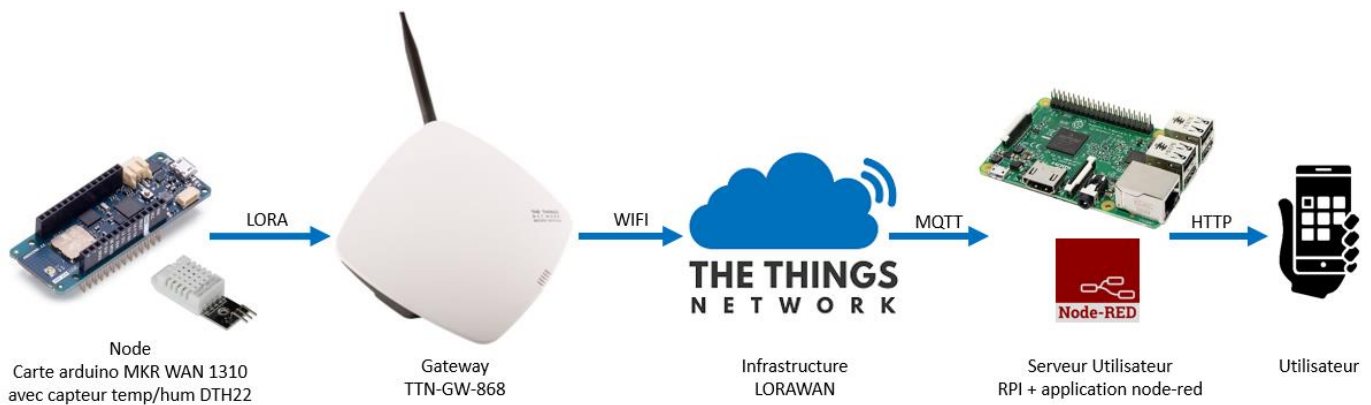


Table des matières

L'Objectif.....	2
Etape 1 : Installation de la gateway TTN.....	2
Etape 2 : Création d'une application sur son compte TTN	5
Etape 3 : Récupérer l'identifiant de votre module MKR WAN 1310.....	6
Etape 4 : Associer votre node MKRWAN 1310 à l'application TTN	8
Etape 5 : Programmer votre node MKRWAN 1310.....	10
Etape 6 : Activation du broker MQTT de votre application TTN.....	12
Etape 7 : Configuration du serveur utilisateur (RPI+NodeRed)	12
A - Configurer node-red pour créer une IHM	13
B - Créer le flow node-red	13
1- Se connecter au broker MQTT de l'application TTN	14
2- Lire les données au format JSON	14
3- Extraction de l'information utile	15
4- Décoder la mesure transmise au format base64.....	15
5- Extraire la température et l'humidité de la chaîne de caractères.....	15
6- Séparer la température de l'humidité.....	16
7- Construction de l'IHM	16
CONCLUSION	17

L'Objectif

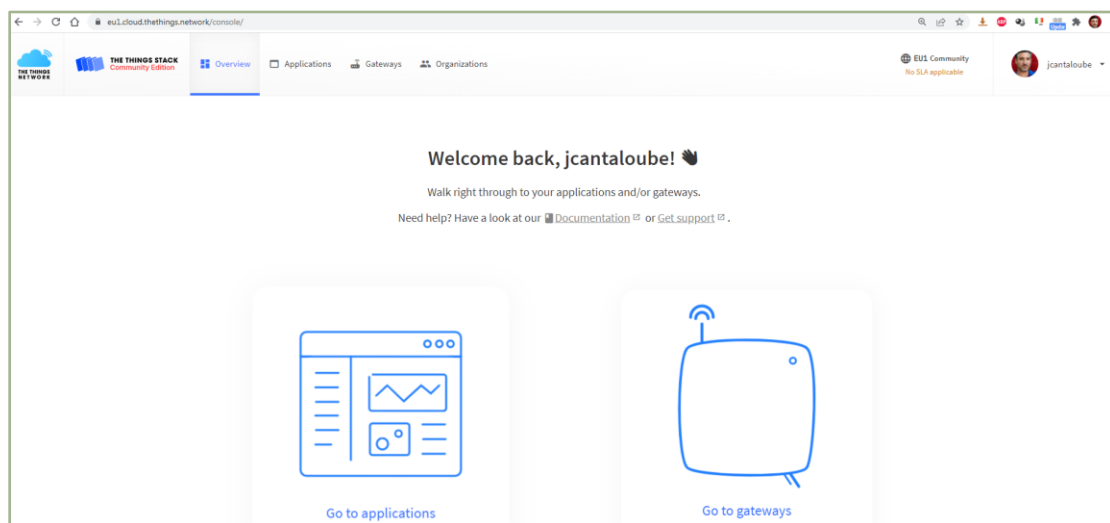
L'objectif est de réaliser l'infrastructure suivante :



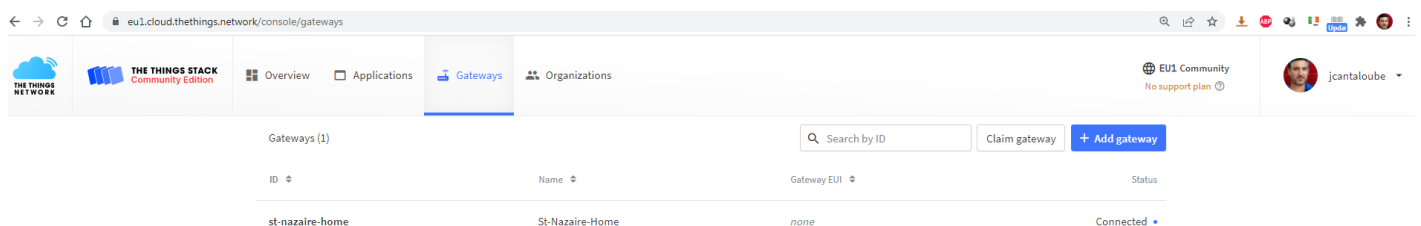
Elle repose sur l'utilisation d'un node MKRWAN 1310 de chez Arduino et du développement d'une application node-red sur Raspberry PI pour récupérer, stocker et mettre en forme les données.

Etape 1 : Installation de la gateway TTN

Une fois votre compte TTN créé, accédez au menu **console** (<https://eu1.cloud.thethings.network/console/>) :



En cliquant sur l'icône **Go to gateways**, vous visualisez toutes vos gateways déployées et vous pouvez en déclarer des nouvelles.



Pour ajouter une gateway, cliquez sur le bouton **+ Add gateway** et complétez les différents champs :

General settings

Gateway ID [?] *

Gateway EUI [?]

Gateway name [?]

Gateway description [?]

Optional gateway description; can also be used to save notes about the gateway

Gateway Server address

The address of the Gateway Server to connect to

LoRaWAN options

Frequency plan [?] *

Schedule downlink late [?]

☐ Enabled

Enable server-side buffer of downlink messages

Enforce duty cycle [?]

☒ Enabled

Recommended for all gateways in order to respect spectrum regulations

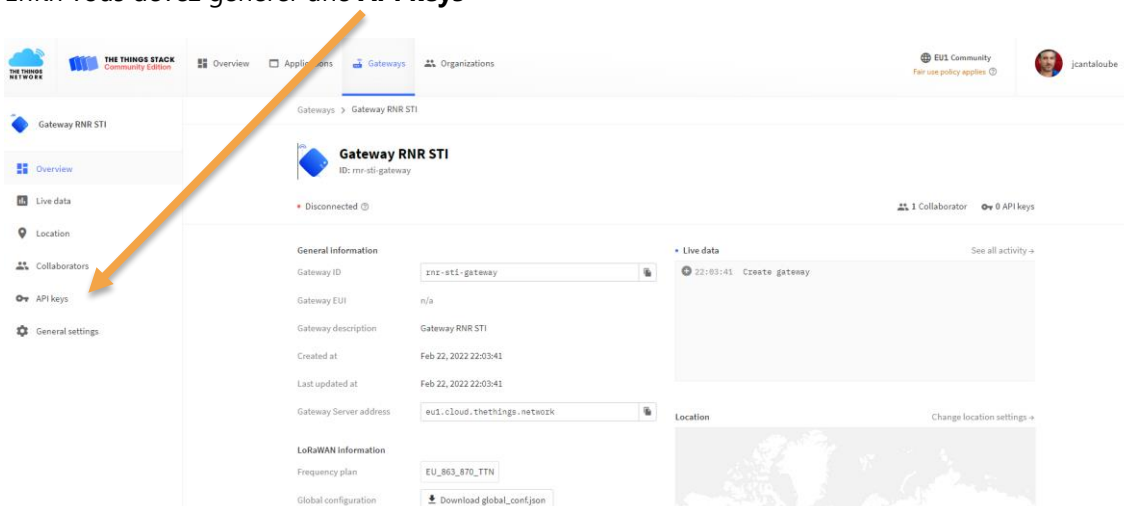
Schedule any time delay [?] *

Configure gateway delay (minimum: 130ms, default: 530ms)

Donnez un identifiant, un nom à votre gateway et sélectionnez la bande de fréquence. Les autres champs gardent leurs valeurs par défaut. Puis validez votre configuration.

