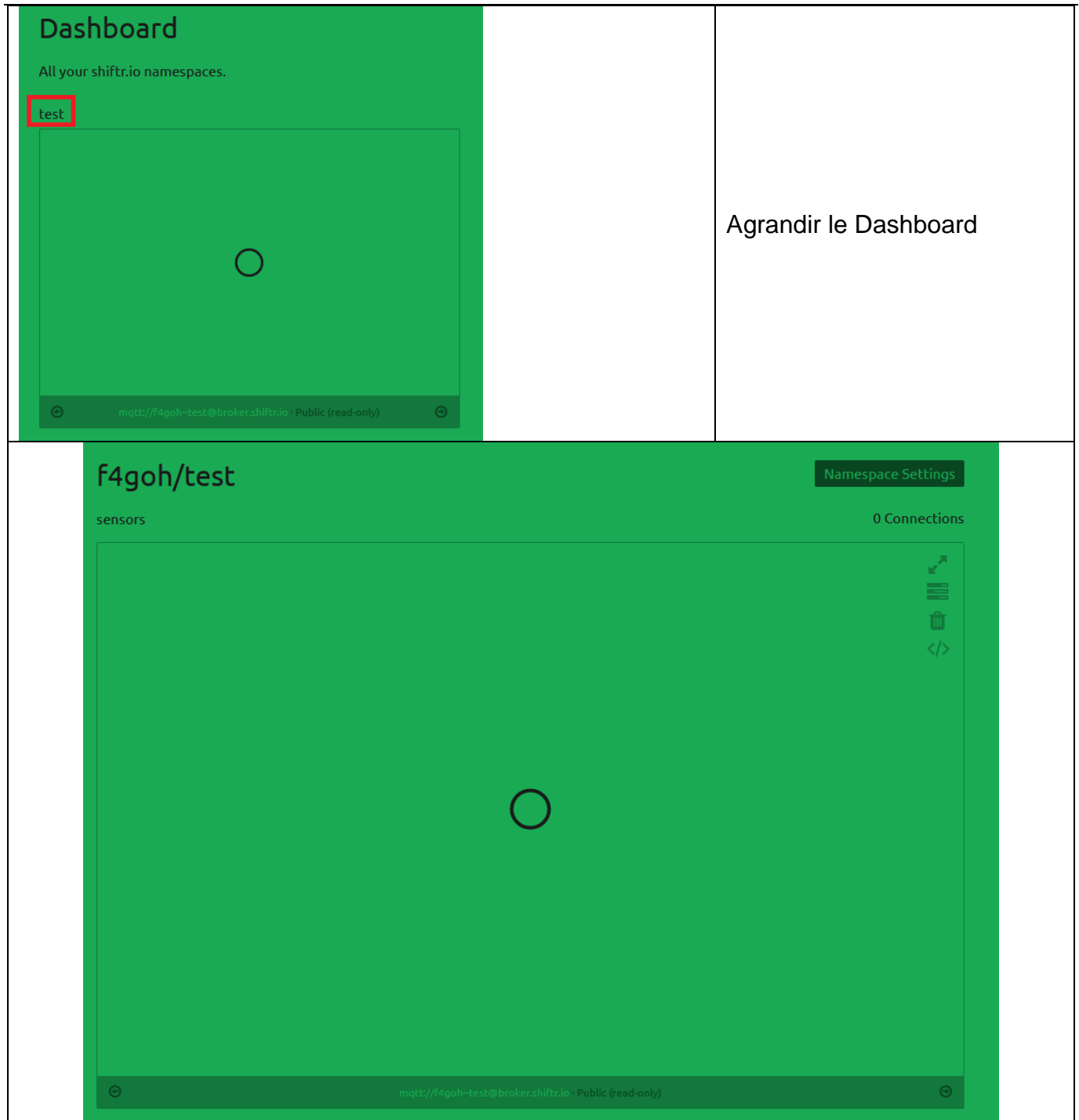
La suite du document va expliquer la configuration du serveur <https://shiftr.io/> puis l'envoi des 3 paramètres température, pression, humidité au serveur à partir d'un Arduino ethernet (publish). Enfin la réception des données avec Processing et une tablette (Subscribe).

2 Configuration d'un broker (serveur MQTT) en ligne

| | |
|---|--|
|  | <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>Créer un « namespace » pour chaque projets</p> |

| | | |
|--|---|---|
| <p>Name *</p> <input type="text" value="test"/> <p>Only characters, numbers and dashes are allowed, recommended is to use a name like 'project-x'.</p> <p>Description</p> <input type="text" value="sensors"/> <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>Namespace Settings</p> | <p>Ajouter un token (jeton) 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> <input type="text" value="weatherSensors"/> <p>A key must be globally unique on shiftr.io.</p> <p>Secret (Password)</p> <input type="text" value="bme280Sensors"/> <p>Permission *</p> <p>full-access ▾</p> <p>Only full-access tokens grant write access to namespaces.</p> <p>Description</p> <input type="text" value="bme 280 sensors test"/> <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)



La configuration du Broker (Serveur MQTT) est terminée.

