

MQTT AVEC NUCLEO ET PROCESSING

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

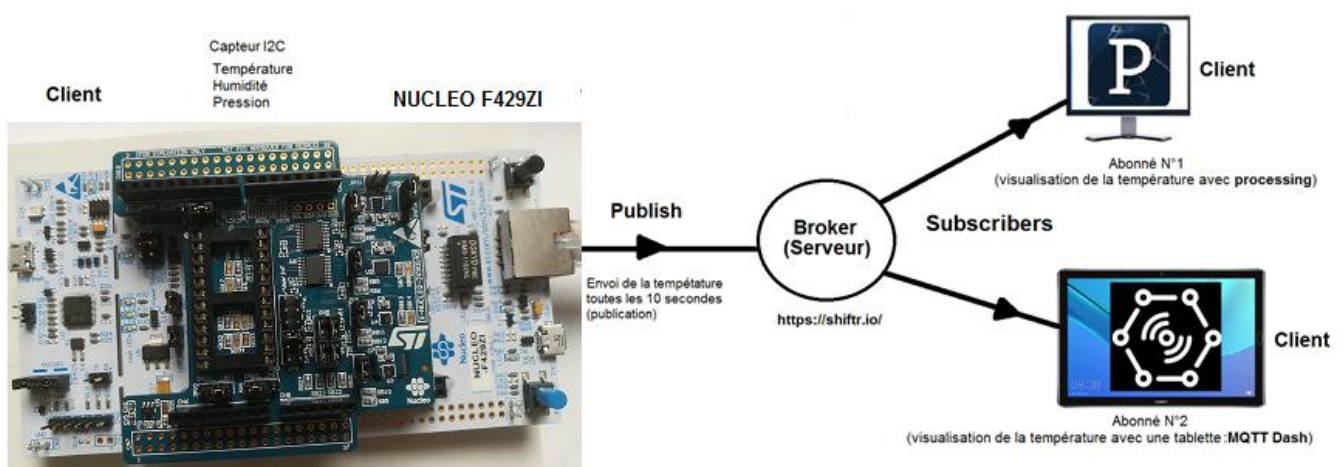
1 Qu'est-ce que MQTT ?	<ul style="list-style-type: none"> Mise en situation
2 Configuration d'un broker en ligne	<ul style="list-style-type: none"> Inscription et configuration Bilan de la configuration
3 Programmation de la NUCLEO	<ul style="list-style-type: none"> Bien démarrer avec la carte NUCLEO F429ZI Gestion des Leds et du bouton poussoir
4 Utilisation avec un NUCLEO Ethernet	<ul style="list-style-type: none"> Partie matérielle Installation de la librairie Utilisation du programme d'exemple
5 Utilisation avec Processing	<ul style="list-style-type: none"> Installation de la librairie Analyse du programme d'exemple
6 et 7 Utilisation avec une tablette Android	<ul style="list-style-type: none"> Configuration du logiciel MQTT dash et MQTT Panel
8 Le protocole MQTT	<ul style="list-style-type: none"> Analyse des trames MQTT avec wireshark

1 Qu'est-ce que MQTT ?

MQTT (**M**essage **Q**ueuing **T**elemetry **T**ransport) est une messagerie publish-subscribe basée sur le protocole TCP/IP. MQTT est utilisée pour les IOT (Internet of Things / objets connectés). MQTT est conçu comme un transport de messagerie de publication / abonnement extrêmement léger en termes de ressources.

Mise en situation :

Vous allez réaliser un système à base de NUCLEO Ethernet permettant de mesurer la température, la pression et l'humidité d'une pièce. Vous voulez connaître ces informations 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'une carte Nucleo ethernet (publish). Enfin la réception des données avec Processing et une tablette (Subscribe).

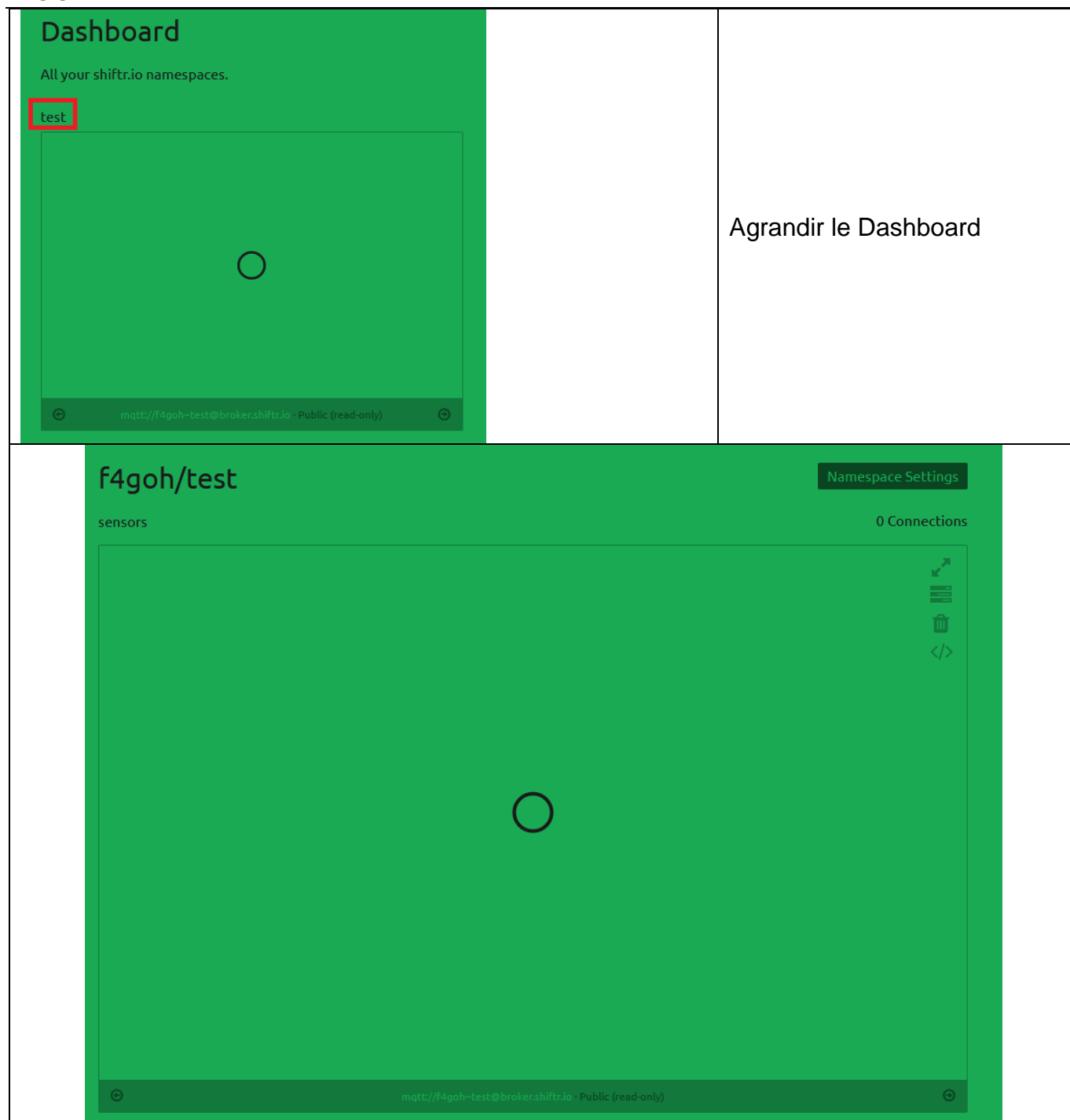
2 Configuration d'un broker (serveur MQTT) en ligne

	<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)</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> <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)

Rem : Le capteur bme 280 n'est pas utilisé dans ce document, mais dans l'utilisation avec un Arduino. Le compte shiftr.io est identique.



La configuration du Broker (Serveur MQTT) est terminée.

Fichiers NUCLEO et Processing ICI

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

Bilan de la configuration

NUCLEO ethernet

Adresse du serveur	broker.shiftr.io
Client id	NUCLEO
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

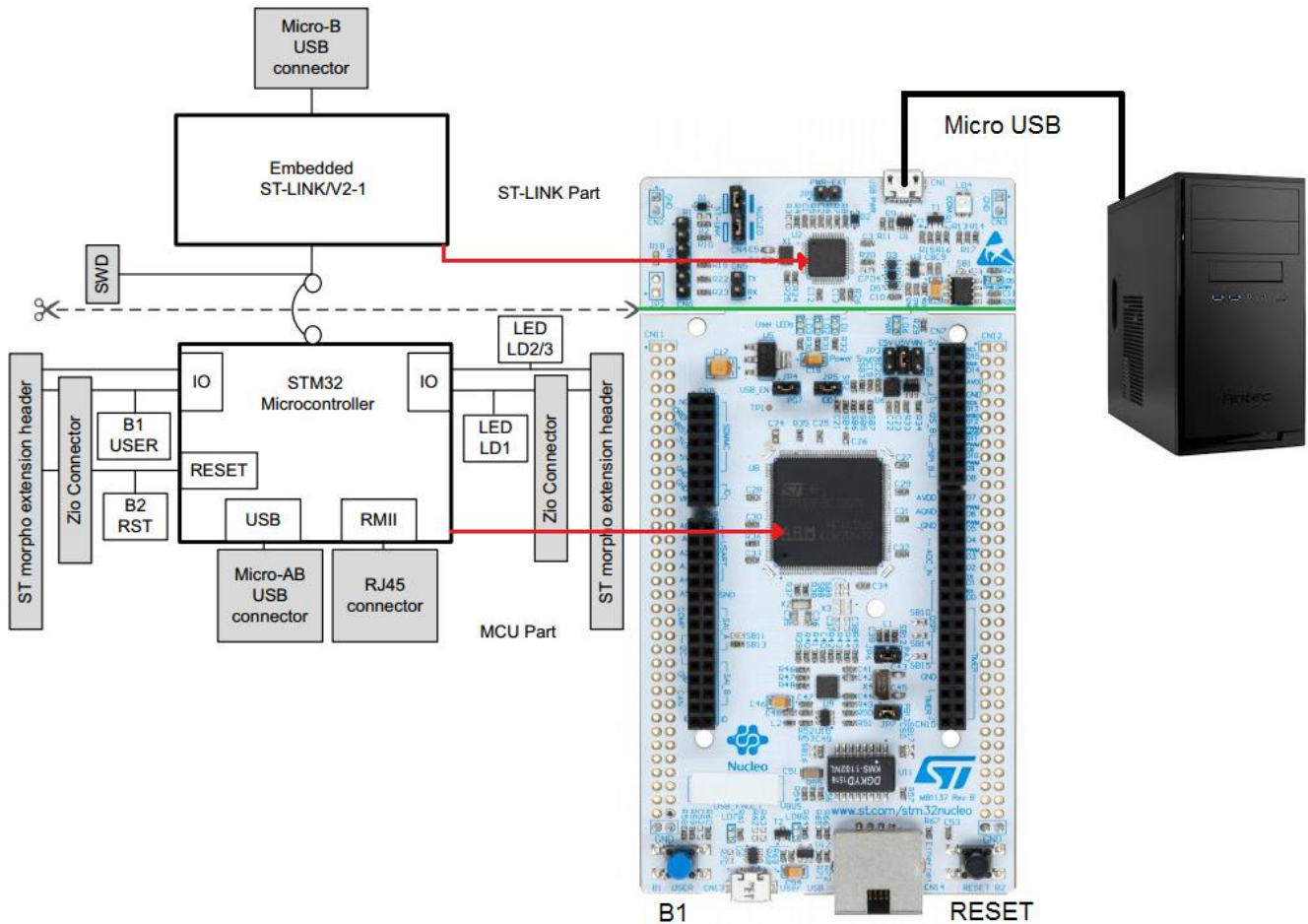
Port 1881 : communication non sécurisée (données en clair sur le réseau) par défaut dans ce document.

