

Ecole Publique d'Ingénieurs en 3 ans

Projet I am ENSICAEN

# DOCUMENTATION DE LA GATEWAY THE THINGS NETWORK

le 26 septembre 2022,  
version 1.1

Lucille ANTOINE,  
3A SATE  
[lucille.antoine@ecole.ensicaen.fr](mailto:lucille.antoine@ecole.ensicaen.fr)

Tuteur école : Hugo Descoubes /  
Philippe Lefebvre / Dimitri Boudier



[www.ensicaen.fr](http://www.ensicaen.fr)

# TABLE DES MATIERES

---

1.	Contexte	3
2.	Configuration de la gateway	3
3.	Connexion de la gateway avec le module de démo	4
4.	Construction des applications sur The Things Network	5
5.	Connexion aux canaux de ThingSpeak	9
6.	Visualisation des données avec Cayenne	16

# TABLE DES FIGURES

---

Figure 1 :	Localisation de l'App/JoinEUI et de l'AppKey	7
Figure 2 :	STM32-localisation du DevEUI	8
Figure 3 :	Mise en évidence du délai de reconnexion d'un device	8
Figure 4 :	Mise en évidence de la macro à modifier	9
Figure 5 :	Channel ID et API Key de ThingSpeak	12

# MISE EN ROUTE DE LA GATEWAY

---

## 1. Contexte

Lors du projet immersif de cette année 2022, qui vise à équiper les bâtiments de l'Ensi de capteurs pour faire un diagnostic de ceux-ci, nous avons décidé d'utiliser la gateway The Things Network (TTN) <https://www.thethingsnetwork.org/docs/gateways/gateway/>, en parallèle de développer une gateway maison, pour faire le lien entre les capteurs et l'application. Le but d'utiliser cette gateway, plutôt que celle maison, est qu'elle est déjà stable car elle est disponible sur le marché. Nous pouvons alors communiquer via LoraWAN, ce qui n'est pas encore possible à ce jour sur la gateway développée lors de ce projet par d'autres étudiants. La fréquence du LoraWAN en Europe est de 868MHz. Nous travaillerons donc autour de celle-ci. Cette gateway a déjà été utilisée par Philippe Lefebvre sur d'autres projets par le passé, le but étant de l'utiliser dans notre projet.



## 2. Configuration de la gateway

Pour se connecter à TTN <https://eu1.cloud.thethingsnetwork/console/>, voici les identifiants :

- Log in : [philippe.lefebvre@ensicaen.fr](mailto:philippe.lefebvre@ensicaen.fr)
- Mot de passe : Jastkemt0

Pour configurer la gateway, il faut d'abord l'ajouter sur TTN, en cliquant sur « Register gateway », puis en saisissant l'identifiant de la gateway dans « Gateway EUI » ou non. A voir avec Philippe Lefebvre pour la configuration, il l'a réalisé avant que nous reprenions le projet.

The screenshot shows the TTN Gateway management interface. The top navigation bar includes 'Overview', 'Applications', 'Gateways', and 'Organizations'. The 'Gateways' section is active, displaying a table with two gateways:

ID	Name	Gateway EUI	Status	Created at
eui-b827ebfffe4841dc	Raspberry_CR02	B8 27 EB FF FE 48 41 DC	Disconnected	7 days ago
tts-ensicaen	Gateway Ensicaen TTS	none	Connected	Sep 15, 2021

Below the table, the 'Register gateway' form is visible. It includes a 'Gateway EUI' input field with a red box around it, and a 'Continue without EUI' button. The text below the form reads: 'To continue, please confirm the Gateway EUI so we can determine onboarding options'.

Il faut que cette gateway soit connectée au réseau avec des requêtes spécifiques pour pouvoir y accéder. Voici celle qu'on a utilisé pour demander à la DSI les autorisations:

- Machine d'adresse MAC: 54:10:ec:41:28:55
- Prise 1E11 bureau A108 (la box devrait être placée dans le bureau A202 à terme)
- La machine doit pouvoir faire :
  - HTTPS (443) sortant
  - ping sur 8.8.4.4
  - NTP (123) sur pool.ntp.org
  - MQTTS sur eu1.cloud.thethings.network sur le port 8881

Nous avons communiqué ces informations à la DSI pour qu'ils puissent nous ouvrir les ports et obtenir les autorisations nécessaires. Théoriquement l'adresse mac de cette gateway a été fixée dans le VLAN 214 (réseau étudiant) de l'école pour pouvoir faire les configurations nécessaires. Nous avons principalement interagi avec Laurent Cousin à la DSI.

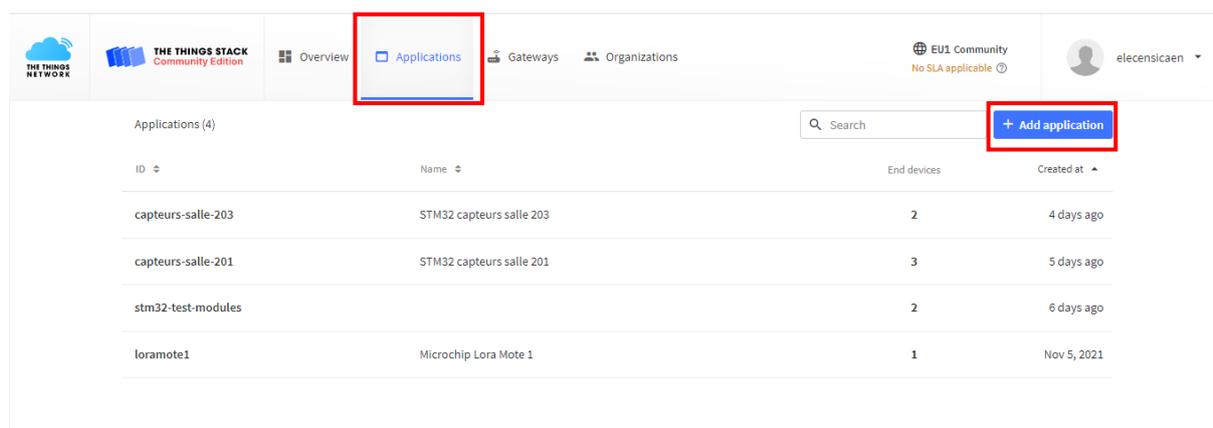
### 3. Connexion de la gateway avec le module de démo

Il existe un module de démonstration. Une application sur TTN est déjà créée pour l'utiliser sous le nom loramote1. Elle a été développée par Philippe Lefebvre. Il a une trace de la configuration et de la trame à faire pour connecter ce module à la gateway. Contacter Philippe Lefebvre au besoin pour essayer de le rejouer.