Fichiers Arduino et Processing ICI

<https://github.com/f4goh/MQTT-Tutoriel>

Bilan de la configuration

Arduino ethernet

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

Processing

| | |
|--------------------|--|
| Adresse du serveur | mqtt://weatherSensors:bme280Sensors@broker.shiftr.io |
| Client id | processing |
| Key | weatherSensors |
| Secret | bme280Sensors |
| Temperature Topic | /sensors/temperature |
| Pression Topic | /sensors/pression |
| Humidité Topic | /sensors/humidite |



Tablette Android avec MQTT dash

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

| | |
|--------------------------|---|
| 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 |
| Pression Topic | /sensors/pression |
| Humidité Topic | /sensors/humidite |

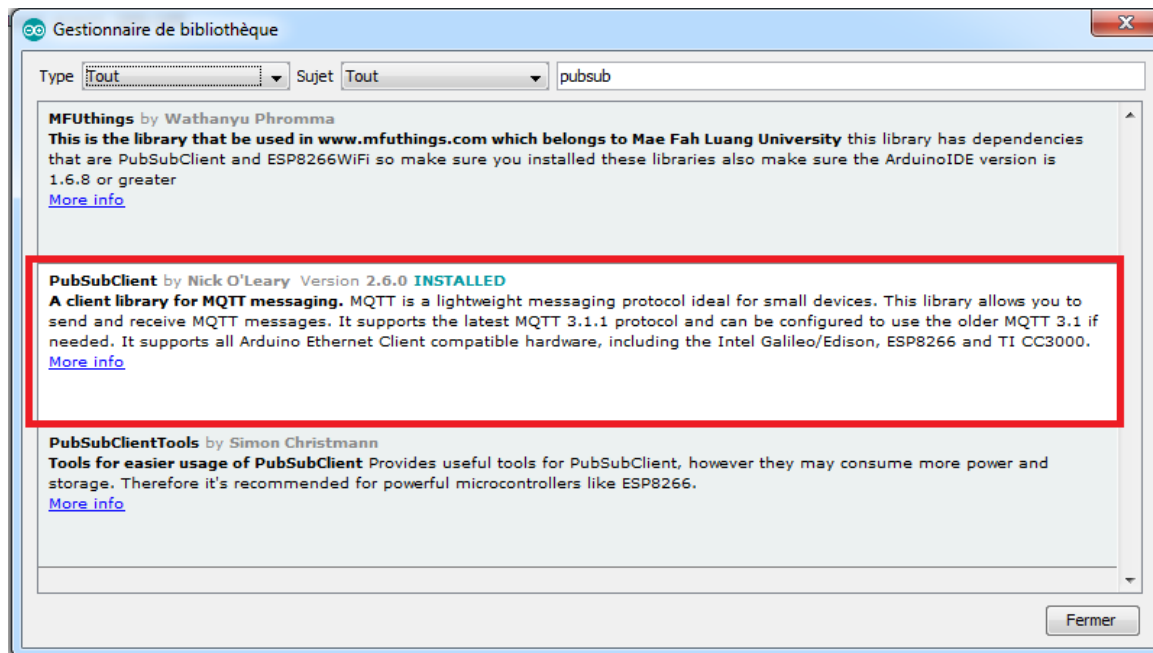
Port 1881 : communication non sécurisée (données en clair sur le réseau) par défaut dans ce document.

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

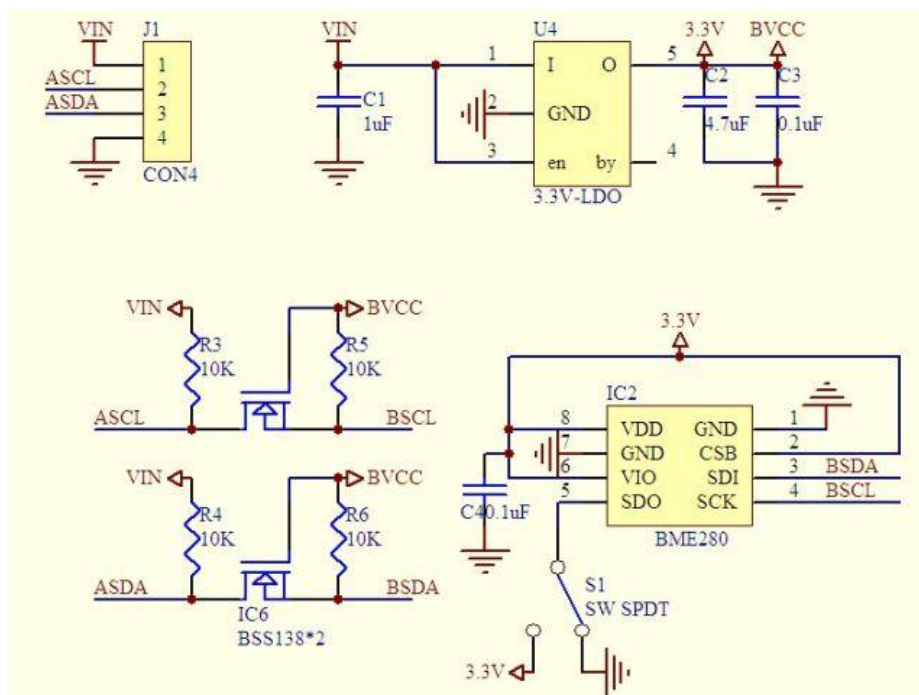
En rouge : obligatoire en lien avec la configuration du broker

