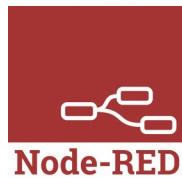




Installation et utilisation de Node-RED



DeltaLab

Espace Maison Milon
2 Place E.Colongin
84600 Grillon

deltalabprototype.fr

Qu'est-ce que DeltaLab ?

DeltaLab est une association 'loi 1901' d'intérêt général, dont l'objectif est la création d'un espace dédié à l'innovation, à la production numérique au prototypage et à l'«expression artistique».

Le principe fondateur de l'association est **d'apprendre à faire soi-même, pour passer de l'idée à l'objet.**

Deltalab se spécialise en **Objets Connectés**, et est en train de créer un vaste «écosystème digital» entre Drôme et Vaucluse, pour répondre à des besoins non-couverts, mettre à disposition ressources et équipements pour usage professionnels et instaurer des partenariats avec les autres structures et initiatives numériques existantes.

Deltalab est aussi un **FabLab** (*Fabrication Laboratory / Laboratoire de Fabrication*), un tiers-lieu de type makerspace où se trouve un atelier qui dispose de machines de fabrication comme des Imprimantes 3D ou des découpeuses Laser.

Deltalab se veut ouvert à tous publics : étudiants, professionnels, associations, inventeurs, designers, artistes, ...

Contexte de cette Documentation

Node-RED est un outil permettant de programmer facilement des serveurs. Il est basé sur Node JS, et son principe de base est l'utilisation de 'Nodes' représentant chacune une fonctionnalité, ce qui permet de ne presque jamais avoir à coder soi-même.

DeltaLab utilise Node-RED pour créer et gérer les serveurs liés aux objets connectés, par exemple pour le réseau d'antenne LoRaWAN ou pour le système centralisé de protection RFID.

Node-RED est très modulable et le nombre de Nodes existantes est très important, mais chaque projet a besoin de quelques Nodes de Base communes. Cette documentation se veut comme une introduction à Node-RED, en présentant un ensemble de Nodes et fonctionnalités utiles dans un bon nombre de vos projets.

Table des matières

1. Installer node-RED	04
2. Présentation générale - Introduction à node-RED	05
3. Personnalisation et Sécurisation de node-RED	09
4. Ajouter des objets connectés via the things network	11
5. Utiliser une Base de données	
1. Influx DB	16
2. PostGreSQL	19
6. Connexion avec une world map	21
7. Configurer une alerte e-mail	23
8. Configurer une alerte sms	25
9. Utiliser un Dashboard	27

I. Installer Node-RED

Vous devrez installer node-RED sur la machine qui vous servira de serveur. Ici, une raspberry pi sur laquelle est installé l'os raspbian stretch.
(note : ce tuto fonctionne avec toutes les versions de raspbian)

Pour installer Node-RED, vous devrez entrer les commandes suivantes :

```
sudo apt-get install build-essential
```

```
bash <(curl -sL https://raw.githubusercontent.com/node-red/linux-installers/master/deb/update-nodejs-and-nodered)
```

Vous pourrez ensuite lancer Node-RED grâce à la commande : *node-red-start*

Pour revenir sur la console, appuyez sur : *ctrl+c* .

Pour arrêter Node-RED, utilisez la commande : *node-red-stop*

Si vous voulez qu'à l'avenir, Node-RED se lance automatiquement au démarrage de votre raspberry pi, utilisez la commande : *sudo systemctl enable nodered.service*

Vous pouvez accéder à l'interface de Node-RED depuis un navigateur, en entrant :

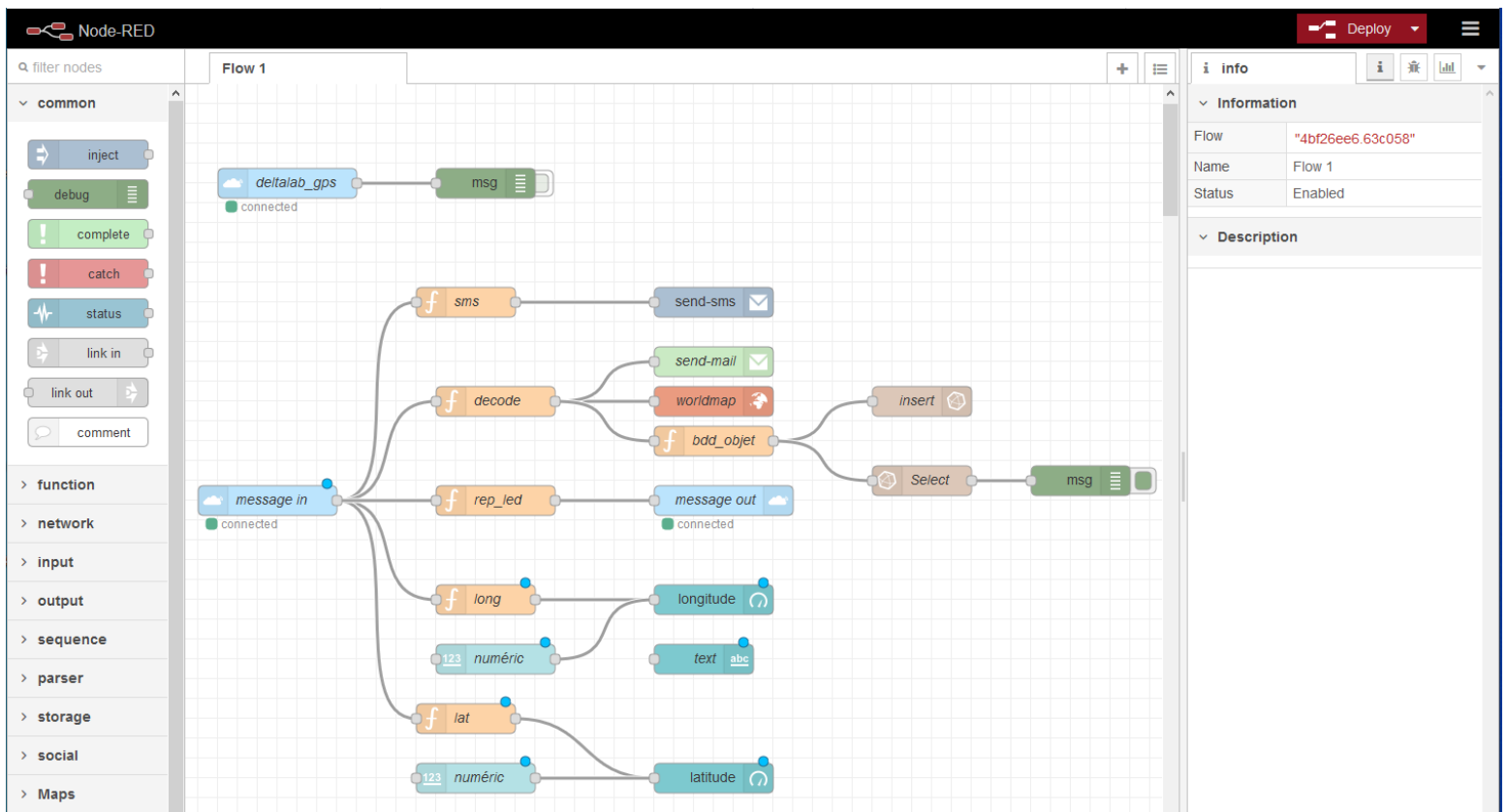
<http://ip-de-votre-raspberry:1880>.

