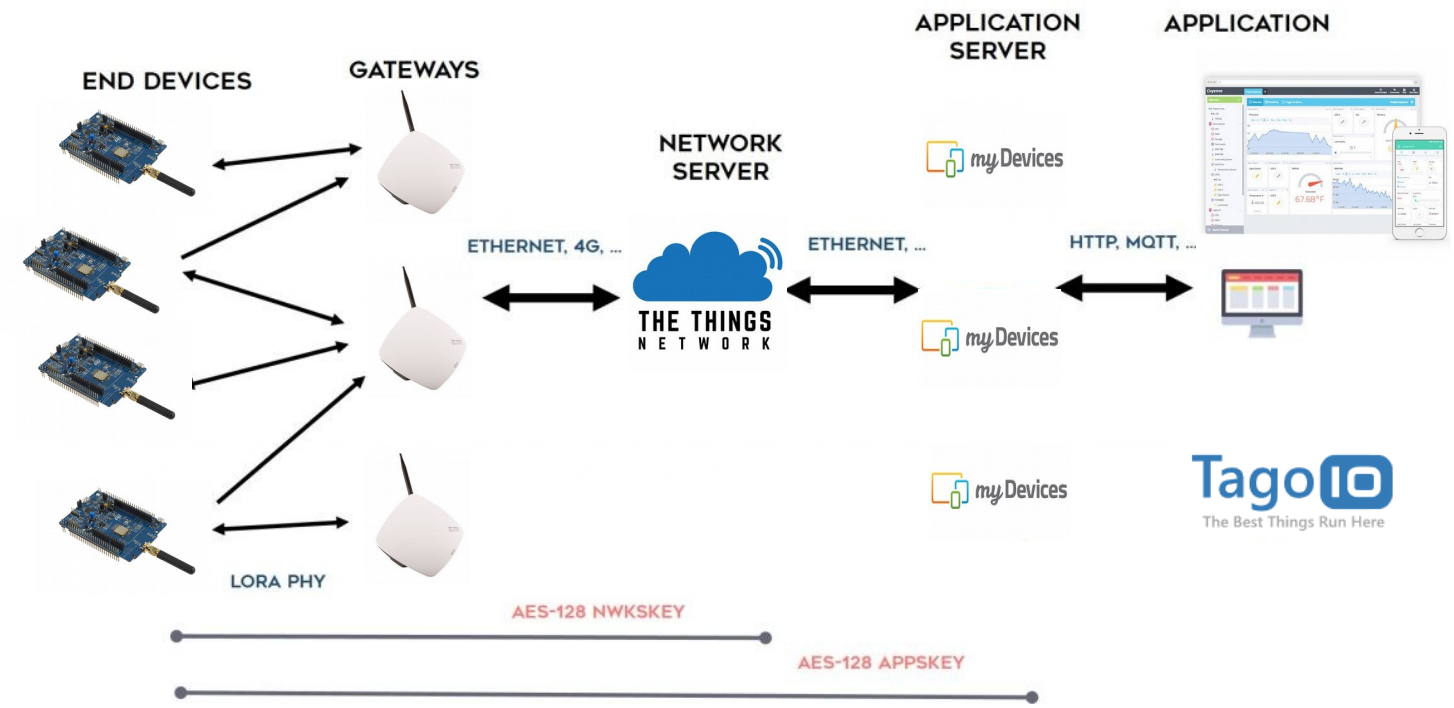
Réalisation d'un objet connecté à une application en ligne avec LoRaWan



Objectifs : Réaliser un objet connecté à une application en ligne par protocole LoRaWan
- Configurer le node B-LO72Z-LRWAN
- Configurer le broker TTN
- Configurer l'application finale mydevices

Durée conseillée : 4h

Le node sera réalisé avec une carte B-L072Z-LRWAN1 fabriquée par STMicroelectronics. Celle-ci est équipée d'un module contenant un microcontrôleur STM32, un émetteur/récepteur LoRa (ou Sigfox), un support de batteries et une interface ST-LINK pour la programmation. La carte B-L072Z-LRWAN1 ne dispose d'aucun capteur, elle peut accueillir la plupart des shields au format Arduino.

Le logiciel embarqué doit être configuré pour la fonction finale du node :

- Intégration au programme de l'identifiant du node et des différentes clés LORAWAN
- Acquisition des valeurs des capteurs
- Encapsulation des données (payload) au format souhaité, ici « cayenne »

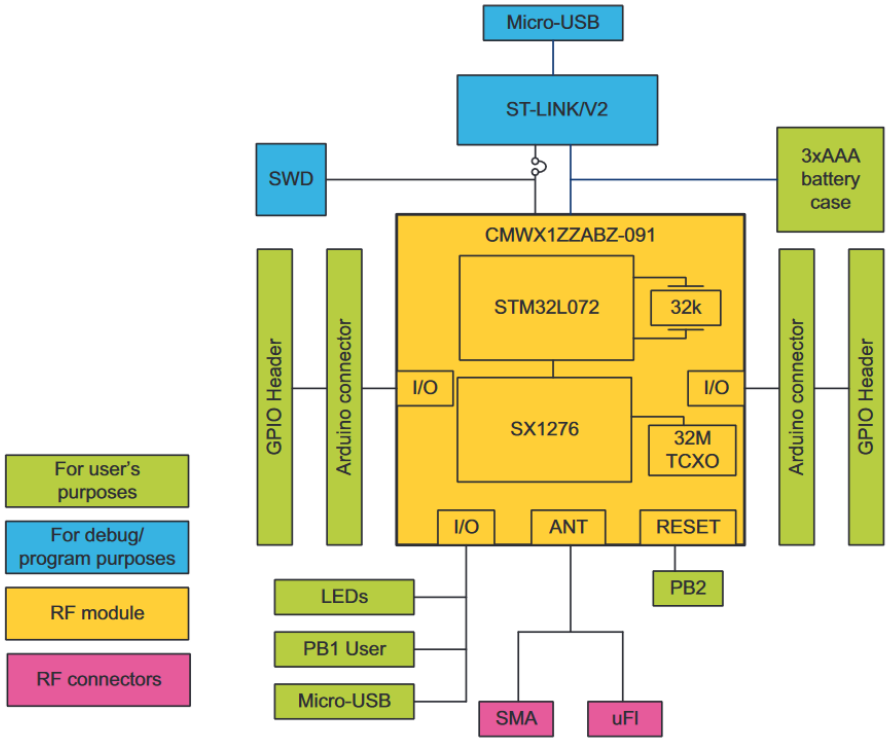
La carte sera programmée en C++ à l'aide de d'environnement de développement intégré en ligne MBEB
Il est indispensable d'être familiarisé avec l'IDE MBED.