3 Utilisation avec un Arduino Ethernet

Installer la librairie à partir de l'IDE Arduino



Le capteur utilisé est un GYbme280 possédant un adaptateur de bus I2C 3.3V vers 5V



Par défaut l'adresse I2C du composant est 0x76

Télécharger la librairie ici : https://github.com/sparkfun/SparkFun_BME280_Arduino_Library

Charger et programmer l'exemple fourni : « **mqtt_basic_bme.ino** »

```
byte mac[] = { 0xDE, 0xED, 0xBA, 0xFE, 0xFE, 0xED };  
IPAddress ip(192, 168, 1, 100); //adresse de l'arduino client  
const char * server = "broker.shiftr.io"; //adresse du serveur
```

```
EthernetClient ethClient;
```

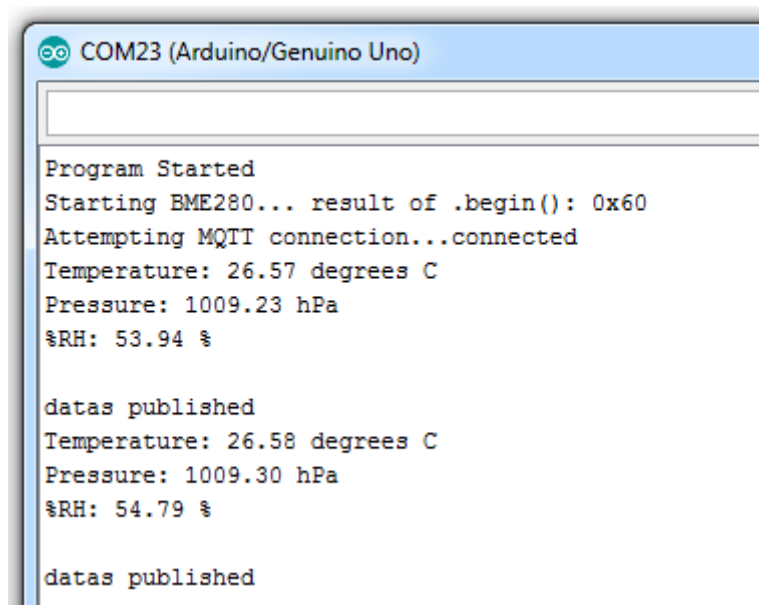
```
PubSubClient mqtt(ethClient);
```

```
BME280 mySensor;
```

```
void setup()  
{  
  Serial.begin(57600);  
  
  Ethernet.begin(mac, ip);  
  
  mqtt.setServer(server, 1883); //adresse et port du serveur  
  
  mqtt.setCallback(callback); //au cas ou l'arduino devient subscriber  
  
  initBme();  
  
  delay(1500);  
}  
  
void loop()  
{  
  if (mqtt.connected()) { //si connection au serveur publish  
    pubCapteur();  
    mqtt.loop();  
  }  
  else {  
    reconnect(); //sinon tentative de reconnection  
  }  
}
```

Résultats à l'écran

Coté client (Arduino Ethernet)



