

## Introduction à l'Internet des Objets, un capteur hygrométrique pour jardin.

Auteur : Paul Pinault

Passionné par l'Internet des Objets, auteur du blog [disk91.com](http://disk91.com), référence sur ces technologies, je réalise des objets et services connectés à titre professionnel comme par curiosité. Ceci alimente mes connaissances que je partage ensuite sur ma chaîne Youtube au travers de MooC et sur mon blog. Contact : [@disk\\_91](https://twitter.com/disk_91) / [www.disk91.com](http://www.disk91.com)



Nous allons voir dans cet article comment utiliser l'Internet des Objets (IoT) pour une réalisation simple telle qu'un capteur d'hygrométrie pour jardin. Cette création est née de la volonté de mon fils de créer un petit jardin potager. Ce jardin doit être arrosé régulièrement, mais comment ne pas oublier cette action primordiale ?

Pour ce faire, j'ai choisi d'utiliser le réseau LoRaWan Helium et initialement des modules RAK Wireless Wisblock, mais je vous présenterai dans cet article une solution basée sur une carte LoRa radio node qui est plus accessible aux débutants puisque 100% basée sur Arduino.

### L'IoT, des objets autonomes, capables de communiquer des années avec très peu d'énergie

L'Internet des Objets peut avoir de nombreuses définitions et intégrer un très large panel d'applications et de technologies. Pour ma part, j'aime définir l'IoT comme étant une catégorie d'objets capables de communiquer de façon totalement autonome (sans le relai d'un téléphone ou d'une box), sur de très longues durées, ceci pouvant être plusieurs mois à plusieurs années grâce à une capacité à communiquer très peu énergivore. Cette caractéristique innovante a permis l'émergence du terme IoT à partir de 2012 environ auprès du grand public et de l'industrie. De nombreux objets (ou plutôt machines) avaient préalablement des capacités de communications reposant sur des technologies conventionnelles impliquant en général une autonomie faible (technologies type GSM, LTE...) ou le recours à une passerelle locale (technologies BLE, Zigbee, Wifi ...) Ces technologies étaient qualifiées de Machine to Machine (M2M).

Si cette classification peut être tout à fait discutée et si personnellement je trouve le périmètre de l'IoT particulièrement flou, au point que l'on puisse y consacrer des ouvrages entiers, je vous propose de l'accepter le temps de cet article pour comprendre que ce dont nous allons parler est bien de l'IoT tel que défini ci-dessus.

### Des réseaux dédiés proposant un très bas cout de déploiement et d'usage

Il existe plusieurs technologies IoT : LoRaWan, Sigfox, Kineis, Wize ... pour ne citer que celles qui sont nées en France. Mais il ne faut pas oublier non plus NB-IoT, la déclinaison IoT des technologies issues du 3GPP, les technologies déployées en même temps que les réseaux mobiles (4G, 5G).

Ces technologies, pour que les objets puissent communiquer de façon autonome, doivent reposer sur la présence d'un réseau capable de capter leurs communications. Dans le cas de Kineis, par exemple, ce réseau est composé d'une flotte de satellites. Pour les autres, il s'agit de réseaux de communication terrestre. Ces réseaux doivent répondre à deux critères importants : une large couverture territoriale et un coût de déploiement faible. A titre de comparaison, un réseau 5G, avec 9000 antennes couvre 41% de la population Française et chaque antenne va coûter entre 50.000 et 150.000 euros pour être installée, sans compter l'achat des bandes de fréquences nécessaires.

Des réseaux terrestres comme Sigfox ou LoRaWan vont nécessiter, respectivement, entre 1500 et 5000 antennes pour une couverture de plus de 95% de la population, chacune coûtant seulement quelques milliers d'euro et sans achat préalable de fréquences.

Cette approche à bas coût permet alors de bénéficier d'un coût de connectivité très faible pour chaque objet.

### LoRaWan, des réseaux dédiés à l'IoT, accessibles au grand public

Dans les technologies citées, une grande partie s'est tournée vers des marchés professionnels, c'est le cas de Wize, Kineis qui ne sont pas réellement accessibles aux particuliers. Sigfox propose une accessibilité meilleure auprès des makers mais reste très orienté professionnels. Toutes ces technologies sont proposées par un unique opérateur qui va privilégier un modèle et une cible commerciale.

Les réseaux basés sur la technologie LoRa et les protocoles LoRaWan vont eux permettre plus de diversité d'offre car cette technologie, propriété d'un fondateur de chip (Semtech), n'est pas associée à un opérateur.

Ainsi, s'il existe plusieurs offres commerciales en France (quasi-exception mondiale) avec Orange et Objenious adressant un marché professionnel avant tout. Mais il existe aussi des offres communautaires (TheThingsNetwork et Helium) accessibles à tous.

### TheThingsNetwork (TTN), quand l'esprit de l'Open-Source s'empare des télécoms

Lancé en 2015 alors que la spécification de LoRaWan n'est pas encore aboutie, TheThingsNetwork reprend les standards de l'Open-source pour créer une offre nouvelle : un réseau gratuit, déployé par des contributeurs volontaires, partout dans le monde, adossé à une offre payante associée à une meilleure qualité de service.

Ce réseau est un succès au point de devancer les offres des autres opérateurs en atteignant fin 2019 60% de part de marché dans le trafic LoRaWan dans les zones couvertes par le réseau. Fort de 20.000 antennes déployées par des particuliers, associations et entreprises, il est entre 2015 et 2020 le premier réseau LoRaWan mondial.

