

# Prise en main des cartes Pycom

Clément Even

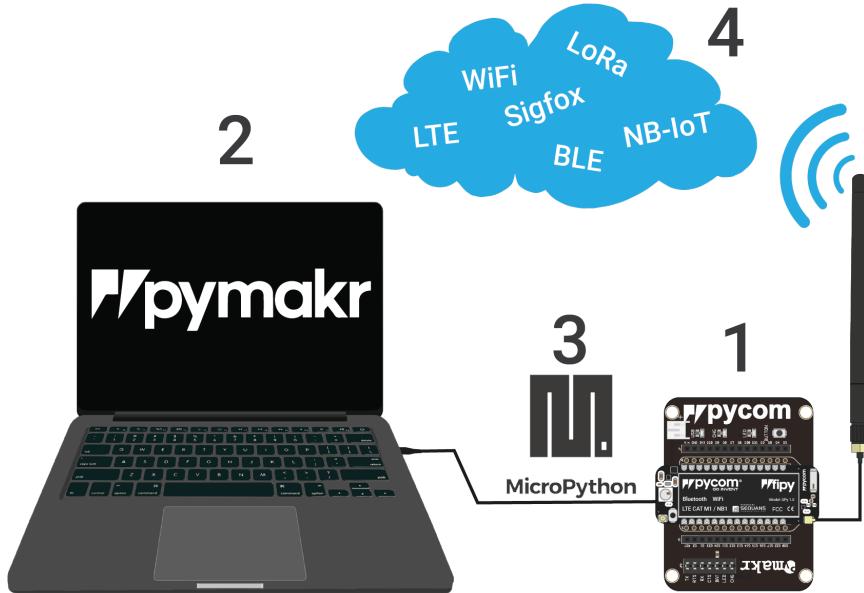
16/04/2020

# Table des matières

<b>Installation des logiciels et du plugin</b>	<b>4</b>
<b>    Installation de Vscode sur Windows</b>	<b>5</b>
Téléchargement . . . . .	5
Installation . . . . .	5
<b>    Installation de NodeJs sur Windows</b>	<b>6</b>
Téléchargement . . . . .	6
Installation . . . . .	6
<b>    Installation de Vscode sur linux</b>	<b>7</b>
Téléchargement Installation . . . . .	7
<b>    Installation de NodeJs sur linux</b>	<b>7</b>
Téléchargement Installation . . . . .	7
<b>    Pymakr sur Vscode</b>	<b>8</b>
<b>    Installation de Atom sur Windows</b>	<b>9</b>
Téléchargement . . . . .	9
Installation . . . . .	9
<b>    Installation de Atom sur Linux</b>	<b>9</b>
Téléchargement Installation . . . . .	9
<b>    Pymakr sur Atom</b>	<b>10</b>
<b>Mise à jour du firmware des cartes d'extension</b>	<b>11</b>
<b>    Mise à jour sur Windows</b>	<b>12</b>
<b>    Mise à jour sur Linux</b>	<b>14</b>
<b>Connexion de la carte à l'ordinateur</b>	<b>15</b>
<b>    Connexion de la carte sur Vscode</b>	<b>16</b>
Connexion via USB série . . . . .	16
Connexion via Telnet . . . . .	18
<b>    Connexion de la carte sur Atom</b>	<b>19</b>
Connexion via USB série . . . . .	19
Connexion via Telnet . . . . .	20
<b>Création d'un compte sur TTN (The Things Network)</b>	<b>21</b>

<b>Montage global avec les cartes Pycom</b>	<b>22</b>
<b>Mettre en place une nano-passerelle Pycom</b>	<b>23</b>
<b>Montage</b>	<b>23</b>
<b>Mise en œuvre de la nano-passerelle</b>	<b>25</b>
<b>Upload du project sur la carte</b>	<b>27</b>
<b>Enregistrement de la passerelle</b>	<b>28</b>
<b>Noeud LoRa pour tester la nano-passerelle LoRa</b>	<b>31</b>
<b>Montage</b>	<b>31</b>
<b>Mise en œuvre du nœud LoRa</b>	<b>33</b>
<b>Configuration de la carte</b>	<b>35</b>
OTAA . . . . .	35
ABP . . . . .	37
<b>Mettre en place une communication TCP entre la nano-passerelle et le serveur OVH</b>	<b>40</b>
<b>Mise en œuvre</b>	<b>41</b>
<b>Utiliser les capteurs de la carte Pysense</b>	<b>43</b>
<b>Montage</b>	<b>43</b>
<b>Mise en œuvre des capteurs</b>	<b>44</b>
Installation des bibliothèques . . . . .	44
Explication des fonctions disponibles . . . . .	45
Envoyer les données des capteurs en LoRa . . . . .	47