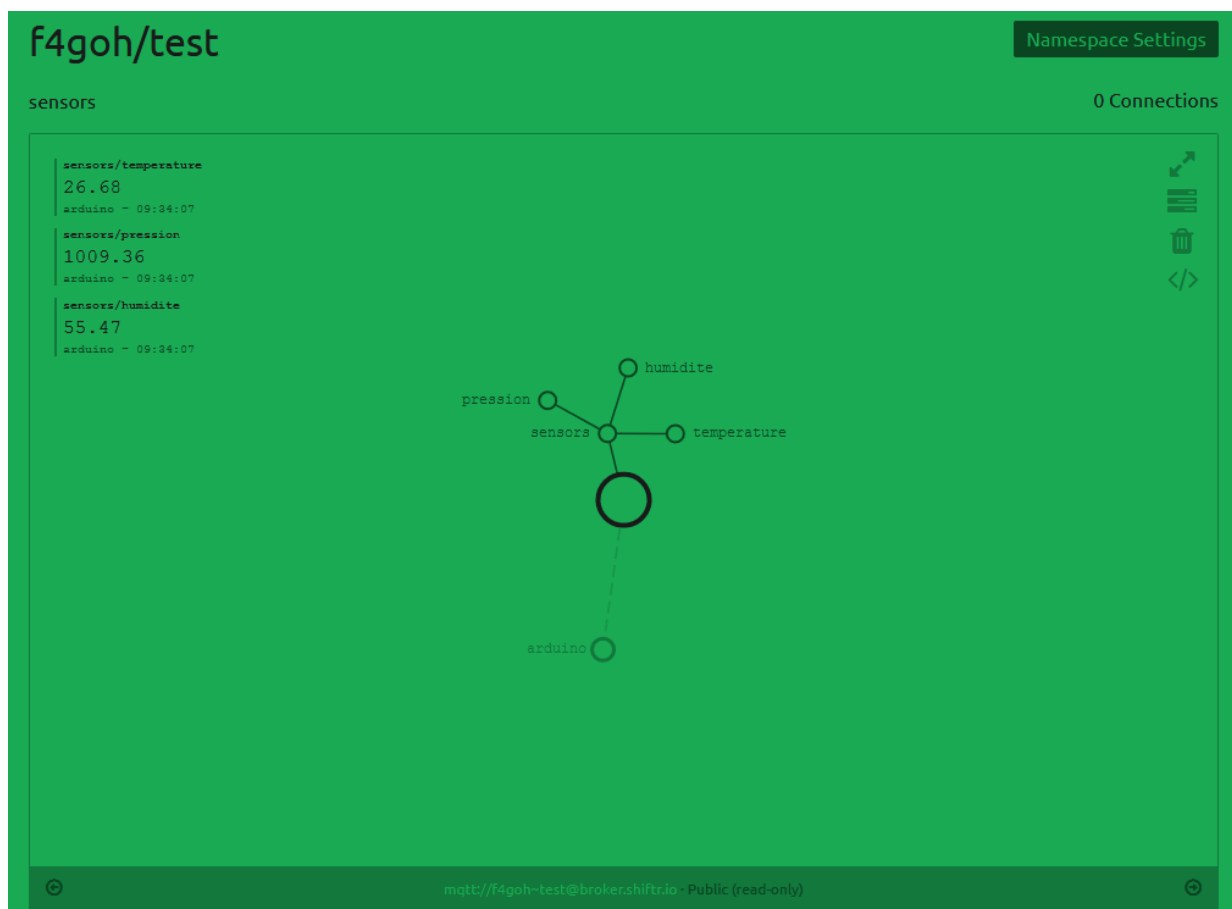
```
COM23 (Arduino/Genuino Uno)

Program Started
Starting BME280... result of .begin(): 0x60
Attempting MQTT connection...connected
Temperature: 26.57 degrees C
Pressure: 1009.23 hPa
%RH: 53.94 %

datas published
Temperature: 26.58 degrees C
Pressure: 1009.30 hPa
%RH: 54.79 %

datas published
```

Coté serveur (dans le Dashboard)



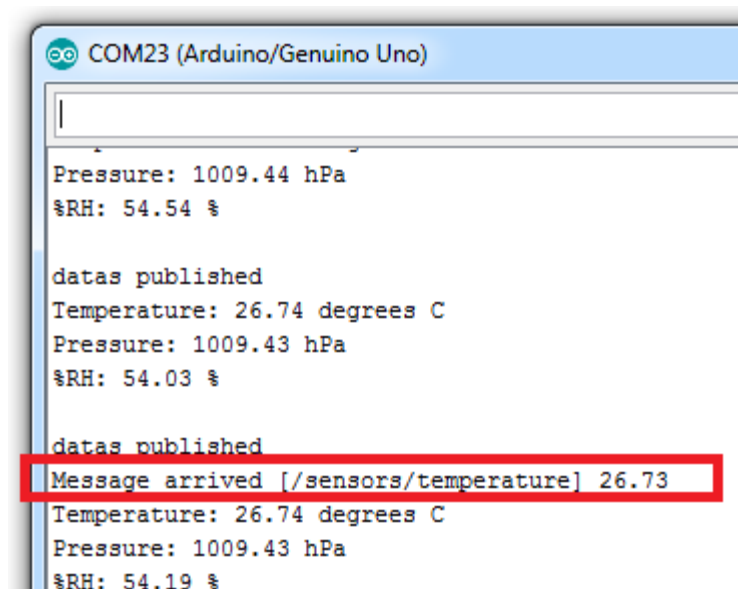
Les valeurs du capteur apparaissent sous la forme d'un arbre en fonction du « Topic » réalisé