Vous pouvez ensuite configurer la localisation de votre gateway (ces informations seront utiles pour la communauté TTN)

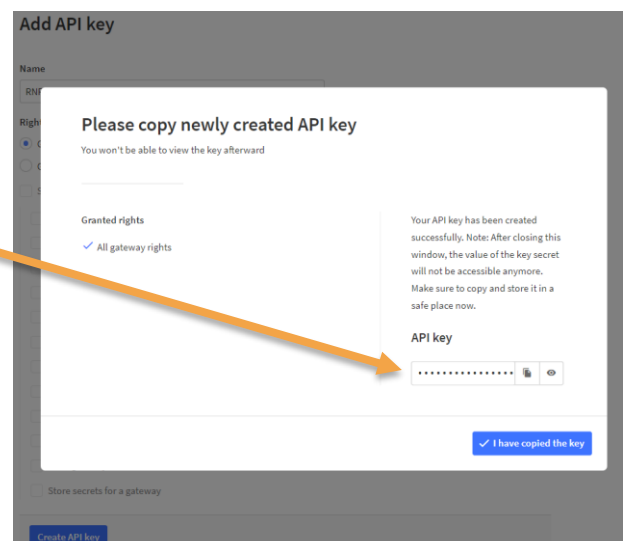
Enfin vous devez générer une **API keys**



Vous pouvez alors ajouter une API key et lui donner un nom.

Après avoir validé votre API key, la fenêtre suivante apparaît et il est absolument nécessaire de **copier le numéro de l'API key** généré.

Cette clé vous sera nécessaire pour finir la configuration de votre gateway.



Au niveau de la gateway, après l'avoir branchée au secteur, retirez le cache blanc et appuyez pendant **5s** sur le **bouton** mode pour effectuer un **Reset** de la gateway :



Une fois la gateway réinitialisée, il faut s'y connecter en wifi via le SSID « **Things Gateway – xxxx** » (mot de passe : thethings).

Ensuite, il faut se rendre sur la page web de configuration de la gateway via votre navigateur à l'adresse

<http://192.168.84.1/>

La page suivante s'affiche, il faut alors compléter les différents champs :

Identifiant donné sur votre compte TTN

wifi disponible à proximité de la gateway pour accéder à internet

<https://eu1.cloud.thethings.network>

API key copiée précédemment

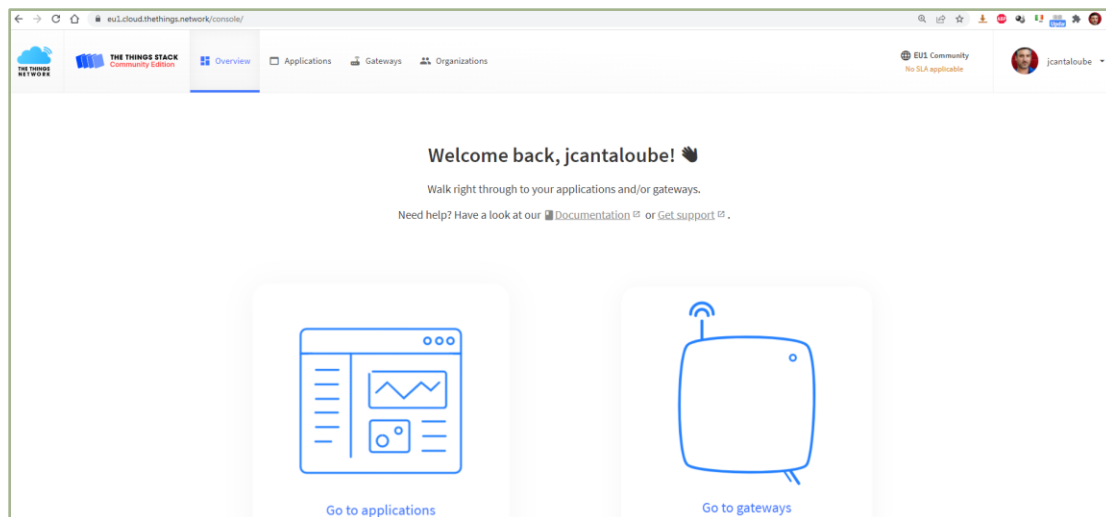
Une fois les paramètres sauvegardés, la gateway redémarre et se connecte au serveur TTN.

Depuis votre console TTN dans la partie Gateway (<https://eu1.cloud.thethings.network/console/gateways>), vous devez constater que votre gateway est bien connectée au réseau TTN :

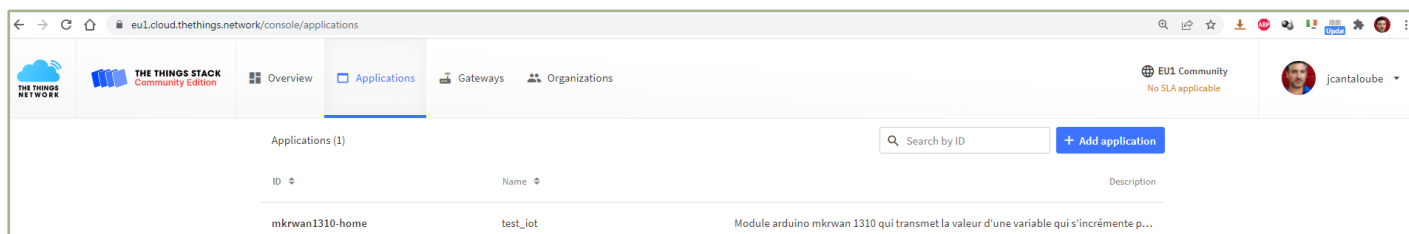
ID	Name	Gateway EUI	Status
XXXXXXXXXX	XXXXXXXXXX	none	Connected

Etape 2 : Création d'une application sur son compte TTN

Depuis votre console TTN (<https://eu1.cloud.thethings.network/console/>) :



En cliquant sur l'icône **Go to applications**, vous accédez alors à toutes vos applications et vous pouvez y créer des nouvelles.



Ici, le terme application est utilisé pour définir le serveur d'application dont le rôle est de centraliser les données issues d'un ou plusieurs nodes.

Pour ajouter une application, cliquez sur le bouton **+ Add application** et complétez les différents champs :

Add application

Application ID *

Application name

Description

Optional application description; can also be used to save notes about the application

Create application

Le champ « Application ID » identifie l'application (utilisez des minuscules, pas d'espace). Ce champ est non modifiable une fois l'application créée.

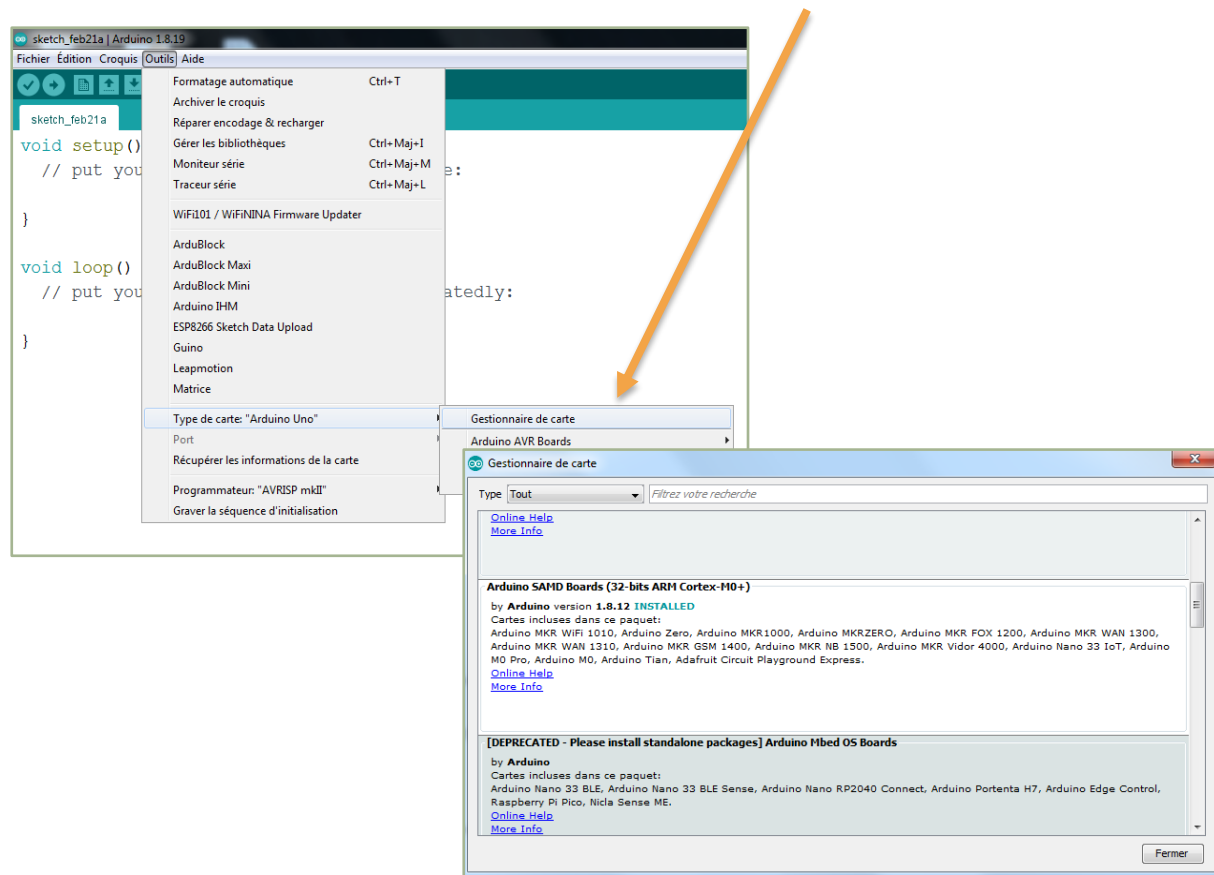
Les autres champs sont facultatifs et modifiables ultérieurement. Cependant, il est préférable de les compléter pour une meilleure traçabilité.