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

En rouge : obligatoire en lien avec la configuration du broker

3 Programmation de la NUCLEO

Bien démarrer avec la carte NUCLEO F429ZI



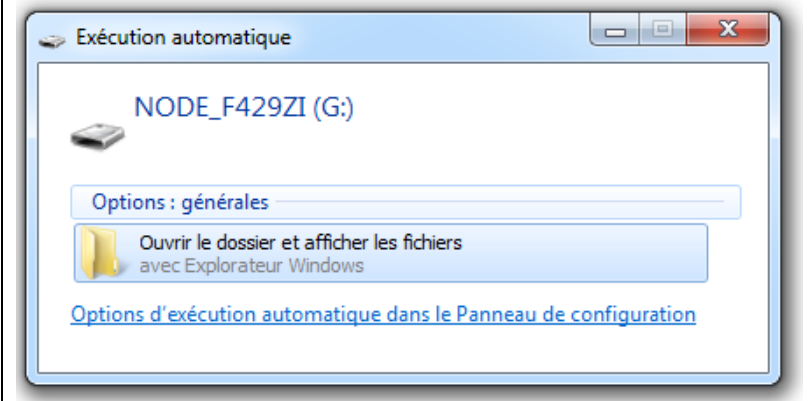
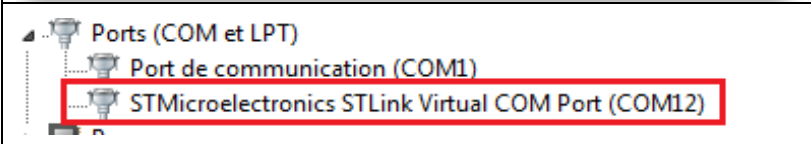
La carte NUCLEO F429ZI comporte une partie sécable. C'est une interface de programmation appelée ST-LINK. Elle est comprise sur toutes les cartes de la gamme NUCLEO, il est donc inutile d'utiliser un programmeur séparé.

Remarque concernant les drivers des cartes nucléo

Quand on connecte une carte NUCLEO sur un PC, deux pilotes sont nécessaires

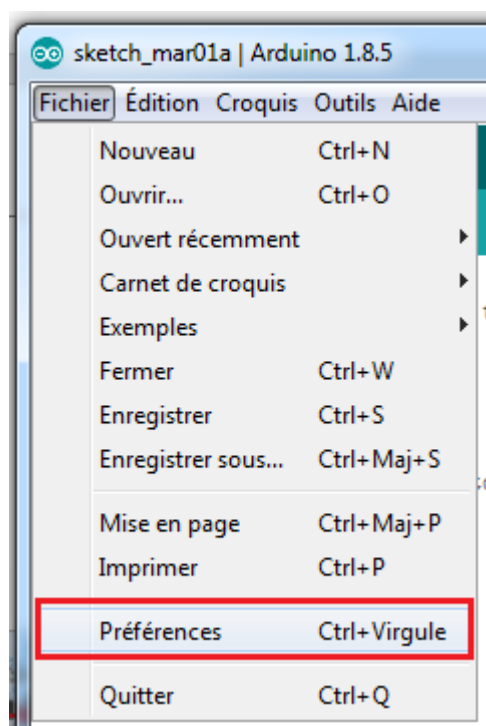
L'un est installé automatiquement, c'est un USB mass Storage Device comme pour une clé USB.

L'autre driver correspond au pilote ST-LINK. Dans ce cas il faudra installer le pilote manuellement.

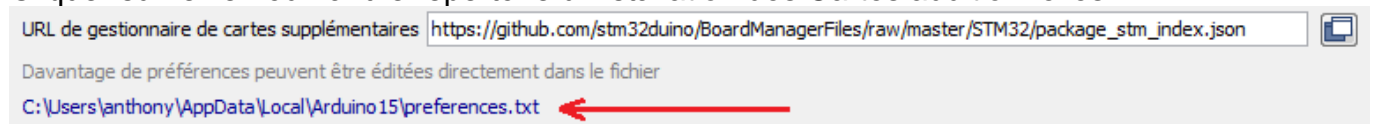
	<p>USB mass Storage Device.</p> <p>Installation automatique du pilote</p>
	<p>Installation manuelle du driver à partir du Gestionnaire de périphériques.</p>

Où se trouve le pilote ST-LINK ?

Dans l'IDE Arduino, aller dans le menu « Préférences »



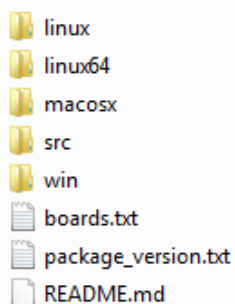
Cliquer sur le lien ouvrant le répertoire d'installation des Cartes additionnelles



Continuer de parcourir les répertoires afin d'accéder aux outils

C:\Users\name\AppData\Local\Arduino15\packages\STM32\tools\STM32Tools\2017.9.22\tools

A ce niveau, un choix s'impose en fonction de votre système d'exploitation



Pour Windows:

C:\Users\name\AppData\Local\Arduino15\packages\STM32\tools\STM32Tools\2017.9.22\tools\win\stlink\ST-LINK_USB_V2_Driver

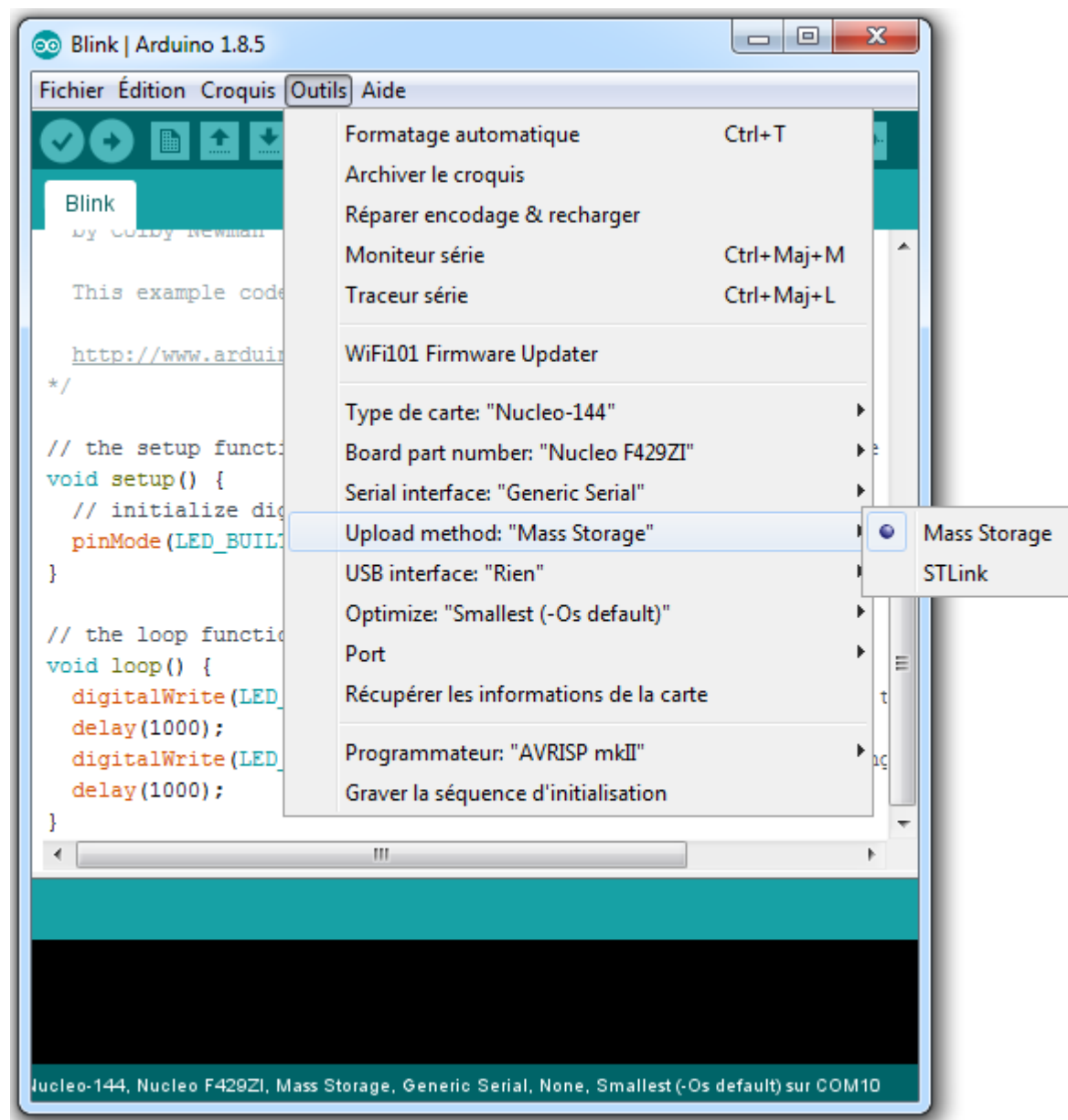
	<p>Dans le Gestionnaire de périphériques, effectuer une installation manuelle en pointant ce répertoire.</p>
--	--

Pour Linux:

/home/name/.arduino15/packages/STM32/tools/STM32Tools/2017.9.22/tools/linux64

	<p>Exécuter le script install.sh en administrateur sudo bash install.sh</p>
--	---

