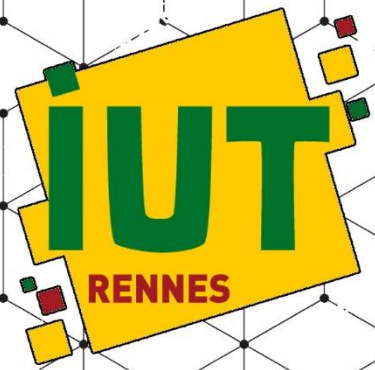
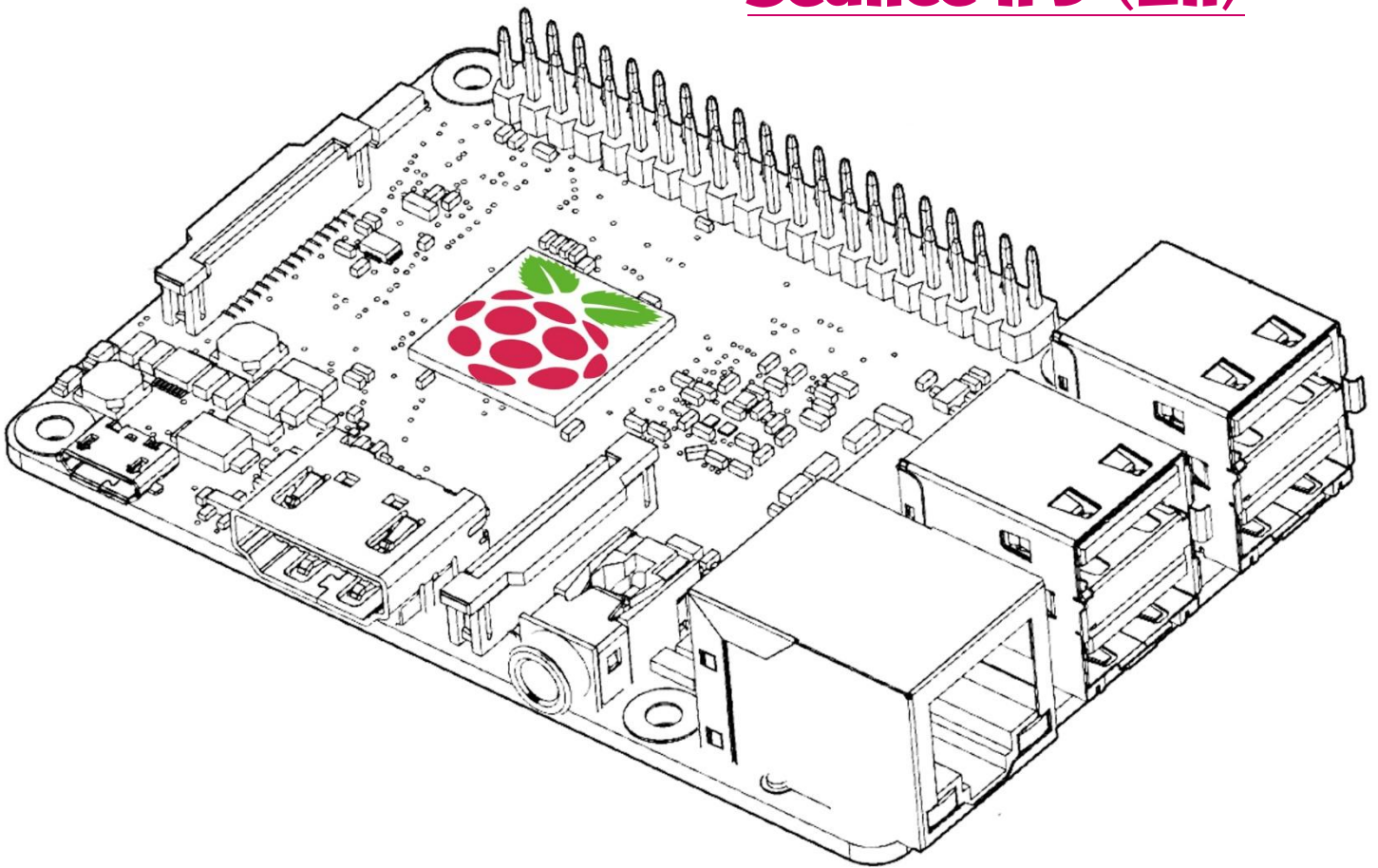


# AT31 - Module IPE



Séance n°5 (2h)

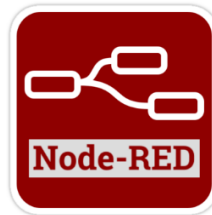


Découverte de Node-Red

**Michaël BOTTIN**

## Objectifs de la séance :

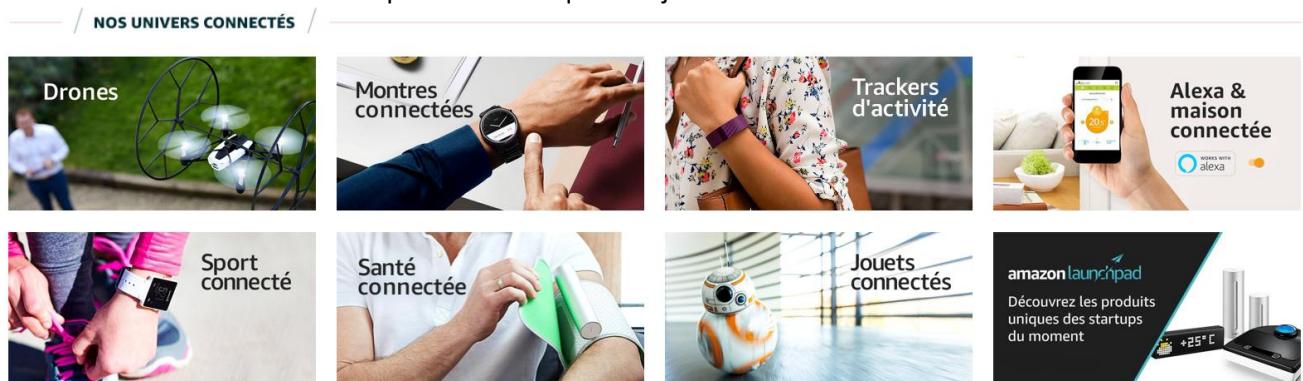
- Découvrir comment contrôler et programmer la Raspberry à travers un réseau local via l'outil de programmation graphique Node-Red



## Préambule :

Les objets connectés envahissent progressivement notre quotidien. Plusieurs nouveaux objets sortent chaque semaine. Ce sont bien souvent des capteurs ou des associations de capteurs.

Exemple de la boutique d'objets connectés d'AMAZON :



Les informations renvoyées par ces capteurs peuvent vite devenir importantes et le traitement et la valorisation de toutes ces données reste un enjeu majeur actuellement.

De nombreuses plateformes de stockage en ligne et d'affichage en ligne (sous forme de 'dashboards') de données ont vu le jour, ces dernières permettant par exemple de centraliser les informations d'un parc d'objets connectés, de générer des alertes ou de piloter des éléments extérieurs (domotique par exemple).

Parmi elles, on retrouve : Blynk, Cayenne (MyDevices), Thingspeak, Initial State, Cloud4Rpi, Ubidots, IBM Watson, Google Cloud, Microsoft Azure, AWS IoT, Thingsboard, Flatlogic, Kaa, Thinger.io, Adafruit.io, ... Malheureusement, ces plateformes offrent souvent un service gratuit quelques mois, voire quelques années puis proposent des formules par abonnement.

Et créer un outil facile d'accès et permettant de couvrir un grand nombre de situations de notre monde numérique actuel n'est pas une chose aisée.

Certains ont fait le choix d'une solution sans écrire un seul programme, mais des limitations peuvent vite apparaître...

Une autre solution est donc apparue :

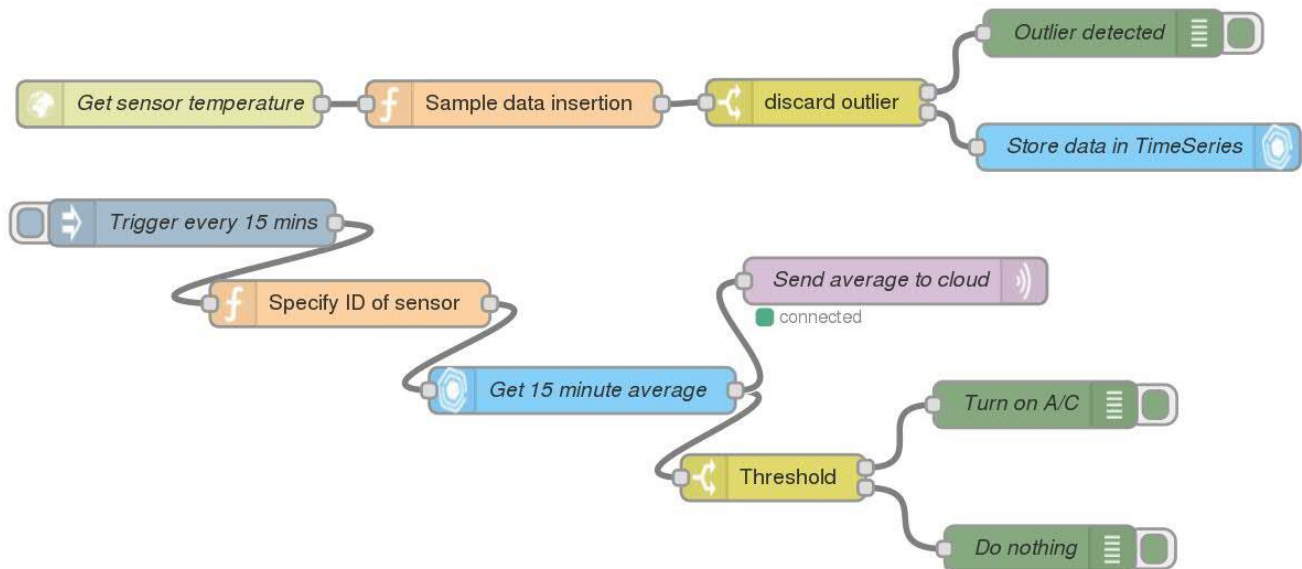
<https://nodered.org/>

Node-Red a fait un choix différent et celui-ci semble porter ses fruits vu son utilisation croissante depuis sa genèse en 2013...

