Montage

20

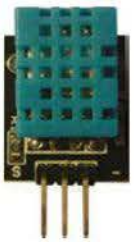
Temboo et la carte Yún – API Twitter

Nous en arrivons à un sujet passionnant – encore un –, qui est l'accès, par le biais d'une carte Yún, aux innombrables services web comme Twitter, Facebook, Google+ ou Dropbox, sans avoir à écrire des kilomètres de code. Les services que je viens d'énumérer sont très populaires et leurs noms sont aujourd'hui dans toutes les bouches. Nous allons nous y intéresser de plus près dans ce chapitre. Le projet Temboo (*https://temboo.com/*) s'est notamment donné pour vocation de permettre à nous autres, utilisateurs d'Arduino, d'accéder facilement à tous ces services Internet. Il a développé plus d'une centaine d'API *(Application Programming Interface)* qui exécutent des tâches ou offrent des services divers et variés. Je vous présenterai ici l'API Twitter.

Au sommaire :

- la structure sous-jacente à Temboo ;
- la création d'un compte Temboo ;
- la récupération d'informations, telles que des *tokens*, auprès de Twitter pour envoyer un tweet via Temboo ;
- l'exécution d'un premier test dans le navigateur ;
- le transfert du code du sketch depuis le navigateur sur la Yún pour envoyer un tweet depuis la carte ;
- la programmation d'un sketch pour mesurer l'humidité et la température au moyen d'un capteur DHT11, puis twitter les valeurs mesurées.

Composants nécessaires



1 capteur d'humidité-température DHT11

Plusieurs cavaliers flexibles de couleurs et de longueurs diverses

Temboo

Pour commencer, regardons ce qui se cache derrière le projet Temboo. Nous allons en décortiquer la structure afin de mieux comprendre son échange de données avec votre carte Yún. Je ne peux malheureusement pas entrer dans tous les détails, car l'affaire est beaucoup trop complexe. Mais je pense que mes explications vous permettront de vous initier à ce système. Si je n'ai pas répondu à toutes les questions que vous pourriez vous poser, je vous recommande de parcourir la rubrique *Get Started* du site Internet de Temboo qui propose notamment une FAQ et quantité d'informations complémentaires.

La structure

La figure 20-1 est une représentation schématisée de la communication entre une carte Yún et Temboo.

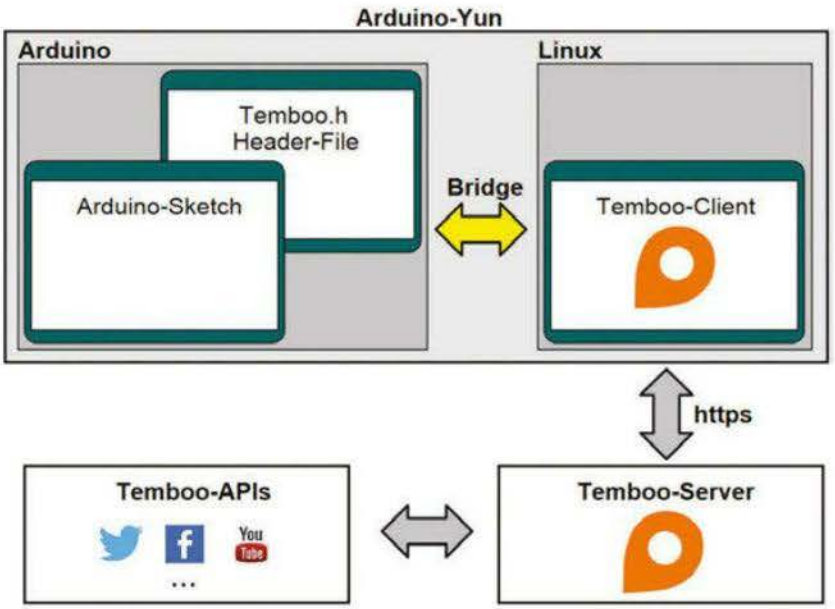
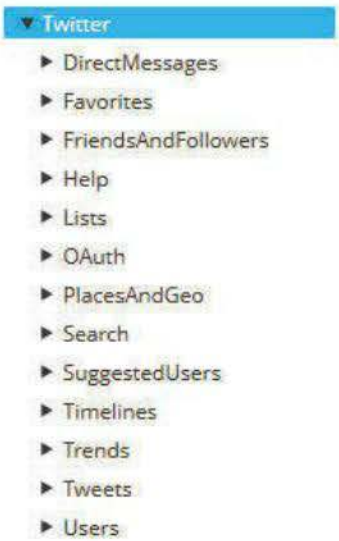


Figure 20-1
Communication entre la carte Yún et le serveur Temboo

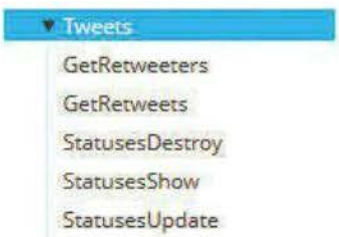
En haut à gauche sur le schéma, vous pouvez voir la carte Yún à laquelle est liée la bibliothèque requise à l'aide d'un fichier d'en-tête `Temboo.h` et dont le code proprement dit se trouve dans le fichier `temboo.cpp`. Le sketch Arduino se connecte via une passerelle (*bridge*) à Linino qui déclenche un processus initial pour exécuter le fichier Temboo. Ce fichier, qui se trouve dans le dossier */usr/bin*, contient un script Python qui charge le client Temboo. Ce client transmet une requête HTTPS au serveur Temboo qui exécute des prestations de services de type *cloud*. C'est par le biais de ce serveur que l'on peut accéder aux multiples API pour profiter de leurs extraordinaires fonctionnalités. Jetez un œil au site web de Temboo et vous verrez à quel point l'offre est étendue. Cliquez sur *LIBRARY* dans la partie supérieure gauche de la fenêtre. Une arborescence apparaît alors sur le côté gauche. Elle énumère toutes les API disponibles. J'ai cliqué sur le petit triangle afin de développer la bibliothèque Twitter qui nous intéresse ici.

Figure 20-2 ▶
API Twitter de Temboo



L'arborescence présente la liste des tâches *(tasks)* disponibles pour Twitter. Chacune d'elles contient une ou plusieurs *choreos*, abréviation de *choreographies* (chorégraphies), qui sont similaires aux méthodes utilisées dans la programmation orientée objet. Ici, nous voulons poster un message sur Twitter. Donc nous choisissons l'entrée *Tweets* dans l'arborescence d'API. Comme vous le savez certainement, dans le jargon de Twitter, on ne dit pas « diffuser un message », mais twitter. Cette API génère un tweet qui diffuse un message sur Twitter. Vous me suivez ?! Voyons maintenant les choreos proposées sous la rubrique *Tweets* :

Figure 20-3 ▶
Choereos des Tweets



L'entrée intitulée *StatusesUpdate* paraît prometteuse ; nous l'utiliserons plus tard pour twitter. Passons maintenant à la pratique, car la théorie est ennuyeuse à la longue. Voici les étapes requises pour pouvoir twitter avec la carte Yún.

