



UNIVERSITÉ  
TOULOUSE III  
PAUL SABATIER



Université  
de Toulouse

## Rapport écrit

### Projet Réalisation Système

Muletta Antoine  
Dedjeh Ove-Anselme

## **Table des matières**

### **1 Travaux Pratiques de base et description de protocoles**

- 1.1 Le capteur de température et d'humidité DHT22
- 1.2 Le protocole One Wire
- 1.3 Le capteur ultrason HCSR04
- 1.4 Le protocole I2C

### **2. Projet BE**

- 2.1 Description du système
- 2.2 Le capteur de poids HX711
- 2.3 Le dispositif LoRa WAN
- 2.4 Le LoRaWAN en pratique

## 1.1 Le capteur de température et d'humidité DHT22

### A. Description

Le DHT22 est un capteur numérique de base à faible coût permettant de mesurer de manière efficace la température et l'humidité de l'air ambiant grâce à sa combinaison deux en un d'un capteur d'humidité capacitif et d'une thermistance.

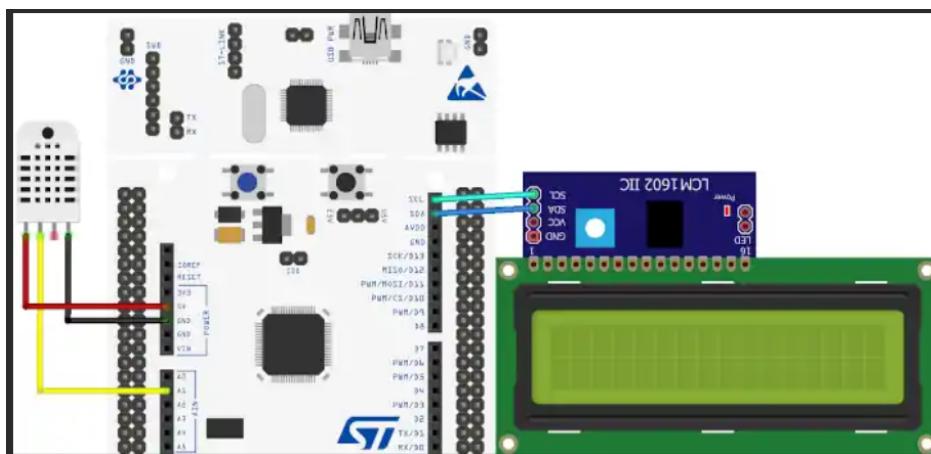
Il est capable de communiquer avec le microcontrôleur de notre choix grâce à une communication numérique sur un fil basé sur le protocole de communication one-wire développé par Dallas Instrument.

Le DHT22 est le capteur le plus précis de sa famille. Il permet de fonctionner une plus grande gamme de température et d'humidité, avec une plage de mesure de température comprise entre -40° et 80°.

La petite taille de ce capteur constitue aussi un très grand avantage, de par sa discrétion, lui permettant d'intégrer plusieurs types de projet.

Enfin, il s'agit d'un capteur de température numérique actif, c'est-à-dire qu'il est alimenté par 5 VDC. En effet, avec une tension d'alimentation de 3.3V le capteur ne fonctionne pas. Par ailleurs, on a remarqué que lorsque le capteur est alimenté, il ne faut pas envoyer d'instruction au capteur dans la seconde qui suit au risque d'avoir une instabilité sur le signal. On ajoute donc un délai d'une seconde dans notre code pour passer l'état instable.

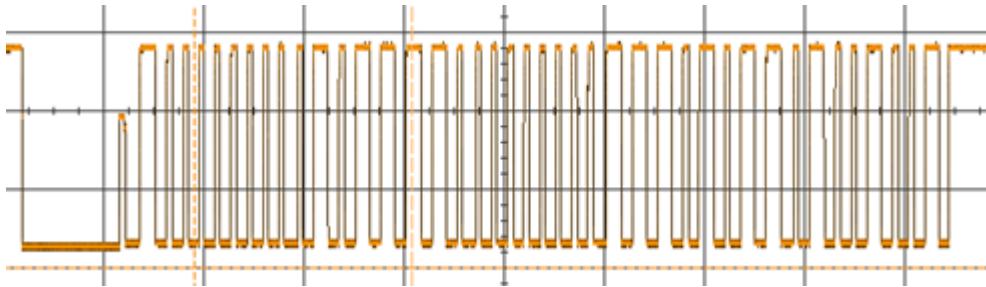
### B. Schéma de câblage



## C. Fonctionnement

Dans le cadre de notre travail, nous avons travaillé avec un STM32L476RG. L'acquisition des données par le DHT22 nécessite l'envoi d'un signal de démarrage par le microcontrôleur. En effet STM32 envoie le signal de démarrage, le capteur passe de l'état d'attente à l'état de fonctionnement. A la fin du signal de démarrage, le DHT22 enverra un signal de réponse. Ce signal de réponse est composé de 40 bits qui s'organisent de la manière suivante :

**DATA = 16 bits RH data + 16 bits Temperature data + 8 bits**



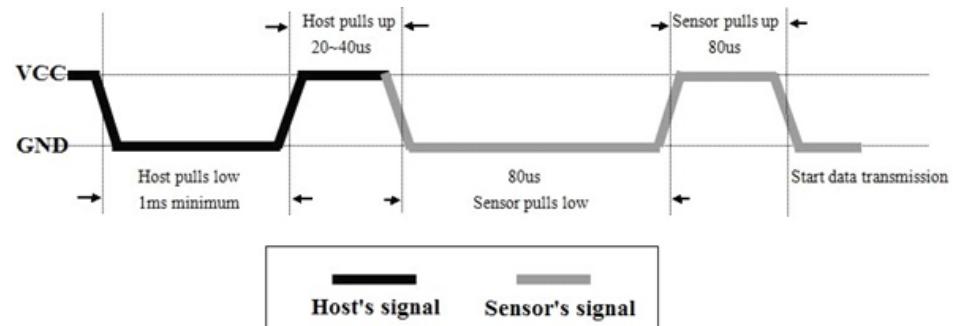
Trame pour une température de 21.5°C à 9.5% d'humidité (500µs/div)

En l'absence de ce signal, le DHT22 n'enverra pas de lui-même les données au microcontrôleur. En effet, le DHT22 est designé pour se mettre en veille à chaque fois qu'il n'est pas sollicité ce qui lui permet d'avoir une basse consommation d'énergie lorsque la collecte des données est terminée. La durée de cette étape doit être supérieure à deux secondes.

### a. Le démarrage

Le MCU (Microcontroller Unit) envoie un signal de démarrage au DHT22 (Start), celui-ci envoie en retour un signal de réponse. Si le bus est libre, un "1" logique est envoyé. Lorsque la communication entre le capteur et la STM32 commence, le bus de données passe à 0 logique pendant une durée comprise entre 1 ms à 10 ms. Ce délai permet de s'assurer que le capteur détecte bien le signal de la STM32. Enfin cette dernière va appliquer un 1 logique et attendre entre 20 et 40µs la réponse du capteur. De cette façon, chacun des participants s'assure d'avoir le champ libre pour communiquer. Pour finir, le capteur envoie un "0" logique sur le bus pendant 80µs en guise de signal de

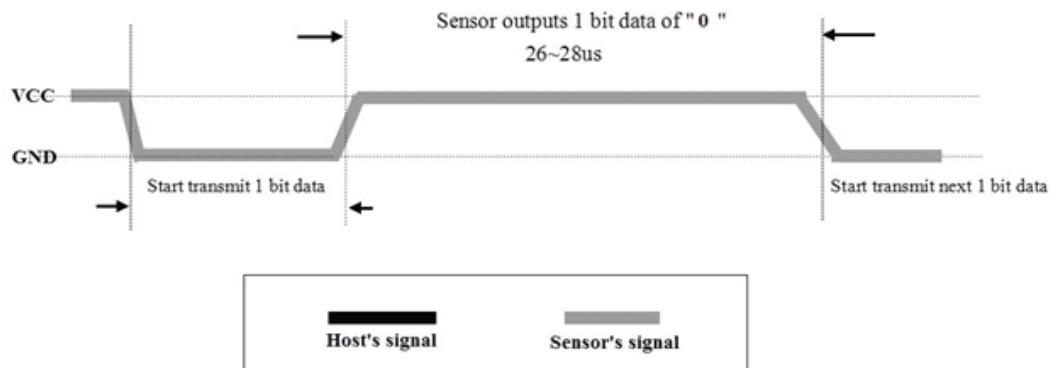
réponse, puis à 1 logique pendant 80 $\mu$ s pour se préparer à envoyer des données.