C'est un outil de programmation graphique basé sur les flux de données. Cette forme de programmation existe depuis les années 70 et elle est actuellement naturellement orientée objet.

Et cet outil est plutôt dirigé vers la conception d'application IoT ('Internet Of Things' ou internet des objets).

Voici à quoi ressemble un exemple de diagramme Node-Red :



Chaque nœud ('**Node**') du diagramme ('**Flow**') gère un flux de données ('**payload**') et cache en fait le plus souvent des lignes de code écrites en JavaScript.

En effet, Node-Red est une application reposant sur '**Node.js**', c'est-à-dire la programmation de flux du langage Javascript, mais l'idée est d'utiliser des fonctions graphiques de haut niveau qui ne nécessitent pas souvent de connaître le langage Javascript.

Dans notre cas, on se limitera à une découverte de cet environnement de développement et de ce langage graphique, mais cette simple découverte va nous permettre de réaliser des applications qui nécessitent pas mal de lignes de code dans les exercices du TP n°4.

## Installation de Node-Red :

Sur la version de l'OS Raspbian que je vous ai installé sur vos cartes SD, Node-Red est déjà installé, vous n'avez donc rien à faire pour le moment.

A noter que s'il n'était pas installé, il existe une façon très rapide de le faire en passant par le menu 'Démarrer' et en allant chercher '**Recommended Software**' dans les préférences.

## Récupération des fichiers utiles pour cette séance :

Pour cela, tapez la commande suivante dans le terminal :

```
$ git clone https://github.com/MbottiniIUT/IUT\_IPE\_TP5.git
```

[rem : il n'y a pas d'espace dans ce lien, juste des '\_']

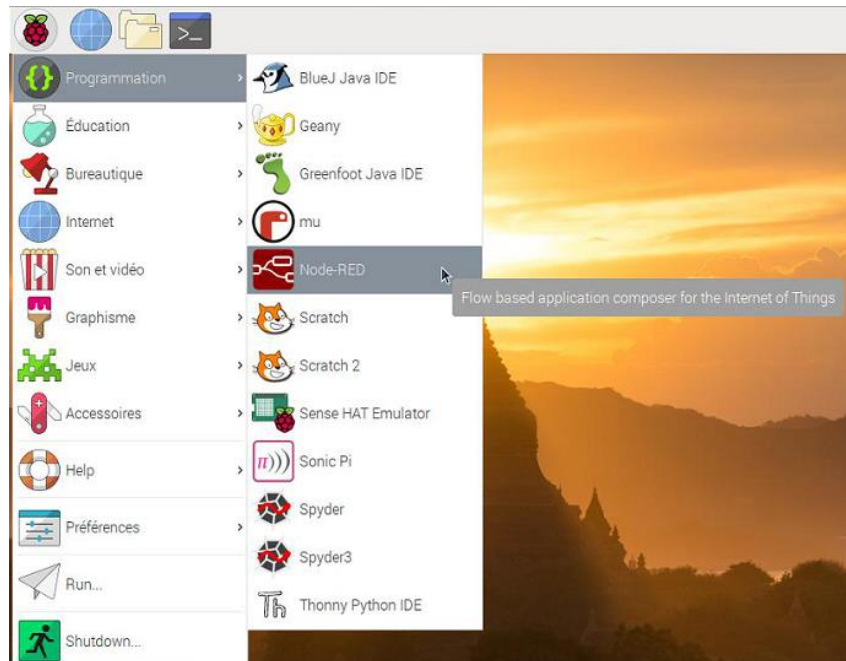
Une fois la tâche effectuée, vous devriez retrouver, en ouvrant un explorateur de fichiers, un dossier '**IUT\_IPE\_TP5**' sous '**/home/pi**'.

## TUTO 1 : 'Tuto 01.json'

Ce tutoriel va être assez long car la prise en main d'un nouvel outil, même graphique, demande un peu de temps.

### Lancement de « Node-Red » :

Le lancement du serveur se fait via le menu « démarrer » dans la section « programmation » :



Dès que vous avez cliqué, un terminal s'ouvre affichant les informations propres au lancement du serveur 'Node-Red'. Au bout de quelques dizaines de secondes, l'affichage se fige sur quelque chose comme ceci :

```
Node-RED console
Fichier  Édition  Onglets  Aide

Start Node-RED

Once Node-RED has started, point a browser at http://192.168.1.22:1880
On Pi Node-RED works better with the Firefox or Chrome browser

Use  node-red-stop      to stop Node-RED
Use  node-red-start     to start Node-RED again
Use  node-red-log       to view the recent log output
Use  sudo systemctl enable nodered.service to autostart Node-RED at every boot
Use  sudo systemctl disable nodered.service to disable autostart on boot

To find more nodes and example flows - go to http://flows.nodered.org

Starting as a systemd service.
```


```
Node-RED console
Fichier  Édition  Onglets  Aide

NODE_MODULE_VERSION 64. Please try re-compiling or re-installing
the module (for instance, using 'npm rebuild' or 'npm install').
20 Aug 17:40:16 - [warn] -----
20 Aug 17:40:16 - [info] Settings file   : /home/pi/.node-red/settings.js
20 Aug 17:40:16 - [info] Context store   : 'default' [module=memory]
20 Aug 17:40:16 - [info] User directory  : /home/pi/.node-red
20 Aug 17:40:16 - [warn] Projects disabled : editorTheme.projects.enabled=false
20 Aug 17:40:16 - [info] Flows file      : /home/pi/.node-red/flows_raspberrypi.js
20 Aug 17:40:16 - [info] Creating new flow file
20 Aug 17:40:16 - [warn] -----
Your flow credentials file is encrypted using a system-generated key.
If the system-generated key is lost for any reason, your credentials
file will not be recoverable, you will have to delete it and re-enter
your credentials.
You should set your own key using the 'credentialSecret' option in
your settings file. Node-RED will then re-encrypt your credentials
file using your chosen key the next time you deploy a change.
20 Aug 17:40:16 - [info] Server now running at http://127.0.0.1:1880/
20 Aug 17:40:16 - [info] Starting flows
20 Aug 17:40:16 - [info] Started flows
```

**Vous ne devez jamais fermer cette fenêtre sous peine  
de ne plus pouvoir travailler avec 'Node-Red' !!**



Vous pouvez alors ouvrir un navigateur (vous pouvez faire le choix de travailler en local sur la Raspberry ou bien à distance sur le PC de votre poste) et y taper l'adresse IP et le port suivant :

 127.0.0.1:1880

Si vous êtes en local sur la Raspberry

 148.60.45.xx:1880

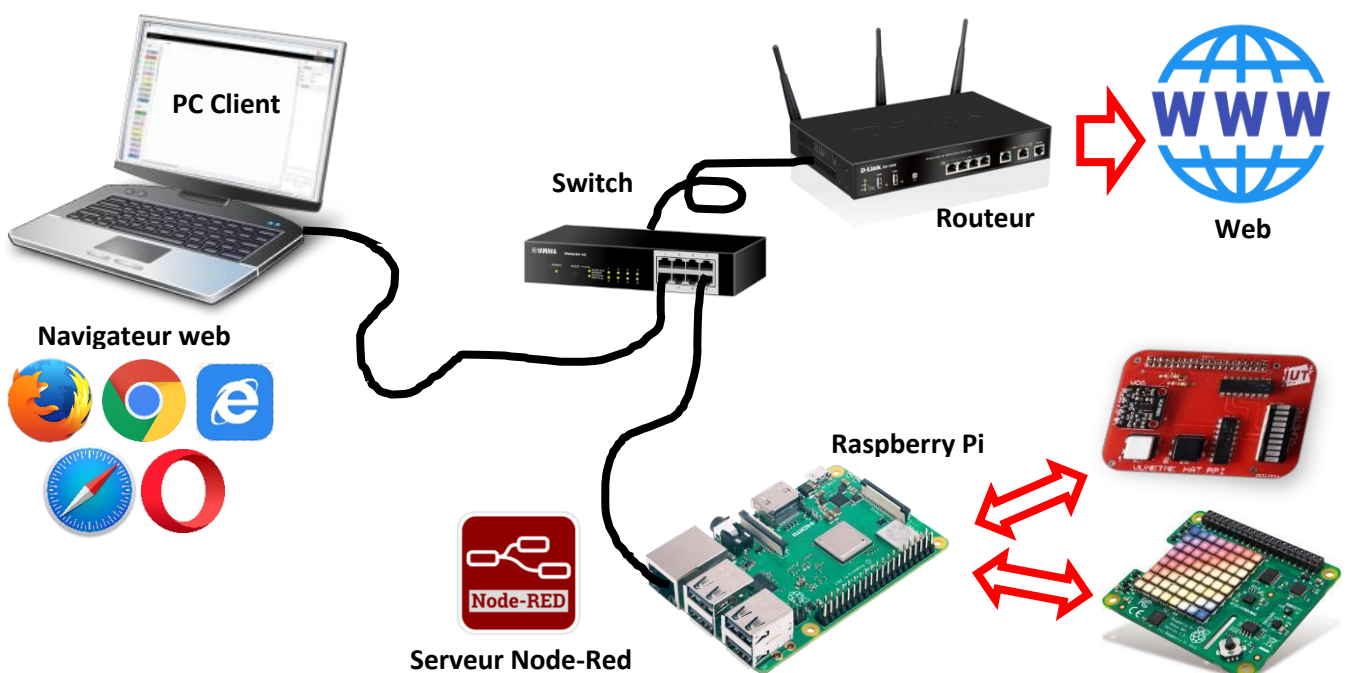
Si vous êtes à distance côté PC

(Bien sûr, remplacer le '148.60.45.xx' par votre propre IP)

