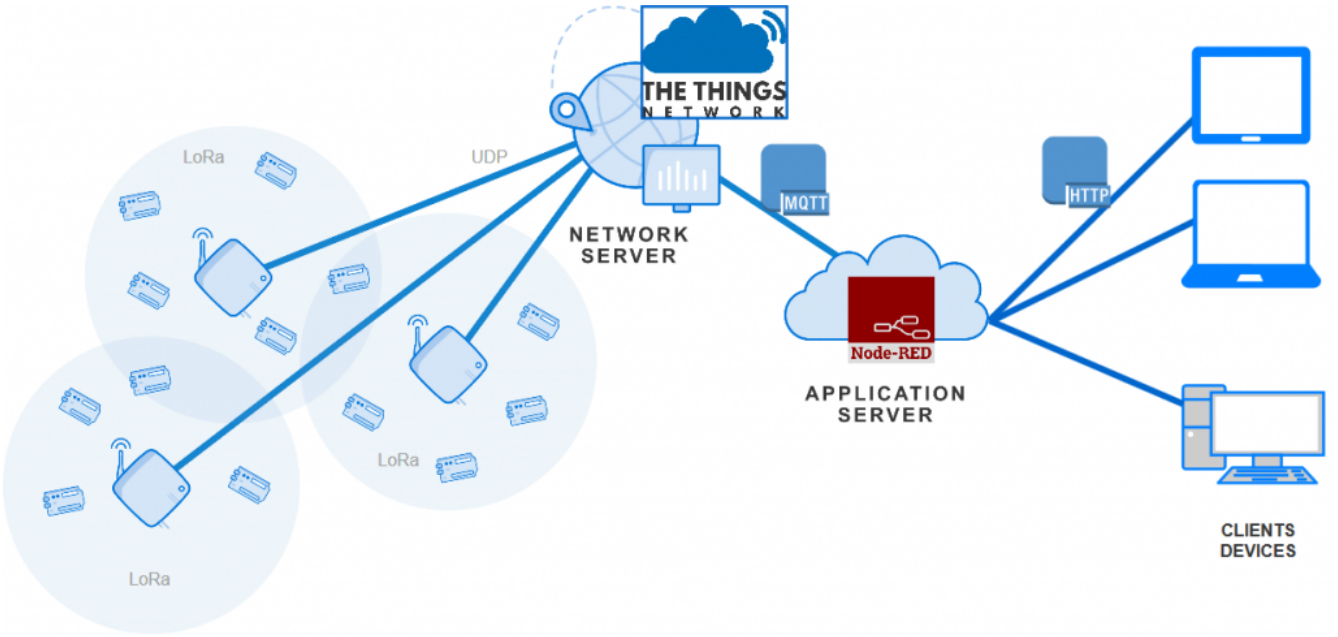
Développement d’un objet connecté LoRa/LoRaWAN

Sommaire

1. Introduction



Les objets connectés ou l'**Internet des Objets** (*IoT : Internet of Things*) sont des termes très utilisés de nos jours même si bien des personnes ont encore du mal les définir.

Au sens large, un objet connecté est un objet communicant. Ils sont à la fois des émetteurs, des capteurs d'informations qui émettent, mais également qui peuvent interagir avec d'autres machines (M2M) comme des serveurs ou d'autres objets connectés. Les données générées par ces objets sont capitalisées dans des centres de données (*data centers*) et de nouveaux usages de cette donnée apparaissent autour de la santé, du sport, des produits financiers par exemple.

Les objets connectés connaissent une croissance exponentielle. Des estimations indiquent qu'ils seraient près de 15 milliards en circulation dans le monde actuel. Ils pourraient être plus de 50 milliards d'ici 2020.

Pour connecter les objets, les technologies actuelles peuvent être utilisées (`RFID`, `Wi-Fi`, `GSM`, `Bluetooth`, `Z-Wave`, `ZigBee`, ...) mais de nouveaux réseaux se développent au niveau mondial. Ils doivent permettre d'envoyer (mais aussi recevoir dans certains cas) **des très petits messages sur des longues portées** sans passer par des systèmes coûteux de réseaux mobile et **en consommant peu d'énergie**.

> Au cours de ce TP, vous allez mettre en œuvre un objet connecté à l'internet des objets par le réseau LoRa de **The Things Network**.

En France, plusieurs réseaux de l'internet des objets sont déjà déployés comme le réseau **SigFox**, les réseaux **LoRa Objenious** de **Bouygues Telecom**, **Datavenue d'Orange**. **Qowisio** a également lancé son propre réseau ainsi que **Archos** qui déploie un réseau collaboratif en fournissant des passerelles `picoWAN` qui seront connectées à Internet pour transmettre les informations du terrain.

De nombreuses entreprises ont déjà choisi d'expérimenter le réseau **LoRa** pour diverses applications. Contrairement à son concurrent **Sigfox**, **LoRa** est un réseau ouvert. Toute entreprise

peut donc créer son propre réseau pour l'exploiter. Il faut pour cela se munir d'une antenne reliée à internet (par Wi-Fi, câble Ethernet, connexion 3G…) avec une station de base émettant **en France sur la bande 868 MHz**.

Le réseau peut être privé ou public suivant le domaine d'application. Une entreprise préférera protéger les données transmises. A noter que la bande de fréquence disponible change par pays. **Aux Etats-Unis**, par exemple, la bande de fréquence utilisée pour le réseau LoRa est **915 MHz**.

2. Objectif de l'activité

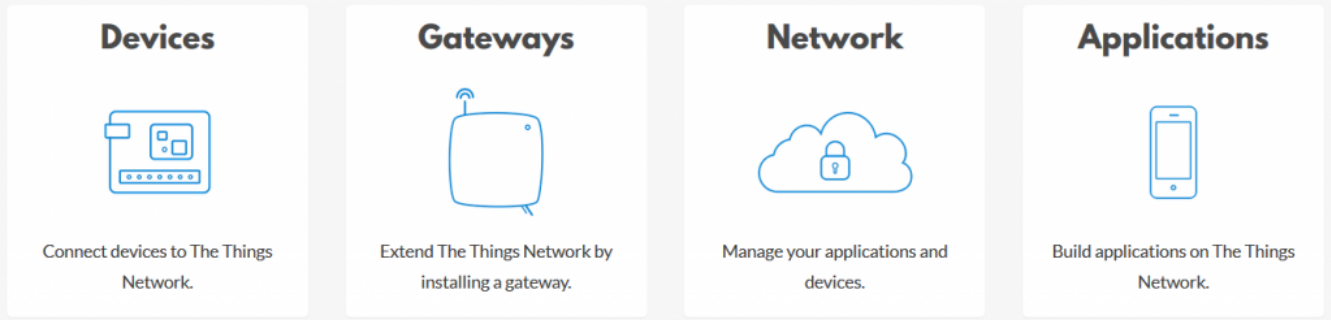
- Réaliser un "node" ou objet connecté : capteur de température et humidité relative.
- Intégrer un "node" ou objet connecté dans le "*cloud*" TTN (The Thing Network)
- Réaliser une application cliente pour la mise à disposition des mesures.

3. The Things Network

Visiter le site web officiel **The Things Network** : https://www.thethingsnetwork.org/

3.1. Présentation

The Things Network est un réseau communautaire fonctionnant avec la technologie LoRa conforme au standard LoRaWAN. Cette technologie permet à des objets de transmettre de petits volumes d'informations en consommant très peu d'énergie.



La communauté déploie des passerelles connectées au réseau Internet. Les passerelles sont utilisables par tout le monde, permettant à vos objets de continuer à émettre des données vers vos applications, même s'ils sont à des milliers de kilomètres de chez vous, en utilisant les passerelles à proximité. De la même façon, vos passerelles sont susceptibles de recevoir des données d'objets ne vous appartenant pas. Votre passerelle transmet les informations vers l'infrastructure centrale sur réseau qui se charge de les acheminer vers les applications qui les utilisent.

The Things Network développe une passerelle économique pour supporter le déploiement communautaire du réseau, mais il est possible d'utiliser des passerelles d'autres fabricants sur ce même réseau.

Figure 1. https://www.thethingsnetwork.org/docs/gateways/gateway

Plus le nombre de passerelles déployées est grand meilleur est la couverture. Aujourd'hui les Pays-Bas ont plusieurs zones à forte densités permettant un déploiement efficace d'applications. D'autres réseaux d'opérateurs reconnus, tel Orange, Bouygue, KPN et d'autres dans différents pays sont disponibles mais leur utilisation est soumise à abonnement payant, avec des garanties associées, qui peuvent ne pas être nécessaire dans le cadre de votre usage.

The Things Network vous permet de profiter d'un service similaire à coût nul.

Figure 2. Implantation des passerelle LoRa TTN en janvier 2020

3.2. Installation d’une passerelle

ATTENTION : Cette partie n’est pas à réaliser dans le cadre du TP. La passerelle que vous utiliserez est déjà installée et configurée. Les login et mot de passe à utiliser pour s’y connecter vous seront communiqués par le formateur.

Pour installer la passerelle **The Things Network**, vous devez disposer d’un ordinateur muni d’une carte wifi. En effet, la seconde étape consiste à se connecter à un réseau wifi fournit par la passerelle.

L’installation se fait en 4 étapes :

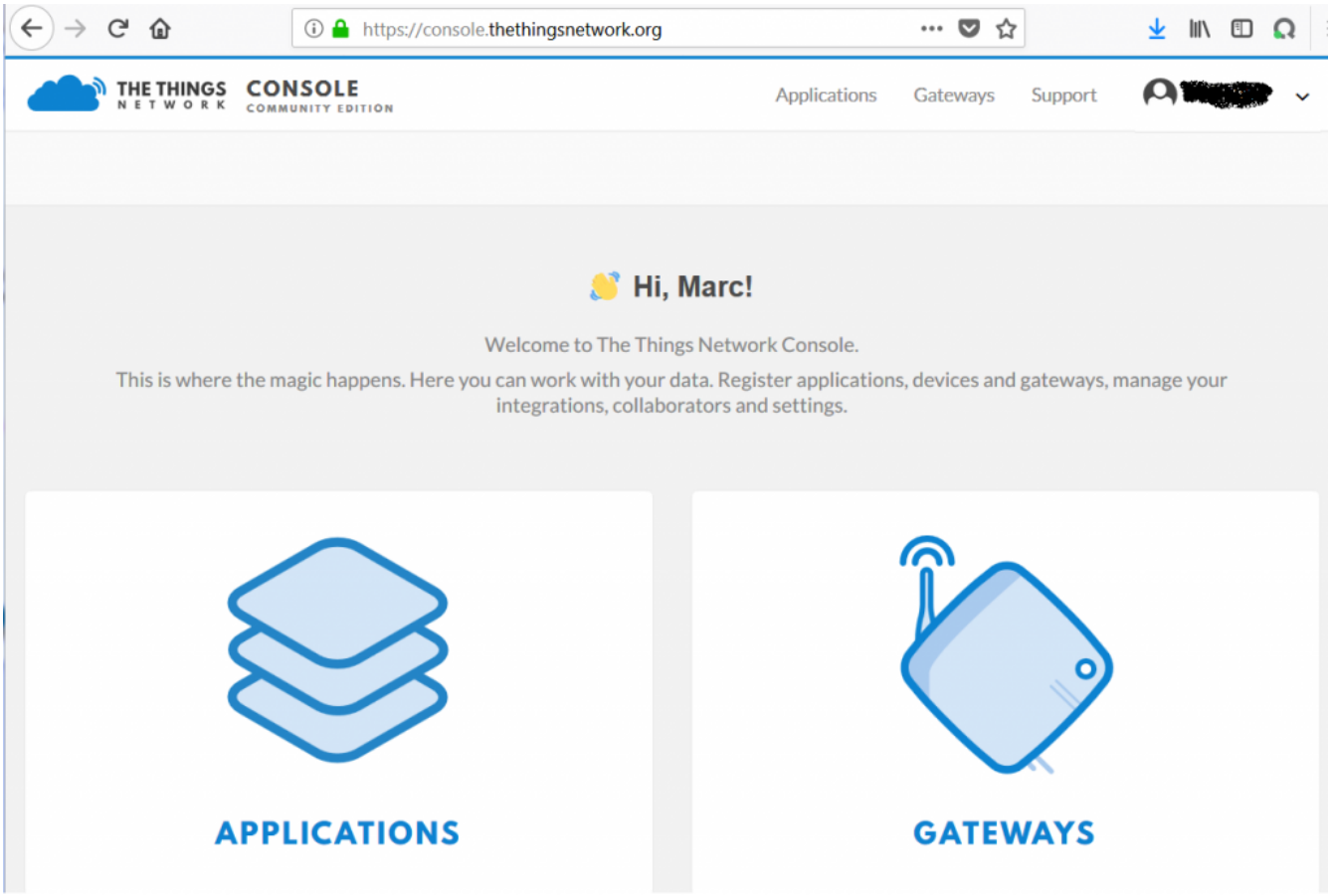
- Enregistrement
- Connexion
- Configuration
- Premier message