Arduino en Subscriber

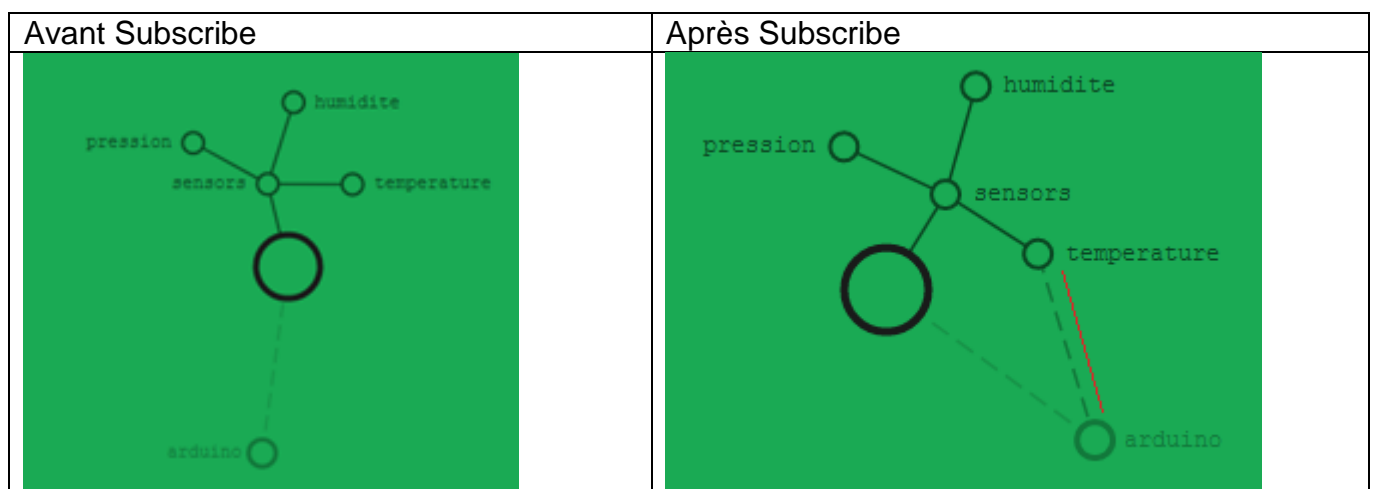
Retirer le commentaire de la ligne

```
mqtt.subscribe("/sensors/temperature");
```

On constate le retour de l'information de température



Le Dashboard a changé de forme



Lien en pointillés du retour de l'information température

Il est possible d'ajouter

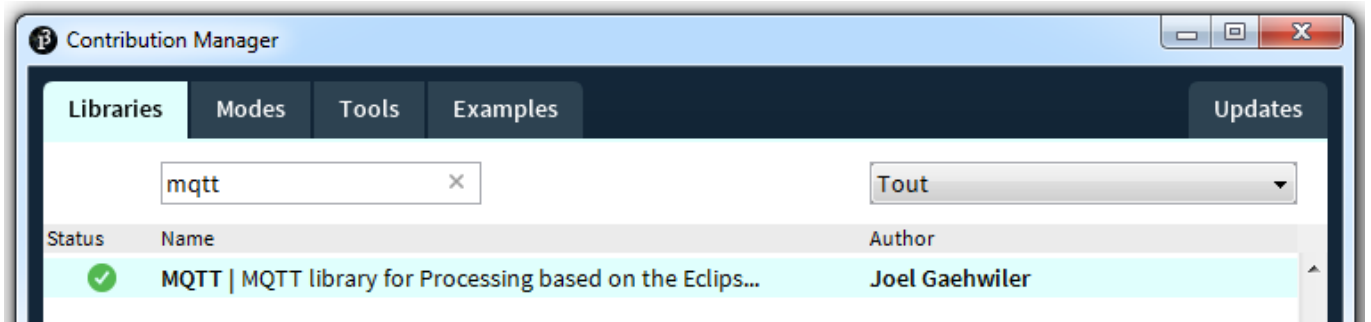
```
mqtt.subscribe("/sensors/humidite");
mqtt.subscribe("/sensors/pression");
```

L'arduino peut envoyer des informations et en recevoir sans passer par une page WEB

4 Utilisation avec Processing

L'objectif est de récupérer les données précédentes avec Processing

Installer la librairie à partir de l'IDE Processing



Charger et programmer l'exemple fourni : « **PublishSubscribe.pde** »

```
// This example sketch connects to shiftr.io  
// https://github.com/256dpi/processing-mqtt
```

```
import mqtt.*;
```

```
MQTTClient client;
```

```
void setup() {  
  client = new MQTTClient(this);  
  client.connect("mqtt://weatherSensors:bme280Sensors@broker.shiftr.io", "processing");  
  client.subscribe("/sensors/temperature");  
  client.subscribe("/sensors/humidite");  
  client.subscribe("/sensors/pression");  
  // client.unsubscribe("/sensors/temperature");  
}
```

```
void draw() {}
```