Utiliser ce réseau est gratuit, s'il n'est pas présent dans votre localité, l'étendre par vous-même et votre voisinage, ne vous coûtera qu'une petite centaine d'euros et quelques minutes de configuration.

### Helium, l'uberization des télécoms est en route

Lancé en 2020, Helium, un réseau LoRaWan, lui aussi basé sur le déploiement de l'infrastructure par des particuliers est devenu en moins d'un an le premier réseau LoRaWan à l'échelle de la planète. Avec 105.000 antennes déployées en un peu plus d'une année et

des perspectives à 500.000 antennes à horizon 18 mois, il est déjà plus gros que la somme de tous les réseaux publics existant sur LoRaWan. Son secret ? L'adjonction d'un fonctionnement décentralisé piloté par une blockchain, adossée à un Token qui crée un encouragement économique auprès de ceux qui déploient le réseau.

Finalement, si la couverture Française est encore un peu en retrait par rapport à TTN à ce jour, là où le réseau est déployé il est en général plus dense que les réseaux concurrents offrant ainsi une meilleure qualité de service.

Helium n'est pas un réseau gratuit, mais comme 10.000 messages sont offerts et que 100.000 messages supplémentaires ne coûtent que \$1 il en est tout comme.

Ce réseau étant nouveau et à l'avenir prometteur, c'est sur celui-ci que je vais m'appuyer pour la suite de cet article.

## LoRa et LoRaWan

Avant de rentrer dans le dur du sujet, faisons un dernier point sur LoRa : il s'agit d'une technologie radio permettant de couvrir de longues distances avec peu d'énergie. Pour cela, elle offre une très grande résilience au bruit radio grâce à un principe de déplacement de fréquence. Grosso-modo, les valeurs à transmettre vont l'être par une émission sur une fréquence qui va croître ou décroître. Cette croissance / décroissance qui constitue un mouvement de fréquence va permettre de coder la valeur émise.

Grâce à cela, un message, émis avec une puissance de 25mW (par comparaison WiFi est de 100mW) va pouvoir être reçu à une distance de plus de 10km (jusqu'à 50km dans de très bonnes conditions, 400km de façon exceptionnelle).

Pour arriver à cet exploit, il faut toutefois faire quelques compromis, ici, c'est celui de la vitesse de transmission. LoRa permet, dans ce que nous allons utiliser, des débits entre 250 bits / s et 5400 bits / s. Pas de quoi tout faire donc.

LoRaWan est une couche de protocole permettant d'utiliser la technologie radio LoRa dans le cadre de réseaux multiples. Ce protocole qui doit donc être implémenté dans l'objet, définit les interactions entre celui-ci et un cœur de réseau appelé LoRa Network Server (LNS). Il existe plusieurs implémentations du protocole LoRaWan. Nous allons utiliser la version LMIC produite initialement par IBM, mieux adaptée aux environnements légers comme Arduino que la stack Semtech plus complète mais plus gourmande.

## Rak Wisblock une sorte de Légo pour l'IoT



Comme je vous l'ai dit, j'ai initialement réalisé ce projet avec un kit de développement RAK Wireless Wisblock. Ce kit à, l'avantage de ne pas requérir de soudure, d'avoir un processeur performant et de nativement fournir une solution d'harvesting solaire dans un boîtier étanche qui correspond tout à fait à ce cas d'usage. Vous trouverez tous les détails de cette solution sur mon blog.



Cette solution reste très adaptée au projet mais son prix avec les taxes d'importation dépasse les 100€. Je vais donc vous proposer une solution à moindre coût basée sur une carte LoRa radio node.

## LoRa Radio Node, tout en en pour petits projets IoT



Cette carte que l'on peut trouver sur les sites classiques fournissant du matériel à bas cout vaut environ 10 à 15€ et possède tout ce qu'il faut pour un petit objet connecté.

Elle réunit sur une même carte une Arduino type Pro Mini avec un module de communication LoRa RFM95. Il vous faudra choisir un modèle 868MHz qui correspond à la fréquence utilisée en Europe. Elle possède en outre des

connecteurs d'extension pour lui ajouter des capteurs supplémentaires. Pour être programmée, il faudra lui adjoindre un adaptateur USB-UART communément appelé câble FTDI. Le Qr-code ci-joint vous renvoie vers un de mes articles qui détaille comment connecter cette carte à votre ordinateur pour la programmer en utilisant le logiciel Arduino.

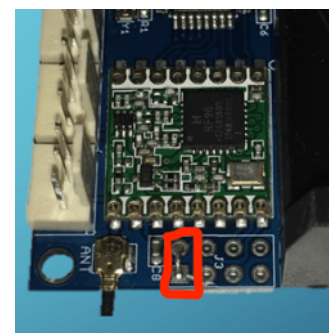


Cette carte peut être alimentée de différente façon : la plus pratique consiste à utiliser une batterie à placer sur le support. Malheureusement, la communication radio sur cette carte demande un pic de courant autour de 120mA et beaucoup de piles ne vont pas supporter cela. Si vous souhaitez utiliser une batterie rechargeable dans le support, il faudra opter pour des LiFeSo4 qui n'intègrent pas une grande capacité d'énergie. Pour cette raison, j'utilise souvent 2 piles Lithium au format ½ AA de 3.6V ou 3V offrant alors une tension totale de 6V à 7.2V et 1000 à 1200mAh.