# UTILISATION DE LA GATEWAY

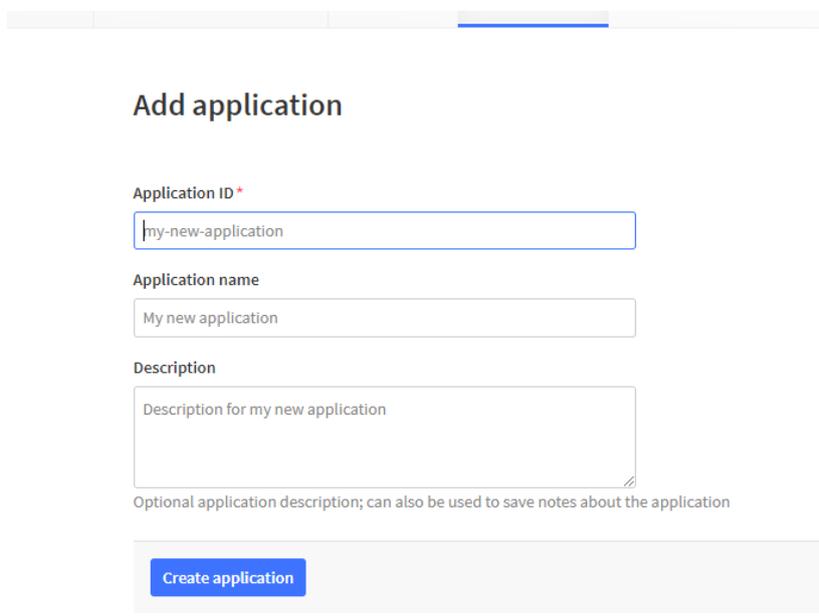
## 4. Construction des applications sur The Things Network

Pour construire une nouvelle application il faut aller dans l'onglet application de TTN et ajouter une nouvelle application en cliquant sur le bouton « [Add application](#) ». Une application est un espace dans lequel on peut coupler différents devices et leur définir un comportement similaire.



ID	Name	End devices	Created at
capteurs-salle-203	STM32 capteurs salle 203	2	4 days ago
capteurs-salle-201	STM32 capteurs salle 201	3	5 days ago
stm32-test-modules		2	6 days ago
loramote1	Microchip Lora Mote 1	1	Nov 5, 2021

Il faut ensuite donner un ID à l'application dans le champ « [Application ID](#) ». Il ne doit contenir que des minuscules, chiffres, ou le caractère « - ». C'est facultatif mais conseillé de donner un nom à l'application.



**Add application**

Application ID\*

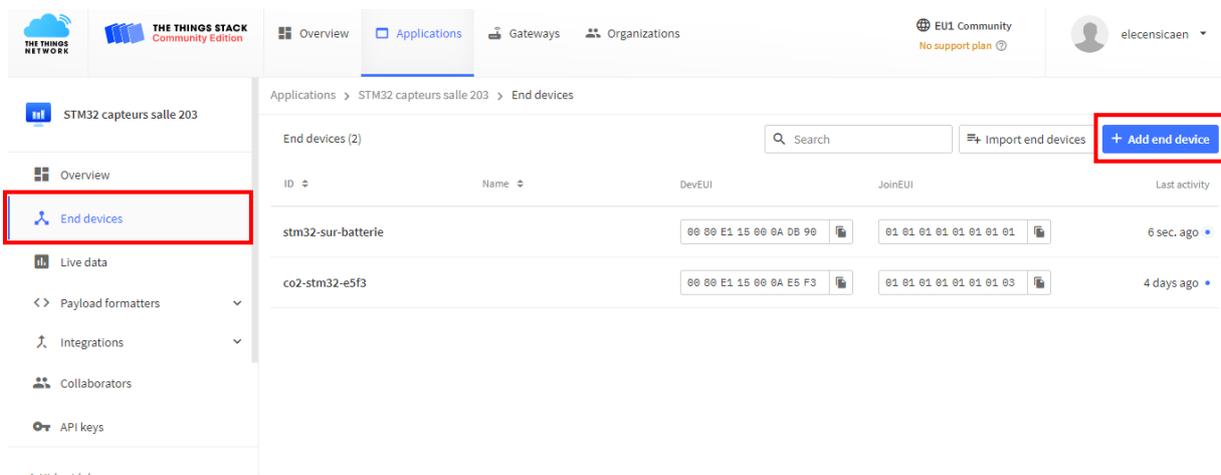
Application name

Description

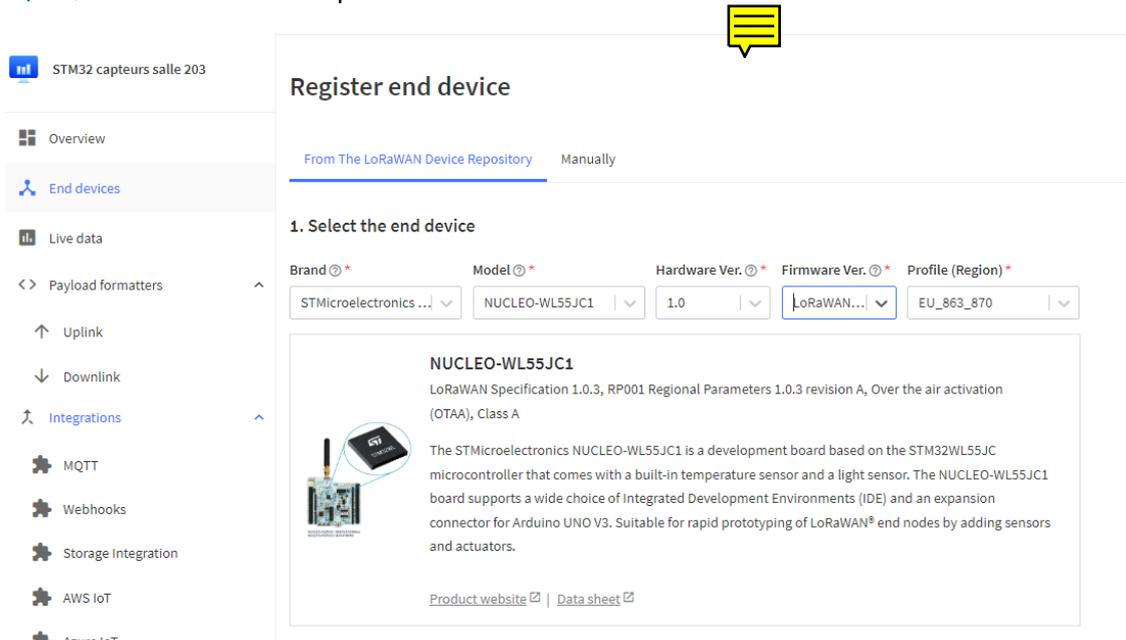
Optional application description; can also be used to save notes about the application

Nous avons choisi d'utiliser une nomenclature particulière pour les IDs : « *capteurs-salle-numéro* » avec le numéro de la salle à définir en fonction de la salle pour laquelle on veut traiter les données des capteurs via cette application.

