Sigfox : un renard sur l'internet des objets (3)

Vos premiers pas de goupil sur l'internet des objets

Frank Schleking & **Bernd vom Berg** (Allemagne)

Notre carte MKR FOX 1200 a été enregistrée sur le réseau Sigfox.
Rien ne s'oppose à une première tentative de communication.
Notre appareil Sigfox est censé envoyer sur le nuage le célèbre message d'accueil „Hello World !".

Comment notre station se présente-t-elle dans le nuage *Sigfox* ? Connectez-vous à l'infrastructure dorsale (*backend*) de Sigfox [1] comme d'habitude, avec votre adresse e-mail et votre mot de passe. Vous serez sur la page d'accueil du portail Sigfox (**fig. 1**). Cliquez maintenant sur l'onglet *Device*. Vous verrez un aperçu des appareils Sigfox enregistrés ou activés par vous (**fig. 2**). Il ne devrait y avoir que la station MKR FOX 1200) :

- **Group** : Le nom du groupe a été créé automatiquement par Sigfox et ne peut pas être modifié par vous. Ce groupe contient maintenant tous vos appareils Sigfox. Cliquez sur le nom du groupe pour obtenir des informations (non modifiables).
- **Device type** = type d'appareil : dans le groupe, vous pouvez trier et classer les stations identiques (type, structure, fonctions) selon le type d'appareil, p. ex. les stations

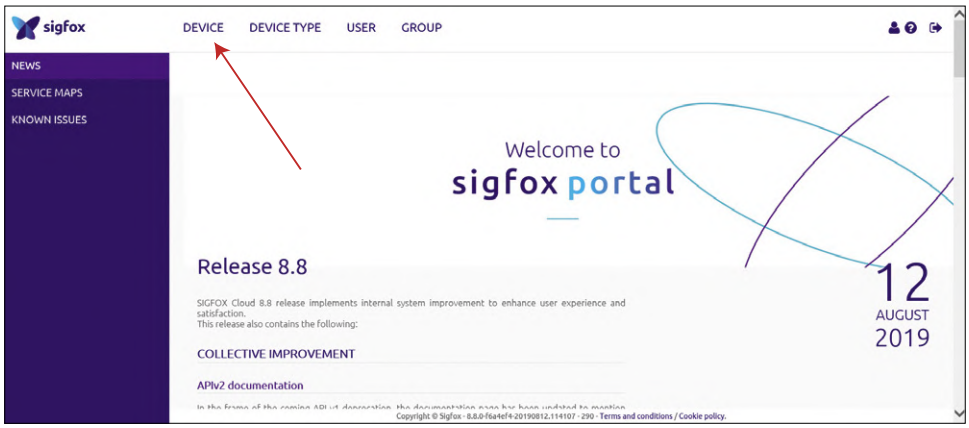

Figure 1 : Le portail Sigfox.

de mesure de température, les stations de lecture des valeurs analogiques des capteurs, les stations d'enregistrement de signaux binaires. Dans notre cas, nous n'avons bien sûr qu'un seul type de station, un seul *type de Device, à* savoir *Arduino_DevKit_1*. Plus tard, vous pourrez changer ce nom selon vos propres besoins.

- **Id** (= numéro d'identification) : le numéro d'identification unique de la station.
- **Last seen** = dernière vue : l'infrastructure saisit ici la date et l'heure de la dernière communication de la station, c'est-à-dire la date à laquelle le dernier télégramme a été reçu par la station. L'indication N/A signifie que l'infrastructure n'a encore reçu aucun télégramme de la station.
- **Nom :** Le nom de la station.
- **Token state** = statut du jeton : un jeton est un droit de transmission pour une station sur le réseau Sigfox. Quand on signe un contrat avec Sigfox (et payé les frais correspondants pour l'utilisation du réseau), on reçoit un certain nombre de jetons. Pour qu'une station du réseau Sigfox puisse devenir active, un jeton lui est attribué. Elle peut dès lors envoyer et recevoir des données sur le réseau. Quand vous achetez une carte MKR FOX 1200 et que vous l'enregistrez dans l'infrastructure Sigfox, vous recevez pour un an un jeton gratuit. Votre carte MKR FOX 1200 peut fonctionner comme station sur le réseau pendant un an. Sigfox attribuera son propre jeton unique à chaque nouvelle carte MKR FOX 1200. L'*état du jeton* indique si la station a déjà utilisé son jeton, c'est-à-dire si la station a déjà émisnvoyé des données sur le réseau Sigfox. Comme ce n'est pas encore le cas pour nous, nous trouvons ici un point d'interrogation ; la station n'a pas encore revendiqué ses droits de diffusion. Dès que la station aura envoyé son premier télégramme et donc utilisé son droit de transmission, une coche apparaît ici. Ensuite seulement le jeton est finalement attribué à cette station.

