

Formation IoT & Big Data



TP : Projet IoT

Table des matières :

1. Avant-propos	3
1.1 Objectif du TP :	3
1.2 Infrastructure du TP :	3
2. Capteur LoRaWAN DiY	4
2.1 Composants :	4
2.2 Programmation :	5
2.2.1 Introduction Arduino	5
2.2.2 Détail du microcontrôleur du TP : Arduino Pro Mini	7
2.2.3 Le “Hello World” de l’Arduino	8
2.2.4 Mesure de Température et d’Humidité	10
3. Communication avec TTN	10
4. Le trio Node-RED – InfluxDB – Grafana	12
4.1 MQTT avec Node-RED	12
4.1.1 Introduction MQTT	12
4.1.1.1 L’architecture du protocole MQTT	13
4.1.1.2 Niveaux de Qualité de Service	13
4.1.2 Node-RED	15
4.2 Introduction d’InfluxDB	16
4.3 Créez une base de données InfluxDB :	17
4.4 Construction du Dashboard avec Grafana	18
5. Conclusion	18

1. Avant-propos

Légende :

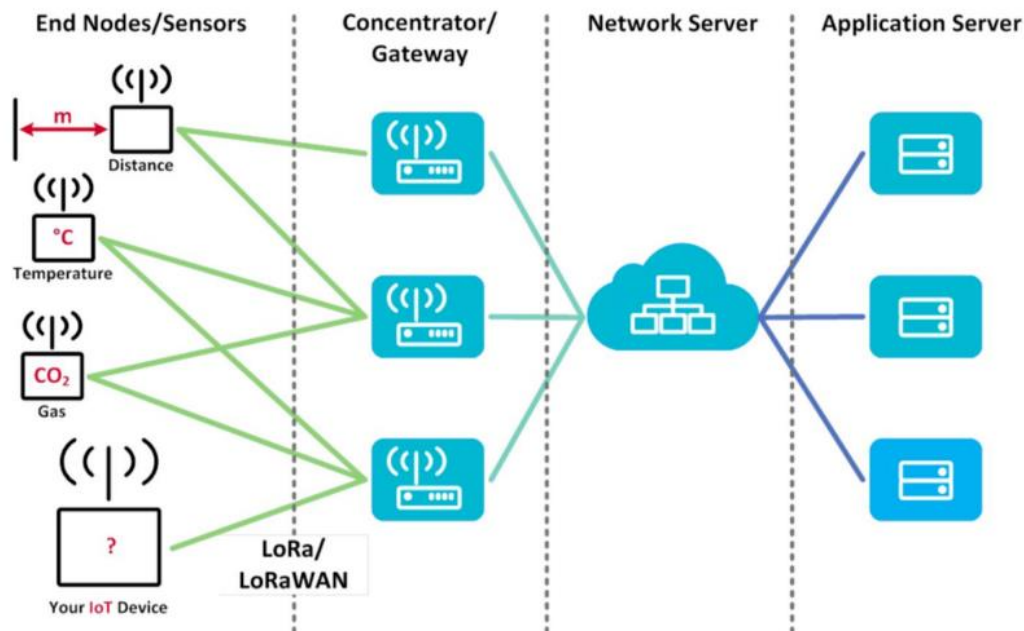
- Exercice à faire
- Question à répondre
- Remarque importante

1.1 Objectif du TP :

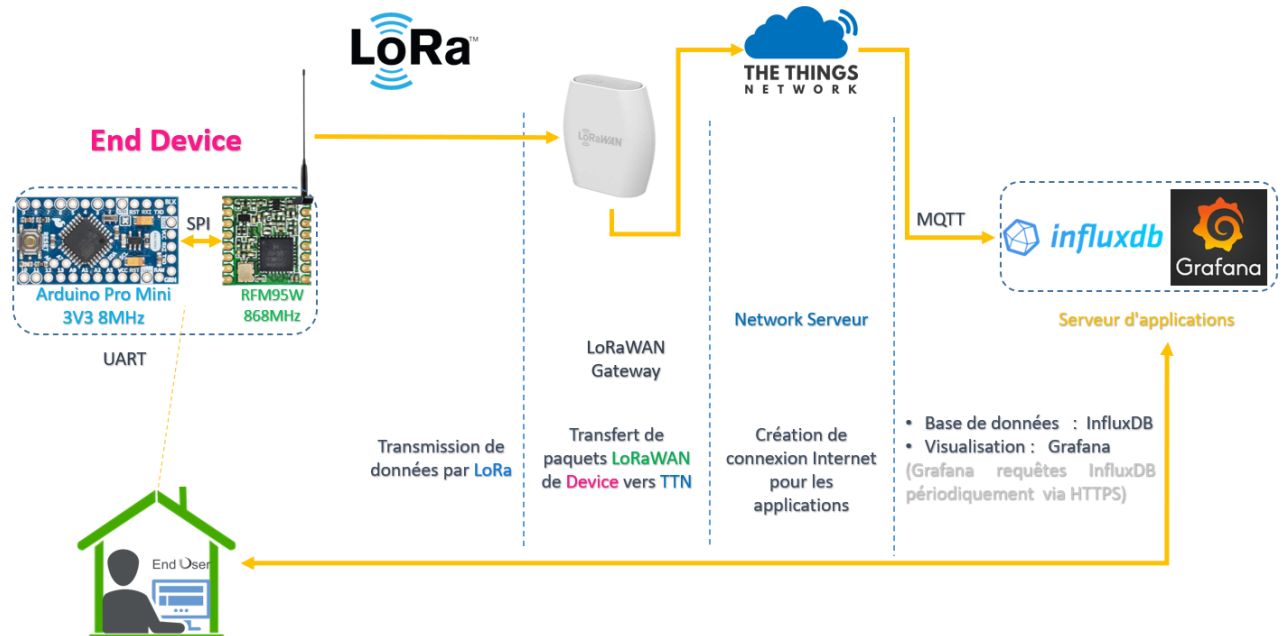
Dans le dernier TP, nous avons travaillé avec des produits du commerce et vous avez pu comprendre et suivre le transit des données de bout en bout. L'objectif de ce nouveau TP est de construire un système IoT simple vous-même (DiY). Pour cela, vous commencez par programmer un capteur de température et d'humidité à l'aide d'une carte électronique prête à l'emploi et afficherez ces données sur un serveur d'application que vous construirez vous-même sur un pc en local à l'aide d'outil open source.