Une fois l'application créée, il faut y ajouter des devices. Un device correspond dans notre cas à une carte STM32 avec son module. Pour ajouter le device il faut aller dans la partie « *End devices* » puis cliquer sur « *Add end device* ».



Il faut ensuite chercher la technologie utilisée : pour nous le NUCLEO-WL55JC1 de chez STMicroelectronics. Nous n'avons pas encore établi avec certitude quel « *Firmware Version* » nous devons utiliser. Mais nous avons testé à la fois la 1.1.0 et la 1.2.0. Pour ce qui concerne le « *Profil* », il faut choisir l'europpéen : EU-868-870.



Pour le « *Frequency plan* », prendre celui recommandé fonctionne très bien et il correspond à la norme européenne de 868MHz. Il faut ensuite renseigner les EUI et keys correspondant à la carte que nous souhaitons coupler. L'« *AppEUI* » (ou « *JoinEUI* » en fonction de la version du firmware) doit être le même que celui configuré sur STM32 Cube IDE, à voir sur la Figure 1. Le « *DevEUI* » est indiqué sur le STM32 comme sur la zone encadrée sur la Figure 2.

- <> Payload formatters
- ↑ Uplink
- ↓ Downlink
- Integrations
  - MQTT
  - Webhooks
  - Storage Integration
  - AWS IoT
  - Azure IoT
  - LoRa Cloud
- Collaborators
- API keys
- General settings

< Hide sidebar

## 2. Enter registration data

Frequency plan  ⓘ \*

AppEUI  ⓘ \*  
 Fill with zeros

DevEUI  ⓘ \*  
 Generate 0/50 used

AppKey  ⓘ \*  
 Generate

End device ID  ⓘ \*  
  
This value is automatically prefilled using the DevEUI

**After registration**

View registered end device

Register another end device of this type

Register end device

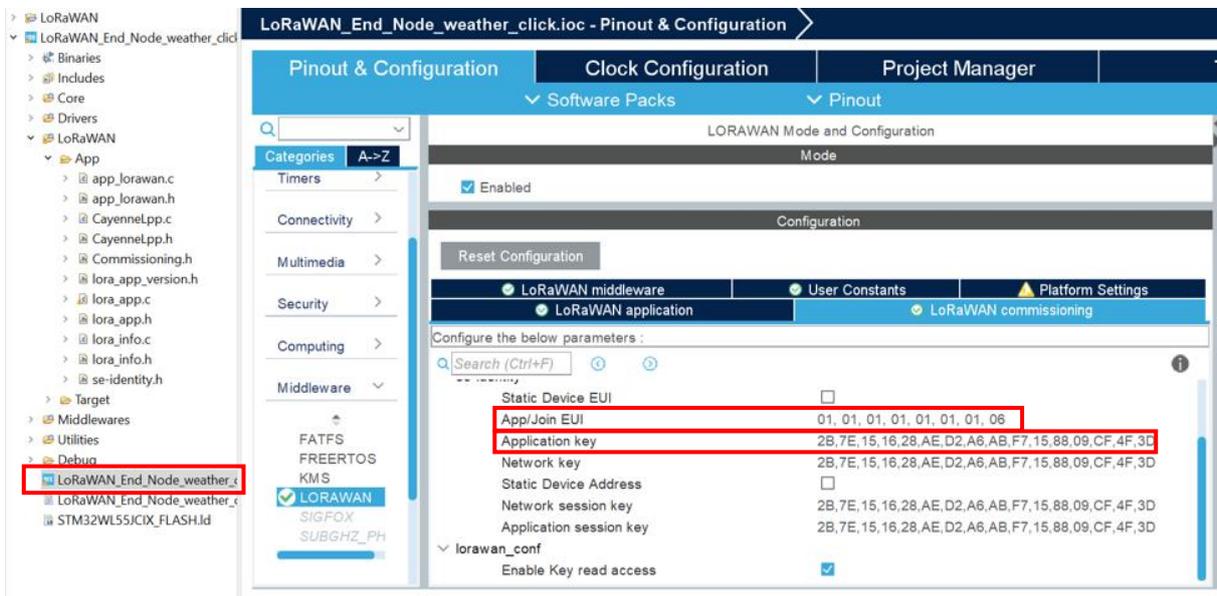


Figure 1 : Localisation de l'App/JoinEUI et de l'AppKey



Figure 2 : STM32-localisation du DevEUI

Nous avons fait face à quelques problèmes et interrogations. Tout d'abord le temps de reconnexion du device, après un reset ou déconnexion de celui-ci, était de plus en plus long à chaque fois. Il y avait des messages d'erreur tels que « *Device Nounce too small* » ou « *Device Nounce has already been used* », dans la partie « *Live Data* » qui permet de voir les flux de données sur l'application. Il fallait un certain délai, entre 1 et 2 minutes pour que le capteur se reconnecte. Ceci est visible sur la Figure 3.