II. Présentation générale - Introduction à Node-RED

Node-RED est un logiciel permettant de gérer des **flows d'événements**, des suites de traitements à effectuer suite à la réception de messages ou événements. Il contient un certain nombre de fonctionnalités de base, mais la plupart des fonctionnalités utiles dans notre cas devront être installées par la suite.

Dans Node-RED , une « fonctionnalité » est représentée sous la forme d'une **node** , un élément pouvant être placé dans votre **flow**, reliée à d'autres nodes en entrée ou en sortie. Le **flow** représente l'ensemble des **nodes**. Il n'est pas linéaire et une **node** reliée à aucune autre peut quand même s'activer si les conditions sont réunies.

1. L'Interface de Node-RED



L'interface de Node-RED se compose de 4 parties, qui sont :

- ◆ A gauche : **la liste des nodes disponibles**. Pour les placer sur le flow, sélectionnez celle que vous voulez et glissez la jusqu'à l'endroit voulu.

- ◆ Au centre : **les flows**. Vous pouvez en ouvrir autant que vous voulez, chaque flow est indépendant et en peut pas agir sur d'autres.

- ◆ A droite : des onglets utiles.
 - ◆ L'onglet **i** permet d'avoir des informations détaillées sur toute node sélectionnée.
 - ◆ L'onglet **debug** (icône d'insecte) apparaît dès qu'une node debug est placée est permet de voir les messages de debug.
 - ◆ L'onglet **dashboard** (icône de graphe) apparaît dès qu'une node de dashboard apparaît et permet d' y avoir accès.
 - ◆ D'autres onglets peuvent apparaître selon les nodes installées et placées

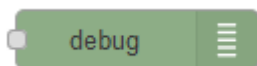
- ◆ En haut : Le bouton **Deploy** permet de « déployer » son flow et de le rendre actif. Le bouton **menu** (Icone de lignes parallèles) ouvre un menu, qui contient notamment les options :
 - ◆ **View** : gérer la vue (afficher ou non les menus des cotés). Permet aussi d'accéder au debug ou au dashboard si actifs.
 - ◆ **Import** : charger un flow sauvegardé
 - ◆ **Export** : sauvegarder les flows ouverts
 - ◆ **Manage Palette** : gérer les nodes installées et en installer de nouvelles
 - ◆ **Flows / Subflows** : créer un nouveau flow ou subflow.

2. Nodes de base

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



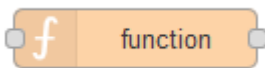
Permet d'afficher un message de debug. Parfait pour tester toute fonction ou node nouvellement ajoutée.



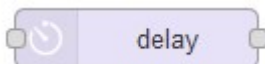
Permet d'écrire un commentaire, sans influence sur le flow.

- ◆ **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



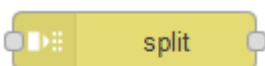
Permet de créer une fonction en JavaScript. Utile pour traiter un message reçu pour le rendre utilisable par une node de sortie.



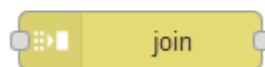
Permet d'imposer un délai aux messages entrants.

- ◆ **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 message sortants.



Permet de regrouper plusieurs messages entrants en un seul message sortant

- ◆ **Network** : Nodes permettant de gérer l'aspect réseau du flow, en paramétrant des requêtes http , des websockets, des messages tcp ou udp. C'est aussi dans cette catégorie que se rangeront les nodes mqtt (mosquitto) , si vous l'installez.
- ◆ **Parser** : Nodes permettant de traiter des données formatées et d'en extraire les objets javascript utilisable par les autres nodes, ou de formater un objet javascript en un format désiré. Ces nodes peuvent traiter du formatage **html ,csv, json, xml** ou **yaml**.
- ◆ **Storage** : Nodes permettant de sauvegarder des données de messages dans des fichiers. Permettent aussi de surveiller des fichiers pour y detecter tout changement. C'est aussi dans cette catégorie que se rangeront les nodes Influxdb et postgre si vous les installez.

Le menu i donnera des explications détaillées sur chacune de ces nodes.

III. Personnalisation & Sécurisation de Node-RED

Pour administrer Node-RED, il existe un outil , **node-red-admin**. Pour l'installer, enrez la commande `sudo npm install -g node-red-admin` dans votre raspberry.

1. Activer l'authentification

Il est possible de sécuriser l'accès à l'interface de Node-RED en imposant une indentation par identifiant & mot de passe. Pour cela, sur votre raspberrypi, vous devrez accéder au fichier de configuration de Node-RED, dont le chemin par défaut est : `/home/[vous]/.node-red/settings.js` . Dans ce fichier se trouve la propriété «**adminAuth**». Décommentez-la et l'authentification sera activée. Le login par défaut est **admin**, et son mot de passe **password**.