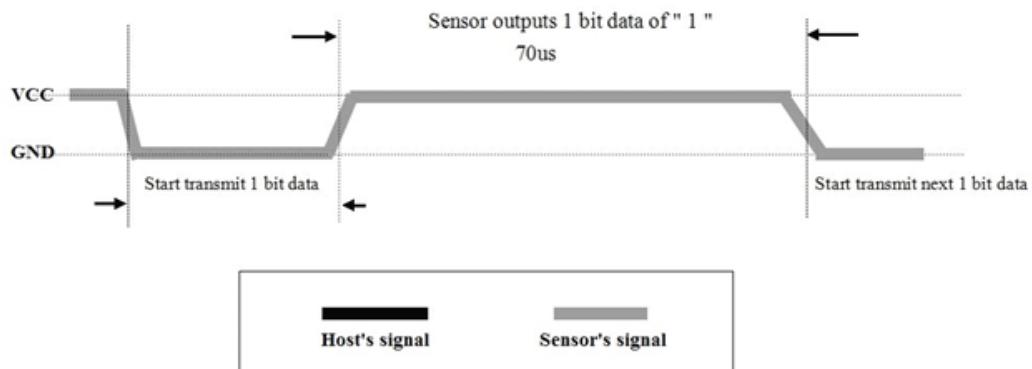
### b. La récupération des données

Lorsque le DHT22 envoie des données, la transmission de chaque bit commence par un 0 logique de 50 $\mu$ s, puis la longueur du signal de niveau de tension haut suivante est définie comme suit :

- si le bit est à "1" le signal aura une durée d'environ 70 $\mu$ s
- si le bit est à "0" le signal aura une durée entre 26 et 28 $\mu$ s

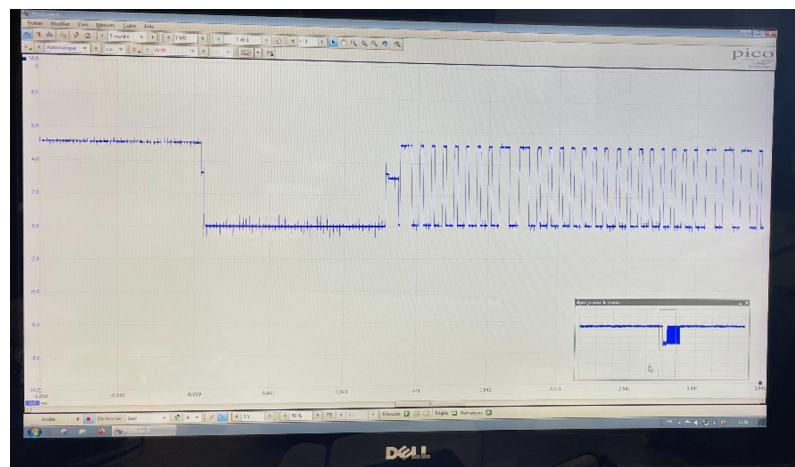
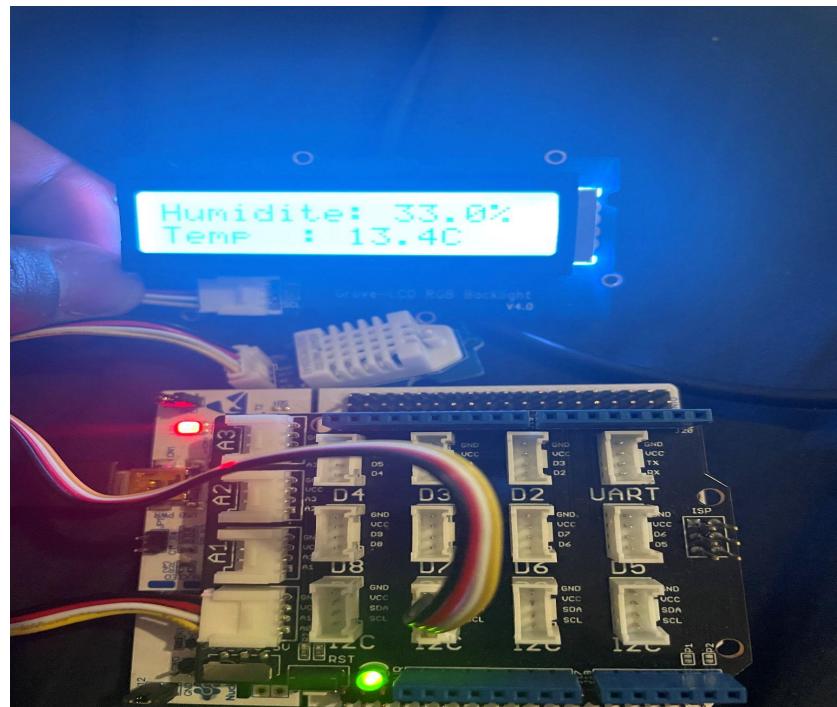


### Représentation du 0 logique



### Représentation du 1 logique

## D. Résultat en pratique



Trame d'une acquisition de données du DHT22

## 1.2 Le protocole One Wire

Le bus 1-Wire (ou OneWire) est un bus conçu par Dallas Semiconductor. Ce protocole fonctionne suivant le principe maître / esclave.

L'avantage de ce bus est qu'il peut être utilisé en mode « parasite » c'est-à-dire que l'alimentation se fait à partir du fil de données. Cela permet d'utiliser seulement 2 fils, un fil de données et un fil de masse.

La communication sur le bus 1-wire est caractérisée par un ensemble de pulse.

L'état par défaut de la ligne data est +5V, ce qui permet d'alimenter les différents composants à partir de la ligne data en mode parasite.

Avant toute communication, le maître met le bus à 0 pendant 480us pour faire un reset des composants connectés.

Le maître reçoit alors la liste des esclaves connectés sur le bus. Il pourra utiliser une commande particulière pour sélectionner l'esclave avec lequel il souhaite communiquer.

Pour émettre un bit sur le bus le maître force le bus à « 0 » pendant 1 à 15us, pour indiquer qu'il souhaite envoyer un bit de données, puis il doit positionner le bus suivant le bit qui doit être émis (0 ou 1) l'esclave « actif » va lire le bus entre 15us et 45us après la détection du front descendant initial.

La durée total d'un bit est donc de 60us, ce qui donne une vitesse de communication maximale de 16 Kbits/s.

Après la réception d'une commande du maître, l'esclave peut renvoyer des données.

Pour lire les données de l'esclave, le maître force le bus à 0 pendant 1us, et il lit ensuite l'état du bus entre 5 et 15us après ce front descendant.