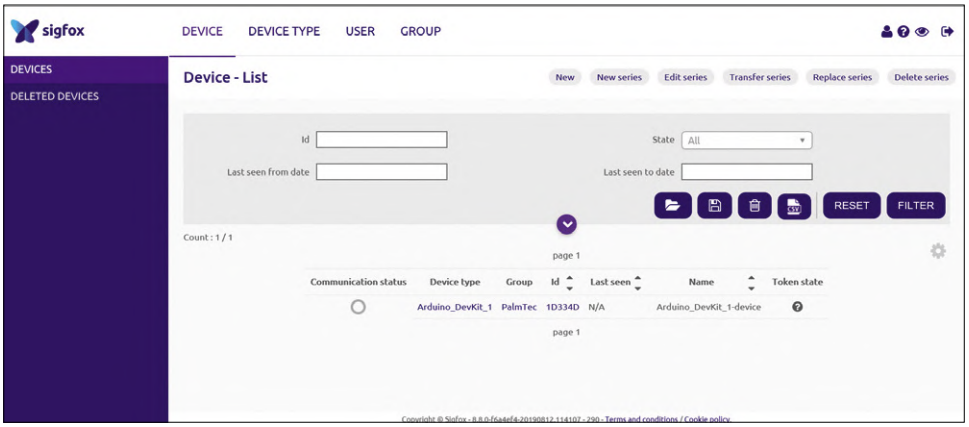

Figure 2 : Les appareils Sigfox enregistrés.

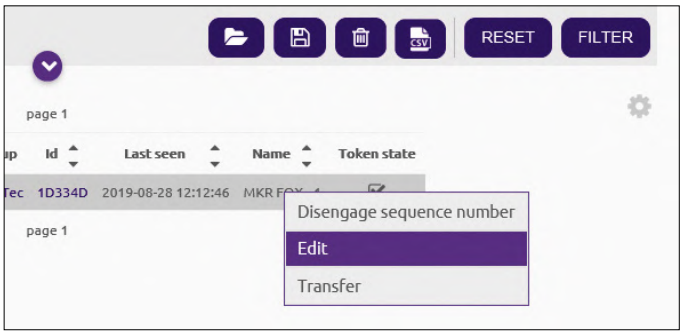

Figure 3 : Le menu déroulant pour *le champ du nom*.

Cliquez maintenant sur le *nom*. Un menu déroulant (**fig. 3**) vous propose l'option *Edit.* Dans la *fenêtre d'édition* suivante (**fig. 4**), certains paramètres de l'*appareil* sont définies ou modifiées. Changez ici le nom de la station Sigfox (si nécessaire), par exemple „MKR FOX - 1". Les autres champs restent inchangés. Cliquez sur *Ok*.

La fenêtre *Device Information* (**fig 5**) apparaît maintenant, qui résume toutes les informations sur le module Sigfox. Regardez, ne touchez ni changez rien. Si vous cliquez sur l'onglet *Device* en haut à gauche, vous revenez à l'aperçu des appareils, où vous verrez le changement de nom effectué. Fermez l'infrastructure de Sigfox.

