

COTI CAROLINE - BENALLAL AYOUB - BENKHIRA YASSINE - MEDFAI KALIL

WIRELESS COMMUNICATION TP4: ENVIRONMENTAL SENSOR



Table des matières

- **INTRODUCTION;**

- **APB**

- A. SITE WEB;

- B. PROGRAMME;

- **OTAA;**

- A. SITE WEB;

- B. PROGRAMME;

- C. CAYENNE

- **COMPARAISON;**

- **CONCLUSION;**

INTRODUCTION;

Le but du TP est de comparer les différentes procédures d'activation mise à disposition, dans ce tp nous utiliserons deux modes grâce au LoRaWan qui est le moyen de communication. (def de Lora)

-Le premier est « Activation by Personalization » (ABP)

-Le deuxième est « Over the Air Activation » (OTAA)

A l'aides des programmes Arduino donnés par les professeurs, nous allons pouvoir comparer les deux moyens, leurs avantages et inconvénients

Nous utiliserons aussi le site web <https://www.thethingsnetwork.org/>, ce site web est un réseau mondiale ouvert pour construire votre prochaine application IoT à faible coût, avec une sécurité maximale et une capacité d'évolution. The Things Network est un réseau communautaire fonctionnant avec la technologie LoRa conforme au standard LoRaWAN. Cette technologie permet à des objets de transmettre de petits volumes d'informations en consommant très peu d'énergie.



APB

A.SITE WEB

La première méthode d'activation du terminal est APB, « Activation By Personalisation », pour pouvoir l'utiliser, il faut utiliser le programme Arduino donne par le professeur (« UCA-ABP Basic.ino »)

Mais avant cela, nous devons configurer sur le site web

« <https://www.thethingsnetwork.org/> » les bons paramètres pour pouvoir utiliser APB correctement et qu'il soit relié avec le programme Arduino.

Il y a plusieurs paramètres à configurer : app device, application EUI, app KEY.

Ces paramètres sont essentiels pour pouvoir communiquer, nous verrons dans la suite du tp, nous verrons l'utilisation sur le programme Arduino.

ID d'application

Valrose

Reference de l'appareil

Ayoub

Méthode d'activation

ABP

Dispositif EUI

<>

⇅

21 CE 06 00 00 00 13

📄

Application EUI

<>

⇅

70 B3 D5 7E D0 01 33 E0

📄

Adresse du périphérique

<>

⇅

26 01 19 A5

📄

Clé de session réseau

<>

⇅

🔍

msb

←

0x00 0x07 0x0B 0x06 0x3E 0x0C 0x71 0x05 0x05 0xE4 0xE3 0x01 0xE

→

📄

Clé de session d'application

<>

⇅

🔍

msb

←

0xE0 0xE6 0xF2 0x11 0xE4 0xEE 0x10 0x11 0xE4 0xE9 0xE0 0x20 0x7

→

📄

Statut

● il y a 2 heures

Cadres en place

0

réinitialiser les compteurs d'images

Cadres bas

0

Figure 1 : Device Overview

B.PROGRAMME ;