Si l'esclave souhaite émettre un bit « 1 », il laisse le bus à « 1 », sinon il tire le bus à « 0 » pendant 15us.

## 1.3 Le capteur ultrason HCSR04

### A) Description

Le capteur HC-SR04 est un capteur actif utilisant l'I2C comme protocole de communication.

C'est donc un capteur émetteur et récepteur qui utilise les ultrasons pour déterminer la distance d'un objet. Il offre une plage de mesure allant de 2cm à 400cm.

Il est composé de quatre broches.

Tout d'abord, il est composé d'une broche d'alimentation en 5V et d'une broche pour la masse.

Ensuite il y a une broche Trigger Input qui est l'entrée de déclenchement des mesures.

Et pour finir, on retrouve la broche Echo Output qui est la sortie renvoyant les données mesurées.

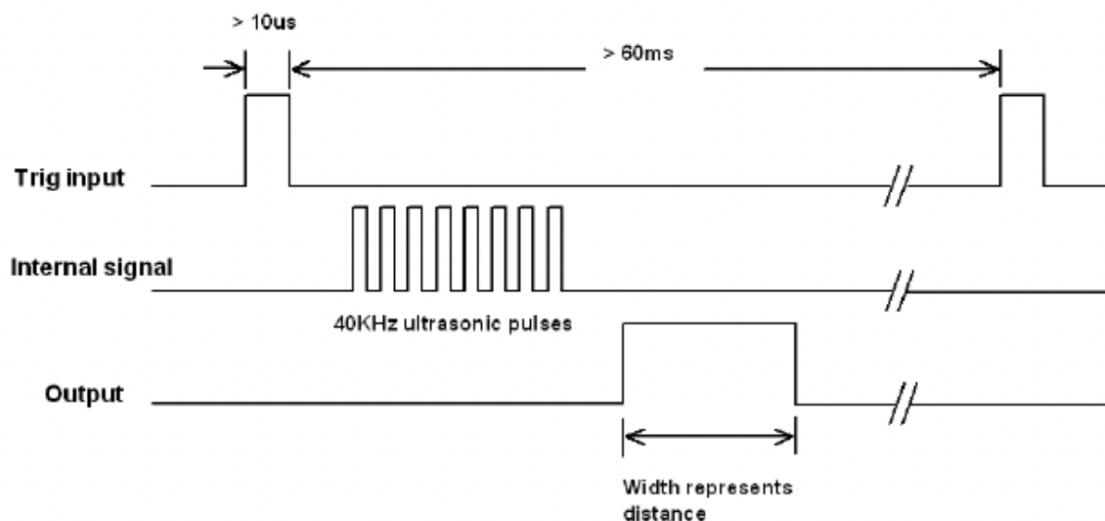
### B) Fonctionnement

D'après la datasheet, pour déclencher une mesure, il faut présenter une impulsion haute (5 V) d'au moins 10  $\mu$ s sur l'entrée Trig.

Le capteur émet alors une série de 8 impulsions ultrasoniques à 40 kHz, puis il attend le signal réfléchi.

Lorsque celui-ci est détecté, il envoie un signal haut sur la sortie Echo, dont la durée est proportionnelle à la distance mesurée.

Puis l'opération est répétée toutes les, au moins, 60 ms.



La distance parcourue par un son se calcule en multipliant la vitesse du son, environ 340 m/s (ou 34 000 cm/1 000 000 µs) par le temps de propagation, soit :

$$\text{distance} = \text{vitesse} * \text{temps}$$

Le HC-SR04 donne une durée d'impulsion en dizaines de µs. Il faut donc multiplier la valeur obtenue par 10 µs pour obtenir le temps t. On sait aussi que le son fait un aller-retour. La distance vaut donc la moitié:

$$\text{distance} = 34 000 \text{ cm}/1 000 000 \mu\text{s} * 10\mu\text{s} * \text{valeur} / 2$$

ou en simplifiant:

$$\text{distance} = 17/100 \text{ cm} * \text{valeur}$$

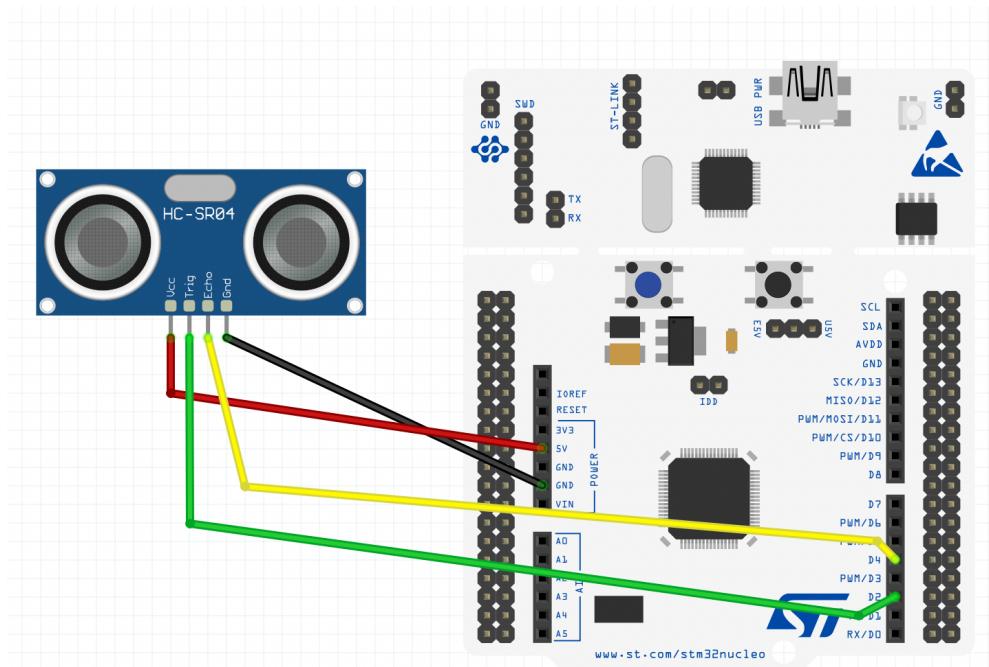
17/1000 est égale à 1/58.8235 c'est pourquoi dans la datasheet du HC-SR04, on retrouve la formule:

$$\text{distance} = \text{valeur} / 58.$$

### C) Résultat en pratique

Nous avons constaté lors des essais qu'il y avait régulièrement un écart d'un à deux centimètres pour les courtes distances et des erreurs allant jusqu'à cinq centimètres pour les distances élevées