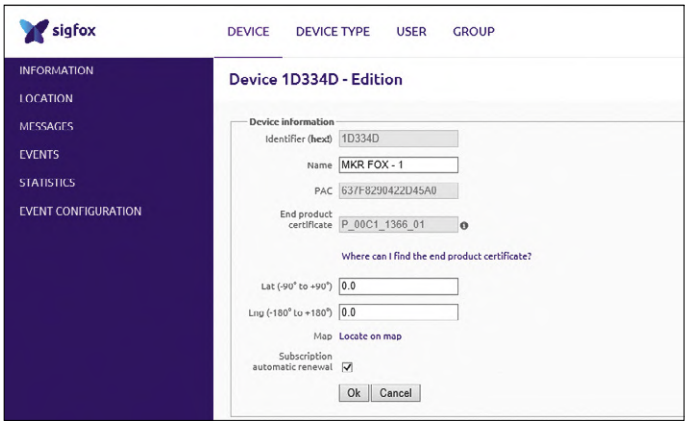

Figure 4 : La fenêtre de l'éditeur de l'appareil.

Bonjour tout le monde !

Pour que l'activation soit effective et pour que la station puisse travailler sur le réseau Sigfox, il reste à envoyer un message valide qui marquera le début de votre abonnement Sigfox gratuit d'un an à raison de 140 messages émis et quatre messages reçus par jour.

Le premier message à envoyer à l'infrastructure Sigfox est le texte réjouissant „ Hello World ! „, soit exactement douze caractères ASCII (= octets) qui tient donc parfaitement dans une charge utile de Sigfox. Les fonctions Sigfox de radiodiffusion ont été décrites dans la première partie de cet article, de sorte que la routine de radiodiffusion du **listage 1** et ses fonctions sont faciles à comprendre :

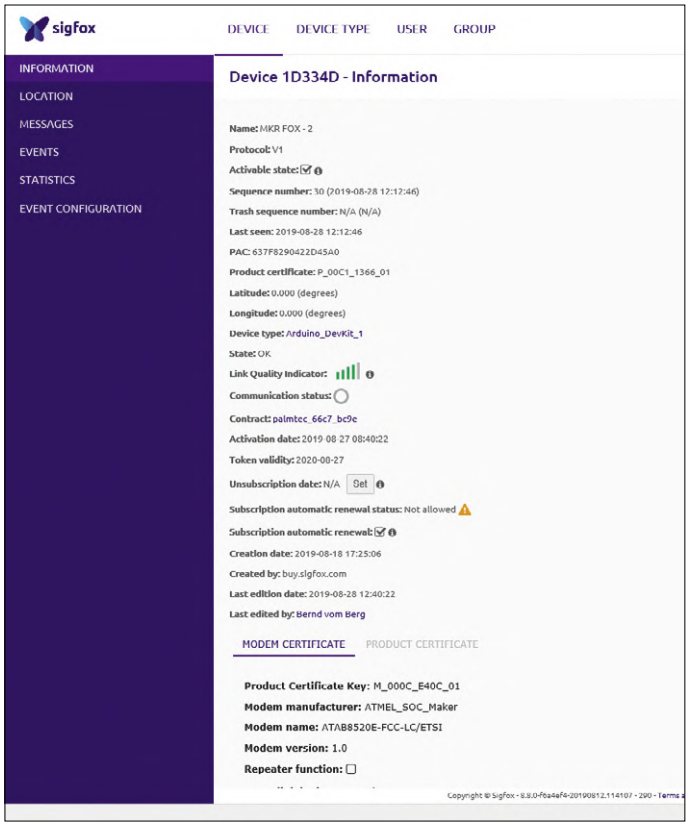

Figure 5 : La fenêtre d'information de l'appareil.