Pour procéder à l’enregistrement de la passerelle sur les serveurs **TheThingsNetwork.org** (TTN), connectez-vous à l’adresse suivante : https://activate.thethingsnetwork.org/ Vous trouverez à l’adresse suivante un tutoriel vidéo pour vous guider dans les différentes étapes :

Suivre le tutoriel vidéo

4. Administration de la passerelle et des applications

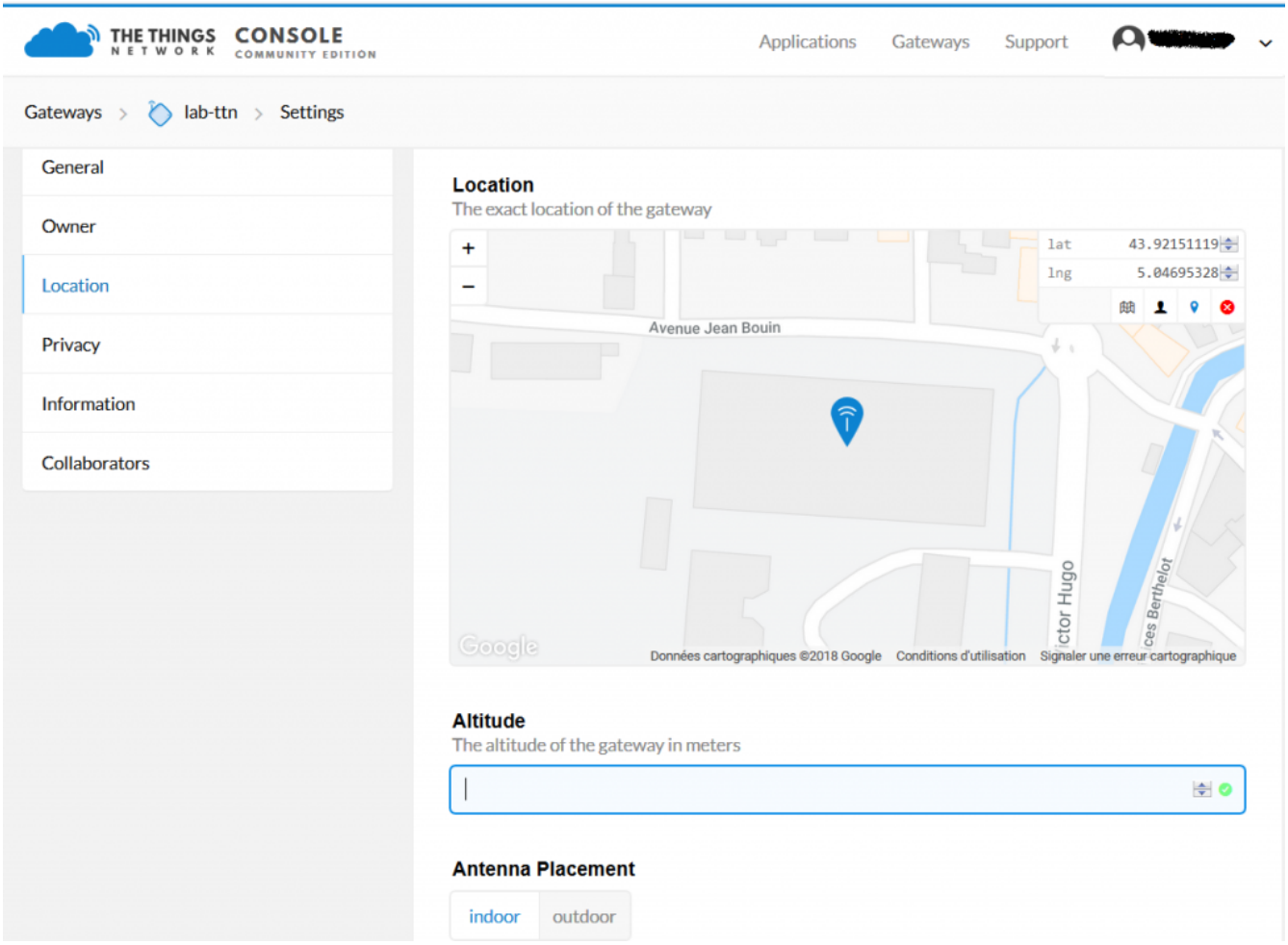
Elle se fait à partir de la console TTN accessible ici : https://console.thethingsnetwork.org/



4.1. Administration

ATTENTION : NE PAS MODIFIER LA CONFIGURATION DE LA PASSERELLE.

Pour modifier ou compléter la configuration de la passerelle, cliquez sur GATEWAY. Vous pourrez ainsi la géolocaliser, indiquer si elle est placée à l'intérieur ou à l'extérieur, de quel type d'antenne elle est dotée, qui l'administre, ...



4.2. Ajout d'une application

- Documentation : https://www.thethingsnetwork.org/docs/applications/
- Tutoriel video : https://www.youtube.com/watch?v=JrNjY-pGuno

Pour ajouter une application, à partir de la console, cliquez sur `APPLICATIONS`.

Par **Applications**, il faut entendre tout ce que vos objets communiquent sur Internet. On peut également voir une application comme une collection d'objets (`Devices`).

Pour pouvoir enregistrer un objet connecté sur la passerelle, il faut nécessairement disposer d'une application dans laquelle le ranger.

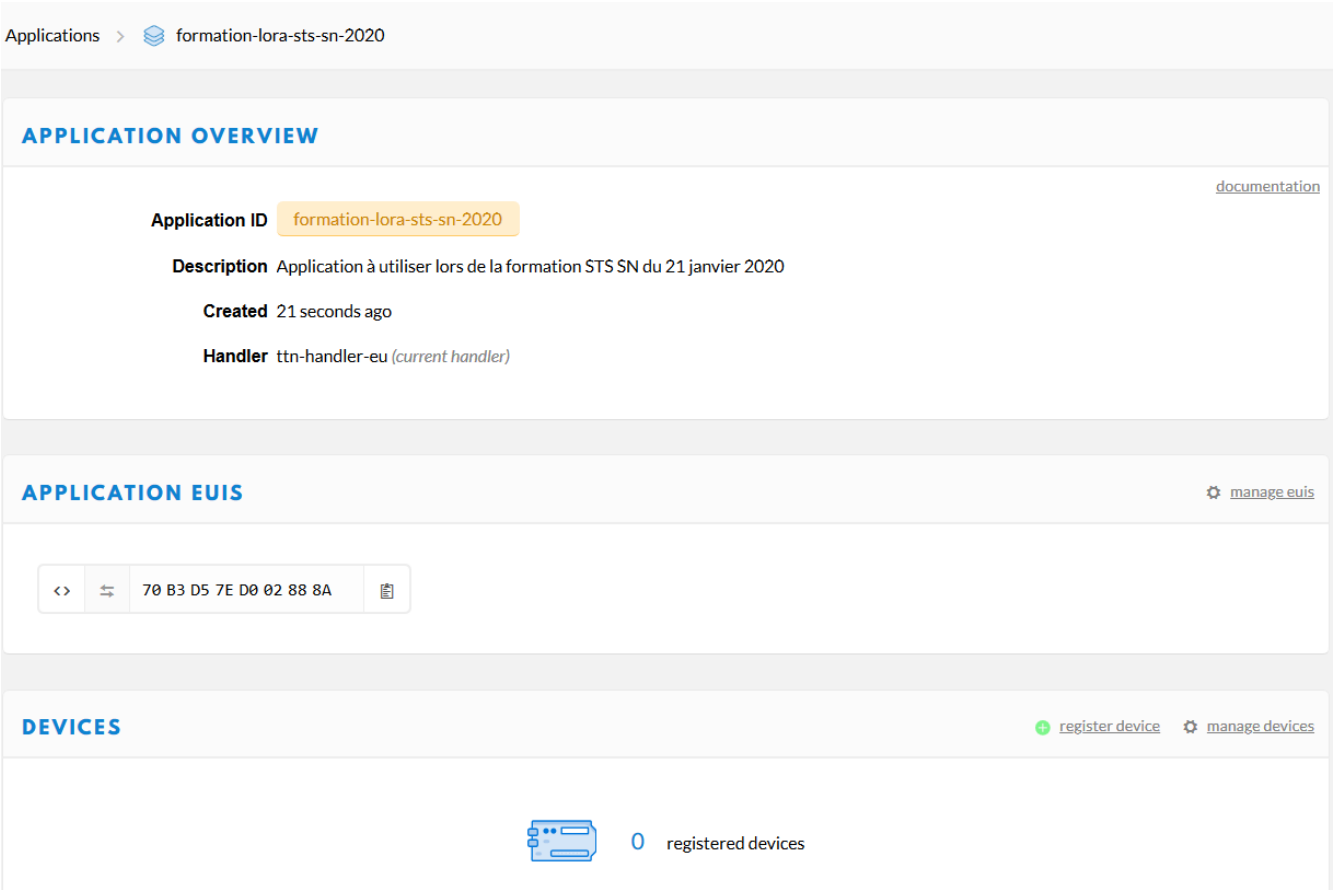
Prenons l'exemple de capteurs et d'actionneurs qui contribuent à la gestion d'un bâtiment qui s'appellerait `Le Machin` dans la ville de `Truc`. On pourrait créer une application nommée `le-machin-a-truc` dans laquelle on déclarera les capteurs `capteur-type-x` ou `type=temperature, humidite, presence, luminosite, ⋯` et `x` un numéro d'identification, idem pour les actionneurs.

Si vous ajouter votre première application, vous pouvez cliquer sur `Get started by adding one`, sinon cliquez sur `+ add application`.

- Complétez dans le formulaire les champs `Application ID` et `Description`. Laissons **The Things Network** décider de l'attribution d'un identifiant `Application EUI` (`Extended Unique Identifier`).
- Le serveur avec lequel les échanges de données se feront sera ici `ttn-handler-eu`.
- Cliquez sur le bouton `Add application` en bas de la page.

Nous pouvons maintenant ajouter des objets connectés à cette application.

Vous pouvez tous utiliser la même application mais la modification du format des données impactent l'ensemble des objets. Il est donc préférable que vous ayez votre propre application, même si elle ne possède qu'un seul objet.