Découvrir MBED : https://os.mbed.com/docs/mbed-os/v5.11/quick-start/online-with-the-online-compiler.html#importing-the-code

Documentation B-L072Z-LRWAN1 : https://www.st.com/en/evaluation-tools/b-l072z-lrwan1.html
A lire en particulier :

UM2115: Discovery kit for LoRaWAN™, Sigfox™, and LPWAN protocols with STM32L0 :
https://www.st.com/resource/en/user_manual/dm00329995.pdf

1 Synoptique de la carte B-L072Z-LRWAN1



.

La carte comporte un composant hybride intégrant un microcontrôleur STM32 et un émetteur récepteur LORA.
Elle dispose d'une interface de programmation/debug ST-LINK, de connecteurs ST Morpho et de connecteurs type Arduino qui lui permettent d'accueillir la plupart des shields Arduino.

A partir de la documentation ST et de toutes autres documentations utiles :
https://os.mbed.com/platforms/ST-Discovery-LRWAN1/

Énumérer les éléments constituants le CMWX1ZZABZ-091 et indiquer leur fonction ?
Repérer ce circuit sur la carte

- Combien de GPIO possède cette carte ?
- Quelle différence existe t 'il entre les connecteurs Arduino et GPIO Header ?
- Quel microcontrôleur STM32 est embarqué ?
- Quelle est sa fréquence de fonctionnement ?
- Quelle est sa capacité de ROM
- Quelle est sa capacité de RAM
- Qu'appelle t on ADC/DAC/SPI/I2C/UART/USB ?

Repérer sur la documentation les connecteurs Arduino et les fonctions qui leurs sont associées (GPIO, I2C …)
Combien de LEDs sont à la disposition de l'utilisateur ?
Quels sont les rôles des boutons poussoirs ?

Que signifie 3xAAA Battery case ?

A quoi sert la fonction ST-LINK V2 ?

2 Travaux pratiques

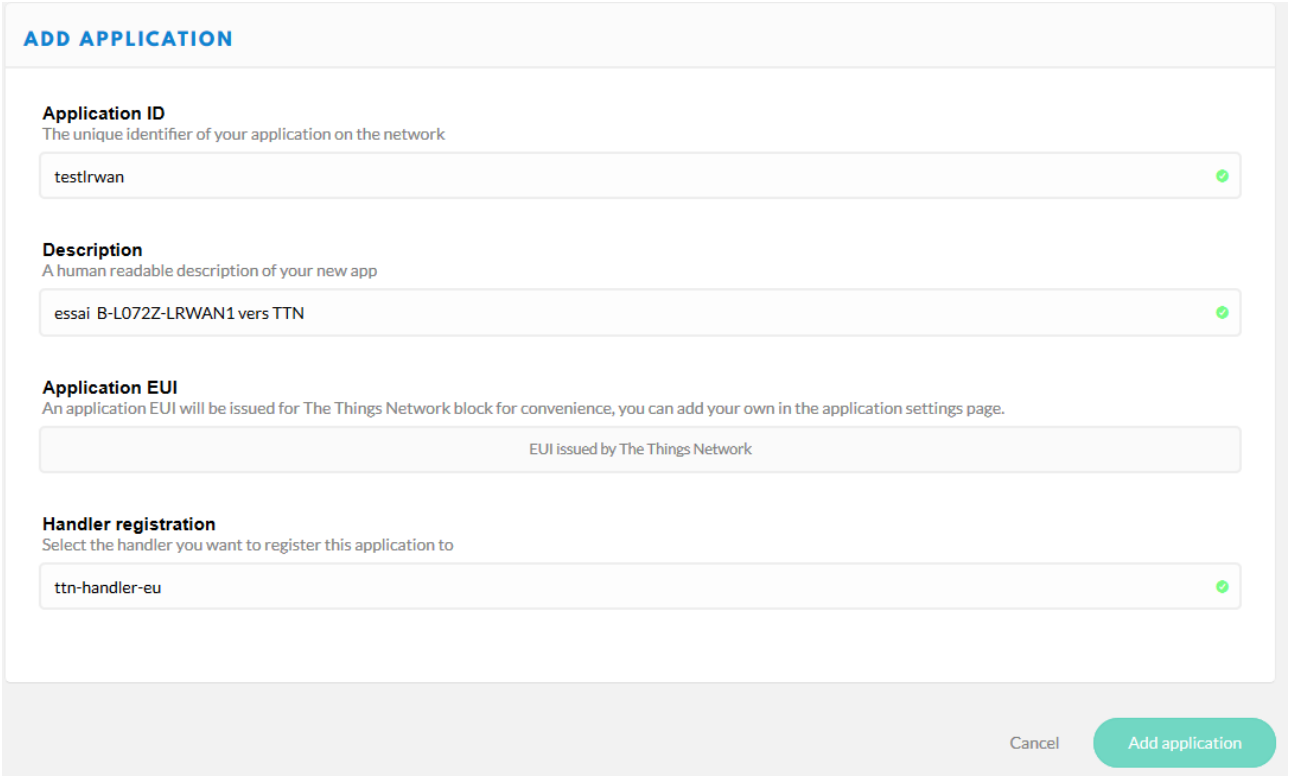
Le TP se compose de trois parties :
- Création et configuration d'un compte TTN sur https://www.thethingsnetwork.org/ et récupération des clés de configuration du node.
- Adaptation du logiciel de démonstration sur MBED avec configuration des capteurs, du payload et des clés LoRaWan récupérées sur TTN
- Construction de l'application finale sur mydevices.com

Le TP permettra l'acquisition de la température ambiante du node et sa lecture sur une application en ligne.