Une fois l'Arduino programmé pour réduire sa consommation d'énergie entre deux mesures, cet objet pourra fonctionner des mois/années durant sans changement des piles.

Cette carte a toutefois certaines limites, comme nous l'avons vu, la pile de protocole LoRaWan est gourmande en ressources et sur le petit 328P avec seulement 32Kb de mémoire flash et 2Kb de mémoire RAM, il ne restera de la place que pour un programme utilisateur simple. La carte est donc tout à fait adaptée à capter d'hygrométrie d'un sol, la température / pression externe, une présence ... mais elle ne permettra pas beaucoup plus. Pour aller au-delà, les kits comme Wisblock offrent une plus grande liberté.

En premier lieu, nous allons devoir légèrement modifier le kit LoRa Radio Node pour lui permettre de fonctionner sur LoRaWan. En effet, il faut réaliser un pont de soudure pour permettre au signal d'interruption de réception de message d'être reçu par le microcontrôleur sur l'un de ses ports d'entrée / sortie.

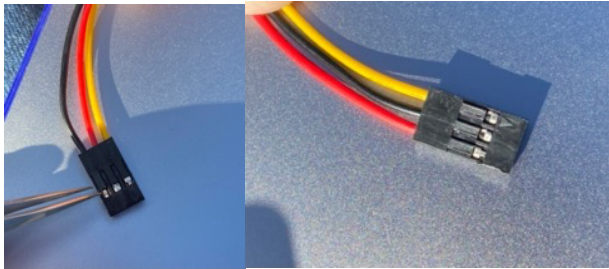


## Capter l'humidité du sol



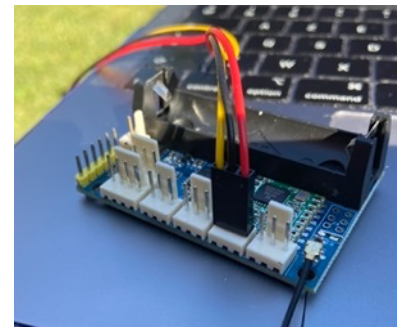
Il est possible de réaliser un capteur très simple avec deux clous comme je n'ai illustré dans l'article présent sur mon blog, mais nous pouvons aussi nous simplifier les choses avec un capteur capacitif que l'on trouvera facilement pour 1 ou 2 € tel que celui-ci. Le connecteur est presque adapté à notre carte, il faudra toutefois

veiller à inverser le fil rouge et le fil noir correspondant à l'alimentation pour que tout fonctionne correctement. Pour cela avec une aiguille ou une pince fine, il faut relever la lame plastique qui retient les pattes.

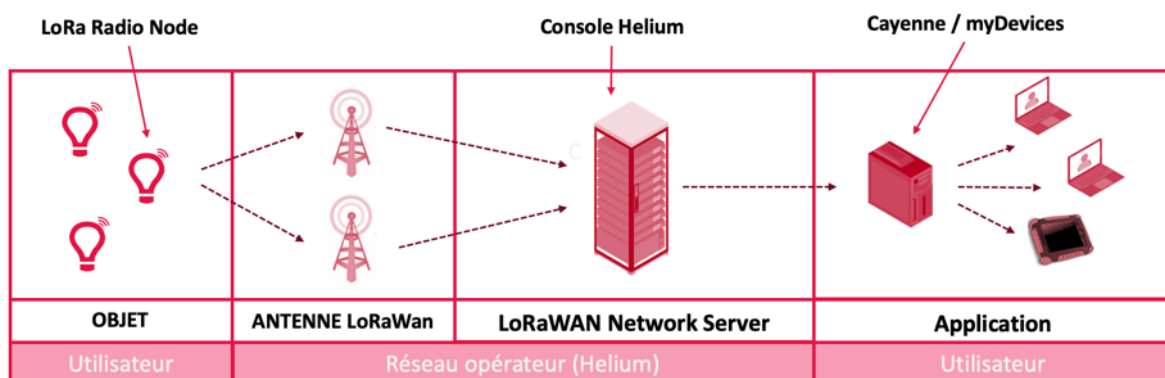


Le principe de ce capteur est de détecter l'humidité du sol par l'impact que cela a sur la capacité du capteur. Il transforme cette information en une tension variable sur sa patte analogique que nous allons pouvoir lire par la suite sur l'Arduino.

Ce capteur nécessite un courant de 700uA qui sera significatif dans la consommation et donc l'autonomie de la solution. Il serait donc opportun de piloter son alimentation, lors des mesures uniquement, en utilisant un GPIO. Toutefois, pour une première version simplifiée, nous allons l'alimenter en continu en le branchant directement sur l'un des ports de la carte comme ci-contre. Nous choisissons le connecteur permettant d'accéder au port analogique A0.



### Les différents composants de notre solution



Nous allons donc réaliser la programmation d'un objet qui va émettre une donnée, de façon régulière, vers un réseau LoRaWan. Pour cela, nous aurons besoin de déclarer cet objet auprès de cet opérateur, au niveau du network server, ici la console Helium. Enfin nous souhaitons visualiser ces données dans un rapport graphique accessible depuis Internet. Pour cela, nous allons utiliser la solution Cayenne de myDevices qui permet une intégration rapide et gratuite des données de notre capteur.

Ceci compose les 3 niveaux d'une solution IoT : l'objet, le réseau et l'application qui traite les données.