Création d'un compte Temboo

Il va de soi qu'un compte est nécessaire pour pouvoir utiliser Temboo. Les fonctions de base sont gratuites. Vous pouvez exécuter 250 choreos par mois et transférer jusqu'à 1024 Mo de données. Pour plus de détails, consultez la FAQ. Cliquez sur le bouton *SIGN-UP* sur la page d'accueil de Temboo, puis saisissez votre adresse électronique dans la fenêtre qui apparaît.

Figure 20-4
Inscription à Temboo

Quelques instants plus tard, vous devriez recevoir un courrier de confirmation intitulé *Create Your Temboo-Account*. Si votre boîte de réception reste vide, vérifiez vos spams. Vous devez cliquer sur le lien qui figure dans le message afin de terminer la création de votre compte en saisissant un identifiant et un mot de passe.

Figure 20-5
Création du compte Temboo

Cliquez sur le bouton *GO* pour valider la création de votre compte. Vous pouvez commencer sans plus attendre, mais je vous recommande de prendre le temps de visionner les vidéos proposées sur le site. Ces tutoriels sont en effet un bon moyen de se familiariser avec Temboo, car ils fournissent de précieux conseils. Nous allons maintenant nous intéresser à Twitter.

Votre compte Twitter

Il vous paraîtra sans doute déplacé que je vous demande si vous avez bien un compte Twitter. En effet, il vous en faut un, sinon vous aurez du mal à twitter. Pour créer un compte, rendez-vous sur le site web *https://twitter.com/signup*. Maintenant que ce point est éclairci, nous pouvons procéder à quelques réglages sur le site de Twitter.

Pour pouvoir accéder à Twitter depuis l'extérieur, vous devez préalablement créer et enregistrer une application. Ouvrez la page web *https://dev.twitter.com/apps/new*. Nommez votre application sans utiliser la séquence de caractères Twitter.

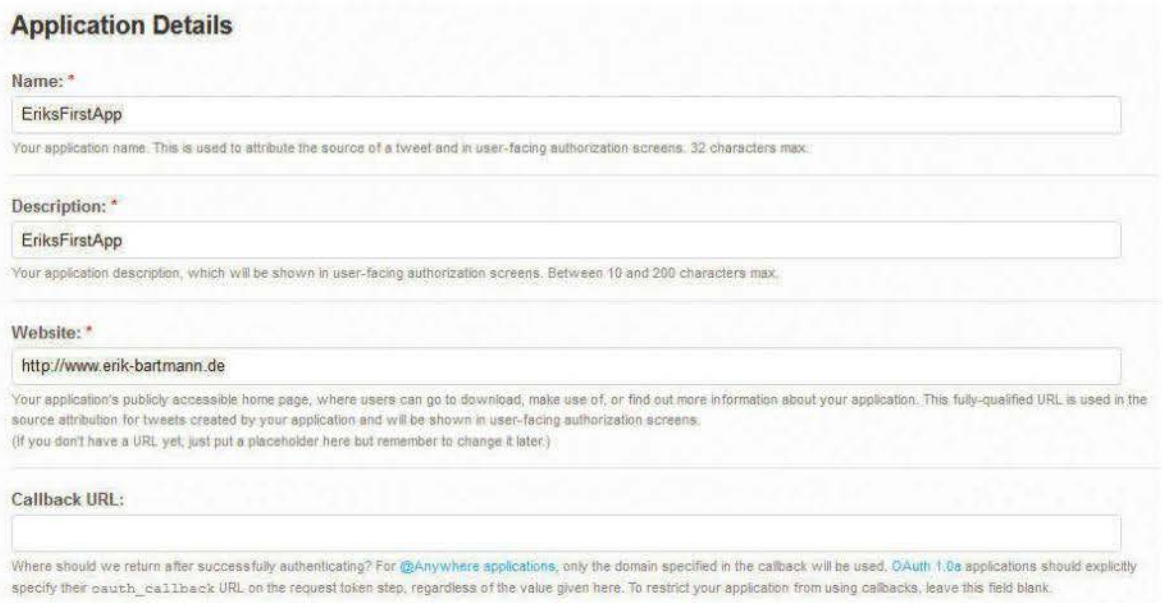


Figure 20-6 ▲
Détails de l'application Twitter

Lorsque vous aurez accepté les conditions générales d'utilisation, saisi une description, puis cliqué sur le bouton *Create Your Twitter Application*, vous verrez s'afficher une page sur laquelle vous pourrez gérer votre appli dans ses moindres détails.

◀ **Figure 20-7**
Gestion de l'application Twitter

Quand vous voudrez poster un message sur Twitter, vous devrez vous identifier par le biais d'*OAuth* qui s'assure que personne n'essaye de se faire passer pour vous. Vous trouverez plus d'informations à ce sujet à l'adresse *https://dev.twitter.com/docs/auth/oauth*.

Avant de commencer, j'aimerais vous rappeler quelques principes de précaution élémentaires : sur Internet, on peut croiser le chemin de personnes ou d'organismes qui n'accordent pas une grande importance au respect de la vie privée. La NSA *(National Security Agency)* représente un exemple inquiétant où les droits des internautes sont foulés aux pieds. La sécurité est une question cruciale aujourd'hui et elle mérite que l'on y accorde la plus haute attention. Le simple fait de diffuser nos données de connexion sur Internet nous expose au risque potentiel de les voir récupérées par des tiers qui s'en serviront pour envoyer des courriers malveillants. Mais les risques ne s'arrêtent pas là. On pourrait aussi évoquer l'espionnage industriel qui fait perdre des millions, voire des milliards, aux entreprises qui en sont victimes. Mais si vous renoncez à communiquer vos identifiants sur Internet, vous ne pouvez pas non plus utiliser les nombreux services web. Comment résoudre ce dilemme ?

C'est là qu'*O*Auth entre en scène : il s'agit d'un protocole libre. Une application contacte un prestataire en échangeant des informations sous la forme de clés (*keys*) et de jetons (*tokens*). Mais ces données ne correspondent pas aux données de connexion de l'utilisateur. Quand nous créons une application, il s'agit dans le jargon spécialisé d'un *client* ou *consumer* qui souhaite établir une connexion avec le service web du prestataire, par exemple. Au moment de son inscription, le client reçoit une authentification en deux parties : la clé (*key*) et le *secret* qui sont des séquences de caractères générées de façon aléatoire. Lorsque des informations doivent être échangées entre l'application – c'est-à-dire le client – et le prestataire, le client envoie la clé au prestataire qui lui transmet à son tour un token provisoire. Pour un accès définitif et pour tous les autres accès, le client doit générer un jeton d'accès à partir du jeton provisoire. Ce token protège l'accès au

prestataire qui effectue une vérification en utilisant un algorithme de hachage sécurisé.

