

BTS SN option IR

Systèmes Numériques option Informatique et Réseaux

E6.2 – PROJET INFORMATIQUE

Dossier de réalisation du projet.

<u>Académies : Besançon-Dijon</u>	<u>Session : 2022</u>
<u>Lycée ou Centre de formation :</u> Lycée Gustave Eiffel	
<u>Ville :</u> Dijon	
<u>Nom du projet :</u> Plateforme IoT	
<u>Projet industriel :</u> OUI	

Équipe de développement.

<u>Professeur :</u>	<u>Joël Moutoussamy, Pascal Dub</u>
<u>L'entreprise :</u>	<u>EURL Tenum et Lycée Eiffel</u>
<u>Interlocuteur(s) de l'entreprise.</u>	<u>Frédéric BOUCHAR</u>
<u>Etudiant 1 :</u>	<u>Couvert Antoine</u>
<u>Etudiant 2 :</u>	<u>Pogossian Artur</u>
<u>Etudiant 3 :</u>	<u>ROUX Mathieu</u>

Rapport de projet :

Partie Générale :

Sommaire :

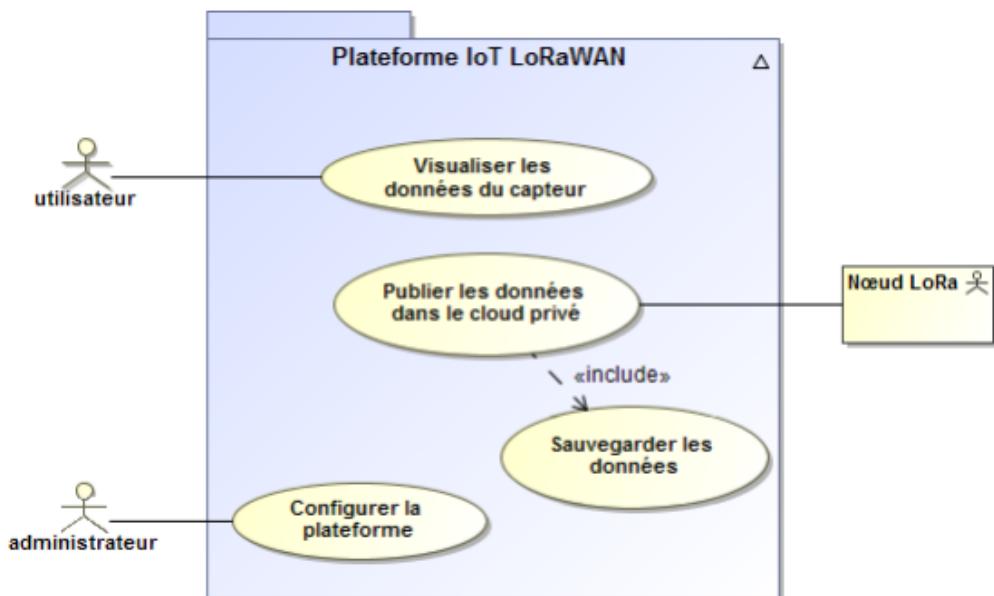
Partie Générale :	1
Sommaire :	1
Présentation du Projet	3
Répartition des tâches	4
Contraintes matérielles et logicielles	5
Présentation de la technologie LoRa	6
Structure d'un réseau LoRaWAN	6
Les avantages de LoRa	7
Étudiant 1 (COUVERT Antoine) :	9
Sommaire	9
Analyse du besoin	9
Prototypage et test	9
Description de la structure matérielle et logicielle	9
Description du fonctionnement actuel	9
Analyse du Besoin	9
Prototypage et Tests	10
Description de la Structure Matérielle et Logicielle	13
Description du Fonctionnement Actuel	15
Etudiant 2 (POGOSSIAN Artur) :	17
Présentation du Nœud CO2	17
Logiciels utilisés	18
Matériel utilisé	19
Description du système	20
Prototypage.	21
Installation / test du capteur (DHT22) température/humidité	22
Installation et test du capteur MICS-VZ-89TE	24
Installation et configuration du module RTC (DS3231) :	26
Installation et test du module radio RF LoRa Wan (RN2483A)	27
Configuration du nœud dans ChispStack	29
Conceptions détaillées	32
Diagrammes de séquences entre objet :	32
Création de la classe Cdht22	33
Création de la classe CMics	36

Création de la classe CLora	40
Présentation du programme principale	42
Explication en détail du programme principal final :	43
Difficultés rencontrées	43
Problème numéros 1	44
Problème numéros 2	44
Etudiant 3 (ROUX Mathieu) :	45
Expression du besoin	45
Prototypage	45
Conception Préliminaire	46
Passerelle LoRaWAN	46
Serveur Privé LoRaWAN	46
Serveur Application Web	46
Fonctionnement entre les objets	49
Conception détaillé	49
Passerelle LoRaWAN	49
Réseau LoRaWAN Privé	51
Installation de Chirpstack	51
Configuration de Chirpstack	52
Serveur Application Web	56
Test du système	66
Problèmes rencontrés	68
Message retenue en MQTT	68
Permission d'écriture	69

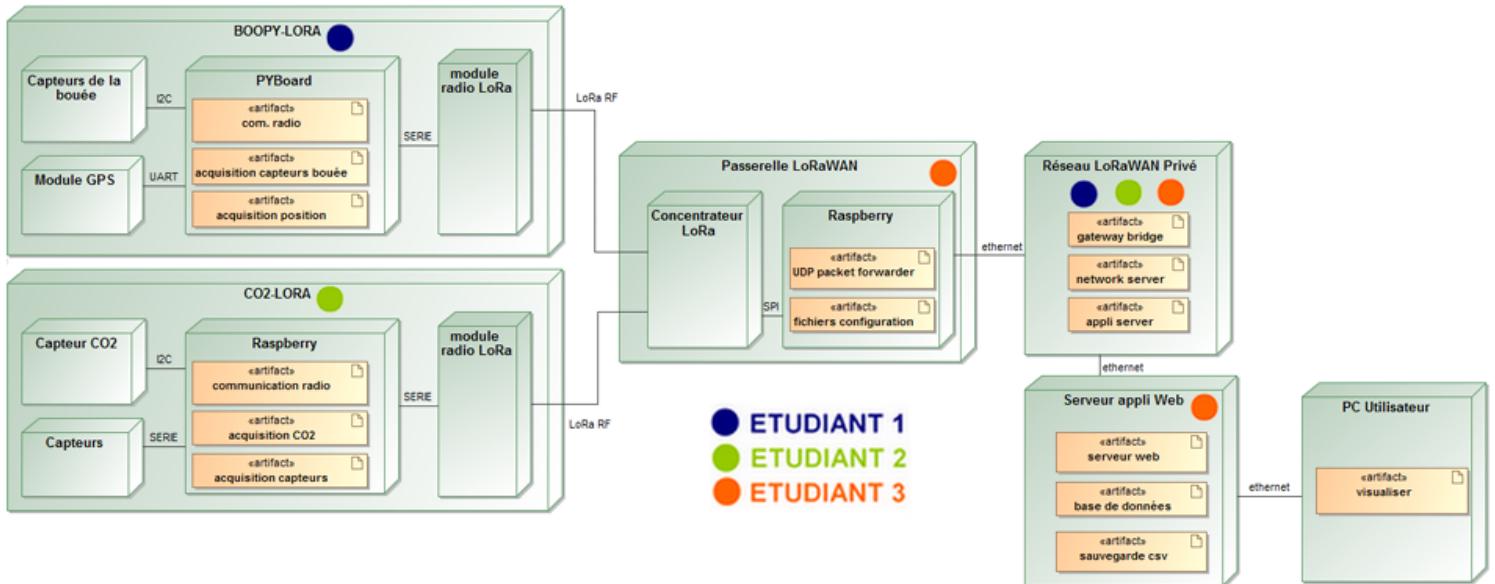
Présentation du Projet

Nous allons vous présenter notre projet : la plateforme IoT. Ce projet nous a été confié par la société Tenum, en partenariat avec le CNES (agence spatiale française) et quelques implémentations de notre établissement. Il consiste en une bouée Boopy, un module CO2-LoRa, la plateforme web ainsi qu'un site web. Notre système a été confié afin d'effectuer un test de réalisabilité, ainsi que pour affiner et évaluer nos compétences d'informaticiens.

En fonctionnement normal, la boopy et le CO2-LoRa sont supposés envoyer leurs données, respectivement mesurées dans l'eau et dans l'air, au site web via la plateforme, ou ils seront enregistrés sous csv et affichés sous format textuel. La boopy devra être codée en Python ou un langage dérivé et le système de communication sera le LoRa.



Répartition des tâches



Vous pouvez voir ci-dessus notre diagramme de déploiement ainsi que la répartition des tâches. Nous n'allons évidemment pas entrer dans les détails de chaque appareil, cependant nous allons tout de même faire une présentation rapide du milieu et du fonctionnement demandé de notre projet.

Le système est composé de deux nœuds :

- Le nœud boopy, répondant aux exigences de la société Tenum
- Le nœud CO2, répondant aux exigences de notre établissement

Le nœud Boopy doit récupérer différentes données comme la température, la luminosité, ou même des informations GPS dans un milieu aquatique.

Le nœud Co2 lui doit récupérer la température, le taux d'humidité et la qualité de l'air.

Le système dispose de son propre réseau privé LoRaWAN auquel communiquent les deux nœuds.

On doit également créer un service web où les données seront stockées sous format csv et dans une base de données. Les données devront être affichées sur une page web en temps réel.

Contraintes matérielles et logicielles

Nœud BOOPY-LORA :

- Microcontrôleur PYBoardv1.1
- Langage MicroPython

Le microcontrôleur embarqué dans la bouée est un PYBoardv1.1. La liste des capteurs présents sur la bouée est fournie dans la liste du matériel. Le module radio LoRa embarqué dans le nœud est à choisir avec le client. Le langage de développement est le langage Python.

Nœud CO2-LORA :

- Raspberry PI2 ou PI3
- Raspbian OS
- Capteur CO2 MiCS-VZ-89TE
- Module RTC DS1307
- Langage Python ou C++

Passerelle LoRaWAN :

- Raspberry PI2 ou PI3
- Raspbian OS

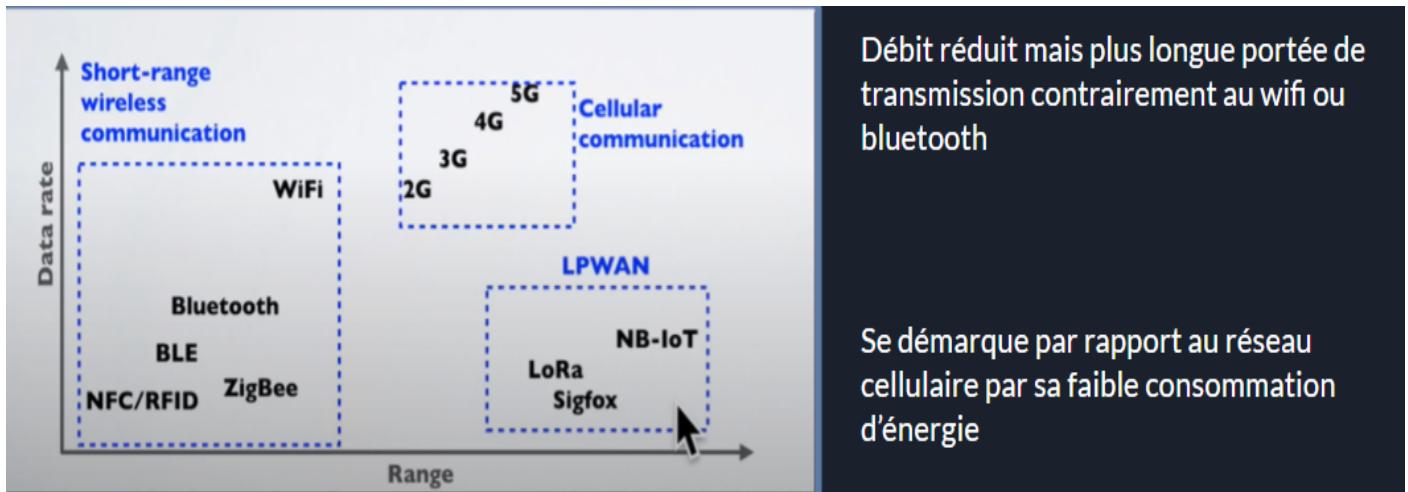
Serveur privé LoRaWAN :

- Raspberry PI2 ou PI3
- Raspbian OS
- Chirpstack (Network Server et Application Server)

Serveur application Web :

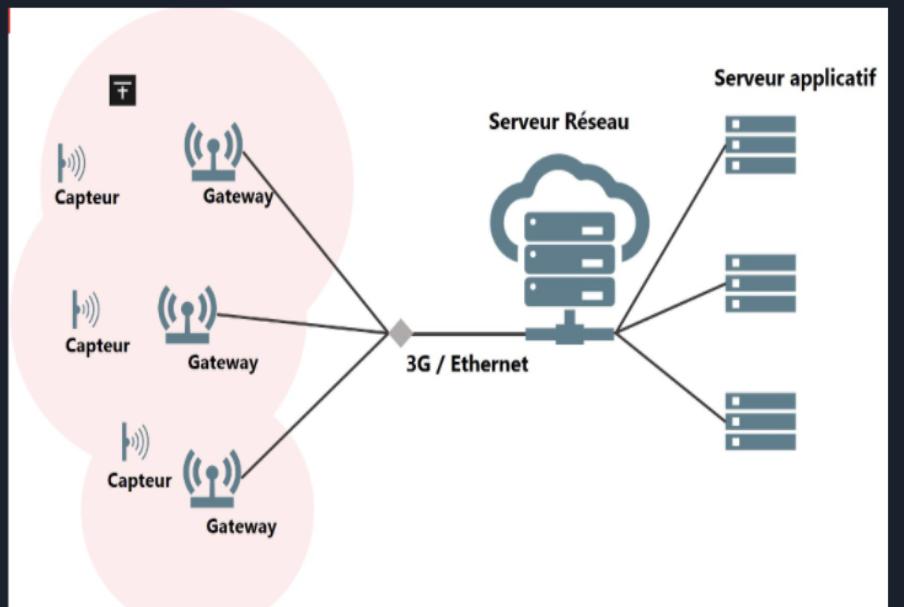
- Raspberry PI2 ou PI3
- Raspbian OS
- Broker MQTT: Mosquitto
- Base de donnée SQLite

Présentation de la technologie LoRa



Structure d'un réseau LoRaWAN

- Les capteurs généralement appelés « noeuds » qui communiquent par radio avec les passerelles
- Les passerelles (gateways) qui établissent le lien entre les communications radios et Internet
- Le serveur réseau qui sécurise et stocke les données
- le serveur d'application finale qui présente les données à l'utilisateur



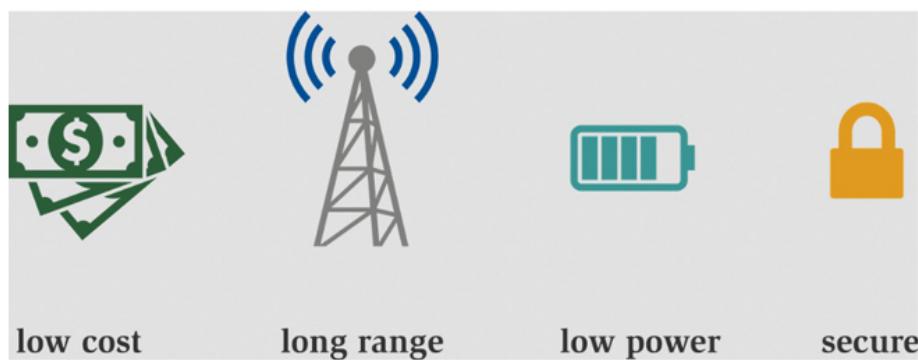
Les avantages de LoRa

Les réseaux LPWAN :

Portées > 10Km

Puissance d'émission très faible, ce qui entraîne une grande autonomie des nœuds (node) > 1an

Grande portée et faible puissance limite le débit binaire



Étudiant 1 (COUVERT Antoine)

Sommaire

- 1) Analyse du besoin
- 2) Prototypage et test
- 3) Description de la structure matérielle et logicielle
- 4) Description du fonctionnement actuel

Analyse du Besoin

Ma partie du projet a été demandée par la société Tenum en partenariat avec le CNES, l'agence spatiale française. Elle est basée sur un test de réalisabilité, où nous devrons essayer de créer une bouée python capable d'envoyer des données via LoRaWAN, et de traiter les données de différents capteurs. La bouée Boopy doit également envoyer sa position ainsi que l'heure. Pour cela je dois connecter deux capteurs, ainsi qu'un GPS, à une carte pyboard, que je dois programmer en MicroPython (version de Python spécialement faite pour les microcontrôleurs) afin de traiter leurs données. La carte dispose de deux bus UART et de deux bus I2C, que j'utilise respectivement, pour mon GPS, et pour mes deux capteurs. La carte en question peut être programmée à l'aide du logiciel Thonny, et d'un ordinateur. La pyboard est connectée à cet ordinateur via un câble USB.

Prototypage et Tests

Mon travail pour réaliser ce projet a évidemment commencé par l'apprentissage du langage python. Cela était indispensable puisque MicroPython était le seul langage autorisé. L'apprentissage théorique des bases en python s'est heureusement bien passé du au fait que j'avais déjà un peu travaillé sur ce langage lors de mes années précédent mes études.

J'ai ensuite commencé à travailler sur le programme en lui-même. Le premier capteur de température que j'ai utilisé se servait d'une librairie dédiée. Cette librairie trouvait d'elle même sur quel bus le capteur de température était connecté et renvoyait les données dans deux variables (comme ci-dessous) . J'ai d'abord eu des soucis pour passer ces deux variables en virgule puis en str (chaîne de caractère) , du fait que la partie décimale (d) était d'une taille variable et difficile à diviser correctement à chaque coups. Mais j'ai ensuite eu l'idée de passer directement par les chaînes de caractères pour une concaténation simple, qui remplit son travail admirablement bien, me permettant de régler ce problème une bonne fois pour toute.

```
(e, d) = mcpTemp.get_temp()
```

Les vrais gros problèmes de variables ne faisaient pourtant que commencer. J'ai en effet commencé à travailler sur le GPS. Ce GPS avait un gros défaut. Celui de ne pas envoyer des données intègres. En effet, contrairement aux autres capteurs qui récupéraient directement des signaux analogiques pour en faire des signaux numériques, le GPS recevait ses données depuis le ciel. Et les données n'étaient pas intègres. Ce que vous voyez comme image ci-dessous est l'apparence des trames que nous avons envoyées. Imaginez à partir de ces trames (prises sur docklight) que je ne reçois pas toutes les lignes, que les lignes ne soient pas entières, que des caractères en deviennent d'autres, et que des lignes différentes se mettent les unes à la suite des autres, et peut-être aurez-vous une petite idée du cauchemar que c'était.

```
$GPGGA,154256.00,4720.08247,N,00504.03651,E,1,06,1.91,269.6,M,47.0,M,,*52<CR><LF>
$GPGSA,A,3,32,22,18,12,29,25,,,,,,2.73,1.91,1.95*07<CR><LF>
$GPGSV,3,1,11,02,25,046,14,04,03,339,,05,02,096,,12,21,097,32*70<CR><LF>
$GPGSV,3,2,11,18,20,172,30,20,09,072,,22,21,246,32,25,59,091,28*71<CR><LF>
$GPGSV,3,3,11,26,26,290,16,29,87,080,28,32,10,229,21*49<CR><LF>
$GPGLL,4720.08247,N,00504.03651,E,154256.00,A,A*60<CR><LF>
$GPRMC,154257.00,A,4720.08235,N,00504.03658,E,0.106,,100222,,,A*70<CR><LF>
$GPVTG,,T,,M,0.106,N,0.196,K,A*2A<CR><LF>
$GPGGA,154257.00,4720.08235,N,00504.03658,E,1,06,1.91,269.8,M,47.0,M,,*51<CR><LF>
$GPGSA,A,3,32,22,18,12,29,25,,,,,,2.73,1.91,1.95*07<CR><LF>
$GPGSV,3,1,11,02,25,046,13,04,03,339,,05,02,096,,12,21,097,31*74<CR><LF>
$GPGSV,3,2,11,18,20,172,30,20,09,072,,22,21,246,32,25,59,091,29*70<CR><LF>
$GPGSV,3,3,11,26,26,290,,29,87,080,28,32,10,229,21*4E<CR><LF>
$GPGLL,4720.08235,N,00504.03658,E,154257.00,A,A*6D<CR><LF>
$GPRMC,154258.00,A,4720.08227,N,00504.03663,E,0.052,,100222,,,A*74<CR><LF>
$GPVTG,,T,,M,0.052,N,0.096,K,A*2B<CR><LF>
$GPGGA,154258.00,4720.08227,N,00504.03663,E,1,06,1.91,269.8,M,47.0,M,,*55<CR><LF>
$GPGSA,A,3,32,22,18,12,29,25,,,,,,2.73,1.91,1.95*07<CR><LF>
$GPGSV,3,1,11,02,25,046,13,04,03,339,,05,02,096,,12,21,097,31*74<CR><LF>
$GPGSV,3,2,11,18,20,172,29,20,09,072,,22,21,246,31,25,59,091,28*7A<CR><LF>
$GPGSV,3,3,11,26,26,290,,29,87,080,27,32,10,229,21*41<CR><LF>
$GPGLL,4720.08227,N,00504.03663,E,154258.00,A,A*69<CR><LF>
$GPRMC,154259.00,A,4720.08218,N,00504.03665,E,0.115,,100222,,,A*7D<CR><LF>
$GPVTG,,T,,M,0.115,N,0.213,K,A*26<CR><LF>
$GPGGA,154259.00,4720.08218,N,00504.03665,E,1,06,1.91,269.9,M,47.0,M,,*5F<CR><LF>
$GPGSA,A,3,32,22,18,12,29,25,,,,,,2.73,1.91,1.95*07<CR><LF>
$GPGSV,3,1,11,02,25,046,12,04,03,339,,05,02,096,,12,21,097,31*75<CR><LF>
$GPGSV,3,2,11,18,20,172,29,20,09,072,,22,21,246,31,25,59,091,28*7A<CR><LF>
$GPGSV,3,3,11,26,26,290,,29,87,080,27,32,10,229,21*41<CR><LF>
$GPGLL,4720.08218,N,00504.03665,E,154259.00,A,A*62<CR><LF>
```

Ce fut un véritable enfer que de mettre au point un code capable de purger ces trames, mais ce fut fait. Ce programme commence par repérer quelle trame est la bonne, à l'aide de la méthode startswith() et de l'identifiant de début de trame \$GPGGA, puis, grâce à split() et len(), je m'assure que ma trame n'est pas une combinaison de deux lignes, et qu'elle n'est pas tronquée. Ainsi, je dois éliminer plus de 90% des données envoyées par le GPS, encore maintenant. Et les problèmes de communication ne sont pas finis.