## Installation de l'environnement Arduino

La carte nécessite l'installation de l'environnement Arduino (disponible sur [www.arduino.cc](http://www.arduino.cc)) et utilise une carte « Arduino Pro ou Pro mini » avec comme sous-type « Atmega 328P (3.3V, 8MHz) ».

Vous aurez besoin en complément d'installer la librairie permettant de communiquer sur LoRaWAN. Pour cela, aller dans le menu « outil » puis le sous-menu « Gérer les bibliothèques ».

Recherchez la bibliothèque « MCCI LoRaWAN LMIC library » écrite par IBM, Matthijs Kooijman... J'utilise pour cet exemple la version 3.3

Avec ceci, l'environnement est prêt.

Nous allons aussi utiliser la bibliothèque « Arduino Low Power » qui nous permettra d'endormir l'Arduino entre deux mesures et donc économiser fortement la batterie. Installez cette bibliothèque.

## Fonctionnement de la LMIC

Pour utiliser la bibliothèque LMIC, il est nécessaire de configurer la zone géographique dans laquelle nous utilisons LoRaWAN. En effet la fréquence des communications en Europe est de 868MHz alors que les USA utilisent du 915MHz qui est la valeur par défaut.

Pour se faire, nous allons éditer le fichier

*Documents/arduino/libraries/MCCI\_LoRaWAN\_LMIC\_library/project\_config/lmic\_project\_config.h*

Il doit contenir uniquement les lignes suivantes :

```
#define CFG_eu868 1 // configuration Europe 868MHz
#define CFG_sx1276_radio 1 // composant utilisé pour la communication
#define LMIC_LORAWAN_SPEC_VERSION LMIC_LORAWAN_SPEC_VERSION_1_0_2
#define DISABLE_BEACONS // suppression de code inutile
#define DISABLE_PING // suppression de code inutile
```

Cette configuration faite, la bibliothèque est utilisable.

La bibliothèque LMIC a un fonctionnement événementiel qui peut être un peu perturbant car les opérations sont exécutées de façon asynchrone par l'appel, dans la boucle principale d'une fonction « `os_runloop_once()` » qui déclenchera les opérations nécessaires.

Nous allons donc pouvoir écrire un premier programme permettant d'envoyer un message chaque 5 minutes. Mais avant de commencer il faut que nous ayons des identifiants utilisables sur le réseau.

## Configuration du cœur de réseau pour recevoir et traiter les messages de notre objet

Nous allons maintenant inscrire notre objet sur le LoRaWAN Network Server (LNS) Helium qui dans la terminologie Helium s'appelle un « routeur » et qui s'utilise avec un composant « console » que l'on accède sur <https://console.helium.com>

Il faudra dans un premier temps créer un compte sur cette console. C'est gratuit et vous obtenez 10.000 crédits de communication. Chaque crédit correspond, pour faire simple, à l'émission d'un message. Bref pour notre exemple ça va être plus d'un an de communication

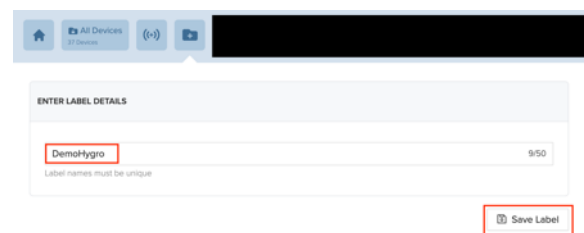
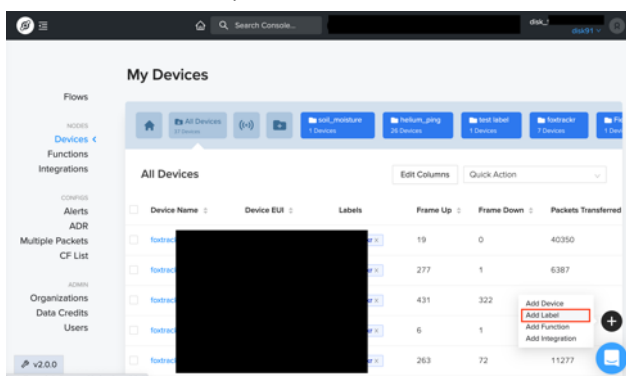


offert. Si vous souhaitez prolonger au-delà, sachez que 10.000 messages correspondent à une valeur de \$0.1.

Pour configurer notre objet, il y a quelques concepts à connaître :

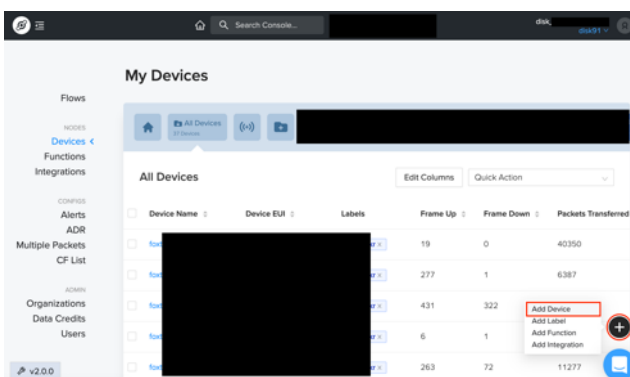
- Un « device » correspond donc à un objet et chacun à des clefs DEVEUI et DEVKEY qui lui sont propres et utilisées pour signer et chiffrer les messages.
- Un « label » correspond à un groupe d'objet pour lesquels nous allons avoir des opérations communes comme les intégrations. A un label correspond un APPEUI qui sera commun aux objets.
- Une « intégration » décrit comment transmettre les données reçues par le router vers l'application (pour nous cayenne myDevices)
- Un « flow » décrit les opérations à effectuer et les liens entre un « label » et une « intégration »