1.2 Infrastructure du TP :

Pour simplifier les manipulations, nous nous concentrerons sur les briques Capteur (End Node) et serveur d'application.



La structure de notre réseaux IoT pour ce TP est donc la suivante :



A noter :

- La gateway « TheThingsIndoor » est déjà connectée sur TTN et est prête à router les trames de votre capteur une fois qu'il sera provisionné.
- Github du TP : https://github.com/bao-optimiz/Projet_IoT_DIY
- Le compte TTN que vous utiliserez pour ce TP est :

Username : **tp.projet.iot@gmail.com**
 Password : **tp_projet_iot**

2. Capteur LoRaWAN DiY

2.1 Composants :



- Microcontrôleur : Arduino Pro Mini



- Module LoRa : RFM96/95W



- Sonde de mesure : DHT22

Tous les composants hormis la sonde sont assemblés sur un PCB, vous n'aurez donc qu'à brancher la sonde pour constituer votre capteur.

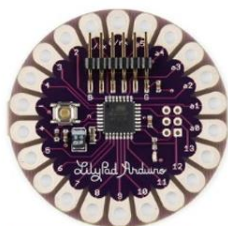
2.2 Programmation :

2.2.1 Introduction Arduino

Arduino Boards



Arduino Nano



Arduino LilyPad



Arduino Mega



Arduino Uno



Arduino Pro Mini



Arduino Leonardo

Avant de commencer cette partie, **télécharger et installer** l'IDE d'Arduino.

Lien du téléchargement : <https://downloads.arduino.cc/arduino-1.8.19-windows.exe>

Ensuite, **installer les drivers** qui sont disponibles sur le Github du TP !

Le logiciel Arduino est un Environnement de Développement Intégré (IDE) open source et gratuit qui permet :

- D'éditer un programme : des croquis (sketch en Anglais), les programmes sont écrits en langage C,
- De compiler ce programme dans le langage « machine » de l'Arduino, la compilation est une traduction du langage C vers le langage (binaire) du microcontrôleur. La console donne des informations sur le déroulement de la compilation et affiche les messages d'erreur,
- De téléverser le programme (binaire) dans la mémoire de l'Arduino, le téléversement (Upload) se passe via le port USB de l'ordinateur une fois dans la mémoire de l'Arduino,
- De communiquer avec la carte Arduino grâce au terminal (ou moniteur série). Pendant le fonctionnement du programme sur l'Arduino, il peut communiquer avec l'ordinateur tant que la connexion est active (câble USB, ...). Cela permettra de suivre l'exécution du programme.

**Téléverser
sur la carte**

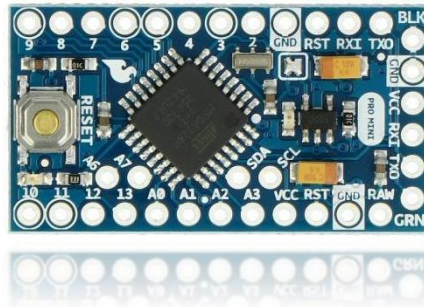
**Vérifier
(compiler)**

**Ouvrir le
moniteur série**



2.2.2 Détail du microcontrôleur du TP : Arduino Pro Mini

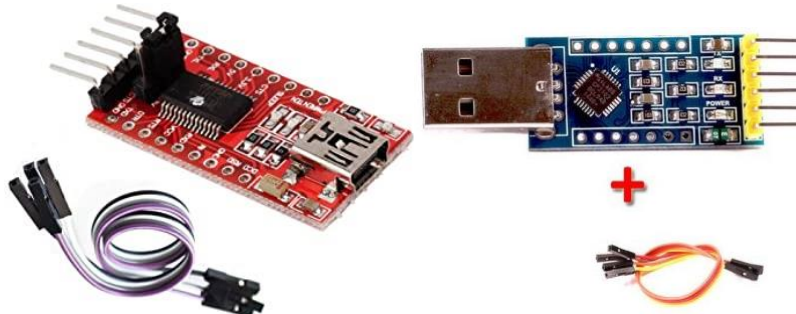
L'ARDUINO PRO MINI est une carte équipée d'un processeur **ATmega328P**. On peut trouver des versions 16MHz (5V) et 8MHz (3V3). Nous nous intéressons ici à la version **8MHz**.