2. Créer un nouvel utilisateur

Dans le fichier **settings.js**, la propriété «**adminAuth**» contient un tableau «**user**». Ce tableau contient la liste des utilisateurs, qui sont définis par les champs :

- **username** : l'identifiant de l'utilisateur
- **password** : le mot de passe de l'utilisateur , hashé au format bcrypt.
Pour pouvoir créer un mot de passe, utilisez l'outil **node-red-admin**.
La commande `node-red-admin hash-pw` vous invitera à entrer le mot de passe que vous voulez, et vous reverra le hashage, que vous n'aurez qu'à copier-coller dans le fichier settings.
- **permissions** : les permissions accordées.
 - ***** : Toutes les permissions
 - **read** : Ne peut pas deployer ni changer les paramètres et tout changement dans le flow ne pourra être sauvegardé.

3. Changer l'adresse de l'éditeur

L'adresse par défaut de l'éditeur est `[ip-raspberry]:1880`.

Vous pouvez la modifier grâce au fichier `settings.js`

Vous pouvez changer le port en avec la ligne : `uiPort: process.env.PORT || 1880`

Vous pouvez ajouter une arborescence en décommentant la ligne : `httpRoot: '/red'`.

Et en remplaçant le `/red` par le chemin que vous souhaitez utiliser

L'adresse deviendra alors `[ip-raspberry]:[votre port] / [votre chemin]`

Si vous souhaitez utiliser le port **80** (ou tout port protégé), vous ne pourrez pas démarrer Node-RED avec la commande de base, le port étant protégé. Vous devrez utiliser la commande `sudo node-red -u .node-red`, qui permet de lancer Node-RED en tant que root tout en utilisant votre répertoire node-red normal.

NOTE : Le port **80** étant le port http, il n'est pas indispensable dans le chemin du navigateur. Ainsi, si vous utilisez le port **80**, le chemin `[ip-raspberry] / [votre chemin]` suffira pour atteindre Node-RED.

4. Avoir plusieurs Node-RED

Vous pouvez avoir plusieurs installations de Node-RED sur le même serveur, chacune ayant sa propre configuration, ses propres fichiers et ses propres nodes installées.

Une fois Node-RED installé, créez le(s) répertoire(s) dans lequel vous voulez le mettre (`mkdir chemin`) et copiez le repertoire de Node-RED dedans, avec la commande `cp -r .node-red chemin/.node-red`. Faites cette manipulation autant de fois que vous voulez de Node-RED différents. Chaque Node-RED est indépendant, et vous devrez installer les nodes voulues sur chaque Node-RED différent.

Pour lancer un Node-RED particulier, utilisez la commande

`node-red -u chemin/.node-red`

Le chemin est absolu ou relatif depuis `/home/VOUS`

N'oubliez pas le `sudo` si vous utilisez un port protégé (EX : 80)

Cette solution ne permet de lancer qu'un Node-RED à la fois.

IV. Ajouter des Objets Connectés via The Things Network

1. Installer les nodes TTN

Sur l'interface de Node-RED, ouvrez le menu en haut à droite et sélectionnez **Manage Palette**. Cliquez sur le volet **Install**, puis entrez **ttn** dans le champs **search modules**. Sélectionnez le module **node-red-contrib-ttn** et cliquez sur **install**.

2. Programmer un objet connecté

Pour qu'un objet connecté puisse se connecter à Node-RED et/ou à TTN , il faut le programmer pour. Plusieurs protocoles sont possibles, et plusieurs librairies pour chacun.

Pour utiliser LoRa, on utilisera la librairie LMIC (LoRa Mac In C).

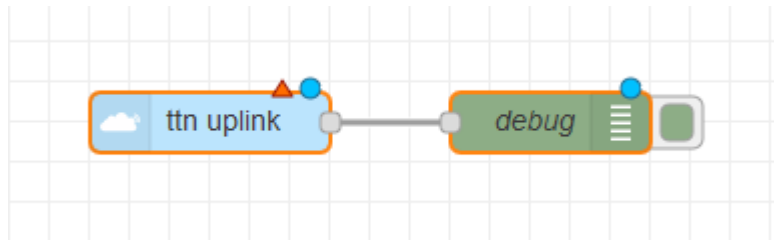
Sur votre ide de type Arduino, installez la librairie **MCCI_LoRaWAN_LMIC_library**, et partez d'un des exemples proposés dans le dossier **examples**, le **ttn-otaa** étant le plus simple pour aller sur TTN. Vous devrez récupérer les EUI de votre application et de votre device, ainsi que votre **ApplicationKey** sur TTN.

Note : Si vous n'avez pas d'ide, vous pouvez récupérer l'ide **Arduino** openSource à l'adresse : <https://www.arduino.cc/en/main/software>

Téléversez le code sur votre objet connecté - sans doute une carte arduino de type uno ou ttgo - et il devrait se connecter à ttn et commencer à emettre des messages, si une antenne réceptrice ttn - type Lorix One - est active à proximité.

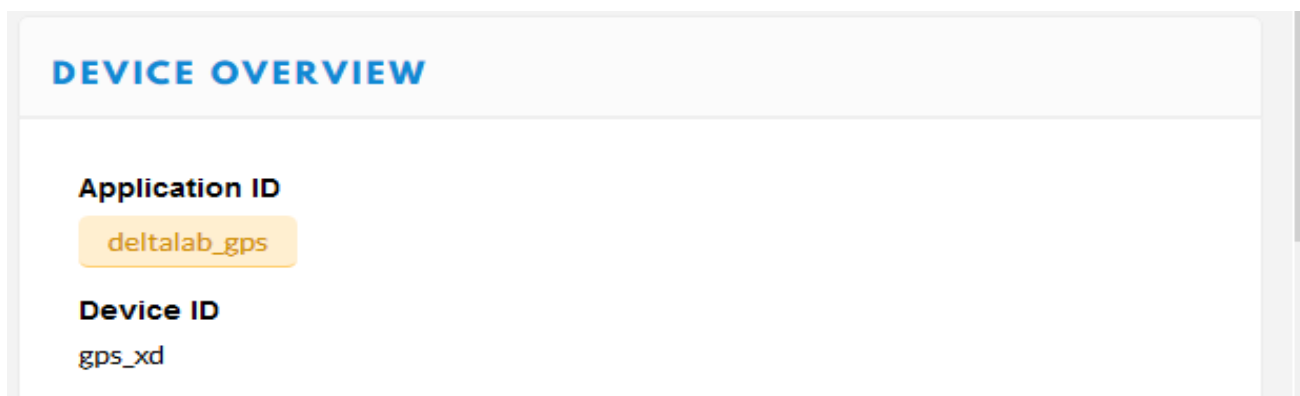
Sur TTN , dans votre application, vous pouvez aller dans l'onglet **Payload Formats** , et écrire des fonctions pour récupérer les données des messages reçus et formater le payload du message émis sur Node-RED.