STM32 capteurs salle 201

- Overview
- End devices
- Live data**
- Payload formatters
- Integrations
- Collaborators
- API keys
- General settings

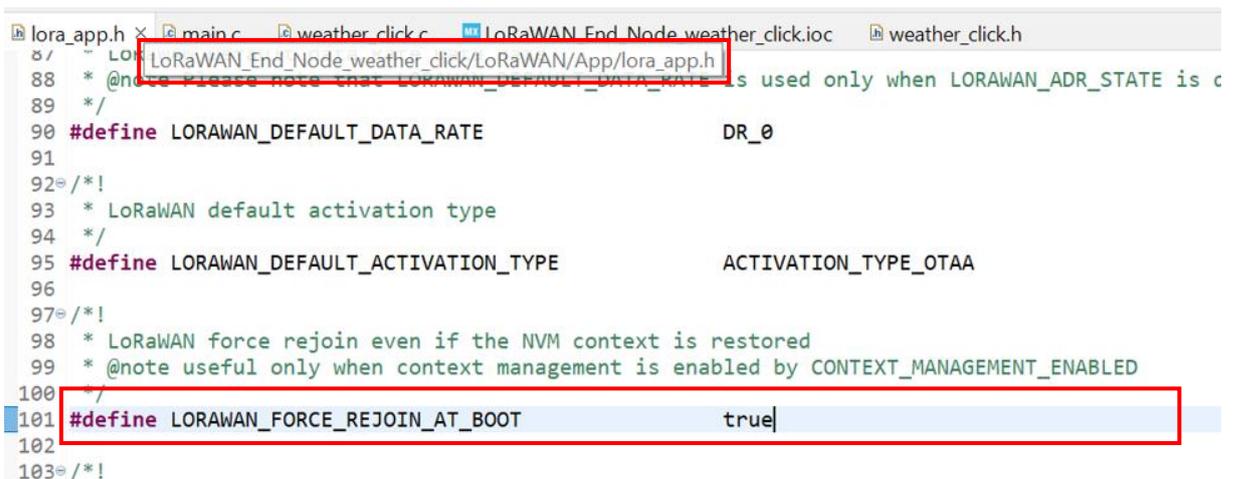
Applications > STM32 capteurs salle 201 > Live data

Time	Entity ID	Type	Data preview	Verbose stream	Export as JSON	Pause	Clear
10:49:03	weather-stm32-d5...	Fail to send webhook	Request	<input checked="" type="checkbox"/>			
↑ 10:49:02	weather-stm32-d5...	Forward uplink data message	DevAddr: 26 0B 71 10 <> Payload: { field1: 21.13, field2: 50.66, field3: 998.1 }	<input type="checkbox"/>			
↑ 10:48:54	weather-stm32-d5...	Forward uplink data message	DevAddr: 26 0B 71 10 <> Payload: { field1: 21.15, field2: 50.81, field3: 998.01 }	<input type="checkbox"/>			
↑ 10:48:45	weather-stm32-d5...	Forward join-accept message	DevAddr: 26 0B 71 10 <>	<input type="checkbox"/>			
⌵ 10:48:43	weather-stm32-d5...	Accept join-request	DevAddr: 26 0B 71 10 <>	<input type="checkbox"/>			
↑ 10:48:33	weather-stm32-d5...	Join-request to cluster-lo...	DevNonce has already been used	<input type="checkbox"/>			
↑ 10:48:23	weather-stm32-d5...	Join-request to cluster-lo...	DevNonce has already been used	<input type="checkbox"/>			
↑ 10:48:13	weather-stm32-d5...	Join-request to cluster-lo...	DevNonce has already been used	<input type="checkbox"/>			
↑ 10:48:03	weather-stm32-d5...	Join-request to cluster-lo...	DevNonce has already been used	<input type="checkbox"/>			
↑ 10:47:52	weather-stm32-d5...	Join-request to cluster-lo...	DevNonce has already been used	<input type="checkbox"/>			

Figure 3 : Mise en évidence du délai de reconnexion d'un device

Nous avons d'abord essayé en ayant le même AppEUI et AppKey pour tous nos devices, mais pour essayer d'améliorer les temps de connexion nous avons décidé d'en créer un différent pour chaque. Toutefois le problème ne semble pas résolu. Le temps est un peu plus court mais il y a toujours plusieurs échecs avant la connexion et la raison nous est encore inconnue. Nous ne savons donc pas s'il y a un intérêt à changer les AppEUI et AppKey, cela reste à investiguer.

Finalement la solution trouvée qui semble être efficace est de changer la macro LORAWAN\_FORCE\_REJOIN\_AT\_BOOT de false à true dans le fichier lora\_app.h présenté en Figure 4. Le join otaa se fait donc très rapidement, en moins de 30sec après le reset de la carte.



```
lora_app.h x main.c weather_click.c LoRaWAN_End_Node_weather_click.ioc weather_click.h
87 // LoRaWAN_End_Node_weather_click/LoRaWAN/App/lora_app.h
88 * @note Please note that LORAWAN_DEFAULT_DATA_RATE is used only when LORAWAN_ADR_STATE is c
89 */
90 #define LORAWAN_DEFAULT_DATA_RATE DR_0
91
92= /*!
93 * LoRaWAN default activation type
94 */
95 #define LORAWAN_DEFAULT_ACTIVATION_TYPE ACTIVATION_TYPE_OTAA
96
97= /*!
98 * LoRaWAN force rejoin even if the NVM context is restored
99 * @note useful only when context management is enabled by CONTEXT_MANAGEMENT_ENABLED
100 */
101 #define LORAWAN_FORCE_REJOIN_AT_BOOT true
102
103= /*!
```

Figure 4 : Mise en évidence de la macro à modifier

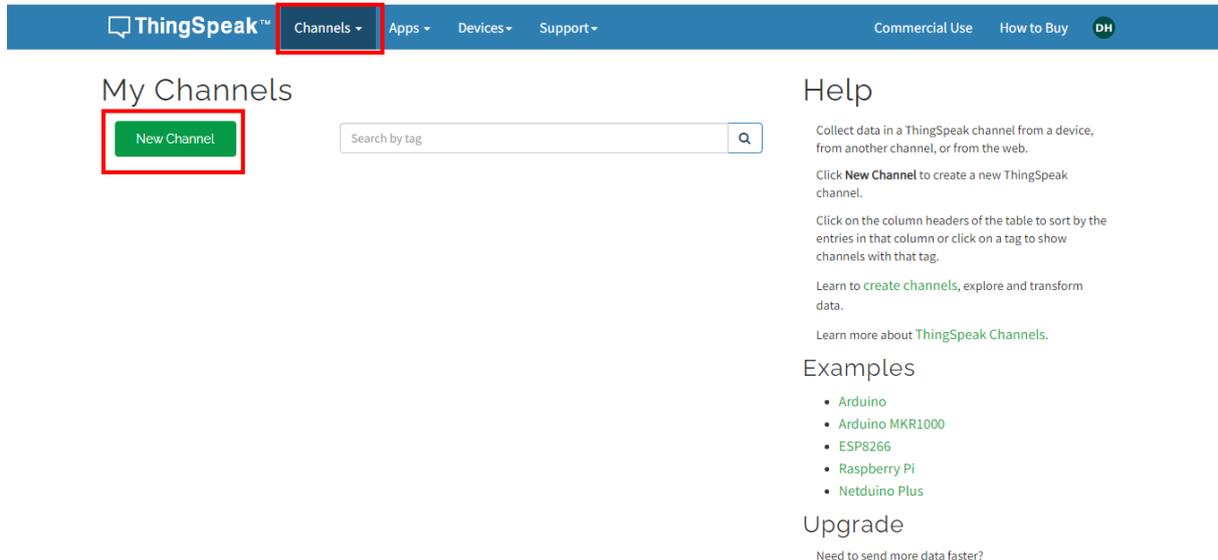
## 5. Connexion aux canaux de ThingSpeak