# Installation des logiciels et du plugin



Le codage des projets pour les cartes **Pycom** peut se faire à l'aide du plugin **Pymakr** qui est disponible sur **Vscode(+NodeJS)** ou sur **Atom**. Une fois ces logiciels installés vous pourrez démarrer vos projets IoT en **MicroPython**.

## **Vscode :**

Visual Studio Code est un éditeur de code extensible développé par Microsoft pour Windows, Linux et macOS.

## **NodeJS :**

Node.js® est un runtime JavaScript basé sur le moteur JavaScript V8 de Chrome .

## **Atom :**

Atom est un éditeur de texte libre pour macOS, GNU/Linux et Windows développé par GitHub.

## **Pymakr :**

Pymakr a été développé par Pycom pour faire des projets aussi simples que possibles.

## **MicroPython :**

MicroPython est une implémentation légère et efficace du langage de programmation Python 3.

# Installation de Vscode sur Windows

## Téléchargement :

1. Allez sur le lien : <https://code.visualstudio.com/>



2. Cliquez sur le bouton à gauche :
3. Le fichier **VScodeUserSetup-x64-X.X.X.exe** va se télécharger.
4. Exécutez le fichier quand le téléchargement sera fini.

## Installation :

1. Cochez "**je comprends et j'accepte les termes du contrat de licence**" puis appuyez sur le bouton "**Suivant >**".
2. Choisissez l'emplacement des fichiers d'installation de Visual Studio Code puis appuyez sur le bouton "**Suivant >**".
3. Choisissez si vous voulez créer des raccourcis programme dans le dossier du menu Démarrer puis appuyez sur le bouton "**Suivant >**".
4. Si vous voulez rajouter des tâches supplémentaires dans l'installation, cochez les cases qui vous intéressent puis appuyez sur le bouton "**Suivant >**".
5. Si les informations sont correctes appuyez sur le bouton "**installer**".

**Installation est terminée.**

# Installation de NodeJs sur Windows

## Téléchargement :

1. Allez sur le lien : <https://nodejs.org/en/>
2. Cliquez sur le bouton de téléchargement en vert avec "LTS" :



3. Le fichier **node-vXX.XX.X-x64.msi** va se télécharger.
4. Exécutez le fichier une fois qu'il est téléchargé.

## Installation :

1. Appuyez sur "Next" puis cochez la case "**I accept the terms in the License Agreement**" puis appuyez sur "Next".
2. Choisissez l'emplacement des fichiers d'installation de NodeJs puis appuyez sur le bouton "Next", ensuite appuyez encore sur "Next" et ensuite appuyez encore sur "Next".
3. Appuyez sur le bouton "Install".

**Installation est terminée.**

# Installation de Vscode sur linux

## Téléchargement & Installation :

1. Ouvrez votre l'invite de commande (prompt) et rentrez la ligne de commande :

```
sudo snap install code - --classic
```

2. Quand le téléchargement est fini rentrez la ligne de commande pour lancer Vscode :

```
code
```

3. Installation est terminée pour voir la version faites la commande :

```
code -v
```

# Installation de NodeJs sur linux

## Téléchargement & Installation :

1. Ouvrez l'invite de commande (prompt) et rentrez la commande pour mettre à jour les paquets disponibles :

```
sudo apt-get update
```

