

Backend Sigfox

Actoboard étant l'interface utilisateur de snootlab, nous avons jugé qu'il était plus judicieux de migrer vers le backend Sigfox pour pouvoir visualiser les données du capteur directement dessus. Pour migrer : se rendre sur <https://backend.sigfox.com/activate> pour pouvoir activer le kit fourni par Snootlab. Choisir le nom du fabricant puis le pays et rentrer le numéro PAC et l'ID du modem Sigfox qui ont été fournis dans le mail que vous nous avez transféré. On a donc pu se créer un compte sur le backend de Sigfox.

Ensuite, il a fallu attendre une bonne heure pour que la migration du modem se fasse.

Voici l'interface du backend Sigfox : avec l'abonnement fourni, nous pouvons envoyer 140 messages (c'est le maximum) par jour, mais amplement suffisant pour contrôler une température sur une journée.

The screenshot shows the Sigfox backend interface in a Mozilla Firefox browser. The page title is "Device - List". The URL is <https://backend.sigfox.com/device/list>. The page features a search bar and several filters: "Id" (text input), "State" (dropdown menu set to "All"), "Average SNR (all)" (slider from 5 dB to 50 dB), and "Last seen from date" (text input). Below the filters, there are icons for file operations and buttons for "RESET" and "FILTER". The table below shows one device:

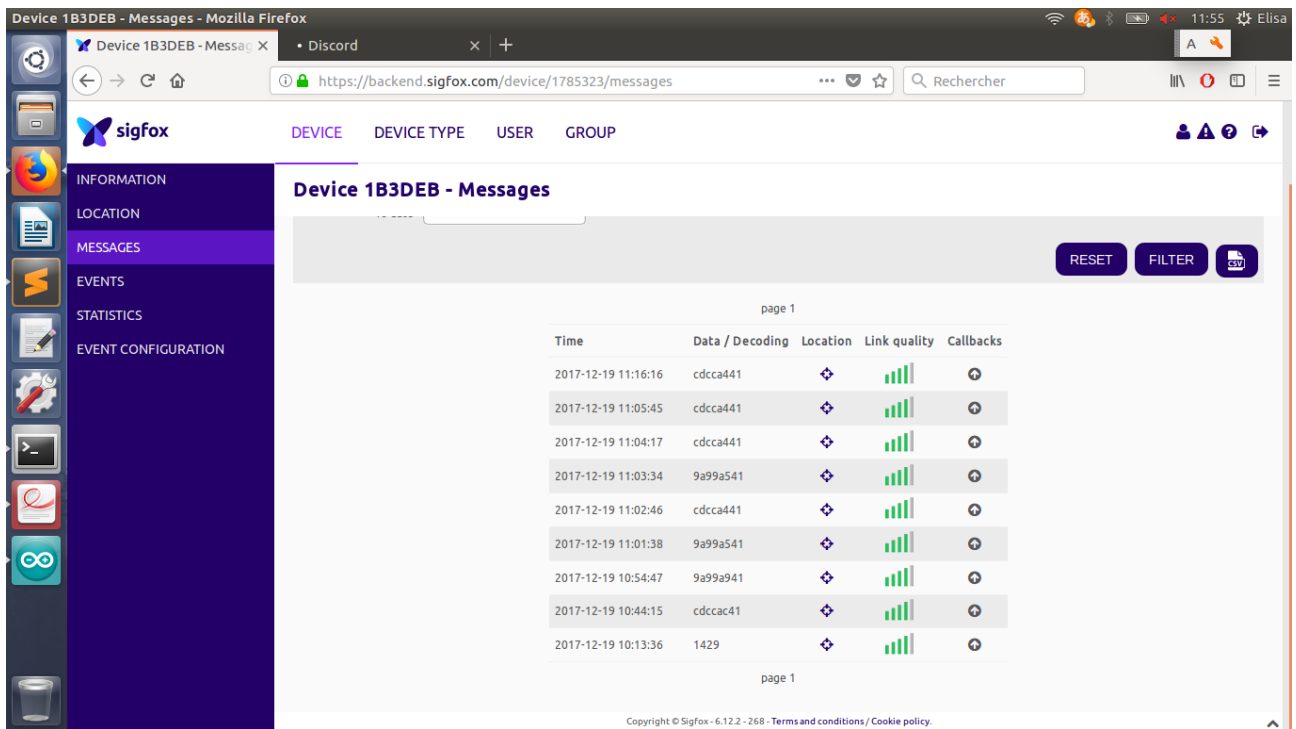
Average Rssi	Average SNR	Communication status	Device type	Id	Last seen	Name	Token state
40.00	70.62		Snootlab Alcatel-Lucent kit	1B3DEB	2017-12-19 10:13:36	Device 1B3DEB	