Une fois nos devices connectés, nous avons choisi de les relier à ThingSpeak : <https://thingspeak.com/channels>. Grâce à ce webhook, nous pouvons visualiser rapidement nos données depuis un ordinateur, et également les récupérer via des requêtes http pour collecter les données et ensuite les afficher dans l'application développée par d'autres étudiants. Cet outil fonctionne avec des canaux, qui possèdent chacun une clé d'écriture et une clé de lecture. On peut créer jusqu'à 8 champs dans chaque canal. Pour nous un champ correspond à une variable venant d'un capteur : par exemple la température. De plus cette solution est proposée par MathWorks. Nous avons donc des comptes disponibles avec la licence de l'école. Toutefois un compte étudiant ne permet de créer que 10 canaux maximum. Néanmoins un compte académique permet de créer 250 canaux, d'où le fait que nous avons utilisé le compte de Hugo Descoubes pour faire nos canaux.

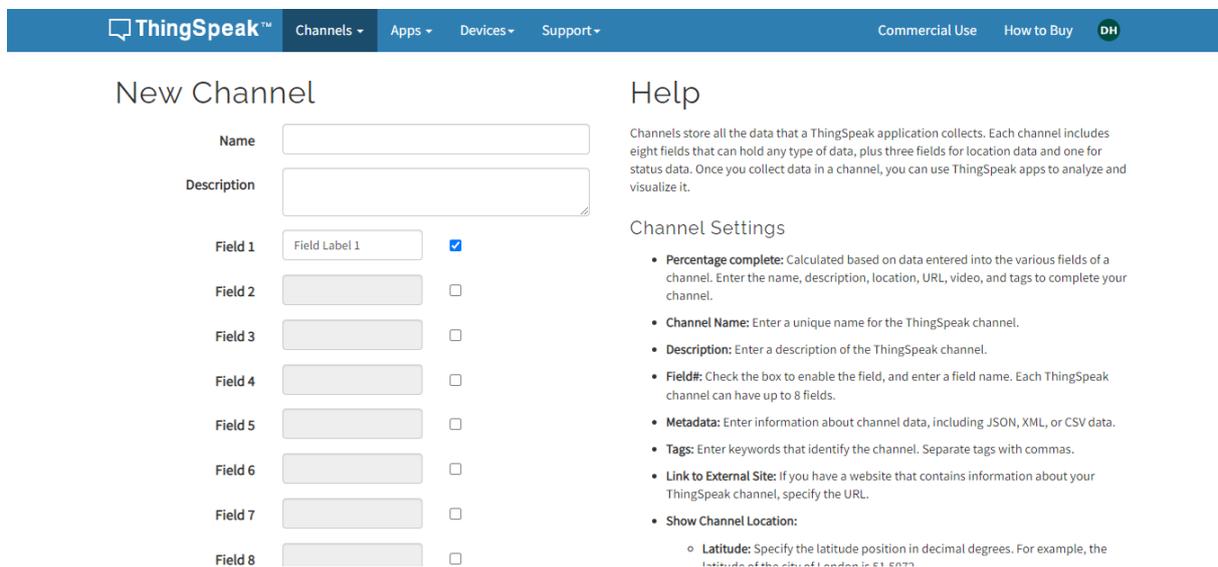
- Identifiants de connexion : [hugo.descoubes@ensicaen.fr](mailto:hugo.descoubes@ensicaen.fr)
- Mot de passe : projetIMMERSIF14

Nous devons également aller voir la DSI afin de savoir s'il est possible d'avoir un compte académique dédié au projet immersif.

Une fois connecté à ThingSpeak, il faut créer un nouveau canal et le paramétrer. Pour cela il faut cliquer que « *New Channel* » dans l'onglet « *Channel* → *My Channels* ».



Il faut ensuite donner un nom à ce canal et débloqué autant de fields que souhaité en les cochant. Il faut renommer chaque field avec le nom de la data qu'on veut visualiser dans ce field.



On peut aussi modifier et rajouter les field dans une limite de 8 fields. Il faut donc construire une application par salle, tant qu'il y a moins de 8 capteurs par salle. S'il y a plus de 8 capteurs, il faudra alors créer plusieurs applications pour la même salle. Pour modifier les fields il suffit d'aller dans les settings du canal.

## salle201-stm32

Channel ID: 1873174  
 Author: mwa000004343771  
 Access: Private

Private View Public View **Channel Settings** Sharing API Keys Data Import / Export

### Channel Settings

Percentage complete 30%

Channel ID 1873174

Name

Description

Field 1

Field 2

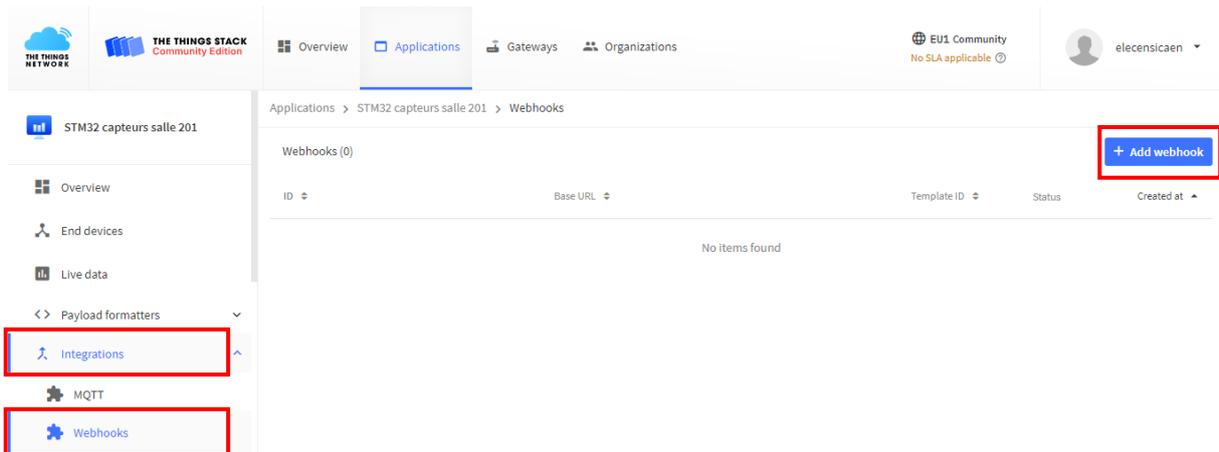
### Help

Channels store all the data that a ThingSpeak application collects. Each channel includes eight fields that can hold any type of data, plus three fields for location data and one for status data. Once you collect data in a channel, you can use ThingSpeak apps to analyze and visualize it.