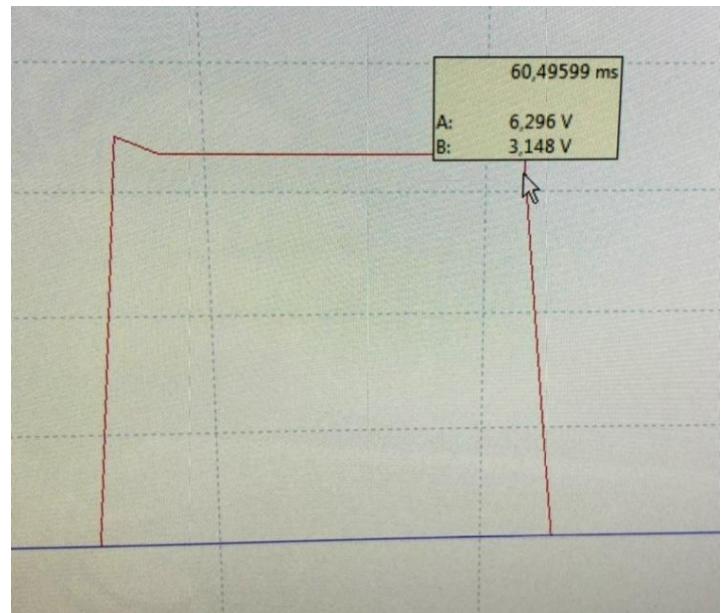
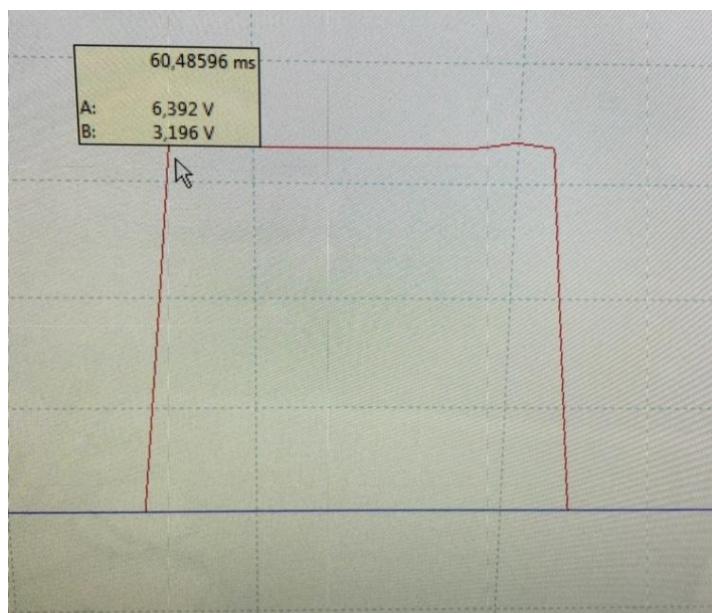
Voici donc, d'après la description, le schéma de câblage du capteur.



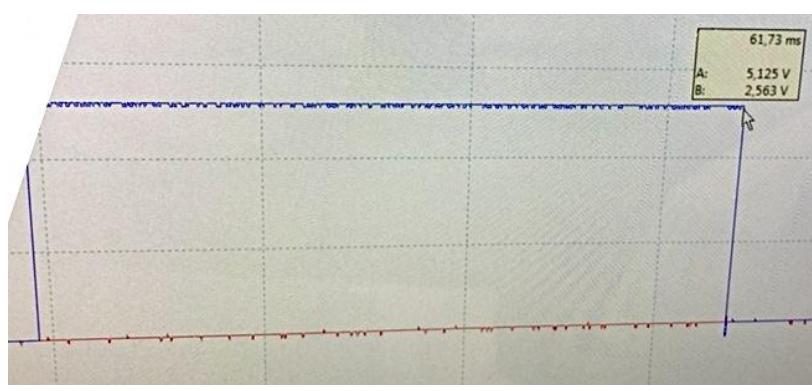
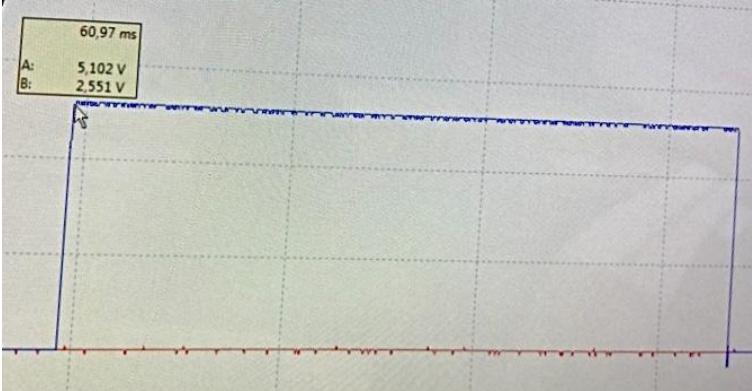
Nous avons également récupéré à l'aide d'un oscilloscope les trames de données:



En rouge le signal Trig qui émet une impulsion haute d'environ 10 us:



En bleu le signal Echo que l'on récupère.



Ici on a donc une durée de  $61,73 - 60,97 = 0,76 = 760$  us  
Ce qui nous donne une distance  $= 17/100 * 760 = 129$  cm

## 1.4 Le protocole I2C

L'I2C est un protocole de bus de communication développé par Philips Semiconductor. Il s'agit d'un protocole lent mais simple et robuste.

C'est un mode de communication synchrone Tx/Rx.

Il ne faut que 2 fils pour communiquer entre différents appareils : SCL (horloge) et SDA (données).

Pour communiquer avec un esclave, le maître envoie d'abord l'octet d'adresse sur la ligne de données. Tous les esclaves connectés au bus font correspondre leur adresse à celle envoyée par le maître. En cas de correspondance, l'esclave concerné envoie l'ACK au maître (ACKnowledgement qui agit comme une sorte d'accusé de réception) et la communication peut commencer. La conversation entre le maître et l'esclave se déroulent comme suit :

- Le maître transmet la condition START ( SDA -> 0 logique ). SCL reste 1 logique.
- Le maître transmet une adresse de 7/10 bits avec un bit R/W (lecture/écriture) pour indiquer le type d'opération.
- Les esclaves commencent à comparer l'adresse transmise avec la leur. En cas de correspondance, l'esclave envoie l'ACK.
- Si l'opération est une écriture, le maître commence à transmettre des octets de données à l'esclave. Après chaque octet, l'esclave envoie l'ACK pour confirmer les données reçues.
- Si l'opération est une lecture, l'esclave commence à transmettre des données au maître et après chaque octet reçu, le maître envoie l'ACK à l'esclave pour confirmer les données reçues.
- Après la fin du transfert, le maître envoie le signal d'arrêt (SDA -> 1 logique) et la transmission s'arrête.

L'avantage du bus I2C, c'est que l'adresse de l'esclave fait 7 bits (ou 10 bits), on peut donc connecter 128 esclaves (respectivement 1024 esclaves) avec un seul maître.

## **2. Projet BE**

### **2.1 Description du système**

Notre projet a pour objectif de concevoir une ruche connectée. En effet, le but est de retourner, à l'aide de modules LoRaWAN, les informations à l'intérieur de la ruche grâce à différents capteurs dont notamment un capteur de poids HX711 et un capteur de température et d'humidité DHT22.

Le capteur de poids permettrait par exemple de savoir quand récupérer le miel. En effet, lorsque environ 80% des cadres des ruches sont recouverts de cire par les abeilles, il faut récupérer le miel. On pourrait alors déterminer le poids de la cire lorsqu'elle aurait recouvert cette surface là pour ensuite envoyer une information à l'apiculteur via les modules LoRaWAN.

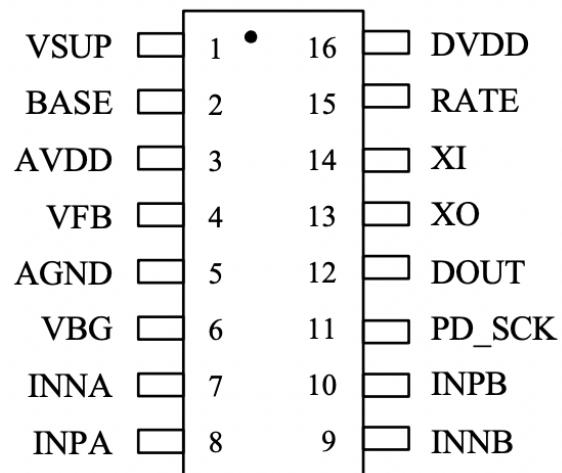
De même, lors de l'hivernage des ruches, il est important de protéger ces dernières contre l'humidité car les abeilles supportent mieux le froid que l'humidité. On pourrait donc intégrer un système de ventilation au sein de la ruche pour éviter la condensation.