Pour que vous n'ayez pas à manipuler ces codes assez complexes, les choreos Twitter de Temboo s'occupent du processus de signature d'OAuth à votre place. Vous devez simplement disposer de l'authentification OAuth.

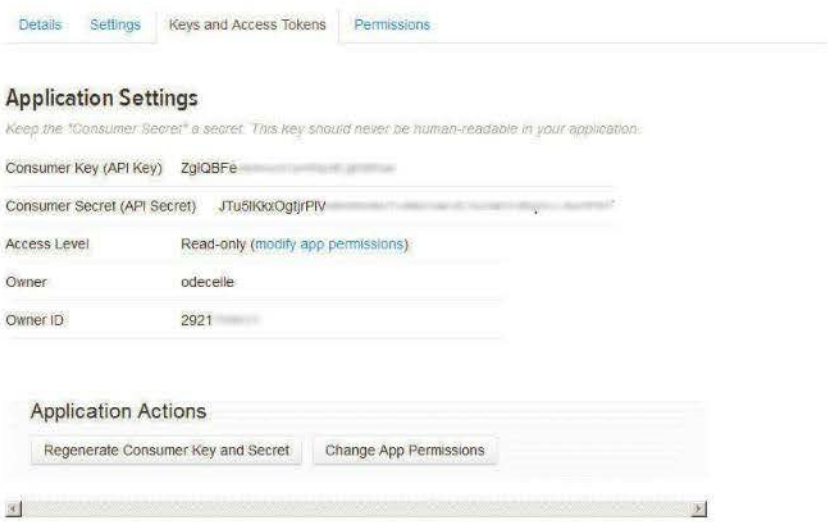
J'ai un peu de mal à suivre. Comment obtient-on toutes ces informations ?

Vous avez raison, Ardus. Ce n'est pas simple, mais ces procédures sont indispensables pour garantir une sécurité maximale. Je vais tout vous expliquer en détail et tout se passera bien. De nombreuses données sont nécessaires pour la création d'une authentification qui, par ailleurs, est gérée par Temboo. Les informations suivantes sont demandées :

- AccessToken
- AccessTokenSecret
- ConsumerKey
- ConsumerSecret

Comme les obtient-on ? Ces informations sont accessibles dans les données du compte Twitter. Jetez un œil à la page de gestion de l'application qui apparaît dès que vous cliquez sur le bouton *Create Your Twitter Application*. Cliquez sur le lien *Manage keys and access tokens* sous la rubrique *Application Settings* de l'onglet *Details* afin d'accéder à deux des clés requises.

Figure 20-8 ▶
Clés OAuth accessibles dans les paramètres de l'application



Vous pouvez noter la présence des clés *Consumer Key* et *Consumer Secret*. Il vous manque donc encore les jetons *AccessToken* et *AccessTokenSecret*. Lorsque vous faites défiler la page vers le bas, vous pouvez voir le bouton suivant :

Cela paraît très prometteur. Allez-y ! Cliquez dessus pour voir ce qu'il se produit. Un message s'affiche en haut de la fenêtre pour vous informer que les clés ont été créées. Mais où sont-elles ? Pour les voir, il suffit de faire défiler la fenêtre vers le bas :

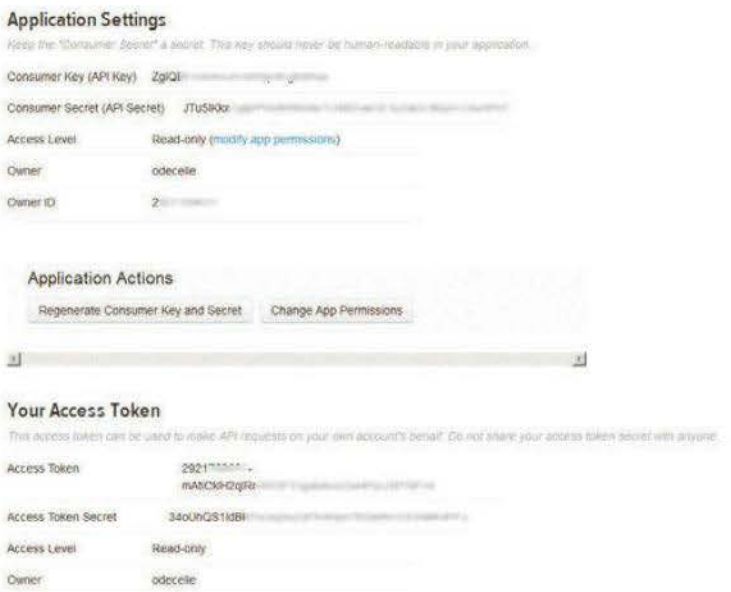


Figure 20-9
Jetons d'accès

L'onglet *Keys and Access Tokens* réunit tout ce dont vous avez besoin pour l'authentification Temboo. Copiez les clés afin de pouvoir les saisir par la suite dans Temboo ou laissez simplement cette page ouverte. Il nous faut encore régler un petit détail afin de disposer d'un accès en écriture à Twitter. Cliquez sur l'onglet *Permissions* pour définir le type d'accès dont vous avez besoin.

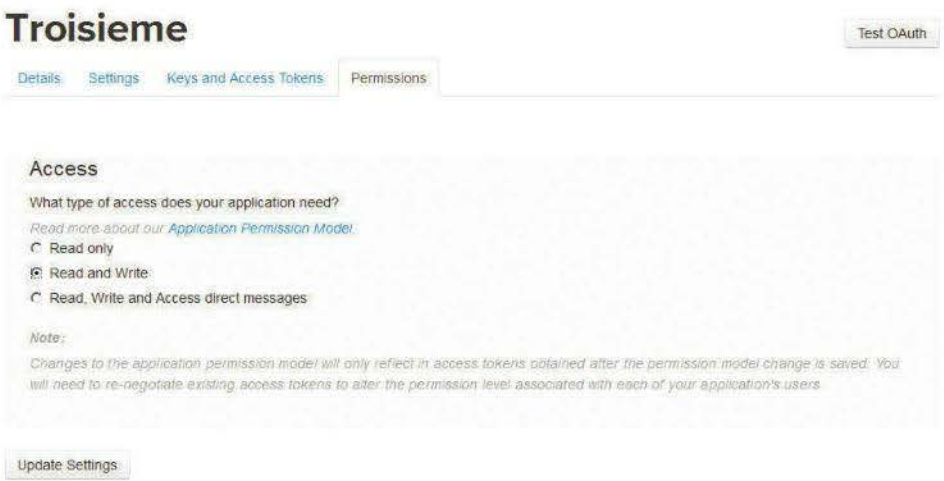


Figure 20-10 ▲
Options disponibles sous l'onglet Permissions des paramètres de l'application

Choisissez l'option *Read and Write,* puis revenez sous l'onglet *Settings* pour cocher l'option *Allow this application to be used to Sign in with Twitter*. Vous devez cliquer sur le bouton suivant afin d'appliquer les modifications effectuées sous chaque onglet :

Patientez quelques secondes, le temps que les modifications soient appliquées. Et voilà ! Tout est maintenant prêt du côté de Twitter. Nous pouvons donc revenir à Temboo.