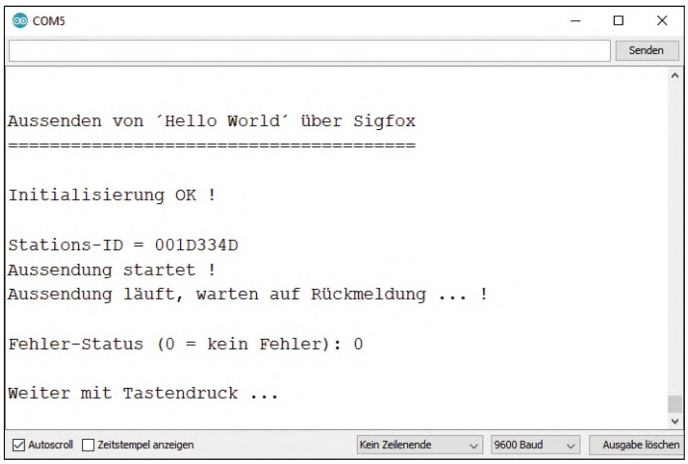

Figure 6 : La transmission de la chaîne de test.

Listage 1. Émission du message „Hello World !".

```
// enable Sigfox modem and check for errors
if (!SigFox.begin())        // error occurred?
{
  Serial.println("Error in Sigfox module!");

  while (1);  // after error, drop into infinite loop
}

// enable Sigfox modem debug mode
SigFox.debug();

// now we will send the string "Hello World!"

// prepare to transmit a packet
SigFox.beginPacket();

// assemble string (sequence of characters) into a Sigfox message
SigFox.print("Hello World!");

// final step in packet assembly and transmission; check for errors
// return code in variable ret
int ret = SigFox.endPacket();

Serial.print("\nError status (0 = no error): ");
Serial.println(ret);

// close Sigfox library and shut module down
SigFox.end();
```

- `SigFox.begin` : initialise la bibliothèque Sigfox et le modem Sigfox et signale une erreur le cas échéant.
- `SigFox.debug` : active le mode de débogage. Les fonctions d'économie d'énergie du microcontrôleur et du modem sont désactivées, la LED jaune de la carte MKR FOX 1200 clignote pour signaler l'envoi et la réception de données Sigfox.
- `SigFox.beginPacket` : lance la transmission d'un paquet (= charge utile) Sigfox.
- `SigFox.print` : place dans la charge utile la valeur ou la chaîne de caractères passée puis l'envoie. Les valeurs numériques sont envoyées sous forme de chaînes de caractères ASCII. Pour les données binaires (nombres „réels", pas de représentation ASCII), on utilise la fonction `SigFox.write` (nous y reviendrons).
- `int ret = SigFox.endPacket` : termine la transmission du paquet Sigfox (charge utile Sigfox) et retourne 0 (= aucune erreur) ou 1 (= erreur).
- `SigFox.end` : ferme la bibliothèque Sigfox et le modem Sigfox

Où l'on voit l'étonnante simplification de la communication de données complexes grâce au recours à une bibliothèque puissante !
Chargez le sketch Sigfox sur la carte MKR FOX 1200, ouvrez le terminal et lancez le sketch. La fenêtre du moniteur affiche le menu principal du sketch. Sélectionnez maintenant le point de menu 2 *Transmettre „Hello World* ! La procédure d'envoi de la chaîne de test est exécutée, certaines informations de statut étant éditées

dans la fenêtre du terminal. Le processus de transmission est signalé par le clignotement de la LED jaune sur la carte MKR FOX 1200. La transmission complète devrait se dérouler sans erreur (**fig. 6**).

De retour dans l'arrière-boutique

Si vous avez une couverture radio Sigfox, l'infrastructure Sigfox reçoit vos données dans un délai très court. Vérifiez en démarrant l'infrastructure (*backend*), en vous connectant et en cliquant sur l'onglet *Device* sur l'écran de démarrage. Dans la liste des appareils Sigfox actifs, cliquez (exactement) sur le champ *Id* de la station pour faire apparaître la page d'information de l'appareil et là, à gauche, sur *Messages*.
La fenêtre suivante (**fig 7**) énumère tous les télégrammes reçus par l'infrastructure Sigfox. Le champ *Time* est explicite ; le champ *LQI* (Link Quality Indicator) indique l'intensité du champ avec lequel le télégramme Sigfox a été reçu par les stations de base. Si vous placez le curseur sur ce champ, vous verrez une description de la qualité de réception et l'opérateur local du réseau Sigfox dans lequel se trouve la station Sigfox sera affiché. Cet opérateur ne s'appelle pas nécessairement Sigfox, il peut aussi être un partenaire (p. ex. à l'étranger). Lorsque vous cliquez sur *Location,* une carte montre où approximativement se trouve la station Sigfox.