3. Ajouter un objet à un flow



Prenez une node **ttn-uplink** et placez là sur votre flow. Double-cliquez dessus pour ouvrir ses Properties.

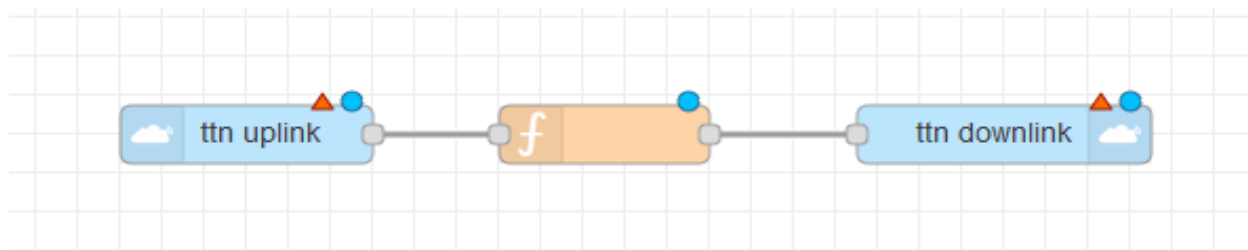
- ◆ **Name** : le nom que vous voulez donner à votre node dans le flow
- ◆ **App** : l'id de votre application, à récupérer sur The Things Network
- ◆ **Device ID** : l'id du device , à récupérer sur The Things Network



La node **debug** permet d'afficher dans la fenêtre de debug le message envoyé par le device à chaque fois qu'il en envoie un.

Elle possède la property **output**, qui permet de choisir si vous souhaitez voir uniquement le payload (les données) ou le message entier avec les headers et tous les champs.

Pour renvoyer un message en réponse au device, prenez une node **ttn downlink** et placez la sur votre flow. Ses Properties sont les même que pour la node **ttn uplink**.



La node **function** placée au milieu permet de paramétrer le message de réponse.

Exemple de fonction :

```
return {  
  dev_id: msg.dev_id,  
  port: msg.port,  
  payload: {  
    led: !msg.payload.led  
  }  
};
```

Cette fonction renvoie un message qui active la LED du device de type arduino.

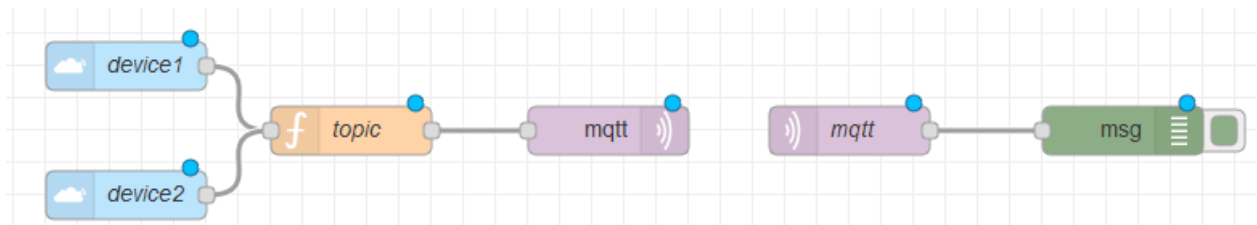
4. Utiliser MQTT (Mosquitto)

MQTT est un protocole d'envoi de données basé sur tcp/ip qui peut être utilisé à la place de protocoles comme LoRa. Il se divise en 3 types d'entités : les **publishers**, qui envoient des messages , les **listeners** ou **subscribers** qui écoutent pour recevoir des messages, et un **broker**, qui est le serveur qui fait le lien entre les 2. Les publishers et listeners communiquent grâce à un topic auquel ils sont «abonnés» (subscribed).

4.1. Installation

Pour utiliser MQTT, il vous faudra installer le broker sur votre raspberry pi, ici Mosquitto, grâce à la commande : `sudo apt-get install mosquitto -y`
Mosquitto utilise de base le port 1883, si vous voulez en changer vous devrez lancer mosquitto manuellement avec la commande `mosquitto -p [port]`.
Sinon, il se lancera automatiquement avec Node-RED.

Sur Node-RED, les nodes MQTT sont installées de base. Vous pouvez choisir d'installer le module `node-red-contrib-mqtt-broker` si vous voulez que le broker soit directement sur Node-RED , et non Mosquitto. Les différences sont minimes.



4.2. Utilisation

Plusieurs devices envoient leurs messages au broker mosquitto, en passant par la node **mqtt out**, le publisher. La **fonction topic** rajoute le champs topic aux messages des devices. Le **mqtt out** s'abonne à ces topics et y envoie les messages. La node **mqtt in** reçoit les messages de tous les topics, en étant abonnée au topic '#', qui représente tous les topics possibles.

Les nodes **mqtt out** et **mqtt in** ont les mêmes propriétés, qui sont :

- ◆ **Server** : Le serveur sur lequel se trouve le broker. Ici, localhost ou 127.0.0.1 font l'affaire. Le port par défaut est 1883.
- ◆ **Topic** : Le Topic auquel s'abonner. # représente tous les topics possibles, laisser le champs libre dans la node mqtt out permet de paramétrer le topic selon le champs topic du message d'entrée.
- ◆ **QoS** : Quality of Service, détermine la façon dont sont traités les messages

La node **mqtt out** possède également la propriété **retain**, qui détermine si le broker doit retenir le message même si aucun listener n'est abonné au bon topic.

La node **mqtt in** possède la propriété **output**, qui permet de configurer le type de sortie que l'on veut (string , buffer, objet msg...)

Les devices peuvent également utiliser directement MQTT pour communiquer avec Node-RED, via wifi. Les devices et le serveur raspberrypi doivent donc être connectés au même réseau wifi. Ces messages-là n'apparaîtront donc pas dans ttn, n'utilisant pas le protocole LoRa. On peut donc utiliser un flow permettant de récupérer le mqtt et de le transmettre à ttn, c'est à dire l'inverse du flow précédent.

Pour cela, il faut reprogrammer le device pour qu'il se connecte au wifi et utilise mqtt pour communiquer.