Q1. Nous alimenterons cette carte en 3,3V. Pourquoi utilise-t-on du 3,3V et non du 5V ?

L'ARDUINO PRO MINI a la particularité de ne pas posséder de convertisseur USB/série mais c'est une carte très intéressante pour réaliser des capteurs connectés alimentés sur piles ou batterie : température / hygrométrie, passage, luminosité, etc.

On utilise donc un convertisseur USB TTL :



A titre d'informations, le câblage utilisé est le suivant :

- | | | |
|-----------|---------|-----------------------------|
| - 3V3 | (rouge) | sur la pin VCC de l'ARDUINO |
| - RTS/DTR | (vert) | sur la pin GRN de l'ARDUINO |
| - RXD | (jaune) | sur la pin TX de l'ARDUINO |
| - TXD | (bleu) | sur la pin RX de l'ARDUINO |
| - GND | (brun) | sur la pin GND de l'ARDUINO |

❖ **Le convertisseur est prêt à se connecter, il faut juste faire attention aux sens de celui-ci (n'hésitez pas à demander si vous avez un doute).**

2.2.3 Le “Hello World” de l’Arduino

Il est maintenant l’heure de commencer par programmer un “Hello World” version Arduino.

Néanmoins, avant cela il faut s’attarder sur la **“Digital Logic”, GPIO** pour Arduino.

Le nombre d’entrées-sorties est variable selon les cartes Arduino. Toutefois leurs principes de fonctionnement se retrouvent quasiment à l’identique.

Si la plupart des broches (pins) d’entrées/sorties disponibles sur les cartes Arduino permettent d’émettre ou de recevoir une valeur numérique binaire (0 ou 1), elles offrent aussi des fonctions spécialisées et il est possible d’affecter, via le programme, de décider de la fonction allouée à une broche particulière.

Quelle que soit le pin de l’Arduino, on ne peut y brancher une tension supérieure à la tension d’alimentation, c’est-à-dire 5V ou 3,3V selon le modèle **(pour nous c’est donc du 3,3V)**.

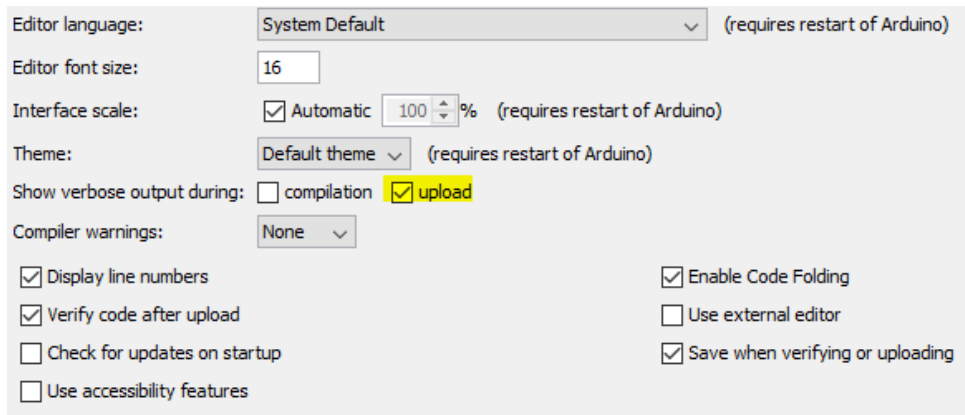
Quasiment tous les pins de l’Arduino peuvent être programmés en entrée ou en sortie numérique (et non les deux en même temps). Par exemple, sur l’Arduino, il s’agit d’une part des pins numérotées de 0 à 13 mais également des broches A0 à A5.

- ENTRÉE - INPUT. L’Arduino peut lire une tension présente sur ce pin en utilisant **digitalRead(...)** et cette tension sera interprétée comme un **logique binaire (0 ou 1)**.
- SORTIE - OUTPUT. Le programme peut écrire un chiffre binaire, au moyen de **digitalWrite(...)**. Ces chiffres sont nommés, **HIGH** pour le **1** et **LOW** pour le **0**, qui sera traduit en une tension de 5V ou 3,3V pour le 1 et de 0V pour le 0.

Durant ce TP nous utilisons que les pins digital (D0 – D13) de la carte (et ils existent aussi des pins analogiques (A0 – A5)).

- La liaison série : Les 2 pins (D0 et D1) étiquetés TX et RX sont respectivement la ligne d’émission série (T pour transmit) et la ligne de réception série (R pour receive). Cette ligne est principalement employée pour dialoguer avec votre PC. Des fonctions permettent d’afficher des messages dans une fenêtre de l’ordinateur hôte et de lire le clavier de l’ordinateur hôte.
- On a aussi un bouton pour RESET l’Arduino.

Avant tout, presser Ctrl + “,”(virgule) et réglez comme ci-dessous :



Editor language: System Default (requires restart of Arduino)

Editor font size: 16

Interface scale: ☒ Automatic 100% (requires restart of Arduino)

Theme: Default theme (requires restart of Arduino)

Show verbose output during: ☐ compilation ☒ upload

Compiler warnings: None

☒ Display line numbers ☒ Enable Code Folding

☒ Verify code after upload ☐ Use external editor

☐ Check for updates on startup ☒ Save when verifying or uploading

☐ Use accessibility features

Puis, choisir l'Arduino Pro Mini dans Tools > Boards et sélectionner le bon Microprocesseur.