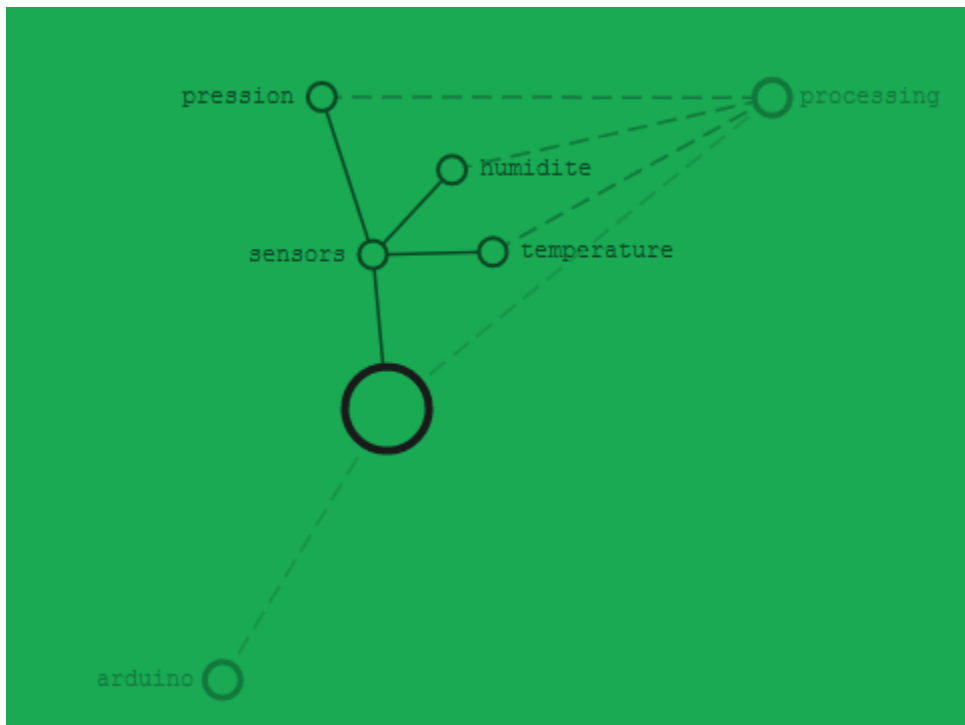
```
void keyPressed() {  
  // client.publish("/sensors/temperature", "20");  
}
```

```
void messageReceived(String topic, byte[] payload) {  
  println("new message: " + topic + " - " + new String(payload));  
}
```

Résultats à l'écran :

```
MQTT 1.6.3 by Joel Gaehwiler https://github.com/256dpi  
[MQTT] connected to: tcp://broker.shiftr.io  
new message: /sensors/temperature - 26.77  
new message: /sensors/pression - 1009.46  
new message: /sensors/humidite - 52.20
```

Le Dashboard a également changé de forme



On observe bien le client Processing en mode Subscriber (traits en pointillés)

Pour publier une donnée (changer la valeur de la température), retirer les commentaires de la ligne suivante :

```
client.publish("/sensors/temperature", "20");
```

Relancer le programme et appuyer sur une touche

5 Utilisation avec une tablette Android

Installer le programme MQTT dash



MQTT Dash (IoT, Smart Home)

Routix software Communication

★★★★★ 1 911

3 PEGI 3

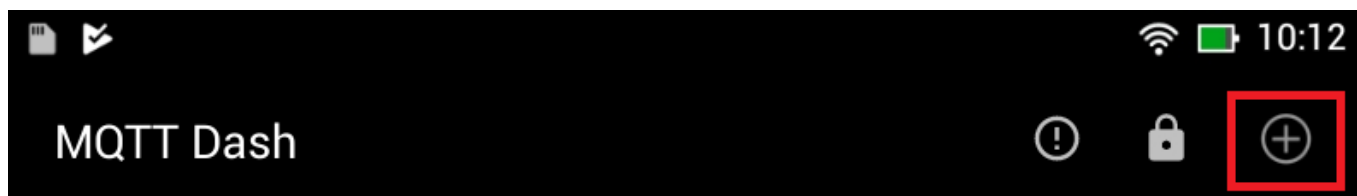
Cette application est compatible avec vos appareils.