Dès lors, votre navigateur doit vous ouvrir l'interface suivante :



Nous allons forcément la détailler par la suite, mais il convient tout d'abord de bien comprendre ce que l'on fait. Pour cela, rien de tel qu'un graphique :



Côté Raspberry, on a juste un serveur Node-Red, qui tout comme Flask, est capable de communiquer avec du matériel et également d'utiliser un grand nombre de protocoles dédiés au web ou d'exécuter du code Javascript (Node.js).

Côté Client (PC/MAC), on dispose dans le navigateur web d'une IHM (interface homme machine) où l'utilisateur peut visualiser des données provenant du serveur (GET) ou agir sur le serveur (POST). Et ce serveur peut interagir de la même façon avec un client côté web. Cela permet donc des applications complexes de transfert d'informations.

Et puisque l'on travaille sur le réseau, IBM a eu l'idée de rendre disponible son environnement de développement graphique via un navigateur.

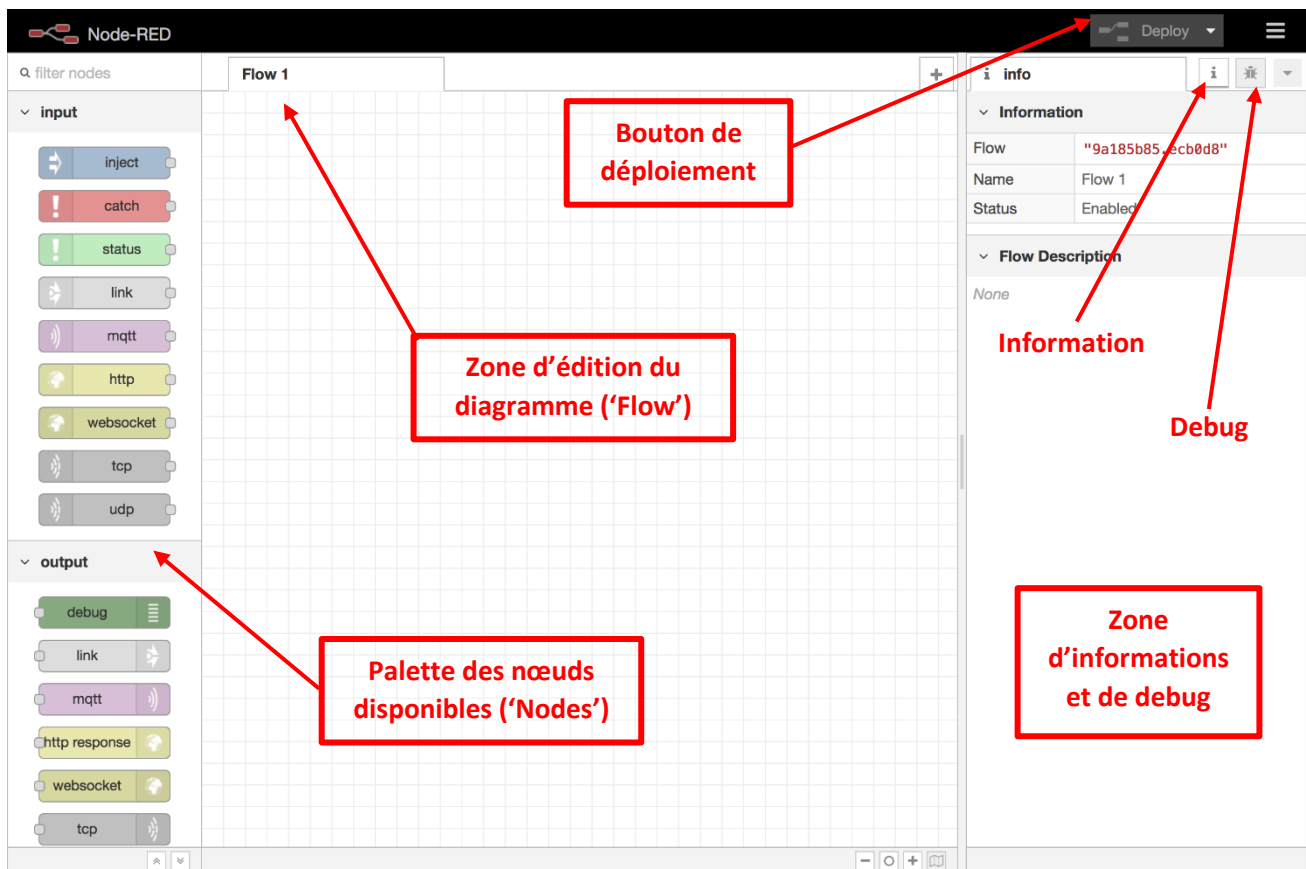
Au final, on peut développer son application côté Client, puis la déployer vers le Serveur et enfin utiliser cette application côté Client.

Voilà grossièrement la philosophie de Node-Red.

## L'interface de « Node-Red » :

Vous trouverez, au besoin, une documentation complète à l'adresse suivante : <https://nodered.org/docs/>

Découvrons maintenant plus précisément l'interface de développement :



La démarche de programmation est la suivante :

1. On dépose dans le diagramme les nœuds dont on a besoin
2. On édite leurs propriétés
3. On les relie entre eux
4. On déploie l'application
5. On vérifie son fonctionnement dans la zone de 'debug'

## Un premier diagramme :

On ne va révolutionner le monde avec ce premier exemple, mais il va nous permettre d'apprendre les bases. Il s'agit du 'Hello World' de Node-Red. Pour le moment, on ne va pas se connecter au hardware.

Utiliser la palette pour déposer dans votre diagramme ('flow') deux nœuds ('nodes') : un nœud d'entrée '**inject**' et un nœud de sortie '**debug**' :




Connectez ensuite ces deux éléments entre eux en cliquant sur les deux connecteurs (petits carrés gris) :

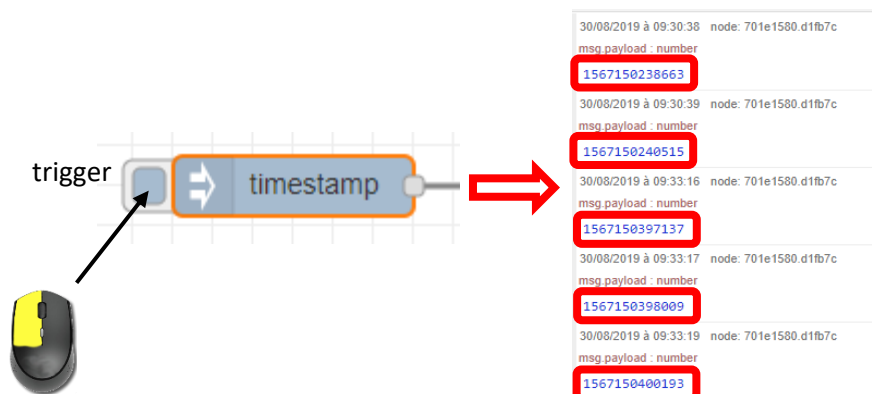


On voit apparaître deux termes : '**timestamp**' et '**msg.payload**'

On va décrire un peu ces deux termes une fois que l'on aura testé notre application.

Ouvrir dans la zone d'information l'onglet de '**debug**' via l'icône  et puis cliquez sur le bouton '**deploy**' pour déployer votre programme vers le serveur Node-Red sur la Raspberry.

Cliquez alors sur le carré devant le '**Timestamp**' et vous devriez voir apparaître dans la zone de '**debug**' un nombre qui augmente à chaque clic :

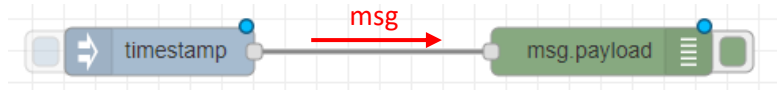


En fait, '**timestamp**' émet donc un message à chaque appui sur son déclencheur ('**trigger**') et ce message contient un nombre. Ce nombre est intimement lié à l'heure actuelle qu'affiche votre Raspberry. Il s'agit du nombre de millisecondes écoulées depuis le 1<sup>er</sup> janvier 1970 00:00:00 UTC. Ce format, s'il peut être pratique pour une machine lorsqu'elle doit déterminer un temps entre deux mesures, n'est pas vraiment adapté à nos habitudes.