De retour dans Temboo

Comme je l'ai déjà expliqué, nous devons saisir notre authentification afin de montrer patte blanche auprès de Temboo. Sélectionnez l'API Twitter dans la bibliothèque, puis cliquez sur la méthode *Statuses-Update* sous *Library>Twitter>Tweets*. Les champs de saisie suivants apparaissent au centre de la fenêtre du navigateur (voir figure 20-11).

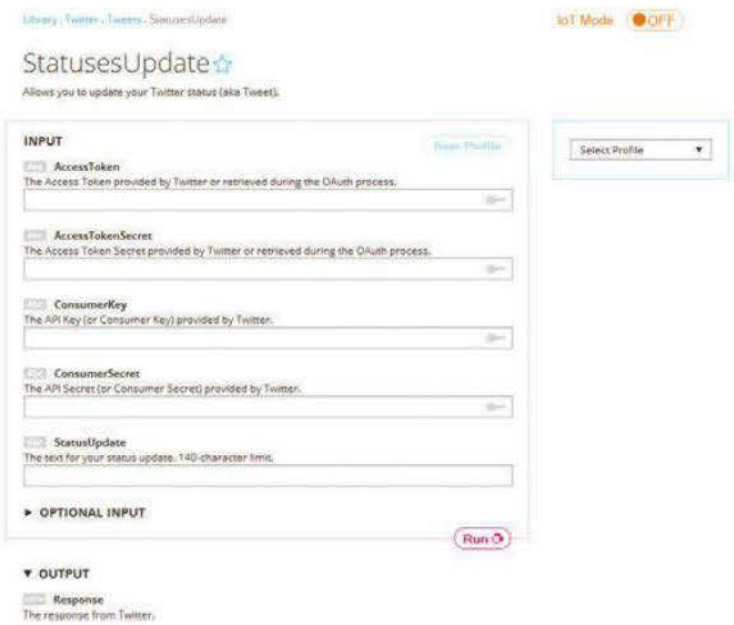


Figure 20-11
Saisie de l'accréditation pour l'API Twitter

Pour vous éviter d'avoir à saisir à plusieurs reprises les quatre clés que vous venez de générer, Temboo vous propose de les enregistrer. Saisissez les clés dans les champs correspondants, puis cliquez sur *Save Profile*.

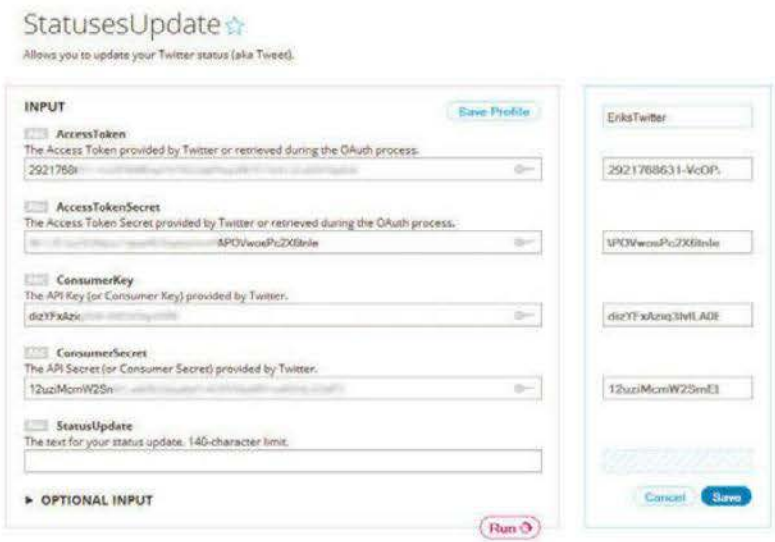


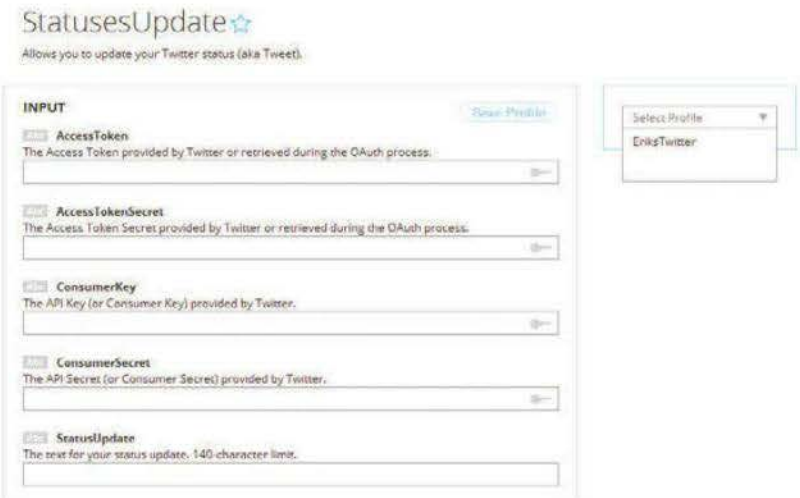
Figure 20-12
Profil enregistré

Nommez l'accréditation dans le champ qui apparaît en haut de la colonne de droite, puis cliquez sur le bouton *Save*. Une entrée portant le nom que vous venez de saisir apparaît dans le menu déroulant *Select Profile*.

Figure 20-13 ▶
Entrée du menu Select Profile

Il est maintenant temps de reprendre l'accréditation dans la fenêtre de la méthode *StatusesUpdate*, l'objectif étant évidemment de s'identifier auprès de Twitter. Cliquez sur le menu *Select Profile* qui se trouve en haut à droite de la fenêtre. La liste des entrées disponibles apparaît. Pour l'instant, il n'y en a qu'une, portant le nom que vous venez de saisir pour votre profil. Lorsque vous cliquez dessus, toutes les clés sont reprises dans les champs correspondants.

Figure 20-14 ▶
Reprise de l'accréditation pour l'API Twitter



Un champ n'est pas renseigné. Il s'agit du champ intitulé *StatusUpdate*.

C'est justement dans ce champ que vous devrez saisir le message qui sera affiché dans Twitter. Saisissez-y un texte de 140 caractères au maximum. Ensuite, vous pouvez vérifier directement dans votre navigateur que l'accès à Twitter a bien été établi. Voici comment procéder.