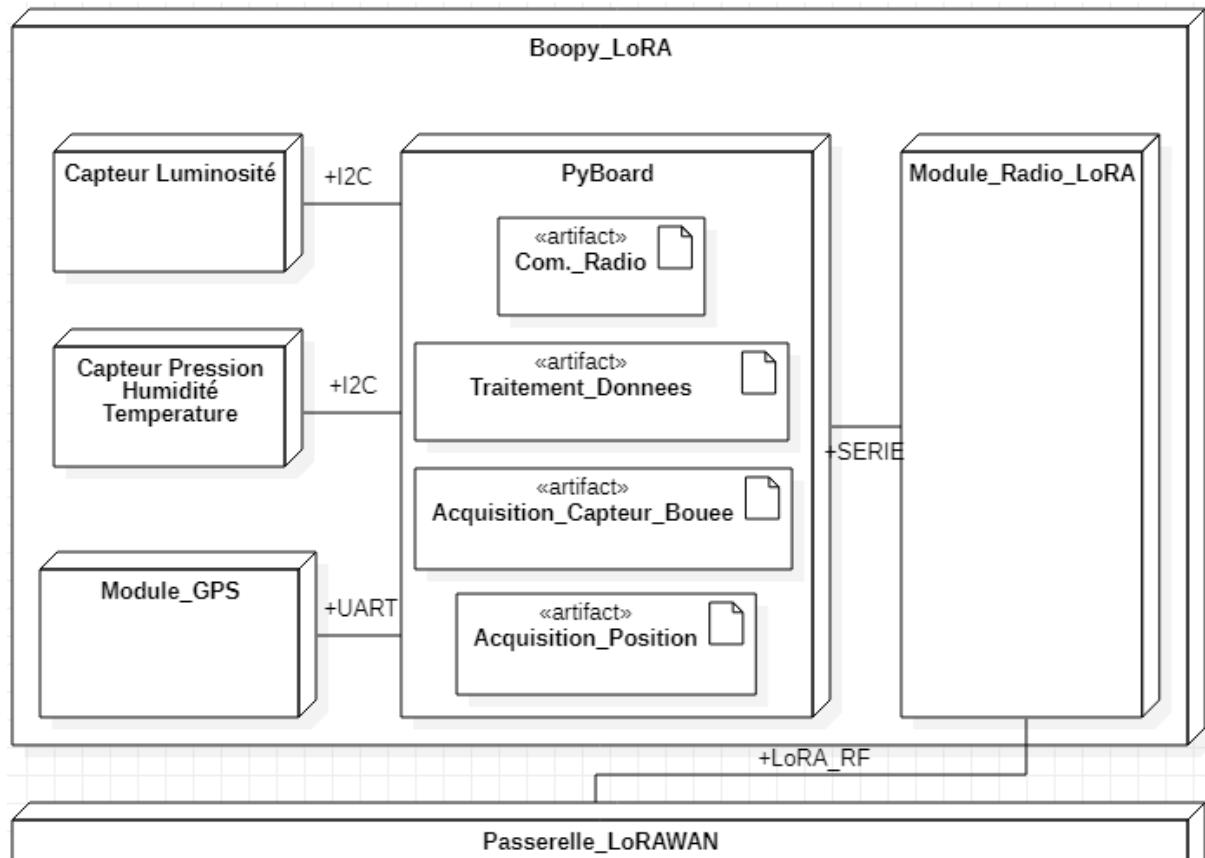
```
def correction(liste):
    cnt = 0
    elemCnt = 0
    for elem in liste:
        elemCnt = elemCnt + 1
        if("GP" in elem):
            cnt = cnt + 1
            if (cnt >= 2):
                del liste[elemCnt - 1]
    return liste

tabTrame = strTrame.split(',')
tabTrame = correction(tabTrame)
if(len(tabTrame) >= 12):
    print(tabTrame)
```

Le reste du travail sur les capteurs a continué sans rien de notable, cependant je doit préciser que jusque-là les données ont été enregistrées sur un fichier interne à l'appareil. À partir de maintenant je dois travailler sur la communication LoRaWAN afin de finaliser ce projet, cependant la communication n'est certainement pas plus aisée. En effet après quelques tests je ne cesse d'avoir la même erreur, à laquelle je ne semble pour l'instant pas trouver de solutions. Le problème des communications radio, c'est que je ne suis même pas sur que les problèmes viennent de moi, et pas de mes camarades ou même juste du milieu de travail. Ceci est donc pour l'instant toujours en cours ...

Description de la Structure Matérielle et Logicielle

Notre matériel de solution actuel comporte principalement la pyboard, contenant le code solution. Cependant elle est aussi connectée à différents capteurs grâce à ses deux bus I2C et un bus UART, respectivement connectés aux capteurs et au GPS. Il sera également connecté au module LoRa dans un prochain futur.

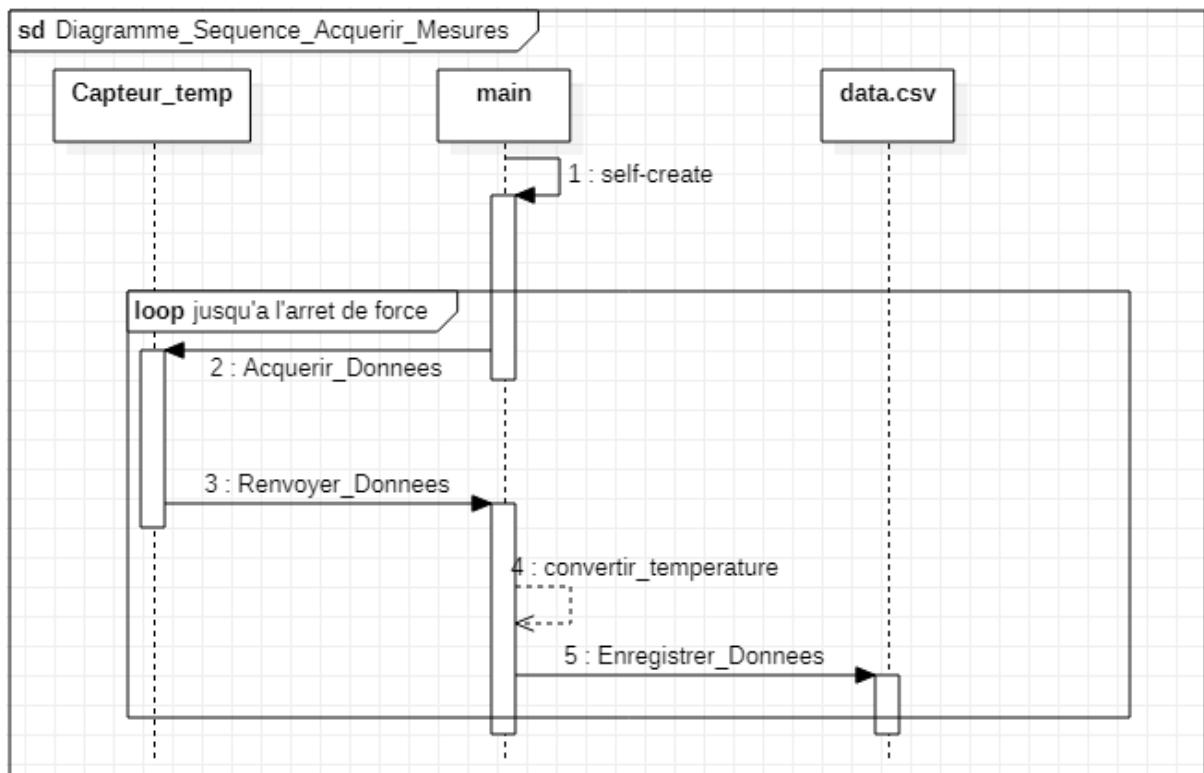


Notre code solution actuel comporte quatre classes principales. Les deux premières sont les classes UART et SoftI2C qui me permettent de déclarer et d'utiliser le bus UART du GPS ainsi que les bus I2C de mes capteurs. J'ai malheureusement été dans l'incapacité de retrouver le code ou une quelconque documentation sur la classe SoftI2C, elle n'est donc pas présente sur mon diagramme de classe. Les deux autres classes sont des classes propres aux capteurs, permettant de les exploiter et d'en recevoir des données. Une permet de recevoir des données depuis le capteur de luminosité tandis que l'autre s'occupe du capteur de température/pression/humidité.

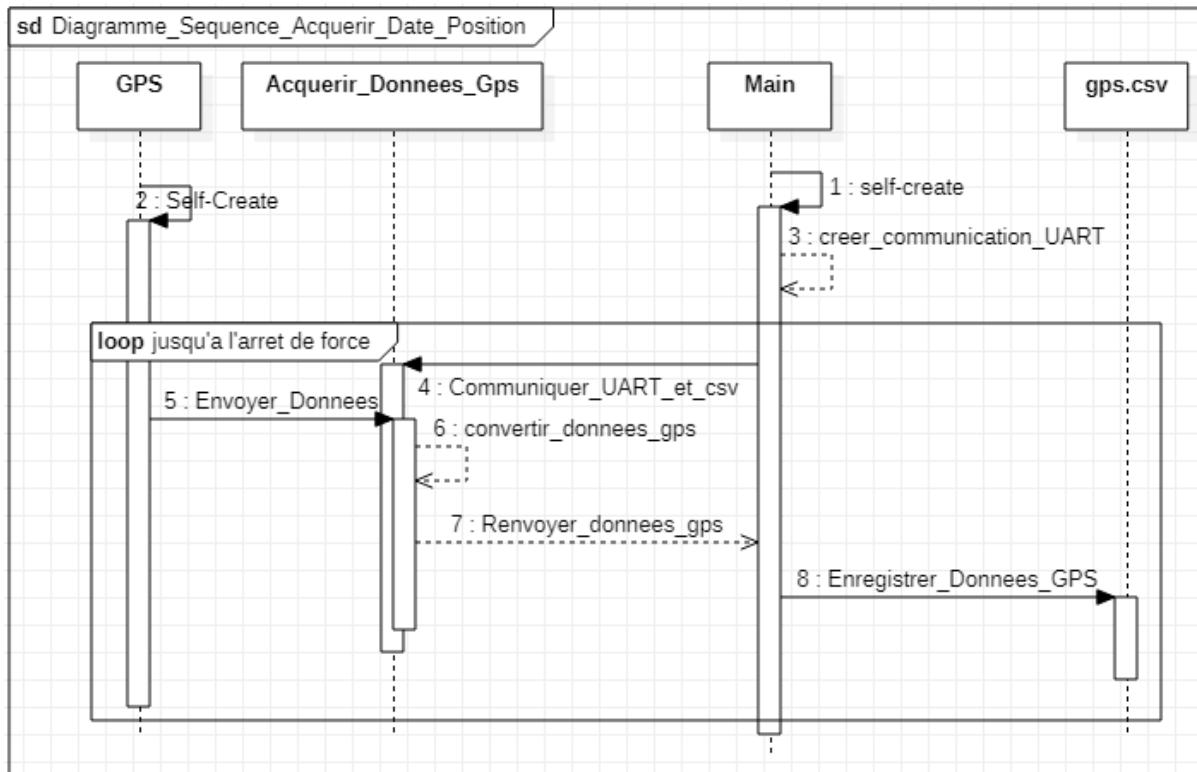
BME280	tsl2591
<pre>+BME280_REGISTER_HUMIDITY_DATA +BME280_REGISTER_TEMP_DATA +BME280_REGISTER_CONTROL +BME280_REGISTER_CONTROL_HUM +BME280_REGISTER_DIG_H5 +BME280_REGISTER_DIG_H6 +BME280_REGISTER_DIG_H4 +BME280_REGISTER_DIG_H7 +BME280_REGISTER_DIG_H3 +BME280_REGISTER_DIG_H2 +BME280_REGISTER_DIG_H1 +BME280_REGISTER_DIG_P9 +BME280_REGISTER_DIG_P8 +BME280_REGISTER_DIG_P7 +BME280_REGISTER_DIG_P6 +BME280_REGISTER_DIG_P5 +BME280_REGISTER_DIG_P4 +BME280_REGISTER_DIG_P3 +BME280_REGISTER_DIG_P2 +BME280_REGISTER_DIG_P1 +BME280_REGISTER_DIG_T3 +BME280_REGISTER_DIG_T2 +BME280_REGISTER_DIG_T1 +BME280_REGISTER_CONTROL +BME280_OSAMPLE_16 +BME280_OSAMPLE_8 +BME280_OSAMPLE_4 +BME280_OSAMPLE_2 +BME280_OSAMPLE_1 +BME280_I2CADDR</pre>	<pre>+_TSL2591_ADDR +_TSL2591_COMMAND_BIT +_TSL2591_ENABLE_POWEROFF +_TSL2591_ENABLE_POWERON +_TSL2591_ENABLE_AEN +_TSL2591_ENABLE_AIEN +_TSL2591_ENABLE_NPIEN +_TSL2591_REGISTER_ENABLE +_TSL2591_REGISTER_CONTROL +_TSL2591_REGISTER_DEVICE_ID +_TSL2591_REGISTER_CHAN0_LOW +_TSL2591_REGISTER_CHAN1_LOW +_TSL2591_LUX_DF +_TSL2591_LUX_COEKB +_TSL2591_LUX_COEFC +_TSL2591_LUX_COEFD +_TSL2591_MAX_COUNT_100MS +_TSL2591_MAX_COUNT_ +GAIN_LOW +GAIN_MED +GAIN_HIGH +GAIN_MAX +INTEGRATIONTIME_100MS +INTEGRATIONTIME_200MS +INTEGRATIONTIME_300MS +INTEGRATIONTIME_400MS +INTEGRATIONTIME_500MS +INTEGRATIONTIME_600MS</pre>
<pre>+__init__(mode = BME280_OSAMPLE_1, address = BME280_I2CADDR, i2c = None, **kwargs) +load_calibration() +read_raw_temp() +read_raw_pressure() +read_raw_humidity() +read_temperature() +read_pressure() +read_humidity() +temperature() +pressure() +humidity()</pre>	<pre>+__init__(self, i2c, address = _TSL2591_ADDR) +_read_u8(self, address) +_read_u16LE(self, address) +_write_u8(self, address, val) +enable(self) +disable(self) +gain(self) +gain(self, val) +integration_time(self) +integration_time(self, val) +raw_luminosity(self) +full_spectrum(self) +infrared(self) +visible(self) +lux(self)</pre>
UART	<pre>+RX_ANY</pre>
<pre>+UART(id, baudrate) +init(baudrate = 9600, bits = 8, parity = None, stop = 1, *) +deinit() +any() +read([nbytes]) +readinto(buf[, nbytes]) +readline() +write(buf) +sendbreak() +irq(trigger, priority = 1, handler = None, wake = machine.IDLE)</pre>	

Description du Fonctionnement Actuel

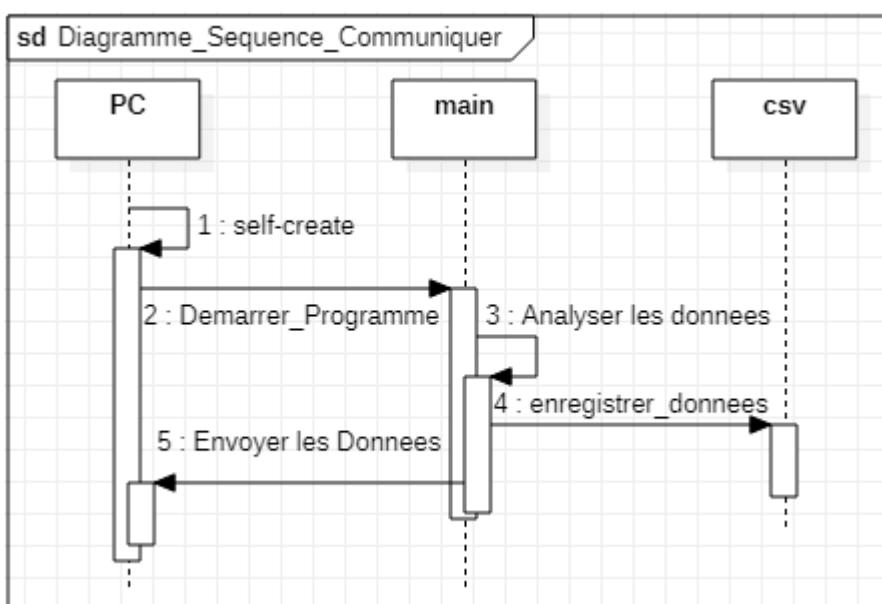
La première partie, celle de récupération des données, se fait comme vous pouvez le voir ci-dessous. Le programme se démarre, puis demande les données au capteurs, chacun leur tour, avant de les recevoir, de les traiter (avec les données du GPS) et de les enregistrer dans un fichier csv. Dans une future mise à jour du programme, les données ne seront plus enregistrées mais envoyées à la passerelle et au site web, qui les affichera et les enregistrera.



La deuxième partie du programme concerne la récupération des données GPS. Dans cette partie une communication UART est d'abord créée, puis utilisée dans un programme tiers, s'occupant du traitement des données. Ce programme vas lire l'entrée de données venant du bus UART (le GPS envoyant ses données sans cesse), les traiter afin de les rendre plus utilisables et d'éliminer les données inutiles, avant de finalement les traiter et les renvoyer au programme principale, qui pour l'instant les enregistres au format csv.



La dernière partie de mon programme concerne l'enregistrement/envoi des données. Cette dernière partie étant toujours en travail, nous parlerons surtout d'enregistrement, mais cette dernière partie est appelée à changer drastiquement. Dans cette partie du code, l'utilisateur commence par démarrer le programme, qui récupérera donc les données du GPS et des capteurs, comme décrit dans les deux diagrammes de séquence précédents. Ensuite, le programme va enregistrer les données dans une liste, puis utiliser une fonction afin de les enregistrer dans le fichier csv correspondant, en précisant son adresse depuis la carte pyboard. Dans le même temps, le programme affichera les données qui seront enregistrées.



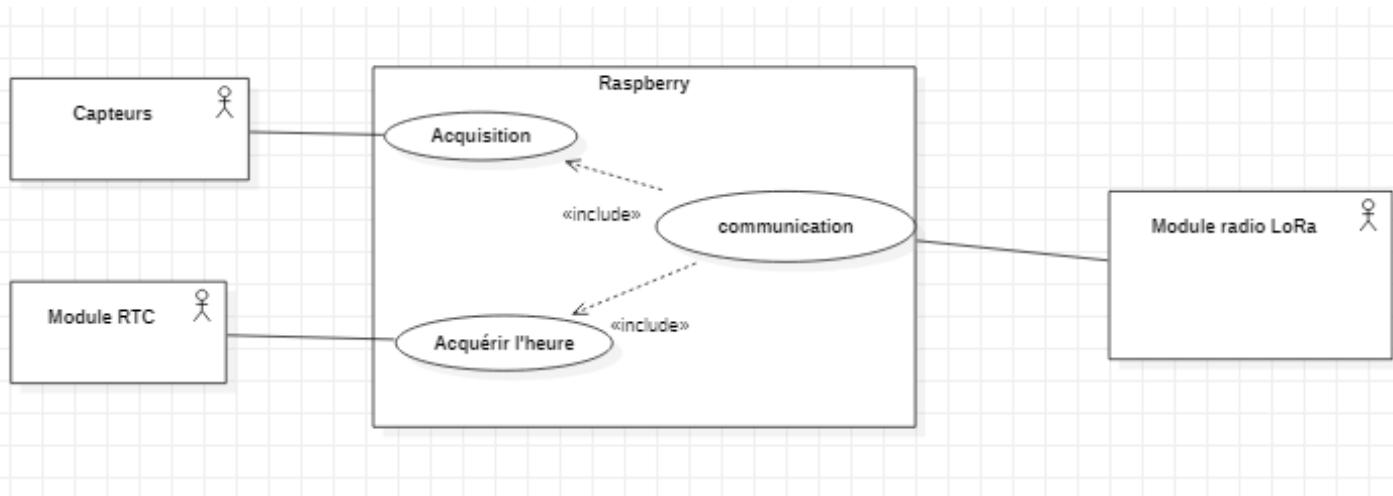
Étudiant 2 (POGOSSIAN Artur) :

Présentation du Nœud CO2

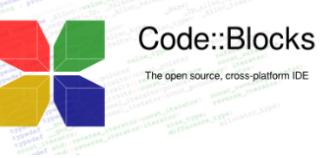
Ma partie consiste en l'acquisition et l'envoi des données à la passerelle.

Je dois créer un nœud de capteur qui fait l'acquisition de plusieurs données capteur en plus du timestamp et qui les envoie grâce à un module radio à L'étudiant n°3 par radio fréquence à une passerelle LoRaWAN qu'il aura configurée en amont.