Ensuite, il a fallu envoyer notre premier message de l'Arduino vers le backend. Une librairie Akeru (fonctionne pour Akeru et Akene) est disponible pour Arduino : <https://github.com/Snootlab/Akeru>

Le code arduino est mis à disposition sur ce git : <https://github.com/elisascheer/Arduino-Sigfox> (je rajoute le lien dans la git Passerelle-Sigfox aussi)

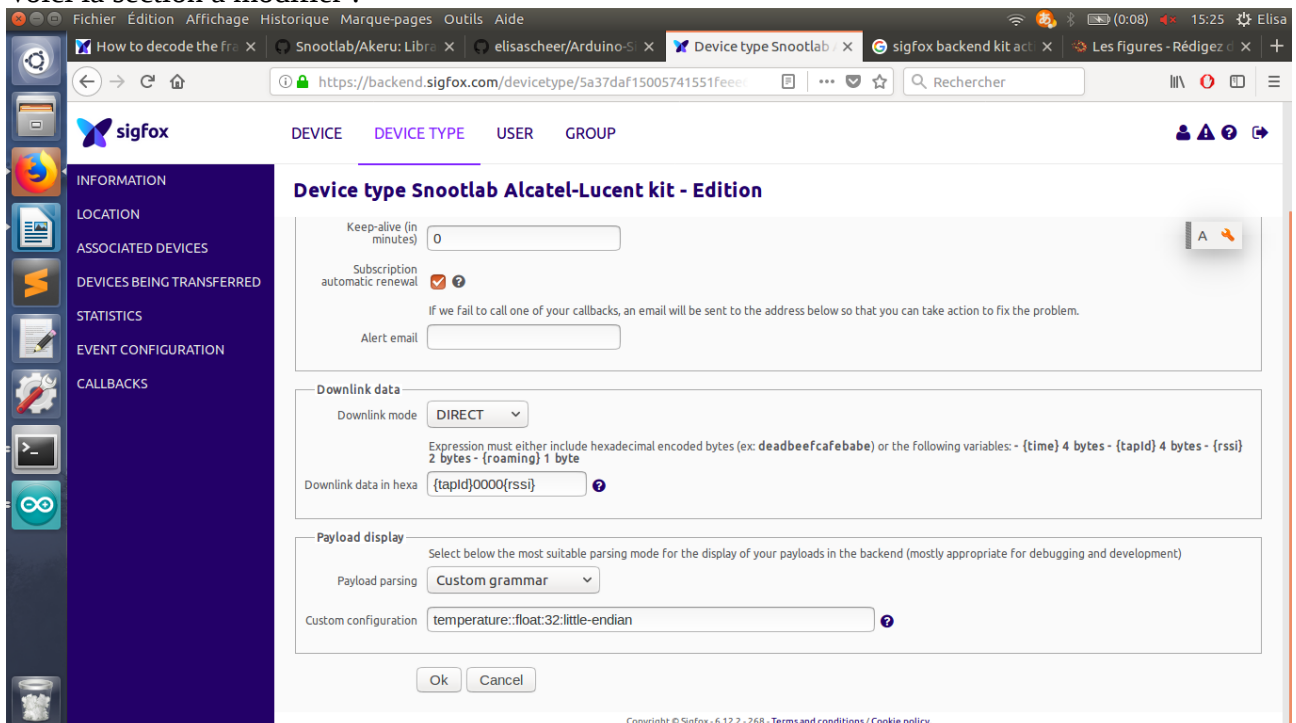
Les messages sont envoyés au réseau Sigfox dans le format hexadécimal, et la charge utile doit être de type String pour toutes les données envoyées. La fonction `akeru.toHex()` assure cette conversion. (On a essayé d'envoyer directement la valeur flottante de la température, ça ne marchait pas)

En revanche, dans le backend, nous visualisons les trames en hexadécimale et donc nous voulions voir les données en claires.















Pour pouvoir afficher la valeur de la température, il a fallu aller modifier dans le backend la section Payload display et choisir l'option Custom Grammar pour le Payload Parsing et écrire dans Custom configuration : temperature::float:32:little-endian

Voici la section à modifier :



Puis ensuite, nous avons pu observer les données en claires

Time	Data / Decoding	Location	Link quality	Callbacks
2017-12-19 15:03:30	3333ab41 temperature: 21.4			
2017-12-19 15:03:21	9a99a941 temperature: 21.2			
2017-12-19 14:21:08	cdcca841 temperature: 21.1			
2017-12-19 14:20:32	0000a841 temperature: 21.0			
2017-12-19 14:19:51	cdcca841 temperature: 21.1	