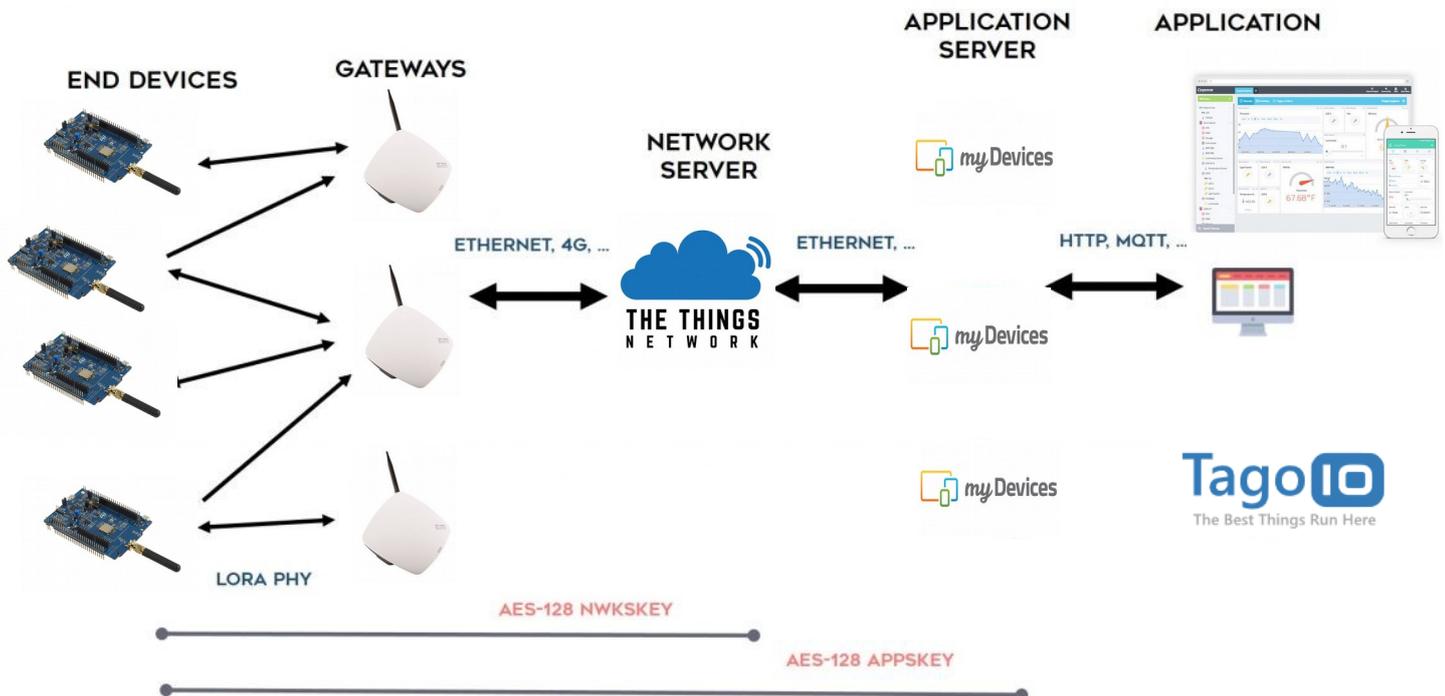


## Réalisation d'un objet connecté à une application en ligne avec LoRaWan



**Objectifs :** Réaliser un objet connecté à une application en ligne par protocole LoRaWan

- Configurer le node B-L072Z-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 MBED  
*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>

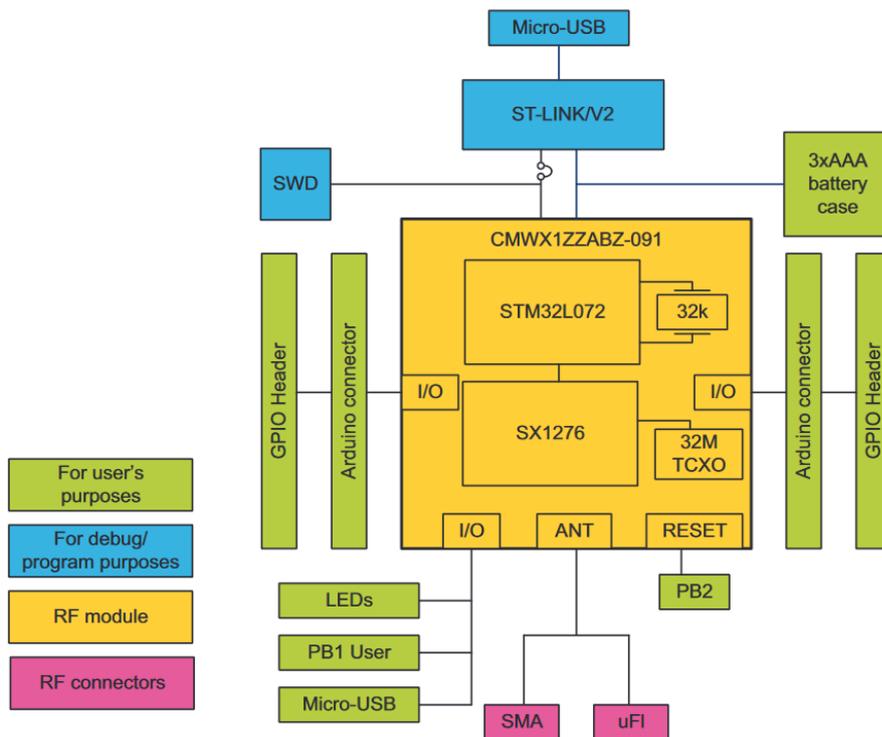
Documentation B-L072Z-LRWAN1 : <https://www.st.com/en/evaluation-tools/b-l072z-lrwan1.html>