❖ A vous de jouer :

(Commencer à présent par ouvrir l'IDE d'Arduino si ce n'est pas déjà fait).

Exercice 1: (Le fameux « Hello World » !!!)

- 1) Faire clignoter la Led branchée sur le pin 3 avec différent intervalle de temps. (Code exemple Blink dans File > Examples > 01. Basics)

Remarque : Nous n'utilisons pas de résistances pour abaisser la tension comme vous pourriez le voir dans d'autres tutoriels sur Internet car avec la tension d'alimentation que nous avons (3,3V) qui est déjà une tension de fonctionnement de la led bleue.

LED Color	Typical Vf Range
Red	1.8 to 2.1
Amber	2 to 2.2
Orange	1.9 to 2.2
Yellow	1.9 to 2.2
Green	2 to 3.1
Blue	3 to 3.7
White	3 to 3.4

- 2) Faire afficher les informations du programme dans une console grâce aux fonctions Serial.print/println. (Exemple : LED allumé, LED éteinte, ...)

Si vous souhaitez suivre un tutoriel plus détaillé sur le matériel Arduino en voici un :

<https://myhomethings.eu/en/arduino-inputs-and-outputs/>

2.2.4 Mesure de Température et d'Humidité

Pour commencer cette partie, importez les librairies de DHT et « Adafruit_Unified_Sensor » depuis le Github du TP dans le dossier de librairies d'Arduino (utiliser le raccourci ctrl + “,” pour savoir où se situe-t-il ou Ajouter le fichier .ZIP depuis Sketch > Include Library).

Exploiter l'exemple “DHT_Unified_Sensor” (dans “Examples from custom library”).

- ❖ Nous n'avons pas besoin de connecter le sonde avec l'Arduino par les fils (c'est-à-dire on peut connecter directement avec les pins digital sur l'Arduino). **Essayez de profiter la fonction digitalWrite().**

Exercice 2 :

- 1) Proposez le plan de câblage du capteur DHT22. Validez le câblage avec le tuteur avant de réaliser le branchement.
- 2) Afficher les données de Température et d'Humidité du DHT22 sur la console d'Arduino IDE.

3. Communication avec TTN

Créer votre device sur TTN avec l'ID de votre groupe comme vous l'avez fait dans le TP LoRaWAN. A partir de maintenant, vous avez les clés pour renseigner (les EUI) dans votre capteur (c'est l'inverse du TP LoRaWAN ! C'est TTN - le LNS, qui vous donne les clés nécessaires pour activer le capteur).

Attention à l'ordre des bits lorsque vous copiez les clés. MSB ou LSB, ce n'est pas le même pour chaque clé. C'est indiqué dans l'exemple en commentaire.

Des boutons utiles sont marqués ci-dessous :

Activation information

AppEUI	54 50 5F 49 6F 54 5F 30	<>	📄
DevEUI	54 50 5F 49 6F 54 5F 31	<>	📄
Root key ID	n/a		
AppKey	📄	👁

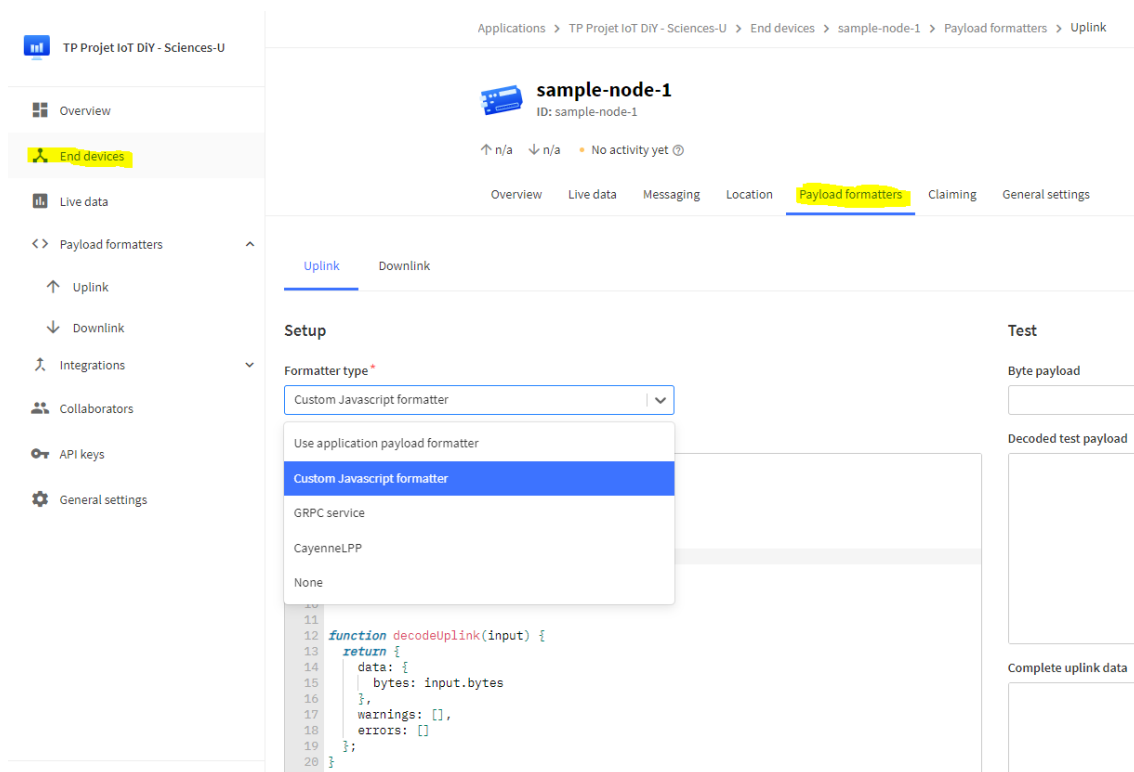
Exercice 3 :