Une fois l'application créée, il faut ajouter le ou les nodes que vous souhaitez associer. Pour cela vous devez récupérer leur identifiant (device EUI).

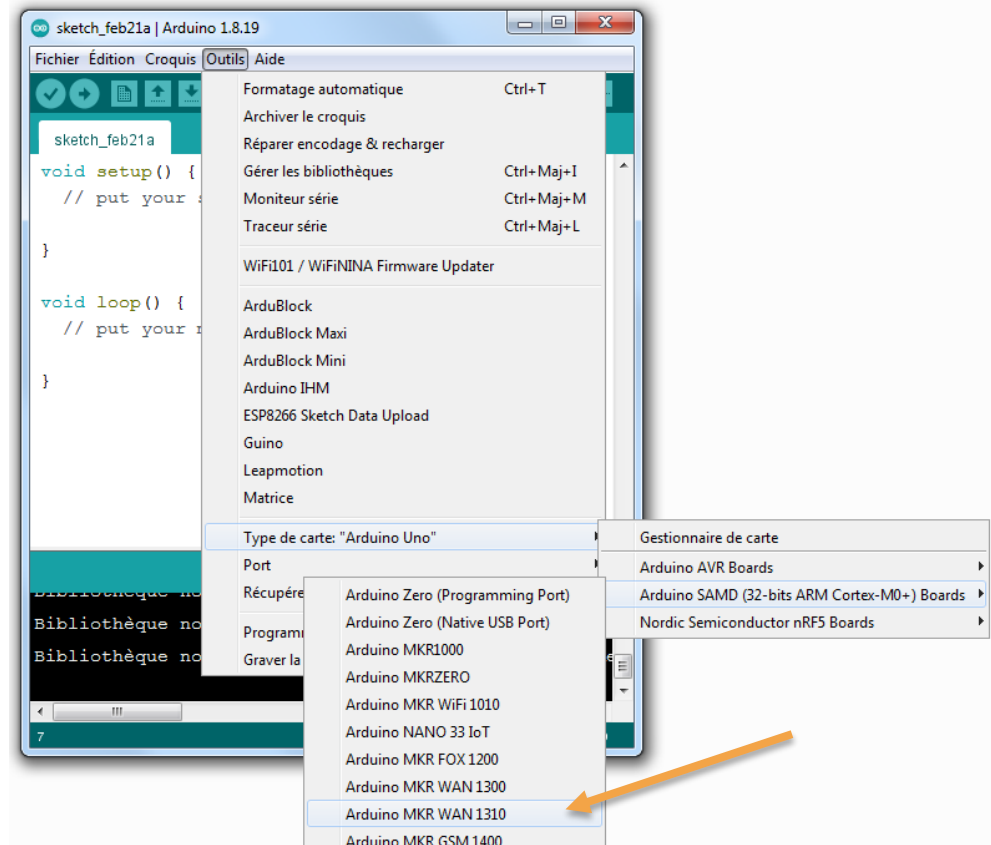
Etape 3 : Récupérer l'identifiant de votre module MKR WAN 1310

L'identifiant du node lora est appelé **device EUI**, il s'agit d'un identifiant unique créé par le constructeur.

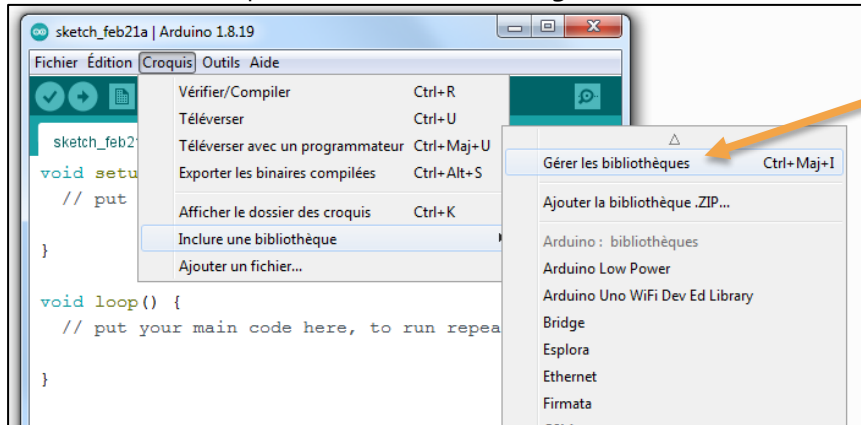
Mais avant tout, il faut installer le driver de la carte mkr wan dans l'IDE Arduino en sélectionnant le package « Arduino SAMD Boards (32-bits ARM Cortex-M0+) » via le **gestionnaire de carte**.



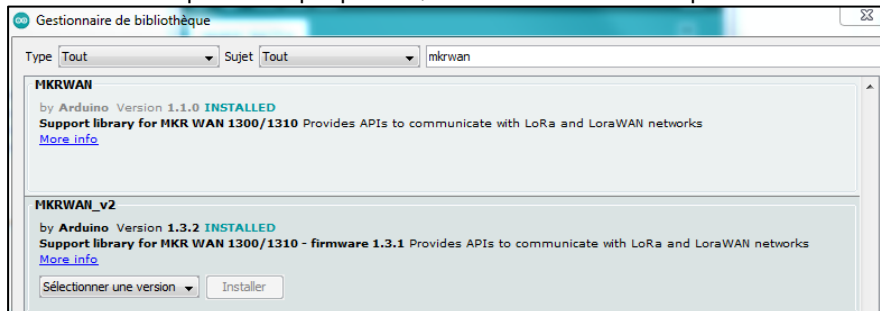
Lorsque le driver est installé, vous pouvez sélectionner la carte et le port COM attribué :



Installer la bibliothèque **MKRWAN** à l'aide du **gestionnaire de bibliothèques** :



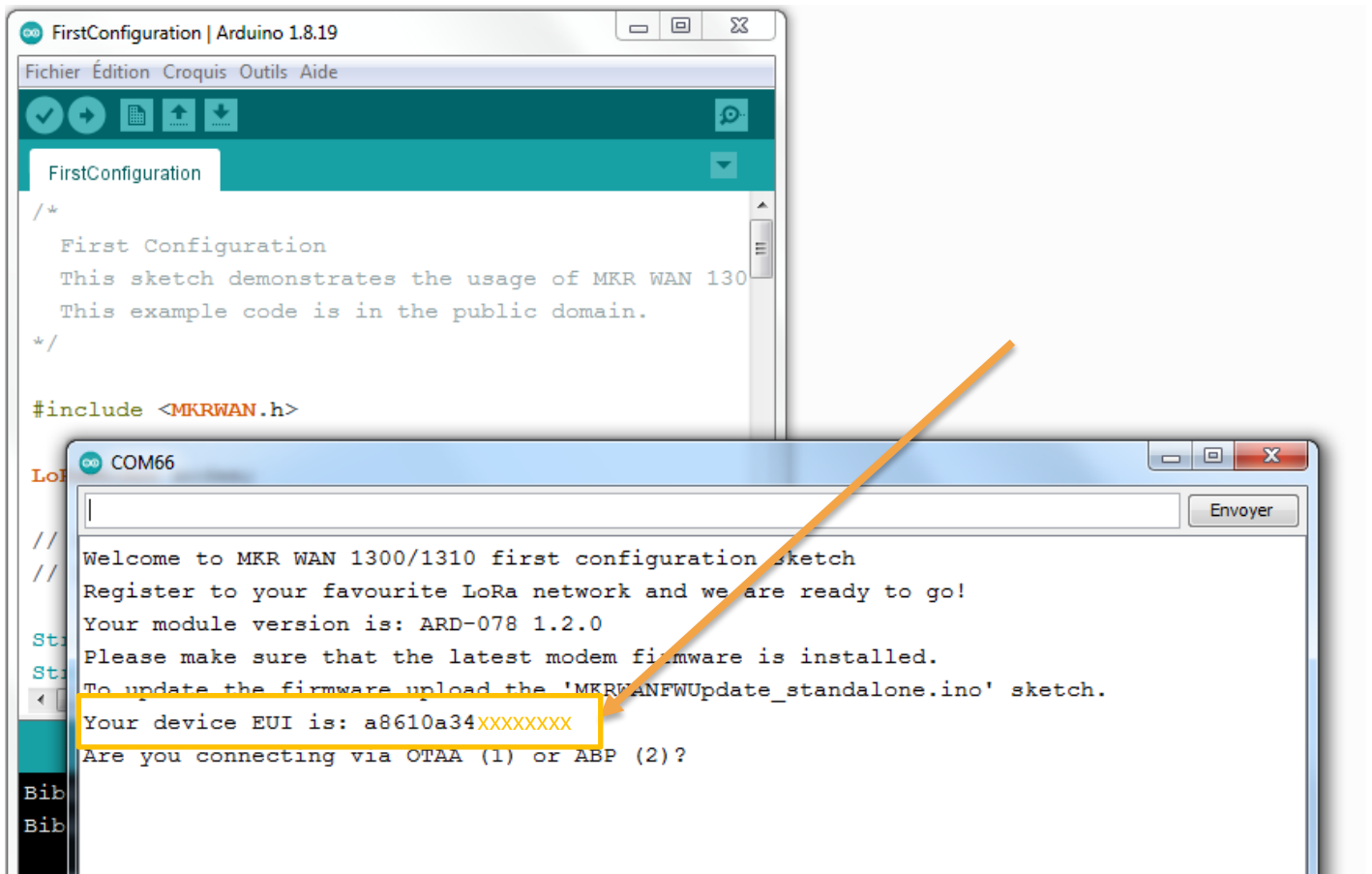
Deux bibliothèques sont proposées, nous allons utiliser la première :