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](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 constituant 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 leur 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

**ADD APPLICATION**

**Application ID**  
The unique identifier of your application on the network

testlrwan ✓

**Description**  
A human readable description of your new app

essai B-L072Z-LRWAN1 vers TTN ✓

**Application EUI**  
An application EUI will be issued for The Things Network block for convenience, you can add your own in the application settings page.

EUI issued by The Things Network

**Handler registration**  
Select the handler you want to register this application to

ttn-handler-eu ✓

Cancel Add application

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

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

The screenshot displays the TTN application management interface. It is divided into several sections:

- APPLICATION OVERVIEW:** Shows Application ID `testlrwan`, Description `essai B-L072Z-LRWAN1 vers TTN`, Created `1 minute ago`, and Handler `ttn-handler-eu (current handler)`. A `documentation` link is visible.
- APPLICATION EUIS:** Shows a hexagonal EUI value `70 B3 D5 7E D0 01 3F 1C`. A `manage euis` link is present.
- DEVICES:** Shows `0 registered devices`. A `register device` button (with a green plus icon) and a `manage devices` link are visible.
- COLLABORATORS:** Shows a user `cdupaty`. Buttons for `collaborators`, `delete`, `devices`, and `settings` are present. A `manage collaborators` link is also visible.
- ACCESS KEYS:** Shows a `default key` with tabs for `devices` and `messages`. A key value `ttn-account-v2.-1cAr02HvQ0v1Za9pQ97yyG5961s01tWrWoaAbscoZI` is shown in base64 format. A `manage keys` link is present.

Blue arrows from the text above point to the `register device` button and the `manage keys` link.

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

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

### REGISTER DEVICE

[bulk import devices](#)

**Device ID**  
This is the unique identifier for the device in this app. The device ID will be immutable.

capteurvirtuel

**Device EUI**  
The device EUI is the unique identifier for this device on the network. You can change the EUI later.

12 34 56 78 90 AB CD 55

**App Key**  
The App Key will be used to secure the communication between you device and the network.

this field will be generated

**App EUI**

70 B3 D5 7E D0 01 3F 1C

Cancel Register

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

### DEVICE OVERVIEW

Application ID testlrwan

Device ID capteurvirtuel

Activation Method OTAA

Device EUI <> 12 34 56 78 90 AB CD 55

Application EUI <> 70 B3 D5 7E D0 01 3F 1C

App Key <> 7A 51 D3 ED E7 AB E6 54 1E 85 33 CB 10 D7 77 78

Status ● never seen

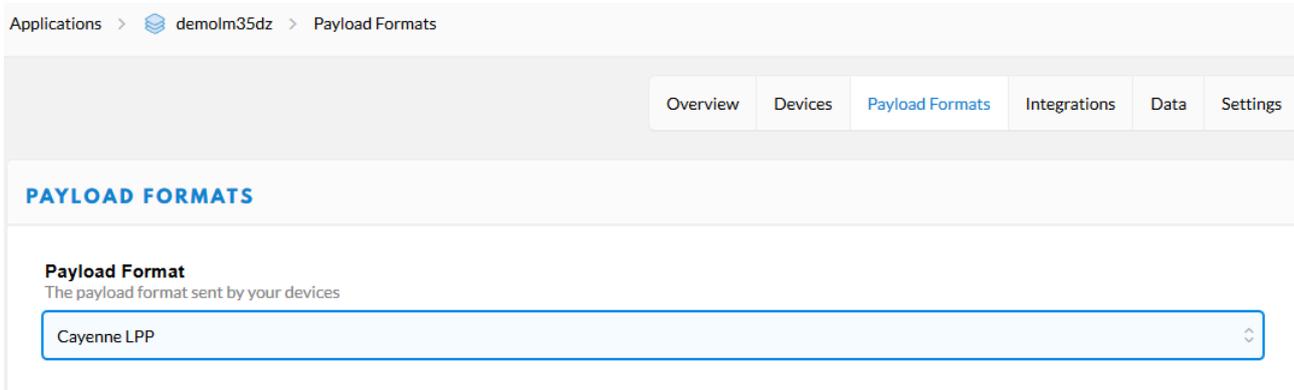
Frames up 0 [reset frame counters](#)