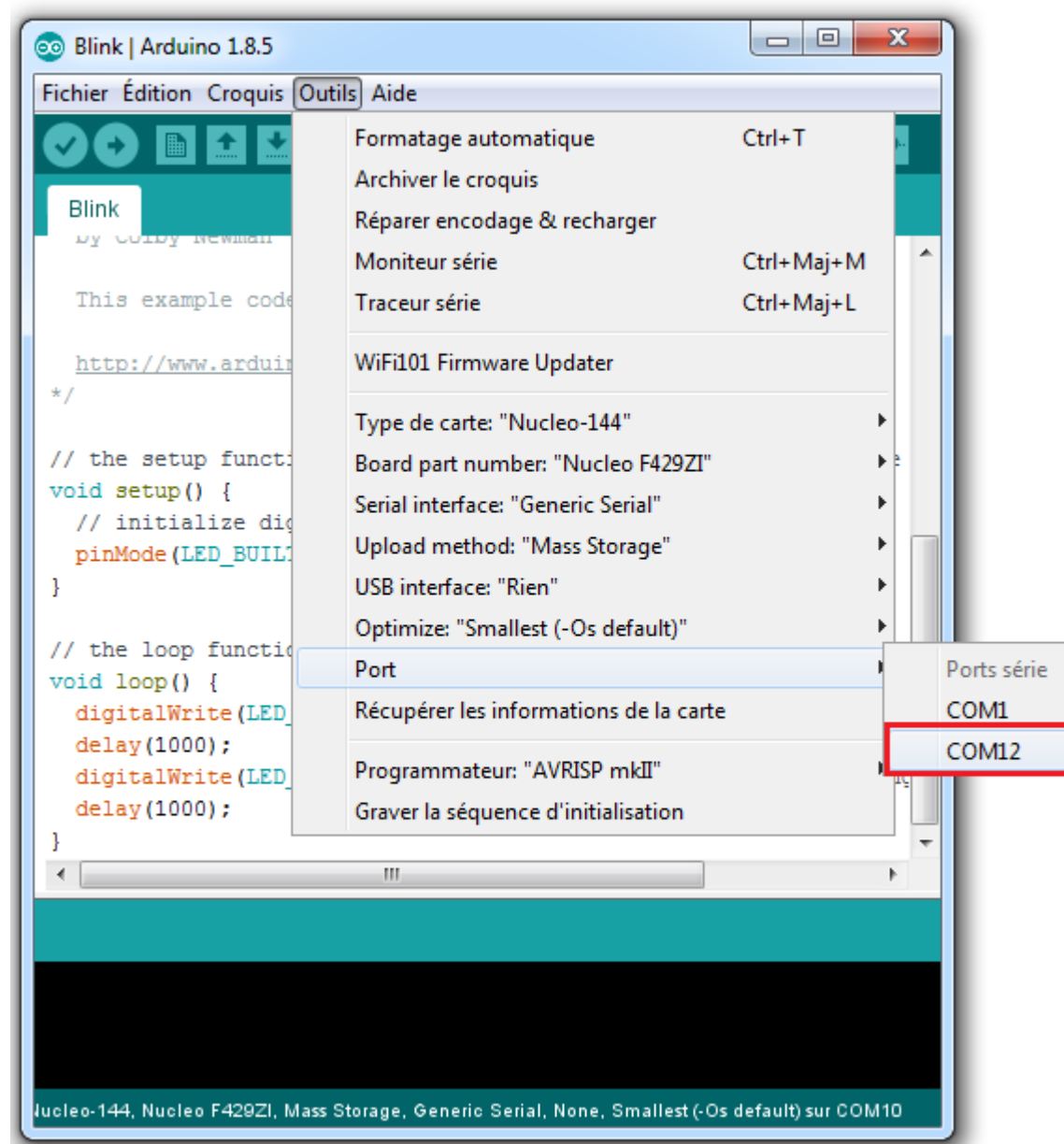
Dans l'environnement Arduino, configurer le menu Outil de la manière suivante :



On retrouve les deux modes de programmation Mass Storage ou STLink

Choisir par défaut **Mass Storage**

Vérifier qu'un port de communication est bien installé. (Autre que COM1)



Sous linux le port de communication se nomme **ttyACM0**

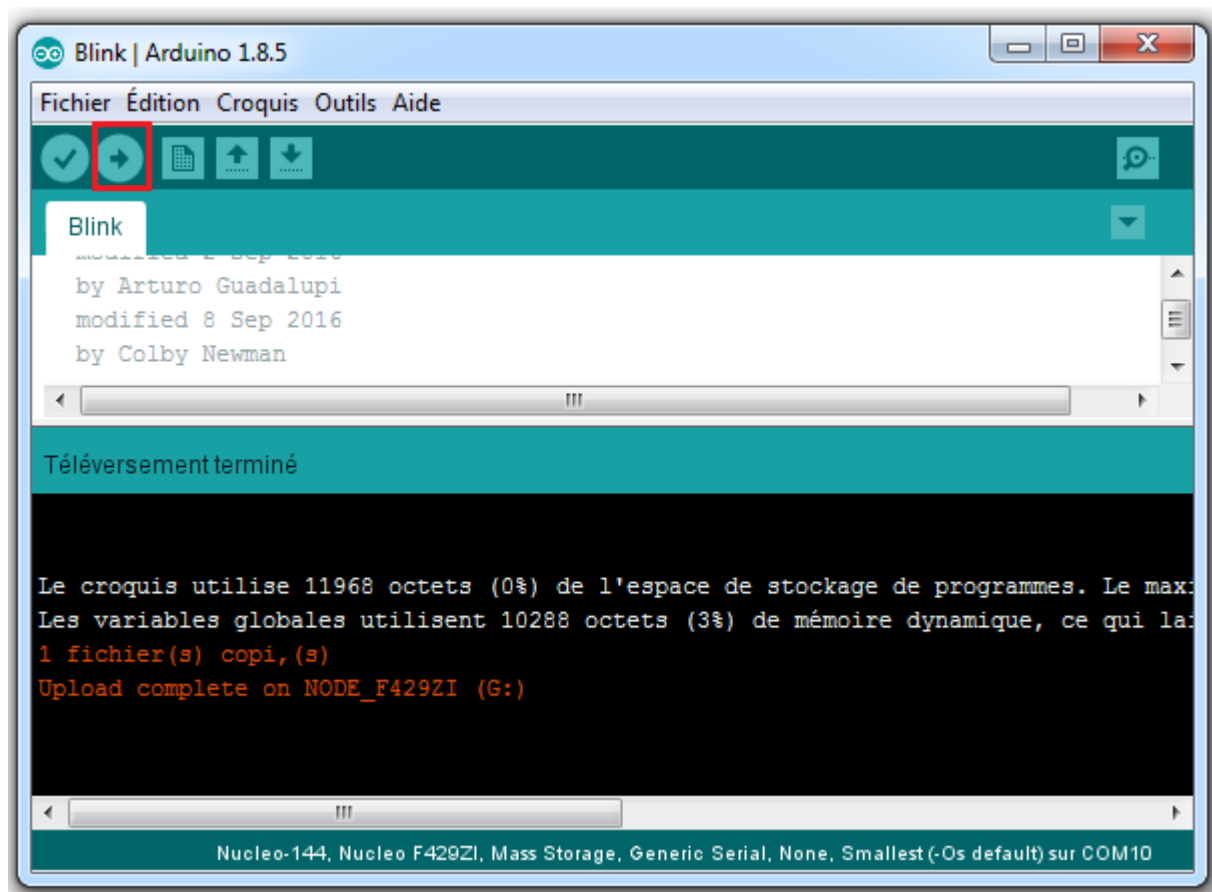
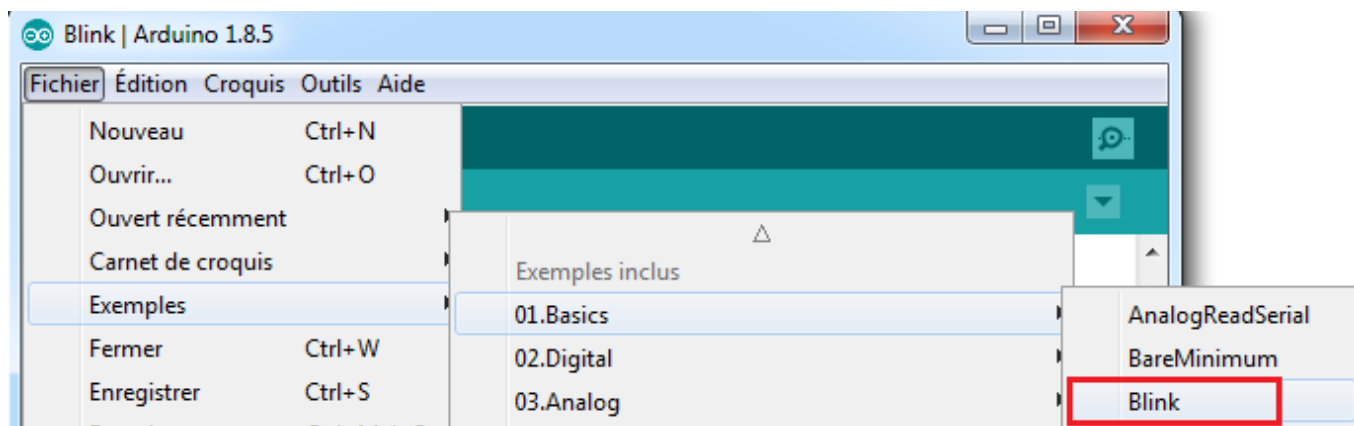
Dans la console taper :

```
ls /dev/ttyA*
```

/dev/ttyACM0 devrait apparaitre.