2. Installer NodeJs avec la commande :

```
sudo apt install -y nodejs
```

3. Installation est terminée pour voir la version faites la commande :

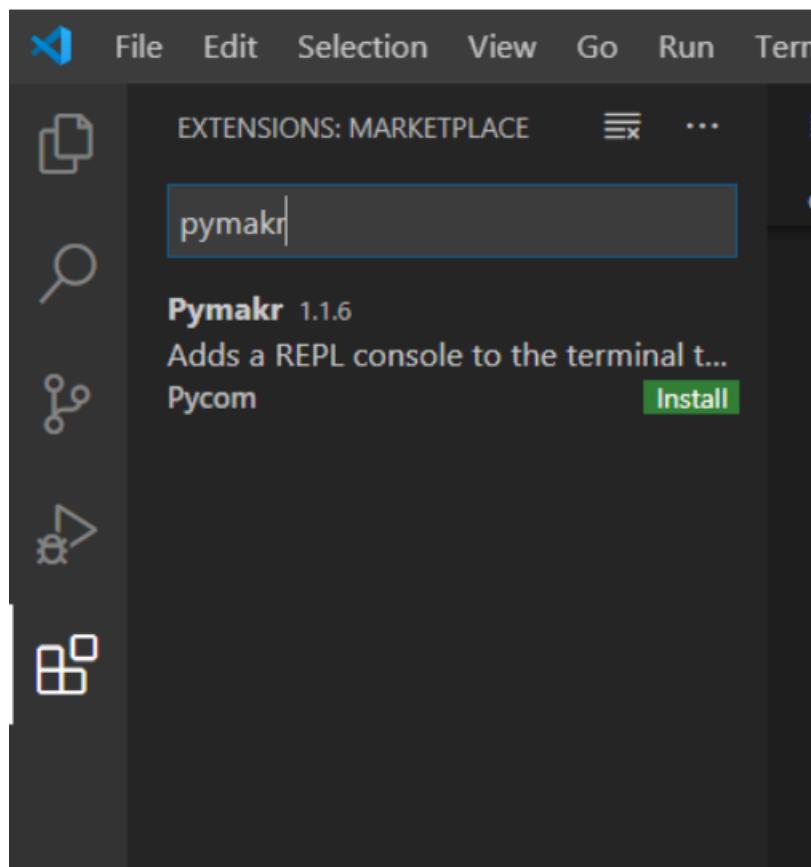
```
nodejs -v
```

# Pymakr sur Vscode

1. Lancez Vscode.
2. Appuyez sur le bouton extension à droite :



3. Dans la barre de recherche, tapez "pymakr" puis appuyez sur le bouton vert "Install" (voir l'image ci-dessous). Pymakr va s'installer sur Vscode.

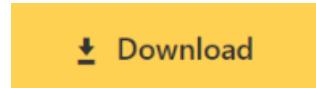


Vous avez fini l'installation complète des logiciels pour Vscode.

# Installation de Atom sur Windows

## Téléchargement :

1. Allez sur le lien : <https://atom.io/>
2. Cliquez sur le bouton à droite :



3. Le fichier **AtomSetup-x64.exe** va se télécharger.
4. Exécutez le fichier une fois qu'il est téléchargé.

## Installation :

1. Rien à faire durant l'installation.

# Installation de Atom sur Linux

## Téléchargement & Installation :

1. Ouvrez votre l'invite de commande (prompt) et rentrez la commande pour mettre à jour les paquets disponibles :

```
sudo apt-get update
```

