Fonctionnement de Node-Red

Idée :

A l’intérieur de notre nacelle se trouve une antenne LoRa, celle-ci envoie par le protocole OTAA toutes les données récoltées par le sparkfun. 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 tweeter.

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 la 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 6 nœuds différents. Une entrée ttn, une fonction, un template, un delay, un bloc influxdb et une sortie twitter

Fonctionnement détaillé :

Donc l’antenne de la nacelle envoie une chaîne de 32 caractères, contenant l’humidité ambiante, la température extérieur, la pression atmosphérique, la lumière, 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. 2 paramètres sont nécessaires pour ajouter un device sur l’entrée, il faut ajouter une GateWay et entrer éventuellement l’id du device pour ne recevoir que les émissions de celui-ci.

Puis, suivent 2 branches différentes, la première est celle reliée au bloc Influxdb, 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 \_\_ messages toutes les \_\_ heures. A cela vient s’ajouter la fonction de filtrage. Celle-ci est codée en JS, elle extrait les données situées dans le buffer de la payload et retourne chaque mesure sous un entier contenu dans une variable. La fonction est renvoyée vers un template qui récupère ces variables et les intègre dans une chaîne de caractère et c’est donc cette chaîne qui est envoyée dans le nœud twitter et qui est postée sur Twitter.

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.