Pour commencer, il faut créer le « label »



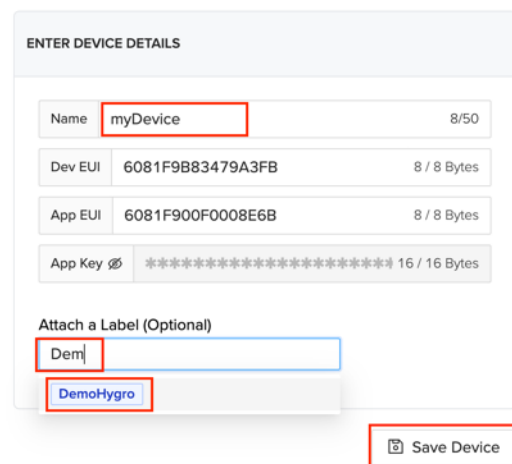
Il faut donner un nom à ce « label », ici DemoHygro. Vous pouvez mettre autre chose, ceci n'a pas d'importance tant qu'il est réutilisé par la suite.

Alors, nous pouvons créer un « device » pour obtenir les clefs nécessaires à notre objet.



#### Add New Device

Important: The first time a device joins the Network could take up to 20 mins

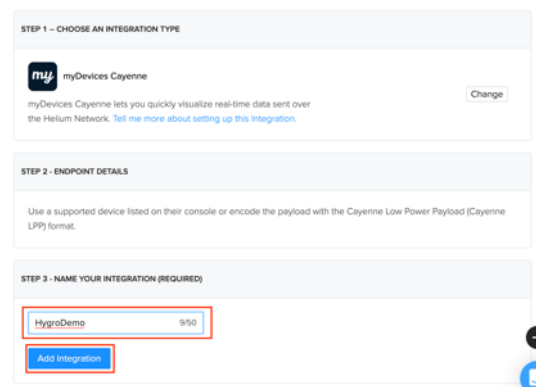
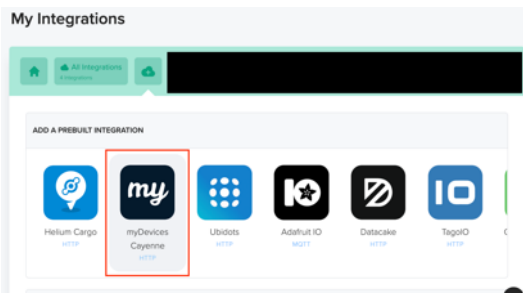
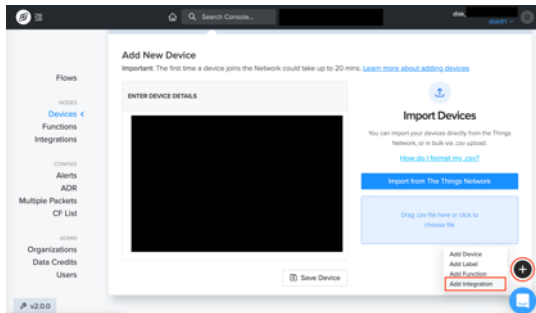


The screenshot shows the 'ENTER DEVICE DETAILS' form for adding a new device. The 'Name' field contains 'myDevice' and is highlighted with a red box. The 'Attach a Label (Optional)' section has 'DemoHygro' selected from a dropdown menu, also highlighted with a red box. A 'Save Device' button is visible at the bottom right of the form.

Nous donnons à l'objet le nom de notre choix (ici myDevice) les Dev EUI / App EUI / App Key sont automatiquement générés. Pour l'instant nous ne nous en occupons pas.

Il est important de rattacher cet objet au « label » que nous avons précédemment créé en le recherchant et en le sélectionnant pour le faire apparaître. Une fois ceci fait, nous pouvons enregistrer la création.

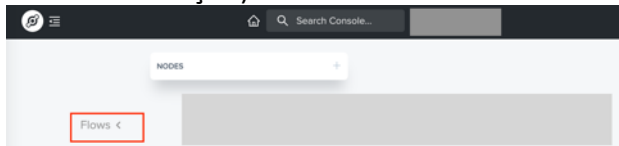
Nous allons maintenant pouvoir créer l'intégration vers le service « cayenne myDevices » qui nous permettra de consulter facilement les données et dont nous verrons les détails par la suite.



Sélectionner myDevices Cayenne puis entrez simplement un nom pour cette intégration et validez.

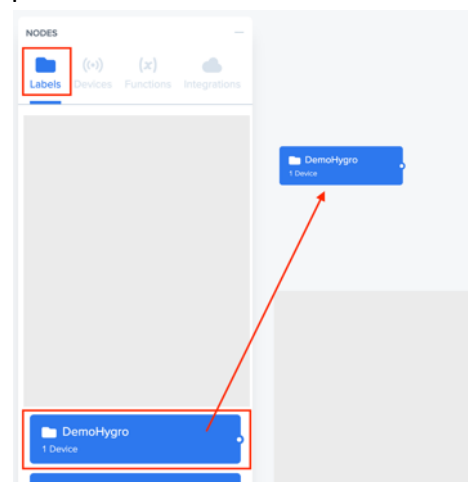
L'avantage d'utiliser une intégration prédéfinie est qu'il n'est pas nécessaire de rentrer les détails techniques relatifs à cette intégration.