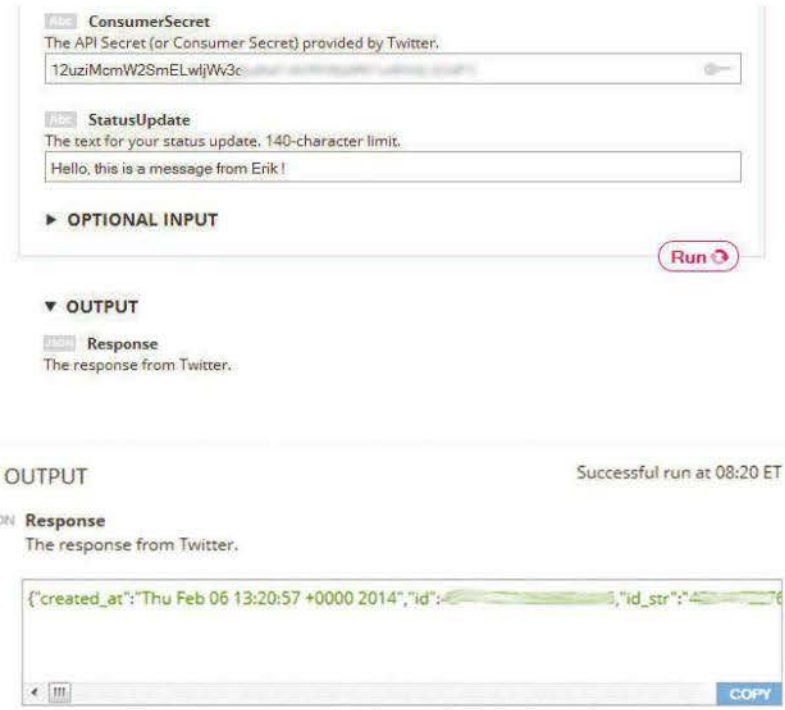


Figure 20-15
Message envoyé sur Twitter et sa réponse

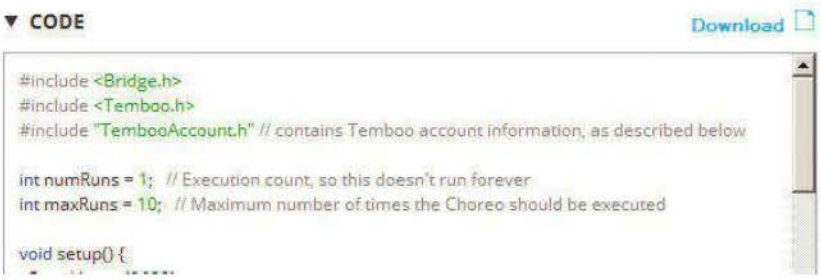
J'ai saisi un message pour le nouveau tweet dans le champ de texte, puis j'ai cliqué sur *Run*. Quelques secondes plus tard, la réponse est apparue en utilisant le format JSON en caractères verts sous la rubrique *OUTPUT*. Ouvrez Twitter. Votre message devrait y apparaître tôt ou tard. Si vous effectuez une série de tests avec Temboo alors que vous ne possédez pas de compte Twitter dédié aux tests, je vous recommande d'effacer les messages immédiatement après leur diffusion. Si vous rencontrez des problèmes lors de l'authentification auprès de Twitter, générez de nouvelles clés et de nouveaux jetons via les boutons *Regenerate* sous l'onglet *Keys and Access Tokens* des paramètres de votre application Twitter. Cette méthode m'a souvent permis de me tirer d'affaire. Dans le cas d'un problème, le message d'erreur a l'apparence suivante :

Figure 20-16
Problème d'authentification de l'API Twitter

Au tour de la Yún

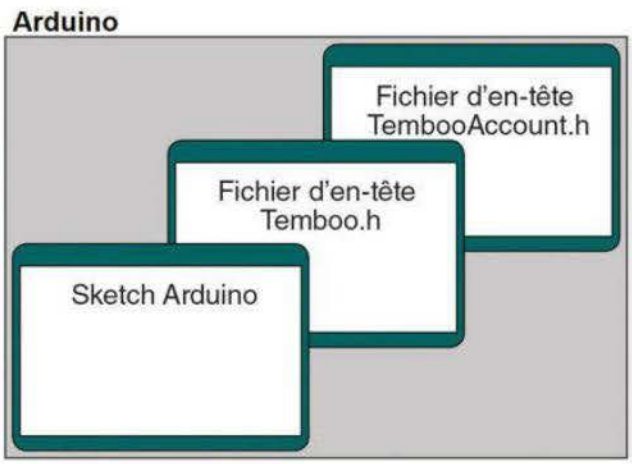
Il est maintenant temps de tout faire fonctionner avec la carte Yún. Le code correspondant est fourni sur le site web de Temboo. Faites défiler la page vers le bas jusqu'à la rubrique *Code*.

Figure 20-17 ▶
Exemple de code de l'API Twitter



J'ai déjà choisi la plateforme de destination appropriée, c'est-à-dire la carte Arduino Yún, dans la liste déroulante qui apparaît au milieu en haut lorsque l'on active l'*IoT Mode*. Temboo peut aussi générer un code adapté à d'autres plateformes, comme Java, Python ou Processing : il se montre donc d'une grande souplesse. Au début de ce montage, j'avais présenté une illustration où l'on pouvait voir que deux fichiers étaient nécessaires du côté Arduino. Ce n'est pas tout à fait vrai, car nous avons aussi besoin d'un fichier d'en-tête ayant pour rôle de fournir les informations du compte Temboo.

Figure 20-18 ▶
Les trois fichiers requis côté Arduino



Il s'agit d'une partie du sketch proprement dit et du fichier d'en-tête `Temboo.h` qui permet au fichier `Temboo.cpp` d'assurer la communication avec la partie Linux.

D'autre part, vous avez aussi besoin du fichier d'en-tête contenant les données du compte : il s'agit du fichier `TembooAccount.h` dont l'initialisation s'effectue comme illustré ci-après. Ce fichier est également disponible sur Internet.

▼ HEADER FILE

```
/*
IMPORTANT NOTE about TembooAccount.h

TembooAccount.h contains your Temboo account information and must be included
alongside your sketch. To do so, make a new tab in Arduino, call it TembooAccount.h,
and copy this content into it.
*/

#define TEMBOO_ACCOUNT "erikbartmann"  // Your Temboo account name
#define TEMBOO_APP_KEY_NAME "myFirstApp"  // Your Temboo app key name
#define TEMBOO_APP_KEY "f[redacted]"  // Your Temboo app key

/*
The same TembooAccount.h file settings can be used for all Temboo SDK sketches.
Keeping your account information in a separate file means you can share the
main .ino file without worrying that you forgot to delete your credentials.
*/
```

COPY

Figure 20-19
Fichier TembooAccount.h

Toutes les conditions préalables sont maintenant remplies et nous allons donc pouvoir créer notre sketch Arduino et l'exécuter. Le nouveau sketch se nomme *EriksTwitterUpdate*. J'ai ajouté le code du fichier d'en-tête `TembooAccount.h` dans l'environnement de développement Arduino après avoir cliqué sur l'icône *Nouveau* pour créer un nouvel onglet.