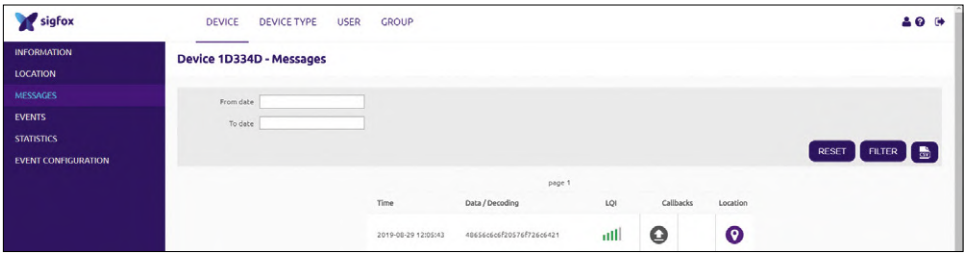

Figure 7 : La fenêtre de message de l'appareil.

Figure 8 : Le champ d'information pour le type d'appareil.

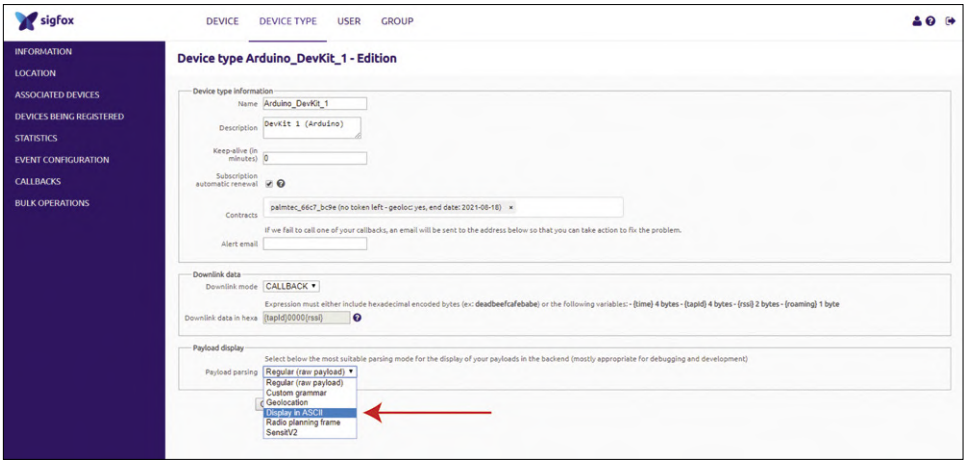

Figure 9 : La fenêtre d'édition pour le type d'appareil.

Venons-en maintenant aux informations importantes dans le champ *Data/Decoding* (Données/Décodage), où le contenu de la charge utile est affiché, une fois sous forme de données brutes et une fois sous forme décodée. L'infrastructure ignore encore comment interpréter les données brutes, il faut le lui apprendre. Examinons le champ de donnée :

```
48 65 6c 6c 6f 20 57 6f 72 6c 64 21
H  e  l  l  o     W  o  r  l  d  !
```