5. Ajout d'un objet connecté

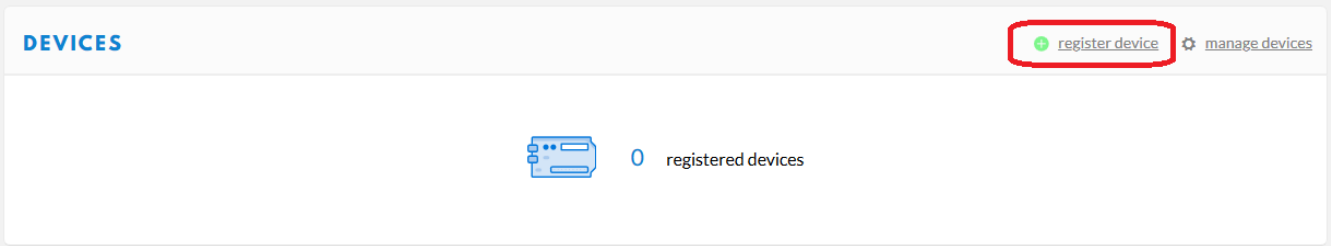
- Documentation : https://www.thethingsnetwork.org/docs/devices/registration.html
- Tutoriel vidéo : https://www.youtube.com/watch?v=28Fh5OF8ev0

Nous allons ajouter un capteur de température et d'humidité en utilisant une carte `The Things Uno` construite à partir d'une carte `Arduino Leonardo` et qui intègre une puce LoRa de chez **Microchip** et on lui connectera un capteur `DHT11` ou `DHT22`.

Figure 3. Carte The Things Uno

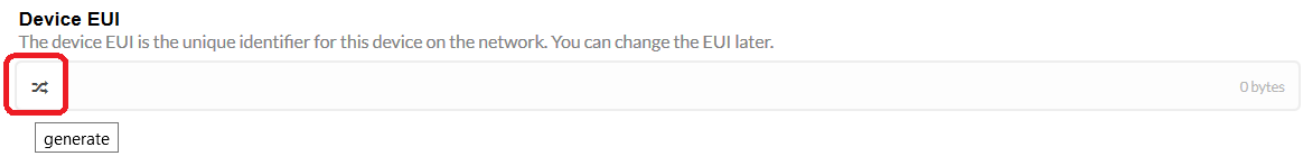
5.1. Enregistrement d'un nouvel objet connecté

- Cliquez sur `register device` pour ajouter un objet connecté à l'application.

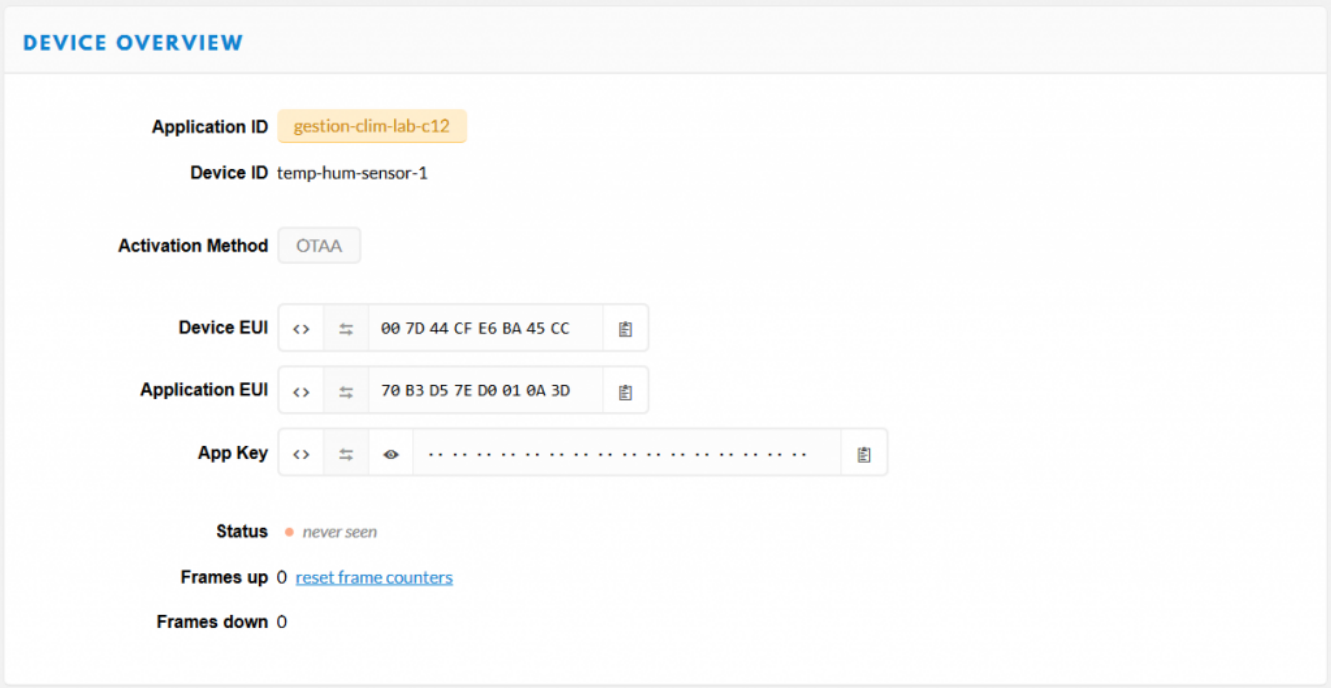


A ce stade, le seul élément nécessaire à configurer est le `Device ID` : `temp-hum-sensor-x` (`x =1..12`).

Il nous faut également renseigner le `Device EUI` qui est un identifiant propre à la puce LoRa utilisée. On peut toutefois en indiquer un manuellement mais dans notre cas nous allons lire le `DevEUI` de la puce en utilisant un programme d'exemple fourni avec la librairie `TheThingsNetwork` d'Arduino. Pour l'instant, laissons **The Things Network** le générer automatiquement :



- Cliquez sur le bouton `Register`.



L'enregistrement à générer un `Device EUI` aléatoire. Nous allons par la suite le remplacer par l'authentique.

Le champs `Activation Method` définie la manière dont les équipements vont obtenir les clés de session nécessaire à l'établissement de la communication entre eux. Deux types de procédures d'activation sont disponibles :

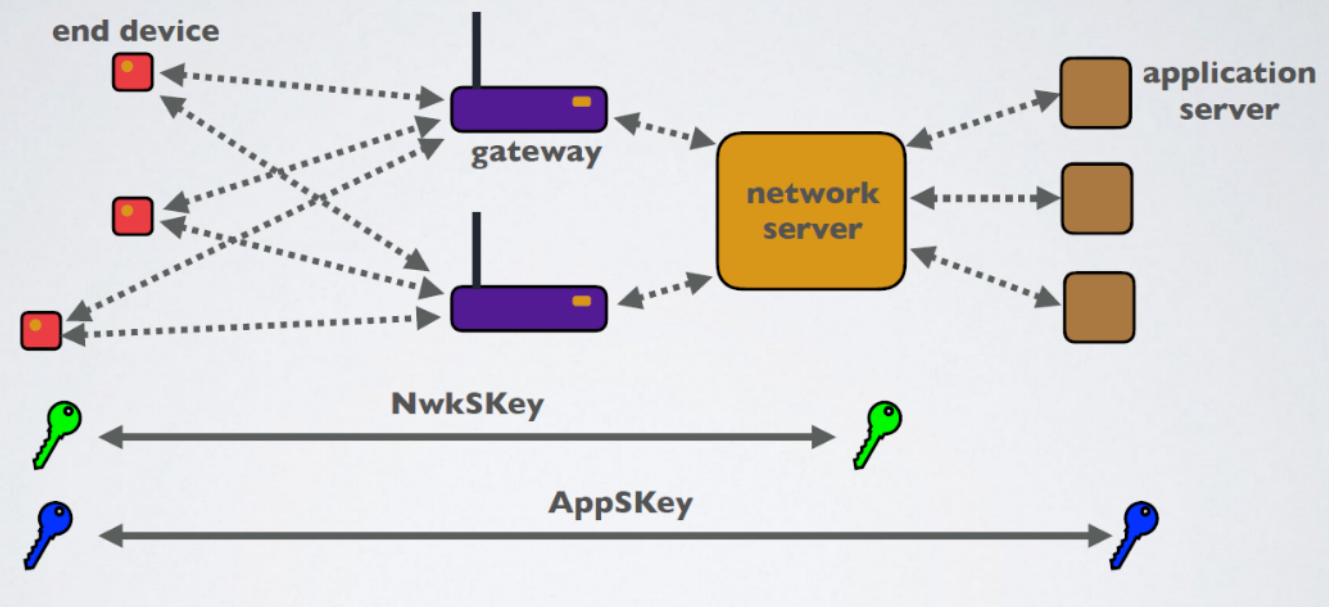
- **Over-The-Air Activation** (`OTAA`)
- **Activation By Personalization** (`ABP`).

Nous choisirons la méthode d'activation `OTAA`.

5.2. Rappel : Méthodes d'activation

L'activation a pour but d'établir une communication sécurisée entre les équipements. Elle est réalisée à partir d'une paire de clés `NwkSKey` pour chiffrer/déchiffrer les données entre le node et le serveur réseau et `AppSKey` pour chiffrer/déchiffrer les données entre le node et le serveur d'applications.

Sécurisation des échanges LoRaWAN



Méthode ABP

Pour la seconde méthode, **ABP**, les clés de session `NwkSKey` et `AppSKey` ainsi que l'adresse de l'équipement (`DevAddr`) sont directement inscrits dans l'équipement LoRaWAN. Ainsi, l'équipement n'a plus besoin d'envoyer de requête avant de communiquer sur le réseau.

L'utilisation de cette méthode implique que les équipements communiquent avec un réseau spécifique car les clés de session sont connues par avance. Et contrairement à la méthode **OTAA**, les clés de session sont statiques.

En résumé, la méthode **OTAA** est plus complexe à implémenter que la méthode **ABP** mais offre un niveau de sécurité supérieur.

Dans le cas d'un prototypage ou pour une utilisation sur un réseau connu, la méthode **ABP** suffit largement. Par contre, lorsqu'un déploiement à plus grande échelle est envisagé, il est conseillé d'utiliser la méthode **OTAA**, plus sécurisée et plus agile.

Méthode OTAA

Pour activer un équipement sur le réseau par la méthode `OTAA`, l'équipement doit transmettre au réseau une demande d'accès : `join request`. Pour ce faire, celui-ci doit être en possession de trois paramètres :

- `DevEUI`, identifiant unique (de type `EUI 64`) de l'équipement (fourni par l'équipementier).
- `AppEUI`, identifiant du fournisseur de l'application (`EUI 64`).
- `AppKey`, clé `AES 128` déterminée par le fournisseur de l'application.

L'équipement envoie, à travers le réseau, la requête `join request`, contenant `DevEUI`, `AppEUI` ainsi qu'un `MIC` (`Message Integriry Code`) calculé via la clé `AppKey`. Cette requête est transmise au serveur d'enregistrement qui vérifie le `MIC` via la clé `AppKey` (qui lui a été communiquée au préalable).

Si l'équipement est autorisé par le serveur d'enregistrement, la requête `join accept` est transmise en réponse à l'équipement. Cette réponse contient des données à partir desquelles l'équipement va pouvoir calculer les **clés de session** (**réseau** : `NwkSKey` et **applicative** : `AppSKey`).