Ajouter à la liste de souhaits

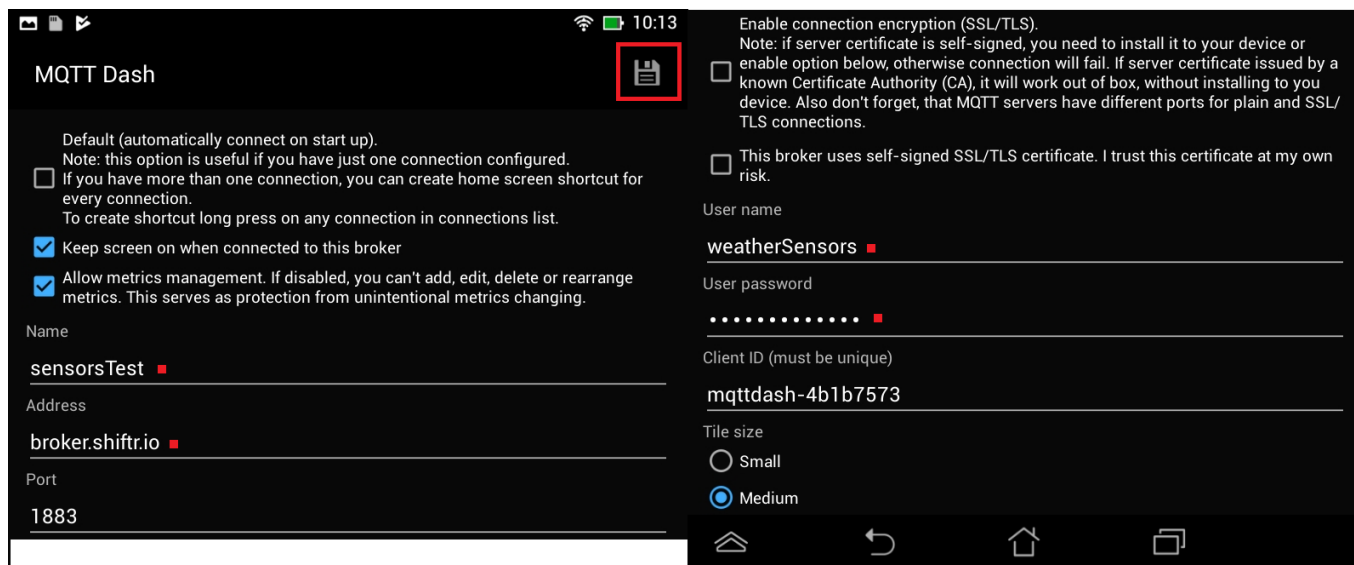
Installer

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

Configuration :



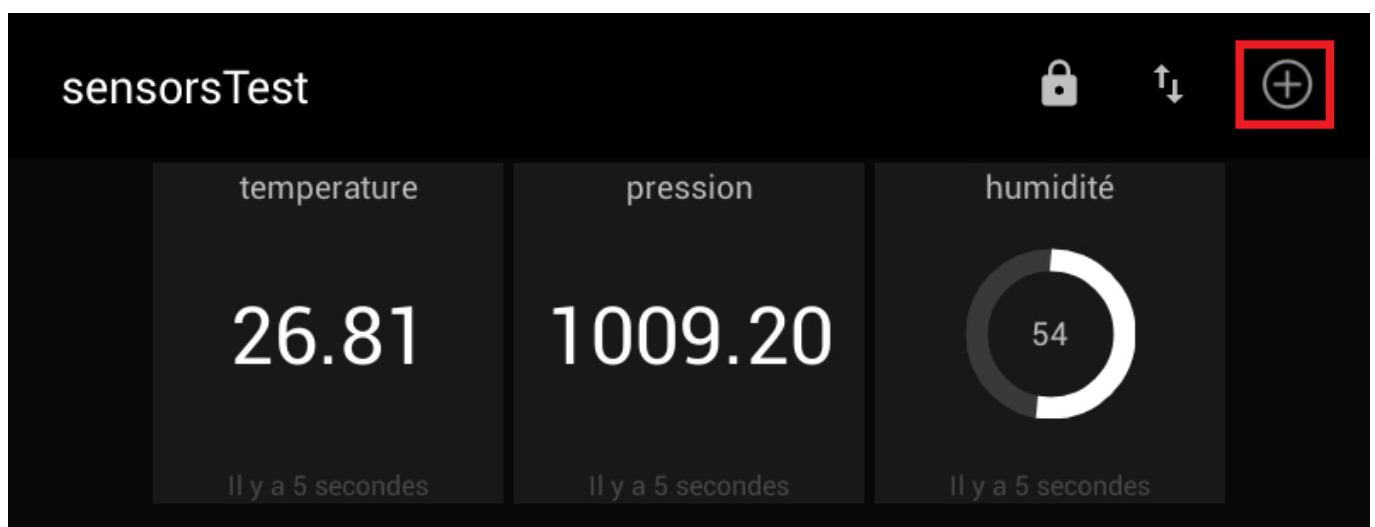
Ajouter la connexion au broker



Cliquer sur sensorsTest



Puis ajouter les 3 topics



La mise a jour des informations aura lieu toutes les 10 secondes

Exemple pour le Topic température

Ne pas oublier de décocher « Enable publishing »

MQTT Dash

This metric is intended for displaying payload text (e.g. temperature displaying). Payload is expected to be string.