Gestion des Leds et du bouton poussoir

Charger le programme « Blink » à partir des exemples

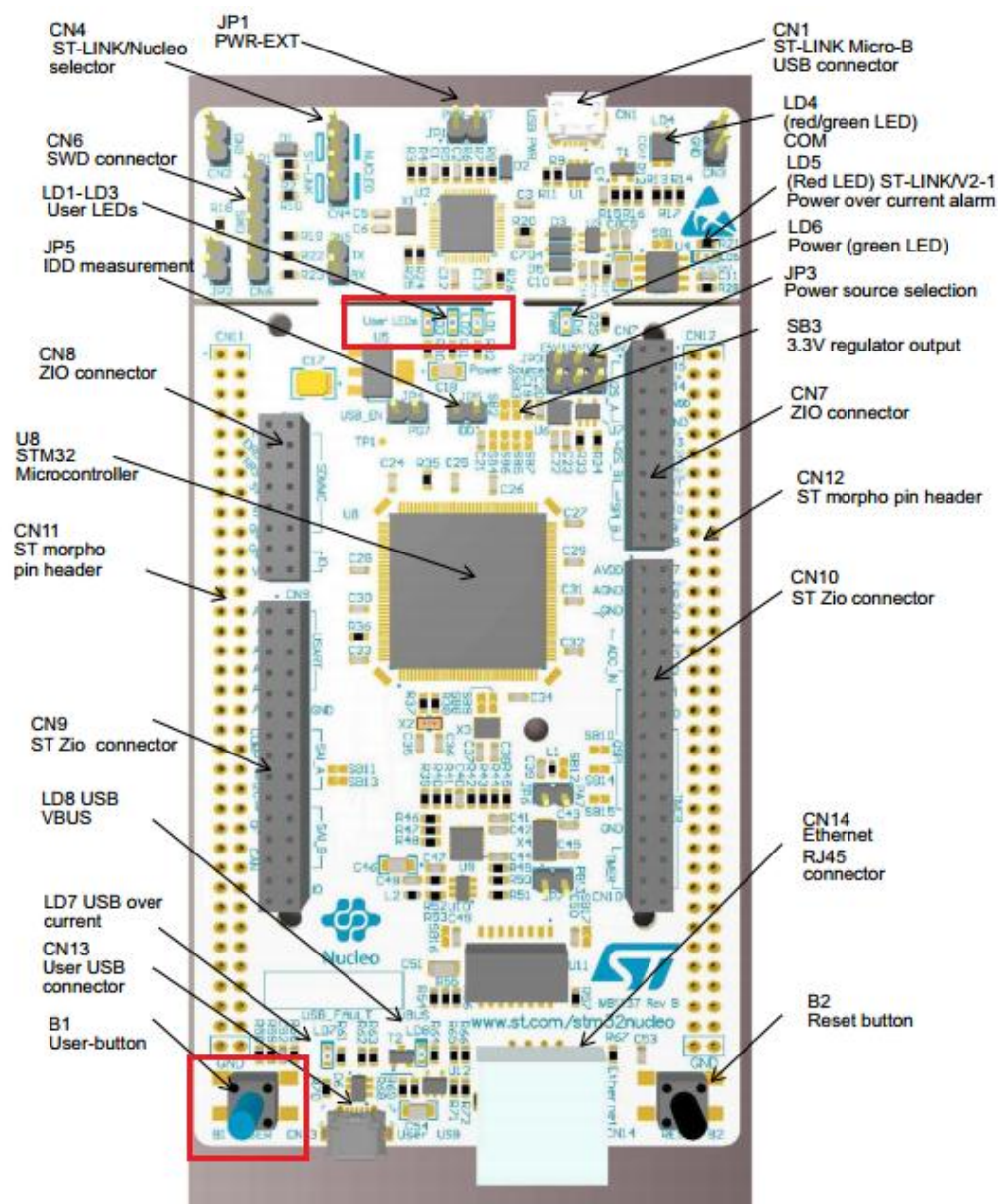


La Led LD1 doit clignoter toutes les secondes.

Remplacer la constante `LED_BUILTIN` par `PB0`.

Une fois le programme téléversé. Il ne doit pas y avoir de changement.

Repérer les Leds LD1, LD2, LD3, ainsi que le bouton poussoir User sur la carte



Pxx : Port d'entrées sorties de la carte STM32 NUCLEO 144, à partir du document en.DM00244518.pdf, pages 30-31 ou en annexe 1

http://www.st.com/resource/en/user_manual/dm00244518.pdf

Niveau Actif : bas ou haut

	LD1	LD2	LD3	BP USER
Pxx	PB0	PB7	PB14	PC13
Niveau Actif	Haut	Haut	Haut	Haut
Couleur de la led	vert	bleu	rouge	

Les 3 Leds et le bouton poussoir sont décrit ici dessous dans le schéma structural de la carte NUCLEO

http://www.st.com/resource/en/schematic_pack/nucleo_144pins_sch.zip

Exemple de programme suivant afin de faire défiler les 3 Leds

<pre> /* scrolling leds */ #define LD1 PB0 #define LD2 PB7 #define LD3 PB14 void setup() { pinMode(LD1, OUTPUT); pinMode(LD2, OUTPUT); pinMode(LD3, OUTPUT); } </pre>	<pre> void loop() { digitalWrite(LD1, HIGH); delay(100); digitalWrite(LD1, LOW); digitalWrite(LD2, HIGH); delay(100); digitalWrite(LD2, LOW); digitalWrite(LD3, HIGH); delay(100); digitalWrite(LD3, LOW); } </pre>
---	---

3 Utilisation avec une carte NUCLEO Ethernet

Afin de mesurer la pression, température et humidité on utilisera la carte shield additionnelle **x-nucleo-iks01a2**

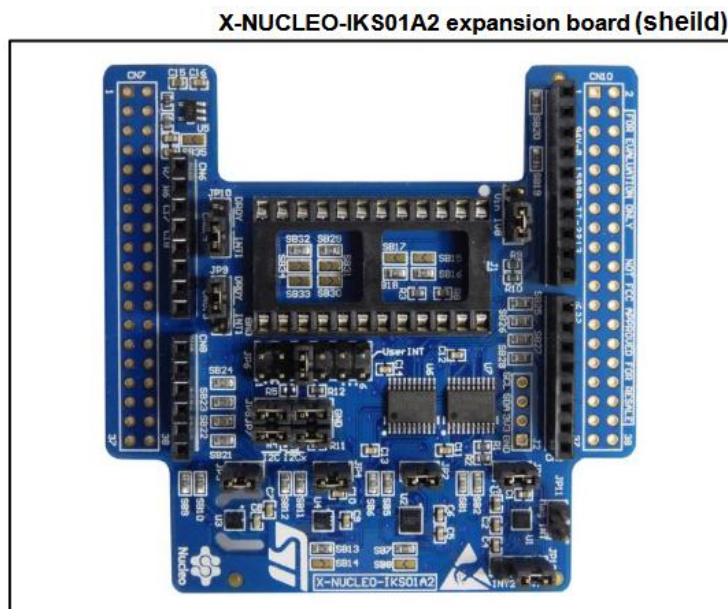


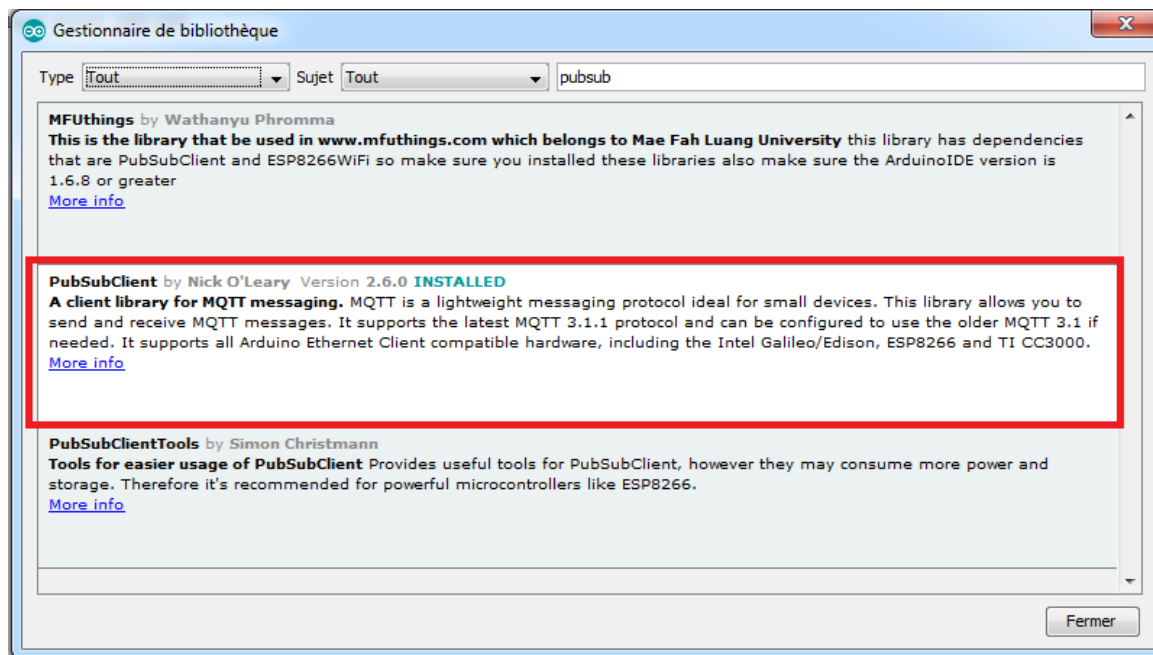
Schéma structurel de la carte capteur MEMS X-NUCLEO-IKS01A2

http://www.st.com/resource/en/user_manual/dm00333132.pdf

- LSM6DSL 3D accelerometer and 3D gyroscope,
- LSM303AGR 3D accelerometer and 3D magnetometer,
- HTS221 humidity and temperature sensor,
- LPS22HB pressure sensor.

On utilisera les capteurs HTS221 et LPS22HB.
Cependant, il est possible d'utiliser le capteur bme280

Installer les librairies suivantes à partir de l'IDE Arduino



STM32duino HTS221 by AST, Wi6Labs Version 1.0.2 **INSTALLED**

Capacitive digital sensor for relative humidity and temperature. This library provides Arduino support for the capacitive digital sensor for relative humidity and temperature HTS221 for STM32 boards.

[More info](#)

Sélectionner une vers...

Installer

Mise à jour

STM32duino LPS22HB by AST, Wi6Labs Version 1.0.2 **INSTALLED**

260-1260 hPa absolute digital output barometer. This library provides Arduino support for the 260-1260 hPa absolute digital output barometer LPS22HB for STM32 boards.

[More info](#)

STM32duino STM32Ethernet by Various Version 1.0.3 **INSTALLED**

Enables network connection (local and Internet) using the STM32 Board. With this library you can use the STM32 board to connect to Internet. The library provides both Client and server functionalities. The library permits you to connect to a local network also with DHCP and to resolve DNS. This library depends on the LwIP library.

[More info](#)

Sélectionner une vers...

Installer

STM32duino LwIP by Adam Dunkels Version 2.0.3 **INSTALLED**

A Lightweight TCP/IP stack lwIP is a small independent implementation of the TCP/IP protocol suite that has been developed by Adam Dunkels at the Computer and Networks Architectures (CNA) lab at the Swedish Institute of Computer Science (SICS). The focus of the lwIP TCP/IP implementation is to reduce the RAM usage while still having a full scale TCP. This making lwIP suitable for use in embedded systems with tens of kilobytes of free RAM and room for around 40 kilobytes of code ROM.