Il n'existe pas de moyen direct pour changer ce format de données. Mais il existe des bibliothèques qui ajoutent des nœuds qui offrent le moyen de le faire (ex : <https://flows.nodered.org/node/node-red-contrib-moment>). Mais pour le moment, on va se concentrer sur la compréhension générale.

Qu'est-ce que '**msg.payload**' ?

On l'a évoqué, '**Node-Red**' travaille avec un flux de données tout comme d'autres langages de programmation graphique comme Labview par exemple. '**msg**' est l'objet qui constitue le flux ici.



Un objet **'msg'** quitte donc **'timestamp'** pour arriver dans **'msg.payload'** à chaque clic de souris sur le **'trigger'**. **'payload'** constitue en fait un attribut (le plus utile pour nous) de l'objet **'msg'**.

Si l'on double clique sur **'timestamp'** pour afficher ses propriétés, on voit apparaître dans le champ **'payload'** le choix **'timestamp'**, ce qui veut dire que la partie informative de **'msg'** contiendra l'heure en ms.

Modifiez maintenant l'attribut **'payload'** en choisissant **'string'** puis entrez dans le champ dédié une chaîne de caractères quelconque :

Déployez à nouveau votre application et visualisez dans la zone **'debug'** ce qui apparaît lors de clics sur le **'trigger'**.

```
30/08/2019 à 15:20:55 node: 701e1580.d1fb7c
msg.payload : string[8]
"IUT GEII"
30/08/2019 à 15:21:00 node: 701e1580.d1fb7c
msg.payload : string[8]
"IUT GEII"
30/08/2019 à 15:21:00 node: 701e1580.d1fb7c
msg.payload : string[8]
"IUT GEII"
```

Remarque : on constate que le type de **'msg.payload'** a changé. On avait un nombre, maintenant on a une chaîne de caractères

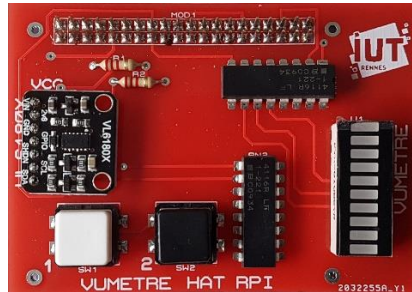
Par curiosité mais également pour vérifier que **'msg'** est bien un objet avec différents attributs, vous pouvez éditer le nœud **'msg.payload'** (double clic pour éditer ses propriétés) et changer ce que l'on désire visualiser. Au lieu de visualiser uniquement la partie utile (i.e. **'payload'**), choisissez de visualiser l'objet **'msg'** en entier :

Pensez toujours à déployer pour vérifier le fonctionnement de votre application.



## PREPARATION DU HARDWARE :

Dans cette première partie du TP, nous allons utiliser la carte d'extension Vumètre (Cf. TP n°2) pendant quelques exercices :



**Pour l'installer, vous devez quitter l'OS via le menu 'démarrer' puis débrancher l'alimentation de la Raspberry !!**

Puis redémarrer la Raspberry, relancez le serveur 'Node-Red' et vous reconnecter via votre navigateur web avant de poursuivre.

## EXERCICE 1 : 'Flow\_01.json'

Pour éviter des explications et des manipulations supplémentaires, je ne vais pas vous expliquer comment créer de nouveaux diagrammes et comment gérer leurs déploiements.

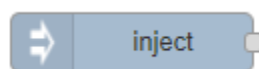
Vous allez donc effacer le contenu de votre précédent diagramme pour aborder cet exercice : il suffit de tout sélectionner et d'appuyer sur la touche '**suppr**'.

### Objectif :

Dans cet exercice, vous allez maintenant créer une application qui vous permettra de commander une des Leds du vumètre via votre diagramme.

### Nœuds utiles et infos :

Pour piloter la led depuis votre diagramme, vous aurez besoin de deux entrées '**inject**' :



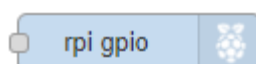
Chacune utilisera en '**payload**' un nombre.

Pour une des deux entrées, la valeur du nombre sera 0 et pour l'autre 1.

Pour bien les distinguer, on va modifier le champ '**name**'.

Pour celle où le '**payload**' est 0, on va mettre OFF et pour l'autre ON.

Et vous aurez également besoin d'un nœud permettant de commander une broche du connecteur GPIO de la raspberry. Il s'agit ici d'une sortie '**rpi.gpio**' :



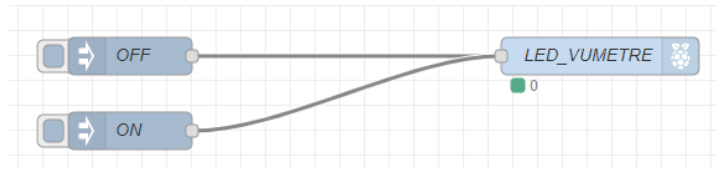
Vous choisirez dans ses propriétés une des broches où est câblée une led du vumètre (cf. TP n°2 pour le brochage).

Le type est sur '**digital output**'.

On peut l'initialiser à 0.

Et dans l'attribut '**name**', vous pouvez mettre LED\_VUMETRE.

Pour les connecter, vous allez pouvoir câbler les deux sorties des nœuds **ON** et **OFF** vers l'entrée du nœud '**rpi.gpio**' :

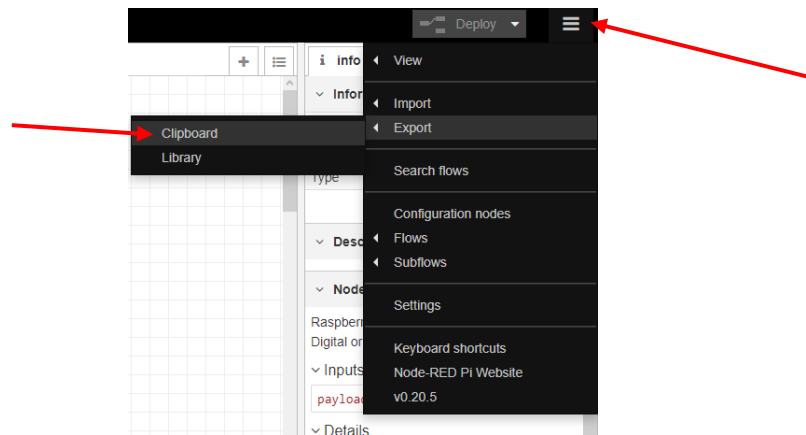


Testez votre application et faites-la contrôler par un enseignant.

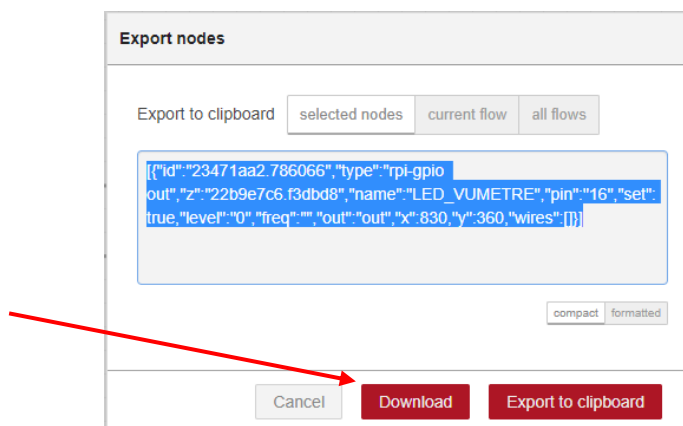
Remarque : on peut se poser la question suivante : comment deux sorties différentes peuvent-elles être câblées sur une seule entrée sans générer de conflit ? Cela est simplement lié au fait que l'on traite des flux de données.

L'objet '**msg**' de OFF et celui de ON ne sont jamais générés simultanément puisque l'on ne peut pas appuyer sur chacun de leurs '**trigger**' en même temps. En conséquence, sur l'entrée LED\_VUMETRE, un seul message arrive à la fois et donc un seul message n'est traité à la fois.

Remarque 2 : Je n'ai pas évoqué comment enregistrer votre diagramme pour une utilisation ultérieure. Il suffit d'aller le menu sur la droite de la page web et de choisir '**Export**' puis '**Clipboard**' :



Dans la fenêtre suivante, il suffit de cliquer sur '**Download**' :



Le fichier qui va être sauvegardé sera avec l'extension '**JSON**'.

Ce n'est pas du tout une extension propriétaire à IBM, mais plutôt d'un format d'échange de données sur le web très populaire.

A partir du moment où des données transitent sur le web et sont échangées entre des serveurs et des clients, il fallait créer un format commun pour tout ce qui est texte (j'exclue ici les données binaires et les bases de données qui ont des formats spécifiques également). En fait, de nombreux formats ont vu le jour depuis la démocratisation d'internet : CSV, RSS, XML, YAML, JSON, ...

Mais ces dernières années, le format **JSON** (Javascript Object Notation) se fait de plus en plus présent au point de faire disparaître progressivement les autres.

Pour le moment, je ne vous en dit pas plus car on en reparlera dans d'autres TP, mais sachez donc que le diagramme graphique que vous avez établi est tout simplement traduit en un fichier texte qui porte l'extension **JSON**.