Le nœud fonctionnera comme suivant :



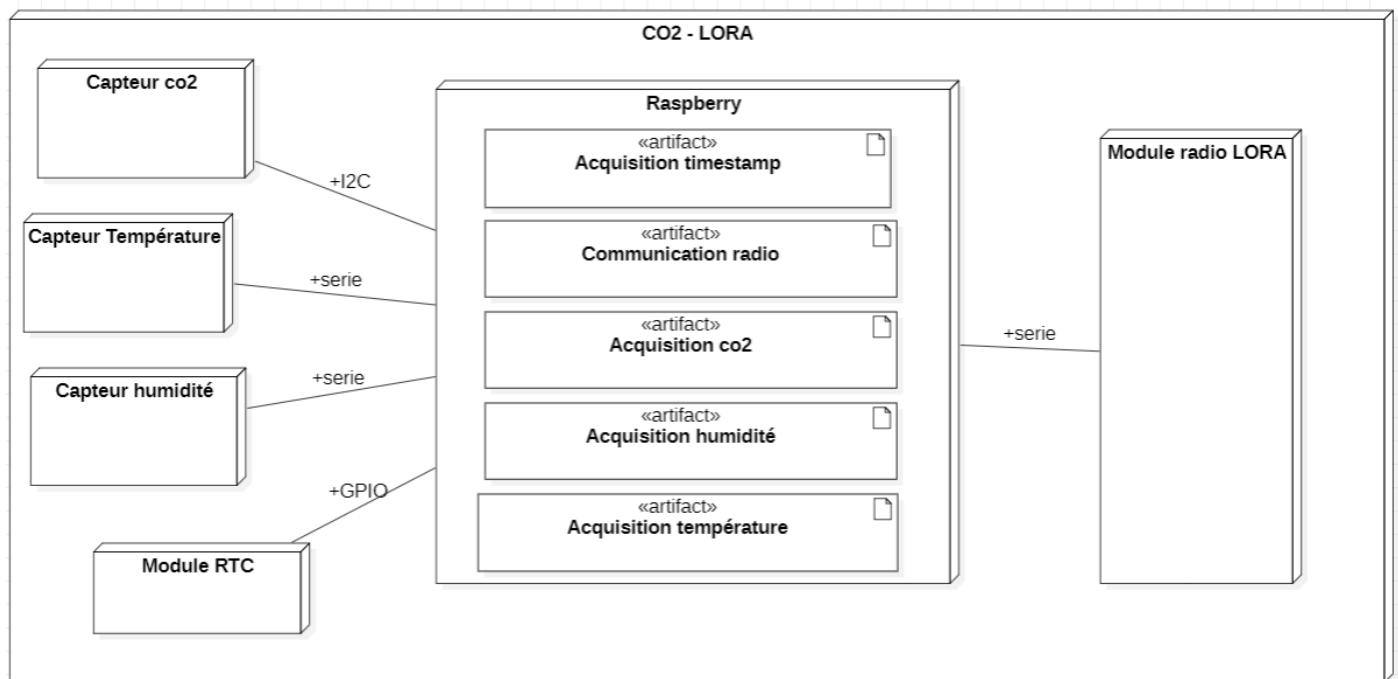
Logiciels utilisés

Nom	Logo	Rôles
Raspbian OS BUSTER		Le système d'exploitation utiliser sur le Raspberry.
Rufus		Logiciel qui sert crée la clé USB bootable pour installer Raspbian OS BUSSTER sur le Raspberry
Code::Blocks		IDE de développement c++ avec un compilateur ARM qui sert à faire de la cross compilation sur Raspberry pour développer l'application sur Windows et l'exécuter sur Raspberry
SmarTTy		Client SSH qui sert transférer et exécuter l'application C++ compiler sur Windows sur le Raspberry

Matériels utilisés

Nom	Logo	Rôles
Raspberry Pi3		Base du nœud LoRa co2 qui va accueillir : le module rtc, le module radio, le capteur co2 et le capteur Température/humidité
1 carte micro sd 32GB		Carte sd qui va servir de disque au Raspberry et qui va donc accueillir L'os du Raspberry et servir de stockage
1 capteur co2	MICS-VZ-89TE 	Capteur de co2 qui va renvoyer le taux de co2 en temps réel.
1 capteur Température / Humidité		Capteur de Température / Humidité qui va renvoyer le taux de Température / Humidité en temps réel.
1 module radio LoRa		Module radio LoRa qui nous sert à envoyer : le taux de co2, la température, l'humidité et le timestamp en Radiofréquence à la passerelle LoRa Wan
1 module RTC		Module RTC qui sert à garder la date et l'heure à jour même après un redémarrage

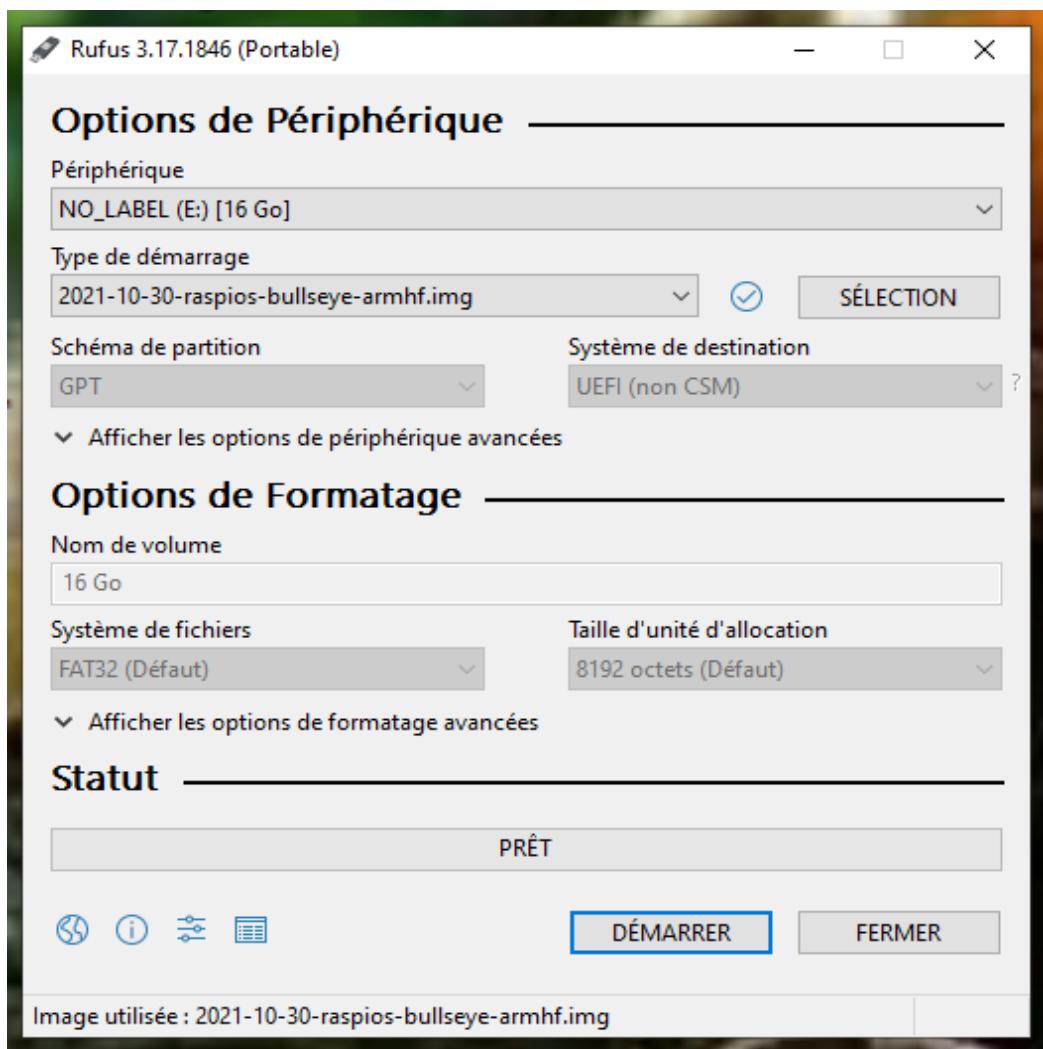
Description du système



Prototypage.

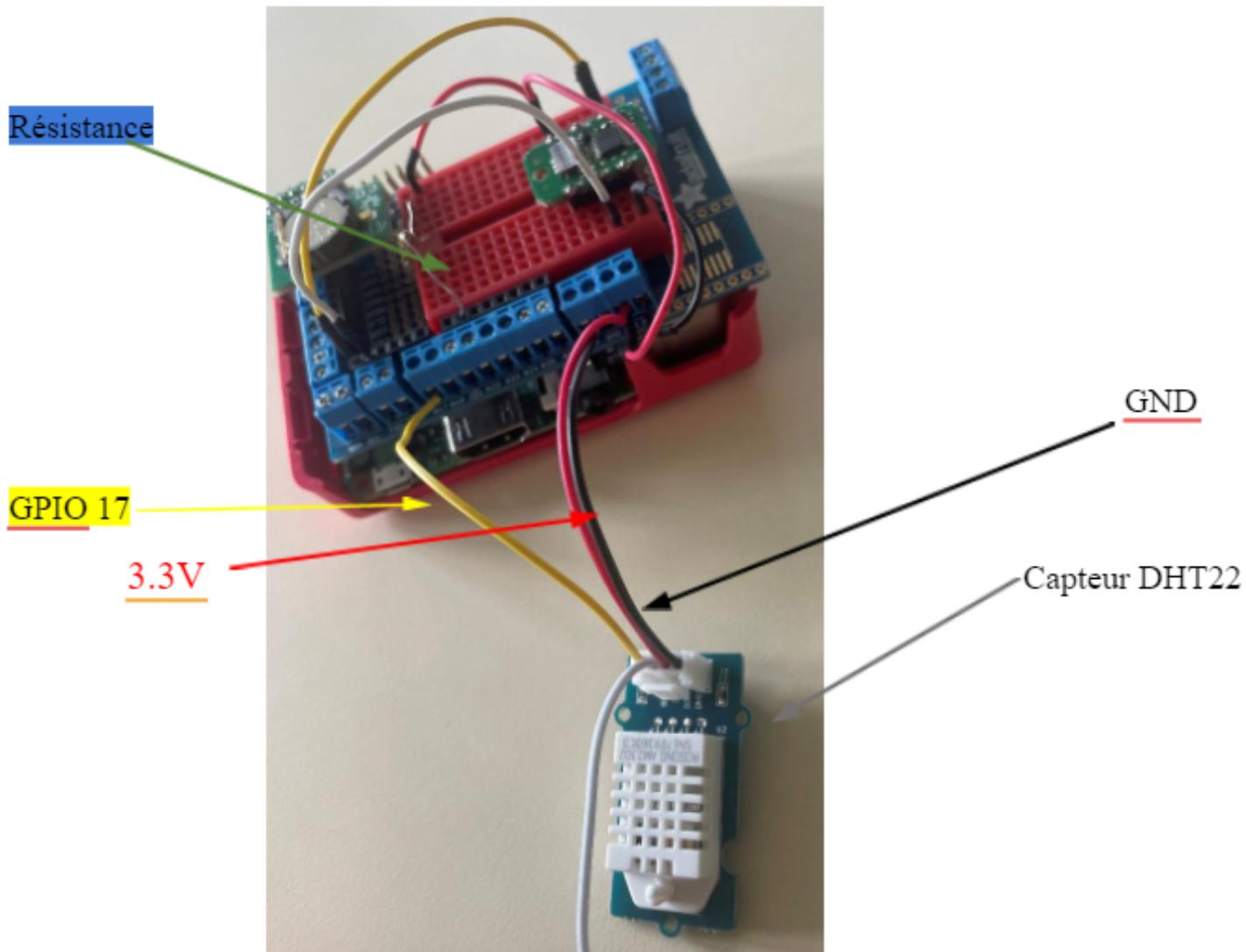
Installation du Raspberry :

Pour Installation du Raspberry il faut une micro SD bootable donc il nous faut un logiciel comme « Rufus » dans lequel on doit sélectionner sa carte MicroSD et son image ISO à flasher et lancer l'écriture.



Installation / test du capteur (DHT22) température/humidité

Pour l'installation du capteur, il faut brancher ses 3 câbles au Raspberry. Le 3.3 V et le GND et le port GPIO qu'il faut choisir (17 dans notre cas) mais pour le bon fonctionnement du capteur, il ne faut pas oublier d'ajouter une résistance de 4.7KOhm entre le 3.3V et le GPIO.



Pour le test du capteur, j'ai récupéré un programme python sur GitHub qui utilise la librairie «Adafruit_DHT» qui récupère les données capteur et qui avec une fonction de conversion me renvoie les valeurs dont j'ai besoin. J'ai donc dû suivre ces étapes :

Installation à la fois python3-dev et python3-pip en exécutant la commande ci-dessous.

```
sudo apt-get install python3-dev python3-pip
```

Exécution de la commande suivante pour nous assurer que nous disposons des dernières versions des packages setuptools , wheel et pip python.

```
sudo python3 -m pip install --upgrade pip setuptools wheel
```

Exécution de la commande suivante pour installer la bibliothèque DHT sur votre Raspberry Pi.

```
sudo pip3 install --install-option="--force-pi" Adafruit_DHT
```

Création d'un script Python dans lequel nous stockerons tout notre code.

```
nano ~/humidity.py
```

Copie du code ci-dessous dans le script humidity.py créé juste avant :

```
1 import Adafruit_DHT
2
3 DHT_SENSOR = Adafruit_DHT.DHT22
4 DHT_PIN = 17
5
6 while True:
7     humidity, temperature = Adafruit_DHT.read_retry(DHT_SENSOR, DHT_PIN)
8
9     if humidity is not None and temperature is not None:
10         print("Temp={0:0.1f}*C  Humidity={1:0.1f}%".format(temperature, humidity))
11     else:
12         print("Failed to retrieve data from humidity sensor")
13
14
```

Dans ce code, on sélectionne le pin du DHT donc « GPIO(17) » ensuite on entre dans une boucle avec la condition de si on arrive à lire le capteur depuis le pin sélectionné on le imprime dans la console si les valeurs «humidity» et «temperature» qui sont renvoyées par la fonction «Adafruit_DHT.read_retry» sinon on affiche un message d'erreur "Failed to retrieve data from humidity sensor".

Exécuter ce script en exécutant la commande suivante :

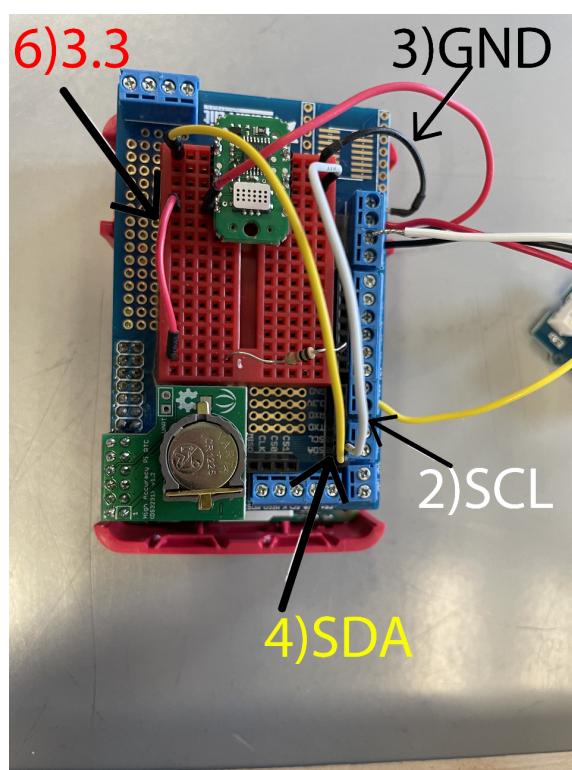
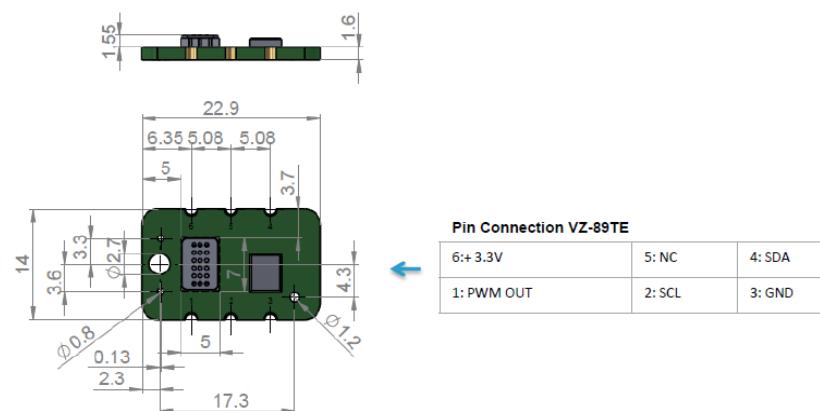
```
python3 ~/humidity.py
```

Résultat :

```
Temp=25.6*C  Humidity=51.8%
Temp=25.2*C  Humidity=45.4%
Temp=25.1*C  Humidity=45.4%
Temp=25.0*C  Humidity=45.3%
```

Installation et test du capteur MICS-VZ-89TE

Pour l'installation du capteur, j'ai dû souder des petits pics sur chaque port du capteur pour pouvoir le rentrer dans la platine rouge rajouter sur le Raspberry et ensuite le câbler comme indiquer dans la photo ci-dessous :



Pour le test du capteur, j'ai utilisé une librairie c++ que j'ai récupéré sur GitHub qui récupère les données capteur et qui avec une fonction de conversion me renvoie la valeur du co2.

```
● 1 int main(int argc, char *argv[]) {
  2     // Initialize and open the bus.
  3     if (isVerbose) {
  4         std::cout << "# Success." << std::endl;
  5     }
  6     if (!parseCommandLine(argc, argv)) {
  7         return 1;
  8     }
  9     if (!openBus()) {
 10         return 1;
 11     }
 12     float voc = 0.0f;
 13     float co2 = 0.0f;
 14     if (!getValues(voc, co2)) {
 15         return 1;
 16     }
 17     // Output the values as JSON
 18     std::cout << "{ \"voc\": " << voc << ", \"co2\": " << co2 << " }" << std::endl;
 19     // Close the bus.
 20     closeBus();
 21     // Success
 22     if (isVerbose) {
 23         std::cout << "# Success." << std::endl;
 24     }
 25     return 0;
 26 }
 27
```

Dans le programme en c++ ci-dessus on commence par initialiser le bus i2c ensuite on initialise les deux variables : voc et co2 dans lequel on va stocker les valeurs récupérer avec la méthode « getValues » et pour finir on printf les valeurs dans la console.

Installation et configuration du module RTC (DS3231) :

Pour l'installation du module RTC il nous faut juste le brancher sur le Raspberry comme ci-dessous :



Pour la configuration du module rtc il faut tout d'abord vérifier qu'il soit bien détecté par le Raspberry avec la commande suivante :

```
sudo i2cdetect -y 1
```

La commande nous montre bien que le module rtc est bien reconnu sur le port I2C « 68 ».

	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
00:	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
10:	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
20:	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
30:	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
40:	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
50:	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
60:	--	--	--	--	--	--	--	--	68	--	--	--	--	--	--	--
70:	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

Ensuite, on doit mettre à jour la date du Raspberry

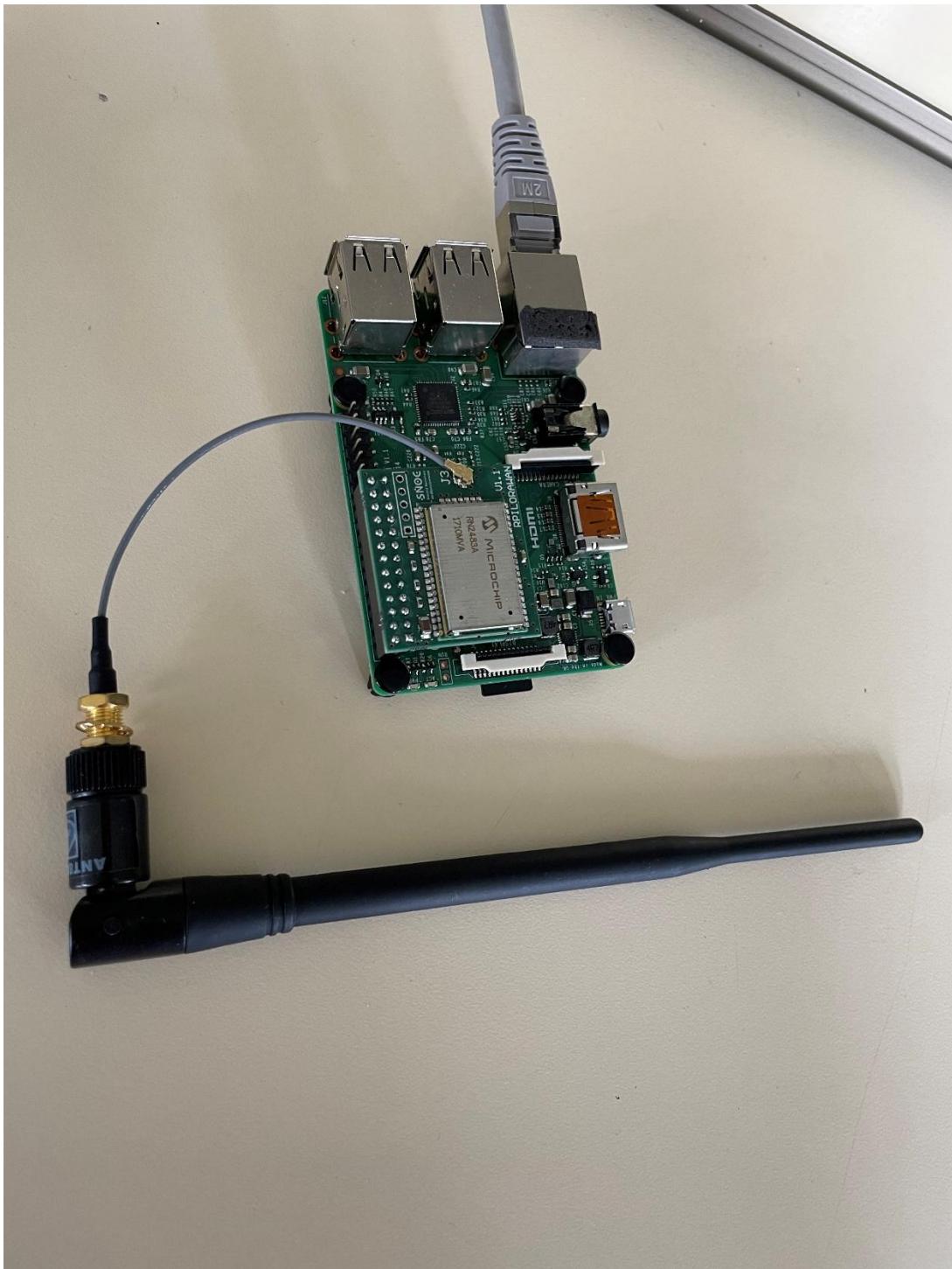
```
pi@raspberrypi:~ $ sudo date -s "Tue May 25 09:42:40 UTC 2022"
mercredi 25 mai 2022, 11:42:40 (UTC+0200)
```

Pour finir on fait cette commande pour injecter l'heure dans le module rtc

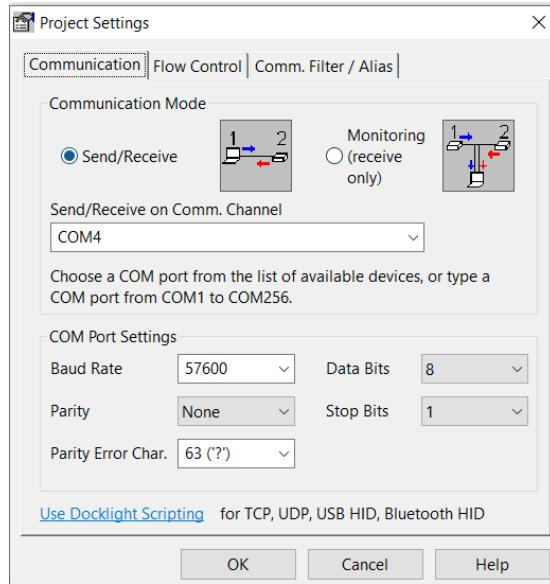
```
sudo hwclock -w
```

Installation et test du module radio RF LoRa Wan (RN2483A)

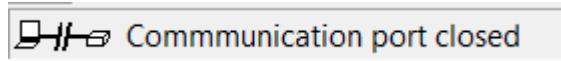
Pour l'installation du module RF il faut juste le brancher sur le Raspberry avec ses pins comme ci-dessous :



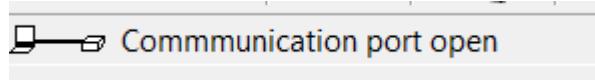
Pour le test du module RF j'ai envoyé des trames LoRa avec docklight à la passerelle. Pour commencer, il nous faut sélectionner le port COM et attribuer le nombre de BAUD dans les paramètres de docklight.