Parmi les données contenues dans cette réponse, se trouve également l'adresse **Device Adress** (`DevAddr`) sur 32 bits – qu'utilisera l'équipement pour communiquer sur le réseau.

A chaque nouvelle session, les clés de session sont renouvelées.

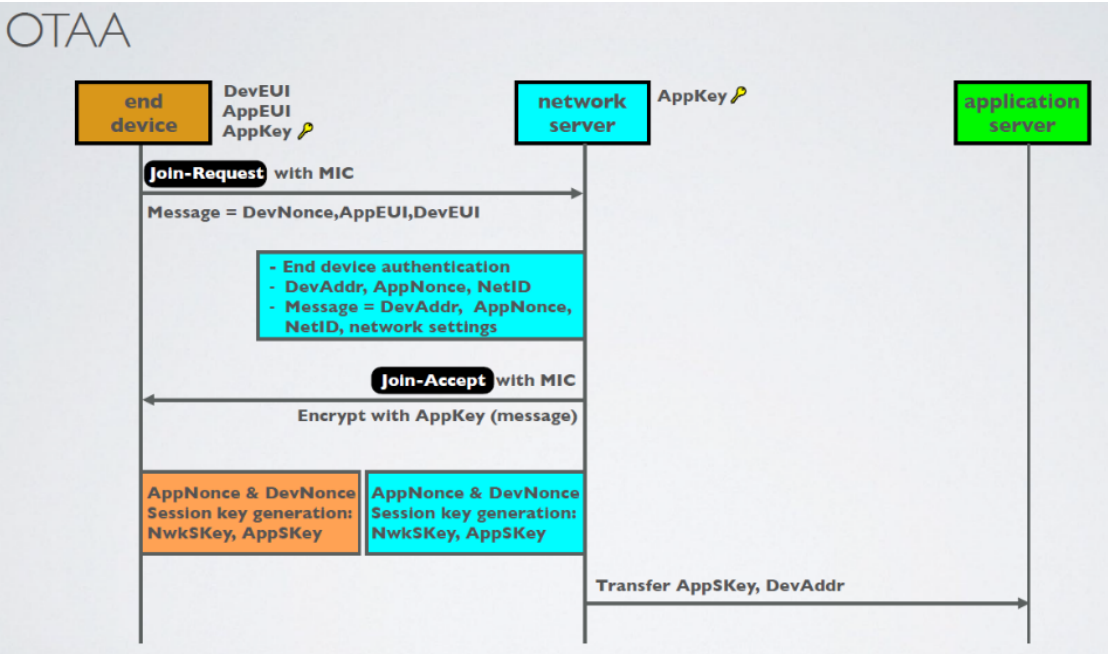


Figure 4. Principe de l' activation OTAA

5.3. Configuration du Device Extended Unique Identifier (DevEUI)

Pour obtenir le DevEUI, commençons par installer la librairie Arduino TheThingsNetwork dans l'EDI Arduino :

- Dans l'environnement de développement d'Arduino, cliquez sur Croquis→Inclure une bibliothèque→Gérer les bibliothèques.
- Dans la barre de recherche, saisissez thethingsnetwork
- Procédez à l'installation de la librairie

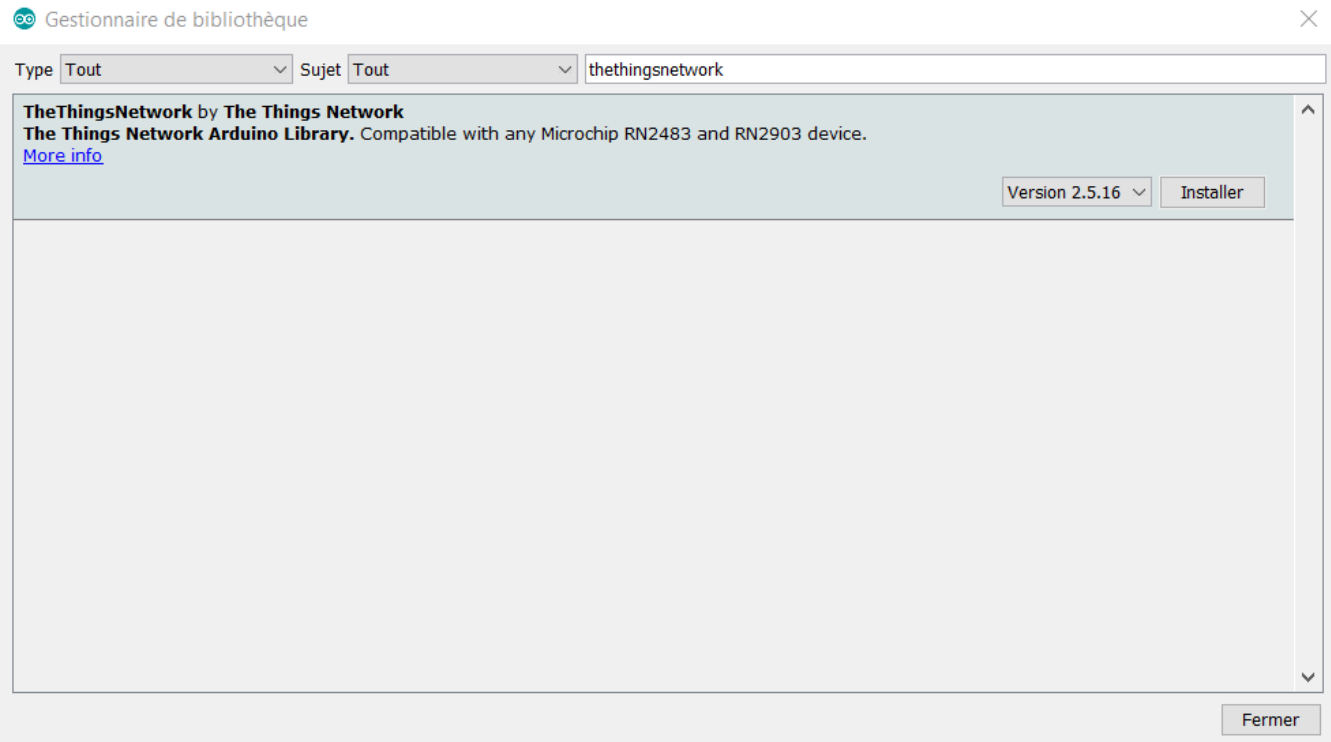


Figure 5. Installation de la librairie TheThingsNetwork

Nous disposons maintenant de la librairie et de ses programmes d'exemples. Nous allons ouvrir le programme DeviceInfo.ino :

Fichier→Exemples→TheThingsNetwork→DeviceInfo

- Modifiez le plan de fréquence pour l'Europe.
- Connectez la carte The Things Uno au PC.
- Indiquez à l'IDE que vous utilisez une carte de type Leonardo.
- Compilez et téléversez.
- Ouvrez le terminal pour lire les infos :