Frames down 0

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

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

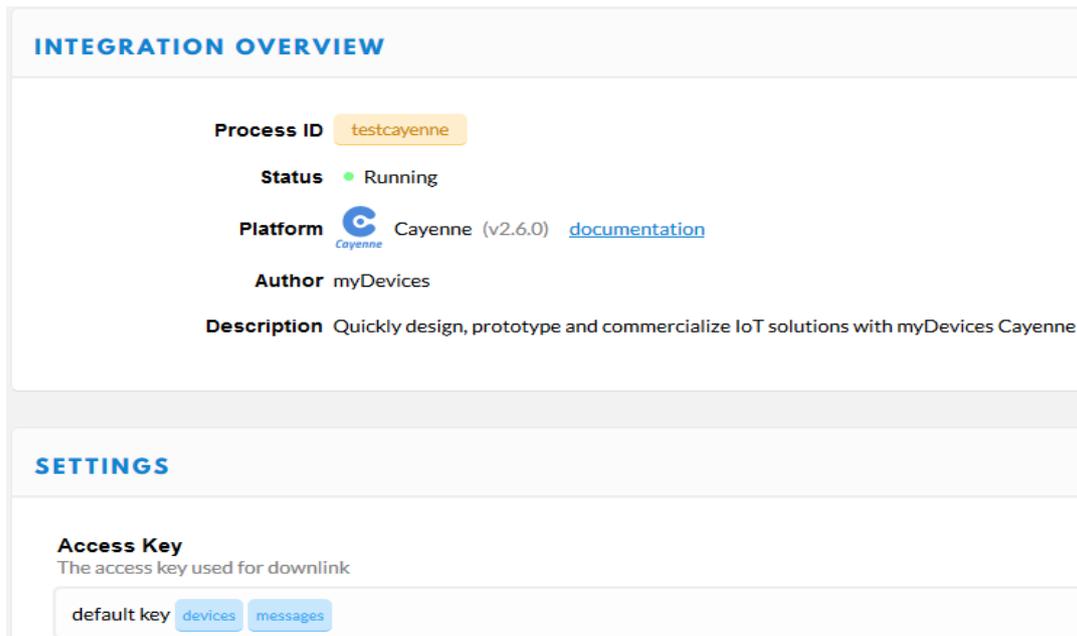
Indiquer le format des données entrantes dans « payload format »



The screenshot shows the 'Payload Formats' configuration page for an application named 'demolm35dz'. The breadcrumb navigation is 'Applications > demolm35dz > Payload Formats'. The page has a top navigation bar with tabs for 'Overview', 'Devices', 'Payload Formats' (selected), 'Integrations', 'Data', and 'Settings'. Below the navigation, the section is titled 'PAYLOAD FORMATS'. Underneath, there is a sub-section 'Payload Format' with the description 'The payload format sent by your devices'. A dropdown menu is open, showing 'Cayenne LPP' as the selected option.

Aller dans applications, sélectionner l'application créée.

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



The screenshot shows the 'Integration Overview' and 'Settings' sections for a Cayenne integration. The 'INTEGRATION OVERVIEW' section displays the following information: 'Process ID' is 'testcayenne', 'Status' is 'Running' (indicated by a green dot), 'Platform' is 'Cayenne (v2.6.0)' with a link to 'documentation', 'Author' is 'myDevices', and 'Description' is 'Quickly design, prototype and commercialize IoT solutions with myDevices Cayenne'. The 'SETTINGS' section is titled 'Access Key' with the description 'The access key used for downlink'. Below this, there are two buttons: 'default key' and 'messages'.