V. Utiliser une Base de Données

1. InfluxDB

InfluxDB est une Base de Données ‘Temporelle’, c’est-à-dire dont tous les enregistrements sont liés au temps. Chaque ajout d’enregistrement est automatiquement associé au moment de l’ajout, sous forme TimeStamp. Ce SGBD est avantageux pour les serveurs d’objets connectés, qui stockent des données envoyées par de multiples objets régulièrement et avec peu de traitements dessus. La modification d’un enregistrement est tout simplement Impossible.

1.1. Installation

Sur votre raspberry pi, vous devrez installer InfluxDB et créer la base de données que vous voudrez utiliser dans Node-RED. Pour ce faire, vous devrez entrer les commandes suivantes :

```
curl -sL https://repos.influxdata.com/influxdb.key | sudo apt-key add -echo "deb https://repos.influxdata.com/debian stretch stable" | sudo tee /etc/apt/sources.list.d/influxdb.list
```

```
sudo apt-get install influxdb
```

Entrez ensuite *sudo service influxd start*, qui démarre le service **influxd**, puis **influx**, pour démarrer influx. **Influxd** doit être démarré pour que Node-RED puisse se connecter aux bases de données.

Une fois sur Influx, les commandes sont les suivantes :

create database DB : Crée la base de données DB

drop database DB : Supprime la base de données DB

show databases : montre toutes les bases de données qui existent

use DB : Choisit DB comme base de données à utiliser

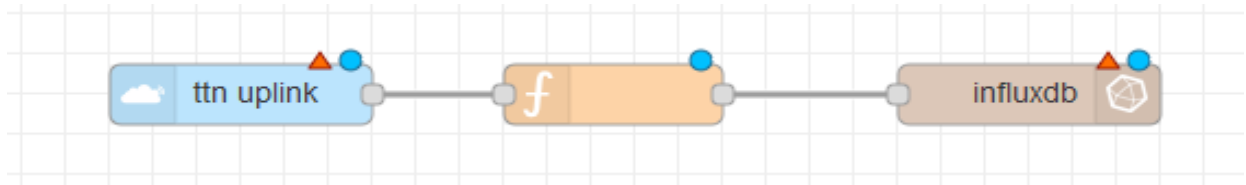
*select * from TABLE (where COND)* : Affiche toutes les données de la table TABLE, ou toutes celles répondant à la condition COND si where est utilisé

delete from TABLE (where COND) : Supprime toutes les données de la table TABLE, ou toutes celles répondant à la condition COND si where est utilisé

Sur Node-RED, allez dans le menu **Manage Palette**, et entrez **InfluxDB** dans le volet **Install**. Sélectionnez le module **node-red-contrib-influxdb** et installez le.

1.2. Utilisation d'InfluxDB dans Node-RED

➔ Insérer des données



Pour insérer des données dans une base de données, prenez une node **Influxdb out** et ajoutez la à votre flow. Ses propriétés sont les suivantes :

- ◆ **Server** : Le serveur sur lequel se trouve votre Base de Données.
Si le votre n'apparaît pas dans la liste déroulante, appuyez sur le crayon pour en ajouter un nouveau.

Les propriétés d'un serveur sont :
 - ◆ **host** : l'ip de votre serveur (raspberrypi ou est installé influx)
Si vous utilisez Node-RED sur l'ordinateur sur lequel est branché la raspberrypi, l'@ip 127.0.0.1 fonctionnera aussi.
 - ◆ **database** : la base de données que vous voulez utiliser
 - ◆ **username** : l'id avec lequel vous voulez vous connecter
 - ◆ **password**: le mot de passe avec lequel vous voulez vous connecter
- ◆ **Measurement** : Le nom du measurement (une table sous influx) dans lequel vous voulez insérez vos données.
- ◆ **Name** : Le nom que vous voulez donner à cette node dans le flow.

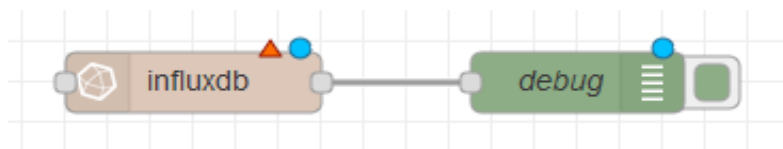
La node **function** placée au milieu permet de transformer le message reçu du device en un objet dont les champs seront stockés dans la base de données.

Exemple de fonction :

```
return {
  topic:"",
  payload:{ "id":"grillon_01" , "lat":msg.payload.lat , "lon":msg.payload.lon }
};
```

Cette fonction récupère les champs utiles du message arrivé et les transmet dans un objet de type msg, en y ajoutant un id et un topic.

➔ Récupérer des données



Pour récupérer des données depuis une base de données Influxdb, prenez une node **Influxdb in**, et placez la dans votre flow. Ses propriétés sont les suivantes :

- ◆ **Server** : Le serveur sur lequel se trouve la base de données, comme pour la node **influxdb out**
- ◆ **Query** : La requête que vous voulez effectuer sur la base de données.
Pour récupérer des données, elle sera du type :

Select CHAMPS from TABLE (where CONDITION)

Récupère toutes les données des champs CHAMPS (ou de tous les champs si vous mettez ' * ' à la place) de la table TABLE , ou ceux qui répondent aux conditions CONDITION si where est utilisé.

Exemple : `Select lat , lon from gps where id='Grillon'`

2. PostGreSQL

PostGre est une Base de Données 'Relationnelle', c'est-à-dire dont tous les tables sont liées entre elles. Ce SGBD est avantageux pour les serveurs plus classiques, qui stockent peu de nouvelles données mais avec beaucoup de traitements, qui peuvent être répétitifs. La modification d'un enregistrement est ainsi très simple, ainsi que des requêtes plus complexes.