Il nous faut ensuite ouvrir le port COM en double-cliquant sur :



Pour ensuite avoir :



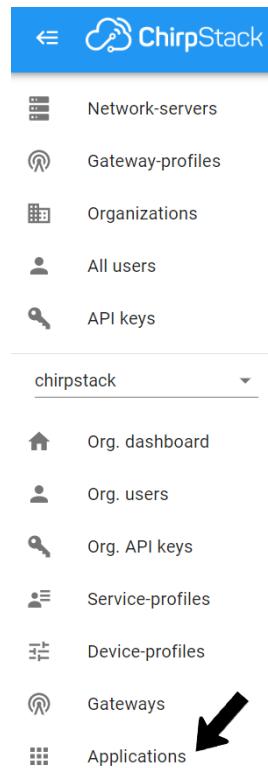
Après avoir ouvert le port, COM nous reste à envoyer les trames :

(Trames envoyées en bleu et trames reçues en rouge)

ASCII	HEX	Decimal	Binary	Colors&Fonts Mode	COM4	57600, None, 8, 1	Communication
24/05/2022 16:27:47.740 [TX] - sys get ver<CR><LF>							Recuperation de la version du module
24/05/2022 16:27:47.769 [RX] - RN2483 1.0.1 Dec 15 2015 09:38:09<CR><LF>							
24/05/2022 16:27:59.032 [TX] - mac set devaddr 260B05F4<CR><LF>							Atribution de la key «device adress»
24/05/2022 16:27:59.059 [RX] - ok<CR><LF>							
24/05/2022 16:27:59.969 [TX] - mac set appskey 39A84977ECA671B2694998338E517AB9<CR><LF>							Atribution de la key «application session key»
24/05/2022 16:27:59.995 [RX] - ok<CR><LF>							
24/05/2022 16:28:00.747 [TX] - mac set nwkskey B9C2ED1ECAD28EE654D885EB968D267E<CR><LF>							Atribution de la key «network session key»
24/05/2022 16:28:00.773 [RX] - ok<CR><LF>							
24/05/2022 16:28:01.605 [TX] - mac set deveui 70B3D57ED0048D3E<CR><LF>							Atribution de la key «device EUI»
24/05/2022 16:28:01.631 [RX] - ok<CR><LF>							
24/05/2022 16:28:06.096 [TX] - mac join abp<CR><LF>							Connexion en mode abp
24/05/2022 16:28:06.121 [RX] - ok<CR><LF>							
accepted<CR><LF>							
24/05/2022 16:28:08.864 [TX] - mac tx cnf 4 68656C6C6F<CR><LF>							Envoie d'une trame contenant «hello» en hexadecimal
24/05/2022 16:28:08.915 [RX] - ok<CR><LF>							
mac_tx_ok<CR><LF>							

Configuration du nœud dans ChispStack

Pour recevoir les trames envoyées, il nous faut configurer les clés dans la partie application de ChispStack en suivant ces étapes :

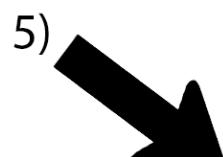


The screenshot shows a table listing applications. The columns are:

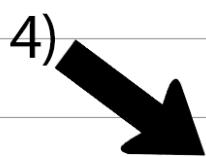
ID	Name	Service-profile	Description
3	app1	profile1	pp1

At the bottom right of the table are pagination controls: "Rows per page: 10", "1-1 of 1", and navigation arrows. A black arrow points to the "app1" entry in the Name column.

The screenshot shows the "Applications / app1" configuration page. At the top right is a red "DELETE" button. Below it is a navigation bar with tabs: "DEVICES" (which is active), "MULTICAST GROUPS", "APPLICATION CONFIGURATION", and "INTEGRATIONS". At the bottom are two buttons: a red "+ CREATE" button and a grey "SELECTED DEVICES" button. A large black arrow points to the "DEVICES" tab.

GENERAL	VARIABLES	TAGS
Device name *	1) Attribution du nom du device	
Noeud_co2	The name may only contain words, numbers and dashes.	
Device description *	2) Description du device	
données capteurs co2/température/humidité		
Device EUI *	3) Attribution du device EUI	
52 99 da 8f 76 c1 3f 31	MSB	↻
Device-profile *	4) Sélection du profil device profil (abp)	
ABP		
<input type="checkbox"/> Disable frame-counter validation	Note that disabling the frame-counter validation will compromise security as it enables people to perform replay-attacks.	
<input type="checkbox"/> Device is disabled	ChirpStack Network Server will ignore received uplink frames and join-requests from disabled devices.	
5)  CREATE DEVICE		

 DELETE

DETAILS	CONFIGURATION	KEYS (OTAA)	ACTIVATION	DEVICE DATA	LORAWAN FRAMES
 CLEAR DEVNONCE					
Device address *	1) Attribution de la key «device address»		MSB   		
01 11 68 b6	While any device address can be entered, please note that a LoRaWAN compliant device address consists of an AddrPrefix (derived from the NetID) + NwkAddr.				
Network session key (LoRaWAN 1.0) *	2) Attribution de la key «network session key»		MSB   		
36 b5 5c b4 3a 0d 91 a9 62 b8 a7 18 69 de ca 38					
Application session key (LoRaWAN 1.0) *	3) Attribution de la key «application session key»		MSB   		
10 cc 7b 4d cb 18 b5 76 9f 67 ed 93 94 d8 2d 22					
Uplink frame-counter *	4)  (RE)ACTIVATE DEVICE				
0					
Downlink frame-counter (network) *					
0					

Réception des trames envoyées :

Trame envoyer (« hello » en hexadécimal) :

25/05/2022 11:15:14.772 [TX] - mac tx cnf 4 48656c6c6f<CR><LF>

Trame reçue :

```
May 25 11:21:12 AM UnconfirmedDataDown 868.3 MHz SF8 BW125 FCnt: 16 DevAddr: 260b05f4 GW: b827ebe6f7acffff

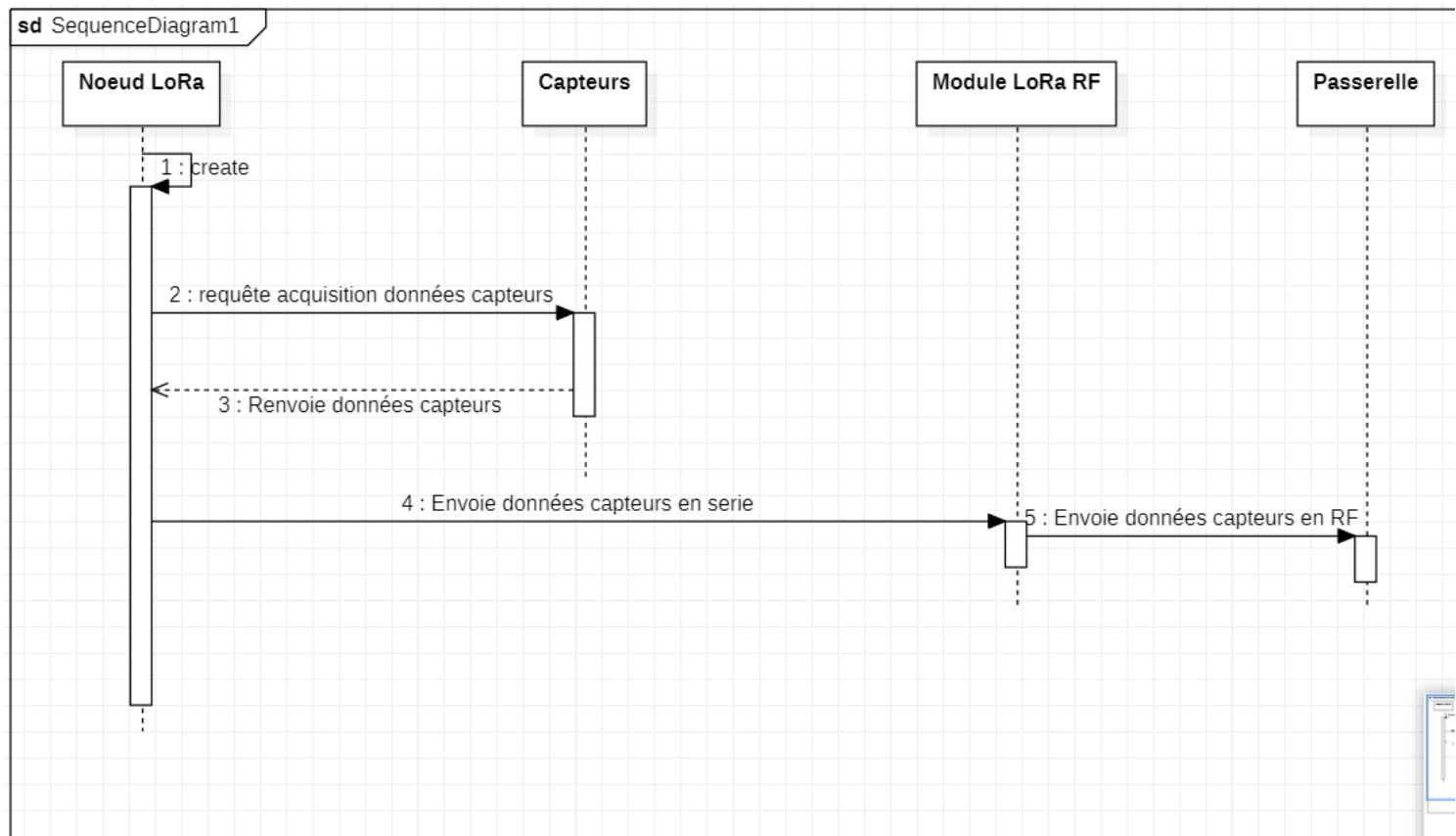
gatewayID: "b827ebe6f7acffff"
▼ txInfo: {} 9 keys
  frequency: 868300000
  power: 14
  modulation: "LORA"
  ▼ LoRaModulationInfo: {} 4 keys
    bandwidth: 125
    spreadingFactor: 8
    codeRate: "4/5"
    polarizationInversion: true
  board: 0
  antenna: 0
  timing: "DELAY"
  ▼ delayTimingInfo: {} 1 key
    delay: "1s"
    context: "60RmbA=="
  ▼ phyPayload: {} 3 keys
    ▼ mhdr: {} 2 keys
      mType: "UnconfirmedDataDown"
      major: "LoRaWANR1"
    ▼ macPayload: {} 3 keys
      ▼ fhdr: {} 4 keys
        devAddr: "260b05f4"
      ▼ fctrl: {} 5 keys
        adr: true
        adrAckReq: false
        ack: true
        fPending: false
        classB: false
      fCnt: 16
      ▼ fopts: [] 1 item
        ▼ 0: {} 2 keys
          cid: "LinkADRReq"
        ▶ payload: {} 4 keys
        fPort: null
        frmPayload: null
        mic: "d917829d"
```

Data reçu dans la trame :

```
adr: true
dr: 5
fCnt: 15
fPort: 4
data: "SGVsbG8="
  ↙ Data en base 64
▼ objectJSON: {} 1 key
  ▼ data_decoded: {} 1 key
    trame: "Hello"
      ↙ Data convertie en ASCII
tags: {} 0 keys
confirmedUplink: true
devAddr: "260b05f4"
publishedAt: "2022-05-25T09:15:33.827454264Z"
deviceProfileID: "5c20c831-04c0-4a6f-a4d1-c60d4672d0f7"
deviceProfileName: "ABP"
```

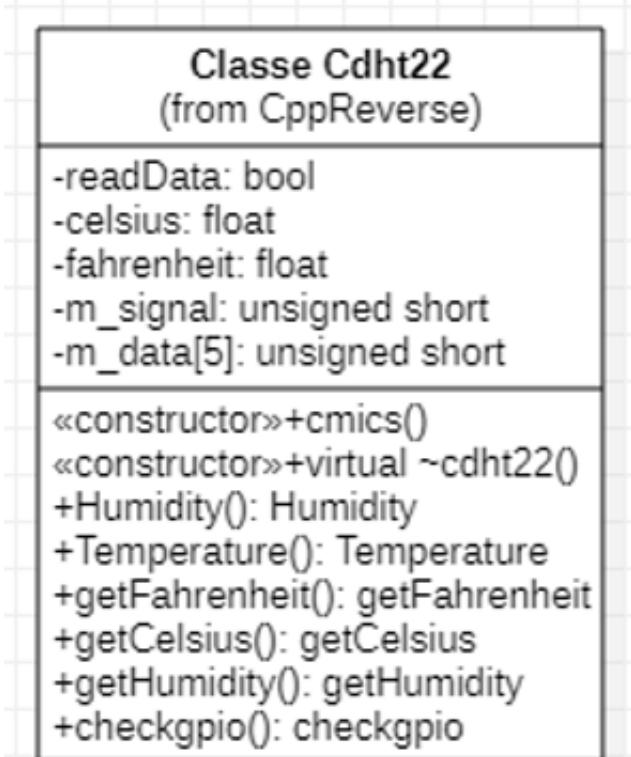
Conceptions détaillées

Diagrammes de séquences entre objet :



Création de la classe Cdht22

Diagramme de classe



Présentation des méthodes de la classe

Constructeur : Initialisation des variables et définition du port GPIO du capteur (gpio 17).

```
cdht22::cdht22()
{
    humidity = 0;
    checksum = 0;
    celsius = 0;
    fahrenheit = 0;
    m_signal = 17;
    for(int i = 0; i<5; i++)
    {
        m_data[i] = 0;
    }
}
```

Méthode « checkgpio() » : Méthode qui nous sert à vérifier si la connexion GPIO est bien établie, sinon nous renvoie un message d'erreur.

```
void cdht22::checkgpio()
{
    if (wiringPiSetupGpio() == -1)
    {
        printf("[x_x] GPIO Initialization FAILED.\n\n");
    }
    else
    {
        printf("GPIO Initialization SUCCEED.\n\n");
    }
}
```

Méthode « readData() » : Méthode qui récupère les données du capteur DHT22.

```
short cdht22::readData()
{
    unsigned short val = 0x00;
    unsigned short signal_length = 0;
    unsigned short val_counter = 0;
    unsigned short loop_counter = 0;
    while (1)
    {
        while (digitalRead(m_signal) == HIGH)
        {
            signal_length++;
            if (signal_length >= 200) return -1;
        }

        if (signal_length > 0) {
            loop_counter++;

            if (signal_length < 10) val <<= 1;
            else if (signal_length < 30) val <<= 1;
            else if (signal_length < 85) {
                val <<= 1;
                val |= 1;
            } else return -1;

            signal_length = 0;
            val_counter++;
        }

        if (loop_counter < 3) {
            val = 0x00;
            val_counter = 0;
        }
        if (val_counter >= 8) {
            m_data[(loop_counter / 8) - 1] = val;
            val = 0x00;
            val_counter = 0;
        }
    }
}
```

Méthode « Temperature() » : La méthode température nous calcule la température Celsius à l'aide d'une fonction de conversion en utilisant les données acquises et stockées dans le tableau “m_data” avec la méthode “readData()”.

```
void cdht22::Temperature()
{
    pinMode(m_signal, OUTPUT);
    digitalWrite(m_signal, LOW);
    delay(20);
    pinMode(m_signal, INPUT);

    readData();

    checksum = (m_data[0] + m_data[1] + m_data[2] + m_data[3]) & 0xFF;

    if (m_data[4] == checksum && checksum != 0x00)
        celsius = ((m_data[2] & 0x7F)*256) + m_data[3]) / 10.0;
}
```

Méthode « humidity() » : La méthode température nous calcule le % d'humidité à l'aide d'une fonction de conversion en utilisant les données acquises et stockées dans le tableau “m_data” avec la méthode “readData()”.

```
void cdht22::Humidity()
{
    pinMode(m_signal, OUTPUT);
    digitalWrite(m_signal, LOW);
    delay(20);
    pinMode(m_signal, INPUT);

    readData();

    checksum = (m_data[0] + m_data[1] + m_data[2] + m_data[3]) & 0xFF;

    if (m_data[4] == checksum && checksum != 0x00)
        humidity = ((m_data[0] * 256) + m_data[1]) / 10.0;
}
```

Création de la classe CMics

Modélisation de la classe CMics

Classe Cmics (from Model3)
-i2c_fd: int
+Openbus(): bool
+Closebus(): bool
+getValues(&voc: float, &co2: float): bool
-writeData(*data: const uint8_t, size: int): bool
-readData(*data: uint8_t, size: int): bool
-getCRC(*buffer: const uint8_t, size: const uint8_t): uint8_t
-writeCommand(Command cmd: const, data : const uint32_t): bool
-readResult(data[7]: uint8_t): bool

Présentation des méthodes de la classe

Constructeur : initialisation de la variable “fd” qui est l’adresse du bus i2c

```
cmics::cmics()
{
    i2c_fd = 0 ;
}
```

Méthode « openBus() »: Cette méthode nous sert à ouvrir le bus i2c

```
bool cmics::openBus()
{
    const std::string cDevicePath = "/dev/i2c-1";
    i2c_fd = open(cDevicePath.c_str(), O_RDWR);
    if (i2c_fd < 0) {
        return false;
    }
    if (ioctl(i2c_fd, I2C_SLAVE, cChipAddress) < 0) {
        return false;
    }
    return true;
}
```

Méthode « closeBus() »: Cette méthode nous sert à fermer le bus i2c

```
bool cmics::closeBus()
{
    close(i2c_fd);
    return true;
}
```

Méthode « writeData() »: Cette méthode nous sert à écrire une data et sa taille en octets sur le bus i2c

```
bool cmics::writeData(const uint8_t *data, int size)
{
    if (write(i2c_fd, data, size) != size) {
        return false;
    }
    return true;
}
```

Méthode « readData() »: Cette méthode nous sert à lire une data et sa taille en octets sur le bus i2c.

```
bool cmics::readData(uint8_t *data, int size)
{
    if (read(i2c_fd, data, size) != size) {
        return false;
    }
    return true;
}
```

Méthode « getCRC() »: Cette méthode nous sert à calculer le CRC

```
uint8_t cmics::getCRC(const uint8_t *buffer, const uint8_t size)
{
    uint16_t sum = 0;
    for (uint8_t i=0; i<size; i++) {
        sum += buffer[i];
    }
    uint8_t crc = static_cast<uint8_t>(sum);
    crc += (sum / 0x0100);
    crc = (0xff-crc);
    return crc;
}
```

Méthode « writeCommand() »: Cette commande nous sert à écrire une commande sur le bus i2c

```
bool cmics::writeCommand(const Command cmd, const uint32_t data = 0)
{
    uint8_t buffer[6];
    buffer[0] = static_cast<uint8_t>(cmd);
    buffer[1] = static_cast<uint8_t>(data & 0x000000ffUL);
    buffer[2] = static_cast<uint8_t>((data & 0x0000ff00UL)>>8);
    buffer[3] = static_cast<uint8_t>((data & 0x00ff0000UL)>>16);
    buffer[4] = static_cast<uint8_t>((data & 0xff000000UL)>>24);
    buffer[5] = getCRC(buffer, 5);
    const auto result = writeData(buffer, 6);
    if (!result) {
    }
    return result;
}
```

Méthode « readResult() »: Cette méthode nous sert à lire le résultat des données envoyées par le capteur obtenu après l'écriture de la commande.

```
bool cmics::readResult(uint8_t data[7])
{
    if (!readData(data, 7)) {
        return false;
    }
    const uint8_t crc = getCRC(data, 6);
    if (data[6] != crc) {
        return false;
    }
    return true;
}
```

Méthode « getValues() »: Cette méthode nous sert à calculer les deux valeurs: "voc et co2" avec une fonction de conversion pour chacune des valeurs.

```
bool cmics::getValues(float &voc, float &co2)
{
    for (uint8_t comTry = 0; comTry < 3; ++comTry) {
        Command status = GetStatus;
        if (writeCommand(status,0)) {
            const std::chrono::milliseconds cReadDelay(300);
            std::this_thread::sleep_for(cReadDelay);
            uint8_t data[7];
            if (readResult(data)) {
                voc = ((data[0] - 13) * (1000 / 229));
                co2 = ((data[1] - 13) * (1600 / 229) + 400);
                return true;
            }
            std::this_thread::sleep_for(std::chrono::milliseconds(20));
        }
    }
    return false;
}
```

Création de la classe CLora

Modélisation de la classe CLora

CLora (from CppReverse)
-serial: int
-baud: int
-port: char*
«constructor»+CLora(p_port: char*, baud: int)
+config(): config
+sendData(trame: char*): sendData
-get_hex_string(buff: char*, buff_len: int, ret: char*): get_hex_string

Méthode « constructeur(char* p_port, int p_baud) » : initialisation des variables et setup du wiringPi

```
CLora::CLora(char* p_port, int p_baud)
{
    port = p_port;
    baud = p_baud;
    wiringpiSetup();
    serial = serialOpen(port, baud);
}
```

Méthode « config() » : commande de config loraWan pour se connecter à la passerelle

```
void CLora::config() {
    serialPuts(serial, "mac reset 868\r\n");
    serialPuts(serial, "mac set devaddr 260B05F4\r\n");
    serialPuts(serial, "mac set appskey 39A84977ECA671B2694998338E517AB9\r\n");
    serialPuts(serial, "mac set nwkskey B9C2ED1ECAD28EE654D885EB968D267E\r\n");
    serialPuts(serial, "mac deveui 70B3D57ED0048D3E\r\n");
    serialPuts(serial, "mac join abp\r\n");
}
```

Méthode « sendData()» : cette méthode va nous servir à envoyer la trame. Dans cette commande ont créé une variable “payload” qui fait 2x la taille de la chaîne de caractère , car payload = la chaîne de caractère en hexadécimal qu'on envoie et a la fin avec les “strcat” on concatène les chaînes de caractère pour obtenir la commande entière.