- 1) Importer la librairie de LMIC et modifier le message "Hello World" dans l'exemple "otaa-XXX". (XXX correspondant à votre carte).
- 2) Analyser le programme et essayer de le comprendre pour activer votre capteur sur TTN.
- 3) Vérifier que vous recevez les trames codées en hexadécimal sur TTN.

Maintenant, on est capable d'envoyer les caractères ASCII depuis notre device vers TTN.

Exercice 4 :

Créer votre décodeur sur TTN (**Dans le Payload formatters de Devices et non celui dans Application**) pour retrouver votre message de l'Exercice 3.



The screenshot shows the TTN console interface for configuring a device's payload formatter. The breadcrumb trail is: Applications > TP Projet IoT DIY - Sciences-U > End devices > sample-node-1 > Payload formatters > Uplink. The device 'sample-node-1' (ID: sample-node-1) is selected. The 'Payload formatters' tab is active. Under the 'Setup' section, 'Formatter type' is set to 'Custom Javascript formatter'. A dropdown menu is open, showing options: 'Use application payload formatter', 'Custom Javascript formatter' (selected), 'GRPC service', 'CayenneLPP', and 'None'. Below the dropdown, a JavaScript code editor shows the following code:

```

11
12 function decodeUplink(input) {
13   return {
14     data: {
15       bytes: input.bytes
16     },
17     warnings: [],
18     errors: []
19   };
20 }

```

On the right, the 'Test' section is visible, with fields for 'Byte payload', 'Decoded test payload', and 'Complete uplink data'.

Exemple :

```

function Decoder(bytes, port) {
  // convertir les trames HEX en ASCII
  return {
    // A remplir
  };
}

```


Exercice 5 : Modifier le programme du capteur pour qu'il envoie maintenant un nombre s'incrémentant de 2 à chaque uplink.

A présent, nous possédons tous les éléments nécessaires pour programmer notre capteur, nous pouvons :

- Envoyer des trames portant les données que l'on souhaite vers TTN grâce au protocole LoRaWAN.
- Nous pouvons récupérer les valeurs de température et d'humidité depuis la sonde DHT22.

Exercice 6 : Il ne nous reste plus qu'à combiner tout cela. Remplissez donc le programme « Ex_6 » que vous trouverez sur le Github du TP.

Exercice 7 : Créer un décodeur sur TTN dans votre Device's Payload Formatter pour afficher les données envoyées par votre capteur -> **Validez avec le tuteur !**

❖ **Attacher le programme de l'exercice 6 et le décodeur créé lors de l'exercice 7 dans le compte rendu.**

Downlink depuis TTN (bonus)

Le but est d'allumer ou d'éteindre la led sur pin 3 via un Downlink depuis TTN. Cherchez dans ce tutoriel <http://www.tamberg.org/chopen/2018/LoRaWANIoTWorkshop.pdf> pour programmer votre capteur.

Enfin, si vous arrivez à aller jusqu'à ici, vous pouvez contrôler la lampe par un relais. Bravo !!!

4. Le trio Node-RED – InfluxDB – Grafana

Si vous n'avez pas réussi à programmer le capteur hier, ce n'est pas grave car vous pouvez utiliser « Xloader » pour télécharger le programme final sur votre capteur afin d'envoyer des données à TTN.

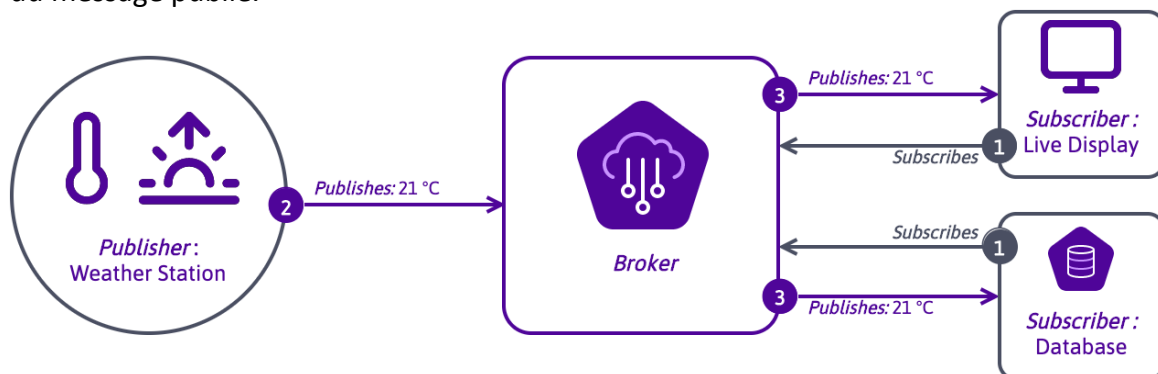
4.1 MQTT avec Node-RED

4.1.1 Introduction MQTT

Message Queuing Telemetry Transport (MQTT) est un protocole de messagerie de publication/abonnement qui permet à deux appareils distants de communiquer par message de manière asynchrone avec une faible bande passante. Ce protocole, spécifiquement dédié au monde du M2M (machine to machine) et de l'IoT (Internet des Objets), est désormais considéré comme un standard du secteur.