[More info](#)

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

```
/* mqttCapteursNucleo.ino - very simple mqtt example */
```

```
#include <LwIP.h>
#include <STM32Ethernet.h>
#include <LPS22HBSensor.h>
#include <PubSubClient.h>
#include <HTS221Sensor.h>
```

```
LPS22HBSensor *PressTemp;
HTS221Sensor *HumTemp;
```

```
/* CHANGE THIS TO YOUR OWN UNIQUE VALUE. The MAC
number should be
different from any other devices on your network or you'll have
problems receiving packets. */
static uint8_t mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
```

```
/* CHANGE THIS TO MATCH YOUR HOST NETWORK. Most
home networks are in
the 192.168.0.XXX or 192.168.1.XXX subrange. Pick an address
that's not in use and isn't going to be automatically allocated by
DHCP from your router. */
```

```
//IPAddress ip(192, 168, 1, 177);
const char * server = "broker.shiftr.io";
EthernetClient ethClient;
```

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

```
void setup()
{
  Serial.begin(57600);
  Wire.begin();
  PressTemp = new LPS22HBSensor (&Wire);
  PressTemp->Enable();
  HumTemp = new HTS221Sensor (&Wire);
  HumTemp->Enable();
```

```
/* initialize the Ethernet adapter */
Ethernet.begin(mac);
//Ethernet.begin(mac, ip);
```

```
/* setup our default command that will be run when the user
accesses
the root page on the server */
mqtt.setServer(server, 1883); //adresse et port du serveur
```

```
mqtt.setCallback(callback); //au cas ou la NUCLEO devient
subscriber
```

```
Serial.print("Nucleo client is at ");
Serial.println(Ethernet.localIP());
}
```

```
void loop()
{
  if (mqtt.connected()) { //si connection au serveur publish
    pubCapteur();
    long timeN = millis();
    while (millis() - timeN < 10000) {
      mqtt.loop();
    }
    //delay(10000);
  }
  else {
    reconnect(); //sinon tentative de reconnection
  }
}
```

```
void pubCapteur()
{
  float pressure, temperature, humidity;
  PressTemp->GetPressure(&pressure);
  PressTemp->GetTemperature(&temperature);
  HumTemp->GetHumidity(&humidity);
  Serial.print("pression :");
  Serial.print(pressure);
  Serial.print(" / temp :");
  Serial.print(temperature);
  Serial.print(" / humidity :");
  Serial.println(humidity);
```

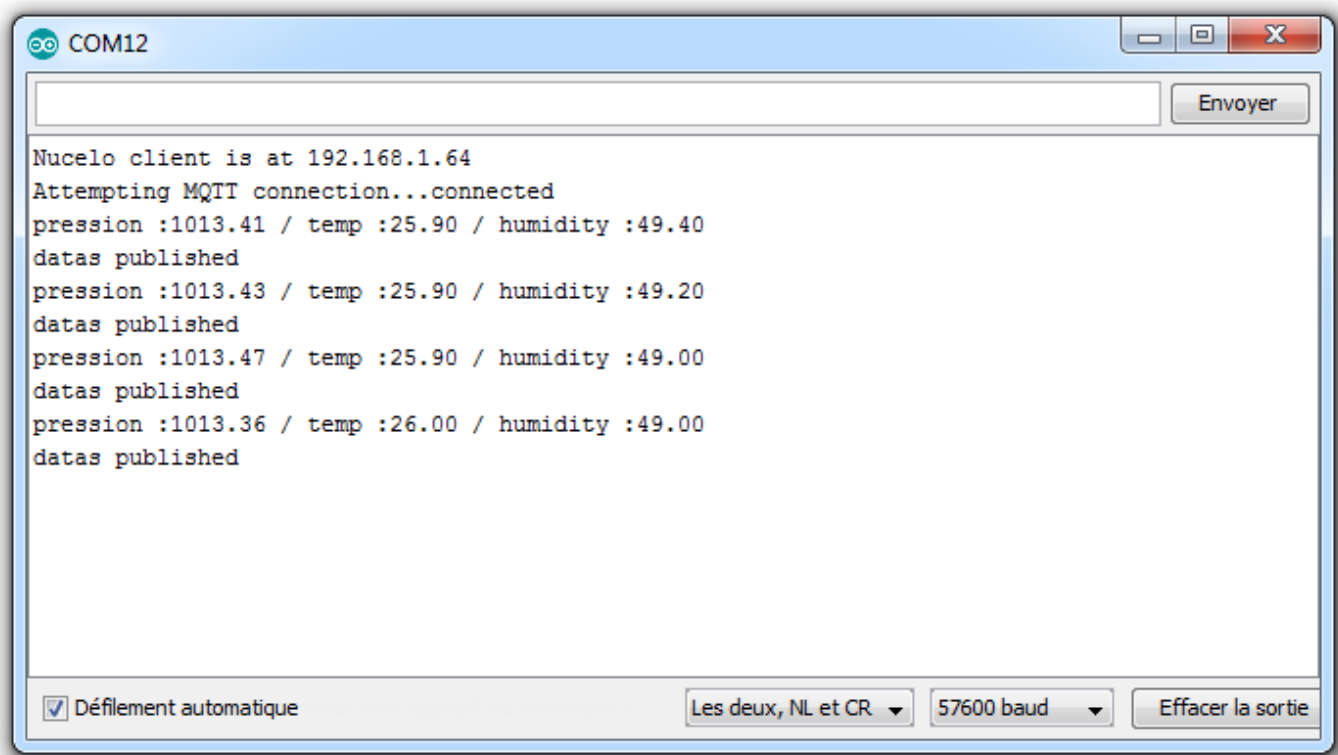
```
char buffer[10];
dtostrf(temperature, 2, 2, buffer); //conversion en chaine de
caractères
mqtt.publish("/sensors/temperature", buffer); //envoi de la
donnée dans le bon topic
dtostrf(pressure, 4, 2, buffer);
mqtt.publish("/sensors/pression", buffer);
dtostrf(humidity, 2, 2, buffer);
mqtt.publish("/sensors/humidite", buffer);
Serial.println("datas published");
delay(10000);
}
```

```
void reconnect() {
  // Loop until we're reconnected
  while (!mqtt.connected()) {
    Serial.print("Attempting MQTT connection...");
    // Attempt to connect
    if (mqtt.connect("nucleo", "weatherSensors", "bme280Sensors")) {
      Serial.println("connected");
      // subscribe if necessary
      //mqtt.subscribe("/sensors/temperature");
      // mqtt.unsubscribe("/sensors/temperature");
    } else {
      Serial.print("failed, rc=");
      Serial.print(mqtt.state());
      Serial.println(" try again in 5 seconds");
      // Wait 5 seconds before retrying
      delay(5000);
    }
  }
}
```

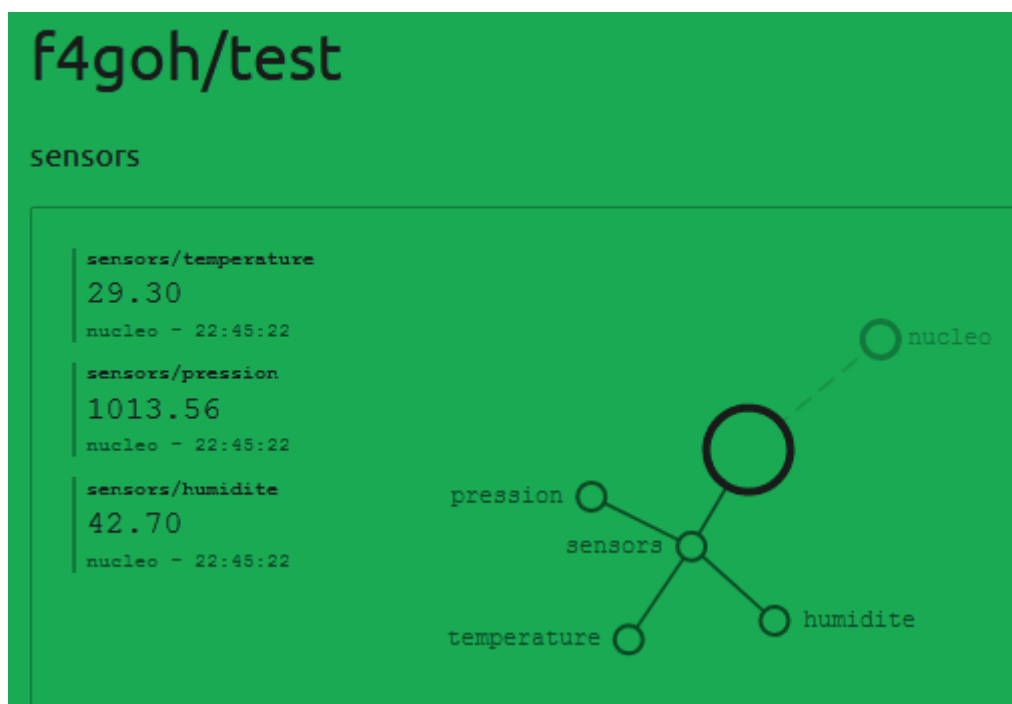
```
void callback(char* topic, byte* payload, unsigned int length) { //si
une donnée arrive en provenance du broker, celle ci est affichée
Serial.print("Message arrived [");
Serial.print(topic);
Serial.print("] ");
for (int i = 0; i < length; i++) {
  Serial.print((char)payload[i]);
}
Serial.println();
}
```

Résultats à l'écran

Coté client (NUCLEO Ethernet)



Coté serveur (dans le Dashboard)



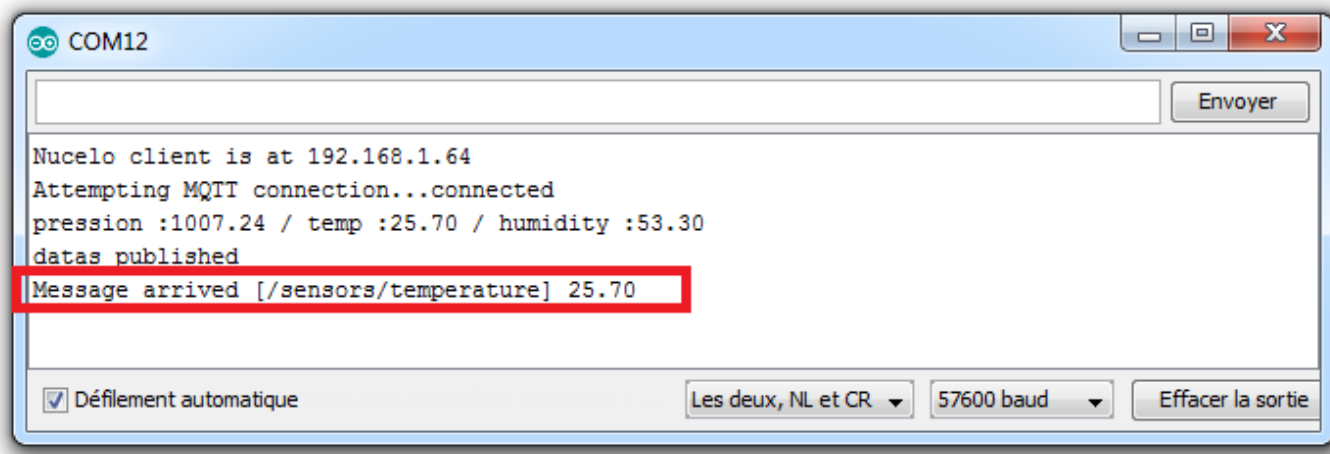
Les valeurs du capteur apparaissent sous la forme d'un arbre en fonction du « Topic » réalisé