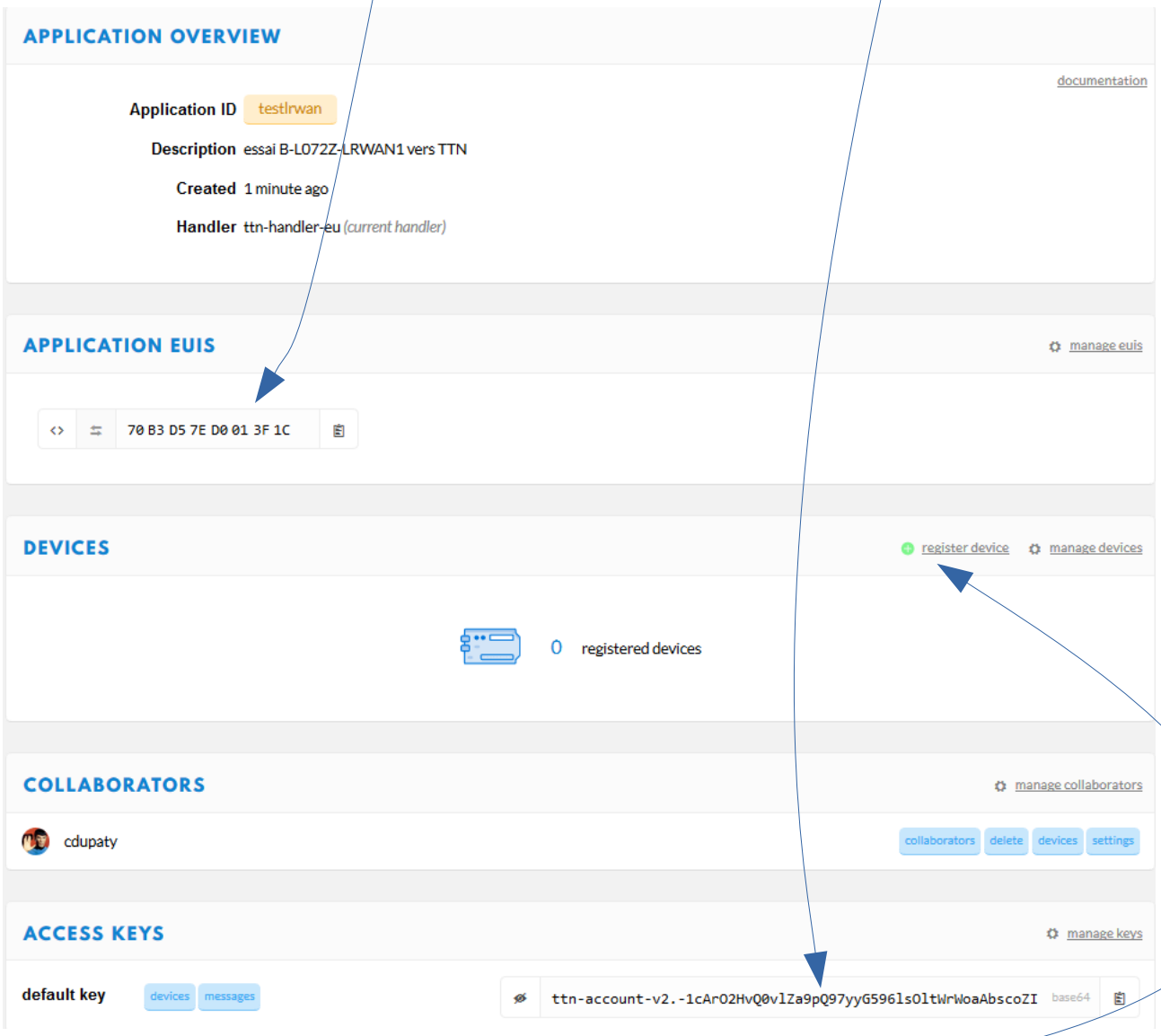
2.1 Broker TTN

Créer un compte sur https://www.thethingsnetwork.org/

Puis sur TTN créer une nouvelle application



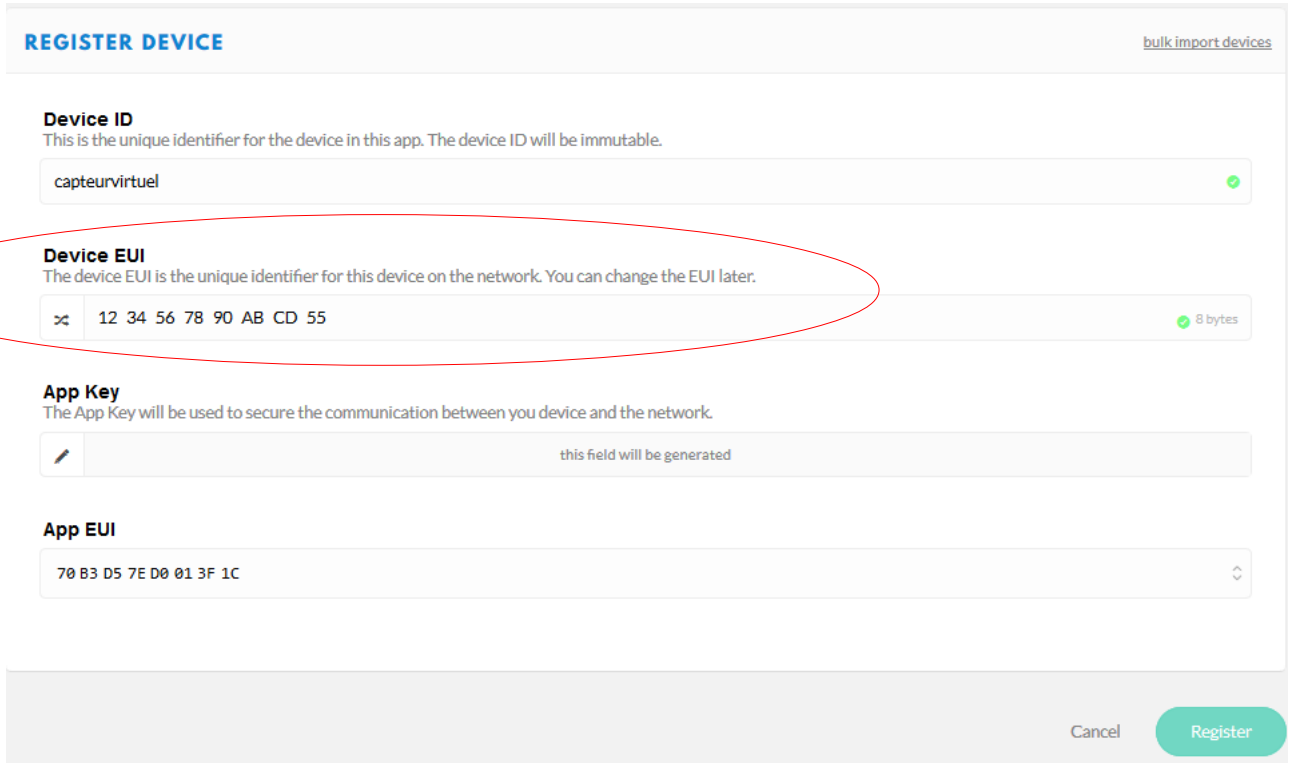
TTN procure alors l'**application EUIS qui sera enregistrée dans le node** et la **clé d'accès qui permettra de récupérer les données depuis l'application terminale**