2. Installez le paquet Atom avec la ligne de commande suivante :

```
sudo apt install atom
```

3. Quand le téléchargement est fini rentrez la ligne de commande pour lancer Atom :

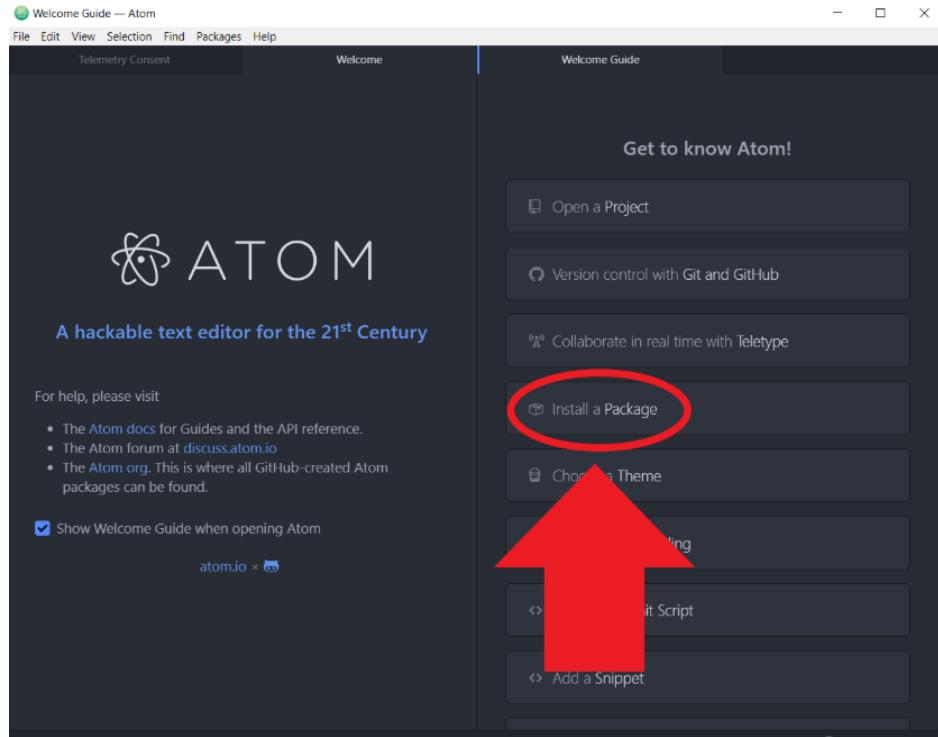
```
atom
```

4. Installation est terminée pour voir la version faites la commande :

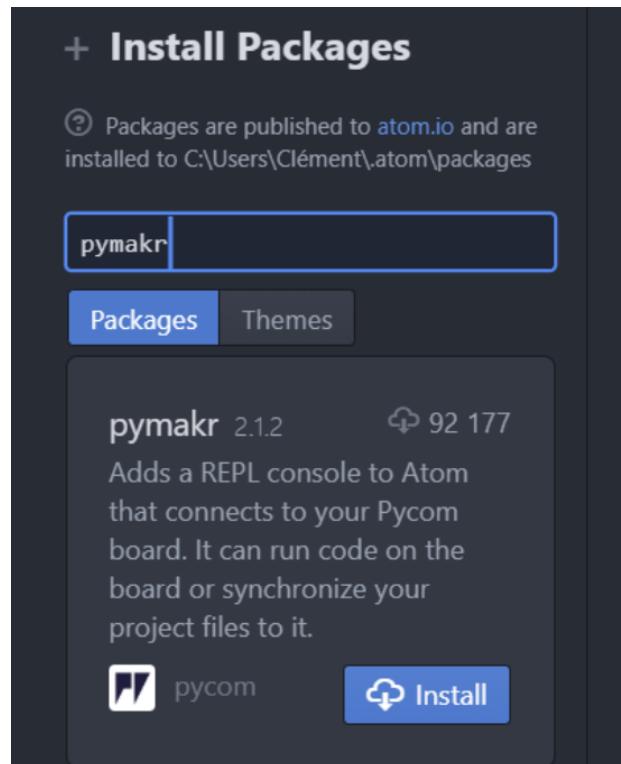
```
atom -v
```

# Pymakr sur Atom

1. Appuyez sur le bouton "Install a Package" puis sur "Open installer".



2. Recherchez dans le moteur de recherche "pymakr" puis appuyer sur le bouton en bleu "Install". Pymark va s'installer sur Atom.



**Vous avez fini l'installation complète des logiciels avec Atom.**

# Mise à jour du firmware des cartes d'extension

Avant d'utiliser les cartes extensions, il faut mettre le firmware de la carte à jour via le port USB. Il faudra utiliser outil "**DFU-util**" pour faire la mise à jour.

## DFU :

DFU est destiné à télécharger et à charger des micrologiciels vers des appareils connectés via USB. Cela va des petits appareils comme les cartes microcontrôleurs aux téléphones portables. En utilisant dfu-util, vous pouvez télécharger le firmware sur votre appareil compatible DFU ou télécharger le firmware depuis celui-ci.