EriksTwitterUpdate | TembooAccount.h

```
1  #include <Bridge.h>
2  #include <Temboo.h>
3  #include "TembooAccount.h" // contains Temboo account information, as described below
4
5  int numRuns = 1;   // Execution count, so this doesn't run forever
6  int maxRuns = 10;   // Maximum number of times the Choreo should be executed
7
8  void setup() {
9    Serial.begin(9600);
10
11   // For debugging, wait until the serial console is connected.
12   delay(4000);
13   while(!Serial);
14   Bridge.begin();
15 }
```

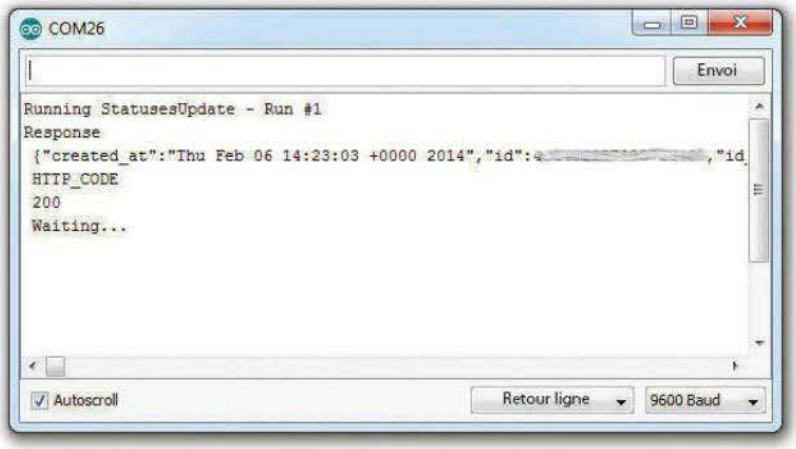
Figure 20-20
Sketch Arduino EriksTwitterUpdate (extrait)

Après le téléversement via le port COM, l'exécution du sketch est interrompue, car la ligne

```
while(!Serial);
```

prévoit qu'il ne se passe rien tant que le moniteur série n'est pas ouvert. Voici ce qui apparaît dans la fenêtre après son ouverture :

Figure 20-21 ▶
Affichage dans le moniteur série



Que signifie la mention HTTP CODE 200 qui est affichée dans le moniteur ? Est-ce que cela correspond à une erreur ?

Non, Ardus. C'est l'un des codes d'état HTTP qui est transmis par un serveur en réponse à une requête HTTP. La valeur 200 indique que la requête a reçu une réponse en bonne et due forme. Vous trouverez plus d'informations sur les codes HTTP à la page *http://fr.wikipedia.org/wiki/Liste_des_codes_HTTP*, par exemple.

Comme vous pouvez le constater, le tweet est bien arrivé sur ma page Twitter :

Figure 20-22 ▶
Le tweet a bien été publié.

Attention !

Lorsque vous programmez un sketch qui renvoie toujours le même tweet à brefs intervalles, une alarme retentit chez Twitter et le tweet suivant est bloqué. Si vous voulez tout de même afficher régulièrement un message à brefs intervalles – au risque d'irriter vos *followers* –, ajoutez un horodatage à votre tweet, par exemple.

Vous avez dit que ce code n'est pas très important, mais le fonctionnement de ce sketch m'intéresse tout de même. Peut-être aurai-je besoin de le modifier, qui sait ? Comment faire ?

Vous avez parfaitement raison, Ardus ! Examinons le sketch de plus près.

Déclaration globale

```
#include <Bridge.h>
#include <Temboo.h>
#include "TembooAccount.h" // contains Temboo account information

int numRuns = 1; // Execution count, so this doesn't run forever
int maxRuns = 10; // Maximum number of times the Choreo should be executed
```

Au début figurent les trois fichiers d'en-tête mentionnés précédemment. La variable `numRuns` compte le nombre d'émissions et `maxRuns` détermine le nombre maximum de tweets. C'est une mesure de précaution, car qui aimerait trouver des milliers de tweets sur sa page du jour au lendemain ?

Initialisation

```
void setup() {
  Serial.begin(9600);
  delay(4000);
  while(!Serial);
  Bridge.begin();
}
```

La fonction `setup` initialise aussi bien l'interface série que la passerelle (*bridge*). L'exécution du sketch reprend après l'ouverture du moniteur série. Vous devez commencer à comprendre comment cela fonctionne.

Envoi des tweets

Au début de la fonction `loop`, le système vérifie d'abord, d'après la variable `numRuns`, si le nombre maximum d'envois de tweets n'est pas atteint.

```
void loop() {
  if (numRuns <= maxRuns) {
    Serial.println("Running StatusesUpdate - Run #" + String(numRuns++));
    TembooChoreo StatusesUpdateChoreo;
    // Invoke the Temboo client
    StatusesUpdateChoreo.begin();
```

Le nombre d'exécutions est indiqué par la méthode `println` au moniteur série. Pour pouvoir utiliser les fonctionnalités de Temboo, il faut d'abord créer l'objet correspondant. On utilise à cette fin une instance de la catégorie `TembooChoreo`. À votre avis, où trouve-t-on cette définition de catégorie ? Essayez de la trouver. La méthode `begin` initialise l'objet en établissant une liaison avec Linino et en y activant le fichier Python `temboo`. Les trois lignes ou méthodes suivantes permettent de communiquer l'authentification du compte Temboo que nous avons insérée dans le fichier `TembooAccount.h` :

```
// Set Temboo account credentials
StatusesUpdateChoreo.setAccountName(TEMBOO_ACCOUNT);
StatusesUpdateChoreo.setAppKeyName(TEMBOO_APP_KEY_NAME);
StatusesUpdateChoreo.setAppKey(TEMBOO_APP_KEY);
```

Mais au fait, où sont utilisées les informations relatives aux clés générées par Twitter ? Ne faut-il pas aussi les insérer ici dans le code du sketch ?

Votre question tombe à pic ! Les clés sont bien communiquées, mais pas depuis le sketch. Seul le service web de Temboo connaît ces clés que nous venons d'enregistrer depuis la page Internet. En revanche, pour notre part, nous connaissons le nom sous lequel nous avons enregistré les quatre clés dans Temboo. Dans notre exemple, il s'agit d'EriksTwitter. Vous pouvez ensuite communiquer ce nom à la méthode `setCredential` qui récupère l'authentification sur le serveur de Temboo :

```
// Set credential to use for execution
StatusesUpdateChoreo.setCredential("EriksTwitter");
```

Nous pouvons maintenant insérer le tweet à diffuser via la méthode `addInput` dans l'objet `StatusesUpdateChoreo`, mais cela ne veut pas dire que le tweet est bel et bien publié.

```
// Set Choreo inputs
StatusesUpdateChoreo.addInput("StatusUpdate", "Hello, this is Erik!");
```

Ensuite, la choreo est identifiée dans la hiérarchie des API par la saisie de son chemin d'accès complet :

```
// Identify the Choreo to run
StatusesUpdateChoreo.setChoreo("/Library/Twitter/Tweets/StatusesUpdate");
```