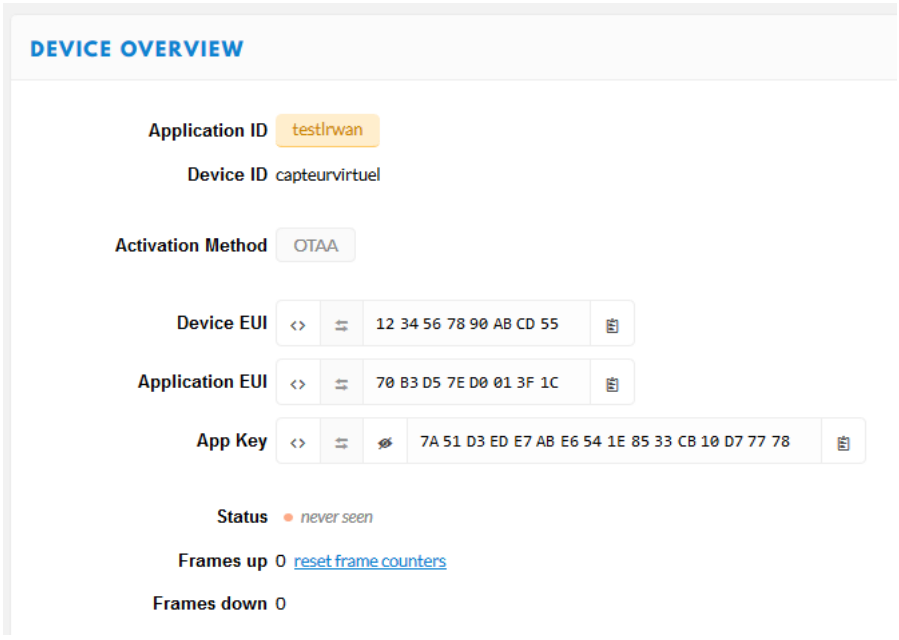
Une application peut contenir plusieurs « devices »

Enregistrer un nouveau « device » (register device)

Choisir un nom et un **device EUI de votre choix mais qui doit être unique**

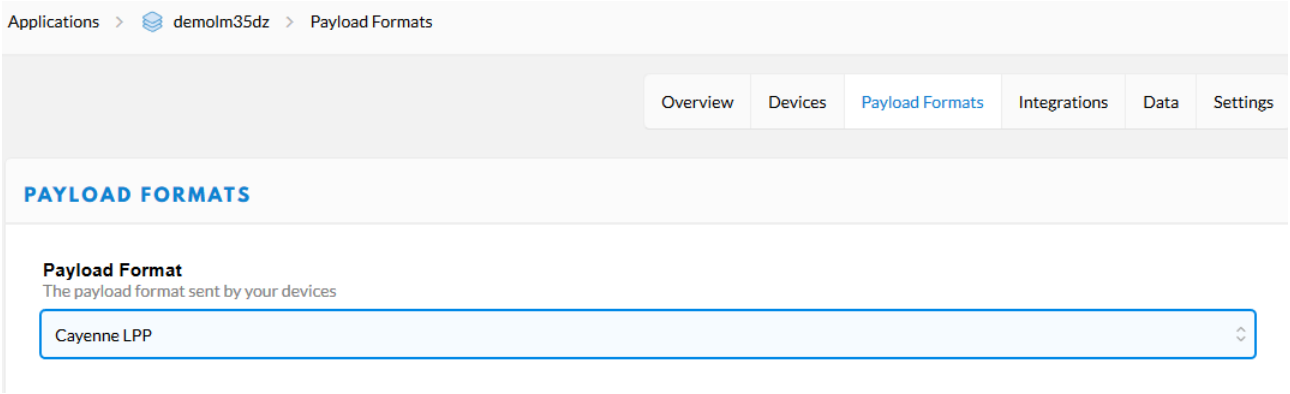


TTN génère l'application EUI ainsi que la clé d'application et affiche un récapitulatif



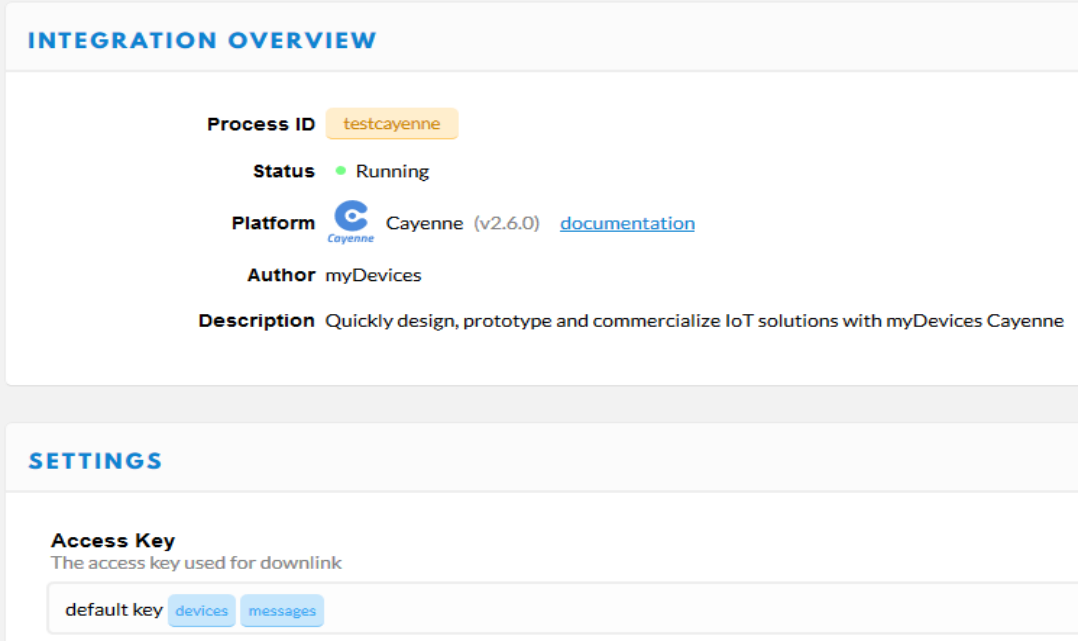
Les données provenant du node sont encapsulées au format « cayenne ». TTN doit être préparé pour ce format de données.

Indiquer le format des données entrantes dans « payload format »



Aller dans applications, sélectionner l'application crée.

Ajouter une « integration », sélectionner « cayenne » , nommer le process ID et valider une clé d'accès, puis cliquer sur ADD INTEGRATION



2.2 Programmation du NODE

Sur MBED effectuer une recherche sur LORA-mbed-os-example-lorawan

Importer le programme dans le compilateur et mettre à jour toutes les bibliothèques (vous changerez le nom du programme en **LRWAN-TTN-CAYENNE)**

La démonstration propose l'émission de données d'un capteur de température virtuel DS1820.

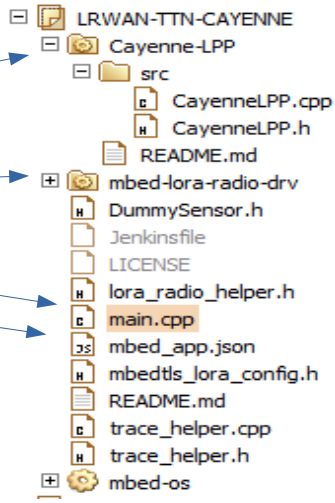
L'application finale étant mydevices cayenne, il est nécessaire d'ajouter la bibliothèque cayenne au projet.