## 2.2 Le capteur de poids HX711

### A) Description

Le capteur HX711 est un convertisseur analogique numérique 24 bits conçu pour peser des objets sur une plage de valeur allant de 0Kg à 10Kg.

Ce capteur utilise un protocole de communication I2C et il est composé d'un amplificateur de gain programmable et de deux canaux d'entrées permettant de choisir parmi des gains de 32 (avec le canal B), 64 ou 128 (avec le canal A).



7: Entrée négative du canal A

9: Entrée négative du canal B

8: Entrée positive du canal A

10: Entrée positive du canal B

11: Entrée permettant de créer le signal d'horloge pour le protocole de communication I2C.

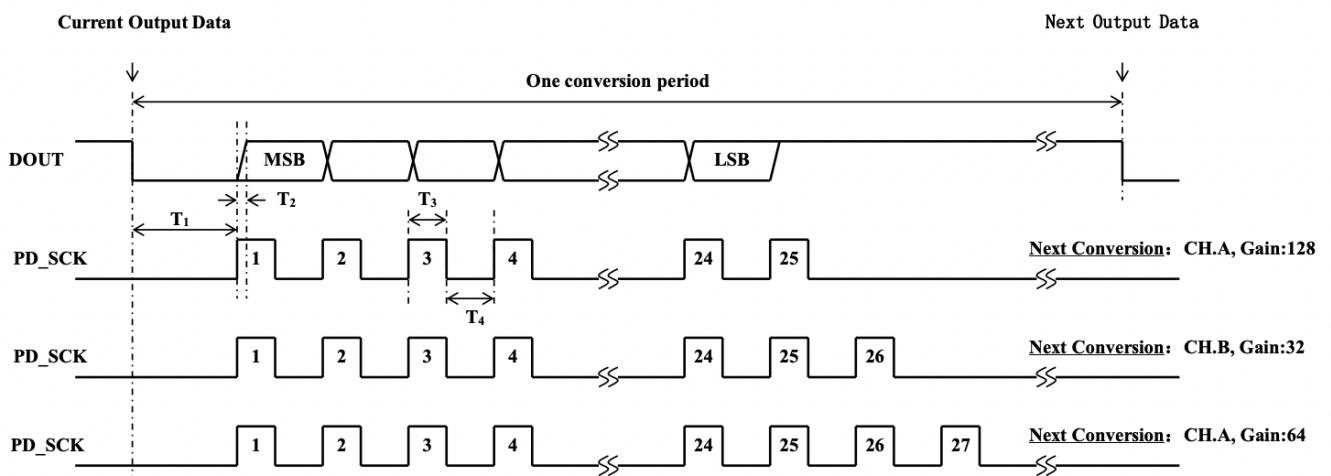
12: Sortie permettant de récupérer les informations du capteur de poids

## B) Fonctionnement

Nous utilisons les pins PD\_SCK et DOUT pour la récupération des données, la sélection des entrées et la sélection du gain.

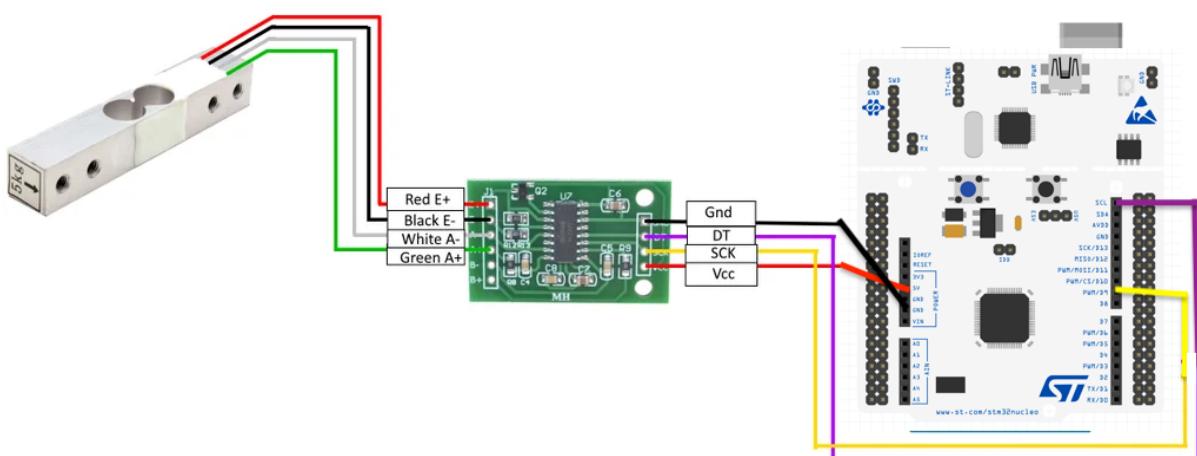
Au lancement du programme, le signal d'horloge (PD\_SCK) est à l'état bas tandis que le signal de récupération de données (DOUT) est à l'état haut. Lorsque la réception de données est prête, le signal de récupération de données passe à l'état bas.

Ensuite selon le gain que l'on souhaite appliquer, on émet entre 25 bits et 27 bits d'impulsion d'horloge durant lesquels nous récupérons 24 bits de données (du poids le plus fort MSB, vers le poids le plus faible, LSB). A la fin de ce signal d'horloge, le signal de récupération de données (DOUT) repasse à l'état haut.

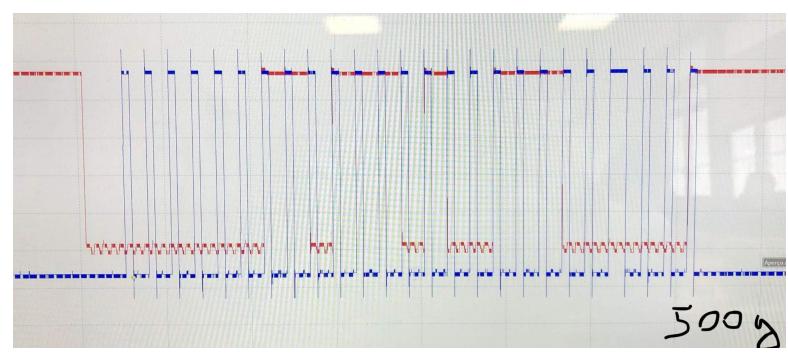
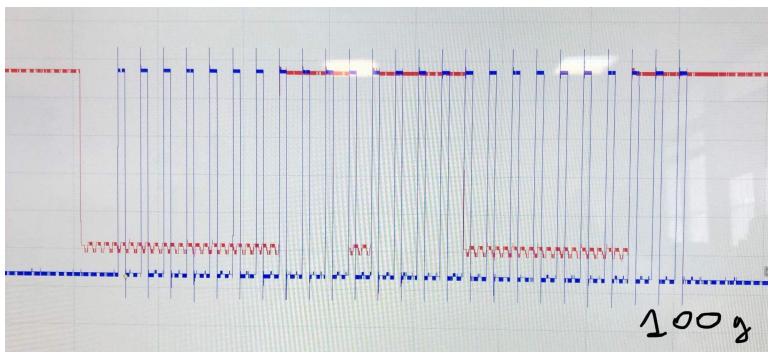
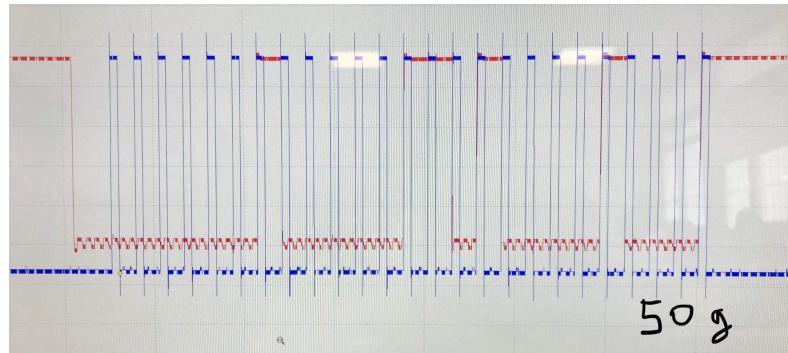
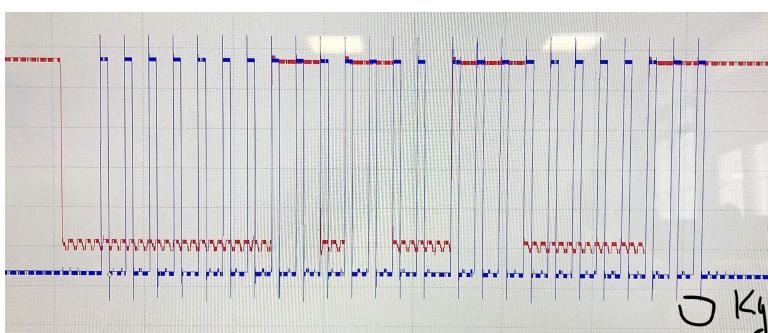