Le chemin d'accès se termine par le nom de la choreo qui s'occupe de la diffusion. Enfin, le tweet est envoyé avec la méthode `run` :

```
// Run the Choreo; when results are available, print them to serial
StatusesUpdateChoreo.run();
```

La choreo répond à l'aide de la méthode `available` et la réponse est transmise par une boucle `while` au moniteur série.

```
while(StatusesUpdateChoreo.available()) {
  char c = StatusesUpdateChoreo.read();
  Serial.print(c);
}
```

La liaison est ensuite interrompue et les ressources sont remises à disposition des autres processus à l'aide de la méthode `close` :

```
  StatusesUpdateChoreo.close();
}
```

Le moniteur série affiche alors un message d'attente avant que la fonction `loop` ne déclenche le prochain envoi – sauf si le nombre maximum d'exécutions a été atteint :

```
  Serial.println("Waiting...");
  delay(30000); // wait 30 seconds between StatusesUpdate calls
}
```

C'est ainsi que se termine le sketch. Vous pouvez maintenant laisser libre cours à votre créativité. On pourrait imaginer la diffusion d'un tweet suite à l'interrogation d'un capteur pour signaler un événement particulier (la lumière est allumée, la cave est inondée, la porte du frigo est ouverte depuis plus d'une heure…). Ainsi, tous vos abonnés sauront ce qu'il se passe chez vous.

Donc, si je veux programmer mon propre sketch pour diffuser régulièrement la température qu'il fait dans mon local de travail, c'est possible ?

En effet, Ardus. Ça ne pose aucun problème. Je vous recommande d'employer un capteur spécial qui, en plus de la température, mesure aussi le taux d'humidité ambiante. Il s'agit du capteur DHT11.

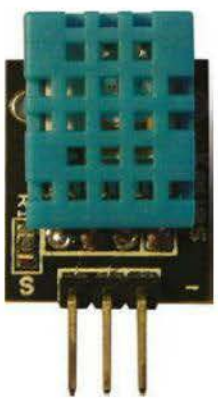


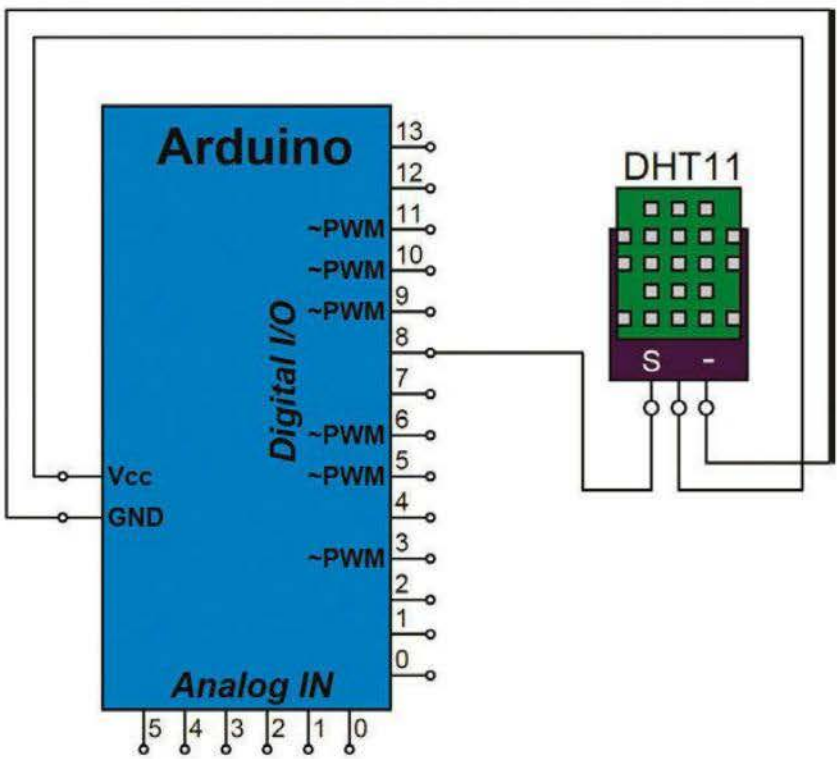
Figure 20-23
Capteur de température et d'humidité DHT11

Pour la lecture des valeurs de l'environnement, on peut faire appel à une bibliothèque qui est accessible à l'adresse suivante :

http://playground.arduino.cc/Main/DHTLib

Le capteur possède trois pattes de raccordement : deux pour l'alimentation électrique et la troisième désignée par la lettre S pour le transfert de données. Les pattes du modèle illustré ici n'ont que deux désignations : S (à gauche) et - (à droite). On peut donc en conclure que le + est au centre. Par conséquent, nous raccordons le capteur de la façon suivante à la carte Yún :

Figure 20-24 ▶
Raccordement du capteur de température et d'humidité DHT11



Certes, le schéma de montage montre une carte Arduino Uno et non une Yún, mais le capteur se connecte exactement de la même façon aux broches de cette dernière. Le montage est extrêmement simple et ne demande pas de constructions compliquées.

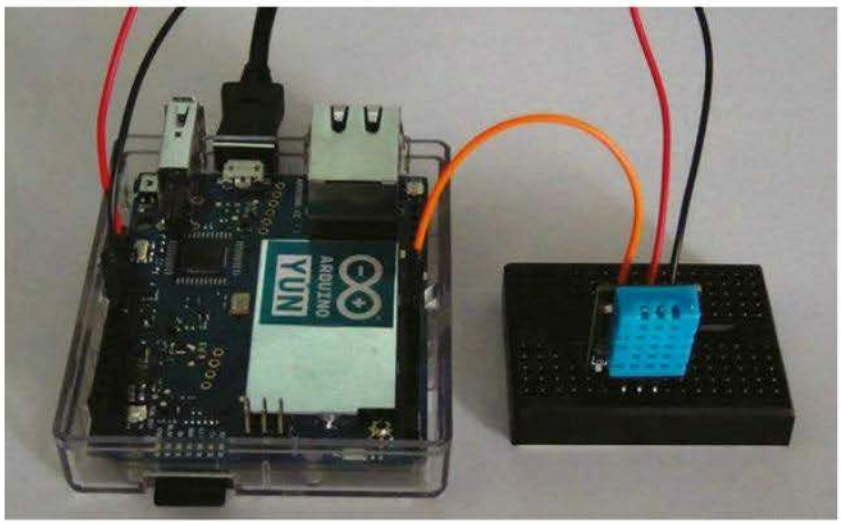


Figure 20-25
Construction du circuit avec la carte Yún et le capteur de température et d'humidité DHT11

Un tweet doit afficher l'humidité et la température du local à intervalles réguliers en précisant la date et l'heure de la mesure.

Figure 20-26
Le tweet reçu présente les données mesurées.