Effectuer une recherche sur « Cayenne Low Power Payload » pour récupérer la bibliothèque

Importer la bibliothèque dans le projet et mettre à jour les bibliothèques. **(sélectionner LRWAN-TTN-CAYENNE dans target path)**

Le projet doit ressembler à ceci avec en particulier :

La bibliothèque Cayenne
La bibliothèque LoRa radio SX1276
Le programme principal
Le fichier json de configuration



Le logiciel propose par défaut de transmettre les données d'un capteur virtuel (DummySensor). Dans une application réelle il faudra retirer ce capteur et ajouter les classes/fonctions des capteurs réels.

Editer mbed_app.json

Entrer les codes TTN générés précédemment dans
"lora.device-eui", "lora.application-eui" "lora.application-key".

Il est possible pour les essais de choisir le rapport-cyclique d'émission LoRa, pour cela il faut passer "lora.duty-cycle-on" en « false » afin de pouvoir définir la période d'émission.

Dans ce TP laisser "lora.duty-cycle-on" en « true »

Editer main.cpp

Ajouter la bibliothèque cayenne : #include "CayenneLPP.h"

Si "lora.duty-cycle-on" est « false » il faut régler la période d'émission comme souhaité : #define TX_TIMER 20000 // 20sec

A l'emplacement des instanciations d'objets , instancier un objet de type « cayenne » qui permettra de formater l'envoi des données (payload) au format cayenne :

```
DS1820  ds1820(PC_9);
CayenneLPP cayenne(100);        // 100 caractères max
```

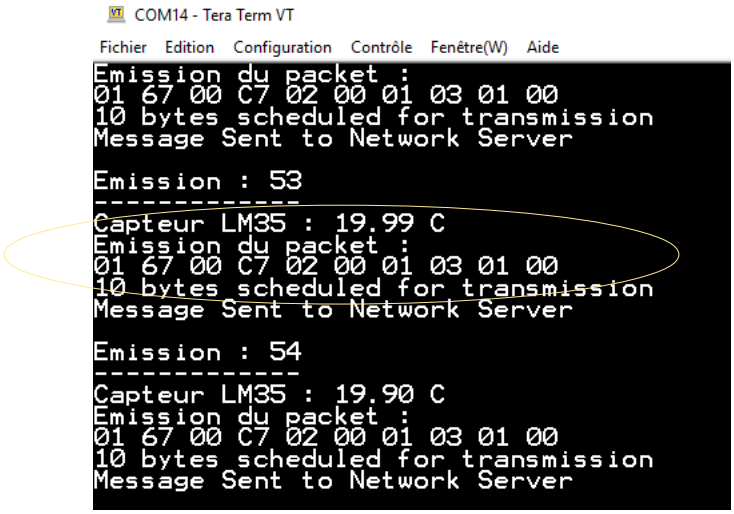
La fonction d'émission send_message doit être modifiée pour récupérer les données des capteurs et les encapsuler au format cayenne

Modifier cette fonction comme suit (ici on utilise le capteur virtuel de la démonstration) :

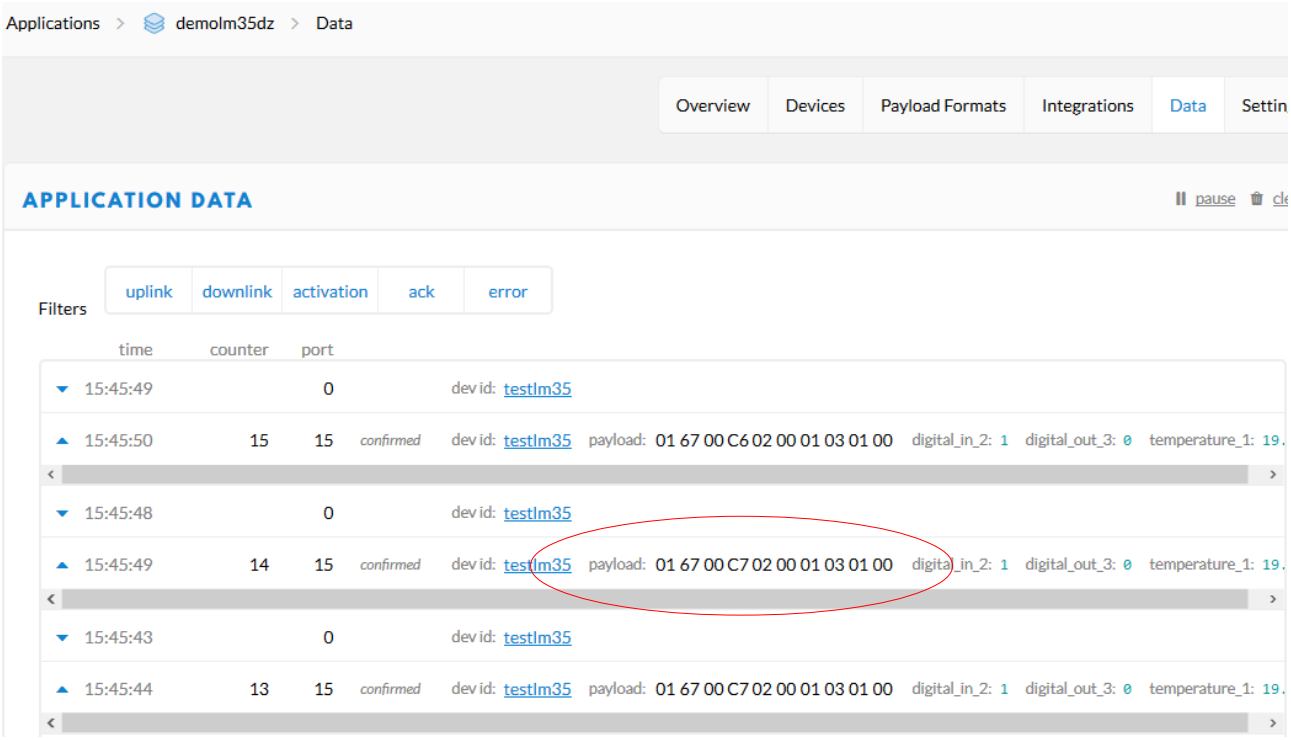
```
// Sends a message to the Network Server
static void send_message()
{
    uint16_t packet_len;
    int16_t retcode;
    float sensor_value;   // valeur de la temperature à transmettre

/* lecture du capteur virtuel ds1820 qui retourne ici une valeur aléatoire
 dans sensor_value */
    if (ds1820.begin()) {
        ds1820.startConversion();
        sensor_value = ds1820.read();
        printf("\r\n Dummy Sensor Value = %3.1f \r\n", sensor_value);
        ds1820.startConversion();
    } else {
        printf("\r\n No sensor found \r\n");
        return;
    }

    // mise en forme format CAYENNE LPP
    // doc ici https://mydevices.com/cayenne/docs/lora/#lora-cayenne-low-power-payload
    cayenne.reset();                        // vidage du payload cayenne
    cayenne.addTemperature(1, sensor_value);   // ajout de la valeur t°
    cayenne.copy(tx_buffer);                // copie dans le buffer LORA
    packet_len=cayenne.getSize();           // calcul de la longueur des données

    retcode = lorawan.send(MBED_CONF_LORA_APP_PORT, tx_buffer, packet_len, MSG_CONFIRMED_FLAG);  // emission

// Pour debug : affichage du message émis en hexa sur le terminal
      printf("\nMessage: ");
      for(int i=0;i<packet_len;i++) printf("%02X ",tx_buffer[i]);
      printf("\n");
```

La compilation peut générer des « warnings » **mais aucune erreur**
Transférer le fichier .bin obtenu dans la carte.