2.1. Installation

Sur votre raspberry pi, vous devrez installer PostGre et créer la base de données que vous voudrez utiliser dans Node-RED. Vous devrez utiliser la commande suivante :

```
sudo apt-get install postgresql postgresql-contrib
```

PostGre crée un utilisateur par défaut, *postgres*. Il vous faudra donc vous connecter à PostGre avec cet utilisateur la première fois, avec les commandes :

```
sudo su - postgres  
psql
```

Une fois sur PostGre, vous pourrez créer un nouvel utilisateur avec le nom et mot de passe que vous voulez, grâce à la commande

```
CREATE ROLE pseudo WITH LOGIN CREATEDB ENCRYPTED PASSWORD 'passwd';
```

Utilisez *\q* pour quitter *psql*, puis *sudo su - pseudo* pour passer de *postgres* à votre compte. Réutilisez *psql* pour relancer *postgre* avec votre compte. Sur PostGreSQL, les commandes de bases sont :

createdb DB : Crée la base de données DB

\connect DB : Choisit DB comme base de données à utiliser

*select * from TABLE (where COND)* : Affiche toutes les données de la table TABLE, ou toutes celles répondant à la condition COND si where est utilisé

delete from TABLE (where COND) : Supprime toutes les données de la table TABLE, ou toutes celles répondant à la condition COND si where est utilisé

update TABLE set (champs = value) (where COND) : Modifie les champs avec les valeurs données dans la table TABLE ou dans les données de celle-ci qui répondent à la condition COND si **where** est utilisé

Sur Node-RED, allez dans le menu **Manage Palette**, et entrez **postgre** dans le volet **Install**. Sélectionnez le module **node-red-contrib-postgrestor-next** et installez le.

2.2 : Utilisation sur Node-RED

La node Postgrestor contient un champs de requête, et un champs de Base de Données. Les paramètres importants de la Base de données sont :

- ◆ **Host / port** : L'IP de l'hôte de la Base de Données, et le port pour l'atteindre
- ◆ **Database** : La Base de Données que vous voulez utiliser
- ◆ **SSL** : Connexion sécurisée ou non
- ◆ **User** : identifiant pour accéder à la BD, doit exister dans psql sur votre serveur
- ◆ **Password** : mot de passe pour accéder à la BD
- ◆ **Pool** : 'taille' de la connexion : combien d'échanges elle permet

La requête peut récupérer des données du message d'entrée. Cela devra alors être écrit sous la forme : `'{{msg.payload.champs}}'`

La requête peut être des 4 grands types :

SELECT champs FROM table WHERE cond : récupérer des données

INSERT INTO table VALUES (v1,v2,...) : insérer des données dans une table

UPDATE table SET (c1=v1,...) WHERE cond : modifier des données existantes

DELETE FROM table WHERE cond : effacer des données

Le résultat de la requête est détaillé dans le payload du message de sortie. Les champs importants sont :

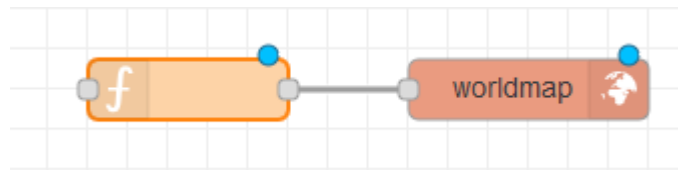
- **rowCount** : nombre de lignes impactées par la requête
- **query** : le type de requête effectuée
- **rows** : les données renvoyées, sous forme de tableau.

IV. Connexion avec une World Map

1. Installation des nodes world-map

Sur Node-RED, allez dans le menu **Manage Palette**, et entrez **worldmap** dans le volet **install**. Sélectionnez le module **node-red-contrib-web-worldmap** et installez le.

2 Utilisation de la world map



WorldMap permet de placer des points de données sur une carte du monde. Pour s'en servir, prenez une node **worldmap** et placez la sur votre flow. Ses propriétés sont :

- ◆ **Start** : Définit le point de départ de la carte , le point sur lequel la carte est centrée quand vous l'ouvrez.
- ◆ **BaseMap** : Définit la map de départ parmi une liste de map.
- ◆ **Cluster** : Définit le niveau de zoom à partir duquel les points proches sont rassemblés
- ◆ **Max Age** : définit une durée au delà de laquelle les points sont supprimés
- ◆ **Web Path** : définit le chemin permettant d'accéder à la carte. Il sera de la forme
http://@ip_de_votre_raspberrypi:1880/chemin
- ◆ **Name** : le nom que vous voulez donner à cette node dans votre flow.

Vous pouvez intégrer une worldmap dans un dashboard de 2 manières : avec la node **ui-worldmap** (si elle marche) ou avec une node **template** qui référence une worldmap indépendante.

La node **function** permet d'insérer une donnée à la carte.
Elle doit être de la forme :

```
return{
  topic:"",
  payload:{
    "name":"grillon",
    "lat":msg.payload.latitude,
    "lon":msg.payload.longitude
  }
};
```

Les champs **lon** et **lat** sont obligatoires et déterminent la longitude et la latitude du point à placer. Le champs **name** est unique sur toute la map, un nouveau point avec un name déjà existant remplacera toujours l'ancien.

Les autres champs possibles sont :

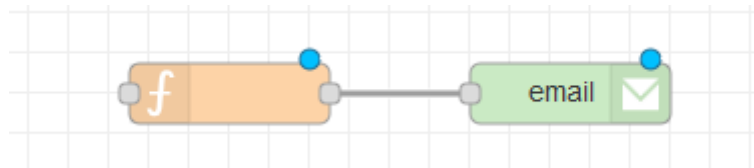
- ◆ **icon** : définit l'icone à utiliser pour représenter le point. Icones possibles :
 - ◆ Font-Awesome Icons avec le préfix **fa-**
(Liste des icones : <https://fontawesome.com/v4.7.0/icons/>) ,
 - ◆ Emoji avec la syntaxe **::emoji::**
(Liste des emojis : <https://github.com/dceejay/RedMap/blob/master/>)
 - ◆ Image , qui sera réduite à du 32x32. Vous devrez écrire le chemin complet.
- ◆ **iconColor** : Permet de modifier la couleur de l'icone, en utilisant un nom standard CSS ou une valeur RGB hexadécimale (#rrggbb)
- ◆ **tll** : Time to live , définit la durée de vie en secondes du point avant qu'il disparaisse. Ce réglage passe au-dessus du Max Age de la map et est forcément supérieur à 20s
- ◆ **photourl / videourl** : Permet d'associer une photo ou une vidéo au point.
- ◆ **Weblink** : Permet d'associer un lien vers une page web au point.
- ◆ **label** : définit une valeur qui sera affichée à coté du point.
- ◆ **tooltip** : définit une valeur qui apparaîtra si on passe sur le point.