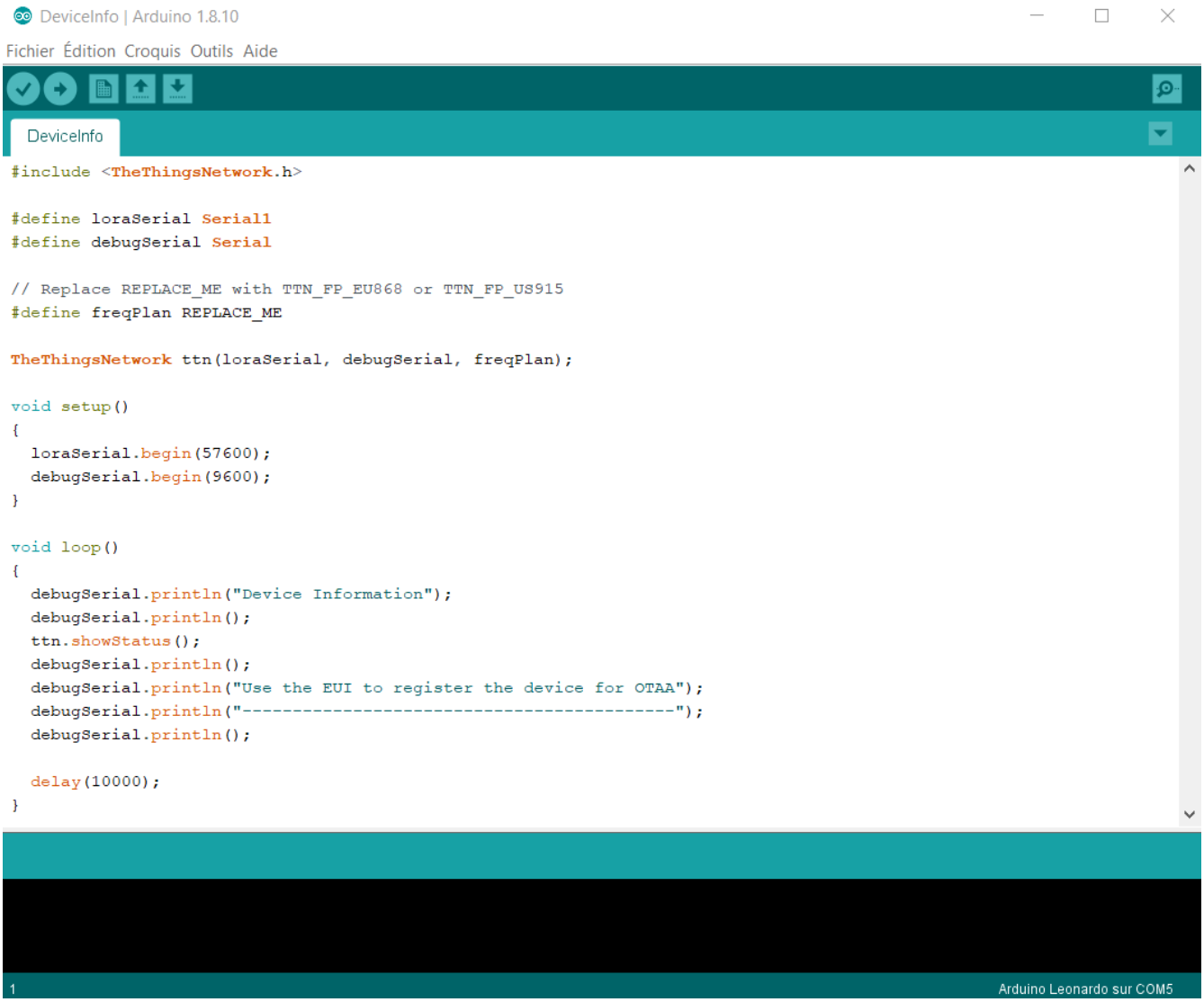


Figure 6. Programme d'exemple de la librairie The TheThingsNetwork

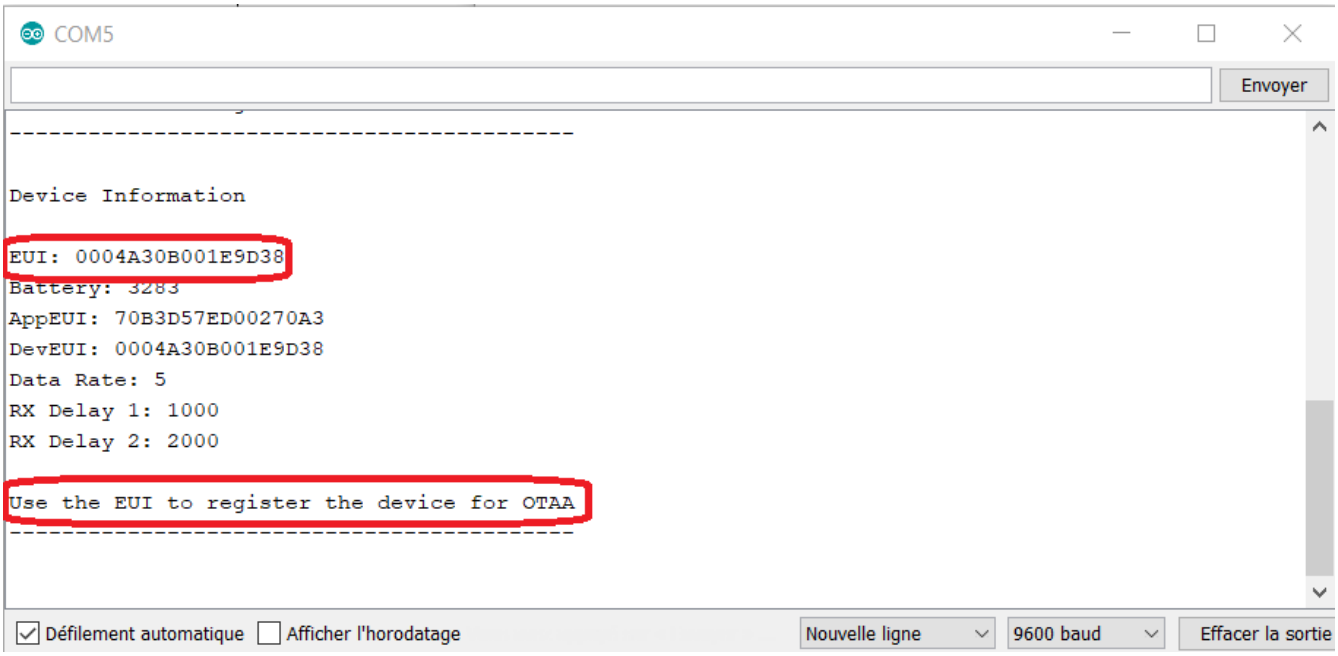


Figure 7. Lecture du DevEUI de la puce LoRa

- Configurez dans la console TTN le `DevEUI` :
  - Dans la page de l'objet connecté, cliquez sur `settings`
  - Remplacez le `DevEUI` précédemment généré aléatoirement par celui de la puce LoRa

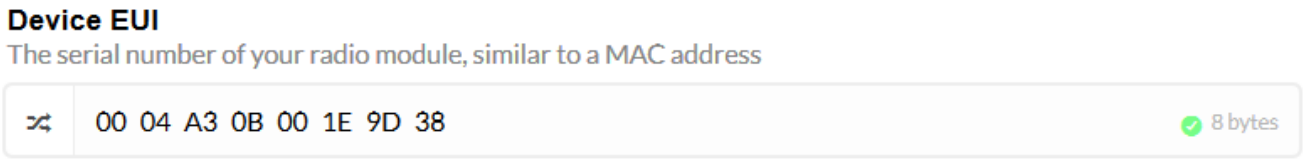


Figure 8. Configurer le DevEUI d'un objet dans la console TTN

- Ajoutez une description qui contient votre nom à l'objet (pratique pour la gestion des objets enregistrés).
- Cliquez sur `Save` pour enregistrer l'objet.

`The Things Network` a généré les clés `AppKey` et `AppEUI` nécessaire à l'établissement de la liaison entre l'objet et le serveur TTN :

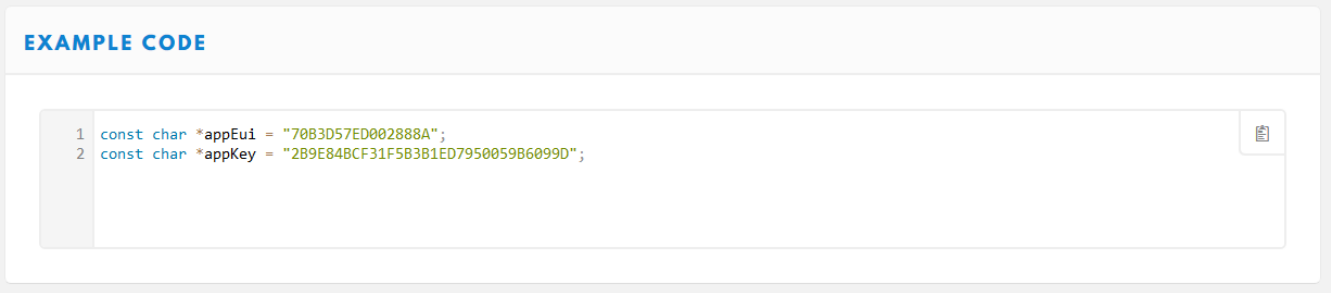


Figure 9. AppEUI et AppKey générées par **The Things Network** *pour l'objet*

5.4. Capteur de température et d'humidité

Dans l'IDE d'Arduino :

- Ouvrez le programme d'exemple **DHT** : `Fichier→Exemples→TheThingsNetwork→Sensors→DHT`
- Modifiez le plan de fréquence, `AppEUI` et `AppKey` conformément aux informations contenues dans la déclaration du `device` dans la console TTN.

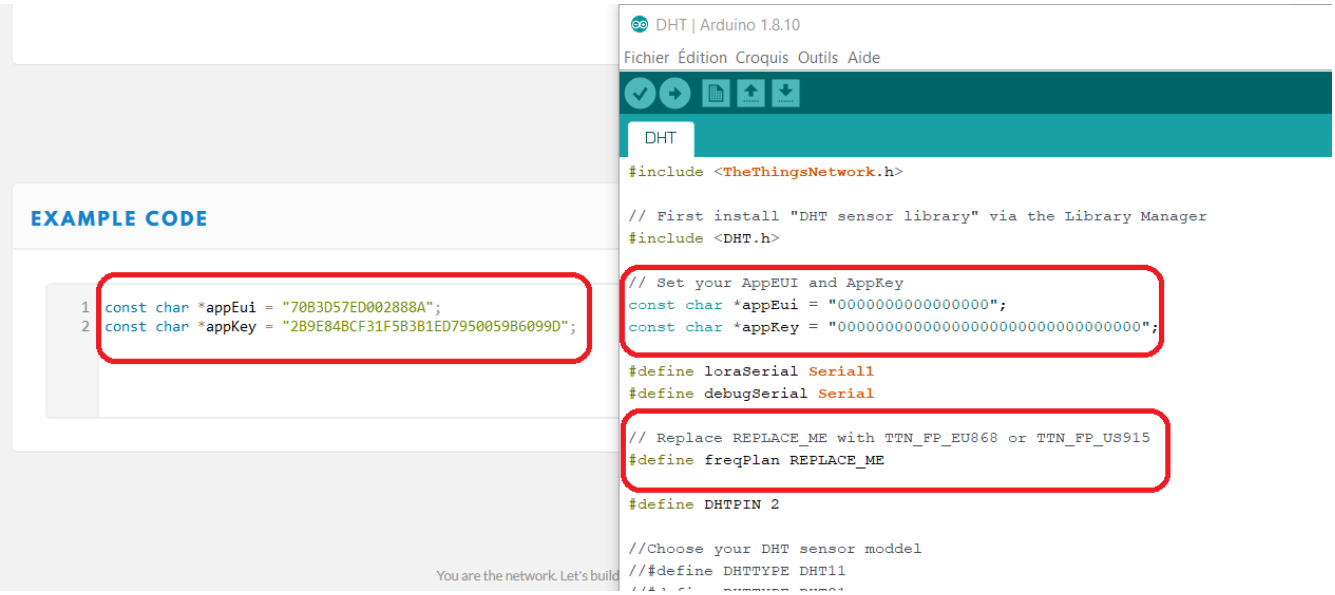
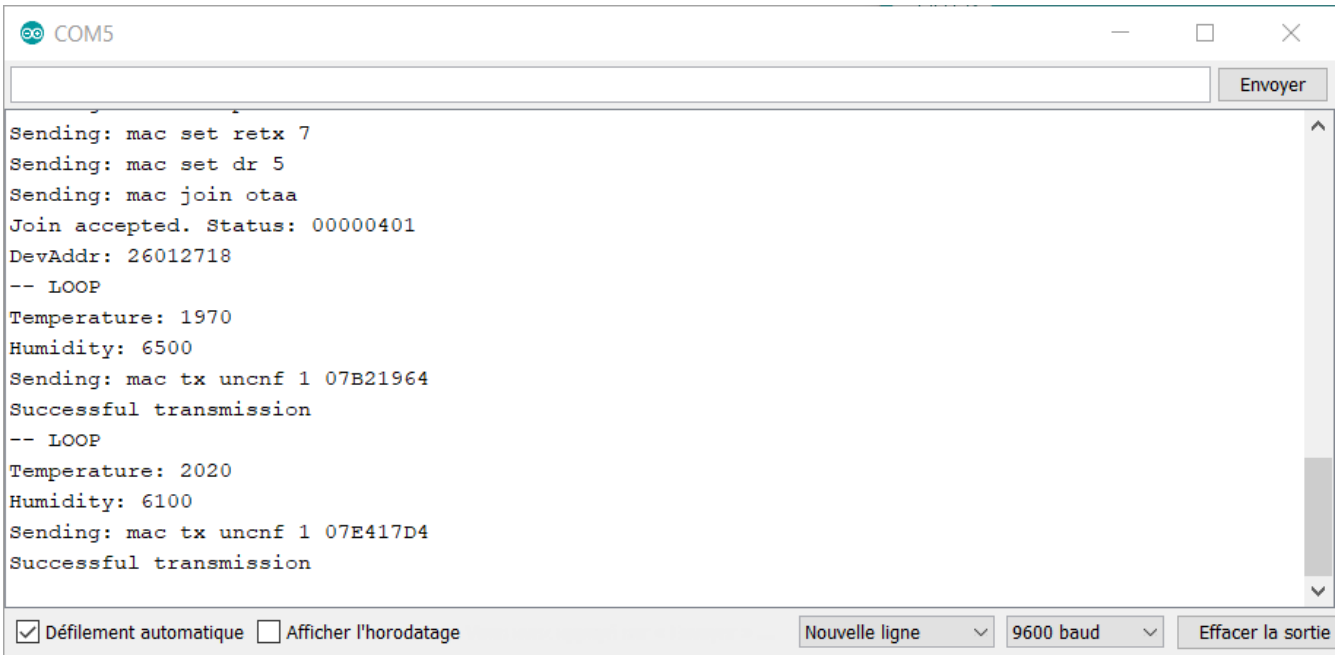
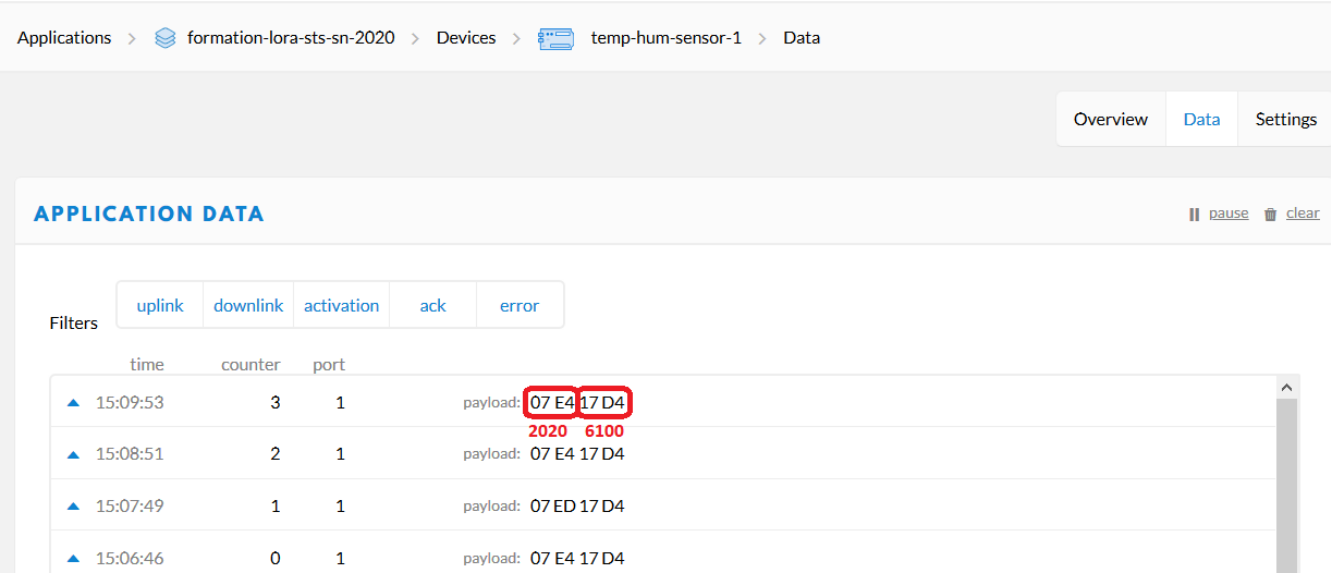