Les données transmises sont visualisables sur la console



Sur le site TTN les données reçues sont visualisables dans l'onglet data de l'application :

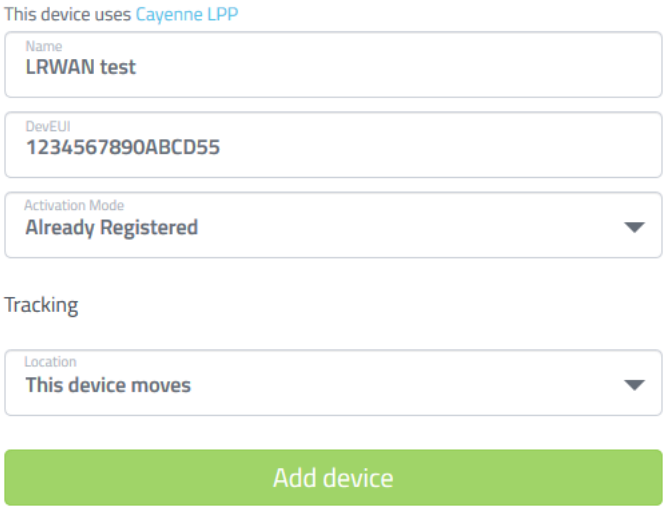


2.3 Configuration de l'application mydevices.com

Créer un compte sur https://mydevices.com/

Create App

Add new Device/Widget – LoRa – The Things Network
Selectionner STM32 B-L072Z-LRWAN1



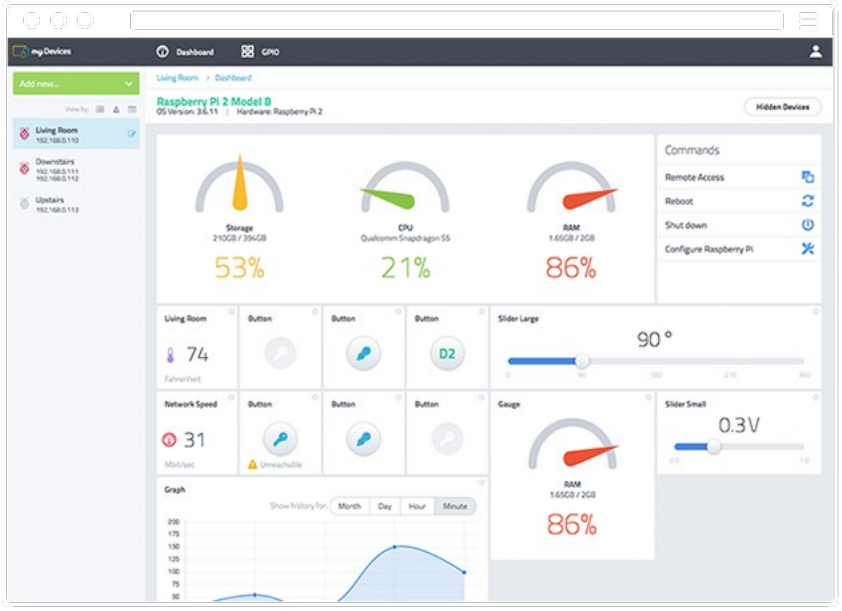
Nommer le device cayenne
Entrer le DevEUI du node.
Entrer la localisation du node

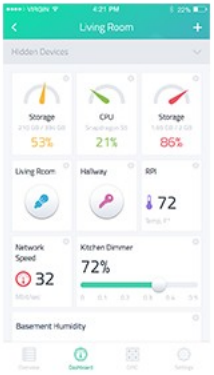
Add device.

L'écran **restera vide** jusqu'à l'arrivée des premières données.

Consulter ensuite la documentation pour intégrer le device à un projet cayenne.
Afficher les données sous forme numérique et sous forme graphique, explorer les possibilités de l'interface mydevices