Tout d'abord allez sur lien pour télécharger la dernière version du firmware de la carte :

- Pytrack DFU
- Pysense DFU
- Expansion Board DFU v3.0
- Expansion Board DFU v3.1



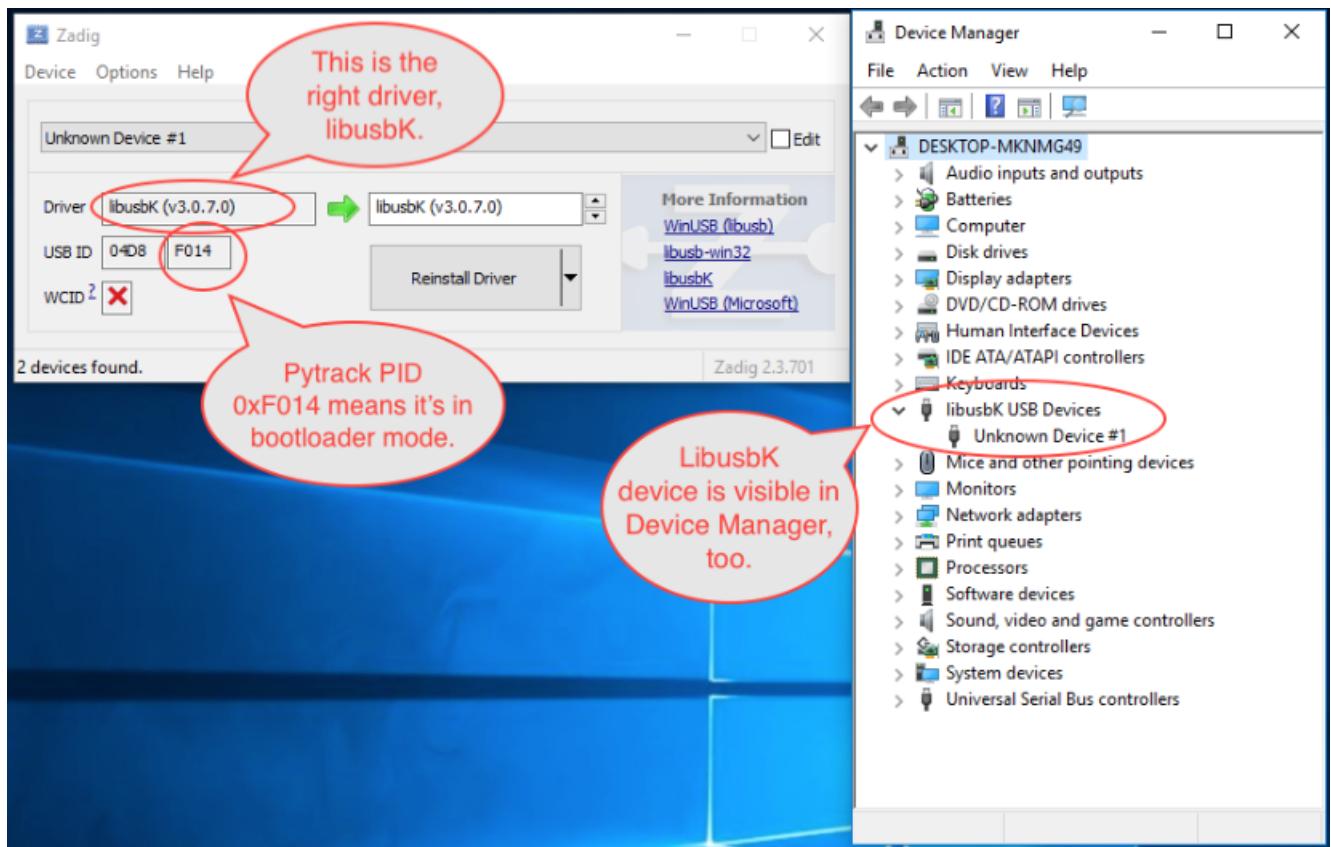
Vous pouvez voir la version de la carte extension à l'arrière de la carte.

En mode d'application normal, la carte Pysense / Pytrack / Pyscan / Expansion Board v3 nécessite un pilote **CDC USB série**, en mode DFU, bootloader, **le pilote DFU est requis**. Ci-dessous, l'ID de produit USB est représenté pour chaque cas.