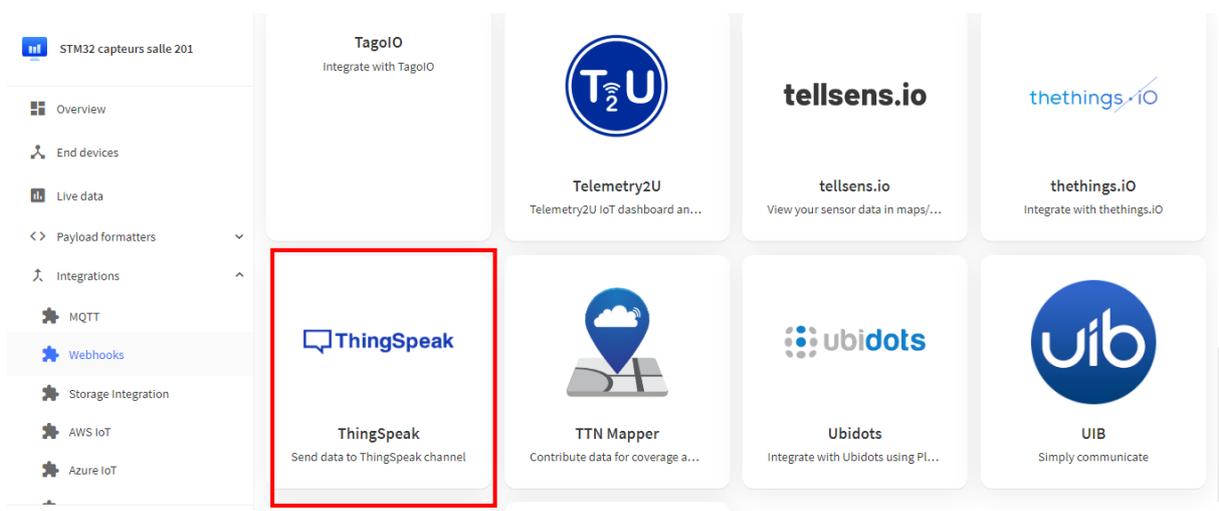
### Channel Settings

- Percentage complete:** Calculated based on data entered into the various fields of a channel. Enter the name, description, location, URL, video, and tags to complete your channel.
- Channel Name:** Enter a unique name for the ThingSpeak channel.
- Description:** Enter a description of the ThingSpeak channel.
- Field#:** Check the box to enable the field, and enter a field name. Each ThingSpeak

Une fois le canal créé et paramétré à nos souhaits, il faut le relier à TTN. Pour ce faire il faut aller dans « [Integrations](#) », puis « [Webhook](#) » et ajouter le webhook via le bouton « [Add webhook](#) ».



Il faut ensuite faire défiler jusqu'à avoir le logo ThingSpeak.



Pour connecter le canal, il faut lui donner un identifiant interne à TTN, dans « *Webhook ID* ». Pour l'identifiant du canal, il faut aller le relever dans ThingSpeak. C'est également le cas pour l'« *API Keys* », il faut y rentrer la « *Write API Key* » de ThingSpeak. Cf Figure 5.

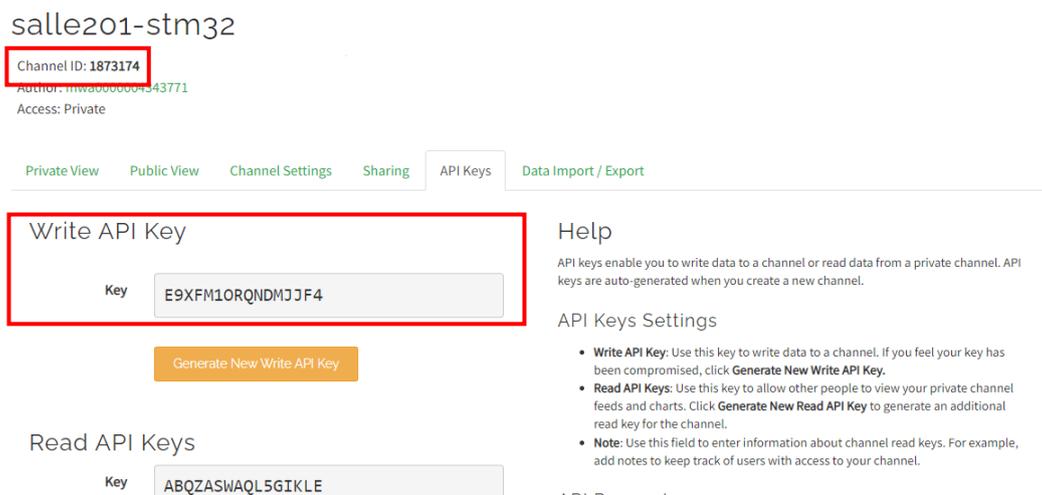
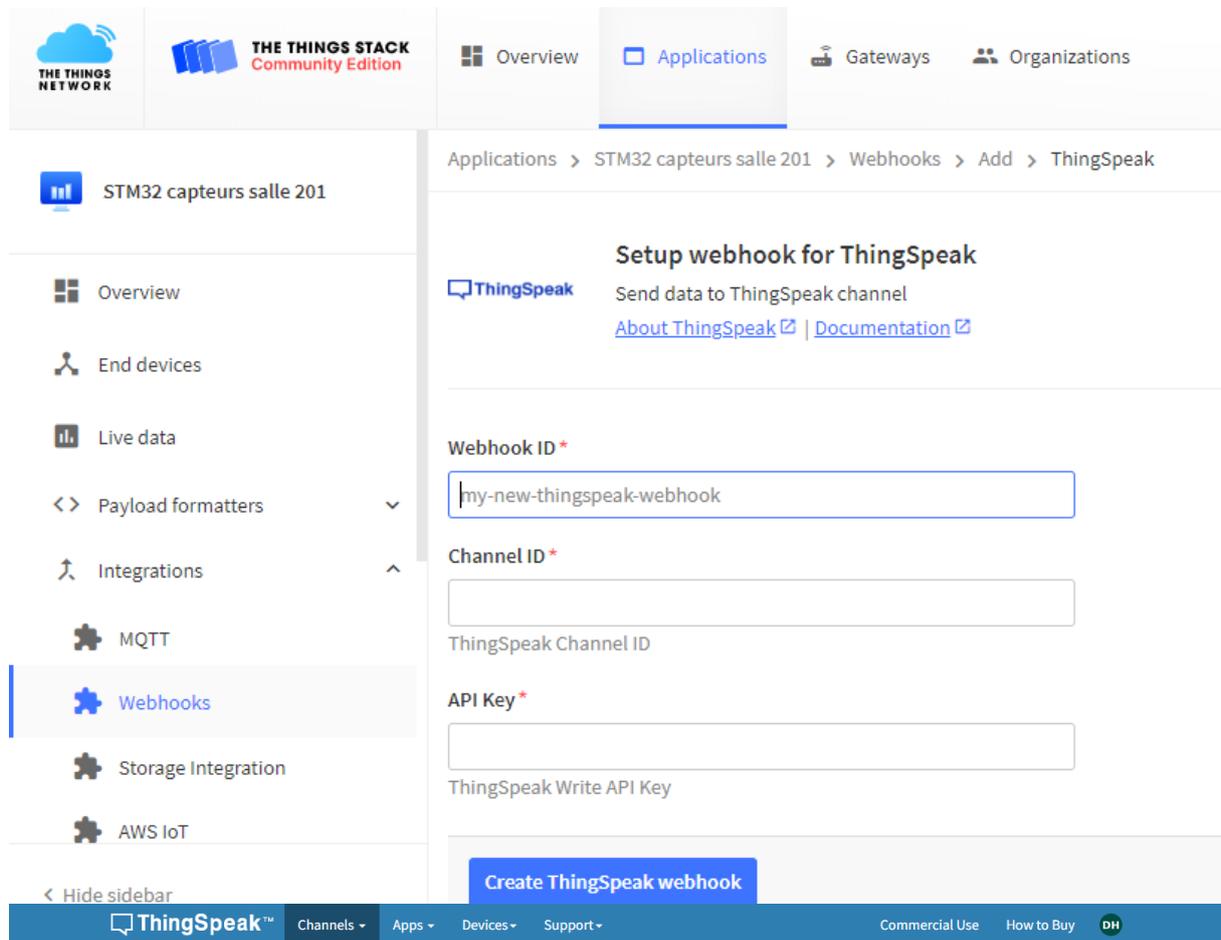