Figure 10. Renseigner les AppEUI et AppKey dans l'objet

- Choisissez le module `DHT` mis à votre disposition et connectez-le à la carte `The Things Uno` conformément à la déclaration faite dans le code.
- Raccourcissez le délai entre deux mesures à 10 secondes pour les essais puis configurez le à 1 minute pour ne pas saturer le réseau (*duty cycle*).
- Compilez (il se peut que vous ayez à ajouter la librairie `DHT sensor library by Adafruit` et toutes ses dépendances), téléversez et observez le terminal série :



- Observez les données dans la console TTN :



- Conformément à la programmation de l'objet, on lit **4 octets**, les **deux premiers** sont la **température** en centième de degrès Celcius et les **deux derniers** sont l'**humidité relative** en centième de pourcent. Ici on lit **20,20°C et 61,00%**.

```
void loop()
{
  debugSerial.println("-- LOOP");

  // Read sensor values and multiply by 100 to effictively have 2 decimals
  uint16_t humidity = dht.readHumidity(false) * 100;

  // false: Celsius (default)
  // true: Farenheit
  uint16_t temperature = dht.readTemperature(false) * 100;

  // Split both words (16 bits) into 2 bytes of 8
  byte payload[4];
  payload[0] = highByte(temperature);
  payload[1] = lowByte(temperature);
  payload[2] = highByte(humidity);
  payload[3] = lowByte(humidity);

  debugSerial.print("Temperature: ");
  debugSerial.println(temperature);
  debugSerial.print("Humidity: ");
  debugSerial.println(humidity);

  ttn.sendBytes(payload, sizeof(payload));

  delay(60000);
}
```

- Dans la console TTN, proposez pour votre application une fonction de décodage pour rendre les données aisément lisible : Cliquez sur le menu `Devices` puis sur le bouton `Payload formats` et modifiez l'exemple de fonction `Decoder(bytes, port)` :

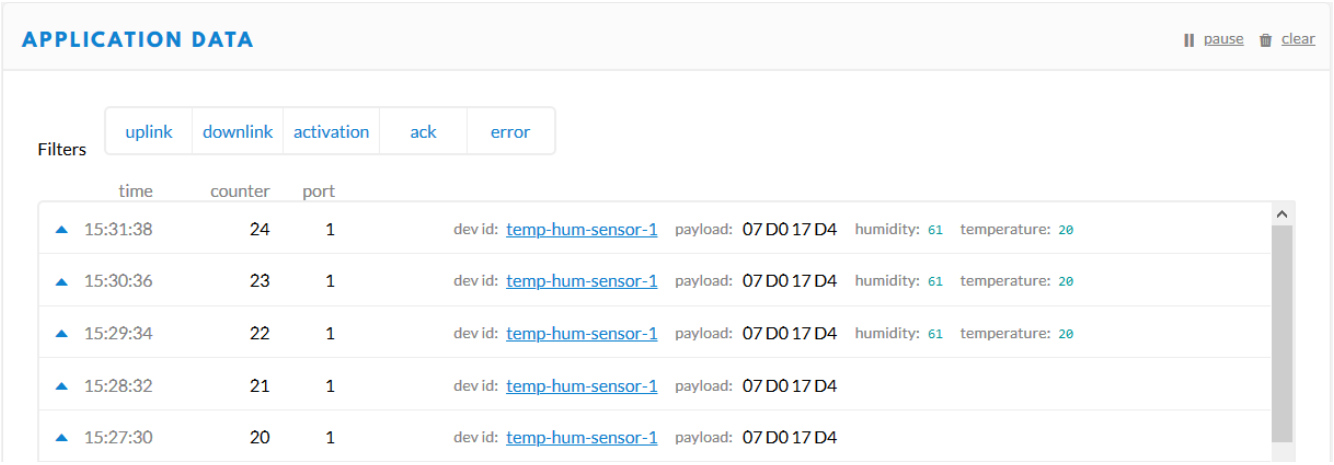
```
function Decoder(bytes, port) {
  // Decode an uplink message from a buffer
  // (array) of bytes to an object of fields.
  var decoded = {};

  decoded.temperature = parseFloat(bytes[0]*256+bytes[1])/100;
  decoded.humidity    = parseFloat(bytes[2]*256+bytes[3])/100;

  // if (port === 1) decoded.led = bytes[0];

  return decoded;
}
```

- Observez les données dans la console TTN :



6. Serveur d'application

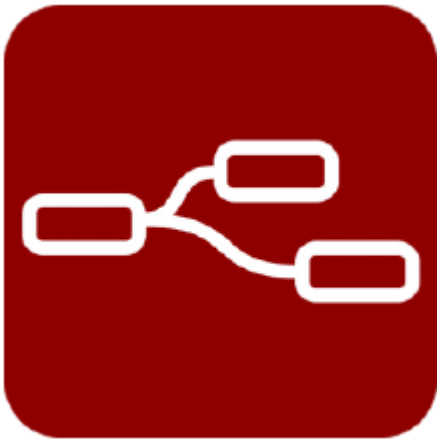
`The Things Network` diffuse les données des objets connectés via le protocole **MQTT** avec les éléments de configuration suivant :

- Host: `<Region>.thethings.network`, where `<Region>` is last part of the handler you registered your application to, e.g. `eu`.
- Port: `1883`
- Username: `Application ID`
- Password: `Application Access Key`
- Topic: `<AppID>/devices/<DevID>/up`

Références de l'API MQTT de TTN

De nombreuses solutions sont alors possibles pour la mise en place d'un serveur d'application :

- Application MQTT client (**HiveMQ**, **MQTT cli**, **mqtt-spy**, ...)



- **Node Red** : Node-RED est un outil puissant pour construire des applications de l'Internet des Objets en mettant l'accent sur la simplification de la programmation qui se fait grâce à des blocs de code prédéfinis, appelés «**nodes**» pour effectuer des tâches. Il utilise une approche de programmation visuelle qui permet aux développeurs de connecter les blocs de code ensemble. Les noeuds connectés, généralement une combinaison de noeuds d'entrée, de noeuds de traitement et de noeuds de sortie, lorsqu'ils sont câblés ensemble, constituent un «**flow**». Le dashboard ainsi réalisé est accessible via un navigateur depuis n'importe quel périphérique du réseau.

- Une application **Qt** utilisant le module `QtMQTT`.

- Un service d'intégration comme par exemple `Cayenne myDevices`. Cette solution est sans doute la plus rapide à mettre en oeuvre. Elle fait appel à un service extérieur qui se connecte au broker MQTT de **The Things Network** et interprète les données envoyées par l'objet pour établir automatiquement un dashboard. Il faut cependant que les données soient correctement formatées au format `Cayenne LPP` (*Cayenne Low Power Payload*).

6.1. Ajouter une intégration Cayenne myDevices

Pour assurer l'interprétation des données par Cayenne myDevices, il faut les formatées depuis l'envoie de l'objet. Elles doivent se conformer au formatage `Cayenne LPP` :

Formatage des données Cayenne LPP

Uplink Payload Structure

1 Byte	1 Byte	N Bytes	1 Byte	1 Byte	M Bytes	...
Data1 Ch.	Data1 Type	Data1	Data2 Ch.	Data2 Type	Data2	...

Figure 11. Trame Cayenne LPP

avec le codage suivant pour `Data Type` et le nombre d'octets à fournir par type de donnée :

Type	IPSO	LPP	Hex	Data Size	Data Resolution per bit
Digital Input	3200	0	0	1	1
Digital Output	3201	1	1	1	1
Analog Input	3202	2	2	2	0.01 Signed
Analog Output	3203	3	3	2	0.01 Signed
Illuminance Sensor	3301	101	65	2	1 Lux Unsigned MSB
Presence Sensor	3302	102	66	1	1
Temperature Sensor	3303	103	67	2	0.1 °C Signed MSB
Humidity Sensor	3304	104	68	1	0.5 % Unsigned
Accelerometer	3313	113	71	6	0.001 G Signed MSB per axis
Barometer	3315	115	73	2	0.1 hPa Unsigned MSB
Gyrometer	3334	134	86	6	0.01 °/s Signed MSB per axis
GPS Location	3336	136	88	9	Latitude : 0.0001 ° Signed MSB
					Longitude : 0.0001 ° Signed MSB
					Altitude : 0.01 meter Signed MSB

Figure 12. Type de données Cayenne LPP

Il faut donc modifier notre programme Arduino pour s'y conformer.

Choisissons un numéro de canal (*chanel*) pour la température et pour l'humidité et formatons notre payload :

- **Température** :
  - `Chanel =0x01`,
  - `Type = 0x67`,
  - data sur 2 octets,
  - précision au dixième
- **Humidité** :
  - `Chanel = 0x02`,
  - `Type = 0x68`,
  - data sur 1 octet,
  - précision au 1/2

```
void loop()
{
  debugSerial.println("-- LOOP");

  uint8_t humidity = dht.readHumidity(false) * 2;

  // false: Celsius (default)
  // true: Farenheit
  uint16_t temperature = dht.readTemperature(false) * 10;

  byte payload[7];

  payload[0] = 0x01;
  payload[1] = 0x67;
  payload[2] = highByte(temperature);
  payload[3] = lowByte(temperature);
  payload[4] = 0x02;
  payload[5] = 0x68;
  payload[6] = humidity;

  debugSerial.print("Temperature: ");
  debugSerial.println(temperature);
  debugSerial.print("Humidity: ");
  debugSerial.println(humidity);

  ttn.sendBytes(payload, sizeof(payload));

  delay(60000);
}
```

- Modifiez le code comme ci-dessus.
- Compilez et téléversez.
- Observez le résultat dans la console TTN après avoir appliqué le `Payload Formats : Cayenne LPP` à l'application.