NUCELO en Subscriber

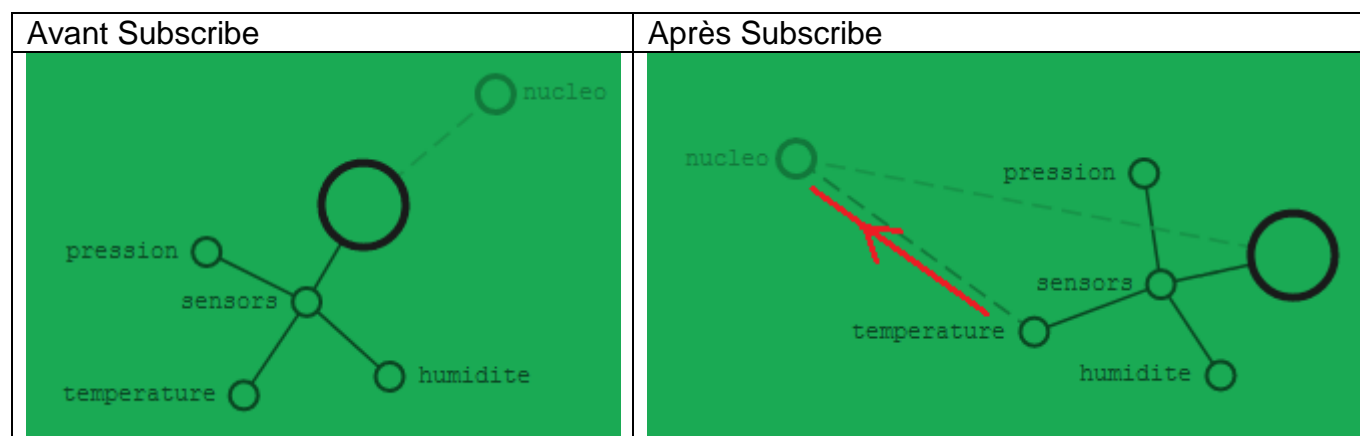
Retirer le commentaire de la ligne

```
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

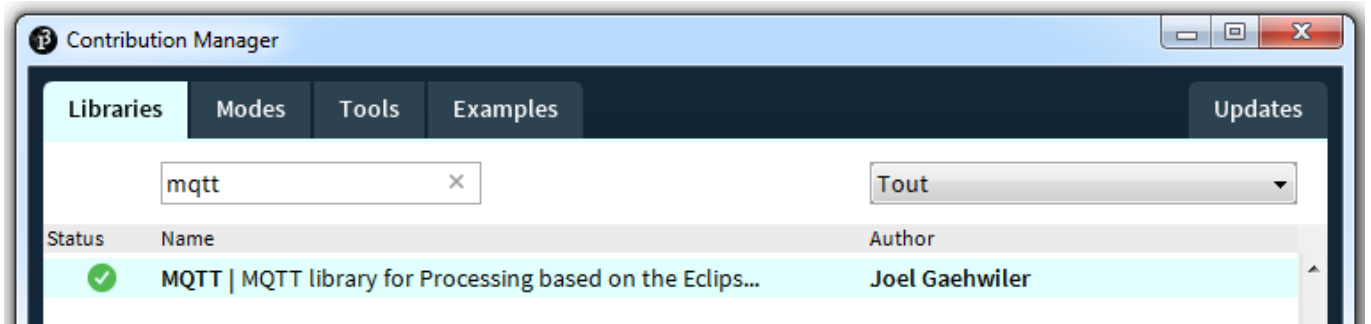
```
mqtt.subscribe("/sensors/humidite");
mqtt.subscribe("/sensors/pression");
```

La carte NUCLEO peut envoyer des informations et en recevoir sans passer par une page WEB

5 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() {}
```