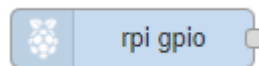
## EXERCICE 2 : 'Flow 02.json'

### Objectif :

Dans ce nouvel exercice, on va maintenant commander une des Leds du vumètre via un bouton de notre carte d'extension 'vumètre'.

### Nœuds utiles et infos :

Pour piloter la led, on va donc maintenant utiliser un des boutons de la carte d'extension, il nous faut donc un nœud qui nous permet d'y accéder. C'est également, comme celui de la led, un nœud '**rpi.gpio**', mais cette-fois-ci, c'est une entrée :



Vous choisirez dans ses propriétés une des deux broches correspondant aux deux boutons de notre carte d'extension (cf. TP n°2 pour le brochage).

Vous préciserez qu'il y a une résistance de pull-up et vous nommerez le nœud 'BOUTON'

Pour la gestion de la led, c'est le même nœud que dans l'exercice précédent qui est utilisé.

Vérifiez le bon fonctionnement de votre application et faites-la contrôler par une enseignant.

## TUTO 2 : 'Tuto 02.json'

Dans l'exercice précédent, vous avez pu remarquer que comme la résistance est en pull-up, l'allumage de la led est en logique inversée par rapport au bouton.

Là aussi, il n'y a pas d'outil direct pour modifier notre diagramme afin de corriger ce comportement.

On va donc profiter de ce tutoriel pour voir comment installer de nouveaux nœuds.

Quand on pense ne pas trouver ce que l'on cherche dans la palette existante, la première chose à effectuer est une recherche sur le site d'IBM qui recense toutes les publications officielles (on trouve également beaucoup de non officielles sur github).

Dans un nouvel onglet de votre navigateur, rendez-vous à l'adresse suivante : <http://flows.nodered.org>

Dans la zone de recherche, proposez un mot clef en lien avec votre recherche, ici 'logic' ou 'boolean', en cochant uniquement 'nodes' :

## Node-RED Library

Find new nodes, share your flows and see what other people have done with Node-RED.

☐ flows ☒ nodes 9 of 3491 things

Sort by:  
☒ recent  
☐ downloads  
☐ rating

**node-red-contrib-boolean-logic-ultimate**  
A set of Node-RED enhanced boolean logic, with persist values after reboot and more  
v1.0.6 126 node

**node-red-contrib-hal**  
A set of nodes to help with basic home automation logic  
v1.0.1 25 node

**node-red-contrib-dsm**  
A dynamic state machine node based on the finite state machine model.  
v0.14.1 175 ★5.0 (1) node

**node-red-contrib-nucleus-ladder**  
Tyron's Node-RED nodes supporting common ladder logic functions.  
v1.1.1 9 node

**snappy-logic-nodes**  
Node red nodes for logic for snappy  
v0.5.273 4 node

**node-red-kuromoji**  
A Node-RED node of Japanese morphological analyzer  
v0.0.4 58 node

**node-red-contrib-bool-gate**  
Node RED node to include boolean logic gates  
v1.0.2 175 node

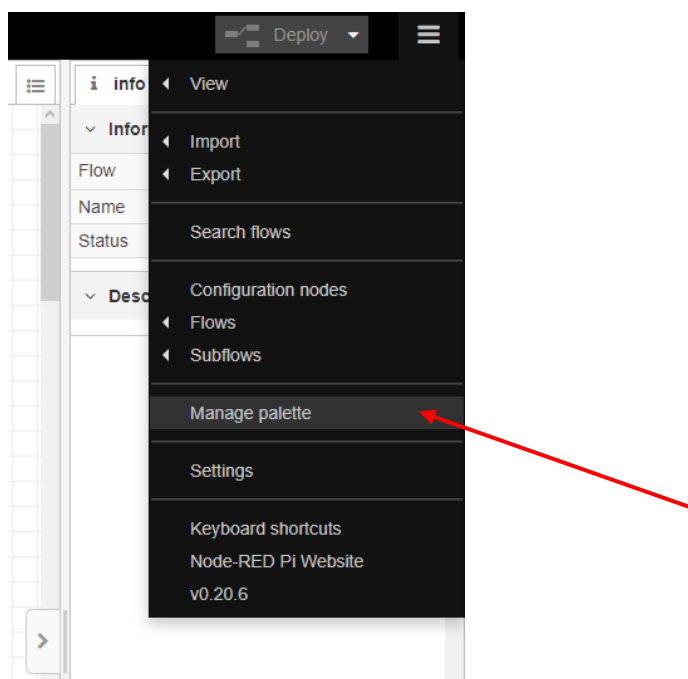
**node-red-contrib-boolean-logic**  
A set of Node-RED nodes for boolean logic  
v0.0.3 212 ★5.0 (1) node

**node-red-contrib-sh-logic**  
Logic Nodes for smarhome requirements  
v0.0.3 7 node

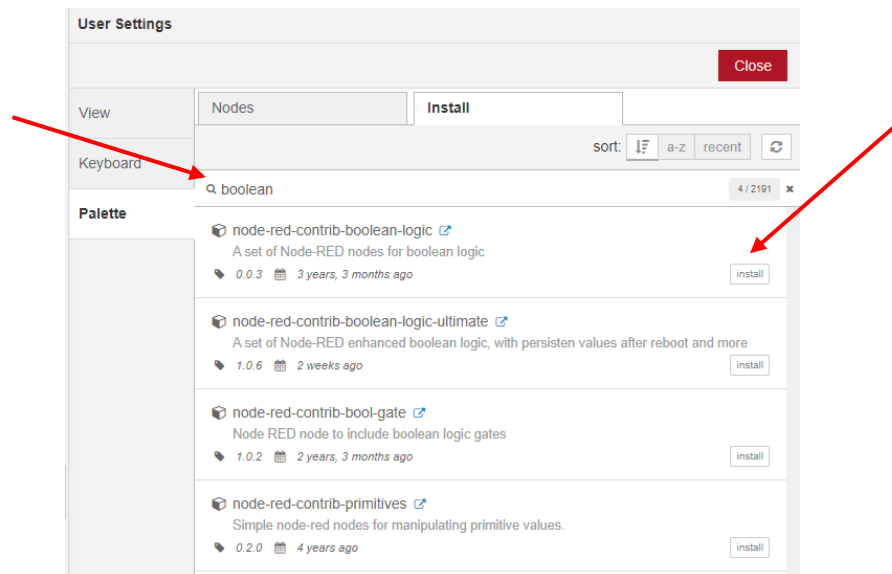
Plusieurs librairies semblent convenir. La description de chacune (clic sur la case pour ouvrir la page web dédiée), le nombre de téléchargement 📁 ou encore le notation ★ peuvent nous aider dans notre choix.

Ici, on va utiliser la librairie '**node-red-contrib-boolean-logic**'.

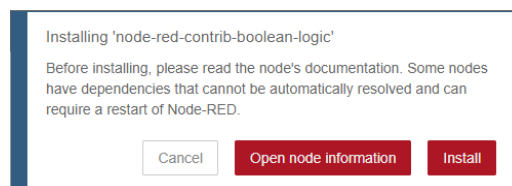
Pour l'installer, il faut dans Node-Red ouvrir la palette de gestion des nœuds via le menu en haut à droite :



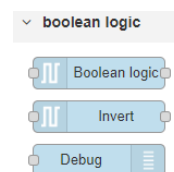
Dans la zone qui vient d'apparaître, vous choisissez l'onglet **'install'** et vous tapez un des mots clefs contenu dans le nom de la librairie, par exemple **'boolean'** :



Vous cliquez alors sur le bouton **'install'** de la librairie qui vous intéresse. Un message de confirmation vous est demandé. Vous cliquez sur **'install'** à nouveau :



L'installation dure quelques dizaines de secondes puis un message vous indique quels sont les nœuds qui ont été ajoutés :



Pour corriger notre souci de logique inversé, on a juste besoin du nœud **'invert'** :



Il n'y a rien à configurer pour ce nœud.  
Il se contente d'inverser le message d'entrée  
(0 ↔ 1 ou True ↔ False).

Vérifiez le bon fonctionnement de votre diagramme avant de passer à la suite.



## EXERCICE 3 : 'Flow\_03.json'

### Objectif :

Dans cet exercice, on va continuer commander notre led du vumètre par un bouton, mais cette fois-ci, celui-ci sera une page web côté client.

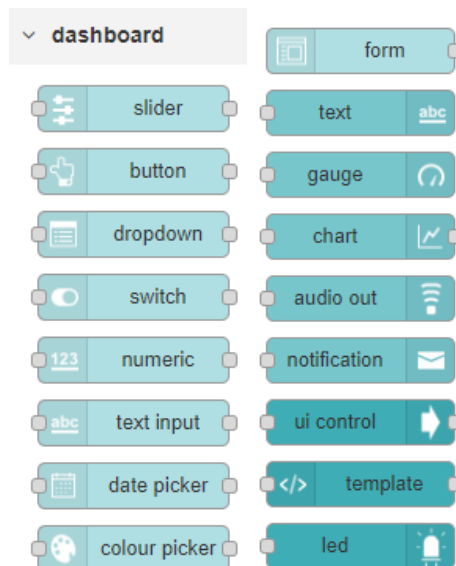
### Nœuds utiles et infos :

Pour créer une page web qui permet de contrôler notre led par le réseau, vous allez voir que cet outil graphique cherche vraiment à simplifier la vie des développeurs.

En effet, sous Node-Red, on a déjà un serveur qui tourne et une interface dans un navigateur web. Il ne nous manque donc juste de rendre un peu plus conviviale cette interface.