Il est important que votre carte ait le bon **firmware** installé. La mise à jour s'effectue en exécutant le code **Fichier > Exemples > MKR WAN > MKR WANFWUpdate_standalone** (ne pas prendre en compte les messages d'erreurs éventuels)

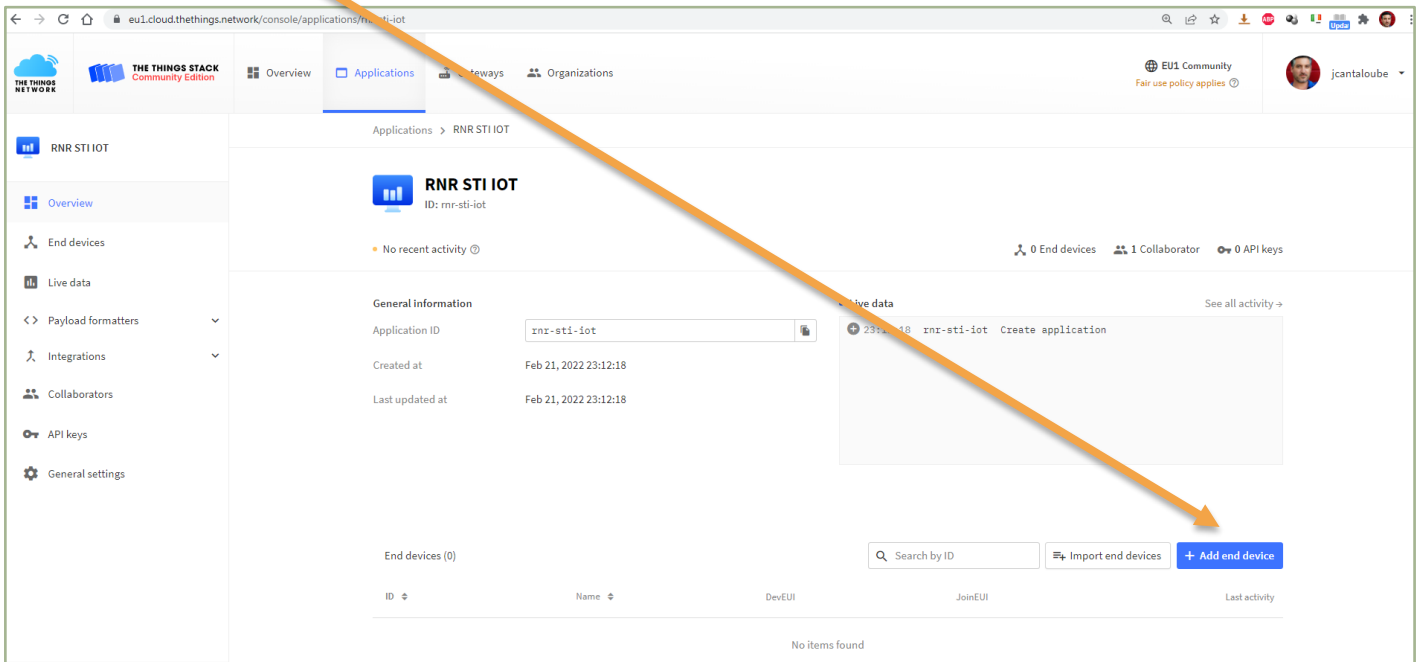
Afin de récupérer le **device EUI**, exécutez l'exemple **Fichier > Exemples > MKR WAN > FirstConfiguration**.

Ouvrez le moniteur série de l'IDE Arduino et relevez le **device EUI**



Etape 4 : Associer votre node MKRWAN 1310 à l'application TTN

Depuis votre console TTN, au sein de votre application créée précédemment, ajoutez un node en cliquant sur le bouton **+Add end devices**




Puis complétez les champs suivants afin d'identifier le type de node :

1. Select the end device

Brand [?] * Model [?] * Hardware Ver. [?] * Firmware Ver. [?] * Profile (Region) ^{*}

Arduino SA | v Model: Arduino MKR WAN 1... | v Hardware Ver.: 1.0 | v Firmware Ver.: 1.2.0 | v Profile (Region): EU_863_870 | v



Arduino MKR WAN 1310

MAC V1.0.2, PHY V1.0.2 REV A, Over the air activation (OTAA), Class A

The Arduino MKR WAN 1310 is a development board that provides a practical and cost-effective solution to add LoRaWAN® connectivity for projects requiring long-range, low-power wireless communication. Sensors and actuators can be connected to the board through the analog, digital, UART, SPI, and I2C pins. The MKR WAN 1310 comes complete with an ATECC508 secure element, a battery charger, 2MByte SPI Flash, and power consumption as low as 104 uA.

[Product website](#)

Remarque : Le choix du firmware doit correspondre au numéro relevé dans le moniteur série précédemment

Puis, sélectionnez la bande de fréquence (celle recommandée), saisissez le devEUI relevé dans le moniteur série juste avant, générer une AppKey et mettre AppEUI à zéro (vous pouvez aussi mettre votre propre AppEUI)

2. Enter registration data

Frequency plan ⓘ *

Europe 863-870 MHz (SF9 for RX2 - recommended) | ▾

AppEUI ⓘ *

00 00 00 00 00 00 00 00 | Fill with zeros

DevEUI ⓘ *

A8 61 0A 34 XX XX XX XX | Generate 0/50 used

AppKey ⓘ *

XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX | Generate

End device ID ⓘ *

eui-a8610a34XXXXXXXXXX

This value is automatically prefilled using the DevEUI

After registration

☒ View registered end device

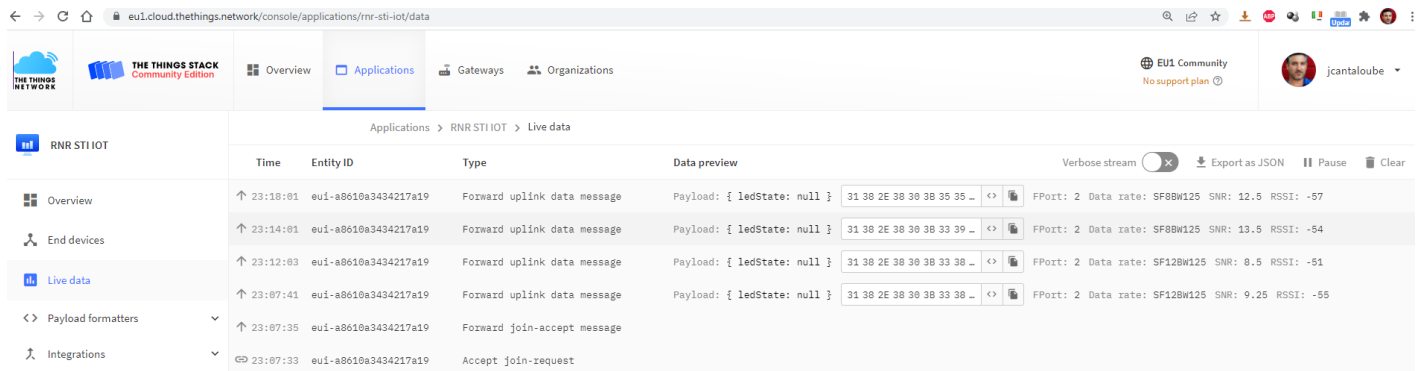
☐ Register another end device of this type

Register end device

Une fois tous les champs complétés, enregistrez votre configuration en cliquant sur le bouton **Register end device**.

Il est important pour la suite, de bien noter votre DevEUI, votre AppEUI et votre AppKey.