Examinons le code requis (je me concentrerai sur les ajouts au sketch précédent) :

Déclaration globale

```
#include <Bridge.h>
#include <Temboo.h>
#include "TembooAccount.h"
#include <dht.h>                // Bibliothèque du capteur DHT11
#define DHT11_PIN 8             // Broche de données du DHT11

dht DHT;                        // Instance DHT11

int numRuns = 1;                // Variable du nombre d'exécutions
int maxRuns = 30;               // 30 tweets seront publiés
```

Nous devons commencer par insérer dans le code la bibliothèque DHT11 précédemment importée et indiquer le numéro de la broche de données sur laquelle le capteur est raccordé. J'ai indiqué que le capteur devait être interrogé 30 fois, mais libre à vous de choisir un autre nombre de requêtes. Comme le tweet est horodaté, il ne risque pas d'être bloqué par Twitter. Je n'ai pas modifié l'initialisation.

Initialisation

```
void setup() {
  Serial.begin(9600);
  delay(4000);
  while(!Serial);
  Bridge.begin();
}
```

Détermination de la date et de l'heure

Le code suivant vous paraîtra familier.

```
String getDateTime() {
  Process time;
  time.runShellCommand("date"); // Date et heure du serveur
  String timeStamp = "";
  while (time.available()) {
    char c = time.read();
    timeStamp += c;
  }
  return timeStamp; // Affichage de l'horodatage
}
```

Envoi des tweets

Nous en arrivons aux lignes de code qui permettent l'envoi du tweet.

```
void loop() {
  if (numRuns <= maxRuns) {
    String timestamp = getDateTime(); // Détermination de l'horodatage
    String tweet = "";                // Message du tweet
    String msgDHT11 = "";             // Affichage de l'état du capteur
```

La variable `timestamp` enregistre aussi bien la date que l'heure. Ainsi, à la lecture du tweet, on sait exactement quand le message a été envoyé. Le tweet est composé des quatre informations suivantes :

- horodatage (`timestamp`)
- état du capteur
- humidité (`humidity`)
- température (`temperature`)

J'ai donc déclaré la variable `tweet` avec le type de donnée `String` afin de pouvoir y enregistrer toutes les informations. Au moment de son initialisation, le capteur DHT11 transmet un message qui rend compte de l'état du processus et qui est enregistré au moyen de la variable `msgDHT11`. Examinons le processus d'initialisation de plus près.

```
int responseDHT11 = DHT.read11(DHT11_PIN); // Initialisation de DHT11
switch (responseDHT11) { // Analyse de la réponse
  case DHTLIB_OK:
    msgDHT11 = "Sensor OK. ";
    break;
  case DHTLIB_ERROR_CHECKSUM:
    msgDHT11 = "Sensor FAIL. Checksum Error! ";
    break;
  case DHTLIB_ERROR_TIMEOUT:
    msgDHT11 = "Sensor FAIL. Timeout! ";
    break;
  default:
    msgDHT11 = "Sensor FAIL. Unknown Error! ";
  break;
}
```

La variable `responseDHT11` enregistre la réponse à l'initialisation afin de l'exploiter dans une instruction `switch`. Pour traduire le code de la réponse dans une forme lisible, la variable `msgDHT11` est transmise dans l'instruction `case` d'un message qui fera ensuite partie du tweet. Pour calculer l'humidité et la température ambiante, nous utilisons les deux méthodes de la catégorie `dht11`.

```
float humidity = DHT.humidity; // Mesure de l'humidité
float temperature = DHT.temperature; // Mesure de la température
```

Les valeurs mesurées sont enregistrées dans les variables `humidity` et `temperature` qui feront ensuite aussi partie du tweet. Les lignes suivantes ne nécessitent pas de longs discours, car vous commencez à en connaître la fonction.

```
Serial.println("Running StatusesUpdate - Run #" + String(numRuns++));
TembooChoreo StatusesUpdateChoreo;
// Invoke the Temboo client
StatusesUpdateChoreo.begin();
// Set Temboo account credentials
StatusesUpdateChoreo.setAccountName(TEMBOO_ACCOUNT);
StatusesUpdateChoreo.setAppKeyName(TEMBOO_APP_KEY_NAME);
StatusesUpdateChoreo.setAppKey(TEMBOO_APP_KEY);
// Set credential to use for execution
StatusesUpdateChoreo.setCredential("EriksTwitter");
```

Le tweet est maintenant composé à partir des bribes d'informations disponibles et il est présenté sous forme de message :

```
// Tweet
tweet = timestamp + " - " +
  "DHT11-Status: " + msgDHT11 + " - " +
  "Humidity: " + humidity +
  "% - Temperature: " + temperature + " degrés Celsius";
```

Puis il est ajouté à l'objet Twitter au moyen de la méthode `addInput` :

```
// Set Choreo inputs
StatusesUpdateChoreo.addInput("StatusUpdate", tweet);
```

Vous connaissez aussi le code suivant, donc il ne mérite pas que l'on s'y attarde. Le sketch se termine ainsi :

```
// Identify the Choreo to run
StatusesUpdateChoreo.setChoreo("/Library/Twitter/Tweets/StatusesUpdate");

// Run the Choreo; when results are available, print them to serial
StatusesUpdateChoreo.run();

while (StatusesUpdateChoreo.available()) {
  char c = StatusesUpdateChoreo.read();
  Serial.print(c);
}
StatusesUpdateChoreo.close();
}
Serial.println("Waiting...");
delay(30000); // wait 30 seconds between StatusesUpdate calls
}
```

Dans cet exemple, 30 tweets sont publiés à intervalles de 30 secondes. Ouvrez, d'une part, votre moniteur série pour y suivre les réponses de Temboo et, d'autre part, le compte Twitter sur lequel les tweets sont publiés. Adaptez le code à vos besoins afin de générer, par exemple, une succession infinie de tweets qui publient les valeurs ambiantes toutes les heures. N'hésitez pas à essayer les différents réglages. Les tweets vous permettent évidemment de communiquer toutes sortes d'informations mesurées par des capteurs (souvenez-vous néanmoins que vous ne disposez que de 140 caractères). Vous pourriez utiliser les capteurs suivants :

- capteur de luminosité
- détecteur de chocs
- capteur à effet Hall

Si vous ne voulez pas qu'un tweet soit généré à intervalles réguliers, vous pouvez très facilement modifier le code du sketch afin que certains événements seulement – c'est-à-dire le dépassement de certaines valeurs seuils prédéfinies – déclenchent la publication d'un tweet.

Qu'avez-vous appris ?

- Vous savez maintenant à quoi sert Temboo.
- Vous avez créé un compte Temboo afin d'avoir accès à ses fonctionnalités.
- Pour avoir accès à Twitter par le biais de Temboo, vous avez généré des *tokens* et des *keys* qui sont nécessaires pour l'authentification.
- Pour twitter des valeurs d'humidité et de température, vous avez raccordé un capteur DHT11 à votre carte Yún et vous avez chargé la bibliothèque DHT requise pour la lecture de ces valeurs.

Exercice complémentaire

Créez un sketch Arduino afin qu'un tweet ne soit diffusé que lorsqu'un seuil prédéfini d'humidité ou de température est atteint.