VII. Configurer une alerte e-mail

1. Installation

Sur Node-RED , allez dans le menu **Manage Palette** , et entrez **email** dans le volet **install**. Sélectionnez le module **node-red-node-email** et installez le.

2. Utilisation



Pour envoyer des e-mails depuis node-RED, prenez une node email et placez la sur votre flow. Ses propriétés sont les suivantes :

- ◆ **TO** : L'adresse e-mail de destination
- ◆ **Serveur smtp / port** : Le serveur smtp et le port par lesquels passeront les emails
- ◆ **Userid / Password** : L'id et le mot de passe de l'expéditeur sur le serveur smtp (aka votre id et mdp pour vous connecter à votre boîte mail que vous souhaitez utiliser pour l'envoi)
- ◆ **Name** : Le nom que vous voulez donner à cette node sur votre flow.

La node **function** permet de définir le mail à envoyer

Exemple de fonction :

```
return{
  topic:"rapport gps grillon",
  payload:"aujourd'hui grillon se trouve aux coordonnées : ( "
    +longitudinalement+"; "+msg.payload.latitude+")."
};
```

Le **topic** définit l'objet du mail , et le **payload** le contenu. Il peut être écrit au format HTML. Vous pouvez également définir des **attachments** , qui sont autant de pièces jointes que d'objets définis. Les **attachments** doivent être écrits aux format **nodemailer**, qui est de la forme :

```
attachments: [  
  { filename: 'text1.txt', path: '/path/to/file.txt' }  
  { filename: 'text2.txt', content: 'content to join' }  
]
```

Les champs possibles dans l'attachment sont

- ◆ **filename** : nom du fichier pièce jointe
- ◆ **path** : chemin local vers le fichier à joindre
- ◆ **url** : url vers le fichier à joindre
- ◆ **content** : contenu de type String ou Buffer à envoyer en pièce jointe

NOTE : Si vous utilisez **gmail**, vous devrez peut-être activer **l'accès moins sécurisé des applications**. Pour cela, accédez à l'onglet **sécurité** dans les paramètres de votre compte Google, et activez **l'accès moins sécurisé des applications**, en bas de la page

VIII. Configurer une alerte sms

1. Installation

- ◆ Sur votre raspberry pi , vous devrez installer un serveur de sms. J'ai ici choisi de présenter sms server tools 3 , mais il en existe plusieurs autres.

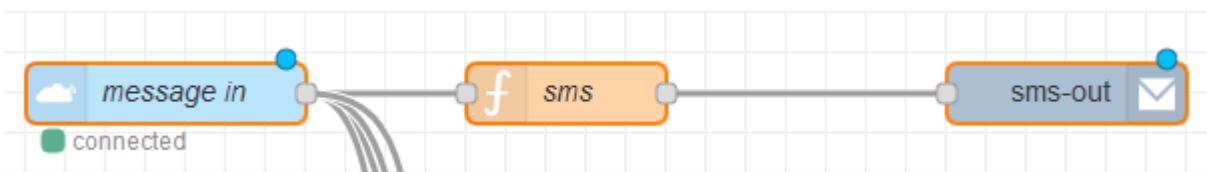
Pour installer le serveur , entrez la commande `sudo apt-get install smstools`

Puis entrez la commande `sudo /etc/init.d/smstools start` pour lancer le serveur.

Pour arrêter le serveur, entrez la commande `sudo /etc/init.d/smstools stop`

- ◆ Sur Node-RED , allez dans le menu **Manage Palette** , et entrez **smstools** dans le menu **install**. Sélectionnez le module **node-red-contrib-smstools** et installez le.

2. Utilisation



SmsTools permet d'envoyer et de recevoir des sms. Pour envoyer un sms , prenez une node **sms-out** et placez la sur votre flow. Ses properties sont les suivantes :

- ◆ **Send to** : Le numéro de téléphone auquel vous voulez envoyer un sms
- ◆ **Name** : Le nom que vous voulez donner à cette node dans votre flow.

La node **function** permet de définir le message à envoyer. La fonction doit être de la forme :

```
return{
  topic:"0621365747",
  payload:"gps : grillon ( "+msg.payload.longitude+" ; "+msg.payload.latitude+" )."
};
```

Le **topic** contient le numéro du destinataire , et le **payload** contient le contenu du message à envoyer.










IX. Utiliser un Dashboard

1. Installation des nodes Dashboard

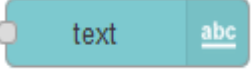
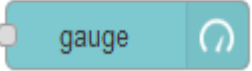
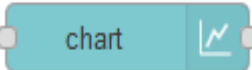

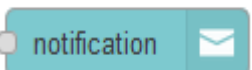
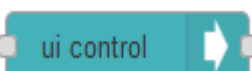

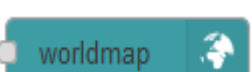
Sur Node-RED , allez dans le menu **Manage Palette** et entrez **dashboard** dans le volet **install**. Sélectionnez le module **node-red-dashboard** et installez le. Vous pouvez aussi installer quelques modules supplémentaires pour avoir plus de fonctionnalités.

2. Créer un Dashboard

Un dashboard est une interface graphique composée de plusieurs éléments, chacun représenté par une node. On peut les diviser en deux types : éléments d'entrée et éléments de sortie. Avec le module node-red-dashboard, vous avez accès à tous les éléments d'entrée suivants :

-  **button** : Un simple bouton
-  **dropdown** : Une liste déroulante
-  **switch** : Un interrupteur (Bouton on/off)
-  **slider** : Une barre de valeurs ajustable
-  **numeric** : Un champs permettant de choisir un nombre
-  **text input** : Un champs permettant d'entrer du texte
-  **date picker** : Un champs permettant de choisir une date
-  **colour picker** : Un champs permettant de choisir une couleur
-  **form** : Un formulaire. Il peut contenir plusieurs champs de plusieurs types différents (texte , mot de passe , email, switch,date...)