Puis vous rendre sur votre application TTN (<https://eu1.cloud.thethings.network/console/applications/rnr-sti-iot/data>) et vérifier que votre application reçoit bien les données :



Time	Entity ID	Type	Data preview
↑ 23:18:01	eu1-a8610a3434217a19	Forward uplink data message	Payload: { ledState: null } 31 38 2E 38 38 38 35 35 _
↑ 23:14:01	eu1-a8610a3434217a19	Forward uplink data message	Payload: { ledState: null } 31 38 2E 38 38 38 33 39 _
↑ 23:12:03	eu1-a8610a3434217a19	Forward uplink data message	Payload: { ledState: null } 31 38 2E 38 38 38 33 38 _
↑ 23:07:41	eu1-a8610a3434217a19	Forward uplink data message	Payload: { ledState: null } 31 38 2E 38 38 38 33 38 _
↑ 23:07:35	eu1-a8610a3434217a19	Forward join-accept message	Payload: { ledState: null } 31 38 2E 38 38 38 33 38 _
⌵ 23:07:33	eu1-a8610a3434217a19	Accept join-request	

En cliquant sur un payload, nous pouvons observer la structure des données sous format json :

```
],
"received_at": "2022-02-22T22:48:02.034311856Z",
"uplink_message": {
  "session_key_id": "AX8jfbTlvX0Ps/2ZWswj+A==",
  "f_port": 2,
  "f_cnt": 18,
  "frm_payload": "MTguNjA7NTkuNzA=",
  "decoded_payload": {
    "ledState": null
  },
  "rx_metadata": [
    {
      "gateway_ids": {
        "gateway_id": "st-nazaire-home"
      },
      "time": "2022-02-22T22:48:02Z",
      "timestamp": 1769626427,
      "rssi": -57,
      "channel_rssi": -57,
      "snr": 8.25,
```

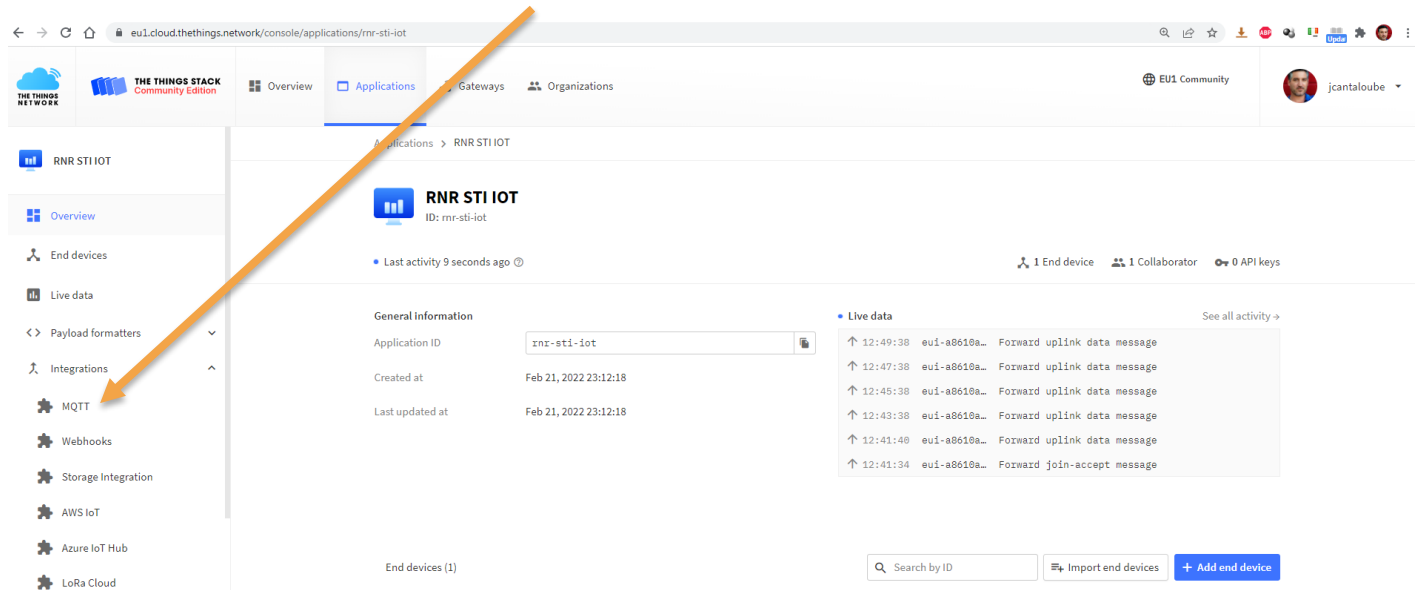
On y retrouve **frm_payload** qui correspond à la donnée émise par notre node. Cette donnée est encodée en base 64 "MTguNjA7NTkuNzA="

Après décodage, on retrouve la chaîne de caractères « 18.60;59.70 » qui correspond à la température suivie du taux d'humidité séparé par le caractère « ; »

Etape 6 : Activation du broker MQTT de votre application TTN

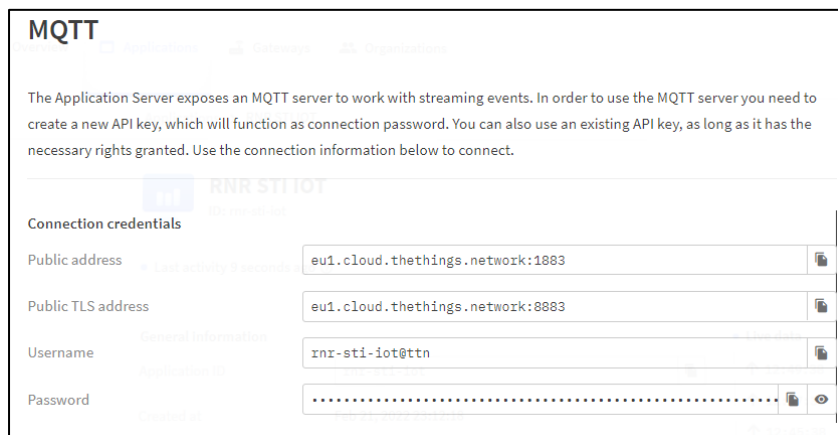
Pour pouvoir venir récupérer les données sur le serveur d'application de TTN, nous allons utiliser le protocole MQTT (voir fiche MQTT).

Mais avant cela, il faut activer le broker MQTT de notre application TTN depuis le menu **Integration/MQTT** :



The screenshot shows the TTN console interface for an application named 'RNR STI IOT'. An orange arrow points to the 'MQTT' option in the left-hand navigation menu under the 'Integrations' section. The main panel displays the application's details, including its ID 'rnr-sti-iot', creation and update timestamps, and a 'Live data' section showing recent uplink messages.

Vous devez alors générer une AppKey qui servira de mot de passe pour accéder au broker à distance (attention à bien conserver la clé)



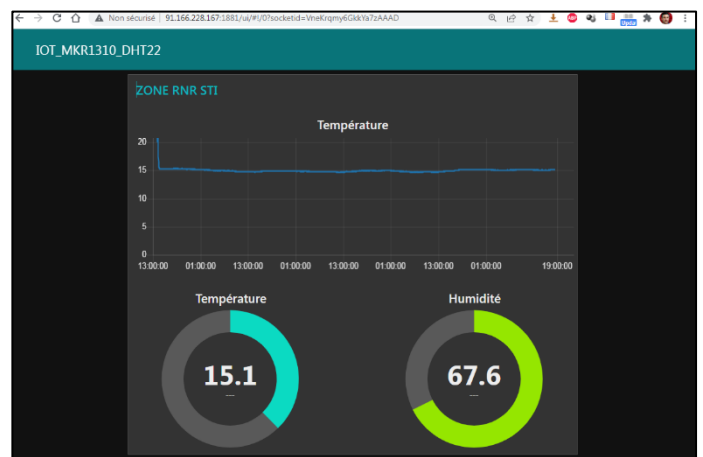
This screenshot shows the 'MQTT' connection credentials page. It provides instructions on how to use the MQTT server and lists the necessary connection information:

- Public address:** eu1.cloud.thethings.network:1883
- Public TLS address:** eu1.cloud.thethings.network:8883
- Username:** rnr-sti-iot@ttn
- Password:** A generated AppKey (represented by dots).

Etape 7 : Configuration du serveur utilisateur (RPI+NodeRed)

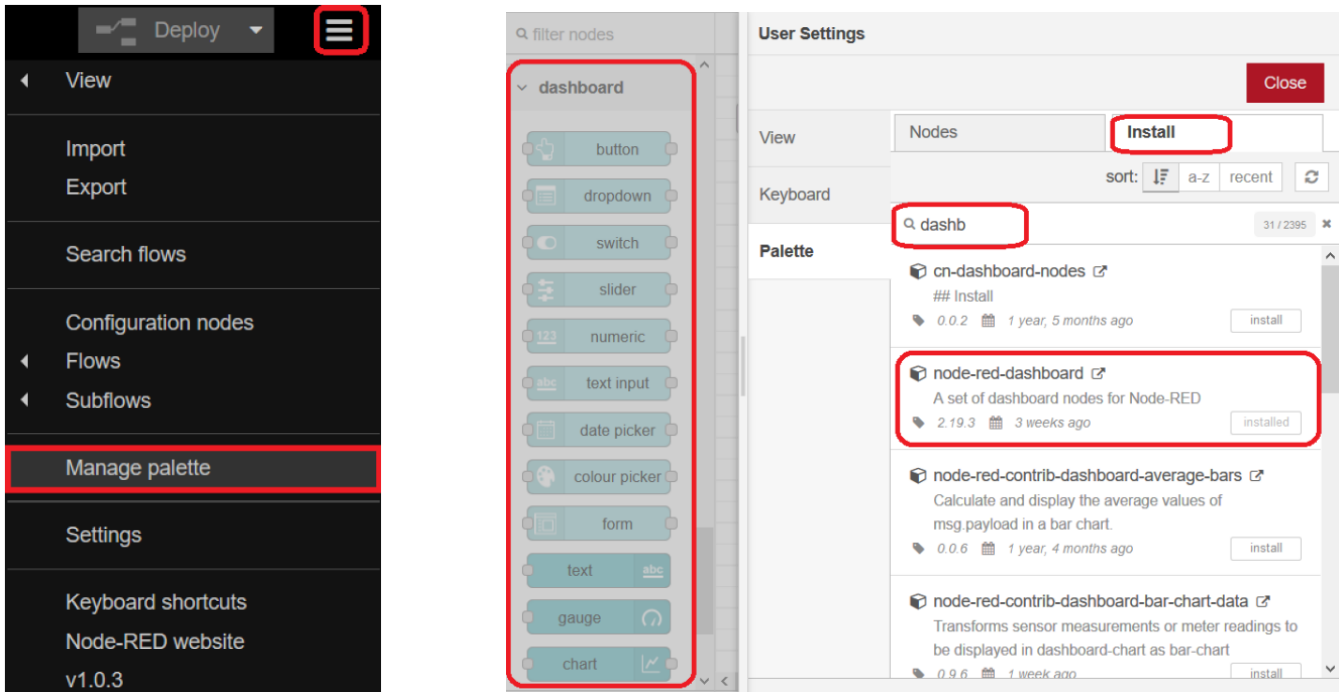
L'objectif est de créer une IHM (jauges + graphique) pour visualiser la température et l'humidité mesurées par notre node.