La chaîne est composée des codes ASCII des caractères du message de test „Hello World ! (20 est le code ASCII pour l'espace). Le mode de décodage peut être réglé à partir de la fenêtre *Device List* (fig. 2). Cliquez sur le nom *Arduino_DevKit_1* sous *Device type*, vous accédez alors au champ d'information de ce type de périphérique (**fig 8).** Après avoir cliqué sur *Edit*, vous accédez à la fenêtre d'édition correspondante (**fig. 9**). Dans la zone *Payload display* (= affichage des charges utiles) modifiez (seulement) *Payload parsing* (= analyse des charges utiles) en choisissant *Display in ASCII* (= afficher en ASCII) et confirmez cette modification. De retour dans la fenêtre *Messages,* vous verrez pour cette station une ligne supplémentaire avec le contenu de la charge utile affiché en caractères ASCII (**fig. 10**).
Deux étapes fondamentales sont nécessaires pour transférer des valeurs numériques dans la charge utile. Vous devez d'abord déterminer le type des données à transférer dans la charge utile de 12 octets, car à chaque type de données correspond un nombre spécifique d'octets ou. Les plus courants sont réunis dans le **tableau 1.** Dans la charge utile, vous pouvez organiser les types de données à transférer comme dans le **tableau 2.** Sur les douze octets disponibles dans la charge utile, dix octets seraient utilisés.

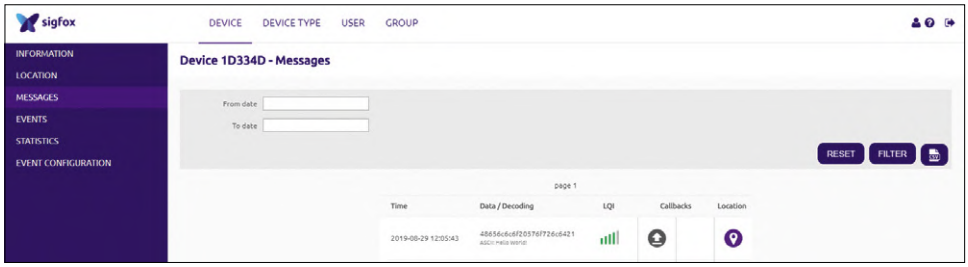

Figure 10 : Décodage du contenu de la charge utile en caractères ASCII.

Tableau 1. Nombre d'octets de différents types de données.

données	octets
char / unsigned char	1
int8_t / uint8_t	1
int16_t / uint16_t	2
int / unsigned int	4
int64_t / uint64_t	8
float	4

Listage 2. Structure du type de données Sigfox requis Sigi_Dat_1.

```
// structure and data type 1 for our example:
// transmission of fixed numeric values
 typedef struct __attribute__ ((packed)) sigfox_message {
  unsigned char  value_1; // first value in payload: 1 byte long
  unsigned int   value_2; // second value in payload: 4 bytes long
  float          value_3; // third value in payload: 4 bytes long
  unsigned char  value_4; // fourth value in payload: 1 byte long
} Sigi_Dat_1;
```

Tableau 2. Organisation possible des types de données dans la charge utile.

Valeur dans la charge	données	octets
1	unsigned char	1
2	unsigned int	4
3	float	4
4	unsigned char	1

Listage 3. Émission de valeurs numériques fixes.

```
// definition of four fixed numeric values
 unsigned char   AIN_1 = 0x1f ;          // first value, 1 byte long
 unsigned int    pressure = 0x12345678; // second value, 4 bytes long
 float           temp = 1233.56;        // third value, 4 bytes long
 unsigned char   AIN_2 = 0x55;          // fourth value, 1 byte long

 // enable Sigfox modem and check for errors
 if (!SigFox.begin())        // error occurred?
 {
   Serial.println("Error initializing Sigfox module. RESET to continue");
   while (1);  // after error, drop into infinite loop
 }
 else
 {
     Serial.println("Sigfox modem OK\n");
 }

 // enable debug LED and disable power-saving modes
 SigFox.debug();

 // deal with all pending interrupts
 SigFox.status();
 delay(1);

 // now we write the current values that are to be transmitted into
 // the SF_send structure variable
 // this process assembles the payload contents
 SF_send.value_1 = AIN_1;        // unsigned char value:  1 byte
 SF_send.value_2 = pressure;     // unsigned int value:   4 bytes
 SF_send.value_3 = temp;         // float value:          4 bytes
 SF_send.value_4 = AIN_2;        // unsigned char value:  1 byte
```

Suite à la page suivante

Ensuite, un type de données spécial est défini à partir de ces spécifications, dans lequel les données souhaitées pour la charge utile peuvent être stockées ensemble très facilement. Dans notre exemple, cela ressemble au **listage 2.** Pour bien comprendre cette construction en détail, une connaissance approfondie de C/C++ est nécessaire, mais une explication simplifiée est possible.