Nous allons maintenant créer le « flow » qui permet de relier cette intégration au label que nous avons créé précédemment. Ainsi, chaque message reçu par un « device » attaché au « label » HygroDemo exécutera l'intégration HygroDemo (puisque nous les avons nommés de la même façon).

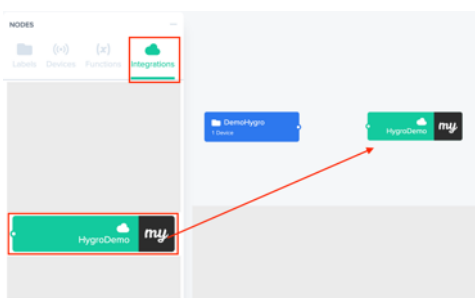


Sélectionner « Flows » dans le menu principal.

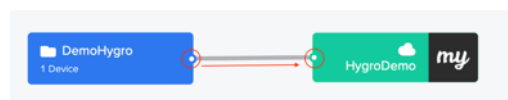
Puis cliquer sur « Nodes » et sélectionner les « Labels ». Choisir celui que nous venons de créer et le faire glisser (drag & drop) sur la zone de construction du « flow ».



Faire de même avec « l'intégration » après avoir sélectionné « intégration dans la barre de menu en haut ».



Ainsi nous avons l'ensemble des éléments que nous devons maintenant relier. Pour cela, relier les connecteurs des deux blocs en cliquant et étirant à l'aide de la souris.





Pensez bien à « sauvegarder » le « flow » créé en cliquant sur le bouton en bas de page.

## Récupération des identifiants réseau

Lors de la configuration précédente, nous avons créé un Device et avons aperçu ses identifiants. Nous allons devoir l'injecter dans le programme ci-dessous. Pour accéder aux identifiants, dans la console Helium, il faut sélectionner le Device pour afficher ses informations.

DEVICE DETAILS

Name myDevice

ID d91e2984-8c...

Device EUI msb: 0x60, 0x81, ...

App EUI msb: 0x60, 0x81, ...

App Key msb: 0xC3, 0x6F, ... 0x55, 0xA6, 0x4B, 0x9B, 0xEC, 0x6D

Activation Method **OTAA**

Nous allons cliquer sur l'icône avec les doubles flèches pour afficher les identifiants sous une forme facilement copié/collable sur Arduino et ensuite utiliser l'icône de copie pour mettre cette partie de l'identifiant dans le presse-papier. Il faut alors mettre cela dans le programme ci-dessous.

## Programme de captation hygrométrie

Nous pouvons donc écrire notre programme qui va émettre la donnée lue sur le capteur d'hygrométrie toutes les 5 minutes sur le réseau Helium

```
#include <lmic.h>
#include <hal/hal.h>
#include <LowPower.h>

const lmic_pinmap lmic_pins = {
    .nss = 10,
    .rxtx = LMIC_UNUSED_PIN,
    .rst = 9,
    .dio = {2, 5, LMIC_UNUSED_PIN},
};

#define TXPERIOD (5*60) // 5 minutes

static const ul_t PROGMEM APPEUI[8] = {0x60, 0x81, 0xF9, 0x00, 0xF0, 0x00, 0x8E, 0x6B};
static const ul_t PROGMEM DEVEUI[8] = {0x60, 0x81, 0xF9, 0xB8, 0x34, 0x79, 0xA3, 0xFB};
static const ul_t PROGMEM APPKEY[16] = {0xC3, 0x6F, 0xAE, 0x48, 0xDD, 0xB3, 0xE7, 0x38, 0x91,
0x2A, 0x55, 0xA6, 0x4B, 0x9B, 0xEC, 0x6D};

void os_getArtEui (ul_t* buf) {
    for ( int i = 0 ; i < 8 ; i++ ) buf[7-i] = APPEUI[i];
}
void os_getDevEui (ul_t* buf) {
    for ( int i = 0 ; i < 8 ; i++ ) buf[7-i] = DEVEUI[i];
}
void os_getDevKey (ul_t* buf) {
    memcpy_P(buf, APPKEY, 16);
}
```

```

void setup() {

    os_init();    // Intialisation de la bibliothèque
    LMIC_reset();
    LMIC_setClockError(MAX_CLOCK_ERROR * 10 / 100); // agrandit la fenetre de reception
    LMIC_setAdrMode(0); // desactive le mode ADR
    // Configure les canaux utilisables pour les communications
    LMIC_setupChannel(0, 868100000, DR_RANGE_MAP(DR_SF12, DR_SF7), BAND_CENTI);
    LMIC_setupChannel(1, 868300000, DR_RANGE_MAP(DR_SF12, DR_SF7B), BAND_CENTI);
    LMIC_setupChannel(2, 868500000, DR_RANGE_MAP(DR_SF12, DR_SF7), BAND_CENTI);
    LMIC_setupChannel(3, 867100000, DR_RANGE_MAP(DR_SF12, DR_SF7), BAND_CENTI);
    LMIC_setupChannel(4, 867300000, DR_RANGE_MAP(DR_SF12, DR_SF7), BAND_CENTI);
    LMIC_setupChannel(5, 867500000, DR_RANGE_MAP(DR_SF12, DR_SF7), BAND_CENTI);
    LMIC_setupChannel(6, 867700000, DR_RANGE_MAP(DR_SF12, DR_SF7), BAND_CENTI);
    LMIC_setupChannel(7, 867900000, DR_RANGE_MAP(DR_SF12, DR_SF7), BAND_CENTI);
    LMIC_setupChannel(8, 868800000, DR_RANGE_MAP(DR_FSK, DR_FSK), BAND_MILLI);
    // Configure la vitesse et la puissance de transmission
    LMIC.dn2Dr = SF9;
    LMIC_setDrTxpow(DR_SF9,14);
    LMIC_setLinkCheckMode(0);
}

boolean canSleep = true;
uint32_t temps = TXPERIOD; // transmettre au demarrage
uint8_t data[] = {
    0x01, 0x68, 0x00, // canal 1, type de donnée Hygrometrie 1 ut = 0.5%
};

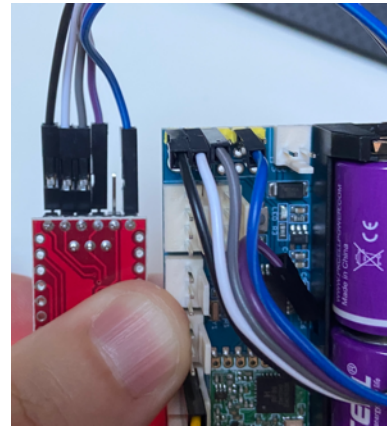
void loop() {
    if ( temps >= TXPERIOD ) {
        canSleep = false;
        // lecture de la donnée du capteur
        int16_t v = analogRead(A0); // 100 % = 313 | 50% = 678
        data[2] = map(v, 636, 313, 0, 100) * 2;
        // emission de la valeur
        LMIC_setDrTxpow(DR_SF9,14);
        lmic_tx_error_t err = LMIC_setTxData2(1, data, sizeof(data), 0);
        temps=0;
    }
    // put your main code here, to run repeatedly:
    os_runloop_once();
    if ( canSleep ) {
        LowPower.powerDown(SLEEP_8S, ADC_OFF,BOD_OFF);
        temps += 8;
    }
}

void onEvent (ev_t ev) {
    switch(ev) {
        case EV_JOINED:
            LMIC_setLinkCheckMode(0);
            break;
        case EV_TXCOMPLETE:
            canSleep = true;
            break;
        default:
            canSleep = false;
            break;
    }
}
}