Il faut d'abord ajouter, à la palette, une nouvelle catégorie de nodes, appelée Dashboard.



A - Configurer node-red pour créer une IHM

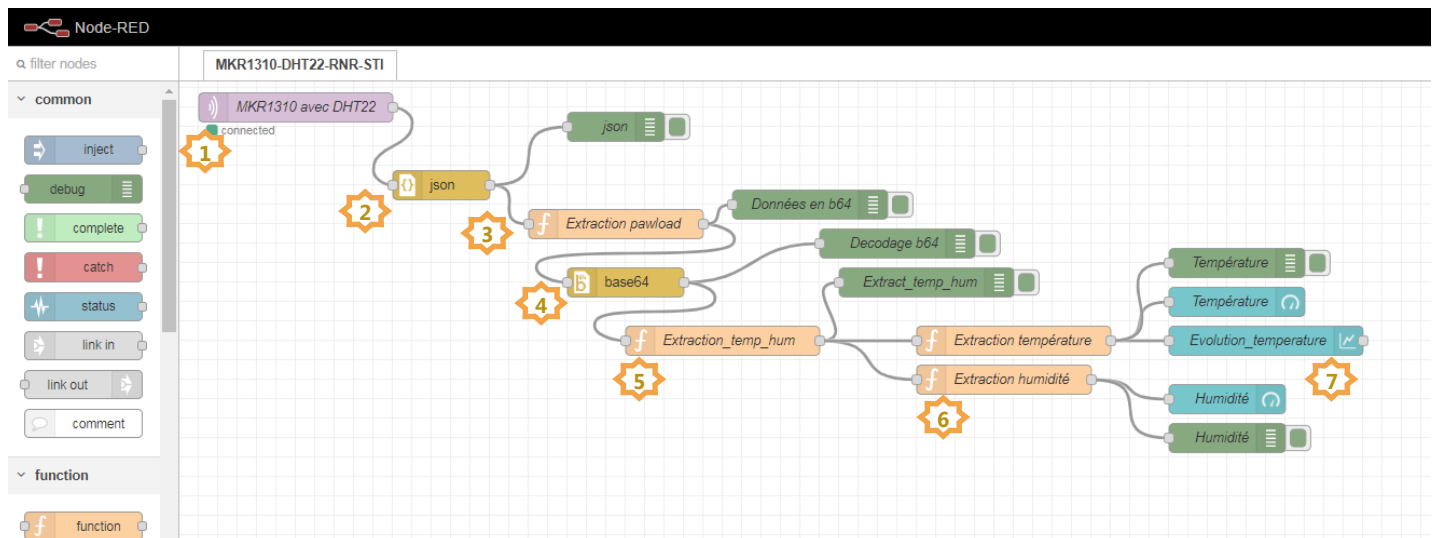
Accéder à Node-red depuis un navigateur et ajouter la palette dashboard via le menu **manage palette** :



Attention : si l'installation de la palette dashboard ne fonctionne pas, il faut surement faire une mise à jour de la RPI (**sudo apt update && sudo apt upgrade**) (mise jour de node.js)

B - Créer le flow node-red

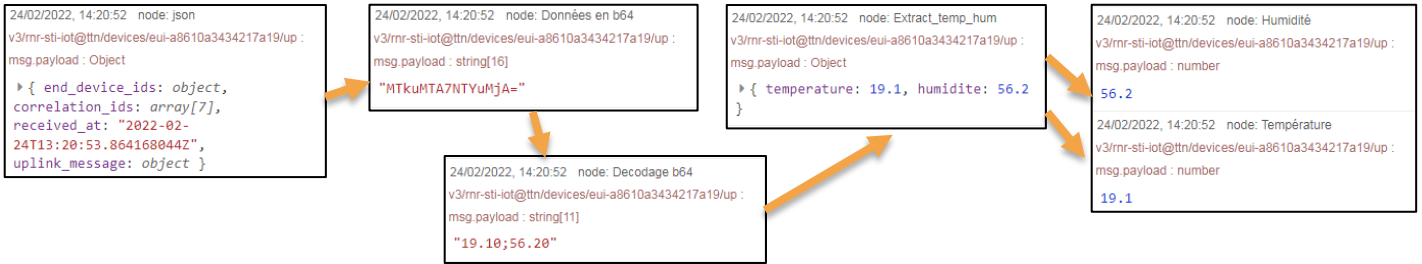
L'objectif est de créer le flow suivant :



Les étapes sont les suivantes :

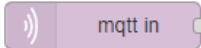
- 1- Se connecter au broker MQTT de l'application TTN.
- 2- Décoder l'information récupérée par MQTT au format JSON.
- 3- Extraire de l'information reçue uniquement la mesure transmise par notre node lora mkr-wan.
- 4- Décoder la donnée au format base64 pour obtenir une chaîne de caractères.
- 5- Dans la chaîne de caractères, extraire la température et l'humidité.
- 6- Séparer la température et l'humidité.
- 7- Utiliser les données pour réaliser l'IHM (IHM composée d'une jauge de température, d'un graphique de température et d'une jauge d'humidité).

Grâce aux nœuds de type debug, nous pouvons observer l'évolution des données (payload) tout au long du flow :



1- Se connecter au broker MQTT de l'application TTN

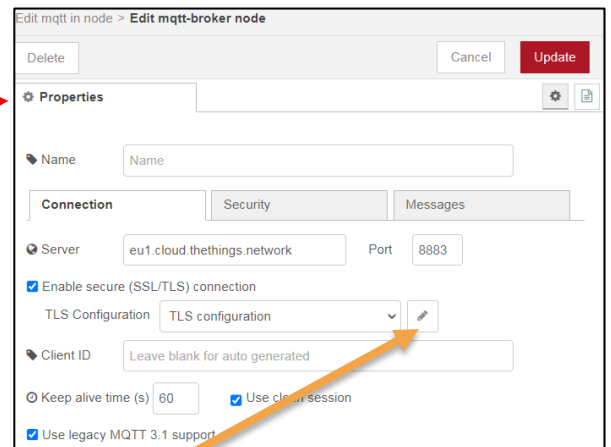
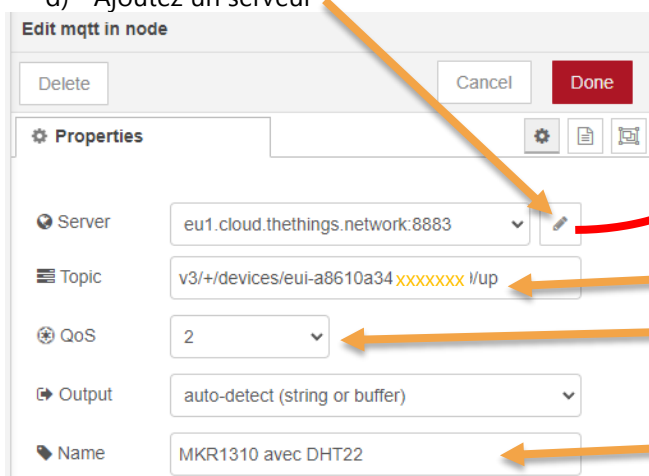
Dans la palette **network**, sélectionnez un nœud de type **mqtt in** :



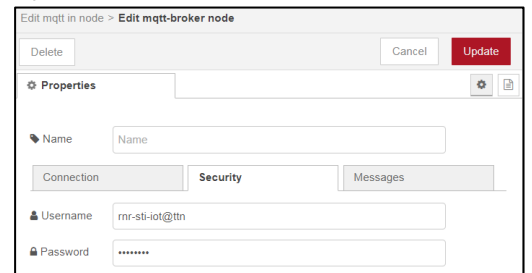
Configurez le nœud :