### C) Résultat en pratique

Voici donc, d'après la description, le schéma de câblage du capteur.



Nous avons essayé dans un premier temps grâce à un programme arduino et à un oscilloscope de décoder les trames de données du capteur pour différents poids.

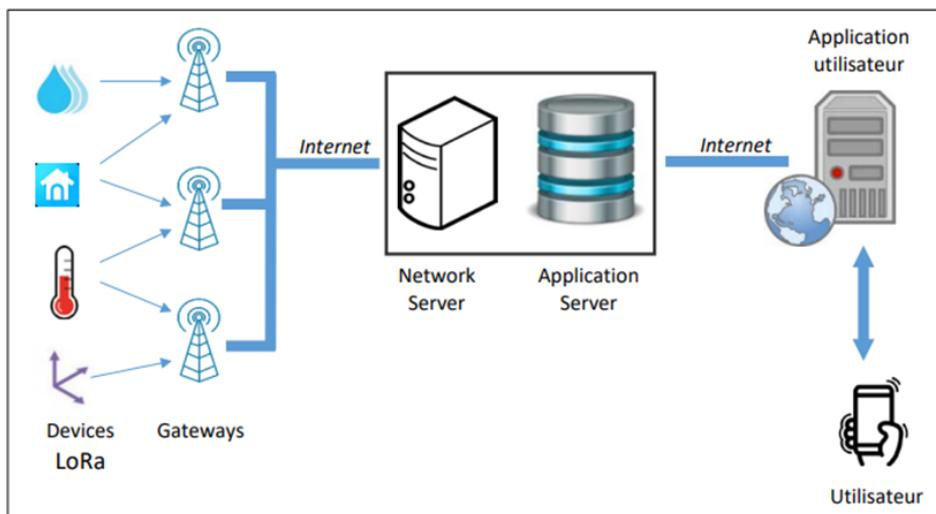


Mais les mesures n'étant pas précises et n'arrivant pas à conclure sur la façon dont le codage était fait, nous avons alors cherché une bibliothèque déjà faite pour ce capteur.

Cependant ce capteur étant assez récent, et par faute de temps, nous n'avons pas réussi à obtenir un programme fonctionnel.

## 2.3 Le protocole LoRaWan

Mis en place par la LoRa Alliance, LoRaWAN est un protocole de communication dédié à l'internet des objets. Ses principales caractéristiques sont d'être peu coûteux à mettre en place et d'avoir une faible consommation énergétique. Son principal avantage est de permettre à des équipements de communiquer sur de longues distances, à savoir plusieurs kilomètres, tout en assurant une autonomie conséquente à ces équipements. D'un point de vue électronique, la certification s'appuie sur une technique de modulation par étalement de spectre appelée LoRa. Elle garantit une absence d'interférence entre des communications ayant des débits différents, ce qui permet à une station de base de gérer un millier d'équipements en théorie. L'une des forces de LoRaWAN provient de ce paramètre et surtout du fait que cela est géré automatiquement par le serveur réseau par un procédé de « débit adaptatif » (ADR). On optimise la durabilité de chaque équipement ainsi que la capacité du réseau en choisissant le débit idéal.



Architecture réseau LoRaWAN

LoRaWAN est un protocole de communication longue distance consommant peu d'énergie. On peut comparer cette technologie avec la technologie WiFi, une technologie de communication sans fil. La WiFi consomme en moyenne 140mA pour une communication fonctionnant sur une distance de 100m, alors que la technologie LoRa ne consomme en moyenne que 40mA pour une communication fonctionnant sur une distance pouvant aller jusqu'à 15Km en condition idéale. En zone urbaine, comme par exemple l'école, le rayon de communication est d'environ 3Km. L'inconvénient de la

technologie LoRa et qu'en dépit de la distance d'envoi, la quantité de données envoyée et le débit est faible.

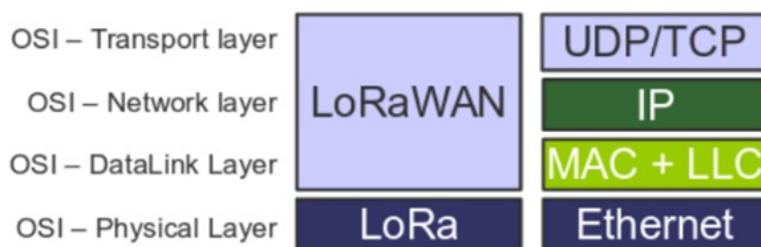
Les Ends-devices, ce sont généralement des cartes électroniques, tel que des cartes Arduino ou encore des STM32, qui sont associées avec des modules LoRa et un ou plusieurs capteurs. La gateway LoRa, passerelles permettant de recevoir les données envoyées par tout autre équipement LoRa, transfère ces dernières sur un système de raccordement.

Les passerelles LoRa sont connectées au serveur du réseau en utilisant des connexions IP standard. Le serveur du réseau LoRa gère le flux de données arrivant des gateways. Le serveur permet d'éliminer les paquets dupliqués et adapte aussi le débit des données.

La technologie LoRa peut être une liaison bidirectionnelle, elle utilise une modulation par étalonnage de spectre, qui font qu'un signal généré avec une certaine bande-passante est étalé sur le spectre de fréquence, cela permet de :

- Sécuriser les communications.
- Une plus grande résistance aux interférences et aux bruits.
- Limiter la consommation.

L'image suivante compare la technologie LoRa et son protocole LoRaWAN avec la technologie WiFi : Sur cette image nous pouvons voir que pour la technologie LoRa il y a deux couches, la couche physique



Les couches des deux protocoles

(LoRa) et la couche MAC (LoRaWAN) qui regroupe la gestion des données, du réseau et du transport de données.

La couche physique configure et gère tous les aspects du signal transmis entre les ends-devices et la gateway LoRa. C'est-à-dire qu'elle gère les fréquences, le mode de modulation, les niveaux de puissance et les largeurs de bande. LoRa utilise une forme de modulation à étalement de spectre, elle permet de rendre le signal moins sensible aux fluctuations sélectives en fréquences. Le signal est transmis sur une bande de fréquences beaucoup plus large que nécessaire.