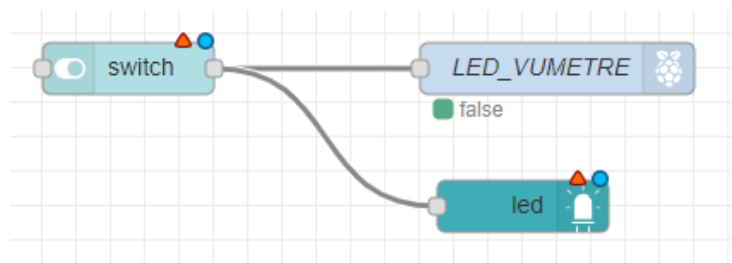
Pour cela, vous allez commencer par installer la librairie '**node-red-dashboard**' ainsi que '**node-red-contrib-ui-led**'.

Une fois terminée, vous pouvez constater que de nombreux nœuds ont été ajoutés :



On ne va pas tous les détailler ici, mais les noms sont plutôt évocateurs.

On va donc utiliser le nœud '**switch**' et le nœud '**led**'. Ajoutez-les à votre diagramme tout en conservant le nœud '**LED\_VUMETRE**' (vous retirez tout le reste) :

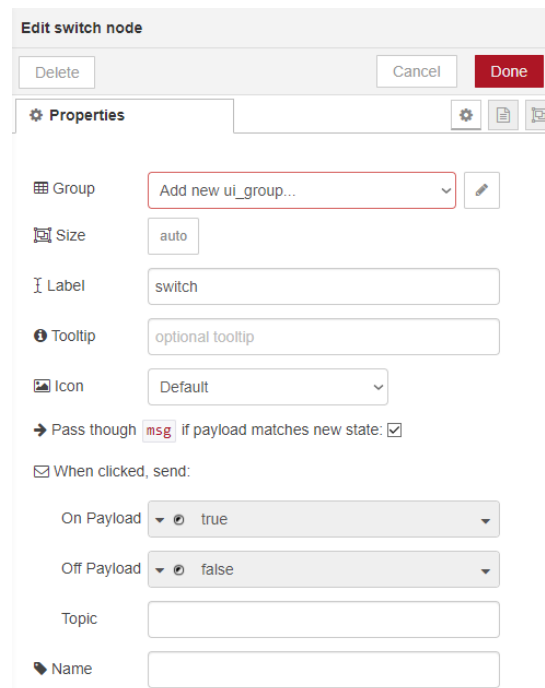


**Remarque intermédiaire :** les informations de couleur qui apparaissent nous donnent des indications importantes.

Ainsi, le rond bleu nous indique que ce nœud n'a pas encore été déployée sur l'application en train de tourner.

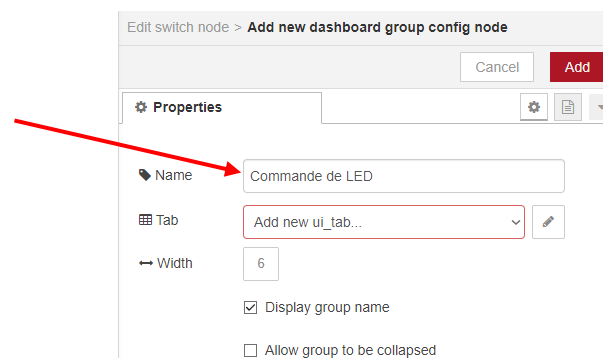
Le triangle orangé nous indique quant à lui, que la configuration du nœud est incomplète.

On va donc terminer la configuration des deux nouveaux nœuds que l'on a déposés dans notre diagramme. Commençons par le 'switch' :

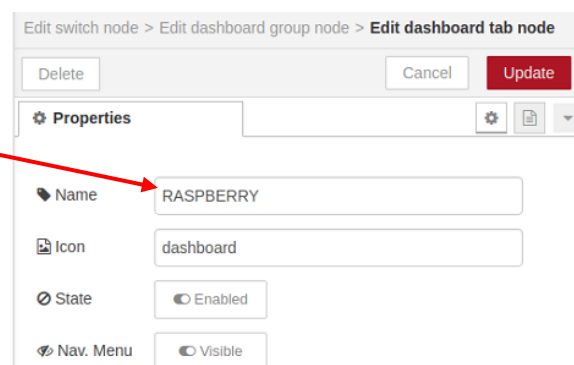


La seule chose à modifier est indiqué en rouge, c'est le champ '**group**'. En effet, notre page web pourrait comporter un très grand nombre d'éléments. Pour les organiser de façon agréable, il faut créer des groupes.

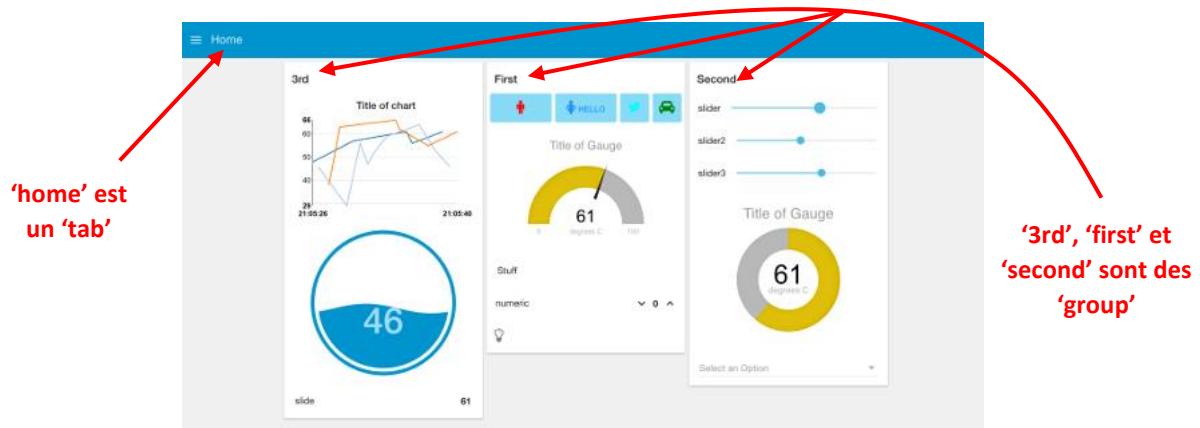
On va donc ici créer un groupe '**Commande d'une LED**' en cliquant sur l'icône en forme de crayon  :



Cela ne suffit pas, il faut de la même façon créer un onglet '**tab**' que l'on va appeler '**RASPBERRY**' et qui contiendra le groupe précédent :




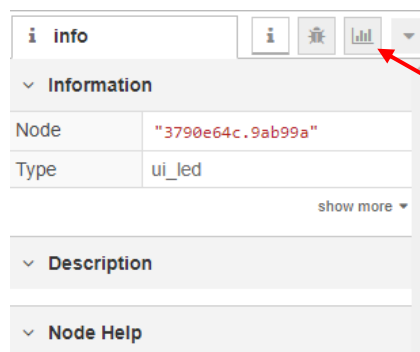
A quoi correspondent ces notions 'group' et 'tab' ? Simplement à l'organisation de votre interface web. Prenons un exemple issu de la page de la librairie 'dashboard' :




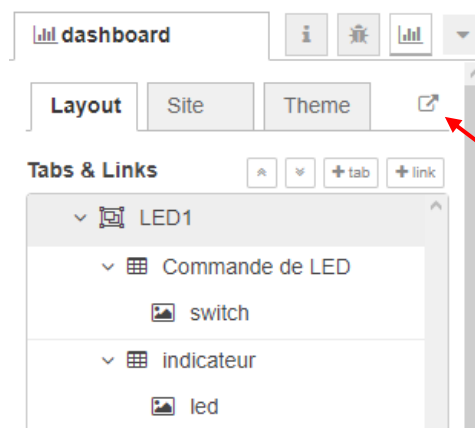
Maintenant que notre diagramme est terminé et configuré, vous pouvez le déployer.

Mais maintenant, nous disposons d'une interface web graphique sur laquelle nous pouvons agir pour piloter notre led depuis un bouton web.

Pour accéder à cette page, appelée '**dashboard**', il vous faut cliquer sur l'icône  à côté du '**debug**' :



Une fois cet onglet activé, il vous suffit de cliquer sur l'icône  pour ouvrir ce 'dashboard' dans un nouvel onglet de votre navigateur.



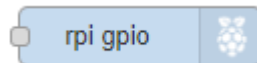
## EXERCICE 4 : 'Flow\_04.json'

### Objectif :

On va compléter l'interface de l'exercice précédente en y ajoutant un curseur avec glissière qui nous permettra de faire varier la luminosité d'une autre LED de notre vumètre.

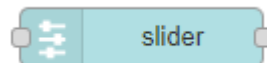
### Nœuds utiles et infos :

En plus des nœuds précédents de l'exercice 3, vous allez devoir ajouter une LED :



Le choix de la broche doit bien sûr correspondre à une LED du vumètre et être différente de celle déjà utilisée.  
Elle doit également être de type 'PWM output'.

Vous allez également avoir besoin du nœud 'slider' :

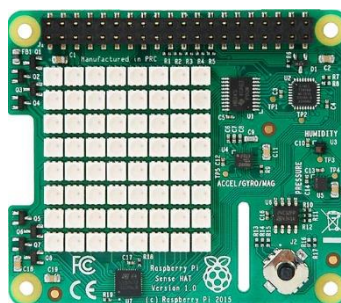


Ajuster ses paramètres pour qu'ils correspondent à ceux du signal PWM qui varie entre 0 et 100.