Figure 5 : Channel ID et API Key de ThingSpeak

La connection est bien établie lorsque le canal a un statut Healthy sur TTN.

Applications > STM32 capteurs salle 201 > Webhooks

ID	Base URL	Template ID	Status	Created at
thingspeak-salle-201	https://api.thingspeak.com/things_network/v3/update	thingspeak	Healthy	3 hours ago

Maintenant que le canal est connecté, il faut y envoyer les bonnes données. Pour appliquer son propre formatage de données, il faut aller dans « [Payload Formatters](#) → [Uplink](#) » et sélectionner le type « [Custom Javascript formatter](#) ». Il est alors possible d'écrire son formatage.

Applications > STM32 capteurs salle 201 > Uplink > Payload formatters

### Default uplink payload formatter

You can use the "Payload formatter" tab of individual end devices to test uplink payload formatters and to define individual payload formatter settings per end device.

**Setup**

Formatter type\*  
Custom Javascript formatter

Formatter code\*

```
1 function Decoder(b, port) {  
2   var identity = b[0];  
3   if (identity == 2) {  
4     var temperature = ((b[1] << 8) + b[2])/100;  
5     var humidity = ((b[6] << 8) + b[7])/100;  
6     var pressure = ((b[3] << 16) + (b[4] << 8) + b[5])/100;  
7     return {  
8       field1: temperature,  
9       field2: humidity,  
10      field3: pressure  
11    };  
12  }  
13  if (identity == 1) {  
14    var C02 = ((b[1] << 8) + b[2]);  
15    var TVOC = ((b[3] << 8) + b[4]);  
16  }
```

De même dans le « [End devices](#) », sur chaque device dans « [Payload formatters](#) », il faut mettre le script de formatage.

STM32 capteurs salle 201

### weather-stm32-d573

ID: weather-stm32-d573

↑ 185 ↓ 15 Last activity just now

Overview Live data Messaging Location **Payload formatters** Claiming General settings

Uplink Downlink

**Setup**

Formatter type\*  
Custom Javascript formatter

Formatter code\*

```
1 function Decoder(b, port) {  
2   var identity = b[0];  
3   if (identity == 2) {  
4     var temperature = ((b[1] << 8) + b[2])/100;  
5     var humidity = ((b[6] << 8) + b[7])/100;  
6     var pressure = ((b[3] << 16) + (b[4] << 8) + b[5])/100;  
7     return {  
8       field1: temperature,  
9       field2: humidity,  
10      field3: pressure  
11    };  
12  }  
13  if (identity == 1) {  
14    var C02 = ((b[1] << 8) + b[2]);  
15    var TVOC = ((b[3] << 8) + b[4]);  
16  }
```