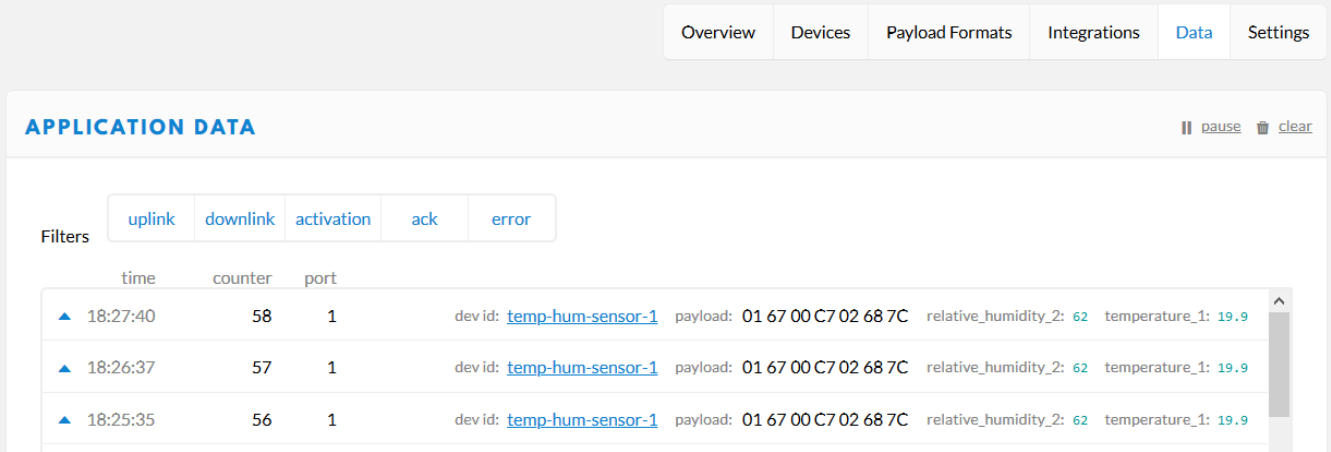


Figure 13. Observation des données conforme au format Cayelle LPP

- Afin de permettre à `Cayenne myDevices` l'accès au données de notre application depuis les serveurs de `The Things Network`, il faut créer une intégration :

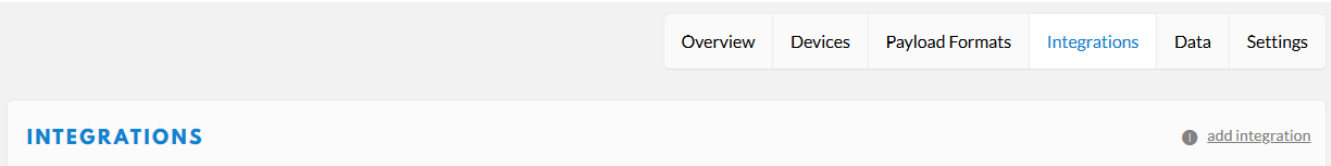


Figure 14. Créer une intégration

- Choisissez `myDevices`

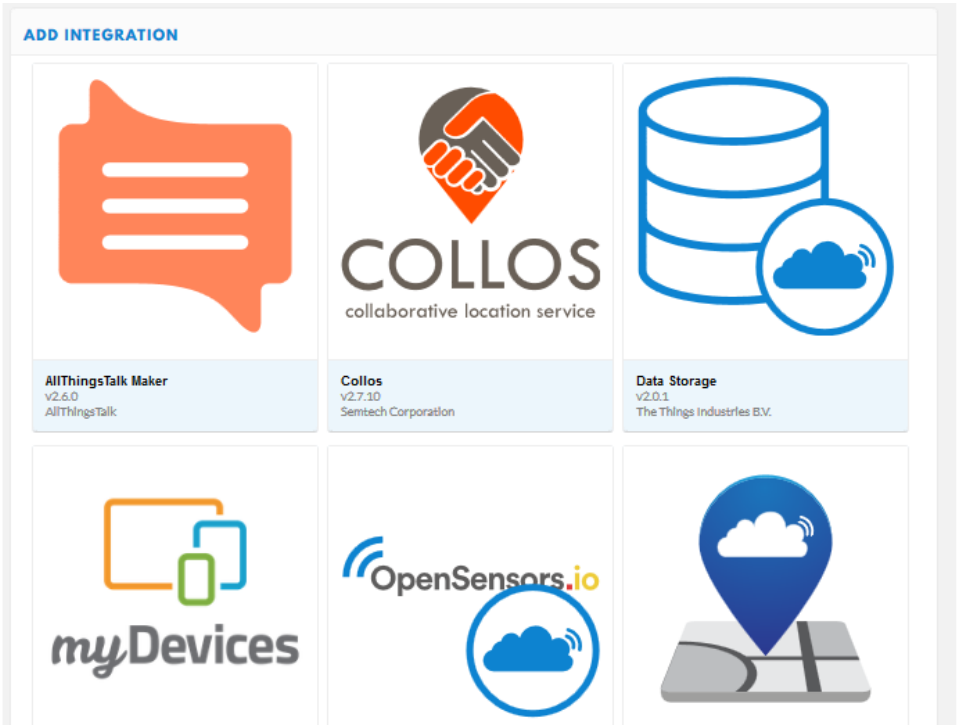
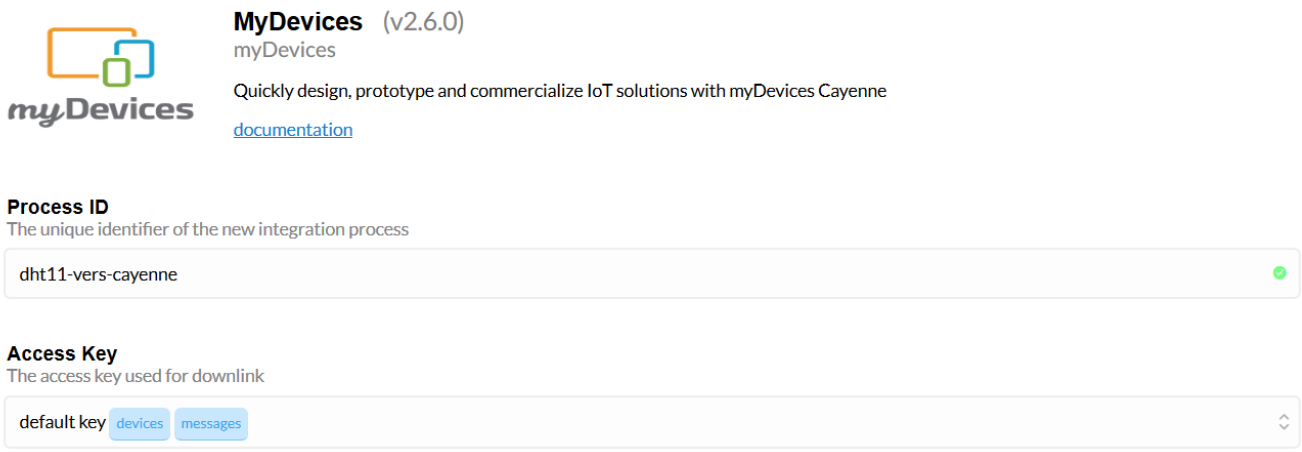


Figure 15. Créer une intégration à Cayenne myDevices

Ajouter une intégration `Cayenne myDevice` consiste à lui donner un identifiant unique (`Process ID`) et à déclarer les clés d'accès aux données de l'application.

- Choisissez `default key` pour faire correspondre automatiquement les clés définies dans l'application et dans les `devices`.
- Enregistrez en cliquant sur le bouton `Add integration`.



6.2. Ajout de l'objet connecté dans Cayenne myDevice

Après avoir ajouter l'intégration depuis la console TTN, `Cayenne myDevices` sera autorisé à accéder aux données d'un objet de l'application si dans sa configuration, il utilise son `DevEUI`.

Vous devez posséder un compte **Cayenne myDevices** pour pouvoir y intégrer votre objet connecté.

Créer son compte Cayenne myDevices

- Connectez vous à votre compte `Cayenne myDevices`
- Cliquez sur l'icone LoRa :

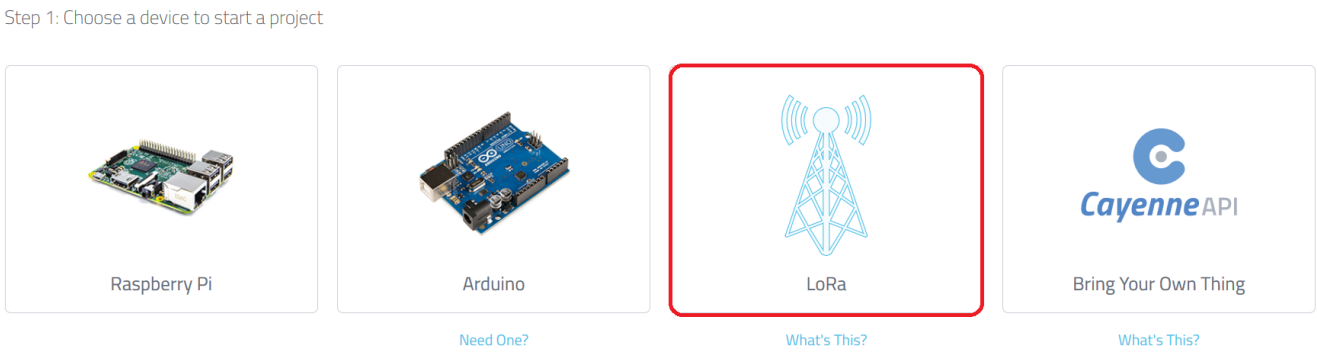


Figure 16. Ajouter un objet connecté dans Cayenne myDevices

- Choisissez le réseau **The Things Network** :

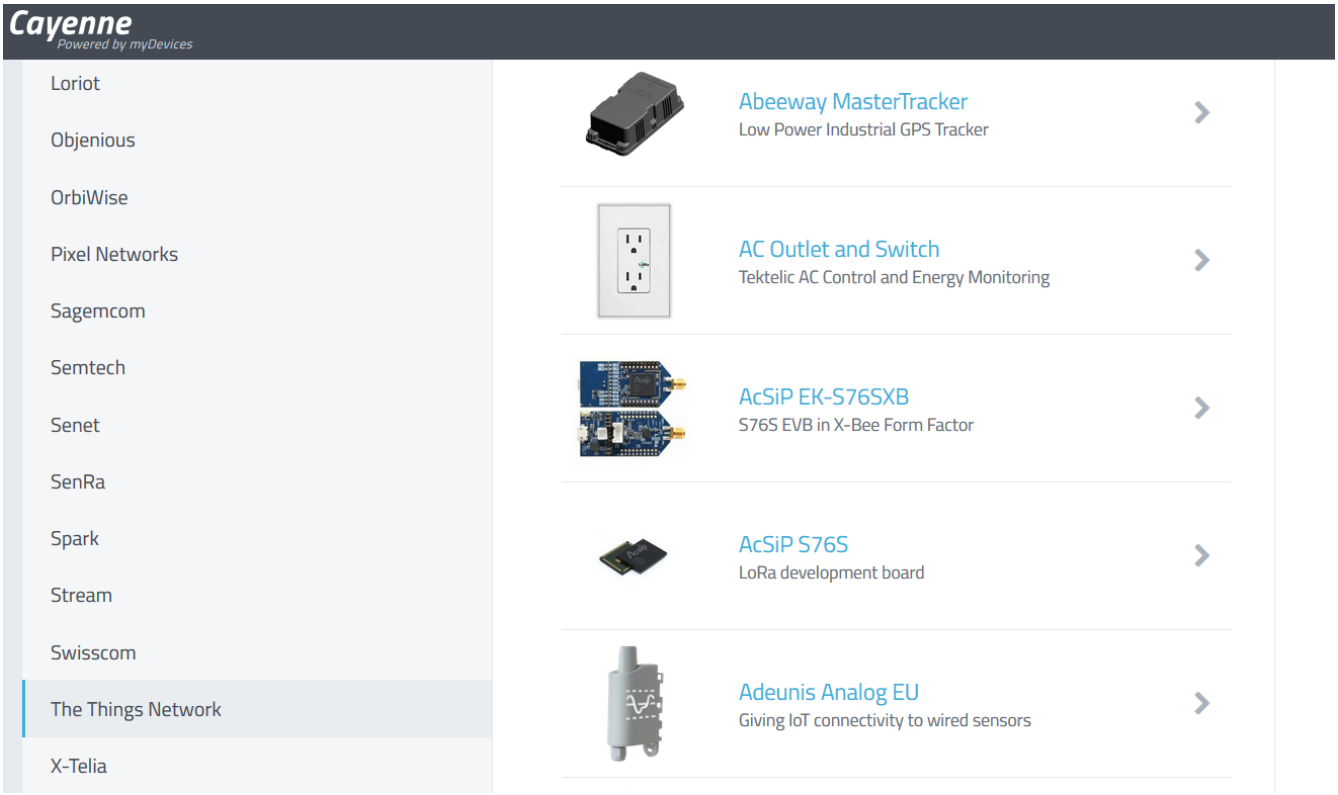


Figure 17. Choisir le réseau

- Choisissez un objet de type `Cayenne LPP` (Low Power Payload)