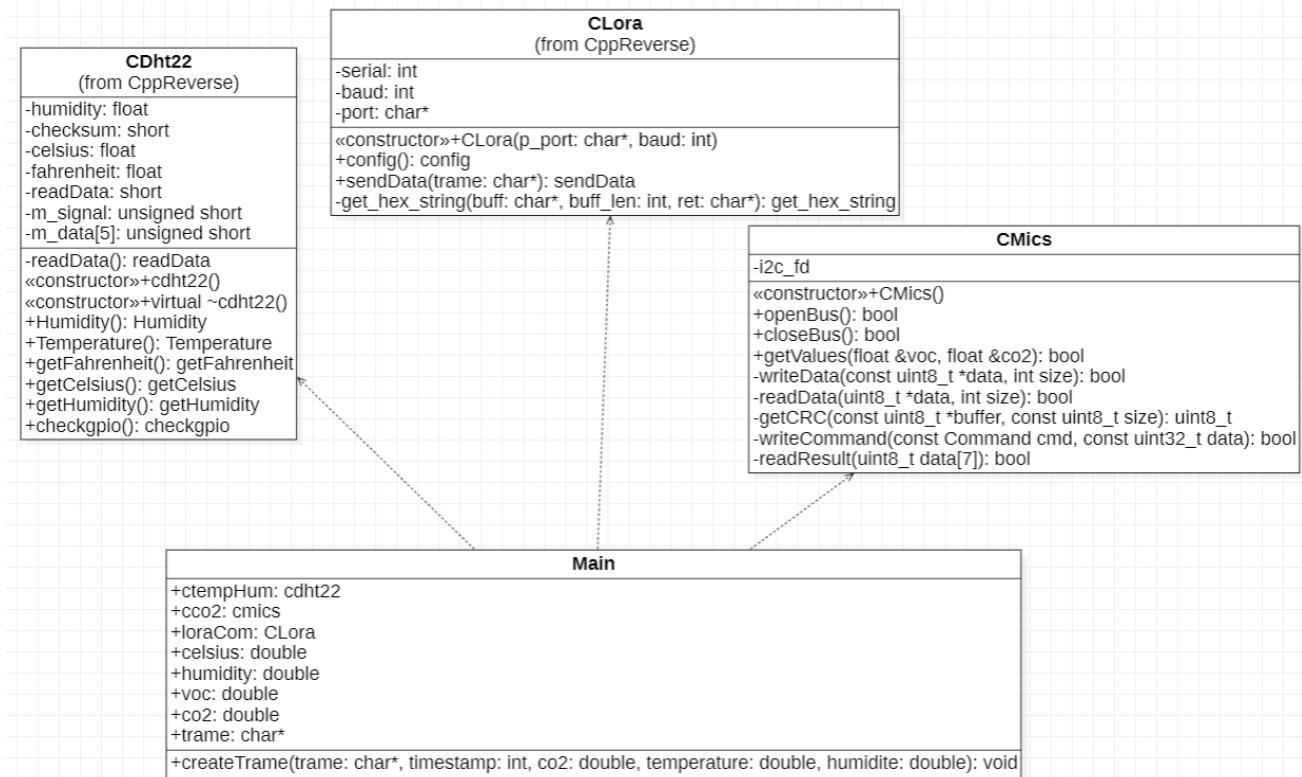
```
void CLora::sendData(char *trame) {  
    char payload[strlen(trame)*2];  
    get_hex_string(trame, strlen(trame), payload);  
  
    char *cmd = "max tx cnf 4 ";  
    strcat(cmd, payload);  
    strcat(cmd, "\r\n");  
  
    serialPuts(serial, cmd);  
}
```

Méthode « get_hex_string(char *buff, int buff_len, char *ret)» : cette méthode va nous servir à convertir la trame qu'on va envoyer en hexadécimal ASCII

```
void CLora::get_hex_string(char *buff, int buff_len, char *ret) {  
    int j; //index of buff  
    int i; //index of string  
    char nibble;  
    const char hex_map[16] = {'0', '1', '2', '3', '4', '5', '6', '7', '8', '9', 'A', 'B', 'C', 'D', 'E', 'F'};  
  
    for (i = 0, j = 0; i < buff_len*2; i++) {  
        nibble = 20;  
  
        if (i % 2 == 0) nibble = (buff[j] >> 4) & 0x0F;  
        else{  
            nibble = buff[j] & 0x0F;  
            j++;  
        }  
        ret[i] = hex_map[nibble];  
    }  
    return;  
}
```

Présentation du programme principale

Modélisation du programme principale



Fonction « createTrame() » : cette fonction va nous servir à mettre dans la chaîne de caractère "trame" les données sous ce format : "**timestamp;co2;temperature;humidite**". L'Étudiant 3 a besoin de recevoir la trame sous cette forme afin de pouvoir traiter les données.

```

void createTrame(char *trame, int timestamp, double co2, double temperature, double humidite) {
    char timestampChar[15];
    char co2Char[15];
    char tempChar[15];
    char humiditeChar[15];

    sprintf(timestampChar, "%d", timestamp);
    sprintf(co2Char, "%.0lf", co2);
    sprintf(tempChar, "%.2lf", temperature);
    sprintf(humiditeChar, "%.2lf", humidite);

    strcat(trame, timestampChar);
    strcat(trame, ";");
    strcat(trame, co2Char);
    strcat(trame, ";");
    strcat(trame, tempChar);
    strcat(trame, ";");
    strcat(trame, humiditeChar);
}

```

Explication en détail du programme principal final :

```

int main(void)
{
    cdht22 *ctempHum = new cdht22();
    cmics *cco2 = new cmics;
    CLora *loraCom = new CLora("/dev/ttyAMA0", 57600);
    ctempHum->checkgpio();
    loraCom->config();
    char trame[150] = "";
    double celsius = 0;
    double humidity = 0;
    double voc = 0 ;
    double co2 = 0 ;
    while(1){
        ctempHum->Humidity();
        ctempHum->Temperature();
        celsius = ctempHum->getCelsius();
        humidity = ctempHum->getHumidity();
        if (!cco2->openBus()) return 1;
        if (!cco2->getValues(voc, co2)) return 1;
        createTrame(trame, (int)std::time(nullptr), co2, celsius, humidity);
        loraCom->sendData(trame);

        cco2->closeBus();
        delay(600000);
    }
    return 0;
}

```

Initialisation des objets

Vérification de la connexion gpio

Configuration du module radio

Initialisations des variables

Acquisitions des données

Création de la trame et envoie des données

Délai de 10 minutes entre chaque envoi

Création d'un service qui permettra à ce programme d'être constamment allumé en tâche de fond sur le Raspberry :

Fichier de configuration du service :

```

GNU nano 3.2                               noeudCO2.service

[Unit]
Description= Service pour la reception et l'envoie des données du noeud co2

[Service]
Type=simple
WorkingDirectory=/home/pi/
ExecStart=./noeudco2.exe

[Install]
WantedBy=multi-user.target

```

Activation et lancement du service :

```

pi@ttn-gateway:~ $ sudo nano noeudCO2.service
pi@ttn-gateway:~ $ sudo cp noeudCO2.service /etc/systemd/system
pi@ttn-gateway:~ $ sudo systemctl enable noeudCO2.service
Created symlink /etc/systemd/system/multi-user.target.wants/noeudCO2.service → /etc/systemd/
system/noeudCO2.service.
pi@ttn-gateway:~ $ sudo systemctl start noeudCO2.service

```

Difficultés rencontrées

Problème numéros 1

Problème rencontré avec le capteur Humidité / température au niveau de la lecture. À cause du module rtc qui utilise le même port GPIO que le capteur humidité / température, il y avait un conflit pendant la lecture des données du capteur. J'ai donc ma méthode "CheckGpio()" qui me renvoyait donc ce message d'erreur

```
[x_x] GPIO Initialization FAILED.
```

J'ai dû donc changer le port du capteur de "GPIO 18" à "GPIO 17" pour régler ce problème. Pour vérifier si tout fonctionne parfaitement, j'ai utilisé ma méthode "checkGpio()" et donc constater que le problème était résolu.

```
GPIO Initialization SUCCEED.
```

Problème numéros 2

Problème avec le Raspberry PI 3 parce qu'il n'y avait pas la librairie WiringPi dans la dernière version de l'os buster.

```
ppp and wiringpi (gpio tool) installing...
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
E: Unable to locate package wiringpi
Process failed
```

J'ai ainsi dû la réinstaller avec cette commande

```
sudo apt-get install wiringpi
```

Étudiant 3 (ROUX Mathieu) :

Expression du besoin

Ma partie consiste à la réception, le stockage puis l'affichage des données.

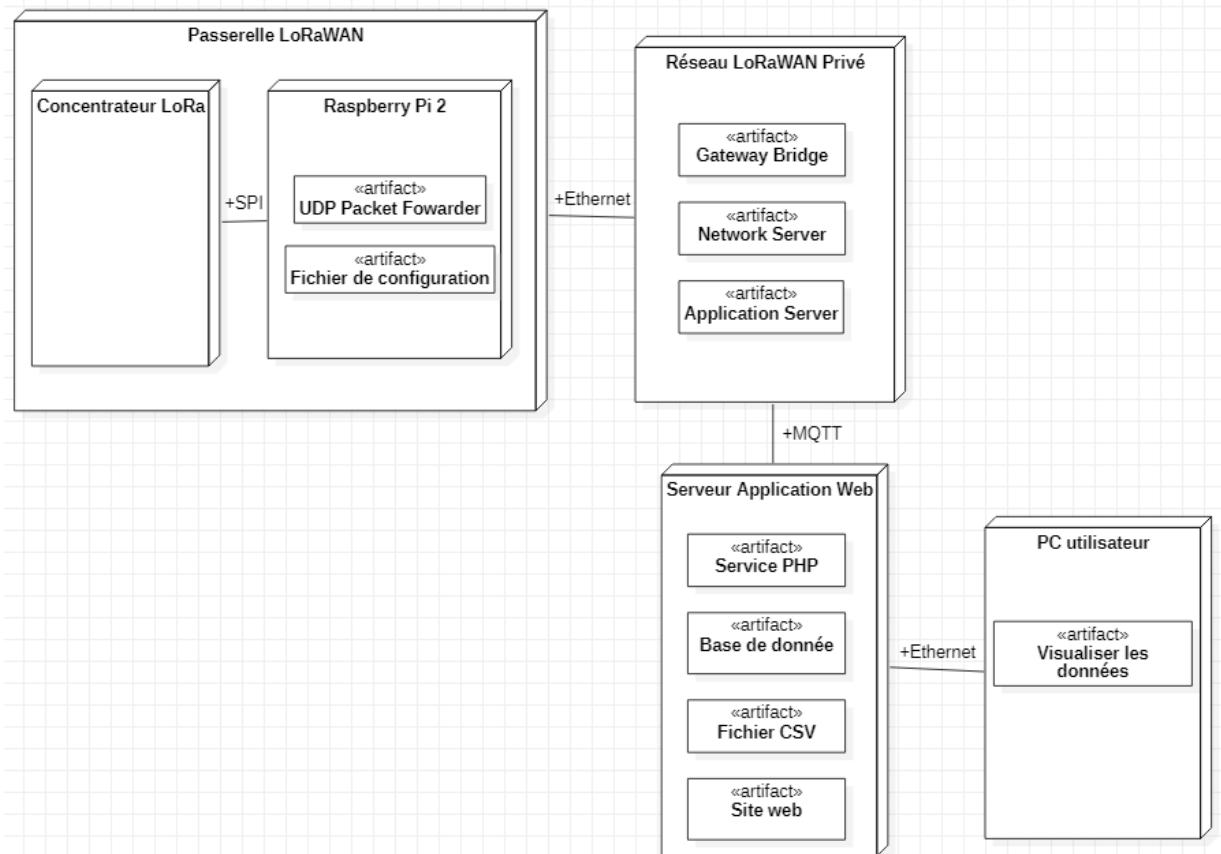
Les étudiants n°1 et n°2 possèdent chacun un nœud qui envoie des données par radio fréquence à une passerelle LoRaWAN. Je dois donc mettre en place cette passerelle qui communiquera par Ethernet à une réseau privé Chirpstack, qui lui-même communiquera par protocole MQTT à mon application web.

Je dois stocker les données reçues dans une base de données, sous format csv et les afficher dynamiquement et en temps réel sur une page web.

Prototypage

Notre passerelle LoRaWAN, Réseau LoRaWAN Privé et Serveur Application Web seront les trois sur des Raspberry Pi 2 ou Raspberry Pi 3 comme indiqué dans les contraintes matérielles.

Le système répond au diagramme de déploiement suivant:



Conception Préliminaire

Passerelle LoRaWAN

Pour faire office de passerelle LoRaWAN, nous avons un concentrateur LoRa ic880a et un Raspberry Pi 2.

Nous aurons donc besoin d'installer le service ttn-gateway correspondant à ce concentrateur, et de configurer la passerelle sur notre Chirpstack (Serveur Privé).

Serveur Privé LoRaWAN

Pour notre Serveur privé LoRaWAN, nous allons utiliser le serveur Chirpstack qui présente de nombreux avantages:

- Prise en main facile
- Open-source
- Totalement modulable
- Interface web pour l'affichage des trames et la configuration des nœuds
- Broker MQTT intégré

Lorsque Chirpstack reçoit des données, il les publie à un topic MQTT automatiquement.

Serveur Application Web

Pour notre Serveur Application Web, nous utiliserons un Raspberry Pi 3 sur lequel nous aurons les services suivants:

- Apache2 (Serveur web)
- Mosquitto (Broker MQTT)
- SQLite (Base de donnée)

Pour traiter la réception et le stockage de données sans interruption, nous aurons besoin de créer un service qui appellera un programme PHP qui devra:

- Réceptionner les données en MQTT
- Écrire les données sous format CSV
- Stocker les données dans une base de donnée

Pour l'affichage des données en temps réel sur une page web, nous aurons un programme PHP qui réceptionnera également les données en MQTT mais se contentera de les actualiser sur la page.

Le fait de séparer ces deux fonctionnements nous permet de garder la réception et le stockage des données en marche continue. Sinon, il aurait fallu avoir constamment un utilisateur présent sur la page web.

Toutes les données seront stockées à la fois dans un fichier csv correspondant au nœud, mais également à sa table associés dans notre base de données.

La base de donnée sera composée des deux tables suivantes:

co2		boopy	
timestamp	text	timestamp	text
co2	text	luminosite	text
temperature	text	temperature	text
humidite	text	humidite	text
		pressionAtm	text
		longitude	text
		latitude	text
		altitude	text

Pour faciliter le stockage et le programme, aucune données ne sera converties en entier ou en flottant pour être envoyées dans la base de données. En effet, la trame reçue par MQTT sera une chaîne de caractère, nous aurons donc simplement besoin de séparer les valeurs.

Nous aurons également 2 fichiers csv:

- co2.csv pour les données du noeud Co2
- boopy.csv pour les données du noeud Boopy

Notre serveur application web fonctionnera donc ainsi:

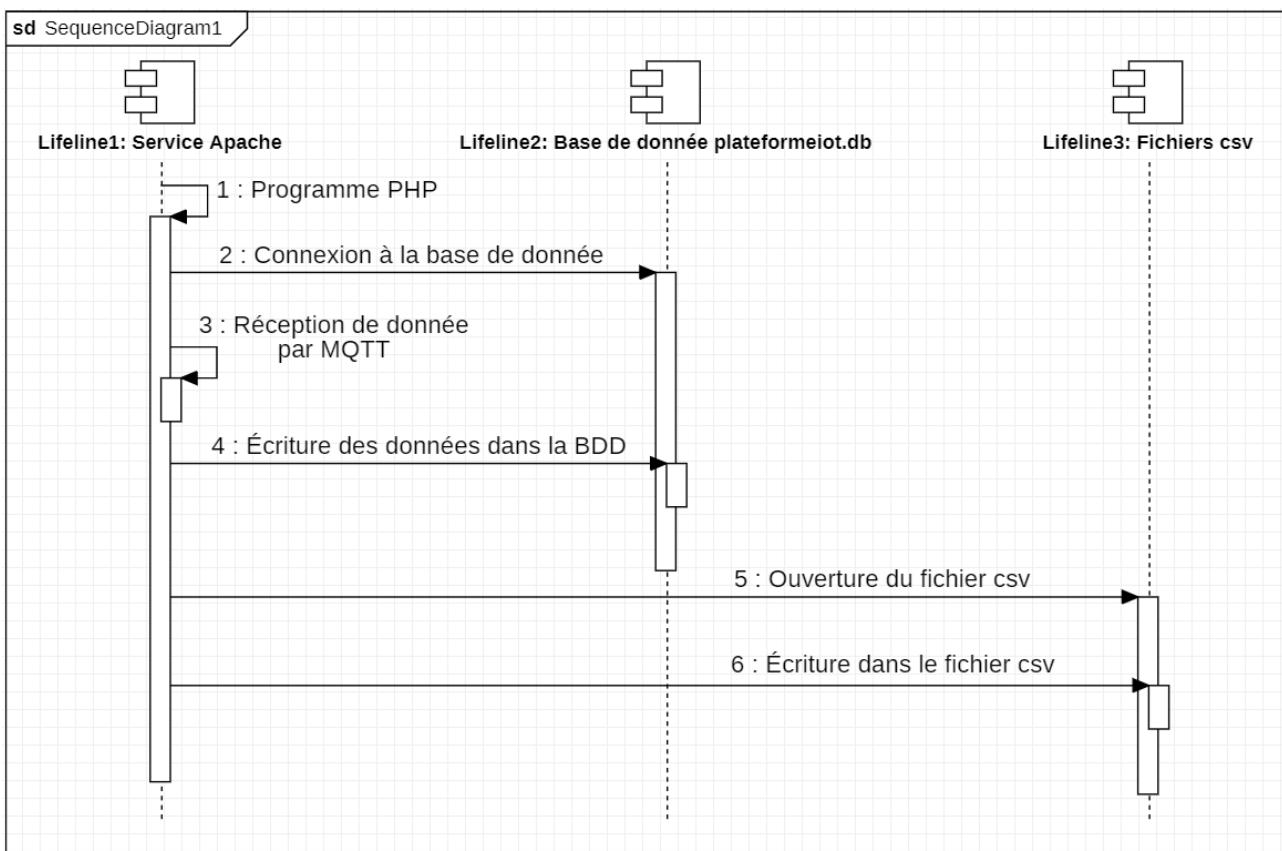
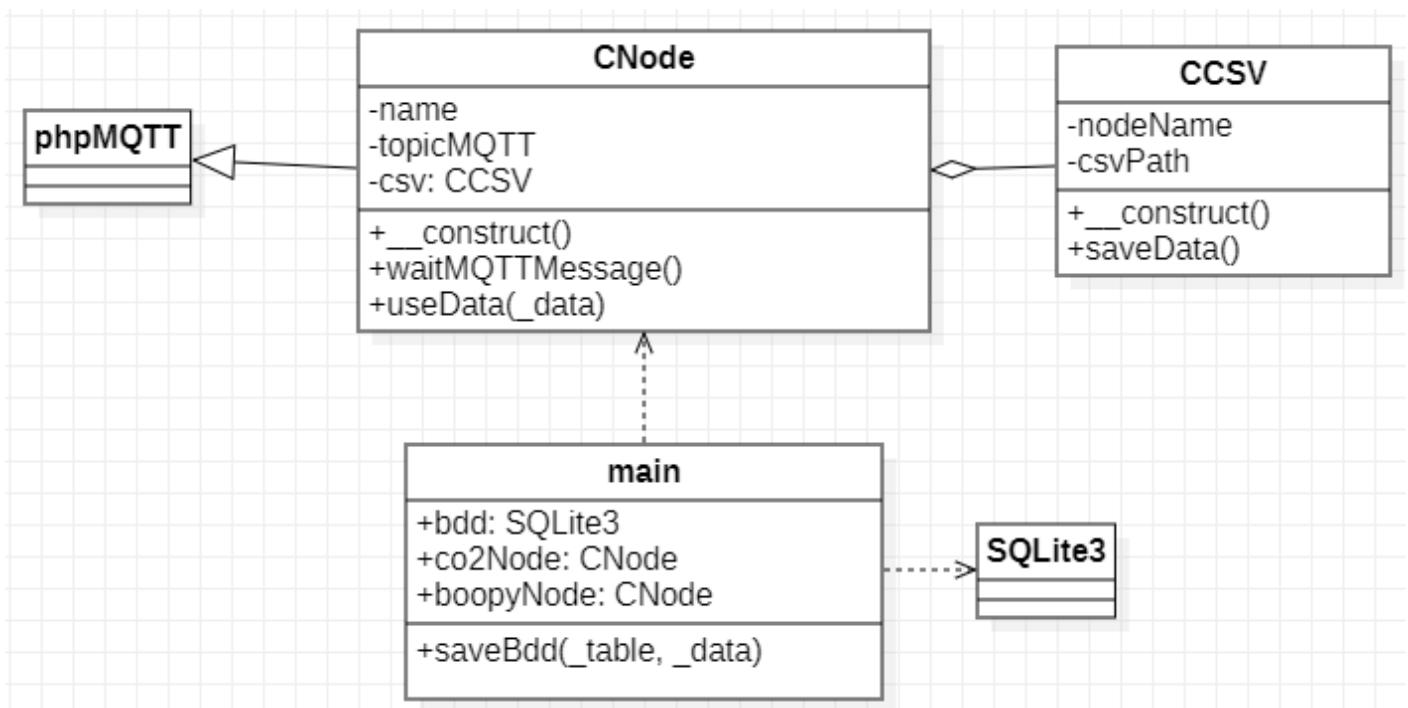
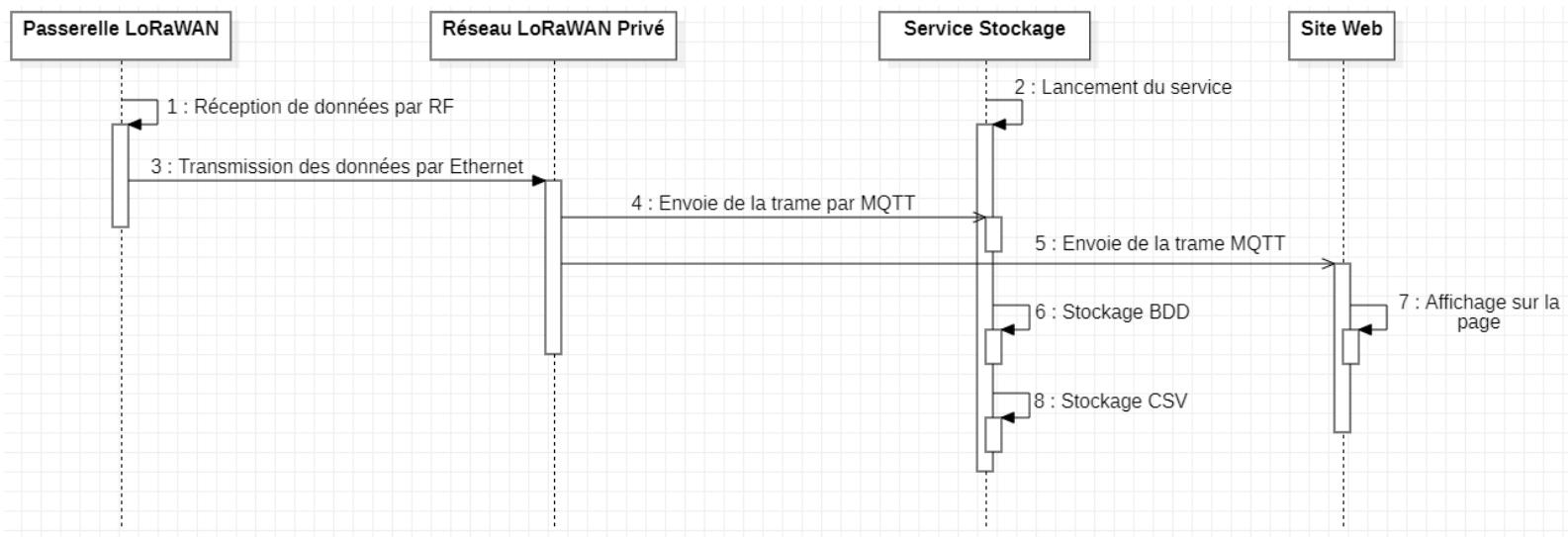


Diagramme de classe prévisionnel du système de réception-stockage:



Fonctionnement entre les objets



Conception détaillé

Pour les trois raspberry Pi, nous faisons les mêmes installation de base:

- Raspbian OS
- Activations des services SSH
- Configuration d'une adresse IP Fixe

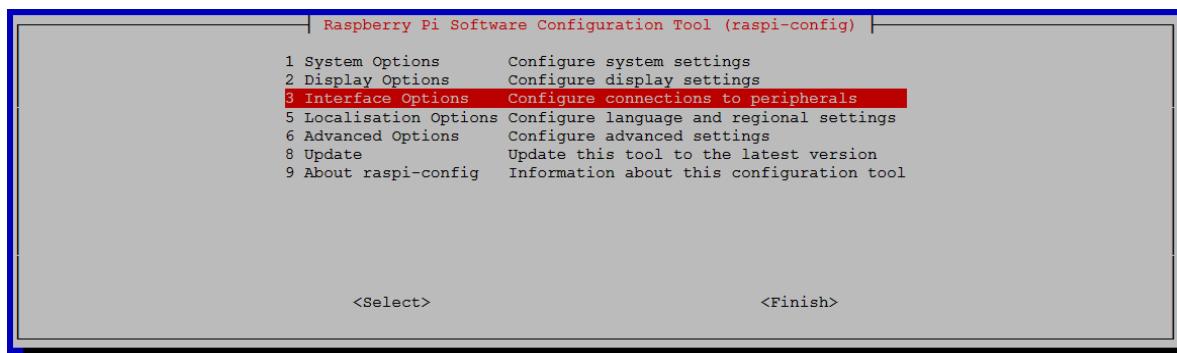
Passerelle LoRaWAN

Nous disposons d'un concentrateur LoRa ic880a communiquant par port série avec notre Raspberry Pi. Nous devons donc le configurer pour qu'ils puissent communiquer.

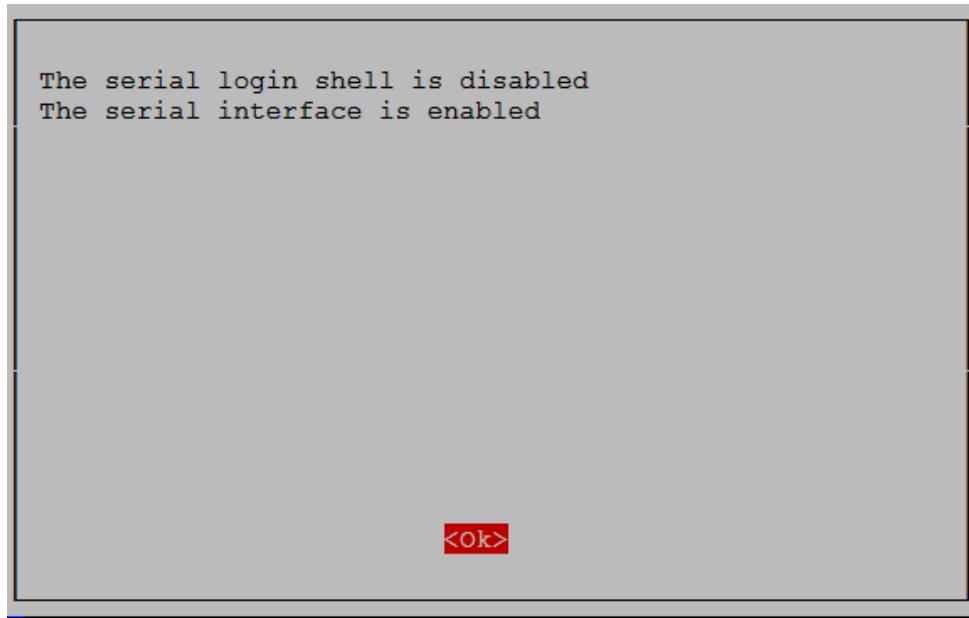
Pour activer le port Série, il suffit de faire la commande:

sudo raspi-config

Cette commande permet d'ouvrir l'interface graphique suivante:



On se rend dans le menu « 3 Interface Options » et on active le « Serial Port ».



Pour installer notre service ttn-gateway, nous avons besoin de git.

Pour installer git on utilise la commande:

sudo apt-get install git

```
pi@ttn-gateway:~ $ git clone https://github.com/ttn-zh/ic880a-gateway.git
Clonage dans 'ic880a-gateway'...
remote: Enumerating objects: 361, done.
remote: Counting objects: 100% (3/3), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 361 (delta 0), reused 1 (delta 0), pack-reused 358
Réception d'objets: 100% (361/361), 7.89 MiB | 9.73 MiB/s, fait.
Résolution des deltas: 100% (212/212), fait.
```

On lance le fichier d'installation et on obtient le EUI de notre concentrateur LoRa:

B827EBFFFEE6F7AC

```
^[[B^[[BDéjà à jour.
Gateway configuration:
Detected EUI B827EBFFFEE6F7AC from eth0
Do you want to use remote settings file? [y/N]_
```

Cet EUI ID nous permettra de configurer la passerelle sur notre Réseau LoRaWAN Privé.

On peut maintenant vérifier que notre service est actif avec la commande:

service ttn-gateway status

```

pi@ttn-gateway:~/ic880a-gateway $ sudo service ttn-gateway status
● ttn-gateway.service - The Things Network Gateway
  Loaded: loaded (/lib/systemd/system/ttn-gateway.service; enabled; vendor preset: enabled)
  Active: active (running) since Mon 2022-05-23 19:44:31 CEST; 12s ago
    Main PID: 7979 (start.sh)
      Tasks: 4 (limit: 4654)
     Memory: 1.0M
        CPU: 0.000 CPU(s) since start
       CGroup: /system.slice/ttn-gateway.service
                 └─7979 /bin/bash /opt/ttn-gateway/bin/start.sh
                   ├─7980 /bin/bash /opt/ttn-gateway/bin/start.sh
                   ├─7981 ping -c1 google.com
                   └─7982 grep 0% packet loss

mai 23 19:44:31 ttn-gateway systemd[1]: Started The Things Network Gateway.

```

Notre passerelle LoRaWAN est donc complètement installée.

Réseau LoRaWAN Privé

Installation de Chirpstack

Nous allons utiliser un serveur Chirpstack. Pour l'installer, nous avons besoin de git et de Docker qui est un gestionnaire de paquet et de service.

Nous allons installer le package Chirpstack docker avec git:

```

pi@raspberry:~ $ git clone https://github.com/brocaar/chirpstack-docker.git
Clonage dans 'chirpstack-docker'...

```

Une fois installé, pour démarrer Chirpstack nous utilisons la commande suivante dans le répertoire /chirpstack-docker:

[sudo docker-compose up](#)

```

cs3@cs3-VirtualBox:~/chirpstack-docker$ sudo docker-compose up
[sudo] Mot de passe de cs3 :
Starting chirpstack-docker_redis_1 ... done
Starting chirpstack-docker_postgresql_1 ... done
Starting chirpstack-docker_mosquitto_1 ... done
Starting chirpstack-docker_chirpstack-gateway-bridge_1 ... done
Starting chirpstack-docker_chirpstack-network-server_1 ... done
Starting chirpstack-docker_chirpstack-application-server_1 ... done

```

Configuration de Chirpstack

Pour configurer Chirpstack, on doit se connecter sur l'interface web.
L'adresse est 10.121.41.128:8080.

On utilise l'identifiant **admin** et le mot de passe **admin**.

En premier lieu, nous configurons le Network Server comme ceci:

The screenshot shows the 'Network-servers / main_Network (EU868 @)' configuration page. It has three tabs: GENERAL (selected), GATEWAY DISCOVERY, and TLS CERTIFICATES. Under the GENERAL tab, there are two fields: 'Network-server name *' containing 'main_Network' and a description 'A name to identify the network-server.' Below it is another field 'Network-server server *' containing 'chirpstack-network-server:8000' with a description 'The 'hostname:port' of the network-server, e.g. 'localhost:8000''. Both fields have a red asterisk indicating they are required.

Avant de configurer la passerelle, on doit créer un « Service Profile » :

The screenshot shows the 'Service-profiles / profile1' configuration page. It has several settings:

- 'Service-profile name *' set to 'profile1' with a description 'A name to identify the service-profile.'
- A checked checkbox 'Add gateway meta-data' with a description 'GW metadata (RSSI, SNR, GW geoloc., etc.) are added to the packet sent to the application-server.'
- An unchecked checkbox 'Enable network geolocation' with a description 'When enabled, the network-server will try to resolve the location of the devices under this service-profile supporting the fine-timestamp feature and that the network-server needs to be configured in a specific way.'
- 'Device-status request frequency' set to '0' with a description 'Frequency to initiate an End-Device status request (request/day). Set to 0 to disable.'
- 'Minimum allowed data-rate *' set to '0' with a description 'Minimum allowed data rate. Used for ADR.'
- 'Maximum allowed data-rate *' set to '5' with a description 'Maximum allowed data rate. Used for ADR.'
- An unchecked checkbox 'Private gateways'

Un « Service Profile » permet de définir les fonctionnalités de « l'utilisateur » sur le réseau.

Nous pouvons maintenant configurer la passerelle.

1. Création d'un profile de Gateway

Name *

Gateway

A short name identifying the gateway-profile.

Stats interval (seconds) *

1

The stats interval in which the gateway reports its statistics. The recommended value is 30 seconds.

Enabled channels *

0, 1, 2, 3, 4, 5, 6, 7, 8

The channels active in this gateway-profile as specified in the LoRaWAN Regional Parameters specification. Separate channels by comma, e.g. 0, 1, 2. Extra channels must not be included in this list.

Network-server *

main_Network

[ADD EXTRA CHANNEL](#) [CREATE GATEWAY-PROFILE](#)

2. Création de la Gateway

Le « Gateway ID » correspond au EUI de la passerelle vu précédemment:

« **B827EBE6F7ACFFFF** »

Gateways / Create

GENERAL	TAGS	METADATA
Gateway name *		
Gw		
The name may only contain words, numbers and dashes.		
Gateway description *		
Gateway		
<hr/>		
Gateway ID *		
b8 27 eb e6 f7 ac ff ff		
<hr/>		
Network-server *		
main_Network		
Select the network-server to which the gateway will connect. When no network-servers are available in this organization.		
Service-profile		
profile1		
Select the service-profile under which the gateway must be added. The available service-profiles depend on the selected network-server.		
Gateway-profile		
gateway_profile		
Optional. When assigning a gateway-profile to the gateway, ChirpStack Network Server will attempt to automatically assign a gateway with ChirpStack Concentrator.		

Une fois la passerelle configurée, nous devons configurer un « Device profile » pour que les nœuds puissent communiquer par ABP.

Nous utilisons les paramètres suivants:

Device-profiles / ABP

GENERAL	JOIN (OTAA / ABP)	CLASS-B
Device-profile name *	ABP	
	A name to identify the device-profile.	
LoRaWAN MAC version *	1.0.0	
	The LoRaWAN MAC version supported by the device.	
LoRaWAN Regional Parameters revision *	A	
	Revision of the Regional Parameters specification supported by the device.	
ADR algorithm *	Default ADR algorithm (LoRa only)	
	The ADR algorithm that will be used for controlling the device data-rate.	
Max EIRP *	0	
	Maximum EIRP supported by the device.	
Uplink interval (seconds) *	1	
	The expected interval in seconds in which the device sends uplink messages. This	

Sur Chirpstack, les trames sont codées en base64, pour les décoder avant de la traiter, on ajoute le programme JavaScript suivant dans l'onglet « CODEC »:

```
6 function Decode(fPort, bytes) {
7     var trame = String.fromCharCode.apply(String, bytes);
8     return {
9         data_decoded:
10            {
11                "trame":trame,
12            }
13    }
14 }
```

Lorsque Chirpstack reçoit une trame, il renvoie automatiquement toutes les données automatiquement sur le topic MQTT automatiquement.

Maintenant, Chirpstack est totalement configuré, on peut ajouter des nœuds fonctionnement en ABP.

Serveur Application Web

Le Serveur Application Web doit réceptionner puis stocker les données dans une base de données et sous format CSV.

On aura également un site web avec les données affichées en temps réel.

Installation des services:

On a besoin d'installer:

- Apache2 en tant que serveur web
- SQLite3 pour la base de donnée
- PHP, libapachePHP et phpSQLite pour PHP et les librairies compatibles
- Un broker MQTT

1. Installation de Apache2

Apache2 est un service de serveur web multiplateforme open-source. Il nous permettra d'héberger nos différents fichiers HTML et PHP.

Pour installer Apache2, on utilise la commande: **sudo apt-get install apache2 -y**

On vérifie l'installation comme ceci:

```
^Cpi@raspberrypi:~ $ sudo service apache2 status
lines 1--1... skipping...
● apache2.service - The Apache HTTP Server
    Loaded: loaded (/lib/systemd/system/apache2.service; enabled; preset=)
    Active: active (running) since Fri 2022-05-20 18:17:15
      Docs: https://httpd.apache.org/docs/2.4/
   Process: 521 ExecStart=/usr/sbin/apachectl start (code=exited, status=0)
 Main PID: 581 (apache2)
    Tasks: 21 (limit: 1598)
      CPU: 2min 22.804s
     CGrou...
```

2. Installation de SQLite3

Pour la base de donnée, on installe SQLite avec la commande suivante:

sudo apt-get install sqlite3

On crée un dossier databases dans lequel on créer une base de donnée avec la commande suivante: **sqlite3 plateformeiot.db**

On crée nos tables avec les requêtes suivantes:

```
sqlite> CREATE TABLE co2 (
...> timestamp TEXT,
...> co2 TEXT,
...> temperature TEXT,
...> humidite TEXT
...> );
sqlite> .tables
co2
sqlite> CREATE TABLE boopy (
...> timestamp TEXT,
...> luminosite TEXT,
...> temperature TEXT,
...> humidite TEXT,
...> pressionAtm TEXT,
...> longitude TEXT,
...> latitude TEXT,
...> altitude TEXT
...> );
sqlite>
sqlite> .tables
boopy  co2
```

3. Installation de PHP et des librairies

Pour installer PHP et qu'il soit compatible avec notre serveur Apache, on tape la commande suivante:

sudo apt-get install php

On vérifie que l'installation est bonne:

```
pi@raspberrypi:~ $ php -v
PHP 7.4.28 (cli) (built: Feb 17 2022 16:17:19) ( NTS )
Copyright (c) The PHP Group
Zend Engine v3.4.0, Copyright (c) Zend Technologies
    with Zend OPcache v7.4.28, Copyright (c), by Zend Technologies
```

On possède donc PHP 7.4, on installe donc les librairies associées à notre version:

sudo apt-get install php7.4 libapache2-mod-php7.4 php7.4-sqlite -y

Puis on relance notre service apache2 avec la commande: **sudo service apache2 restart**

4. Installation du broker MQTT Mosquitto

Le protocole MQTT est un protocole de messagerie qui assure des communications non permanentes entre des appareils. Il se repose sur TCP/IP et permet de transmettre des données généralement entre des objets connectés.

Ce protocole est composé de deux types de clients:

- Les publisher qui envoient des messages sur un topic
- Les subscriber qui réceptionnent les messages des publisher sur un topic

Nous utiliserons ce protocole pour communiquer entre notre Réseau LoRaWAN privé et notre Serveur Application Web.

On installe le broker MQTT « Mosquitto » et le client « Mosquitto Client » qui permettra de tester le broker:

sudo apt install mosquitto mosquitto-clients

On test le broker:

Publisher:

```
pi@raspberrypi:~ $ mosquitto_pub -h localhost -t topic -m message -r  
pi@raspberrypi:~ $ _
```

Le paramètre « -r » permet de laisser le message en « retenu » dans le topic.