```

## Programmation de la carte

Pour programmer la carte, il est nécessaire d'utiliser un câble ou une carte FTDI qui permet de faire communiquer un microcontrôleur utilisant une liaison série avec une ordinateur via son port USB. Ces cartes sont faciles à trouver sur les sites marchant pour quelques euros. Il faudra relier les pins du programmeur avec celle de la carte comme indiqué ci-contre. Nous allons relier les masses (GND) ensemble, croiser RX et TX et relier DTR ensemble. L'alimentation VCC n'est pas reliée du côté de notre carte car celle-ci sera fournie par les piles. Il est donc important lors de la programmation de s'assurer que les piles sont insérées dans la carte et fournissent une tension supérieure à 3,2V.




## Vérification

Nous pouvons maintenant consulter la console Helium pour visualiser les messages transmis par l'objet. Il faut pour cela sélectionner l'objet dans la partie « Device » et descendre dans la page pour visualiser les communications.

Si aucune communication n'apparaît, il est alors nécessaire de vérifier quelques points :

- Êtes-vous bien couvert par le réseau Helium ? (voir [mappers.helium.com](https://mappers.helium.com))
- Avez-vous bien entré les identifiants de l'objet dans le programme ?
- L'objet est-il bien inscrit sur le réseau (il y a un temps d'attente entre la création et son inscription qui peut atteindre 15 minutes)

Normalement vous devriez voir les communications suivantes dans l'écran de l'objet :

Event	Type	No. of Hotspots	Time
+  0	Uplink	1	Sep 12, 2021 6:10:09.119 PM
+  0	Join Accept	1	Sep 12, 2021 6:10:07.482 PM
+  0	Join Request	1	Sep 12, 2021 6:10:05.481 PM

Le message JOIN REQUEST est envoyé par l'objet pour rejoindre le réseau. Ceci est la première étape d'échange des clés permettant l'ouverture d'une session de communication chiffrée. Le message JOIN ACCEPT est envoyé en réponse par le réseau à l'objet pour ouvrir cette session. Cet échange n'aura lieu qu'une seule fois lors de la première communication. Il peut arriver que plusieurs communications comme celles-ci s'enchaînent dans le cas où l'objet n'a pas réussi à recevoir la réponse du réseau. Dans un tel cas :

- Attendre d'autres essais
- Si l'antenne du réseau est à quelques mètres de l'objet, il faut éloigner l'objet (le signal est trop fort)
- Si l'antenne du réseau est loin, il faut peut-être placer l'objet à l'extérieur pour lui offrir une meilleure chance de recevoir la réponse du réseau.

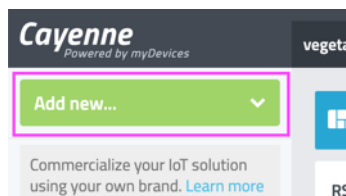
Suivent ensuite les communications UPLINK émises de façon régulières par l'objet pour communiquer les nouvelles valeurs d'hygrométrie du sol.