Le programme est totalement liée a l'APB, pour pouvoir envoyer les packets, il faut utiliser les parametres obtenus sur le site, il faut ensuite inscrire les données de l'app device, application EUI, app KEY. Les 3 servents notamment à configurer pour qu'il n'y a aucun problème de connexion et par mesure de sécurité (pour pas qu'une autre personne se connecte ou reçois les paquet). On voit aussi le message que l'on va envoyer avec les paquets programme ci-dessous (« Hello, world »).

```
// This is the default Semtech key, which is used by the early prototype TTN
// network.
static const u1_t PROGMEM APPSKEY[16] = { 0xA1, 0xF0, 0x5E, 0x3D, 0xE1, 0x16, 0x90, 0x65, 0x8B, 0xB7, 0xD2, 0x2D, 0xD6, 0x9C, 0x8D, 0x0A } ;

// LoRaWAN end-device address (DevAddr)

static const u4_t DEVADDR = 0x260117D9;

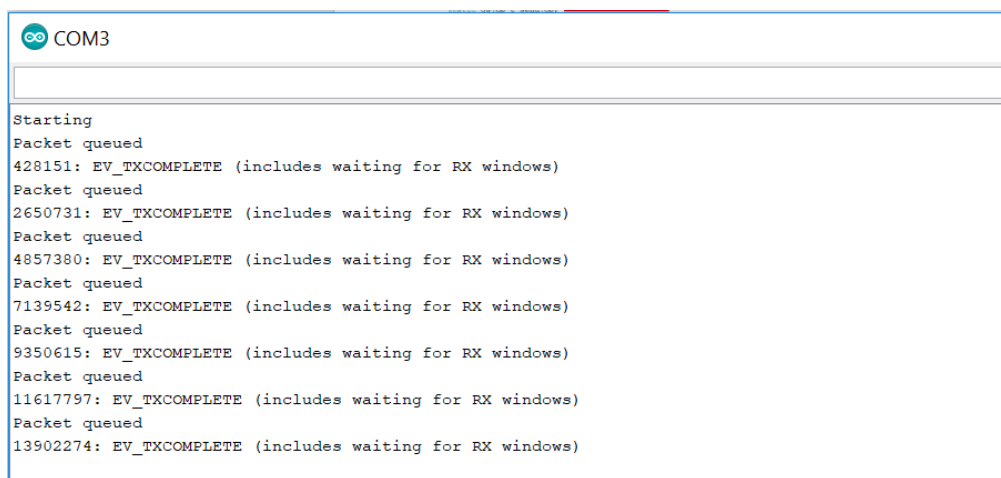
// These callbacks are only used in over-the-air activation, so they are
// left empty here (we cannot leave them out completely unless
// DISABLE_JOIN is set in config.h, otherwise the linker will complain).
void os_getArtEui (u1_t* buf) { }
void os_getDevEui (u1_t* buf) { }
void os_getDevKey (u1_t* buf) { }

static uint8_t mydata[] = "Hello, world!";
static osjob_t sendjob;

// Schedule TX every this many seconds (might become longer due to duty
// cycle limitations).
const unsigned TX_INTERVAL = 30;
```

Figure 2:Programme Arduino APB

On voit le résultat sur le serial monitor avec le nombre d'octet reçu dans la liaison descendante. On peut aussi voir les paquets qui vont être envoyé sur la photo ci-dessous.



```
Starting
Packet queued
428151: EV_TXCOMPLETE (includes waiting for RX windows)
Packet queued
2650731: EV_TXCOMPLETE (includes waiting for RX windows)
Packet queued
4857380: EV_TXCOMPLETE (includes waiting for RX windows)
Packet queued
7139542: EV_TXCOMPLETE (includes waiting for RX windows)
Packet queued
9350615: EV_TXCOMPLETE (includes waiting for RX windows)
Packet queued
11617797: EV_TXCOMPLETE (includes waiting for RX windows)
Packet queued
13902274: EV_TXCOMPLETE (includes waiting for RX windows)
```

Figure 3:Serial Monitor

Ensuite, on peut aller sur le site web <https://www.thethingsnetwork.org/> et voir si on a reçu les paquets, si on clique dessus on a plus de détail avec les coordonnées de l'expéditeurs.

| APPLICATION DATA | | | | | | | | | |
|---|----------|---------|------|-------|-----------------------|-------------------------------|---------------------------------|--|--|
| <div> <div> <div>uplink</div> <div>downlink</div> <div>activation</div> <div>ack</div> <div>error</div> </div> </div> | | | | | | | | | |
| Filters | time | counter | port | | | | | | |
| ▲ | 14:57:10 | 0 | 1 | retry | payload: 02 02 01 52 | analog_in_2: 3.38 | | | |
| ▲ | 14:57:09 | 0 | 1 | retry | payload: 02 02 01 52 | analog_in_2: 3.38 | | | |
| ⚡ | 14:57:14 | | | | dev addr: 26 01 27 23 | appeui: 70 B3D5 7E D001 33 E0 | deveui: 21 CE 06 00 00 00 00 13 | | |
| ⚡ | 14:56:08 | | | | dev addr: 26 01 29 B7 | appeui: 70 B3D5 7E D001 33 E0 | deveui: 21 CE 06 00 00 00 00 13 | | |