Ces éléments envoient des messages lorsqu'ils sont activés (bouton, switch) ou modifiés (numéric / text input , date / color picker). Ces messages sont ensuite reçus par des éléments de sortie , si ils sont connectés aux élément d'entrée. Chaque élément de sortie attend un message d'entrée spécifique, il sera peut être nécessaire d'ajouter une node fonction pour transformer le message envoyé par l'élément d'entrée en un message compréhensible par l'élément de sortie. Les éléments de sortie sont :

-  : Un texte , non modifiable sur l'interface
-  : Une jauge , qui peut être de plusieurs formes
-  : Un graphique , qui peut être de plusieurs types
-  : Permet de jouer un fichier audio (.mp3, .wav) , ou une String de texte si le « text-to-speech » est activé sur votre navigateur.
-  : Un pop-up qui s'active quand il reçoit un message
-  : Permet un contrôle dynamique de l'interface
-  : Permet d'utiliser des directives HTML et Angular
-  : Si vous avez installé Worldmap (cf part 4), permet d'intégrer une carte du monde. Fonctionne comme Worldmap.

Tous les éléments graphiques ont quelques propriétés en commun, qui sont :

- ◆ **group** : le groupe d'affichage auquel appartient l'élément. La séparation est uniquement graphique, deux éléments de deux groupes différents pouvant communiquer ensemble. Un **group** possède les propriétés suivantes :
 - ◆ **name** : le nom du groupe
 - ◆ **tab** : l'onglet dans lequel le groupe apparaîtra. Un dashboard peut se composer d'autant d'onglets que nécessaires.
 - ◆ **Width** : la largeur du groupe. Les éléments du groupe ne pourront pas la dépasser
 - ◆ **display group name** : détermine si le nom du groupe doit être affiché
 - ◆ **allow group to be collapsed** : détermine si le groupe peut être réduit

- ◆ **Size** : La taille de l'élément
- ◆ **label** : Un texte à afficher à coté de l'élément
- ◆ **tooltip** : Un texte à afficher si on passe par dessus l'élément
- ◆ **value format** : Le format de la valeur attendue (pour un élément de sortie) ou émise (pour un élément d'entrée)

Pour accéder à votre dashboard , vous pouvez soit aller dand l'onglet **dashboard** du menu de droite, qui apparaît dès que vous avez ajouté une node de dashboard à votre flow, et cliquer sur l'icone  en haut à droite ; soit utiliser une @ip du type :

[http:// \[votre_ip\] : \[votre_port\] / \[votre_chemin\] / ui /](http://[votre_ip]:[votre_port]/[votre_chemin]/ui/)

Ce chemin par défaut est modifiable dans le fichier settings.js , dans le répertoire .node-red de votre raspberrypi, en décommentant et modifiant la ligne :

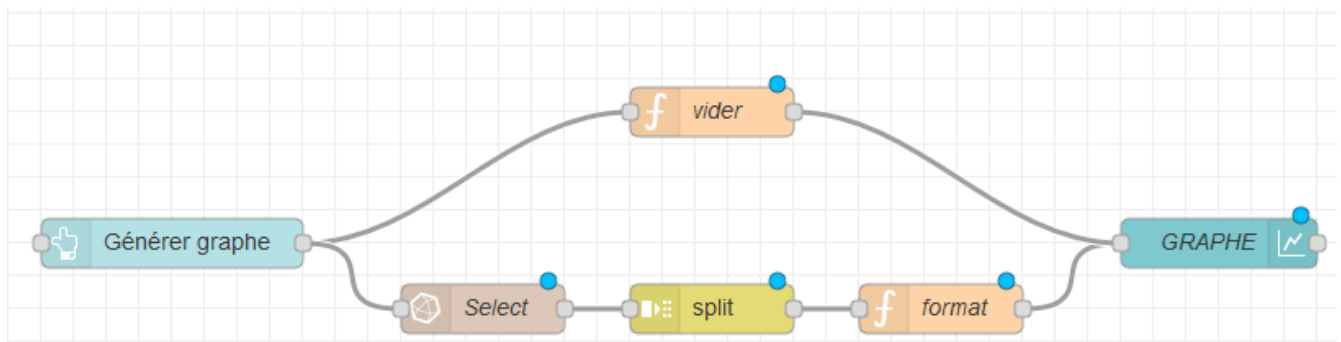
ui : {path=« ui »} ,

Vous pouvez rendre votre dashboard read-only, faisant que toute modification sur le dashboard (même le remplissage de champs ou l'utilisation de silders) n'aura aucun effet

3. Exemple d'utilisation d'un dashboard :

faire un graphique de données venant de la base de données

Flow :



Ce flow permet de générer un graphique en barres contenant des données issues de la base de données Influxdb.

Le **button** «générer graphique» démarre le flow. La **function** « vider » envoie un message vide au graphe, ce qui le fait se vider. Un graphe contenant déjà des données pourra en rajouter mais ne rafraichira pas celles qu'il a déjà.

La node **Influxdb in** (Select) permet de récupérer les données voulues.

```
Select time,lat From gps Where id='grillon_01'
```

Ici, la requête récupère la date et la latitude de tous les enregistrements du device 'grillon_01'. Elle renvoie un tableau d'objets , un objet représentant un enregistrement.

La node **split** permet de diviser le tableau d'objets renvoyé par la requête Select en des messages successifs contenant chacun un objet.

La node **function format** met en forme le message pour que le graphe puisse le lire.

```
var time = msg.payload.time;  
var dat = time.getDate()+"/"+(time.getMonth()+1)+"/"+time.getFullYear();  
var heur = time.getHours()+":"+time.getMinutes()+":"+time.getSeconds();  
return { label:dat+" - "+heur , payload:msg.payload.lat }
```

Le **label** contient le label que vous voulez attribuer à la donnée, le **payload** contient la valeur en question. Il ne doit contenir aucun attribut, juste la donnée directement.

La node **chart** (GRAPHE) permet de paramétrer le graphique. Ici j'utilise un «**bar chart**» (diagramme en colonnes) , mais un «**bar chart H**» (diagramme en barres horizontales) marcherait tout aussi bien. Les « **line chart** » (diagrammes en lignes) sont parfaits pour représenter des messages au fur et à mesure de leur arrivée, mais pas pour récupérer des données déjà dans la BD, ils se rafraichissent tous seuls au fil du temps et l'axe X ne peut représenter que le temps. Il est aussi possible de générer des diagrammes « **pie chart** » (diagramme en camembert) , « **polar area chart** » et « **radar chart** » , ces derniers étant particuliers et donc moins utilisés.

Résultat :