Avec l'instruction `struct ..... sigfox_message` et les spécifications suivantes entre accolades, nous avons créé une structure appelée `sigfox_message`. Ce n'est rien d'autre qu'un ensemble de données connexes sous un nom (principal) commun. C'est comme pour un tableau ou une matrice, mais ceux-ci sont toujours constitués de données du même type. Une structure rassemble des données de différents types, ici à partir de deux caractères non signés `unsigned char`, un entier non signé `unsigned int` et une valeur flottante `float`. Les types de données des éléments de la structure doivent correspondre exactement à la définition de la charge utile et doivent également se présenter dans le bon ordre.

Il est facile d'accéder maintenant à un élément spécifique de la structure : `Nom de la structure.Nom de element` Ainsi, par exemple, avec `sigfox_message.valeur_3 =25.78;` la valeur de l'élément flottant interne `valeur_3` se voit attribuer la valeur `25.78`.

Le pas suivant et nous créons à partir de cette structure un nouveau type de données propre. Bien sûr, vous connaissez les types de données de base en C comme les caractères `non signés`, `int` ou `float`. Grâce à l'instruction supplémentaire `typedef` au début de la définition de la structure, nous créons un autre type de données qui correspondra exactement à la structure qui suit. Ce nouveau type de données nécessite naturellement aussi un nouveau nom propre, lequel est spécifié tout à la fin de la définition, ici `Sigi_Dat_1`. Le paramètre `__attribut__ ((packed))` permet de supprimer de la structure les octets de remplissage superflus afin que

Suite à la page précédente :

```
// next we use the Sigfox library to transmit the data we have
// assembled in the structure variable SF_send, which will become the
// payload contents

// prepare to transmit a packet
SigFox.beginPacket();

// pass structure variable to the Sigfox library
SigFox.write((char*)&SF_send, sizeof(SF_send));

// error check: if endPacket() returns 1, report an error
int ret = SigFox.endPacket();
if (ret > 0)
{
  Serial.println("Error: transmission failed. RESET to continue");
  while(1);     // infinite loop
}
else
{
  Serial.println("Sigfox transmission OK");
}

// close Sigfox library and shut module down
SigFox.end();
```

suite de la page 23

◇ Salon des seniors
du 25 au 28 mars à Paris
Porte de Versailles
www.salondesseniors.com/

◇ L'INDUSTRIE DE DEMAIN S'INVENTE SUR GLOBAL INDUSTRIE
31 mars au 3 avril à Paris Nord - Villepinte
https://www.global-industrie.com/fr

avril 2020

◇ Congrès sur la cybercriminalité et la protection des données on-line
du 01 au 02 avril à Paris
http://akjassociates.com/event/france

◇ Maker Faire Lille
du 03 au 05 avril 2020 à Lille - Gare Saint Sauveur
https://lille.makerfaire.com/

Photo : Quentin Chevrier

◇ 12ème Journée Française des Tests Logiciels
le 7 avril au Beffroi de Montrouge
http://www.cftl.fr/JFTL/accueil/

◇ RICV - Rencontres internationales de cerfs-volants
du 6 au 14 avril à Berck-sur-Mer
www.cerf-volant-berck.com

Photo : RICV - mairie de Berck sur Mer

◇ ProDurable (économie durable)
Du 27 au 29 avril - Paris Palais des congrès
www.produrable.com

◇ CONCOURS LEPINE Salon international des inventions
du 30 avril au 11 mai à Paris
www.concours-lepine.com/

Tableau 3. Charge utile entrante.

1f	78563412	ec319a44	55
(A)	(B)	(C)	(D)

seuls les octets de données utilisés soient transférés à la charge utile puis émis.