La modulation peut être modifiée en FSK ou LoRa. Le Spreading Factor (SF- Facteur d'étalement) représente la longueur du code envoyé. Pour éviter les interférences, les codes d'étalements sont or- thogonaux, cela permet la transmission simultanée de plusieurs canaux, chacun étant étalé en temps et en fréquence.

Le tableau suivant représente la plage des valeurs accessible avec un module LoRa quelconque :

Spreading Factor	Chips/Symbol	LoRa demodulator SNR	LoRa MAC (Class compatibility)
6	64	-5 dB	No
7	128	-7.5 dB	Yes
8	256	-10 dB	
9	512	-12.5 dB	
10	1024	-15 dB	
11	2048	-17.5 dB	
12	4096	-20 dB	

Tableau des différentes valeurs du SF et du SNR

Nous pouvons voir ici que plus le SF est important, plus le SNR est faible, la capacité du récepteur LoRa à recevoir des signaux avec un SNR négatif augmente sa sensibilité et son bilan de liaison. Le débit des données étant assez faible, plusieurs bandes passantes sont nécessaires aux multiples communications. On retrouve celles comprises entre 7.8kHz et 500kHz (elles ont des valeurs précises), les deux bandes passantes les plus utilisées sont 125kHz et 500kHz. La puissance utilisée pour l'envoie d'un message est adaptative, c'est-à-dire qu'elle dépend de la taille des données, du débit d'envoi, la portée d'émission... Un algorithme est donc utilisé pour déterminer

le niveau de puissance requis pour l'envoie du message. L'adaptabilité de la puissance de transmission permet donc de maximiser la durée de vie de la batterie. Ces communications, entre les divers dispositifs, utilisent des canaux de fréquences différents, et utilisent des débits de données distincts. Donc la technologie à étalement de spectre permet la communication avec des débits de données divers et garantie que les communications se déroulent sans interférences. Ce protocole utilisé par la technologie LoRa a été conçu pour fonctionner de façon optimale avec des dispositifs alimentés par batterie, mais aussi sans fil, qui nécessite une connexion à faible coût.

LoRaWAN utilise une topologie de réseau en étoiles, c'est-à-dire que les ends-devices échangent avec la gateway. Les ends-devices sont regroupés dans trois classes de dispositifs (les classe A,B et C), la classe du dispositif change en fonction du types de communications nécessaires. La solution qui pour le moment est utilisée dans la plupart des cas est la classe A car les deux autres classes ne sont implémentées que sur quelques modules spécifiques.

La classe A est une implémentation d'end-devices qui utilise, du point de vue de l'utilisateur, des communications unidirectionnelles. L'utilisateur peut recevoir des données de ces dispositifs embarqués mais ne peut pas piloter les dispositifs à distance. Mais, les dispositifs qui communiquent avec la gateway reçoivent et traitent des requêtes de la gateway, ces requêtes ont pour but d'ajuster le débit, l'étalement du spectre et la puissance d'envoi du module. Elles permettent le maintien d'une bonne connexion entre les dispositifs et la gateway.

Il existe deux mode d'activation principale dans le protocole LoRa: l'Activation By PersonalizationP) et l'Over The Air Activation (OTAA)

- **ABP**

Ce mode de connexion utilise six informations qui permettent d'identifier et de permettre la connexion d'un module. Ces informations sont :

- AppEUI, cet identifiant est unique à l'application, il permet de regrouper les modules LoRa. Cette adresse est sur 64 bits, il permet de classer les périphériques par application, ce paramètre est bien sûr modifiable sur la plupart des modules vendus sur le marché.
- DevEUI, cet identifiant rend chaque module unique. Cet identifiant est programmé en usine, et donc n'est normalement pas modifiable.

- AppKey, cet identifiant secret qui n'est partagé qu'entre le module et la gateway. Cet identifiant est utilisé pour calculer les clefs de chiffrement utilisées pour les communications entre le module et la gateway.

Il y aussi les clés de chiffrement qui sont préprogrammées dans ce mode, l'utilisateur les fixe et elles ne sont donc pas calculables. Ces clefs sont les suivantes :

- DevAddr, cet identifiant est l'adresse du module, il sert à l'identification du module dans le réseau.
- NetSKey, cet identifiant est une clef de chiffrement entre le module et la gateway qui est utilisé pour les transmissions et valider l'intégrité des messages envoyés.
- AppSKey, cet identifiant est aussi une clef de chiffrement entre le module et la gateway qui est utilisé pour les transmissions et valider l'intégrité des messages envoyés.

Ce mode de connexion possède un avantage et un inconvénient, l'inconvénient de ce mode est que les clefs de chiffrement permettant les communications avec le réseau sont pré-configurées dans l'objet ce qui rend la sécurité plus faible. L'avantage de ce mode est que le raccordement au réseau est très simple ce qui rend l'objet rapidement opérationnel.

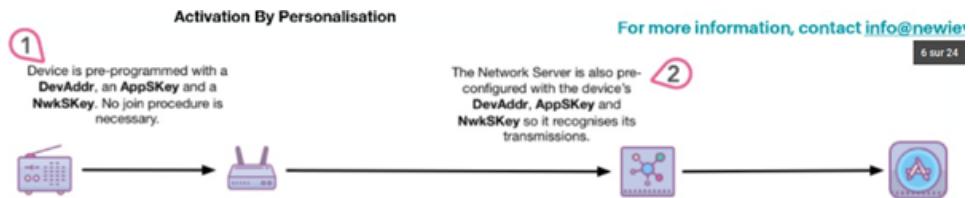


Schéma explicatif du mode ABP

### • OTAA

Ce mode de connexion utilise six informations qui permettent d'identifier et de permettre la connexion d'un module. Ces informations sont :

- AppEUI, cet identifiant est unique à l'application, il permet de regrouper les modules LoRa. Cette adresse est sur 64 bits, il permet de classer les périphériques par application, ce paramètre est bien sûr modifiable sur la plupart des modules vendus sur le marché.
- DevEUI, cet identifiant rend chaque module unique. Il est programmé en usine, et donc n'est normalement pas modifiable.

- AppKey, cet identifiant secret n'est partagé qu'entre le module et la gateway. Cet identifiant est utilisé pour calculer les clefs de chiffrement utilisées pour les communications entre le module et la gateway.

Il y aussi les clés de chiffrement qui sont partagées lors d'un début de communication. Si l'objet est autorisé à rejoindre le réseau, le module et la gateway vont échanger des clefs de chiffrement qui seront propre à cette communication, la gateway va donc attribuer ces clefs de chiffrement au module, pour la suite de la communication avec le module. Ces clefs sont les suivantes :

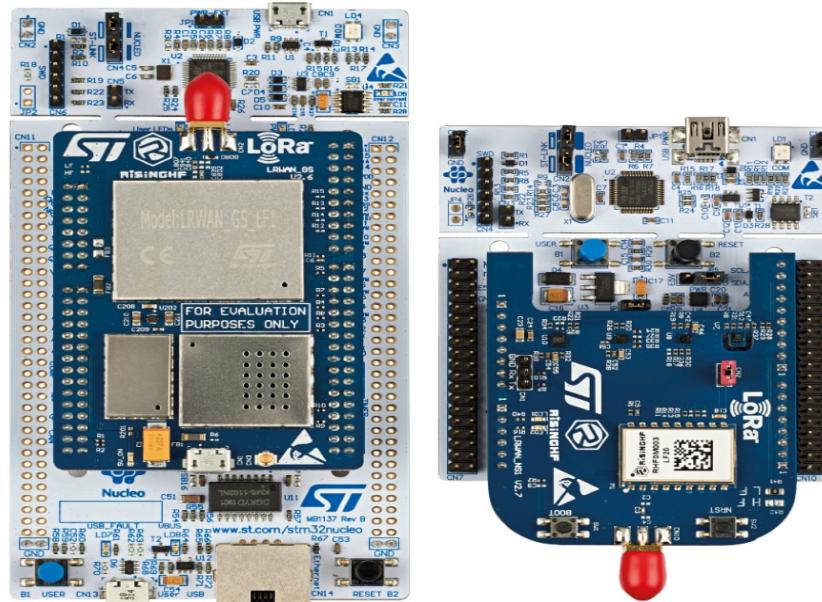
- DevAddr, cet identifiant est l'adresse du module, il sert à l'identification du module dans le réseau.
- NetSKey, cet identifiant est une des clefs de chiffrement entre le module et la gateway qui est utilisé pour les transmissions et valider l'intégrité des messages envoyés.
- AppSKey, cet identifiant est aussi une des clefs de chiffrement entre le module et la gateway qui est utilisé pour les transmissions et valider l'intégrité des messages envoyés.