Name
sensors

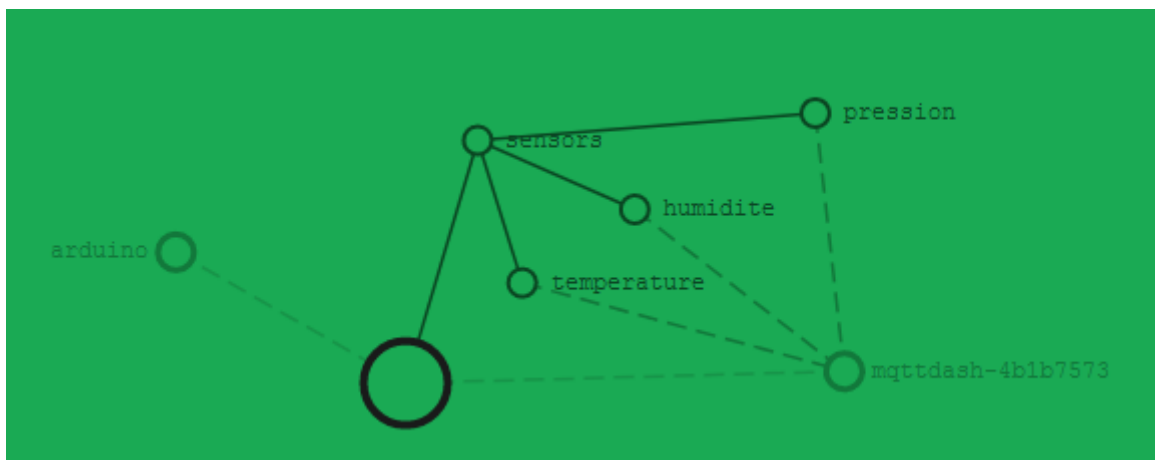
Topic (sub)
/sensors/temperature

Extract from JSON path (if payload is in JSON format), e.g.: \$.level.value. JSON path documentation at the URL below:
<https://github.com/jayway/JsonPath/blob/master/README.md>

☐ Enable publishing

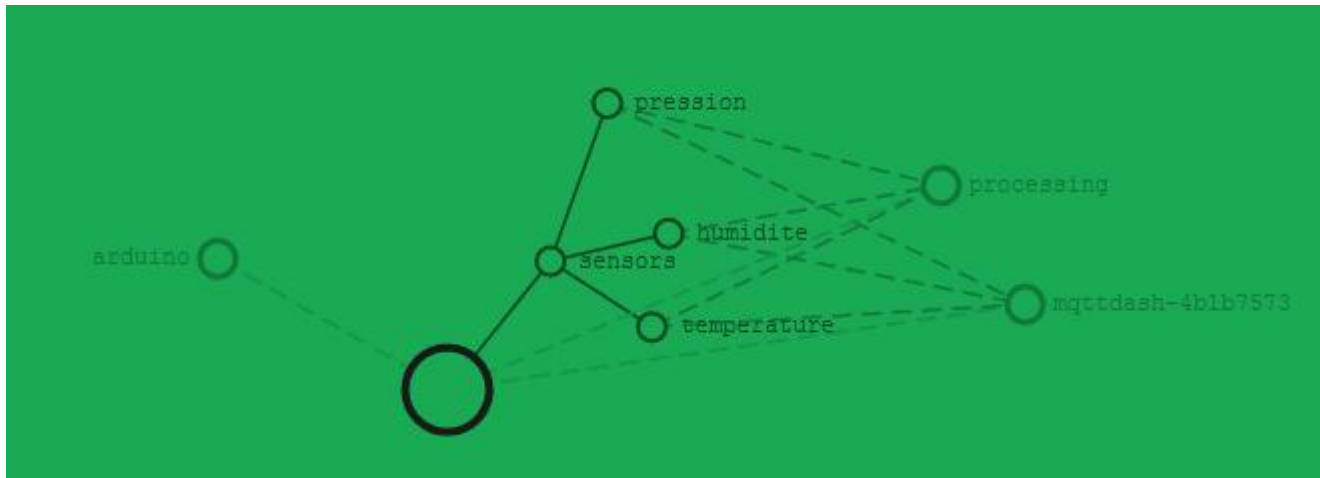
Prefix Postfix

Le Dashboard a changé de forme à nouveau



Mqtttdash-4b1b7573 est le nom de la tablette (client ID)

Le Dashboard complet avec un publisher (Arduino) et deux Subscriber (processing et tablette)



Informations complémentaires :

| | |
|--|--|
| Other settings <input checked="" type="radio"/> QoS(0) <input type="radio"/> QoS(1) <input type="radio"/> QoS(2) | <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 |
|--|--|

Conclusion

Le protocole MQTT est très bien adapté aux IOT et la mise en œuvre autour de l'Arduino est très simple à réaliser grâce aux bibliothèques prêtes à l'emploi.

Un deuxième document sera réalisé en utilisant un Broker sur un Raspberry Pi (mosquito)

Il est possible de réaliser ce tutoriel sans le module BME280. Utiliser une entrée analogique sur A0 avec un potentiomètre simulant un capteur.

Une vidéo présentant un projet similaire :

<https://www.youtube.com/watch?v=QJe87WXkeWo>