Figure 4:data APB

• OTAA;

A.SITE WEB

La deuxième methode d'activation du terminal est l'OTAA, "Over the Air Activation", on utilisera aussi un programme Arduino("OTAA_Basic.ino ») pour pouvoir envoyer a partir du programme et le recevoir sur le site web.

Avant de commencer la procédure de jointure, il faut avoir ces paramètres essentiels

- un DevEUI: qui est une adresse
- AppEUI: l'adresse de destination
- AppKey: clé d'authentification

14:57:14
dev addr: 26 01 27 23
app eui: 70 B3 D5 7E D0 01 33 E0
dev eui: 21 CE 06 00 00 00 00 13

Activation

Device Address

26 01 27 23

Device EUI

21 CE 06 00 00 00 00 13

App EUI

70 B3 D5 7E D0 01 33 E0

Metadata

```

{
  "time": "2019-03-01T13:57:04.843998574Z",
  "frequency": 868.3,
  "modulation": "LORA",
  "data_rate": "SF7BW125",
  "coding_rate": "4/5",
  "gateways": [
    {
      "gtw_id": "eui-c0ee40ffff29456c",
      "timestamp": 373846371,
      "time": "",
      "channel": 1,
      "rssi": -55,
      "snr": 10.5,
      "rf_chain": 1
    }
  ]
}

```

Figure 5:DATA

B. PROGRAMME;

Comme pour le ABP, il faut aussi utiliser les parametres obtenu sur le site et les inserer sur le programme Arduino afin que le site recoit les packets.

On constate sur le programme les parametres qu'il fallait inserer dans le programme afin qu'on puisse le recevoir sur le data. Les données a metre sur le programme doivent être en hexadecimal.


```

static const ul_t PROGMEM APPEUI[8] = { 0xE0, 0x33, 0x01, 0xD0, 0x7E, 0xD5, 0xB3, 0x70 };
void os_getArtEui (ul_t* buf) {
    memcpy_P(buf, APPEUI, 8);
}

// This should also be in little endian format, see above.
static const ul_t PROGMEM DEVEUI[8] = { 0x13, 0x00, 0x00, 0x00, 0x00, 0x06, 0xCE, 0x21 };
void os_getDevEui (ul_t* buf) {
    memcpy_P(buf, DEVEUI, 8);
}

// This key should be in big endian format (or, since it is not really a
// number but a block of memory, endianness does not really apply). In
// practice, a key taken from ttncnl can be copied as-is.
// The key shown here is the semtech default key.
static const ul_t PROGMEM APPKEY[16] = { 0xEF, 0x0D, 0x25, 0x04, 0x3D, 0xED, 0x39, 0x62, 0x9C, 0x3A, 0x46, 0x64, 0xFC, 0xFE, 0x75, 0x83 };
void os_getDevKey (ul_t* buf) {
    memcpy_P(buf, APPKEY, 16);
}

static osjob_t sendjob;

```

Figure 6:Programme Arduino OTAA

On voit bien donc sur le serial monitor après avoir vérifié et téléverser, on voit les paquets sont en train d'être envoyé, on voit aussi une première liaison montante et elle demande une connexion, en suite un deuxième paquet avec les paramètres des donnés, on aperçoit sur le moniteur en série encore plein de donnée et pour termine l'appareil s'endort après avoir joint le Tx.