4.1.1.1 L'architecture du protocole MQTT

Au cœur de tout protocole MQTT standard basé sur une structure IoT se trouve un serveur distant, appelé Broker. Tous les objets et services s'y connectent en tant que clients. Le Broker transmet les messages entre les clients. Les clients peuvent envoyer des messages en tant que publicateur et recevoir des messages en tant que souscripteur. Les messages publiés contiennent un topic qui décrit le contenu du message (par exemple : la météo à Lyon, France). Les souscripteurs reçoivent chacun une copie du message s'ils ont souscrit au topic du message publié.



Résumé : Les souscripteurs s'abonnent à un topic auprès du Broker (1), le publicateur publie des informations sur le Broker (2) suivant un topic et le Broker publie le sujet auprès des abonnés (3).

En MQTT, les clients s'identifient à l'aide d'un ClientID unique. S'il est laissé vide, le Broker en génère un au hasard.

Lorsqu'un client MQTT se connecte au Broker, il peut choisir de démarrer une nouvelle session ou de reprendre la session existante. Une session contient les abonnements du Client et les messages en attente.

Les topics sont utilisés pour transmettre les messages publiés aux souscripteurs. Un sujet est une chaîne de caractères indiquant le contenu du message (exemple : `maison/2eme étage/cuisine/température`). Avec MQTT, par défaut, tous les souscripteurs reçoivent une copie des messages publiés avec les sujets correspondants.

4.1.1.2 Niveaux de Qualité de Service

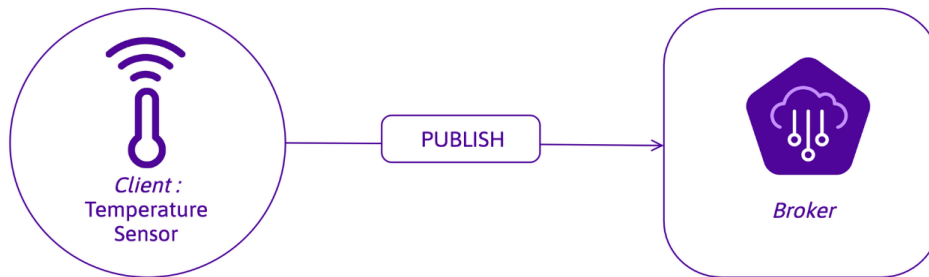
Les niveaux de qualité de service de MQTT déterminent comment la relation entre le client et le Broker doit se dérouler lorsqu'ils communiquent. Ils indiquent le niveau de disponibilité relatif à la distribution d'un message entre les parties communicantes.

Veuillez noter que les Niveaux de Qualité de Service s'appliquent entre un client et le Broker, et non entre un publicateur et un souscripteur.

- QoS 0 - « At most once »

Il également connu sous la dénomination « fire and forget ». Au niveau 0, un message envoyé (PUBLISH) ne peut être reçu qu'une seule fois. L'expéditeur ne stocke pas le message pour les futures transmissions. Par conséquent, si un message est envoyé et que le destinataire n'est pas disponible au moment de l'envoi, la charge utile ne lui parviendra pas. De plus, aucune confirmation de réception ne sera fournie.

QoS 0

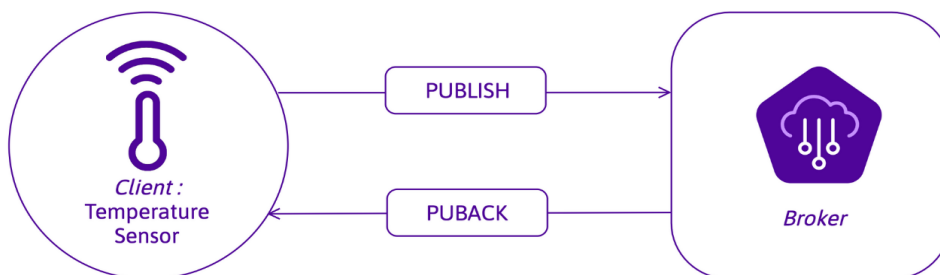


Niveau de Qualité de Service 0 - "fire and forget" : Un client (capteur de température) envoie un message (PUBLISH) au Broker.

- QoS 1 - « At least once »

Ce niveau garantit qu'un message envoyé (PUBLISH) sera reçu au moins une fois. Dans ce cas, si l'expéditeur ne reçoit pas d'accusé de réception (PUBACK) dans le délai prévu, le message peut être envoyé plusieurs fois jusqu'à recevoir une confirmation. La confirmation est envoyée une seule fois par le destinataire en tant que réponse au message d'origine. Par conséquent, si l'expéditeur d'origine d'un message est indisponible au moment de l'acheminement de la réponse, il ne recevra pas de confirmation de réception et continuera à envoyer le même message jusqu'à en recevoir une. Cependant, le destinataire ne comprendra pas qu'il s'agit d'un message dupliqué et l'interprète comme une nouvelle information.

QoS 1



Niveau de Qualité de Service 1 - "at least once" : Un message (PUBLISH) est envoyé par un client (capteur de température) au Broker et un accusé de réception (PUBACK) est envoyé par le Broker au client.