- Donnez un nom au nœud (exemple : MKR1310 avec DTH22)
- Indiquez le topic auquel l'on souhaite souscrire (v3/+ /devices/eui-a8610a34xxxxxx/up)
- QoS (qualité de service) de niveau 2 (valeur par défaut)
- Ajoutez un serveur

End Device ID donné sur votre compte TTN

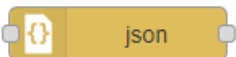


- Au niveau de la fenêtre configuration du serveur, ajoutez le certificat TLS (le fichier est fourni avec ce guide : **mqtt-ca.pem**)
- Indiquez l'url du serveur (eu1.cloud.thethings.network) et le port (8883)
- Cliquez sur l'onglet Security et saisissez le username (qui correspond au nom de votre application TTN) et le password (qui correspond à l'API key)
- Une fois la configuration finie et le flow déployé, le nœud doit indiquer qu'il est bien connecté au broker de TTN



2- Lire les données au format JSON

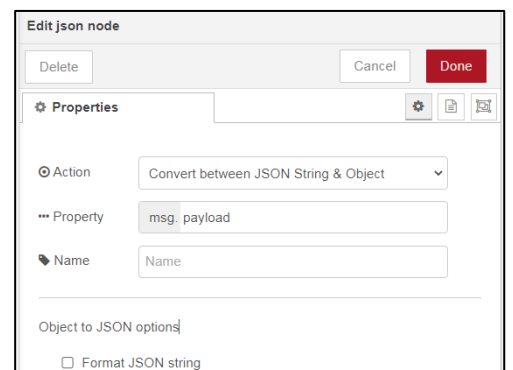
Dans la palette **parser**, sélectionnez un nœud de type **json** :



En sortie de ce nœud json, on observe (grâce au nœud debug placé en sortie) le payload suivant :

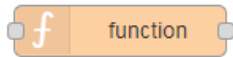
```
24/02/2022, 14:20:52 node: json
v3/mr-sti-iot@ttn/devices/eui-a8610a3434217a19/up :
msg.payload : Object
» { end_device_ids: object,
correlation_ids: array[7],
received_at: \"2022-02-24T13:20:53.864168044Z\",
uplink_message: object }
```

Il s'agit d'un objet json contenant toutes les informations récupérées à chaque uplink de notre node lora mkr-wan (la mesure transmise mais aussi les infos sur la gateway et bien d'autres)



3- Extraction de l'information utile

Dans la palette **function**, sélectionnez un nœud de type **function** :



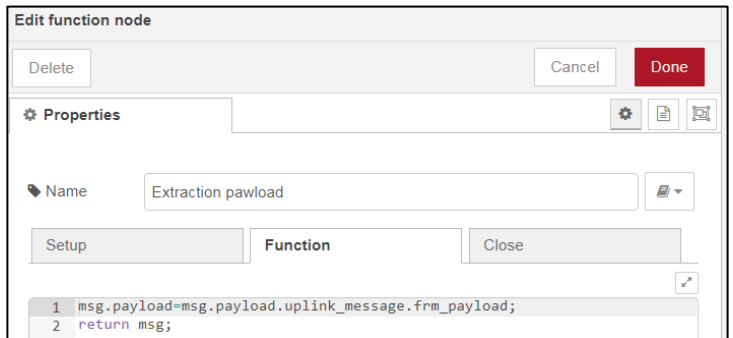
Ce nœud, nous permet d'ajouter du code (langage javascript) pour traiter le flux de données :

```
msg.payload=msg.payload.uplink_message.frm_payload;
return msg;
```

En sortie de ce nœud, on observe le payload suivant :

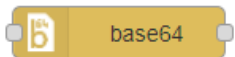
```
24/02/2022, 14:20:52 node: Données en b64
v3/mr-sti-iot@ttn/devices/eui-a8610a3434217a19/up :
msg.payload : string[16]
"MTkuMTA7NTYuMjA="
```

Il s'agit de la mesure transmise par notre node lora mkr-wan (uplink_message.frm_payload), mais celle-ci est codée au format base64.



4- Décoder la mesure transmise au format base64

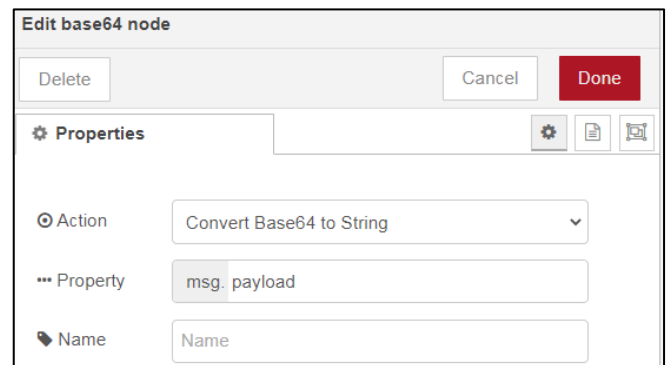
Dans la palette **parser**, sélectionnez un nœud de type **base64** :



En sortie de ce nœud, on observe le payload suivant :

```
24/02/2022, 14:20:52 node: Decodage b64
v3/mr-sti-iot@ttn/devices/eui-a8610a3434217a19/up :
msg.payload : string[11]
"19.10;56.20"
```

Il s'agit d'une chaîne de caractères composée de la température et de l'humidité (les deux valeurs sont séparées par le caractère « ; »)



5- Extraire la température et l'humidité de la chaîne de caractères

Dans la palette **function**, sélectionnez à nouveau un nœud de type **function** :



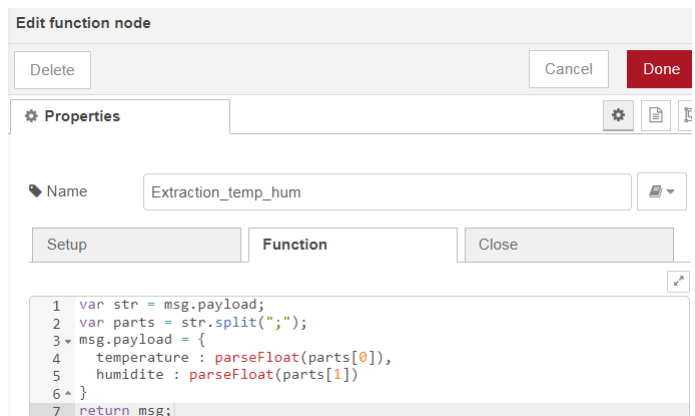
Utilisez le code suivant pour découper la chaîne de caractères à l'aide la fonction **split** :

```
var str = msg.payload;
var parts = str.split(";");
msg.payload = {
  temperature : parseFloat(parts[0]),
  humidite : parseFloat(parts[1])
}
return msg;
```

Après avoir extrait la température et l'humidité de la chaîne de caractères, ce code permet de créer un nouvel objet json composé de deux ensembles clé/valeur correspondant à la température et l'humidité. Ce nouvel objet json sera le payload transmis en sortie du nœud.

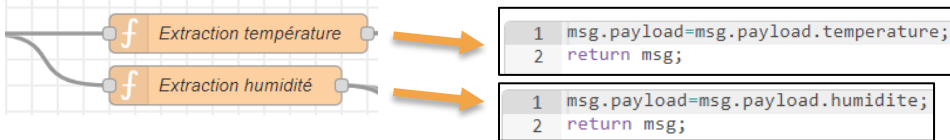
En sortie de ce nœud, on observe alors le payload suivant :

```
24/02/2022, 14:20:52 node: Extract_temp_hum
v3/mr-sti-iot@ttn/devices/eui-a8610a3434217a19/up :
msg.payload : Object
  { temperature: 19.1, humidite: 56.2 }
```



6- Séparer la température de l'humidité

Nous allons utiliser deux nœuds **function**. Chacun va s'occuper d'extraire du payload, la température ou l'humidité.



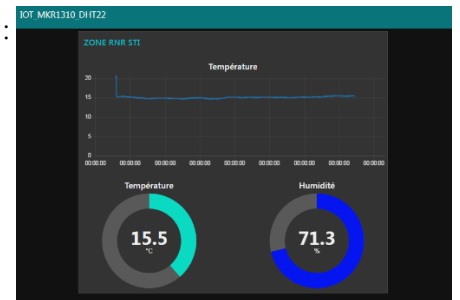
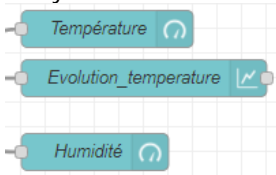
En sortie de ces deux nœuds, on observe les deux payloads suivant :

```
24/02/2022, 14:20:52 node: Humidité
v3/mr-sti-iot@ttn/devices/eui-a8610a3434217a19/up :
msg.payload : number
56.2

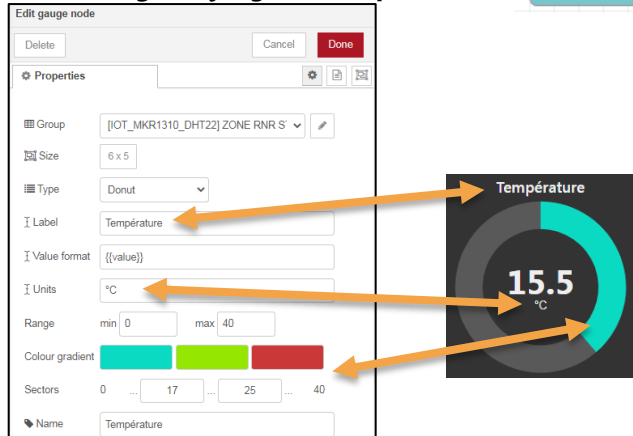
24/02/2022, 14:20:52 node: Température
v3/mr-sti-iot@ttn/devices/eui-a8610a3434217a19/up :
msg.payload : number
19.1
```