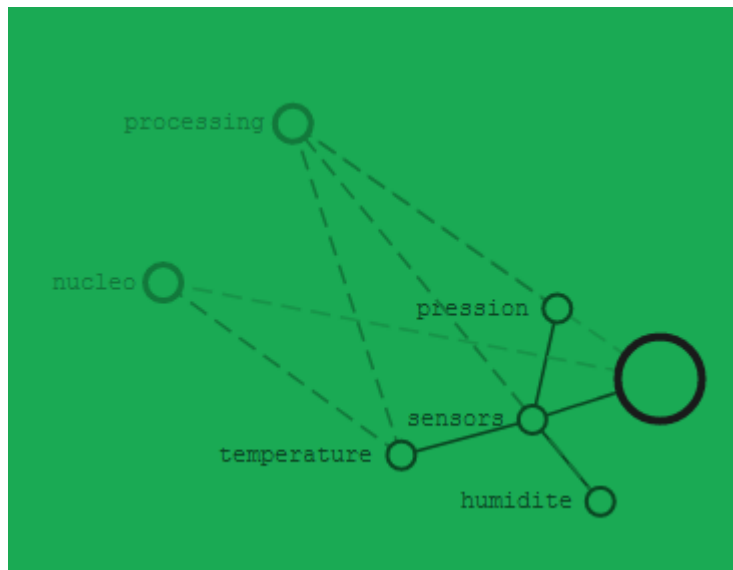
```
void keyPressed() {
  // client.publish("/sensors/temperature", "20");
}
```

```
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

6 Utilisation avec une tablette Android MQTT Dash

Installer le programme MQTT dash



MQTT Dash (IoT, Smart Home)

Routix software Communication

★★★★★ 1 911

3 PEGI 3

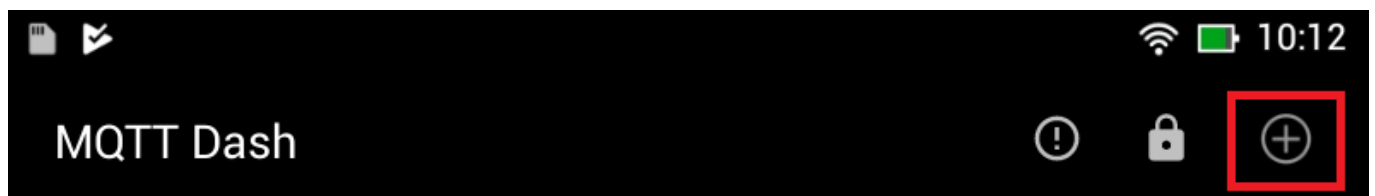
Cette application est compatible avec vos appareils.

Ajouter à la liste de souhaits

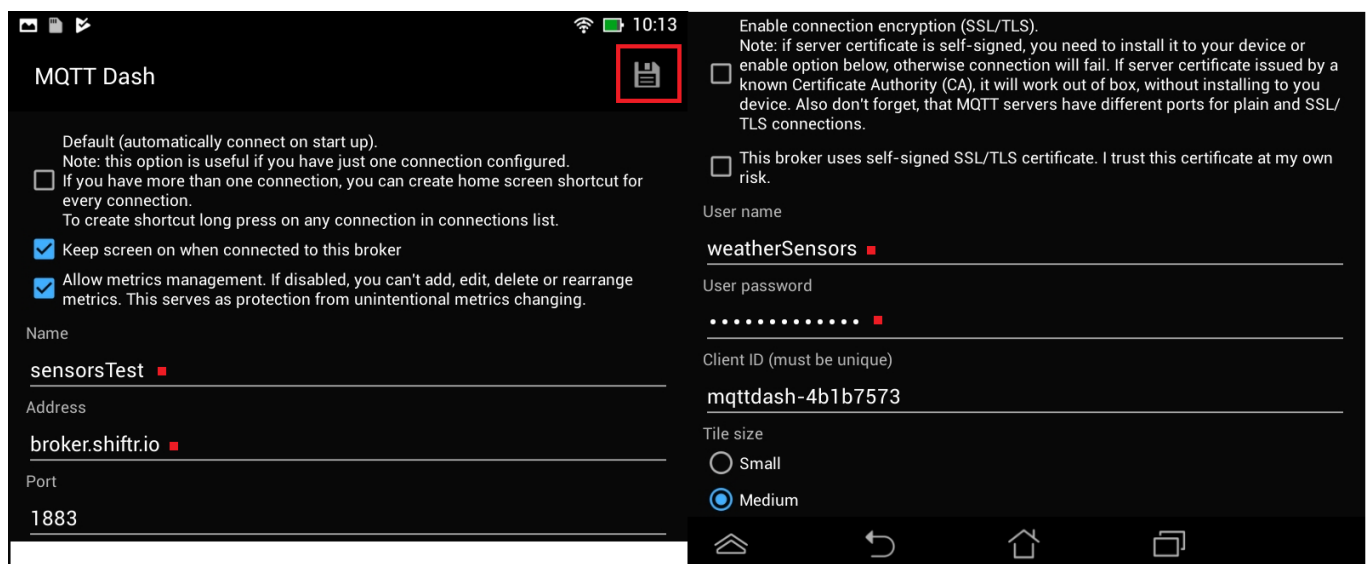
Installer

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

Configuration :



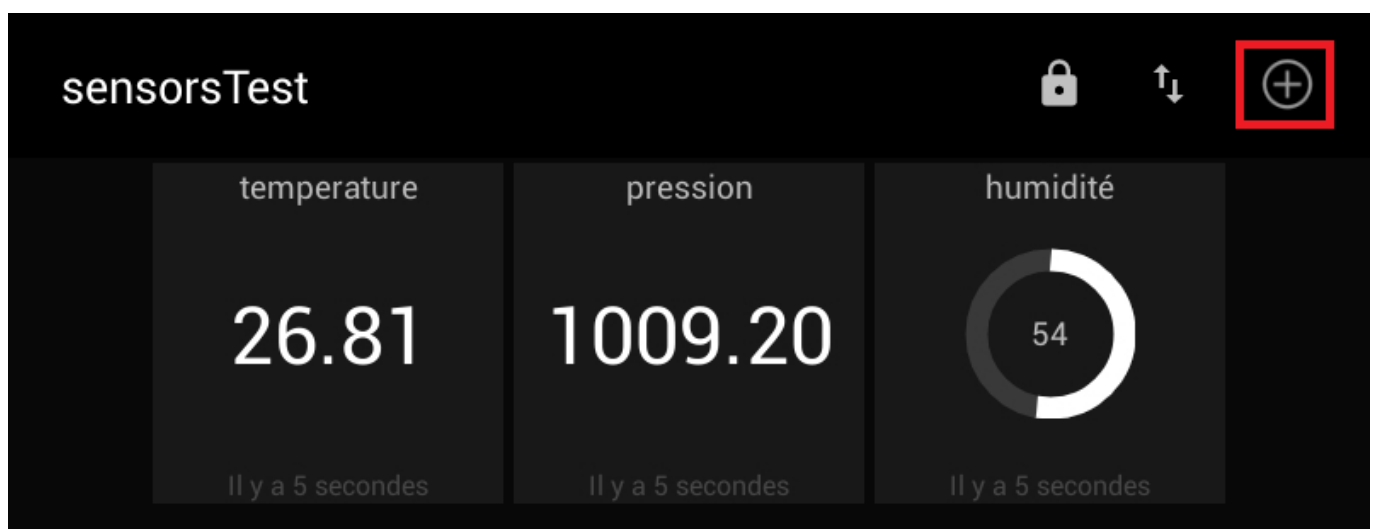
Ajouter la connexion au broker



Cliquer sur sensorsTest



Puis ajouter les 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)

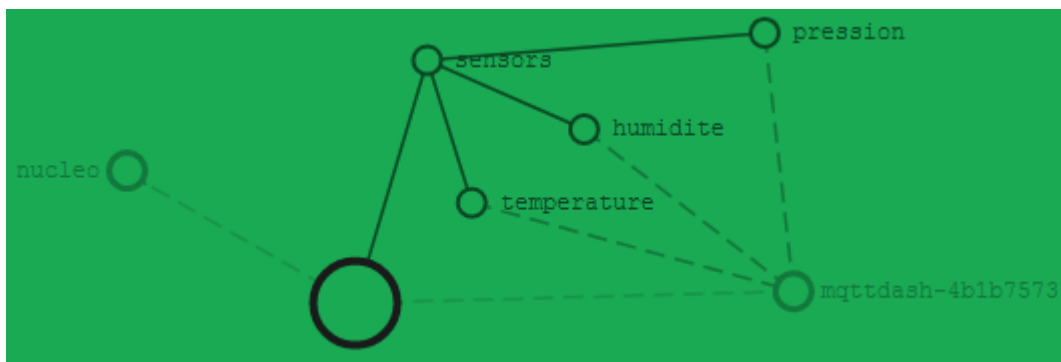
/sensors/temperature

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

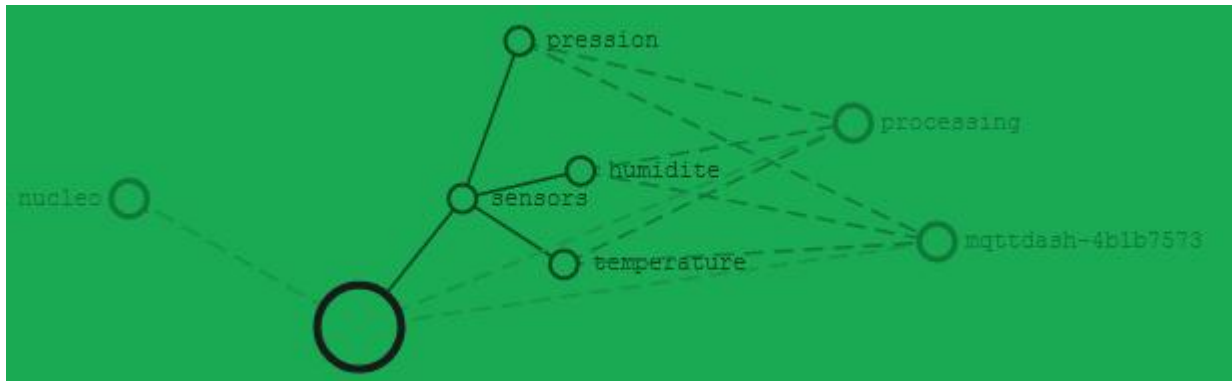
Prefix Postfix

Le Dashboard a changé de forme à nouveau



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

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

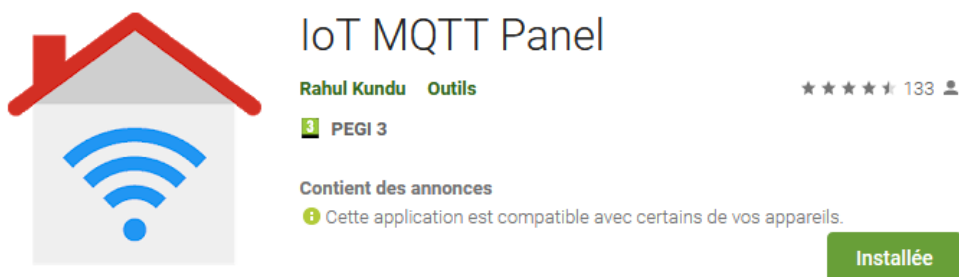


Informations complémentaires :

<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
--	--

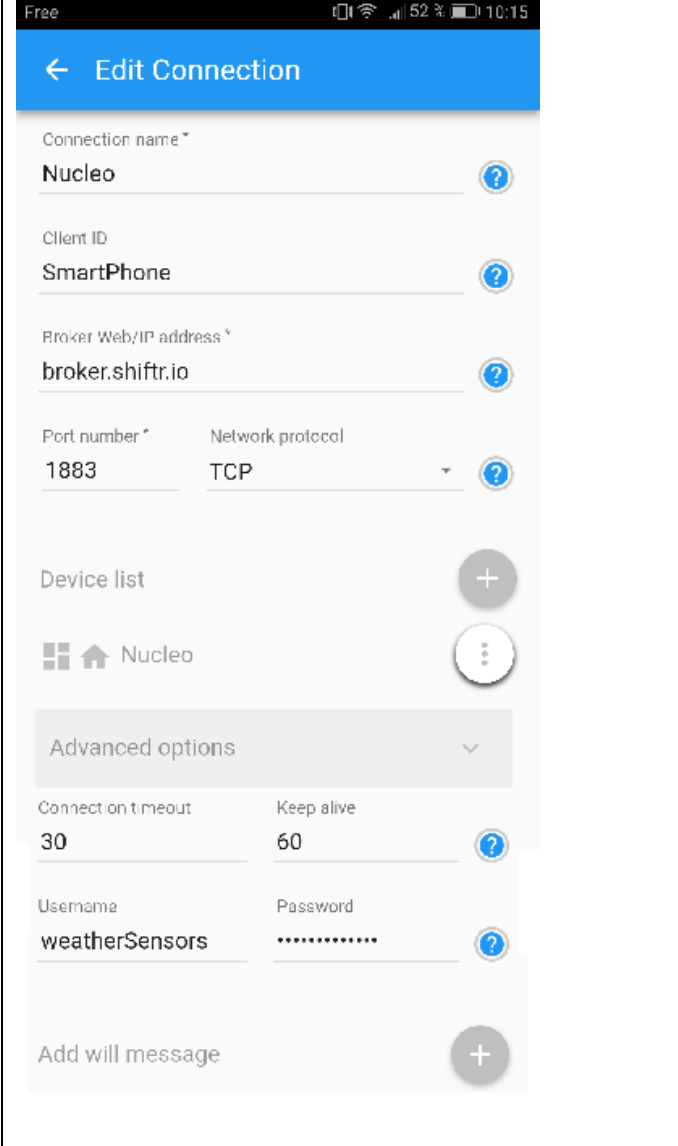
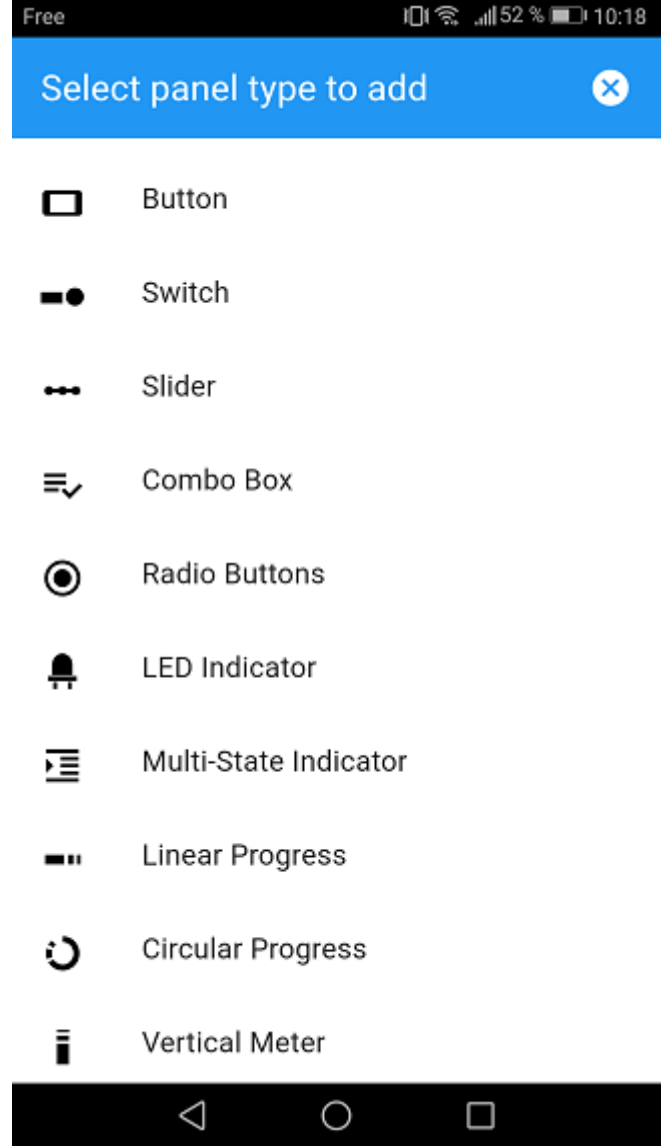
7 Utilisation avec une tablette Android MQTT Panel

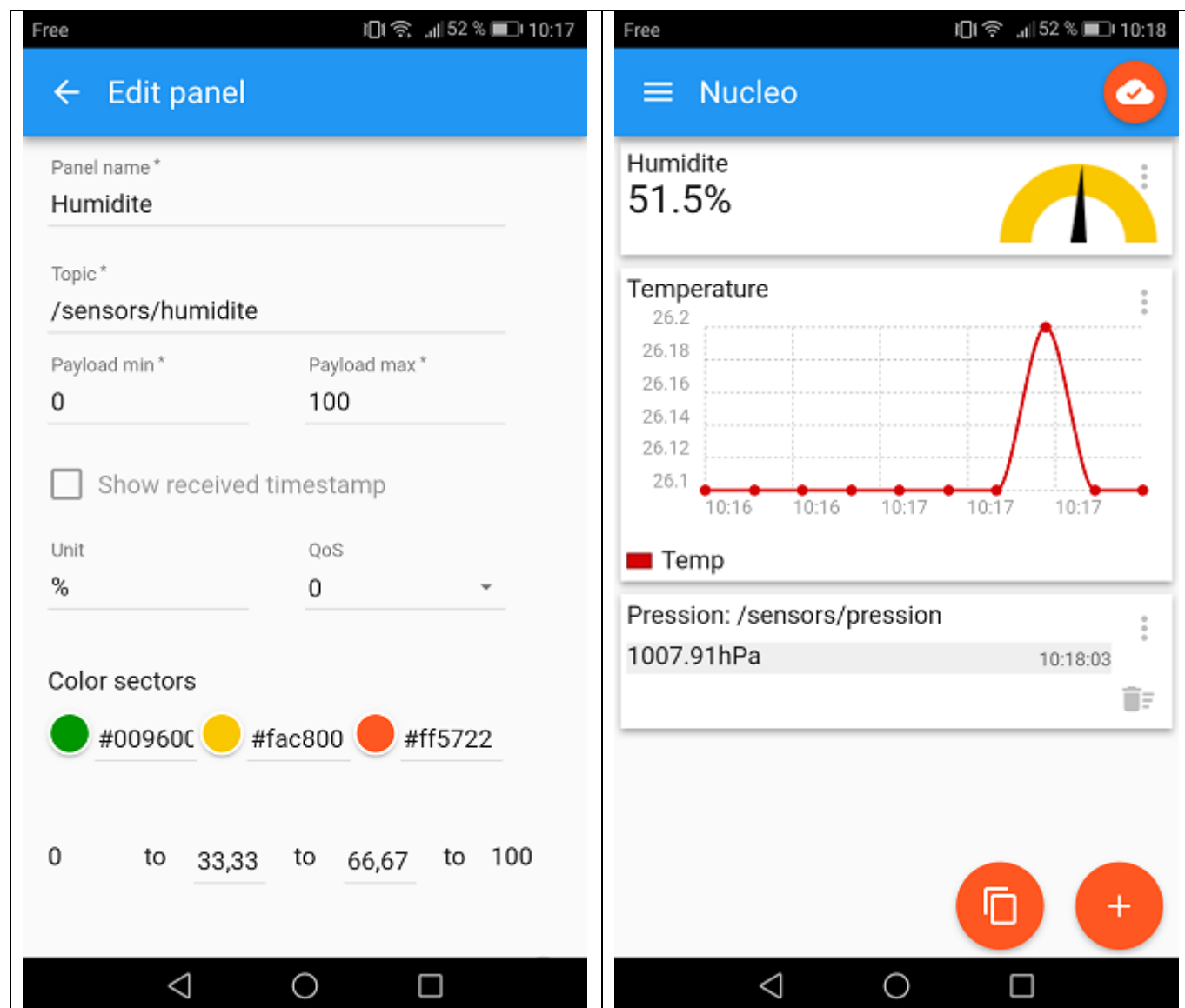
Installer le programme IoT MQTT Panel



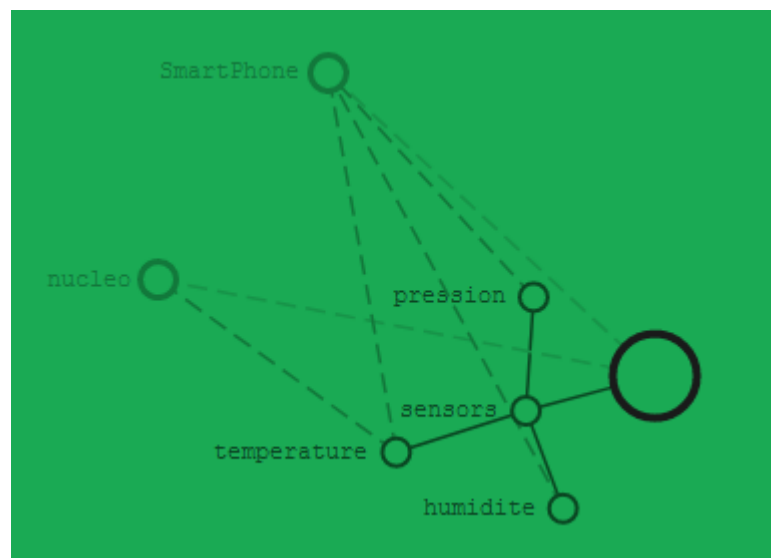
<https://play.google.com/store/apps/details?id=snr.lab.iotmqttpanel.prod>

Ce programme est 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



Le Dashboard complet avec un Publisher (Nucleo) et un Subscriber (SmartPhone)



8 Analyse du protocole MQTT



<http://mqtt.org/documentation>

Connect Command (client to server)

http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html#_Toc398718028

MQ Telemetry Transport Protocol

Connect Command

▶ 0001 0000 = Header Flags: 0x10 (Connect Command)

Msg Len: 53

Protocol Name: MQTT

Version: 4

▶ 1100 0010 = Connect Flags: 0xc2

Keep Alive: 60

Client ID: processing

User Name: weatherSensors

Password: bme280Sensors

0000	90 4d 4a a3 0e 00 00 25 22 8a 86 a0 08 00 45 00	.MJ....% ".....E.
0010	00 6b 93 a5 40 00 40 06 96 d9 c0 a8 01 15 36 4c	.k..@.@.6L
0020	18 05 b7 ca 07 5b 79 8f ed 67 52 30 38 88 80 18[y. .gR08...
0030	00 e5 4a 0b 00 00 01 01 08 0a d4 85 d3 f7 07 17	...J.....
0040	39 fa 10 35 00 04 4d 51 54 54 04 c2 00 3c 00 0a	9..5..MQ TT...<..
0050	70 72 6f 63 65 73 73 69 6e 67 00 0e 77 65 61 74	processi ng..weat
0060	68 65 72 53 65 6e 73 6f 72 73 00 0d 62 6d 65 32	herSens rs..bme2
0070	38 30 53 65 6e 73 6f 72 73	80Sensor s

Connect Command

0001 0000 = Header Flags: 0x10 (Connect Command)

Msg Len: 53

Protocol Name: MQTT

Version: 4

1100 0010 = Connect Flags: 0xc2

Keep Alive: 60

Client ID: processing

User Name: weatherSensors

Password: bme280Sensors

0000	10 35 00 04 4d 51 54 54 04 c2 00 3c 00 0a 70 72	.5..MQTT...<..pr
0010	6f 63 65 73 73 69 6e 67 00 0e 77 65 61 74 68 65	ocessing..weathe
0020	72 53 65 6e 73 6f 72 73 00 0d 62 6d 65 32 38 30	rSensors..bme280
0030	53 65 6e 73 6f 72 73	Sensors

Connect Ack (server to client)

http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html#_Toc398718033

MQ Telemetry Transport Protocol

▼ Connect Ack

▶ 0010 0000 = Header Flags: 0x20 (Connect Ack)

Msg Len: 2

.... 0000 0000 = Connection Ack: Connection Accepted (0)

0000	00	25	22	8a	86	a0	90	4d	4a	a3	0e	00	08	00	45	00	..%"....M J.....E.
0010	00	38	02	8f	40	00	ea	06	7e	22	36	4c	18	05	c0	a8	.8..@... ~"6L....
0020	01	15	07	5b	b7	ca	52	30	38	88	79	8f	ed	9e	80	18	...[..R0 8.y.....
0030	00	72	ab	7f	00	00	01	01	08	0a	07	17	3a	0e	d4	85	.r..... :.....
0040	d3	f7	20	02	00	00										

Connect Ack

0010 0000 = Header Flags: 0x20 (Connect Ack)

Msg Len: 2

.... 0000 0000 = Connection Ack: Connection Accepted (0)

0000 20 02 00 00

Subscribe Request (client to server)

http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html#_Toc398718063

MQ Telemetry Transport Protocol

Connect Ack

0010 0000 = Header Flags: 0x20 (Connect Ack)

Msg Len: 2

.... 0000 0000 = Connection Ack: Connection Accepted (0)

0000	00	25	22	8a	86	a0	90	4d	4a	a3	0e	00	08	00	45	00	..%"....M J.....E.
0010	00	38	02	8f	40	00	ea	06	7e	22	36	4c	18	05	c0	a8	.8..@... ~"6L....
0020	01	15	07	5b	b7	ca	52	30	38	88	79	8f	ed	9e	80	18	...[..R0 8.y.....
0030	00	72	ab	7f	00	00	01	01	08	0a	07	17	3a	0e	d4	85	.r..... :.....
0040	d3	f7	20	02	00	00										

Subscribe Request

1000 0010 = Header Flags: 0x82 (Subscribe Request)

Msg Len: 25

Message Identifier: 1

Topic: /sensors/temperature

.... ..00 = Granted Qos: Fire and Forget (0)

0000 82 19 00 01 00 14 2f 73 65 6e 73 6f 72 73 2f 74/sensors/t

0010 65 6d 70 65 72 61 74 75 72 65 00 emperature.

Subscribe Ack (server to client)

http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html#_Toc398718068

Subscribe Ack

1001 0000 = Header Flags: 0x90 (Subscribe Ack)

Msg Len: 3

Message Identifier: 1

.... ..00 = Granted Qos: Fire and Forget (0)

A PUBLISH Control Packet is sent **from a Client to a Server** or **from Server to a Client** to transport an Application Message.

Publish Message (server to client)

http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html#_Toc398718037

MQ Telemetry Transport Protocol

Publish Message

0011 0000 = Header Flags: 0x30 (Publish Message)

Msg Len: 27

Topic: /sensors/temperature

Message: 25.80

0000	00	25	22	8a	86	a0	90	4d	4a	a3	0e	00	08	00	45	00	."	M	J	E	.	
0010	00	51	02	93	40	00	ea	06	7e	05	36	4c	18	05	c0	a8	.	Q	..	@	...	~	.6L
0020	01	15	07	5b	b7	ca	52	30	38	9b	79	8f	ed	e9	80	18	...	[..	R0	8	y	
0030	00	72	2e	40	00	00	01	01	08	0a	07	17	3c	ad	d4	85	.	r	@	<		
0040	d4	3e	30	1b	00	14	2f	73	65	6e	73	6f	72	73	2f	74	.	>	0	...	/s	ensors/t		
0050	65	6d	70	65	72	61	74	75	72	65	32	35	2e	38	30		e	mperatu		r	e	25.80		

Publish Message **incoming**

0011 0000 = Header Flags: 0x30 (Publish Message)

Msg Len: 27

Topic: /sensors/temperature

Message: 25.80

