Fonctionnement de Node-Red

Node-Red :

Node-Red est un langage de programmation graphique par assemblage de blocs fonctionnels, très utile pour le développement de l’IoT.

Node-Red est accessible en exécution depuis Node.js qui nous permet d’utiliser le langage Javascript d’une autre manière, c’est-à-dire côté serveur et non côté client.

Donc Node-Red fonctionne selon des blocs interactifs. On en trouve de toutes les natures, des blocs simples, comme des blocs de debugs, de delay, de template, de fonctions, des entrées de ports séries et d’autres. On a également des nœuds plus complexe comme des entrées et sorties twitter, arduino, Facebook ou encore Rasberry Pie.

Pour notre projet, nous utilisons 7 nœuds différents. Une entrée ttn, un message de debug, un delay, une fonction, un template, un bloc influx dB et une sortie twitter.

Idée :

A l’intérieur de notre nacelle se trouve une antenne LoRa, celle-ci envoie par le protocole OTAA toutes les données récoltées par la carte Sparkfun Weathershield. Donc sur le serveur d’application, l’idée est de récupérer les données envoyées par la board, de les filtrer puis de les convertir dans le but de les poster sur Twitter.

Fonctionnement détaillé :

Donc, l’antenne de la nacelle envoie une chaîne de 32 caractères, contenant les valeurs du taux d’humidité, la température extérieure, la pression ambiante, l’intensité de la luminosité, la vitesse moyenne du vent et la vitesse max du vent.

Ces données sont envoyées par LoRa avec le protocole d’activation OTAA. C’est à partir de là qu’intervient Node-Red. Le premier bloc utilisé est une entrée ttn. Puis, suivent 2 branches différentes, la première est celle reliée au bloc Influx dB, connecté à notre base de données où l’on stocke donc toutes les données envoyées par la carte. L’autre branche est en lien avec un delay qui filtre, pour ne laisser passer que 4 messages sur une période de 24 heures. A cela vient s’ajouter la fonction de filtrage. Celle-ci est codée en JavaScript, elle extrait les données situées dans le buffer de la payload et retourne chaque mesure sous un entier contenu dans une variable. La sortie de la fonction est redirigée vers l’entrée d’un bloc template qui récupère ces variables et les intègre dans des chaînes de caractères, une pour chaque variable. Et enfin, ce sont donc ces chaînes qui sont envoyées dans le nœud Twitter pour être postées.

Fonction d’extraction :

return{a:String.fromCharCode( (msg.payload[1]) , (msg.payload[2]), (msg.payload[3]) , (msg.payload[4]), (msg.payload[5])), b:String.fromCharCode((msg.payload[6]) , (msg.payload[7]) , (msg.payload[8]), (msg.payload[9]), (msg.payload[10])), c:String.fromCharCode((msg.payload[11]) , (msg.payload[12]) , (msg.payload[13]), (msg.payload[14]), (msg.payload[15]), (msg.payload[16]), (msg.payload[17])), d:String.fromCharCode((msg.payload[18]), (msg.payload[19]), (msg.payload[20]), (msg.payload[21])), e:String.fromCharCode((msg.payload[22]) , (msg.payload[23]) , (msg.payload[24]), (msg.payload[25])), f:String.fromCharCode((msg.payload[26]) , (msg.payload[27]) , (msg.payload[28]), (msg.payload[29]))};

A écrire en une seule ligne. On retourne dans une variable les caractères sortis de la payload par index. On utilise cette méthode car la payload est contenue dans un buffer. Les caractères sont au préalablement convertit en chaîne de caractère par la fonction String.fromCharCode() car les caractères du buffer sont en code ascii et ne peuvent pas être retournés tels quels.

Template de formatage :

The humidity is: {{a}}

The temp is: {{b}}

The ambient pressure is: {{c}}

The luminosity is: {{d}}

The wind speed is: {{e}}

The max wind speed is: {{f}}

Les variables retournées par la fonction d’extraction sont placées ici de manière à compléter une chaîne de caractère pour être poster sur Twitter.

Paramètres de connexion de chaque nœud :

Nœud ttn uplink : Ajouter une gateway, device id pour filtrer un device

Nœud de InfluxdB : Ajouter une base de données créée avec InfluxdB avec son nom, un nom dans le measurement field

Nœud Twitter (La difficulté est ici) : Nécessité d’ajouter des API keys et des access token. Ces clés ne sont disponibles que par un compte développeur Twitter, dont il faut la démarche de demande, il faut suivre le lien donné par le nœud lui-même. Une fois le compte créé, il faut créer une application (le lien requis comme source des tweets n’est pas le lien de Node-Red). Une fois toutes les démarches faites, le nœud marche très bien.

Difficultés rencontrées :

J’ai rencontré 2 grosses difficultés lors de la mise en place de ce système de récupération et réutilisation de données.

Première et grosse difficulté : la connexion du compte Twitter au nœud de sortie.

L’autre a été tout simplement de Twitter. J’ai rencontré le plus souvent le message d’erreur « msg.payload.slice is not a function ». Il faut comprendre ici qu’il faut une chaîne de caractère en entrée du nœud Twitter. J’ai donc trouvé comme solution la fonction d’extraction ci-dessus dans le cas où la payload reçue se situe dans un buffer. En revanche, si le message complet reçu comporte différents champs, comme un payload, un metadata et d’autre, un simple « return{a :msg.payload} ; » fait l’affaire. Si votre payload comporte des sous champs parce que vous transmettez plusieurs variables il faut procéder de la même manière « return{a :msg.payload.*champ1*, b :msg.payload.*champ2* …} ; ».