**Faites contrôler le bon fonctionnement de cet exercice par un enseignant.**

## PREPARATION DU HARDWARE :

Dans cette seconde partie du TP, nous allons utiliser la carte d'extension Sense Hat (Cf. TP n°3&4) :



**Vous devez tout d'abord quitter l'OS via le menu 'démarrer' puis débrancher l'alimentation de la Raspberry !!**

**Retirer ensuite délicatement la carte d'extension 'vumètre' et remplacez-la par la carte d'extension 'Sense Hat'**

Redémarrer ensuite la Raspberry, relancez le serveur '**Node-Red**' et vous reconnecter via votre navigateur web avant de poursuivre.

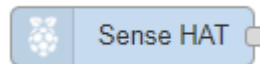
## EXERCICE 5 : 'Flow\_05.json'

### Objectif :

On va commencer par créer un diagramme qui nous permet de récupérer les informations d'environnement (température/humidité/pression) mesurées par la carte et de les afficher dans la zone de 'debug'.

### Nœuds utiles et infos :

Vous allez avoir besoin d'utiliser bien évidemment le nœud '**debug**' comme nous l'avons déjà fait plusieurs fois. Pour la carte '**Sense Hat**', rechercher le nœud suivant dans la catégorie '**Raspberry**' :



Configurer ce nœud pour ne mesurer que les grandeurs d'environnement.

**Remarque :** s'il ce nœud n'apparaît pas dans la liste, c'est que la librairie correspondante n'est pas installée.

Utilisez alors l'outil de téléchargement pour installer la librairie '**node-red-node-pi-sense-hat**'.

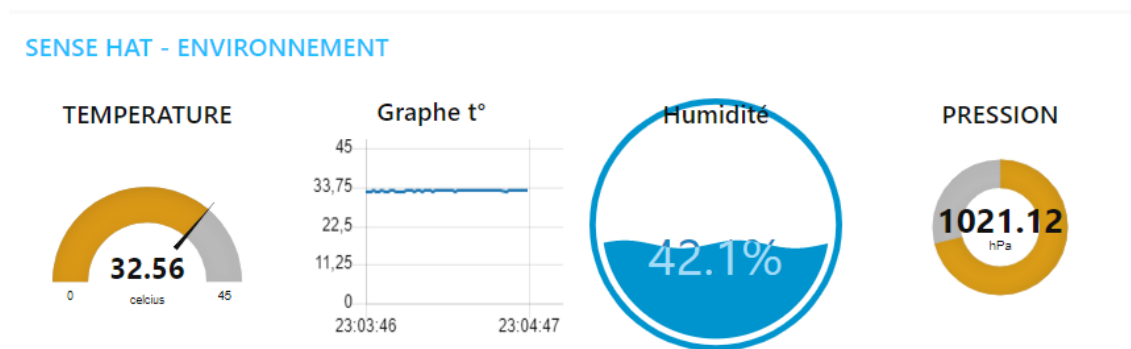
Une fois votre diagramme terminé, déployez-le et observez les informations arrivant dans la zone de 'debug'. Contrôlez leur cohérence et regardez bien le format des données.

## EXERCICE 6 : 'Flow\_06.json'

### Objectif :

On va modifier le diagramme précédent pour afficher cette fois-ci les informations d'environnement (température/humidité/pression) mesurées par la carte dans le '**dashboard**'.

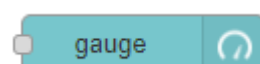
Vous essayerez de vous approcher de l'aspect visuel suivant :



### Nœuds utiles et infos :

Vous pouvez conserver les deux nœuds de l'exercice précédent.

Vous allez devoir y ajouter 3 nœuds '**Gauge**' :



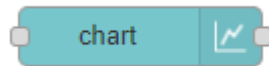


Le type différera selon la grandeur mesurée :

- **'gauge'** pour la température
- **'donut'** pour la pression
- **'level'** pour l'humidité

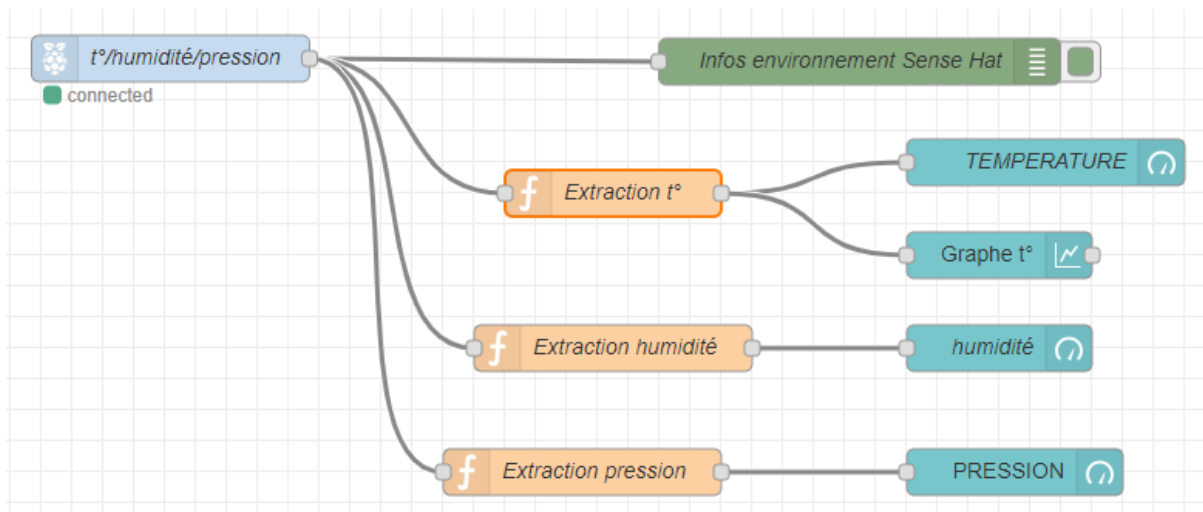
Bien sûr, vous adapterez les échelles pour rendre l'affichage le plus lisible possible.

Vous aurez également besoin du nœud **'Chart'** :



Vous utiliserez le type **'Line Chart'** avec un enregistrement des 60 dernières secondes.  
Là aussi vous adapterez l'échelle en Y.

Voici le diagramme que vous devez saisir :

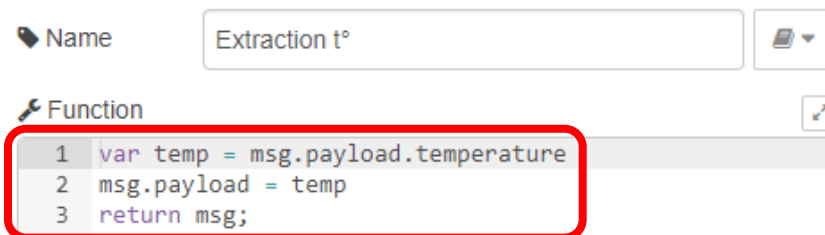


On reconnaît donc tous les nœuds précédemment évoqués sauf 3 d'entre eux. Il s'agit pour chacun d'un nœud **'function'**. Le message **'msg.payload'** récupéré de la 'Sense Hat' est en fait un objet contenant (cf. zone 'debug' de l'exercice précédent) l'ensemble des 3 grandeurs. Le nœud 'function' va donc nous permettre, en écrivant un peu de code, d'extraire chaque donnée.

Toutefois **'Node-Red'** est construit sur **'Node-js'** et donc le langage utilisé ici n'est pas le python mais plutôt le javascript.

Il ne s'agit donc pas ici de vous former sur ce nouveau langage. Je vous fournirai donc les lignes de code nécessaires à chaque fois.

Par exemple, pour extraire la température, il suffit d'écrire les lignes suivantes dans la configuration du nœud 'function' correspondant :



Vous pourrez facilement adapter ce code aux deux autres nœuds nécessaires pour extraire la pression et l'humidité.

**Faites contrôler le bon fonctionnement de l'ensemble par un enseignant.**

## EXERCICE 7 : 'Flow 07.json'

### Objectif :

Le but de cet exercice va être de reproduire notre niveau à bulles électronique du TP n°3 mais cette fois-ci, l'affichage ne se fera plus sur la matrice mais sur une page web.

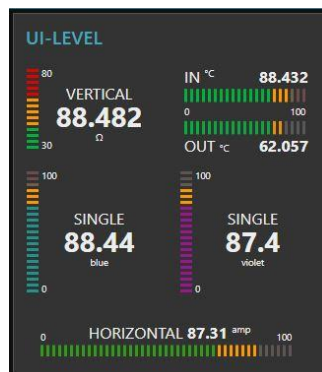
### Installation d'une librairie supplémentaire :

Dans les nœuds actuellement disponibles dans la catégorie 'dashboard', aucun ne serait vraiment adapté à la matérialisation d'un niveau à bulles. On va donc à nouveau utiliser l'outil d'ajout de librairies pour enrichir notre palette. Installer donc la librairie '**node-red-contrib-ui-level**'.