Ce mode de connexion possède un avantage et un inconvénient, l'inconvénient de ce mode est que l'objet doit implémenter ce mécanisme pour effectuer des communications avec le réseau, ce qui introduit une complexité supplémentaire. L'avantage de ce mode est que la création des clefs de chiffrement est effectué par la gateway, ce qui renforce la sécurité des communication effectué sur le réseau. Ce mode est donc le mode le plus utilisé dans le mode de l'IoT, en ce qui concerne les réseaux LoRaWAN car elle est plus sécurisée.

En ce qui concerne la bande de fréquence, dans le cadre de notre BE on a choisi les bandes de 433 Mhz Europe.

## 2.4 Le LoRaWAN en pratique

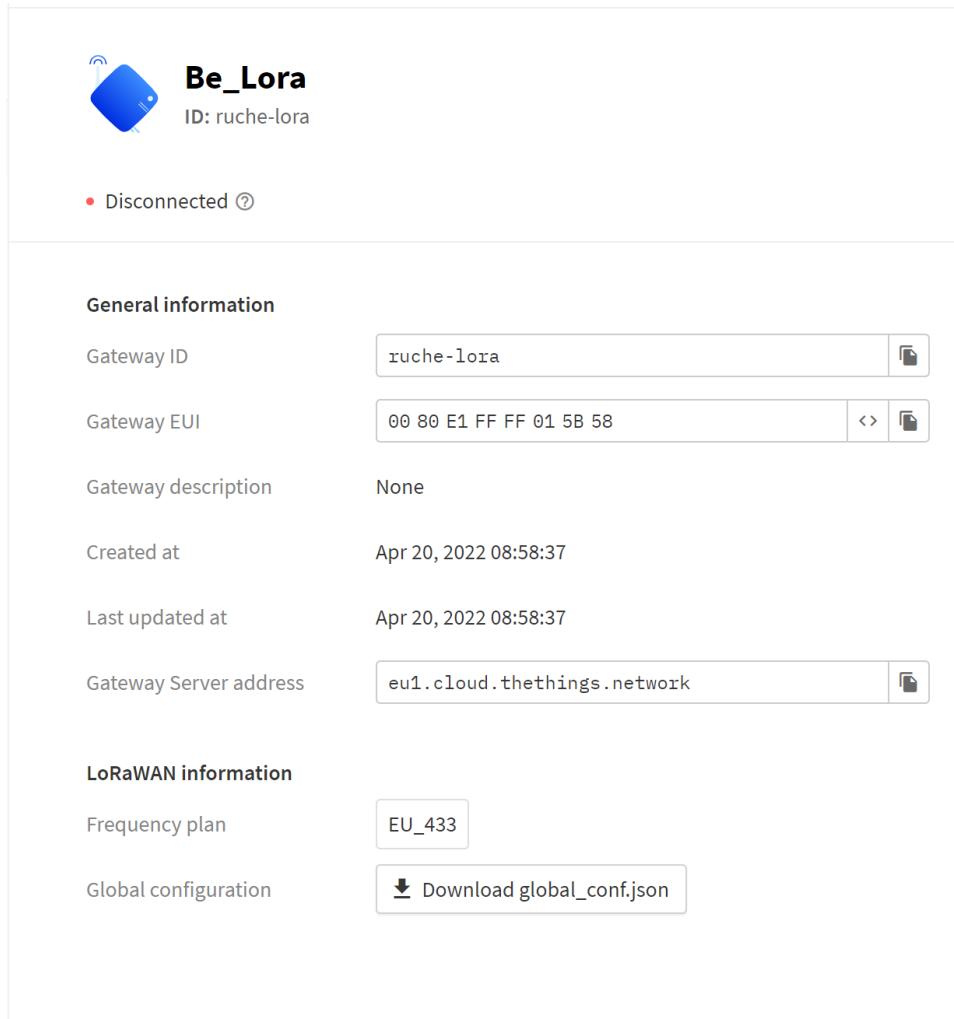
Nous avons travaillé sur le starter pack de STM32 Nucleo appelé le P-NUCLEO-LRWAN3.



The serial number of the NUCLEO-L073RZ MB1136 reference board is indicated on a sticker under the MB1136.  
If the number is within the range from A191400001 to A191402004, the board must be updated with a new firmware before use. Please download the last firmware version available at [www.st.com/i-cube-lrwan](http://www.st.com/i-cube-lrwan)

Il est constitué d'un gateway basé sur un NUCLEO-F746ZG. Ce gateway est muni d'une antenne de 434 MHZ. En ce qui concerne le end-nodes, on a un sensor device embarquant des capteur de température et d'humidité, ou encore de pression.

En ce qui concerne le travail fourni sur ce composant, il est conforme à celui fait par Aurélien BENOIT et guillaume LE GOFF l'année scolaire dernière. Mais nous avons essayé de trouver une nouvelle approche pour faire fonctionner le système. Pour ce faire, nous avons opté pour la couche applicative de The ThingNetwork (TTN) qui fournit une bande de travail de 433 MHz en Europe. Ce qui nous a permis de connecter le gateway au réseau et même d'envoyer des uplinks (uplinks envoyé automatiquement avec la configuration par défaut du gateway).



The screenshot shows a device management interface for a LoRa sensor. At the top, there's a blue icon of a cube with a signal wave, followed by the text "Be\_Lora" and "ID: ruche-lora". Below this, a red dot indicates the device is "Disconnected".

**General information**

Gateway ID	ruche-lora	
Gateway EUI	00 80 E1 FF FF 01 5B 58	
Gateway description	None	
Created at	Apr 20, 2022 08:58:37	
Last updated at	Apr 20, 2022 08:58:37	
Gateway Server address	eu1.cloud.thethings.network	

**LoRaWAN information**

Frequency plan	EU_433
Global configuration	Download global_conf.json

La tâche à été plus compliqué avec le LoRa sensor device que nous n'avons pas réussi à faire fonctionner avec le Firmware qui à été fourni par ST pour la configuration du end node. Nous n'avons malheureusement pas pu déceler l'origine de ce dysfonctionnement. Face à l'incapacité d'avancer sur notre projet, nous avons tenté de commander un module LoRa E5 que nous n'avons pas pu réceptionner à ce jour. La dernière piste que nous avons exploré est celle du LoRa Shield V1.4 de dragino qui est compatible au STM32 L476RG et qui est fonctionnel du moment qu'on met en place notre propre gateway à l'aide d'un Raspberry pi par exemple. Par manque de temps nous n'avons pas pu explorer cette solution en profondeur. Mais si ce projet devait être repris dans les années à venir, nous encourageons vivement nos camarades qui travailleront dessus à prendre cette idée comme point de départ de leur travail.