Subscriber:

```
pi@raspberrypi:~ $ mosquitto_sub -h localhost -t topic  
message
```

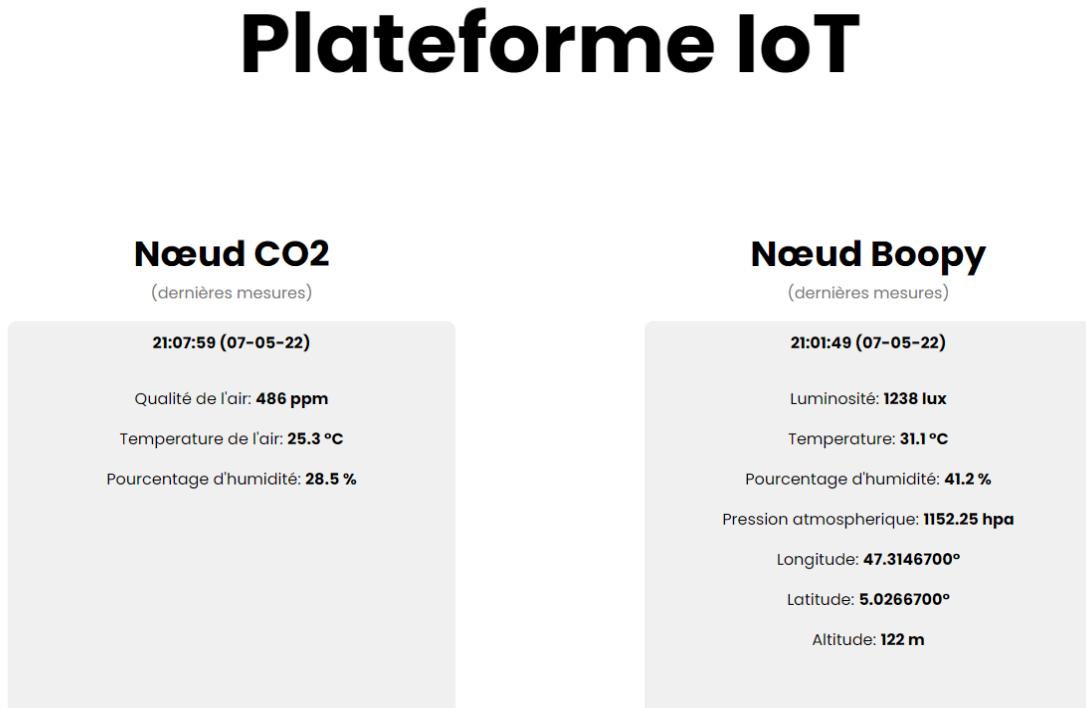
Notre installation est correcte.

a) La page web

Maintenant que tous nos services sont installés et fonctionnels, on peut commencer à créer le site web.

Il doit afficher les dernières données envoyées par les deux nœuds et en temps réel.

J'ai créé la page HTML suivante:



Dans nos carrés gris, il est chargé un fichier .php dans lequel on affiche les données récupérées à l'aide d'un subscriber MQTT.

Pour le rechargement constant de la page, on utilise la librairie Javascript « JQuery » qui permet d'interagir avec le DOM de la page.

Le script utilisé est le suivant:

```
28      <script src="jquery-3.6.0.min.js"></script>
29      <script>
30          $(document).ready(_ => {
31              $('.CO2').load('co2MQTT.php');
32              $('.BOOPY').load('boopyMQTT.php');
33              refresh();
34          });
35
36          function refresh() {
37              setTimeout(_ => {
38                  $('.CO2').load('co2MQTT.php');
39                  $('.BOOPY').load('boopyMQTT.php');
40                  refresh();
41              }, 10000);
42          }
43      </script>
```

Dans ce script, on charge les fichiers php une fois que la page est entièrement prête. Une fois chargés, on appelle notre fonction **refresh()** qui s'exécute au bout de 10 secondes, et qui s'appelle elle-même continuellement, ça nous permet d'avoir une boucle continue.

Les trames reçues par MQTT sont sous la forme suivante (Exemple avec le CO2):

timestamp ; qualité de l'air ; température ; humidité

Exemple du script PHP pour le noeud CO2: co2MQTT.php

```
1  <?php
2
3  require('phpMQTT.php');
4  require('displayData.php');
5
6  $co2Topic = 'application/3/device/70b3d57ed0048d3e/event/up';
7
8  $co2MQTT = new Bluerhinos\phpMQTT('10.121.41.128', 1883, 'co2ID');
9
10 if(!$co2MQTT->connect(true, NULL, 'co2Username', 'co2Password')) exit(1);
11
12 $co2AllData = $co2MQTT->subscribeAndWaitForMessage($co2Topic, 0);
13
14 // On récupère uniquement les valeurs qui nous interesse
15 $co2Trame = explode('trame":', $co2AllData);
16 $co2Trame = substr($co2Trame[1], 0, -4);
17
18 // On les sépare sous forme de tableau
19 $co2Array = explode(';', $co2Trame);
20
21
22 webDisplayData($co2Array, 'co2');
23
24 $co2MQTT->close();
25 |
26 ?>
```

Les deux nœuds fonctionnent de la même manière.

Le fichier « `phpMQTT.php` » est une classe disponible afin de créer un subscriber MQTT.

La fonction **webDisplayData()** permet d'afficher les données sur le site:

```

3   function webDisplayData($_data, $_node)
4 {
5     $date = new DateTime();
6     switch($_node){
7       case 'co2':
8         $date->setTimestamp(intval($_data[0]));
9         echo '<span class="date"><b>' . $date->format('G:i:s') .
10          " (" . $date->format('d-m-y') . ")</b></span>';
11        echo '<p>Qualité de l\'air: <b>' . $_data[1] . ' ppm</b></p>';
12        echo '<p>Temperature de l\'air: <b>' . $_data[2] . ' °C</b></p>';
13        echo '<p>Pourcentage d\'humidité: <b>' . $_data[3] . ' %</b></p>';
14      break;
}

```

Création d'une instance
DateTime

Défini la date selon timestamp

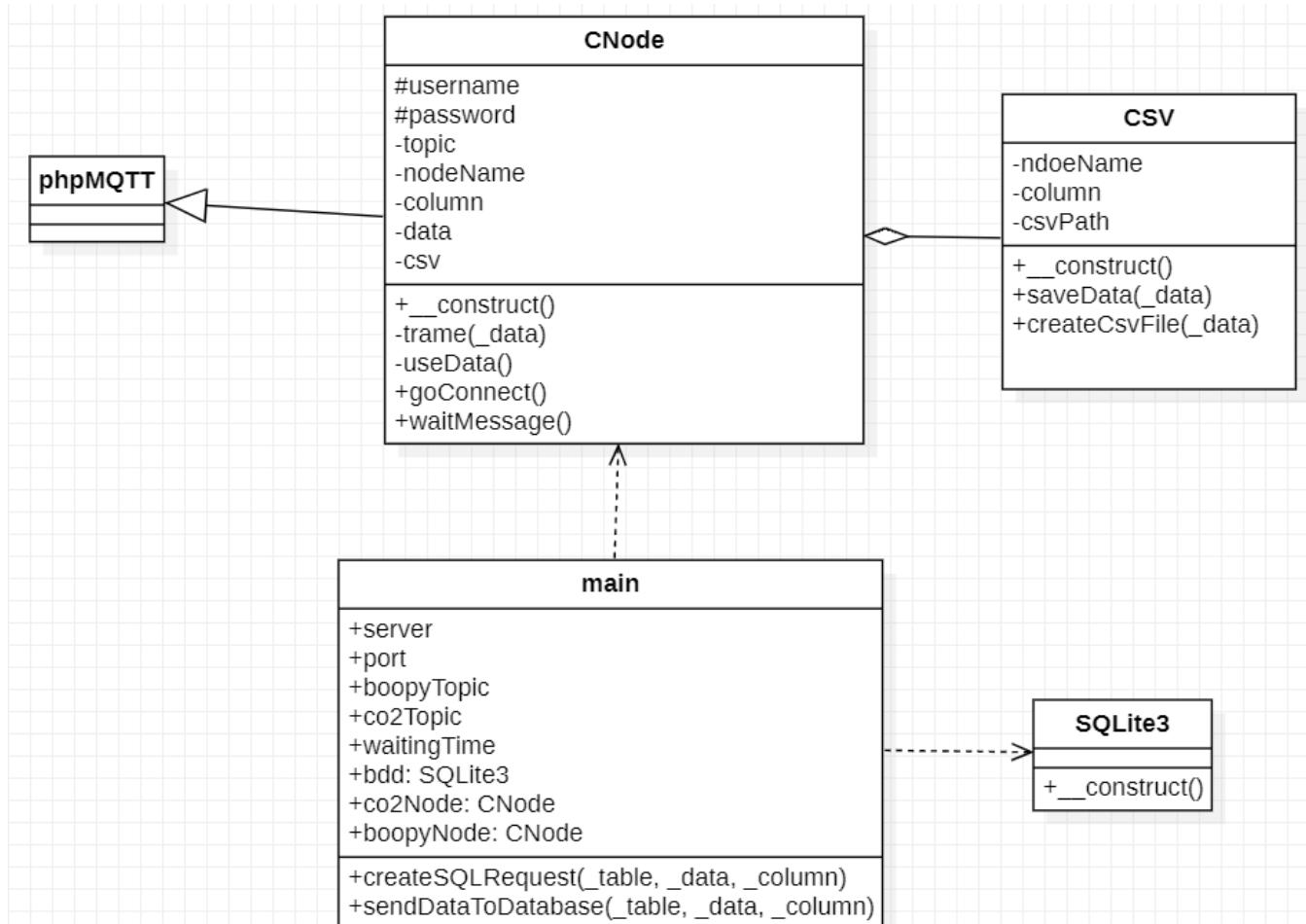
Affiche les données sur le site
web

b) Le service

La partie la plus importante du Serveur Application Web est le stockage des données dans la base de données et sous format CSV comme l'a demandé la société Tenum.

Pour cela, j'ai créé un programme PHP qui devra tourner continuellement sur le Raspberry afin de réceptionner et stocker les données.

Le programme suit le diagramme de classe suivant:



J'ai créer premièrement une classe CNode qui hérite de phpMQTT:

```
7  class CNode extends Bluerhinos\phpMQTT{  
8  
9      protected $username;  
10     protected $password;  
11     private $topic;  
12     private $nodeName;  
13     private $column;  
14     private $data;  
15     private $csv;  
16  
17     public function __construct($_server, $_port, $_username, $_password,  
18                               $_id, $_topic, $_nodeName, $_column) {  
19  
20         parent::__construct($_server, $_port, $_id);  
21  
22         $this->username = $_username;  
23         $this->password = $_password;  
24  
25         $this->topic = $_topic;  
26         $this->nodeName = $_nodeName;  
27         $this->column = $_column;  
28  
29         $this->csv = new CSV($this->nodeName, $this->column);  
30     }  
31  
32     private function trame($_data) {  
33         $trame = explode('trame":', $_data);  
34         $trame = substr($trame[1], 0, -4);  
35  
36         return $trame;  
37     }  
38  
39     private function useData() {  
40         // Créer un tableau de donnée  
41         $arrayData = explode(";", $this->data);  
42         // Sauvegarder dans le CSV  
43         $this->csv->saveData($arrayData);  
44         // Sauvegarder dans la base de données  
45         sendDataToDatabase($this->nodeName, $arrayData, $this->column);  
46     }  
47  
48     public function goConnect() {  
49         if(!parent::connect(true, NULL, $this->username, $this->password)) exit(1);  
50     }  
51  
52     public function waitMessage() {  
53         $allData = parent::subscribeAndWaitForMessage($this->topic, 0);  
54         $this->data = $this->trame($allData);  
55         $this->useData();  
56         parent::close();  
57     }  
58  
59     private function sendDataToDatabase($nodeName, $arrayData, $column) {  
60         $db = new Database();  
61         $db->insert($arrayData, $column);  
62     }  
63  
64     private function saveData($arrayData) {  
65         $csv = new CSV($this->nodeName, $this->column);  
66         $csv->saveData($arrayData);  
67     }  
68  
69     private function close() {  
70         parent::close();  
71     }  
72  
73     private function __destruct() {  
74         $this->close();  
75     }  
76  
77     private function __clone() {  
78         $this->close();  
79     }  
80  
81     private function __wakeup() {  
82         $this->close();  
83     }  
84  
85     private function __sleep() {  
86         $this->close();  
87     }  
88  
89     private function __get($name) {  
90         if($name == "data") {  
91             $this->data = $this->trame($allData);  
92         }  
93         return $this->data;  
94     }  
95  
96     private function __set($name, $value) {  
97         if($name == "data") {  
98             $this->data = $value;  
99         }  
100    }  
101  
102    private function __call($name, $arguments) {  
103        if($name == "useData") {  
104            $this->useData();  
105        }  
106        if($name == "sendDataToDatabase") {  
107            $this->sendDataToDatabase($arguments[0], $arguments[1], $arguments[2]);  
108        }  
109        if($name == "close") {  
110            $this->close();  
111        }  
112        if($name == "trame") {  
113            $this->trame($arguments[0]);  
114        }  
115        if($name == "CSV") {  
116            $this->CSV($arguments[0], $arguments[1]);  
117        }  
118        if($name == "Database") {  
119            $this->Database($arguments[0]);  
120        }  
121        if($name == "subscribeAndWaitForMessage") {  
122            $this->subscribeAndWaitForMessage($arguments[0], $arguments[1]);  
123        }  
124        if($name == "insert") {  
125            $this->insert($arguments[0], $arguments[1]);  
126        }  
127        if($name == "close") {  
128            $this->close();  
129        }  
130        if($name == "get") {  
131            $this->get($arguments[0]);  
132        }  
133        if($name == "set") {  
134            $this->set($arguments[0], $arguments[1]);  
135        }  
136        if($name == "destruct") {  
137            $this->destruct();  
138        }  
139        if($name == "clone") {  
140            $this->clone();  
141        }  
142        if($name == "wakeup") {  
143            $this->wakeup();  
144        }  
145        if($name == "sleep") {  
146            $this->sleep();  
147        }  
148        if($name == "get") {  
149            $this->get($arguments[0]);  
150        }  
151        if($name == "set") {  
152            $this->set($arguments[0], $arguments[1]);  
153        }  
154        if($name == "destruct") {  
155            $this->destruct();  
156        }  
157        if($name == "clone") {  
158            $this->clone();  
159        }  
160        if($name == "wakeup") {  
161            $this->wakeup();  
162        }  
163        if($name == "sleep") {  
164            $this->sleep();  
165        }  
166        if($name == "get") {  
167            $this->get($arguments[0]);  
168        }  
169        if($name == "set") {  
170            $this->set($arguments[0], $arguments[1]);  
171        }  
172        if($name == "destruct") {  
173            $this->destruct();  
174        }  
175        if($name == "clone") {  
176            $this->clone();  
177        }  
178        if($name == "wakeup") {  
179            $this->wakeup();  
180        }  
181        if($name == "sleep") {  
182            $this->sleep();  
183        }  
184        if($name == "get") {  
185            $this->get($arguments[0]);  
186        }  
187        if($name == "set") {  
188            $this->set($arguments[0], $arguments[1]);  
189        }  
190        if($name == "destruct") {  
191            $this->destruct();  
192        }  
193        if($name == "clone") {  
194            $this->clone();  
195        }  
196        if($name == "wakeup") {  
197            $this->wakeup();  
198        }  
199        if($name == "sleep") {  
200            $this->sleep();  
201        }  
202        if($name == "get") {  
203            $this->get($arguments[0]);  
204        }  
205        if($name == "set") {  
206            $this->set($arguments[0], $arguments[1]);  
207        }  
208        if($name == "destruct") {  
209            $this->destruct();  
210        }  
211        if($name == "clone") {  
212            $this->clone();  
213        }  
214        if($name == "wakeup") {  
215            $this->wakeup();  
216        }  
217        if($name == "sleep") {  
218            $this->sleep();  
219        }  
220        if($name == "get") {  
221            $this->get($arguments[0]);  
222        }  
223        if($name == "set") {  
224            $this->set($arguments[0], $arguments[1]);  
225        }  
226        if($name == "destruct") {  
227            $this->destruct();  
228        }  
229        if($name == "clone") {  
230            $this->clone();  
231        }  
232        if($name == "wakeup") {  
233            $this->wakeup();  
234        }  
235        if($name == "sleep") {  
236            $this->sleep();  
237        }  
238        if($name == "get") {  
239            $this->get($arguments[0]);  
240        }  
241        if($name == "set") {  
242            $this->set($arguments[0], $arguments[1]);  
243        }  
244        if($name == "destruct") {  
245            $this->destruct();  
246        }  
247        if($name == "clone") {  
248            $this->clone();  
249        }  
250        if($name == "wakeup") {  
251            $this->wakeup();  
252        }  
253        if($name == "sleep") {  
254            $this->sleep();  
255        }  
256        if($name == "get") {  
257            $this->get($arguments[0]);  
258        }  
259        if($name == "set") {  
260            $this->set($arguments[0], $arguments[1]);  
261        }  
262        if($name == "destruct") {  
263            $this->destruct();  
264        }  
265        if($name == "clone") {  
266            $this->clone();  
267        }  
268        if($name == "wakeup") {  
269            $this->wakeup();  
270        }  
271        if($name == "sleep") {  
272            $this->sleep();  
273        }  
274        if($name == "get") {  
275            $this->get($arguments[0]);  
276        }  
277        if($name == "set") {  
278            $this->set($arguments[0], $arguments[1]);  
279        }  
280        if($name == "destruct") {  
281            $this->destruct();  
282        }  
283        if($name == "clone") {  
284            $this->clone();  
285        }  
286        if($name == "wakeup") {  
287            $this->wakeup();  
288        }  
289        if($name == "sleep") {  
290            $this->sleep();  
291        }  
292        if($name == "get") {  
293            $this->get($arguments[0]);  
294        }  
295        if($name == "set") {  
296            $this->set($arguments[0], $arguments[1]);  
297        }  
298        if($name == "destruct") {  
299            $this->destruct();  
300        }  
301        if($name == "clone") {  
302            $this->clone();  
303        }  
304        if($name == "wakeup") {  
305            $this->wakeup();  
306        }  
307        if($name == "sleep") {  
308            $this->sleep();  
309        }  
310        if($name == "get") {  
311            $this->get($arguments[0]);  
312        }  
313        if($name == "set") {  
314            $this->set($arguments[0], $arguments[1]);  
315        }  
316        if($name == "destruct") {  
317            $this->destruct();  
318        }  
319        if($name == "clone") {  
320            $this->clone();  
321        }  
322        if($name == "wakeup") {  
323            $this->wakeup();  
324        }  
325        if($name == "sleep") {  
326            $this->sleep();  
327        }  
328        if($name == "get") {  
329            $this->get($arguments[0]);  
330        }  
331        if($name == "set") {  
332            $this->set($arguments[0], $arguments[1]);  
333        }  
334        if($name == "destruct") {  
335            $this->destruct();  
336        }  
337        if($name == "clone") {  
338            $this->clone();  
339        }  
340        if($name == "wakeup") {  
341            $this->wakeup();  
342        }  
343        if($name == "sleep") {  
344            $this->sleep();  
345        }  
346        if($name == "get") {  
347            $this->get($arguments[0]);  
348        }  
349        if($name == "set") {  
350            $this->set($arguments[0], $arguments[1]);  
351        }  
352        if($name == "destruct") {  
353            $this->destruct();  
354        }  
355        if($name == "clone") {  
356            $this->clone();  
357        }  
358        if($name == "wakeup") {  
359            $this->wakeup();  
360        }  
361        if($name == "sleep") {  
362            $this->sleep();  
363        }  
364        if($name == "get") {  
365            $this->get($arguments[0]);  
366        }  
367        if($name == "set") {  
368            $this->set($arguments[0], $arguments[1]);  
369        }  
370        if($name == "destruct") {  
371            $this->destruct();  
372        }  
373        if($name == "clone") {  
374            $this->clone();  
375        }  
376        if($name == "wakeup") {  
377            $this->wakeup();  
378        }  
379        if($name == "sleep") {  
380            $this->sleep();  
381        }  
382        if($name == "get") {  
383            $this->get($arguments[0]);  
384        }  
385        if($name == "set") {  
386            $this->set($arguments[0], $arguments[1]);  
387        }  
388        if($name == "destruct") {  
389            $this->destruct();  
390        }  
391        if($name == "clone") {  
392            $this->clone();  
393        }  
394        if($name == "wakeup") {  
395            $this->wakeup();  
396        }  
397        if($name == "sleep") {  
398            $this->sleep();  
399        }  
400        if($name == "get") {  
401            $this->get($arguments[0]);  
402        }  
403        if($name == "set") {  
404            $this->set($arguments[0], $arguments[1]);  
405        }  
406        if($name == "destruct") {  
407            $this->destruct();  
408        }  
409        if($name == "clone") {  
410            $this->clone();  
411        }  
412        if($name == "wakeup") {  
413            $this->wakeup();  
414        }  
415        if($name == "sleep") {  
416            $this->sleep();  
417        }  
418        if($name == "get") {  
419            $this->get($arguments[0]);  
420        }  
421        if($name == "set") {  
422            $this->set($arguments[0], $arguments[1]);  
423        }  
424        if($name == "destruct") {  
425            $this->destruct();  
426        }  
427        if($name == "clone") {  
428            $this->clone();  
429        }  
430        if($name == "wakeup") {  
431            $this->wakeup();  
432        }  
433        if($name == "sleep") {  
434            $this->sleep();  
435        }  
436        if($name == "get") {  
437            $this->get($arguments[0]);  
438        }  
439        if($name == "set") {  
440            $this->set($arguments[0], $arguments[1]);  
441        }  
442        if($name == "destruct") {  
443            $this->destruct();  
444        }  
445        if($name == "clone") {  
446            $this->clone();  
447        }  
448        if($name == "wakeup") {  
449            $this->wakeup();  
450        }  
451        if($name == "sleep") {  
452            $this->sleep();  
453        }  
454        if($name == "get") {  
455            $this->get($arguments[0]);  
456        }  
457        if($name == "set") {  
458            $this->set($arguments[0], $arguments[1]);  
459        }  
460        if($name == "destruct") {  
461            $this->destruct();  
462        }  
463        if($name == "clone") {  
464            $this->clone();  
465        }  
466        if($name == "wakeup") {  
467            $this->wakeup();  
468        }  
469        if($name == "sleep") {  
470            $this->sleep();  
471        }  
472        if($name == "get") {  
473            $this->get($arguments[0]);  
474        }  
475        if($name == "set") {  
476            $this->set($arguments[0], $arguments[1]);  
477        }  
478        if($name == "destruct") {  
479            $this->destruct();  
480        }  
481        if($name == "clone") {  
482            $this->clone();  
483        }  
484        if($name == "wakeup") {  
485            $this->wakeup();  
486        }  
487        if($name == "sleep") {  
488            $this->sleep();  
489        }  
490        if($name == "get") {  
491            $this->get($arguments[0]);  
492        }  
493        if($name == "set") {  
494            $this->set($arguments[0], $arguments[1]);  
495        }  
496        if($name == "destruct") {  
497            $this->destruct();  
498        }  
499        if($name == "clone") {  
500            $this->clone();  
501        }  
502        if($name == "wakeup") {  
503            $this->wakeup();  
504        }  
505        if($name == "sleep") {  
506            $this->sleep();  
507        }  
508        if($name == "get") {  
509            $this->get($arguments[0]);  
510        }  
511        if($name == "set") {  
512            $this->set($arguments[0], $arguments[1]);  
513        }  
514        if($name == "destruct") {  
515            $this->destruct();  
516        }  
517        if($name == "clone") {  
518            $this->clone();  
519        }  
520        if($name == "wakeup") {  
521            $this->wakeup();  
522        }  
523        if($name == "sleep") {  
524            $this->sleep();  
525        }  
526        if($name == "get") {  
527            $this->get($arguments[0]);  
528        }  
529        if($name == "set") {  
530            $this->set($arguments[0], $arguments[1]);  
531        }  
532        if($name == "destruct") {  
533            $this->destruct();  
534        }  
535        if($name == "clone") {  
536            $this->clone();  
537        }  
538        if($name == "wakeup") {  
539            $this->wakeup();  
540        }  
541        if($name == "sleep") {  
542            $this->sleep();  
543        }  
544        if($name == "get") {  
545            $this->get($arguments[0]);  
546        }  
547        if($name == "set") {  
548            $this->set($arguments[0], $arguments[1]);  
549        }  
550        if($name == "destruct") {  
551            $this->destruct();  
552        }  
553        if($name == "clone") {  
554            $this->clone();  
555        }  
556        if($name == "wakeup") {  
557            $this->wakeup();  
558        }  
559        if($name == "sleep") {  
560            $this->sleep();  
561        }  
562        if($name == "get") {  
563            $this->get($arguments[0]);  
564        }  
565        if($name == "set") {  
566            $this->set($arguments[0], $arguments[1]);  
567        }  
568        if($name == "destruct") {  
569            $this->destruct();  
570        }  
571        if($name == "clone") {  
572            $this->clone();  
573        }  
574        if($name == "wakeup") {  
575            $this->wakeup();  
576        }  
577        if($name == "sleep") {  
578            $this->sleep();  
579        }  
580        if($name == "get") {  
581            $this->get($arguments[0]);  
582        }  
583        if($name == "set") {  
584            $this->set($arguments[0], $arguments[1]);  
585        }  
586        if($name == "destruct") {  
587            $this->destruct();  
588        }  
589        if($name == "clone") {  
590            $this->clone();  
591        }  
592        if($name == "wakeup") {  
593            $this->wakeup();  
594        }  
595        if($name == "sleep") {  
596            $this->sleep();  
597        }  
598        if($name == "get") {  
599            $this->get($arguments[0]);  
600        }  
601        if($name == "set") {  
602            $this->set($arguments[0], $arguments[1]);  
603        }  
604        if($name == "destruct") {  
605            $this->destruct();  
606        }  
607        if($name == "clone") {  
608            $this->clone();  
609        }  
610        if($name == "wakeup") {  
611            $this->wakeup();  
612        }  
613        if($name == "sleep") {  
614            $this->sleep();  
615        }  
616        if($name == "get") {  
617            $this->get($arguments[0]);  
618        }  
619        if($name == "set") {  
620            $this->set($arguments[0], $arguments[1]);  
621        }  
622        if($name == "destruct") {  
623            $this->destruct();  
624        }  
625        if($name == "clone") {  
626            $this->clone();  
627        }  
628        if($name == "wakeup") {  
629            $this->wakeup();  
630        }  
631        if($name == "sleep") {  
632            $this->sleep();  
633        }  
634        if($name == "get") {  
635            $this->get($arguments[0]);  
636        }  
637        if($name == "set") {  
638            $this->set($arguments[0], $arguments[1]);  
639        }  
640        if($name == "destruct") {  
641            $this->destruct();  
642        }  
643        if($name == "clone") {  
644            $this->clone();  
645        }  
646        if($name == "wakeup") {  
647            $this->wakeup();  
648        }  
649        if($name == "sleep") {  
650            $this->sleep();  
651        }  
652        if($name == "get") {  
653            $this->get($arguments[0]);  
654        }  
655        if($name == "set") {  
656            $this->set($arguments[0], $arguments[1]);  
657        }  
658        if($name == "destruct") {  
659            $this->destruct();  
660        }  
661        if($name == "clone") {  
662            $this->clone();  
663        }  
664        if($name == "wakeup") {  
665            $this->wakeup();  
666        }  
667        if($name == "sleep") {  
668            $this->sleep();  
669        }  
670        if($name == "get") {  
671            $this->get($arguments[0]);  
672        }  
673        if($name == "set") {  
674            $this->set($arguments[0], $arguments[1]);  
675        }  
676        if($name == "destruct") {  
677            $this->destruct();  
678        }  
679        if($name == "clone") {  
680            $this->clone();  
681        }  
682        if($name == "wakeup") {  
683            $this->wakeup();  
684        }  
685        if($name == "sleep") {  
686            $this->sleep();  
687        }  
688        if($name == "get") {  
689            $this->get($arguments[0]);  
690        }  
691        if($name == "set") {  
692            $this->set($arguments[0], $arguments[1]);  
693        }  
694        if($name == "destruct") {  
695            $this->destruct();  
696        }  
697        if($name == "clone") {  
698            $this->clone();  
699        }  
700        if($name == "wakeup") {  
701            $this->wakeup();  
702        }  
703        if($name == "sleep") {  
704            $this->sleep();  
705        }  
706        if($name == "get") {  
707            $this->get($arguments[0]);  
708        }  
709        if($name == "set") {  
710            $this->set($arguments[0], $arguments[1]);  
711        }  
712        if($name == "destruct") {  
713            $this->destruct();  
714        }  
715        if($name == "clone") {  
716            $this->clone();  
717        }  
718        if($name == "wakeup") {  
719            $this->wakeup();  
720        }  
721        if($name == "sleep") {  
722            $this->sleep();  
723        }  
724        if($name == "get") {  
725            $this->get($arguments[0]);  
726        }  
727        if($name == "set") {  
728            $this->set($arguments[0], $arguments[1]);  
729        }  
730        if($name == "destruct") {  
731            $this->destruct();  
732        }  
733        if($name == "clone") {  
734            $this->clone();  
735        }  
736        if($name == "wakeup") {  
737            $this->wakeup();  
738        }  
739        if($name == "sleep") {  
740            $this->sleep();  
741        }  
742        if($name == "get") {  
743            $this->get($arguments[0]);  
744        }  
745        if($name == "set") {  
746            $this->set($arguments[0], $arguments[1]);  
747        }  
748        if($name == "destruct") {  
749            $this->destruct();  
750        }  
751        if($name == "clone") {  
752            $this->clone();  
753        }  
754        if($name == "wakeup") {  
755            $this->wakeup();  
756        }  
757        if($name == "sleep") {  
758            $this->sleep();  
759        }  
760        if($name == "get") {  
761            $this->get($arguments[0]);  
762        }  
763        if($name == "set") {  
764            $this->set($arguments[0], $arguments[1]);  
765        }  
766        if($name == "destruct") {  
767            $this->destruct();  
768        }  
769        if($name == "clone") {  
770            $this->clone();  
771        }  
772        if($name == "wakeup") {  
773            $this->wakeup();  
774        }  
775        if($name == "sleep") {  
776            $this->sleep();  
777        }  
778        if($name == "get") {  
779            $this->get($arguments[0]);  
780        }  
781        if($name == "set") {  
782            $this->set($arguments[0], $arguments[1]);  
783        }  
784        if($name == "destruct") {  
785            $this->destruct();  
786        }  
787        if($name == "clone") {  
788            $this->clone();  
789        }  
790        if($name == "wakeup") {  
791            $this->wakeup();  
792        }  
793        if($name == "sleep") {  
794            $this->sleep();  
795        }  
796        if($name == "get") {  
797            $this->get($arguments[0]);  
798        }  
799        if($name == "set") {  
800            $this->set($arguments[0], $arguments[1]);  
801        }  
802        if($name == "destruct") {  
803            $this->destruct();  
804        }  
805        if($name == "clone") {  
806            $this->clone();  
807        }  
808        if($name == "wakeup") {  
809            $this->wakeup();  
810        }  
811        if($name == "sleep") {  
812            $this->sleep();  
813        }  
814        if($name == "get") {  
815            $this->get($arguments[0]);  
816        }  
817        if($name == "set") {  
818            $this->set($arguments[0], $arguments[1]);  
819        }  
820        if($name == "destruct") {  
821            $this->destruct();  
822        }  
823        if($name == "clone") {  
824            $this->clone();  
825        }  
826        if($name == "wakeup") {  
827            $this->wakeup();  
828        }  
829        if($name == "sleep") {  
830            $this->sleep();  
831        }  
832        if($name == "get") {  
833            $this->get($arguments[0]);  
834        }  
835        if($name == "set") {  
836            $this->set($arguments[0], $arguments[1]);  
837        }  
838        if($name == "destruct") {  
839            $this->destruct();  
840        }  
841        if($name == "clone") {  
842            $this->clone();  
843        }  
844        if($name == "wakeup") {  
845            $this->wakeup();  
846        }  
847        if($name == "sleep") {  
848            $this->sleep();  
849        }  
850        if($name == "get") {  
851            $this->get($arguments[0]);  
852        }  
853        if($name == "set") {  
854            $this->set($arguments[0], $arguments[1]);  
855        }  
856        if($name == "destruct") {  
857            $this->destruct();  
858        }  
859        if($name == "clone") {  
860            $this->clone();
```