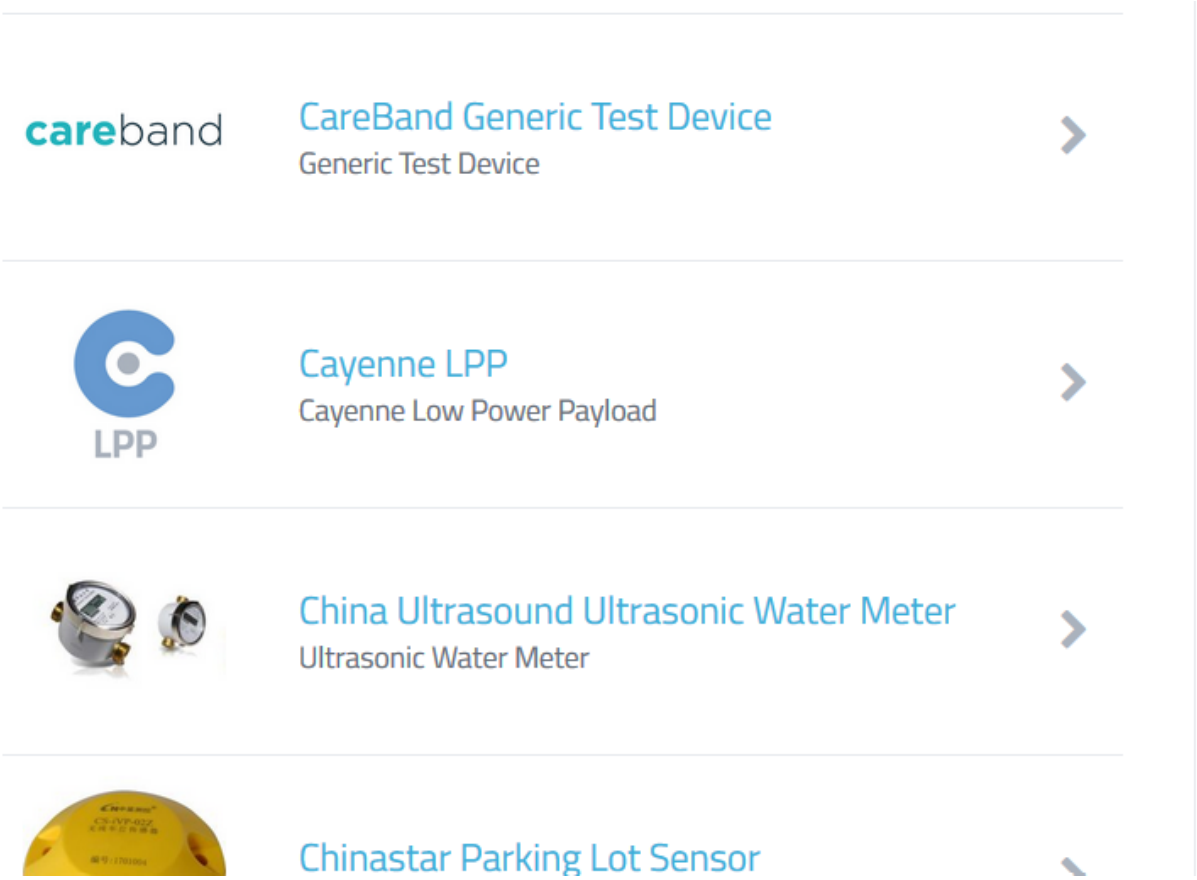
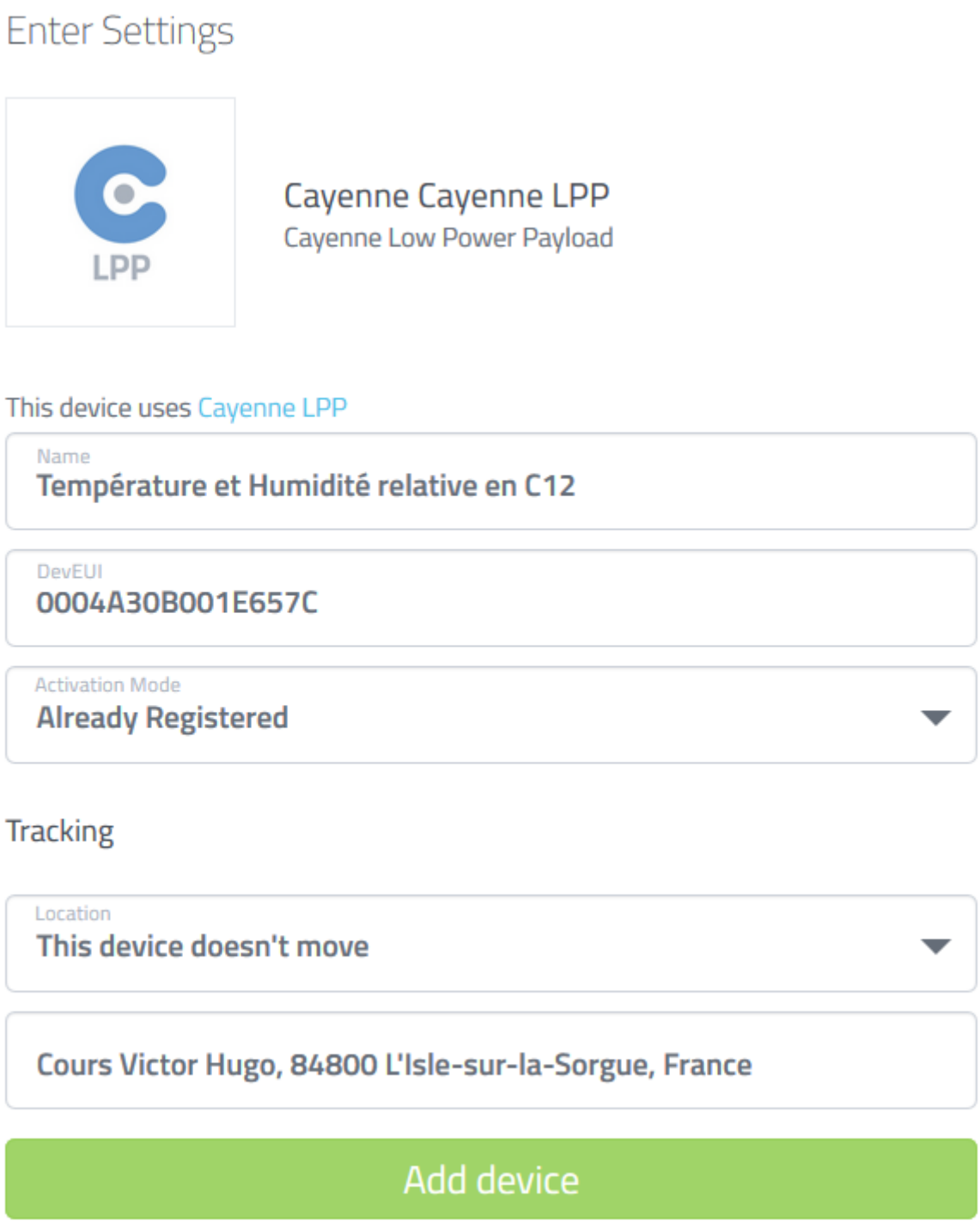
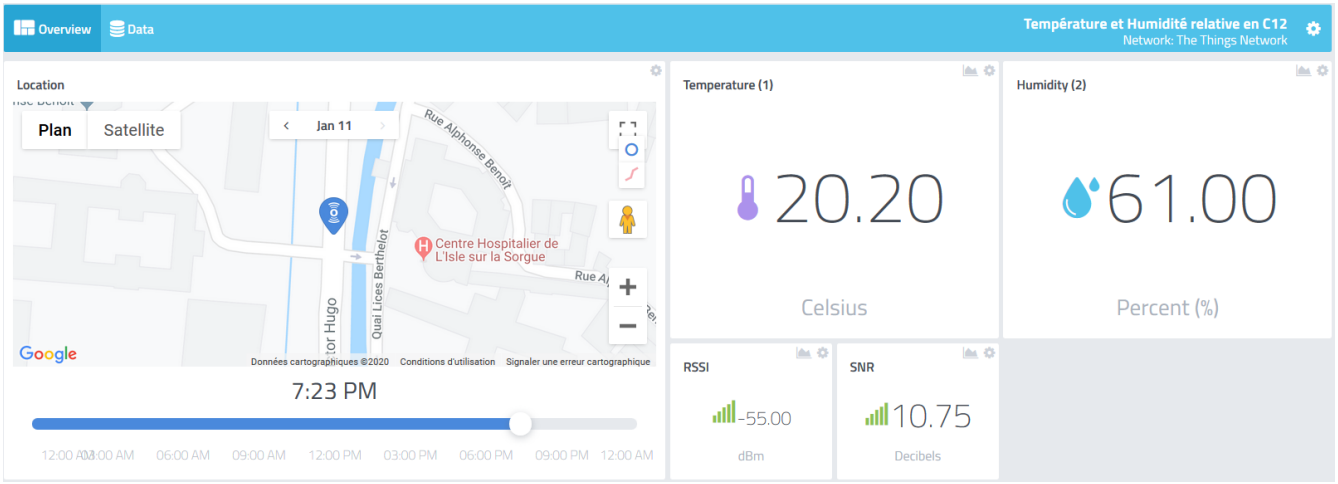


Figure 18. Choisir l'objet à ajouter

- Configurez `Cayenne myDevices` pour accéder aux données de l'objet. Renseignez le `DevEUI` et la localisation de l'objet.



- Cliquez sur `Add device`



Les widgets sont déplaçables, redimensionnables, personnalisables, … Amusez vous !

Vous pouvez également accéder aux données brutes en cliquant sur l'icône `Data` dans le bandeau bleu.

7. Conclusion

L'utilisation de `The Things Network` combiné à `Cayenne myDevices` fournie un environnement très pratique et puissant pour mettre à disposition des données issues de capteurs ou pour piloter un système à distance en s'appuyant sur un réseau IOT LoRa longue distance et faible énergie.

Le format `Cayenne LPP` peut représenter une certaine difficulté pour la mise en œuvre du programme dans l'objet. Cependant, il existe une librairie Arduino `CayenneLPP` qui s'occupe de tout :

`Croquis→Inclure une bibliothèque→Gérer les bibliothèques→Recherchez CayenneLPP`