Carte d'extension	Mode DFU (mode de mise à jour)	Application firmware (mode normal)
Pytrack	0xF014	0xF013
Pysense	0xF011	0xF012
Pyscan	0xEF37	0xEF38
Carte d'extension v3	0xEF99	0xEF98

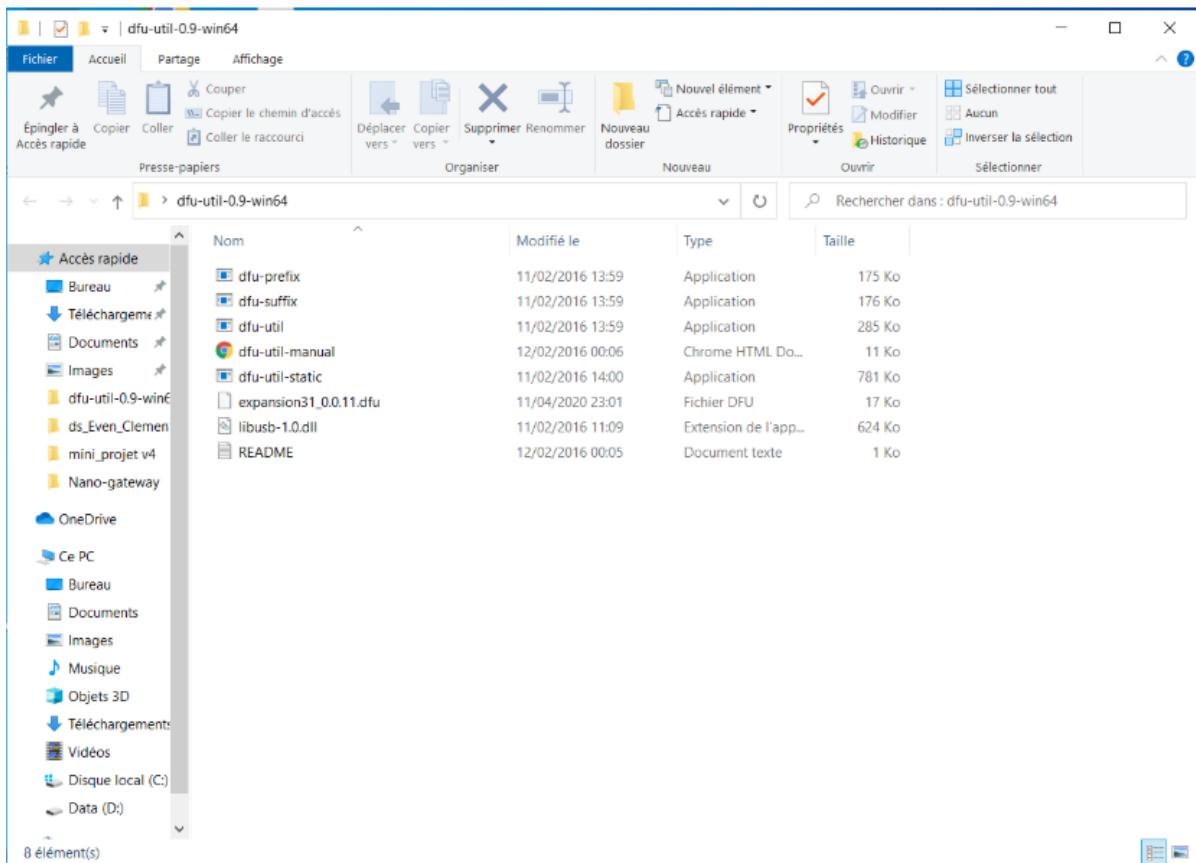
# Mise à jour sur Windows

1. Téléchargez outils **DFU** via le lien suivant : [DFU-util v0.9](#)
2. Téléchargez **Zadig** via le lien suivant : [Zadig](#)  
(**Zadig** est une application Windows qui installe des pilotes USB génériques.)
3. Lancez Zadig et préparez vous à installer **libusbK** sur la carte extension.
4. Appuyez sur le bouton de la carte (S1 pour la carte extension) et branchez la carte en USB sur votre PC.
5. Maintenez le bouton pendant 1 sec.
6. Relâchez le bouton, la carte est connecté en mode DFU **pendant 7 seconde**.
7. Cliquez sur "**install driver**" le plus vite possible (Recommencez à partir de l'étape 3 si vous n'avez pas réussi à installer le driver).



**Remarque :** Ce n'est pas facile à réaliser, la première fois vous pourrez recommencer 5 fois avant de pourvoir installer **linusbK**.

- Décompresser les fichiers de **dfu-util** et mettez le fichier du firmware dans le dossier décompresser



- Ouvrez la console de windows **dans le dossier dfu-util** et préparez vous à rentrer la commande suivante :

**dfu-util-static.exe -D nom-du-fichier.dfu**

Remplacez **nom-du-fichier.dfu** par le nom du fichier du firmware téléchargé au début.