**Test**

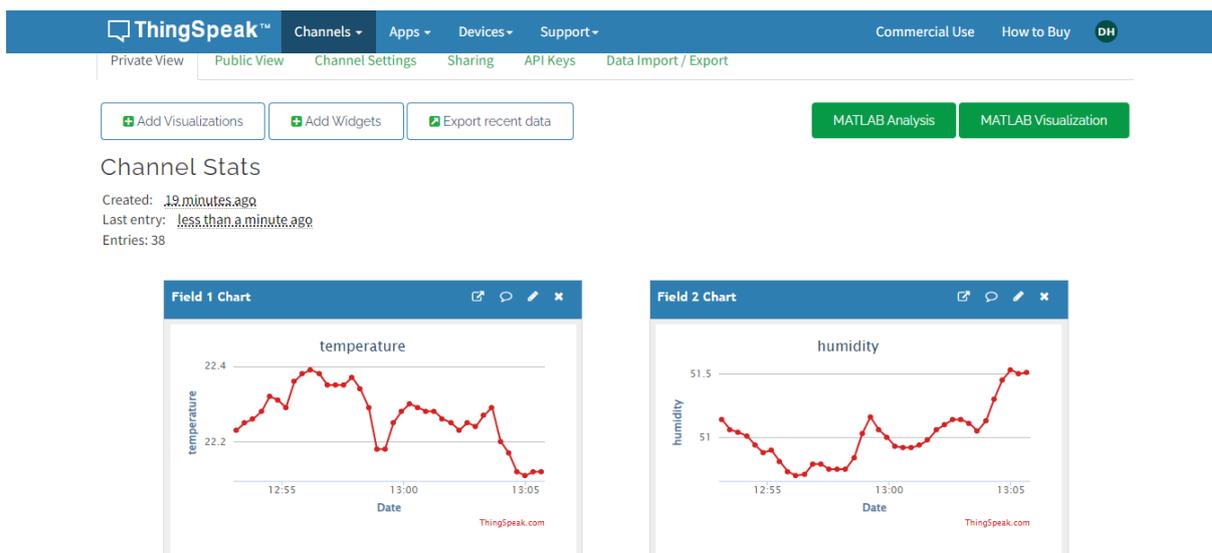
Byte payload FPort  
Test decoder

Decoded test payload

On a identifié chaque capteur sur le premier octet. 0x01 correspond au capteur de CO2/TVOC, celui en 0x02 au capteur de température/humidité/pression et celui en 0x03 au capteur de luminosité. Et ensuite les octets restants sont traités différemment en fonction du capteur pour obtenir les bonnes valeurs pour chaque variable.

```
1 function Decoder(b, port) {
2   var identity = b[0];
3   if (identity == 2) {
4     var temperature = ((b[1] << 8) + b[2])/100;
5     var humidity = ((b[6] << 8) + b[7])/100;
6     var pressure = ((b[3] << 16) + (b[4] << 8) + b[5])/100;
7     return {
8       field1: temperature,
9       field2: humidity,
10      field3: pressure
11    }
12  }
13  if (identity == 1) {
14    var CO2 = ((b[1] << 8) + b[2]);
15    var TVOC = ((b[3] << 8) + b[4]);
16    return {
17      field4: CO2,
18      field5: TVOC,
19    }
20  }
21  if (identity == 3) {
22    var light = b[1];
23    return {
24      field6: light,
25    }
26  }
27 }
```

Voici un exemple de valeurs que nous avons pu récupérer et afficher dans l'interface de ThingSpeak.



Pour pouvoir afficher les données dans l'application, il suffit de donner le lien pour lire dans la partie des « API Keys », avec aussi la clé de lecture.

ThingSpeak™ Channels Apps Devices Support Commercial Use How to Buy DH

Generate New Write API Key

### Read API Keys

Key: ABQZASWAQL5GIKLE

Note:

Save Note Delete API Key

Add New Read API Key

#### API Keys Settings

- Write API Key:** Use this key to write data to a channel. If you feel your key has been compromised, click **Generate New Write API Key**.
- Read API Keys:** Use this key to allow other people to view your private channel feeds and charts. Click **Generate New Read API Key** to generate an additional read key for the channel.
- Note:** Use this field to enter information about channel read keys. For example, add notes to keep track of users with access to your channel.

#### API Requests

Write a Channel Feed

```
GET https://api.thingspeak.com/update?api_key=E9XFM1ORQNDM3JF4&field
```

Read a Channel Feed

```
GET https://api.thingspeak.com/channels/1873174/feeds.json?api_key=
```

Read a Channel Field

```
GET https://api.thingspeak.com/channels/1873174/fields/1.json?api_ke
```

Read Channel Status Updates

```
GET https://api.thingspeak.com/channels/1873174/status.json?api_key=
```

On peut repérer quelques axes à améliorer dans cette partie. L'envoi des données échoue assez régulièrement et nous ne savons pas si cela est dû à une trop grande fréquence d'envoi d'information et que le délai de traitement est plus long. Ou bien simplement notre outil n'est pas assez fiable pour établir le lien avec le canal à chaque fois. Il faut continuer d'observer ce comportement et essayer de comprendre pourquoi il y a ces « *Fail to send webhook* » qui apparaissent.

STM32 capteurs salle 201

Overview End devices Live data Payload formatters Uplink Downlink Integrations MQTT Webhooks Storage Integration

Time	Entity ID	Type	Data preview
15:47:59	co2-stm32-d520	Fail to send webhook	Request
↑ 15:47:59	co2-stm32-d520	Forward uplink data message	DevAddr: 26 0B E5 D6 Payload: { field4: 400, field5: 0 } 01 01 90 00 00
15:47:56	weather-stm32-d5...	Fail to send webhook	Request
↑ 15:47:56	weather-stm32-d5...	Forward uplink data message	DevAddr: 26 0B 5F 71 Payload: { field1: 23.52, field2: 54.23, field3: 997.15 }
15:47:49	co2-stm32-d520	Fail to send webhook	Request
↑ 15:47:49	co2-stm32-d520	Forward uplink data message	DevAddr: 26 0B E5 D6 Payload: { field4: 400, field5: 0 } 01 01 90 00 00
↑ 15:47:46	weather-stm32-d5...	Forward uplink data message	DevAddr: 26 0B 5F 71 Payload: { field1: 23.51, field2: 54.31, field3: 997.13 }
15:47:39	co2-stm32-d520	Fail to send webhook	Request
↑ 15:47:39	co2-stm32-d520	Forward uplink data message	DevAddr: 26 0B E5 D6 Payload: { field4: 400, field5: 0 } 01 01 90 00 00
15:47:36	weather-stm32-d5...	Fail to send webhook	Request
↑ 15:47:36	weather-stm32-d5...	Forward uplink data message	DevAddr: 26 0B 5F 71 Payload: { field1: 23.48, field2: 54.48, field3: 997.15 }

## 6. Visualisation des données avec Cayenne

Avant de réussir à faire fonctionner ThingSpeak pour les capteurs, nous avons réussi à visualiser les données des capteurs des weather click avec Cayenne comme webhook. Toutefois cette solution n'était pas très modulable car il semblait impossible de créer nos propres widgets et de récupérer les données pour les envoyer vers l'application développée par les étudiants. Nous sommes donc repassés sur ThingSpeak, mais cette étape nous a permis de valider nos capteurs simplement avant de créer notre propre formatage.

L'application qui est couplée avec Cayenne est celle intitulée « stm32-test-modules ».





## Ecole Publique d'Ingénieurs en 3 ans

6 boulevard Maréchal Juin, CS 45053  
14050 CAEN cedex 04