7- Construction de l'IHM

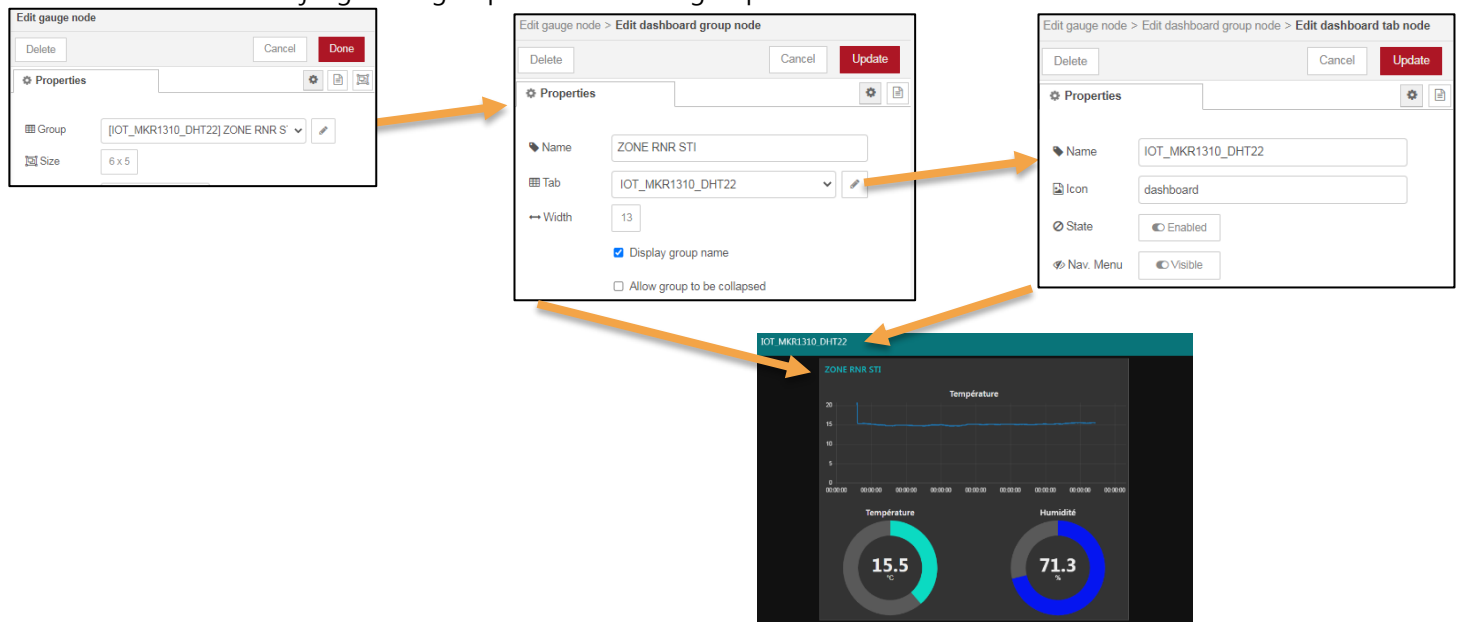
L'objectif est d'obtenir l'IHM (dashboard) ci-contre à l'aide des nœuds ci-dessous :



Paramétrage de jauge de température :



Il faut ensuite associer la jauge à un groupe et associer ce groupe au dashboard :



Paramétrage du graphique des températures :

Edit chart node

Delete Cancel Done

Properties

Group [IOT_MKR1310_DHT22] ZONE RNR S

Size 13 x 5

Label Température

Type Line chart ☐ enlarge points

X-axis last 1 weeks OR 1000 points

X-axis Label HH:mm:ss ☒ as UTC

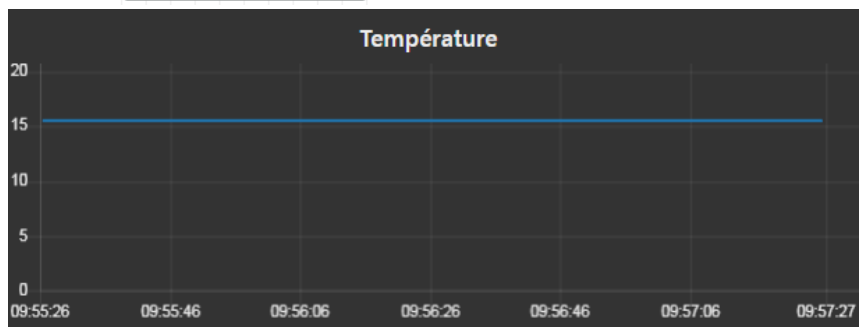
Y-axis min 0 max +20

Legend None Interpolate linear

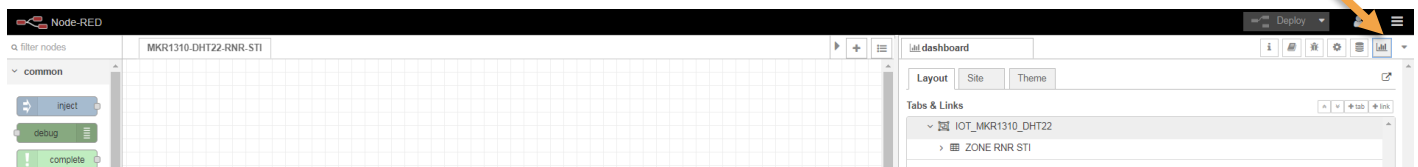
Series Colours

Blank label display this text before valid data arrives

Name Evolution_temperature



Configuration du dashboard :



Paramétrage générale :

dashboard

Layout Site Theme

Title ZONE RNR STI

Options

Show the title bar

Click to show side menu

No swipe between tabs

Node-RED theme everywhere

Date Format DD/MM/YYYY

Sizes

	Horizontal	Vertical
1x1 Widget Size	48	48
Widget Spacing	6	6
Group Padding	0	0
Group Spacing	6	6

Choix du thème :

dashboard

Layout Site Theme

Style

Dark

Base Settings

Colour

Font System Font (default)

Pour visualiser le résultat, cliquer sur le bouton suivant :



CONCLUSION

Il est bien sûr possible d'améliorer notre application en ajoutant par exemple à notre dashboard un widget de géolocalisation afin de localiser sur une carte OpenStreetMap la gateway utilisée et son RSSI. Il peut être aussi intéressant d'utiliser une base de données afin de sauvegarder les mesures. Pour cela, l'outil [InfluxDB](#) est parfaitement calibré. Node-red prend facilement en charge la base de données InfluxDB.

Dernier point, afin de **sécuriser l'accès à votre flow**, il peut être judicieux de configurer un accès à node-red par authentification.

Pour cela, allez dans le fichier `settings.js` situé dans le dossier `/home/pi/.node-red`.

Puis décommentez le bloc `"adminAuth"` :

```
adminAuth: {
  type: "credentials",
  users: [{
    username: "test",
    password: "S2bS08$SjtgWUaLCnLZS83bvUjM9ecgpYQgz7TlosPmit8SjHotMUx6k0vH.",
    permissions: "*"
  }],
},
```

Choisir un nom d'utilisateur et un mot de passe hashé.

Pour le mot de passe hashé, taper la ligne de commande suivante: `node-red admin hash-pw`

Puis entrer le mot de passe à hasher:

```
pi@raspberrypi:~ $ node-red admin hash-pw
Password:
$2b$08$KHpfRQh0.7v7kt0h0M.mG.SRm1StQd7jJ6c3tAMCbQlo0tTs8.ZY0
pi@raspberrypi:~ $
```

Copier/coller le résultat dans le fichier `settings`.

Remarque : Le champ permission avec la valeur `"*"` donne tous les droits à l'utilisateur. On peut la modifier avec la valeur `"read"`, ce qui permet à l'utilisateur d'uniquement consulter les flows (il ne peut pas modifier les flows).

L'accès par authentification sera actif au prochain démarrage de node-red.

Il est possible de créer d'autres utilisateurs en les ajoutant entre les accolades `{ }`, chacun séparés par une virgule.

Pour sécuriser l'accès au dashboard, c'est aussi dans le fichier `settings.js`

Décommenter les deux lignes ci-dessous, puis indiquer un nom d'utilisateur et un mot de passe « hashé » (méthode vue précédemment) :

```
// To password protect the node-defined HTTP endpoints (httpNodeRoot), or
// the static content (httpStatic), the following properties can be used.
// The pass field is a bcrypt hash of the password.
// See http://nodered.org/docs/security.html#generating-the-password-hash
httpNodeAuth: {user:"user",pass:"$2b$08$X4N92weYvKCyrIoy9nU2i09q/vxTaUE7R08x0MAI1JTUyNEvE1Gpw"},
httpStaticAuth: {user:"user",pass:"$2b$08$X4N92weYvKCyrIoy9nU2i09q/vxTaUE7R08x0MAI1JTUyNEvE1Gpw"}
```