Toute cette structure est maintenant placée tout au début du sketch comme variable globale, de sorte que ce type de données est accessible depuis n'importe quel point du sketch.
Pour utiliser ce type de données, il faudra bien sûr définir des variables pour ce type de données. Cela se passe comme d'habitude :
`Type_de_data Nom_de_variable;`
La ligne `Sigi_Dat_1 SF_send;` crée une variable nommée `SF_send` qui a exactement la structure définie précédemment (il est préférable de la définir aussi comme variable globale). Les valeurs individuelles de ces variables peuvent alors être accessibles avec `SF_send.valeur_1 = 12;` par exemple.

Listage 4. Nouveau type de données „Sigi_Dat_2" pour l'émission des valeurs mesurées.

```
// structure and data type 2 for our example:
// transmission of sensor readings
  typedef struct __attribute__ ((packed)) sigfox_message_2 {
    float      value_1;     // first value in payload:   4 bytes long
    float      value_2;     // second value in payload:  4 bytes long
    uint16_t   value_3;     // third value in payload:   2 bytes long

    // this structure therefore occupies a total of ten bytes

} Sigi_Dat_2;
```

Listage 5. Écriture des valeurs de mesure dans la variable de structure.

```
    // now we write the current values that are to be transmitted into
    // the SF_send_mw structure variable
    // this process assembles the payload contents
    SF_send_mw.value_1 =   temp;        // float value:        4 bytes
    SF_send_mw.value_2 =   pressure;    // float value:        4 bytes
    SF_send_mw.value_3 =   LDR;         // uint16_t value:     2 bytes
```

Émettre

Pour voir comment les valeurs numériques sont empaquetées dans la charge utile Sigfox, nous écrivons une fonction qui envoie des valeurs numériques fixes sur le réseau Sigfox (**listage 3**). La fonction est appelée par l'option 3 du menu dans le sketch de démonstration. Il n'y a plus qu'à ajouter que c'est la fonction `SigFox.write` qui est maintenant utilisée pour envoyer la charge utile, puisque nous envoyons maintenant des valeurs numériques au lieu de caractères. Si tout fonctionne, la charge utile apparaît dans l'infrastructure comme l'indique le **tableau 3**.

Pour finir, nous allons écrire une fonction pour la transmission de valeurs de mesure réelles (option 4 du menu dans l'exemple de sketch Sigfox), à savoir la température mesurée par le capteur BMP280 (float, 4 octets), la pression atmosphérique mesurée par le capteur BMP280 (float, 4 octets) et la valeur mesurée par la LDR (pour cela, le cavalier JP5 doit être placé en position 1-2 sur la carte mère) à l'entrée analogique A2 (uint16_t, 2 octets). Ainsi, dix des douze octets de la charge utile sont occupés.

Pour cet ensemble de données, nous définissons un autre type de données Sigfox (**listage 4**) et une nouvelle variable appelée `SF_send_mw` :
`// Variable pour le type de données 'Sigi_Dat_2' :`
`Sigi_Dat_2 SF_send_mw;`
Dans une nouvelle fonction du sketch, les trois valeurs mesurées sont enregistrées, écrites dans la variable de structure et envoyées (**listage 5**). Les sorties de contrôle via le terminal série et sur l'afficheur ePaper affichent les valeurs de mesure courantes ainsi que le processus de communication Sigfox. Des explications sont données dans les commentaires détaillés du programme.

À suivre

Nous voici à la fin de notre courte présentation du réseau 0G-IoT Sigfox, faite à l'aide de la carte Arduino Maker MKR FOX 1200. Avec les programmes et fonctions que nous avons créés, vous disposez d'une base solide pour faire vous lancer sur le réseau Sigfox.

Ce qui manque encore, c'est une visualisation des données du côté de l'utilisateur, c'est-à-dire sur son ordinateur ou son téléphone à l'aide d'un tableau de bord configurable. Ce sera le thème du prochain et dernier épisode de cette série. ⏮

(190281 – 02 VF)

Lien

[1] Identification dans l'infrastructure : https://backend.sigfox.com/auth/login