```
0000 30 1b 00 14 2f 73 65 6e 73 6f 72 73 2f 74 65 6d 0.../sensors/tem
0010 70 65 72 61 74 75 72 65 32 35 2e 38 30          perature25.80
```

Publish Message (client to server)

http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html#_Toc398718037

MQ Telemetry Transport Protocol

Publish Message

0011 0000 = Header Flags: 0x30 (Publish Message)

Msg Len: 27

Topic: /sensors/temperature

Message: 25.80

0000	00	25	22	8a	86	a0	90	4d	4a	a3	0e	00	08	00	45	00	."	M	J	E	.	
0010	00	51	02	93	40	00	ea	06	7e	05	36	4c	18	05	c0	a8	.	Q	..	@	...	~	.6L
0020	01	15	07	5b	b7	ca	52	30	38	9b	79	8f	ed	e9	80	18	...	[..	R0	8	y	
0030	00	72	2e	40	00	00	01	01	08	0a	07	17	3c	ad	d4	85	.	r	@	<		
0040	d4	3e	30	1b	00	14	2f	73	65	6e	73	6f	72	73	2f	74	.	>	0	...	/s	ensors/t		
0050	65	6d	70	65	72	61	74	75	72	65	32	35	2e	38	30		e	mperatu	r	e	25.80			

Publish Message **outcoming**

0011 0000 = Header Flags: 0x30 (Publish Message)

Msg Len: 24

Topic: /sensors/temperature

Message: 20

```
0000 30 18 00 14 2f 73 65 6e 73 6f 72 73 2f 74 65 6d 0.../sensors/tem
0010 70 65 72 61 74 75 72 65 32 30                    perature20
```

Disconnect (client to server)

http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html#_Toc398718090

Disconnect Req

1110 0000 = Header Flags: 0xe0 (Disconnect Req)

Msg Len: 0

0000 e0 00

9 Conclusion

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

Cependant il est possible d'écrire ses propres bibliothèques MQTT