## 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**)

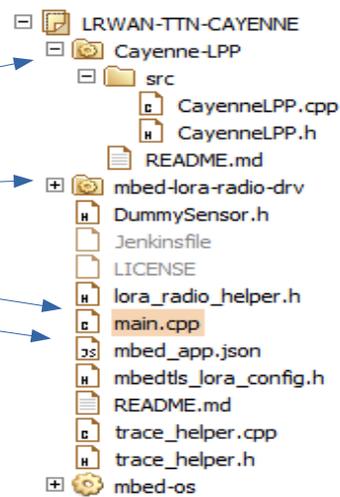
Le projet doit ressembler à ceci avec en particulier :

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
```

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

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.

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

Les données transmises sont visualisables sur la console

```
COM14 - Tera Term VT
Fichier Edition Configuration Contrôle Fenêtre(W) Aide
Emission du packet :
01 67 00 C7 02 00 01 03 01 00
10 bytes scheduled for transmission
Message Sent to Network Server

Emission : 53
-----
Capteur LM35 : 19.99 C
Emission du packet :
01 67 00 C7 02 00 01 03 01 00
10 bytes scheduled for transmission
Message Sent to Network Server

Emission : 54
-----
Capteur LM35 : 19.90 C
Emission du packet :
01 67 00 C7 02 00 01 03 01 00
10 bytes scheduled for transmission
Message Sent to Network Server
```

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

Applications > demolm35dz > Data

Overview Devices Payload Formats Integrations **Data** Settings

### APPLICATION DATA

Filters: uplink downlink activation ack error

time	counter	port	dev id: testlm35	payload: 01 67 00 C6 02 00 01 03 01 00	digital_in_2: 1	digital_out_3: 0	temperature_1: 19.99
15:45:49	0		dev id: testlm35				
15:45:50	15	15	confirmed dev id: testlm35	01 67 00 C6 02 00 01 03 01 00	1	0	19.99
15:45:48	0		dev id: testlm35				
15:45:49	14	15	confirmed dev id: testlm35	01 67 00 C7 02 00 01 03 01 00	1	0	19.90
15:45:43	0		dev id: testlm35				
15:45:44	13	15	confirmed dev id: testlm35	01 67 00 C7 02 00 01 03 01 00	1	0	19.90

## 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

Sélectionner 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

This device uses [Cayenne LPP](#)

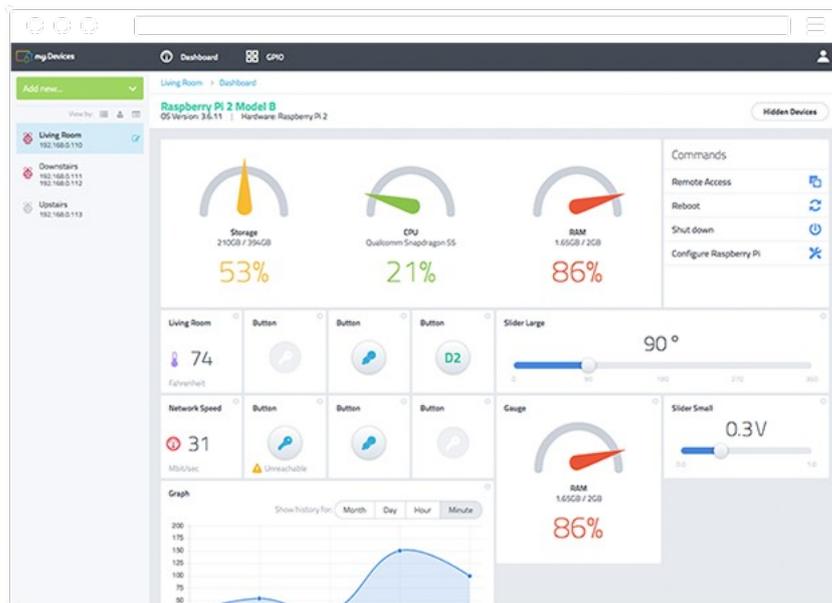
Name  
LRWAN test

DevEUI  
1234567890ABCD55

Activation Mode  
Already Registered

Tracking  
Location  
This device moves

Add device





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

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.

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

### 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 défaut
```

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.

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

```
COM14 - Tera Term VT
Fichier Edition Configuration Contrôle Fenêtre(W) Aide
Message Sent to Network Server
Emission : 4
-----
Dummy Sensor Value = 0.4
Capteur LM35 : 19.99 C
Emission du packet :
01 65 00 6F 02 66 01 03 67 00 00 04 68 05 05 00 01 06 02 02 5D 07 01 00
08 67 00 C7 09 67 00 04
32 bytes scheduled for transmission
Received message from Network Server
Reception on port : 99
Flags are : 2
Data: 0F 00 64 FF
Data Length: 4
Message Sent to Network Server
Emission : 5
-----
Dummy Sensor Value = 0.5
Capteur LM35 : 20.23 C
Emission du packet :
01 65 00 6F 02 66 01 03 67 00 00 04 68 05 05 00 01 06 02 02 60 07 01 00
08 67 00 CA 09 67 00 05
32 bytes scheduled for transmission
Received message from Network Server
Reception on port : 99
Flags are : 2
Data: 0F 00 00 FF
Data Length: 4
```

## 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