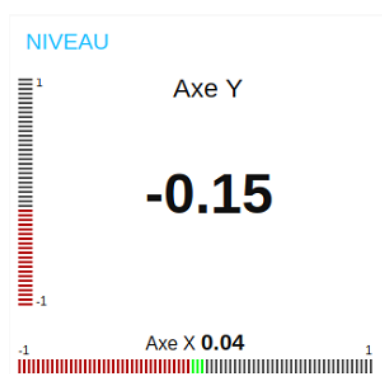
Cette librairie va ajouter dans la catégorie '**dashboard**' le nœud '**level**' :



La page web (<https://flows.nodered.org/node/node-red-contrib-ui-level>) nous indique ce que propose ce nœud côté '**dashboard**' :



On va donc pouvoir utiliser deux '**ui-level**', un vertical pour l'axe Y et un horizontal pour l'axe X de manière à obtenir la représentation suivante sur la page web :



**Diagramme :**

Votre diagramme doit ressembler à ceci :

**Configuration des 'level' :**

Les deux 'levels' doivent, conformément aux infos fournies par la 'Sense Hat', travailler sur la plage -1/+1.

L'un doit être en 'single vertical', l'autre en 'single horizontal'.

La zone centrale doit être de couleur verte (entre -0.05 et +0.05) et ailleurs rouge.

**Configuration de 'Sense Hat' :**

On ne s'intéresse maintenant qu'aux 'motion events'.

**Configuration des 'function' :**

On se rapproche beaucoup de ce que l'on a fait dans l'exercice précédent. Par exemple, pour l'accélération sur l'axe X, le code est le suivant :

```

Name      Extraction accel X

Function
1  var accelx = msg.payload.acceleration.x
2  msg.payload = accelx
3  return msg;

```

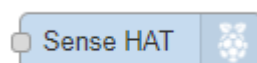
**Faites contrôler le bon fonctionnement de l'ensemble par un enseignant.**

## TUTO 3 : 'Tuto 03.json'

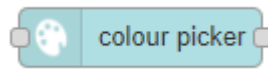
Dans ce petit tutoriel, on va voir comment changer la couleur de la matrice 'Sense Hat' via le 'dashboard'.

Comme d'habitude, effacez la totalité de votre diagramme pour démarrer avec un diagramme vierge.

Le nœud qui nous permet de gérer la matrice est le nœud 'Sense hat output' :



Et pour saisir une couleur dans une page web, rien de plus pratique qu'un '**colour picker**' que vous utilisez régulièrement dans vos traitements de texte ou vos logiciels de graphisme et de photos.  
Voici à quoi il ressemble :



Instinctivement, on aurait envie de faire le diagramme suivant :



Mais cela ne fonctionnera pas. Je vais essayer de vous expliquer pourquoi de manière à vous donner un peu plus d'autonomie sur 'Node-Red'.

Pour cela, il faut prendre le temps de lire les informations dans le volet de droite pour chacun de ces deux nœuds :

On voit donc que la seule façon pour qu'ils puissent se comprendre, c'est qu'ils utilisent le même format pour la couleur. Les seuls en commun sont : RGB et HEX.

Ensuite le message que doit recevoir le nœud 'Sense Hat' doit être de la forme '**\*,\*,couleur**'.

Il faudra donc trouver le moyen d'insérer les caractères '**\*,\*,**' en début de message.

Dans la configuration du '**color picker**', choisissons par exemple le format '**HEX**' :

**Edit colour picker node**

Delete Cancel Done

**Properties**

Group: [Home] SENSE HAT - Matrice

Size: auto

Label: Choix de la couleur

Format: **hex** (selected) round

Show hue slider: ☒

Show lightness slider: ☒

If width is 4 or greater:

Always show swatch: ☒

Always show picker: ☒

Always show value field: ☒

→ If **msg** arrives on input, pass through to output: ☒

Send: one value when released/closed

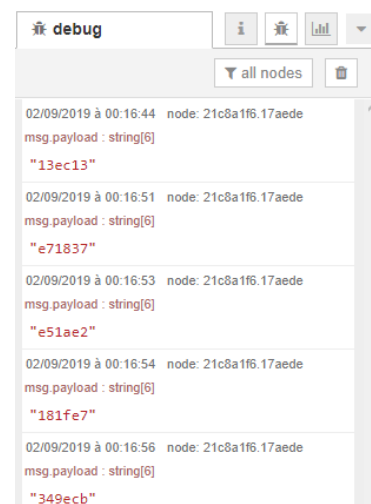
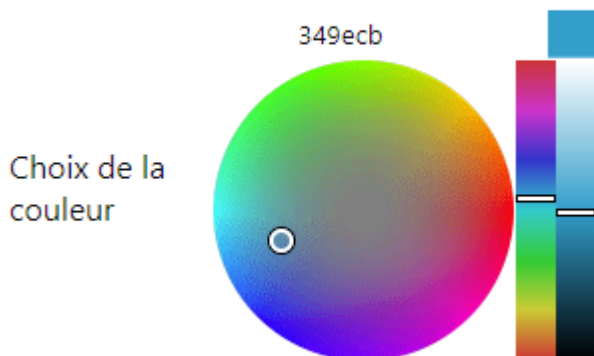
Payload: current value as a string

Topic: optional topic

Name: COULEUR

Déployons le diagramme en y ayant ajouté un nœud de 'debug' à la sortie du 'color picker'. On constate alors qu'en changeant la couleur dans le 'dashboard', le format de la couleur est sur 6 caractères :

## SENSE HAT - Matrice



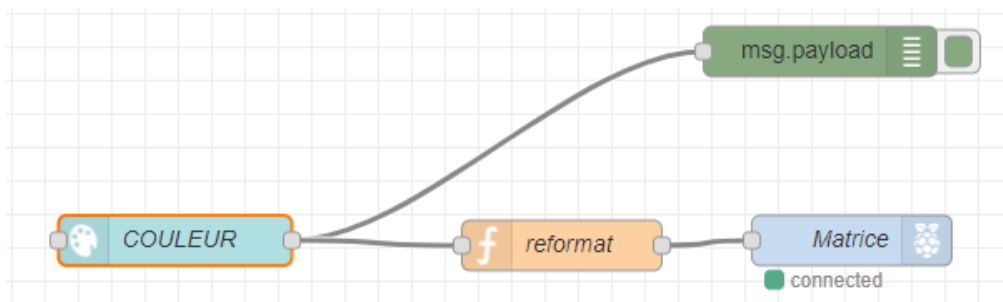
Et donc d'après les infos du nœud 'Sense Hat', on voit bien qu'il va falloir ajouter le caractère '#' devant la couleur pour qu'elle soit interprétée au format 'HEX'.

Exemple de transformation pour remplir la matrice d'une seule et même couleur :

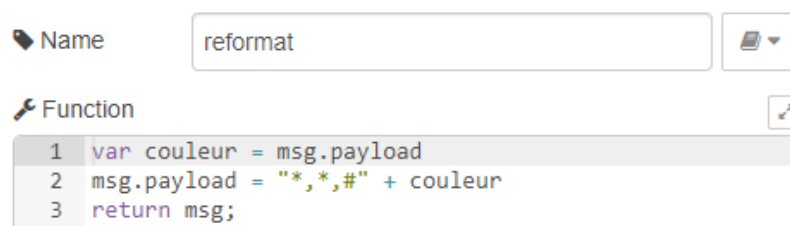
**349ecb → \*,\*,#349ecb**



C'est un nœud 'function' qui va nous permettre de faire cette transformation.  
Voici donc le diagramme complet :



Avec pour code du nœud 'function' nommé ici 'reformat' :



## EXERCICE 8 : 'Flow\_08.json'

### Objectif :

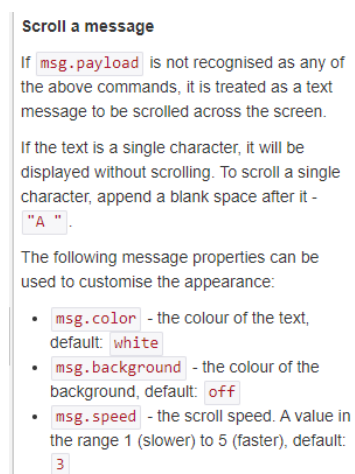
On va terminer par une application qui permet de faire défiler un texte saisi côté 'dashboard' sur la matrice de la 'Sense Hat'.

Vous allez essayer de faire cet exercice par vous-même sans aucune indication.

Commencez par une application très simple où l'on saisie juste un texte sur la page web et il défile ensuite sur la matrice.

**Faites contrôler le bon fonctionnement de l'ensemble par un enseignant.**

Si vous avez un peu de temps, vous pouvez essayer de modifier la couleur du texte ou la vitesse de défilement. N'oubliez pas d'utiliser les infos pour savoir comment faire :



## Remarques finales :

Ce que l'on vient de faire durant cette séance ne représente qu'une initiation à 'Node-Red'. C'est un outil extrêmement puissant et particulièrement adapté au monde de l'IoT.

Les protocoles phares de l'IoT sont implémentés, on peut accéder à des bases de données, envoyer des mails ou des SMS, créer et gérer des fichiers d'échange de données aux formats du web,...

Ses possibilités sont très larges et ce TP ne lui rend pas vraiment hommage, mais il vous a permis de découvrir le concept et quelques principes d'utilisation.

Si vous deviez un jour aller plus loin (en stage par exemple), sachez que de nombreuses ressources sont disponibles en ligne, que la communauté est assez réactive, que de nouvelles librairies naissent régulièrement,

...

Pour information, voici déjà simplement la description en français des nœuds disponibles dans 'Node-red' dans sa version de base :

<https://wiki.mchobby.be/index.php?title=Rasp-Node-Red-Basic-Nodes>