4.1.2 Node-RED

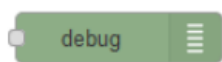
Télécharger et Installer Node-RED.

Suivre la documentation sur : <https://nodered.org/docs/getting-started/local>

Node-RED contient plusieurs nodes de base qui sont très utiles ou pratiques. Ces nodes se retrouvent dans tout flow quelque soit le domaine. Ces nodes sont classés par fonctionnalité. Les fonctionnalités incluses de base dans node-red sont :

- ◆ **Common** : Nodes communes, permettant des opérations simples sans traitements.

Exemples



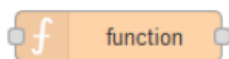
Permet d'afficher un message de debug. Parfait pour tester toute fonction ou node nouvellement ajoutée.



Permet d'écrire un commentaire, sans influence sur le flow.

- ◆ **Function** : Nodes permettant d'agir sur les messages, de modifier leur contenu, de leur soumettre des traitements, et d'influer légèrement sur la façon dont ils sont délivrés

Exemples



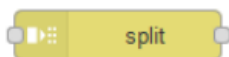
Permet de créer une fonction en JavaScript. Utile pour traiter un message reçu pour le rendre utilisable par une node de sortie.



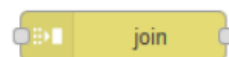
Permet d'imposer un délai aux messages entrants.

- ◆ **Sequence** : Nodes permettant d'agir sur la séquence de messages transmis et ainsi d'agir sur le déroulé du flow.

Exemples :



Permet de diviser un message entrant en plusieurs messages sortants.



Permet de regrouper plusieurs messages entrants en un seul message sortant

Exercice 8 :

- 1) Listez les parties importantes dans la connexion MQTT entre TTN et Node-RED d'après votre compréhension.

Suivre ce tutoriel de TTN <https://www.thethingsindustries.com/docs/integrations/node-red/> et connecter Node-RED avec TTN via MQTT – QoS 1.

- 2) Configurer le node “Function” entre node MQTT et Debug pour récupérer **juste les données décodées** et les afficher sur le “Debug”.

4.2 Introduction d’InfluxDB

Télécharger et installer **InfluxDB 1.8.10** via ce lien <https://portal.influxdata.com/downloads/>

Pour Windows :

```
wget https://dl.influxdata.com/influxdb/releases/influxdb-1.8.10_windows_amd64.zip -  
UseBasicParsing -OutFile influxdb-1.8.10_windows_amd64.zip  
Expand-Archive .\influxdb-1.8.10_windows_amd64.zip -DestinationPath 'C:\Program  
Files\InfluxData\influxdb\'
```

Nous devons d'abord comprendre comment fonctionne InfluxDB. InfluxDB est “time series database”. Ils stockent les données sous forme de points dans le temps et chaque point dans le temps est associé à divers attributs. De plus, l'autre concept principal dont nous devons être conscients lors de l'utilisation d'InfluxDB est que l'enregistrement de données a un timestamp, un set des tags et une valeur mesurée. Cela permet par exemple de créer une valeur nommée Temp et de la taguer en fonction du capteur source :

Temp: Value=19.1 , Sensor=Room1

Temp: Value=21.9 , Sensor=Room2

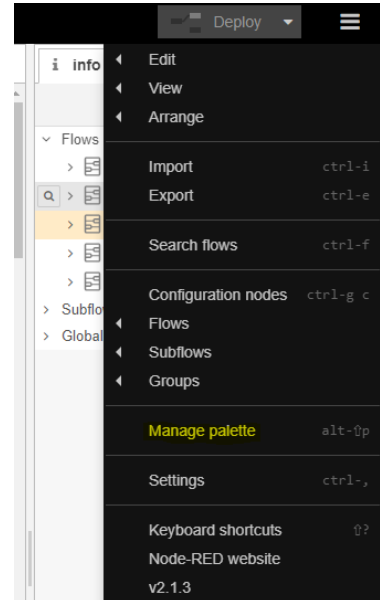
InfluxDB nous permet de traiter toutes les données ou de ne traiter que les données basées sur une ou plusieurs tags spécifiques.

4.3 Créez une base de données InfluxDB :

La création de base de données est un processus simple. Pour créer la base de données, nous devons accéder à votre PC hébergeant le serveur InfluxDB et exécuter la commande « **influx** ».

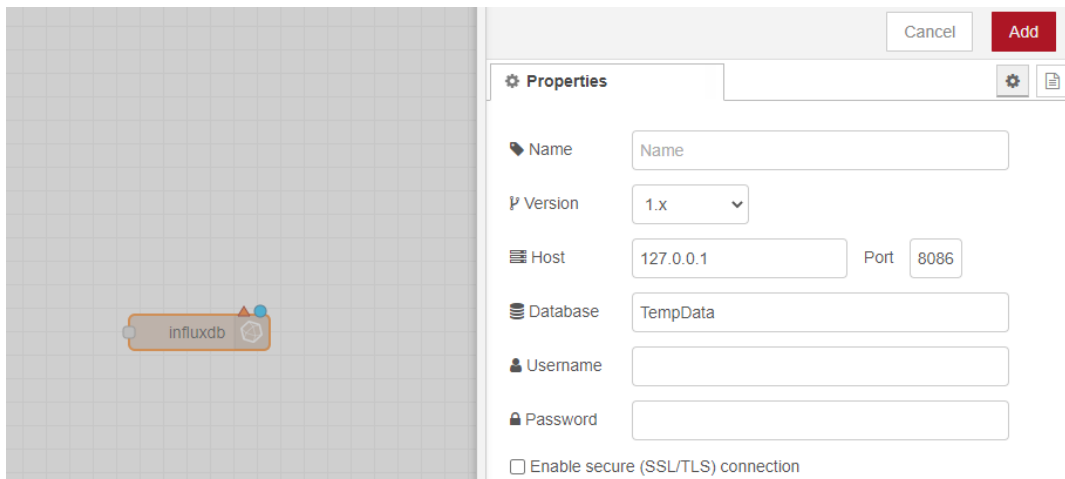
```
~$host influx
Connected to http://localhost:8086
InfluxDB shell version: 1.8.10
> create database TempData
> show databases
name: databases
name
---
_internal
TempData
>
```