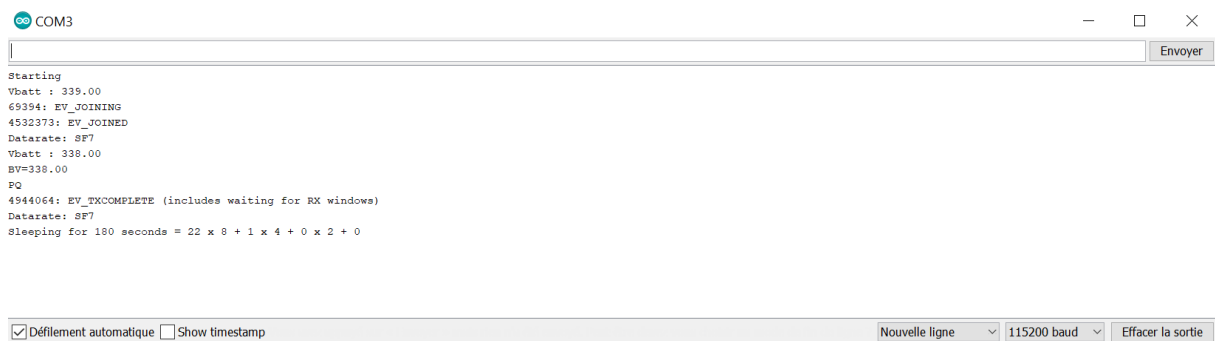


Figure 7:Serial Monitor OTAA

Donc après avoir lancer le moniteur en série, les résultats s'affichent sur le site web dans le data, on voit sur la photo ci- dessous les deux message reçus (avec le petit triangle bleu), on voit le payload et quand on clique sur un des messages, on aperçoit plus en détail avec les paramètres d'activation.

| APPLICATION DATA | | | | | | | | | |
|--|---------|------|-------|-----------|-------------|--------------|-----------------------|---------|-------------------------|
| <div> <div> <div>uplink</div> <div>downlink</div> <div>activation</div> <div>ack</div> <div>error</div> </div> <div>Filters</div> </div> | | | | | | | | | |
| time | counter | port | | | | | | | |
| ▲ 14:57:10 | 0 | 1 | retry | payload: | 02 02 01 52 | analog_in_2: | 3.38 | | |
| ▲ 14:57:09 | 0 | 1 | retry | payload: | 02 02 01 52 | analog_in_2: | 3.38 | | |
| ⚡ 14:57:14 | | | | dev addr: | 26 01 27 23 | appeui: | 70 B3D5 7E D001 33 E0 | deveui: | 21 CE 06 00 00 00 00 13 |
| ⚡ 14:56:08 | | | | dev addr: | 26 01 29 B7 | appeui: | 70 B3D5 7E D001 33 E0 | deveui: | 21 CE 06 00 00 00 00 13 |

Figure 8:Data OTAA

C.CAYENNE

L'OTAA a un autre système permettant de recevoir les données en Lorawan et grâce notamment au programme Arduino. Pour l'utiliser il faut aller sur le site <https://mydevices.com/> et se créer un compte et en suite. Il faut ensuite bien paramétrer en utilisant le bon device. Mais pour le fonctionnement du Cayenne, il faut utiliser quand même TheThingNetwork et il faut aussi au moins envoyer une fois un paquet pour que Cayenne puisse aussi recevoir.

On voit donc sur l'image dessous les résultats, il affiche ce qu'on lui a envoyé grâce au programme Arduino (« LP_SI7021_ TEMT6000.ino ») qui mesure la température, l'humidité et la luminosité.