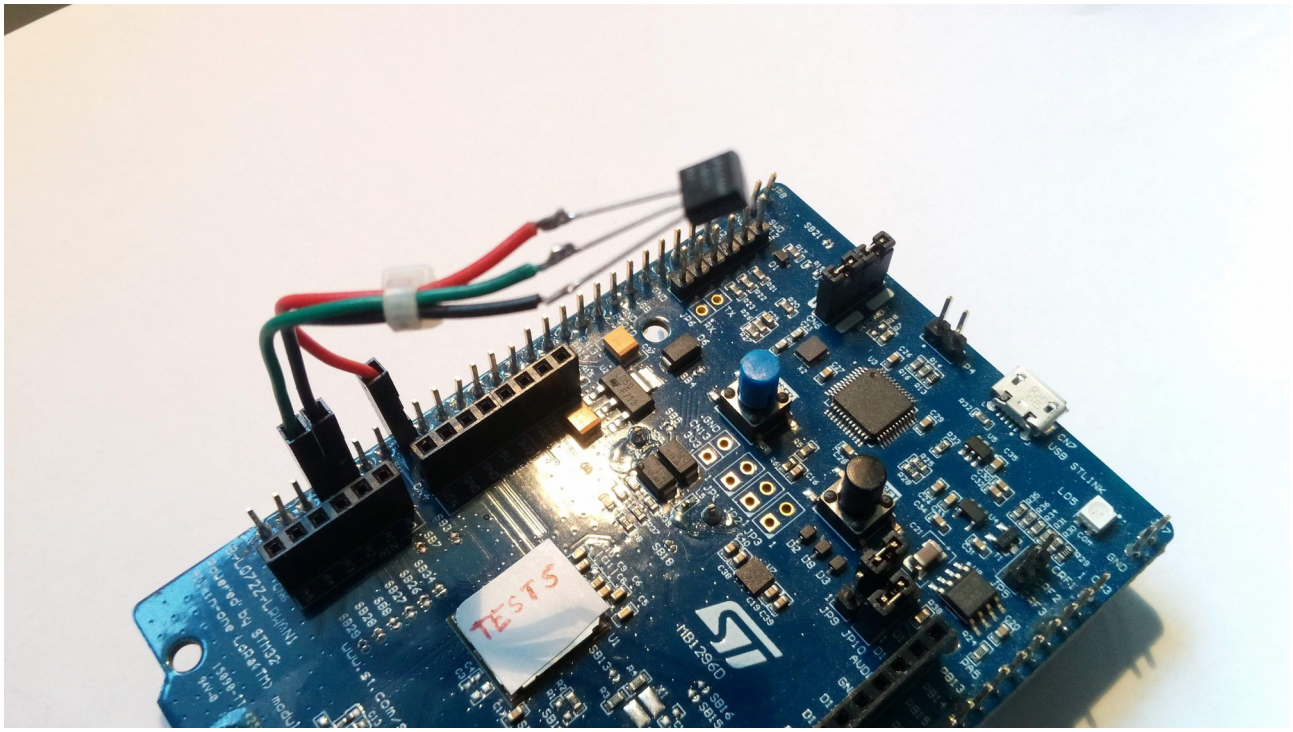
Voici un exemple d'interface mydevices

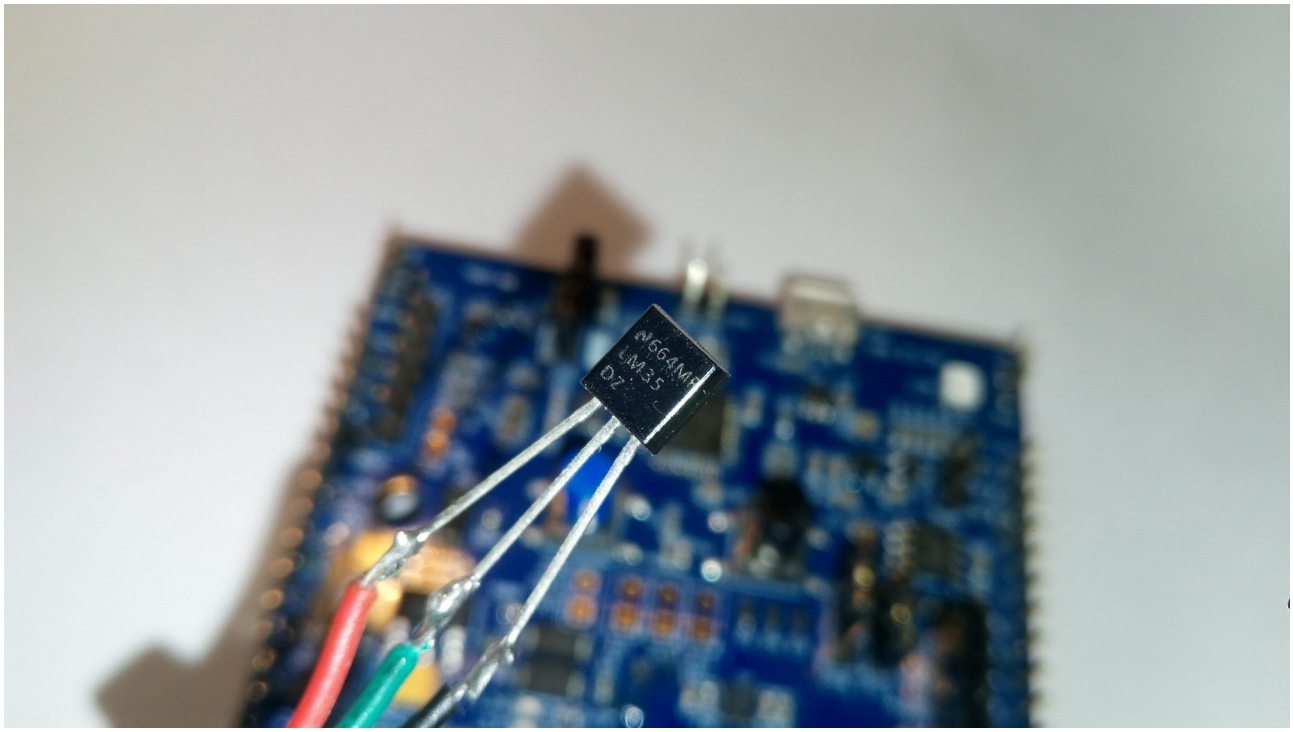




2.4 Exercice avec un capteur de température LM35DZ

Le capteur de température est de type LM35
Le capteur sera alimenté entre 3.3v et 0v sa sortie sera connectée à l'entrée PA0.
Documentation LM35 : http://www.ti.com/lit/ds/symlink/lm35.pdf
Farnell: https://fr.farnell.com/texas-instruments/lm35dz-nopb/capteur-de-temp-0-4-c/dp/1469236?st=lm35





Sur MBED, dans main.cpp, à la suite de l'instanciation de l'objet « cayenne » , instanciez un objet AnalogIn:

```
AnalogIn lm35(A0);                  // capteur LM35 sur A0
```

puis modifier la fonction send_message comme suit

```
/**
 * Sends a message to the Network Server
 */
static void send_message()
{
    uint16_t packet_len;
    int16_t retcode;
    float lm35temp;
/*
    placer ici l'acquisition de données
*/

 // Vs=0v+10mV/°C VDD=3.3v
    lm35temp=lm35.read()*330.0;


    // mise en forme format CAYENNE LPP
    // doc ici https://mydevices.com/cayenne/docs/lora/#lora-cayenne-low-power-
payload

    cayenne.reset();
    cayenne.addTemperature(8, lm35temp);  // objet graphique 8 (par exemple)

    cayenne.copy(tx_buffer);
    packet_len=cayenne.getSize();

    retcode = lorawan.send(MBED_CONF_LORA_APP_PORT, tx_buffer, packet_len,
MSG_CONFIRMED_FLAG);

 // affichage du payload sur le terminal pour test
    printf("\n lm35 : %.2f\n",lm35temp);
    printf("\nMessage: ");
    for(int i=0;i<packet_len;i++) printf("%02X ",tx_buffer[i]);
    printf("\n");
```

…

Compiler / transférer sans erreur.

Compléter maintenant l'application mydevices.com afin d'afficher la température.