Cela crée une base de données appelée « TempData » dans laquelle nous pouvons maintenant commencer à publier des valeurs à partir de TTN.



Exercice 9 :

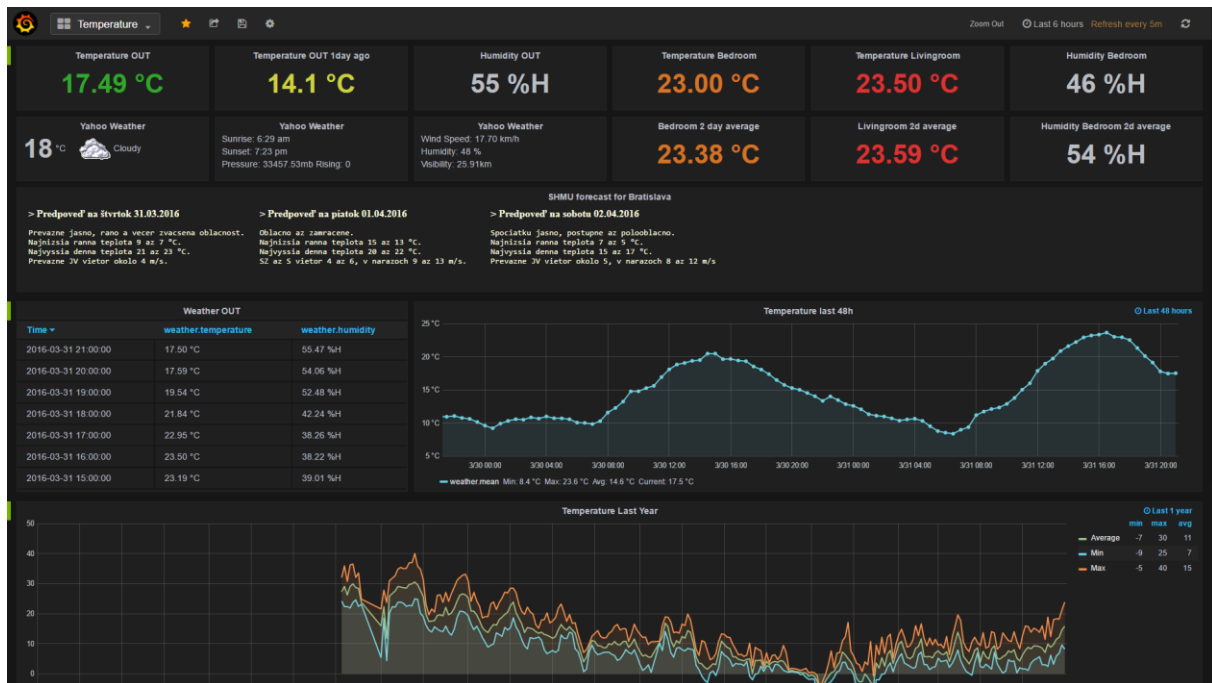
- 1) Revenez à Node-RED et ajoutez la palette « **node-red-contrib-influxdb** ». Ensuite, spécifiez le « **Measurement** » dans node « influxdb out » avant configurer la base de données.



- 2) Vérifier que vos données de l'Exercice 8 sont bien enregistrées sur InfluxDB en utilisant :

```
> show series on TempData
```


4.4 Construction du Dashboard avec Grafana



Télécharger et Installer Grafana sur votre PC.

<https://grafana.com/docs/grafana/latest/installation/>

Exercice 10 :

Le but pour cette dernière partie est de visualiser les données de Température et d'Humidité (sont enregistrées dans InfluxDB) sur Grafana. Il faut juste envoyer les captures d'écran de votre Dashboard dans le compte rendu.

A vous de jouer :

C'est un tutoriel « simple » mais complet pour vous adapter avec votre créativité :

<https://www.thethingsnetwork.org/forum/t/mqtt-node-red-dashboard-with-influxdb-and-grafana-graph-in-aws-howto/49854>

5. Conclusion

Après ce TP nous avons travaillé du niveau physique (capteurs) aux niveaux de télécommunications ainsi que serveur d'application pour nous aider à avoir une vision globale et plus complète d'un système IoT.

Nous contacter



contact@optimiz-network.fr



09 51 57 28 22
06 50 96 03 98



Bâtiment des Hautes Technologies

20 Rue Professeur Benoît Luras,
42000 Saint-Étienne



@optimiznetwork