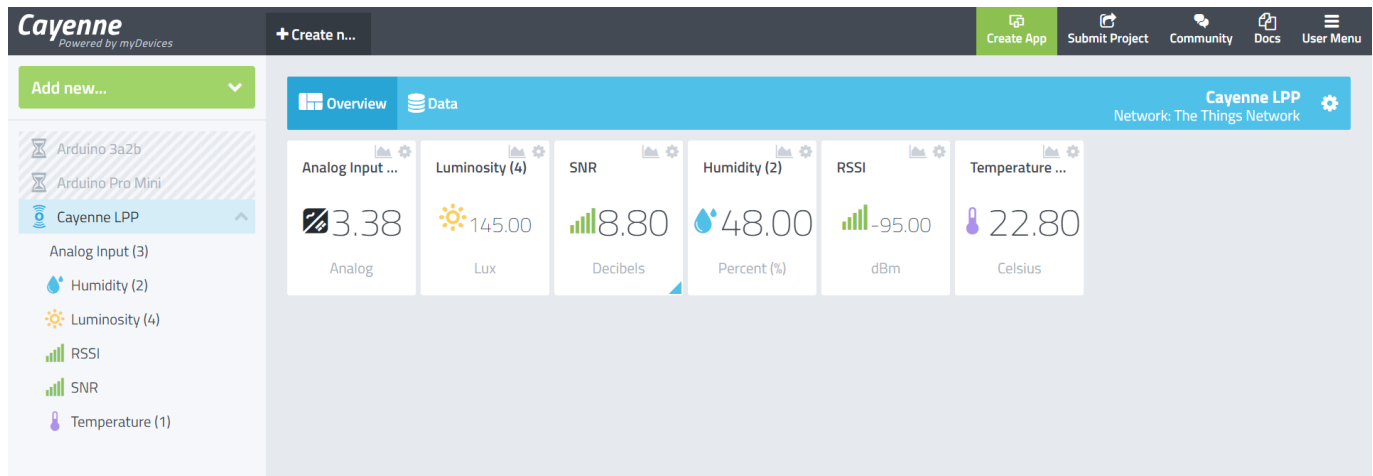


Figure 9: Cayenne

• COMPARAISON;

Pour terminer on va comparer les deux méthodes d'activation du Terminal, d'un côté OTAA, l'équipement doit transmettre au réseau une demande d'accès : join request. Pour ce faire, celui-ci doit être en possession de trois paramètres :

- DevEUI, identifiant unique (de type EUI64) de l'équipement (fourni par l'équipementier).
- AppEUI, identifiant du fournisseur de l'application (EUI 64).
- AppKey, clé AES 128 déterminée par le fournisseur de l'application.

OTAA lui peut être plus appréciée car on peut l'utiliser de partout d'où son nom over the air, un peu comme un « cloud ».

De l'autre côté APB, les clés de session NwkSKey et AppSKey ainsi que l'adresse de l'équipement (DevAddr) sont directement inscrits dans l'équipement LoRaWAN, lui par contre n'a pas à envoyer de requêtes pas comme OTAA, cette méthode implique que les équipements communiquent avec un réseau spécifique car les clés de session

sont déjà connues. Et contrairement à la méthode OTAA, les clés de session sont statiques

L'inconvénient de ABP est qu'il utilise des clés de session qui doivent être renouvelées à chaque fois à l'inverse de OTAA de plus il paraît donc beaucoup plus sécurisé que APB car il est un peu plus dur à paramétrer, OTAA est un peu un cloud donc il est nécessaire de l'utiliser à plus grande échelle alors que APB est largement suffisant pour un déploiement normal.

La méthode OTAA est donc plus sécurisée et conseillée pour des déploiements à grande échelle.



● CONCLUSION;

En conclusion, pendant ce TP nous avons étudié deux méthodes d'activation qui ont leurs contraintes et avantages. OTAA est plus sécurisée mais est plus utile pour les grandes échelles tandis que APB est conseillé pour un déploiement normal.

On a pu voir notamment grâce aux programmes Arduino l'importance d'entrer les bons paramètres (Appkey, devices...) afin de recevoir les paquets dans le data.

Donc, nous recommandons OTAA qui est plus sécurisé que APB, qui est utilisé comme un cloud et n'a pas besoin d'un renouvellement de clé à chaque utilisation à l'instar de APB.