Complément :

- Allumage d'une LED sur le node depuis l'interface mydevice.com
- lecture d'un bouton du node

Ajouter un objet DigitalOut et DigitalIn:

```
DigitalOut ledTest(LED1);
DigitalIn bp(USER_BUTTON);
```

Ajouter simplement dans le payload l'état du bouton

```
cayenne.addDigitalInput(2, bp.read());  // port 2 par exemple
```

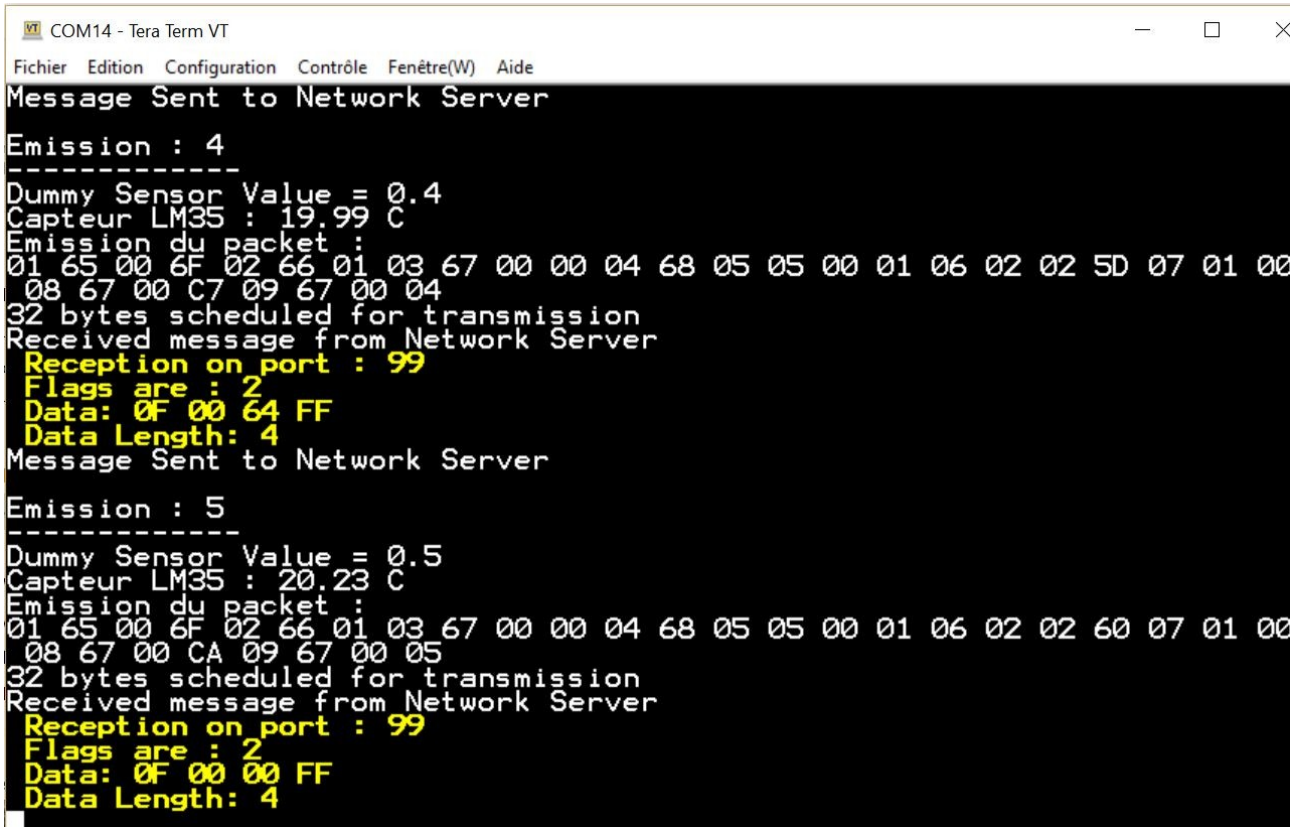
Pour déclarer le flux descendant digital au format « cayenne » :
Ajouter dans la partie construction du payload de la fonction send_message :

```
cayenne.addDigitalOutput(3,0); // port 3 , 0 par defaut
```

Pour allumer la led, modifier la fonction receive_message() comme suit :

```
/**
 * Receive a message from the Network Server
 */
static void receive_message()
{
    int16_t retcode;
    uint8_t port; // var to store port number provided by the stack
    int flags; // var to store flags provided by the stack
    retcode = lorawan.receive( rx_buffer,sizeof(rx_buffer), port, flags);
    printf("\x1B[1m");  // yellow text
    if (retcode < 0) {
        printf("receive() - Error code %d \r\n", retcode);
        return;
    }
    printf(" Reception on port : %d \n",port);
    printf(" Flags are : %d \n",flags);
    printf(" Data: ");

    for (uint8_t i = 0; i < retcode; i++) {
        printf("%02X ", rx_buffer[i]);
    }

    printf("\n\r Data Length: %d\r\n", retcode);
    printf("\x1B[0m");  // white text
    printf("End reception\n\r");

    if (rx_buffer[2]==0x64) ledTest=1;
    if (rx_buffer[2]==0x00) ledTest=0;

    memset(rx_buffer, 0, sizeof(rx_buffer));
}
```

Dès la réception des données, l'application mydevices.com fait apparaître un bouton Digital Output (7).
Lors du clic sur ce bouton la LED s'allume sur la carte ST.



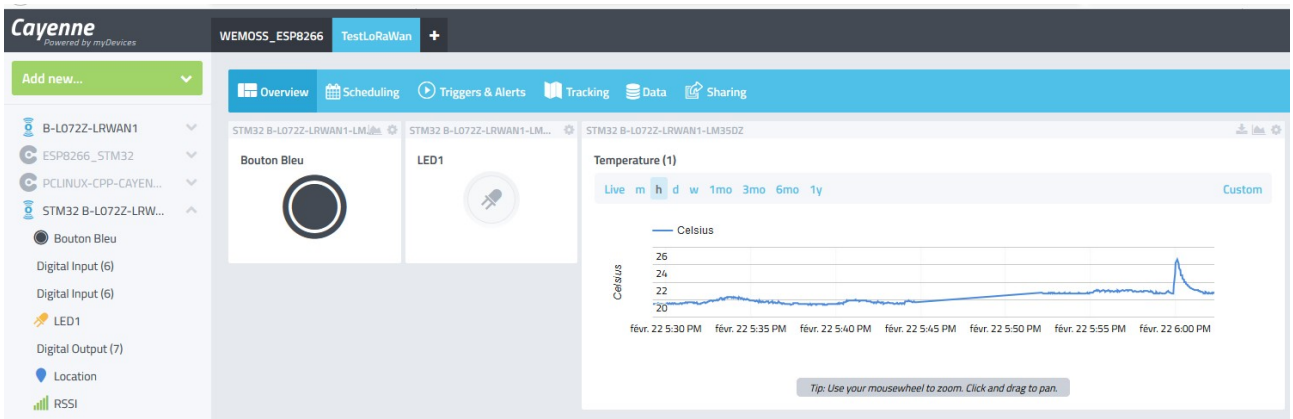
2.5 Interface utilisateur

La dernière partie est consacrée à la réalisation de l'interface utilisateur mydevices.com

Add→ New→ Project

Donner un nom au projet. Ex TestLoRaWan

L'espace des widgets est vide.
En cliquant – tirant placer les icônes souhaitées sur l'espace du projet.
Configurer les widgets pour l'apparence souhaitée



Le projet terminé, le bouton « sharing » permet de générer un lien pour l'utilisateur