J'ai également créer une classe CSV qui nous sert pour écrire dans les fichiers csv des noeuds:

```
2 < class CSV {  
3  
4     private $nodeName;  
5     private $column;  
6     private $csvPath;  
7     public function __construct($_nodeName, $_column) {  
8         $this->nodeName = $_nodeName;  
9         $this->column = $_column;  
10        $this->csvPath = '/home/pi/databases/'. $this->nodeName . '.csv';  
11    }  
}
```

Chemin d'accès du fichier csv

Dans cette classe CSV, nous avons deux méthodes:

- Une pour écrire dans le fichier
- Une pour créer le fichier si il n'existe pas

```
22     public function saveData($_data) {  
23         if(file_exists($this->csvPath)) {  
24             $file = fopen($this->csvPath, 'a');  
25             fputcsv($file, $_data, ",");  
26             fclose($file);  
27         } else $this->createFile($_data);  
28     }  
29  
30     private function createFile($_data) {  
31         $file = fopen($this->csvPath, 'a');  
32         fputcsv($file, $this->column, ",");  
33         fclose($file);  
34         $this->saveData($_data);  
35     }  
}
```

Si le fichier existe
Ouverture en écriture et à la fin du fichier
Si le fichier existe pas on appelle la méthode createFile()
On entre le nom des données en 1ère ligne
Ex:
timestamp co2 temperature
humidite

En dehors de toutes classes et fonctions, on crée une instance de SQLite3 appelée « bdd », ça permet de se connecter à la base de donnée pour agir dedans plus tard.

```
// Connexion à la base de donnée  
$bdd = new SQLite3('/home/pi/databases/plateformeiot.db');
```

Connexion à la base de donnée SQLite3

Chemin d'accès de la BDD

J'ai également créé la fonction permettant d'écrire dans la base de données. Cette fonction est appelée dans la méthode useData() de la classe CNode.

```
function sendDataToDatabase($_table, $_data, $_column){  
    if(count($_column) != count($_data)) return -1;  
    $request = createSQLRequest($_table, $_data, $_column);  
    global $bdd;  
    $bdd->exec($request);  
}
```

Vérifie que le tableau de donnée fait la bonne taille

Créer la requête SQL

Pour accéder à l'instance \$bdd dans notre méthode

La fonction « createSQLRequest() » est une fonction qui permet de créer la requête SQL à envoyer sur la base de données pour écrire les données reçues.

Maintenant que les classes sont faites, on peut créer nos nœuds. Pour cela, j'ai besoin des informations suivantes:

- Le topic MQTT correspondant
- Le nom des champs (timestamp, humidité, etc...)

Les topics pour les applications Chirpstack se font de la forme suivante:

application/[ID DE L'APPLICATION]/device/[DEVICE EUI]/event/up

Exemple: application/3/device/70b3d57ed0048d3e/event/up

J'ai créer un fichier main.php:

```
$server = '10.121.41.128';
$port = '1883';

$boopyTopic = 'application/3/device/+/event/up';

$co2Topic = 'application/3/device/70b3d57ed0048d3e/event/up';
//           $_server, $_port, $_username, $_password, $_id, $_topic, $_nodeName
$co2Node = new CNode($server, $port,'','','co2Id', $co2Topic, 'co2',
                     array('timestamp', 'co2', 'temperature', 'humidite'));

$boopyNode = new CNode($server, $port, '', '', 'boopyId', $boopyTopic, 'boopy',
                      array('timestamp','luminosite','temperature','humidite',
                            'pressionAtm','longitude','latitude','altitude'));

$waitingTime = 10 * 60; // 10 * 60 pour 10 minutes

while(true) {
    $co2Node->goConnect();
    $boopyNode->goConnect();

    $co2Node->waitMessage();
    $boopyNode->waitMessage();

    // Attendre
    sleep($waitingTime);
}
```

Création des différents noeuds

Temps d'attente entre chaque réception

Boucle sans fin qui réceptionne les données et les traite

Maintenant, j'ai besoin de créer un service qui permettra à ce programme d'être constamment allumé en tâche de fond sur le raspberry.

Pour cela, je créer un fichier plateformeIoT.service avec le contenu suivant:

```
GNU nano 5.4                                     plateformeIoT.service
[Unit]
Description=PHP script that receives MQTT Chirpstack data and sends it to BDD and CSV file

[Service]
Type=simple
ExecStart=php /var/www/html/service/main.php

[Install]
WantedBy=multi-user.target
```

Commande que le service doit exécuter

Description du service

Pour activer le service, on fait les commandes suivantes:

Pour copier le fichier service dans le répertoires des services:

sudo cp plateformeoT.service /etc/systemd/system/

Pour l'activer:

sudo systemctl enable palteformeoT.service

```
pi@ttn-gateway:~ $ sudo cp plateformeIoT.service /etc/systemd/system/
pi@ttn-gateway:~ $ sudo systemctl enable plateformeIoT.service
Created symlink /etc/systemd/system/multi-user.target.wants/plateformeIoT.service → /etc/systemd/system/plateformeIoT.service
```

Pour interagir avec le service :

sudo systemctl [start/stop/status/restart] plateformeoT.service

```
pi@ttn-gateway:~ $ sudo systemctl start plateformeIoT.service
pi@ttn-gateway:~ $ sudo systemctl status plateformeIoT.service
● plateformeIoT.service - PHP script that receives MQTT Chirpstack data and sends it to BDD and CSV file
   Loaded: loaded (/etc/systemd/system/plateformeIoT.service; enabled; vendor preset: enabled)
   Active: active (running) since Sun 2022-05-22 12:45:09 CEST; 28s ago
     Main PID: 5160 (php)
        Tasks: 1 (limit: 4654)
       Memory: 4.6M
      CGroup: /system.slice/plateformeIoT.service
              └─5160 /usr/bin/php /var/www/html/main/main.php
```

Le service php fonctionne correctement.

Test du système

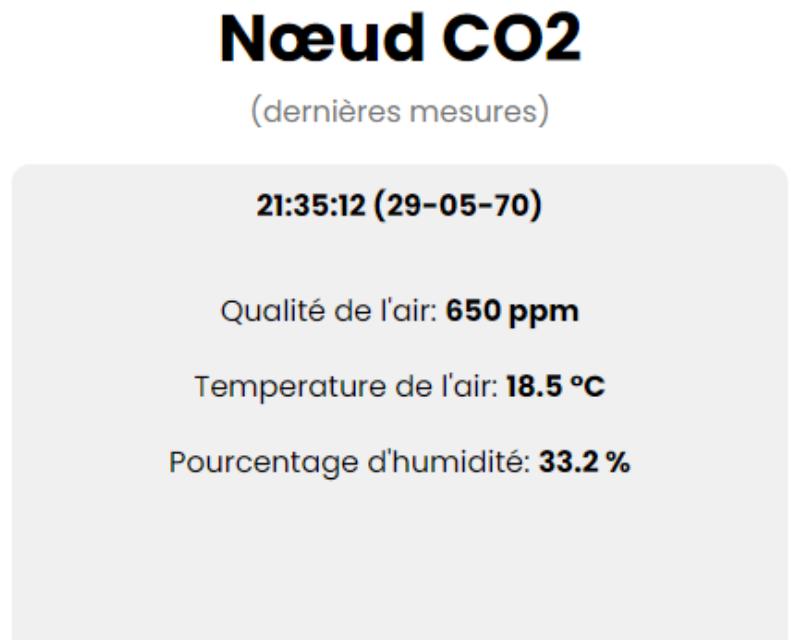
Maintenant que le service fonctionne en continue, et que le site web est en place, on peut tester la réception de trame:

Exemple d'une trame qu'on reçoit par MQTT:

```
pi@raspberrypi:~/databases $ mosquitto_sub -h 10.121.41.128 -t "application/3/device/70b3d57ed0048d3e/event/up"
{"applicationID": "3", "applicationName": "app1", "deviceName": "abp", "deviceProfileName": "ABP", "deviceProfileID": "5c20c831-04c0-4a6f-4d1-c60d4672d0f7", "devEUI": "70b3d57ed0048d3e", "rxInfo": [{"gatewayID": "b827ebe6f7acffff", "uplinkID": "058c1880-e3b8-4cac-85dc-78fa8d3ea2", "name": "gw2", "time": "2022-05-20T22:15:08.0810792", "rssI": -102, "LoRaSNR": -5.2, "location": {"latitude": 0, "longitude": 0, "altitude": 0}}, {"txInfo": {"frequency": 867300000, "dr": 0}, "adr": false, "fCnt": 2, "fPort": 4, "data": "MTI4NjEzMjI7NjUwOzE4LjU7MzMzMg==", "obj": {"data_decoded": {"trame": "12861312;650;18.5;33.2"}}}}
```

Avec la méthode **trame()** de la classe CNode, on ressort de cette trame la chaîne de caractère suivante: “**12861312;650;18.5;33.2**”, ce qui correspond aux données envoyées par le nœud.

Résultat de l'affichage sur le site:



La base de donnée enregistre bien les données :

```
sqlite> select * from co2;
1651950479|486|25.3|28.5
```

Les données sont bien écrites dans le fichier .csv :

```
GNU nano 5.4
timestamp;co2;temperature;humidite
1651950479;486;25.3;28.5
```

Problèmes rencontrés

Message retenue en MQTT

Notre site web doit afficher les données en temps réel, pour ça, j'ai mis une actualisation rapide des données du site.

Pour actualiser, on charge toutes les 10 secondes les programmes PHP qui ont des subscriber MQTT.

Le problème était que Chirpstack, par défaut, publie ses messages MQTT sans avoir activé la retenue.

La retenue est un principe du protocole MQTT permettant à un subscriber d'accéder au dernier message envoyé sur le topic en s'y abonnant. Le dernier message est en quelque sorte « gardé en mémoire ».

Sans cette option activée, notre page web devrait recevoir les données des nœuds toutes les 10 secondes afin d'afficher constamment des données sur la page.

En activant cette option sur Chirpstack, ça nous permettra de toujours avoir les dernières données affichées sur le site, même si le dernier message du publisher date d'il y a 9 minutes.

Pour activer cette option, on se rend dans le dossier de configuration **/chirpstack-application-server** puis on ouvre le fichier de configuration **chirpstack-application-server.toml**

On ajoute à la fin du fichier les lignes suivantes:

```
[application_server.integration.mqtt]
retain_events=true
```

Permission d'écriture

Sur le Serveur Application Web, lorsque j'ai créer mon programme de test pour écrire dans un fichier CSV, j'ai observé un problème de permission qui empêchait le programme d'écrire:

```
pi@ttn-gateway:/var/www/html $ php csv.php
PHP Warning: fopen(/home/pi/database/data.csv): failed to open stream: Permission denied
in /var/www/html/csv.php on line 4
PHP Warning: fputcsv() expects parameter 1 to be resource, bool given in /var/www/html/cs
v.php on line 5
PHP Warning: fclose() expects parameter 1 to be resource, bool given in /var/www/html/cs
.php on line 6
```

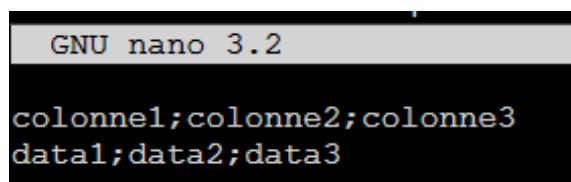
Pour résoudre ce problème, j'ai décidé que le programme créer lui-même le fichier dans lequel il va écrire. En créant le fichier, il aura alors la permission d'écrire à l'intérieur vu qu'il sera l'auteur du fichier.

J'ai donc supprimé l'ancien fichier et créé une fonction « `createCsvFile()` » dans mon programme PHP. Cette fonction est appelée dans la fonction « `saveData()` » si le fichier de destination n'existe pas.

Cette fonction ouvre un fichier csv inexistant (et donc le créer) et y met en 1ère ligne les colonnes correspondant aux données voulues.

```
function createCsvFile($_data) {
    global $csvPath;
    $file = fopen($csvPath, 'a');
    // 1ère Ligne du fichier csv
    fputcsv($file, array('colonne1','colonne2','colonne3'), ";");
    fclose($file);
    saveData($_data);
}
```

Avec cette solution, nos données sont bien écrites dans le fichier:



```
GNU nano 3.2
colonne1;colonne2;colonne3
data1;data2;data3
```