## Création d'un tableau de bord avec Cayenne / myDevices

Il nous reste une dernière étape pour que notre solution soit complète : créer un tableau de bord permettant de visualiser les données de notre capteur. Nous allons utiliser la solution Cayenne de myDevices qui permet de gratuitement constituer ce type de tableau de bord.

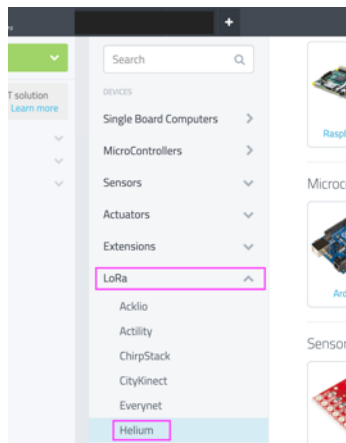
MyDevice possède une autre force : celle de décoder automatiquement le contenu des messages émis grâce à un format de données spécifique. C'est pour cette raison que dans notre programme, nous envoyons 4 octets pour une valeur de 16 bits signée. Le premier octet que nous envoyons indique un numéro de capteur, ici 1 puisqu'il n'y a qu'un seul capteur, celui d'hygrométrie. Ensuite le second octet de valeur 2 indique que le format est un entier signé sur 16 bits. Il y a différents types que Cayenne va reconnaître et standardiser : température, pression, position GPS ... De cette façon, le tableau de bord va pouvoir se constituer automatiquement avec les bonnes unités et les bonnes représentations.

Nous devons donc créer un compte sur <https://cayenne.mydevices.com/> et vous pourrez alors ajouter un nouvel objet :



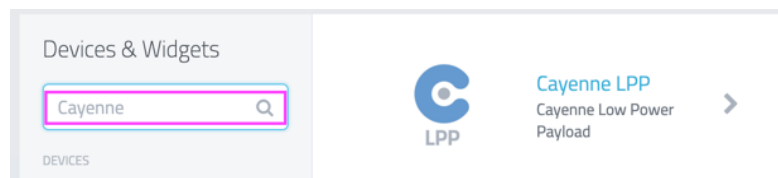
Nous allons choisir d'ajouter un nouvel objet. Pour cela, cliquer sur « Add new » puis choisir « Device ».

Ceci va ouvrir une nouvelle fenêtre dans la zone de droite.



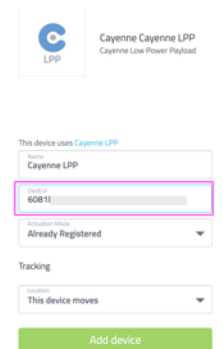
Dans cette partie de l'écran, reproduit ci-contre, il faudra sélectionner la technologie ici « LoRa » puis le réseau, ici « Helium ».

Ensuite, il faudra utiliser le champ de recherche pour trouver « Cayenne LPP ».

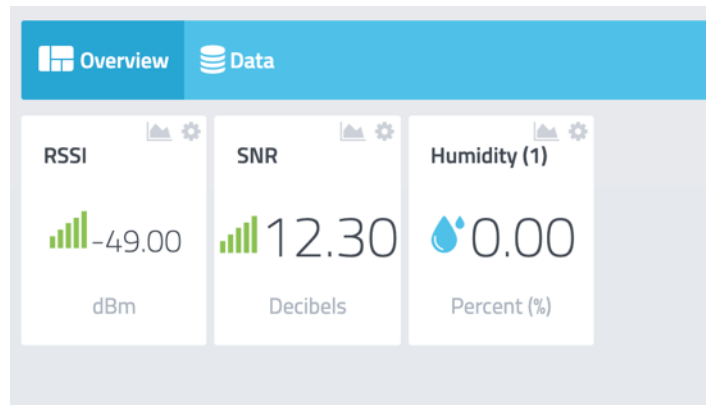


Une fois ceci fait, vous pourrez renseigner les informations identifiant l'objet. Il faut entrer dans le champ DEVEUI l'identifiant de l'objet que nous avons précédemment obtenu depuis la console et utilisé dans le programme.

Nous pouvons laisser les autres informations telle-quelle et valider la création de l'objet.



Nous pouvons maintenant visualiser les données dans le tableau de bord comme représenté ci-contre. Il sera possible de transformer celui-ci pour l'adapter à notre besoin : une gauge correspondant au niveau d'hygrométrie. Nous pourrions aussi placer une alerte par email en dessous d'un seuil.



## Finalisation



Une fois les tests terminés, il faudra fixer les composants pour rendre le système utilisable dans un pot de fleur ou une solution étanche s'il est utilisé en extérieur.

Vous pourrez aussi augmenter la période entre chaque communication, l'hygrométrie du sol évoluant à faible vitesse, 1 donnée chaque heure sera suffisante. La ligne `#define TXPERIOD` pourra donc être passée à la valeur (60\*60).

Vous pourrez aussi configurer avec Cayenne un « trigger » pour recevoir un email dès lors que le capteur retournera une valeur inférieure à une limite.

Pour aller un peu plus loin, vous pourrez piloter l'alimentation du capteur par un GPIO de sorte à augmenter l'autonomie de la solution.

Ces modifications faites, la consommation en veille sera inférieure à 50uWh environ ce qui devrait permettre une autonomie de plusieurs années avec les piles utilisées.