- Débranchez la carte.
- Appuyez sur le bouton de la carte (S1 pour la carte extension) et branchez la carte en USB.
- Maintenez le bouton pendant 1 sec.
- Relâchez le bouton, la carte est connecté en mode DFU **pendant 7 seconde**.
- Lancez la commande préparée dans la console (si vous n'avez pas réussi recommencez à partir de l'étape 9).

```

dfu-util 0.9

Copyright 2005-2009 Weston Schmidt, Harald Welte and OpenMoko Inc.
Copyright 2010-2016 Tormod Volden and Stefan Schmidt
This program is Free Software and has ABSOLUTELY NO WARRANTY
Please report bugs to http://sourceforge.net/p/dfu-util/tickets/

Match vendor ID from file: 04d8
Match product ID from file: ef99
Opening DFU capable USB device...
ID 04d8:ef99
Run-time device DFU version 0100
Claiming USB DFU Runtime Interface...
Determining device status: state = dfuIDLE, status = 0
WARNING: Runtime device already in DFU state ?!?
Claiming USB DFU Interface...
Setting Alternate Setting #0 ...
Determining device status: state = dfuERROR, status = 14
dfuERROR, clearing status
Determining device status: state = dfuIDLE, status = 0
dfuIDLE, continuing
DFU mode device DFU version 0100
Device returned transfer size 64
Copying data from PC to DFU device
Download      [=====] 100%          16384 bytes
Download done.
state(2) = dfuIDLE, status(0) = No error condition is present
Done!

C:\Users\Clément\Desktop\dfu-util-0.9-win64>

```

La mise à jour est terminé.

## Mise à jour sur Linux

1. Installez les outils DFU-util en ouvrant un terminal et via la commande suivante :

**`sudo apt-get install dfu-util`**

2. Mettez vous dans le dossier où vous avez téléchargé le firmware de la carte et préparez vous à rentrez la commande suivante :

**`sudo dfu-util -D nom-du-fichier.dfu`**

3. Remplacez **nom-du-fichier.dfu** par le nom du fichier du firmware téléchargé au début.
4. Appuyez sur le bouton de la carte (S1 pour la carte extension) et branchez la carte en USB.
5. Maintenez le bouton pendant 1 sec.
6. Relâchez le bouton, la carte est connecté en mode DFU **pendant 7 seconde**.
7. Lancez la commande préparée dans la console (si vous n'avez pas réussi recommencez à partir de l'étape 2).

# Connexion de la carte à l'ordinateur

Après avoir installé le plugin **Pymakr**, vous devez prendre quelques secondes pour le configurer pour la première utilisation sur votre PC.

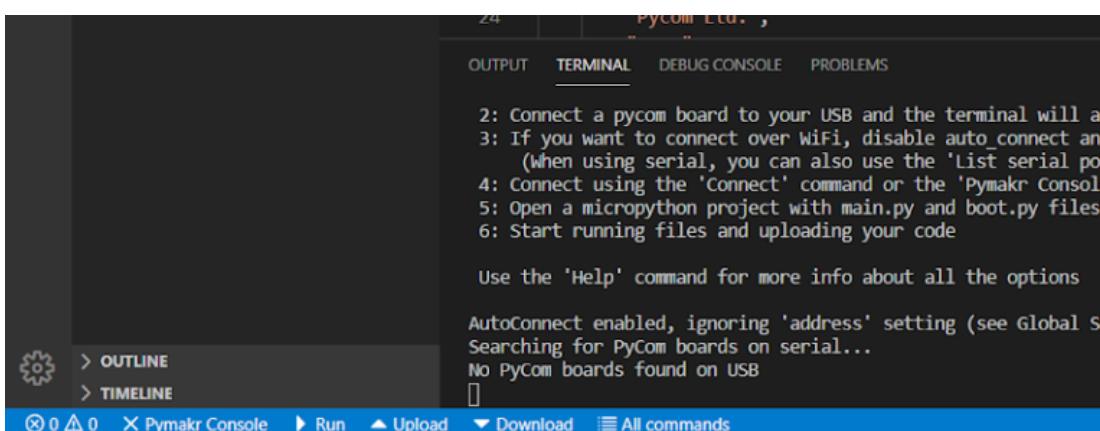
**2 connexions sont possibles :**

- Connexion via USB série
- Connexion via Telnet

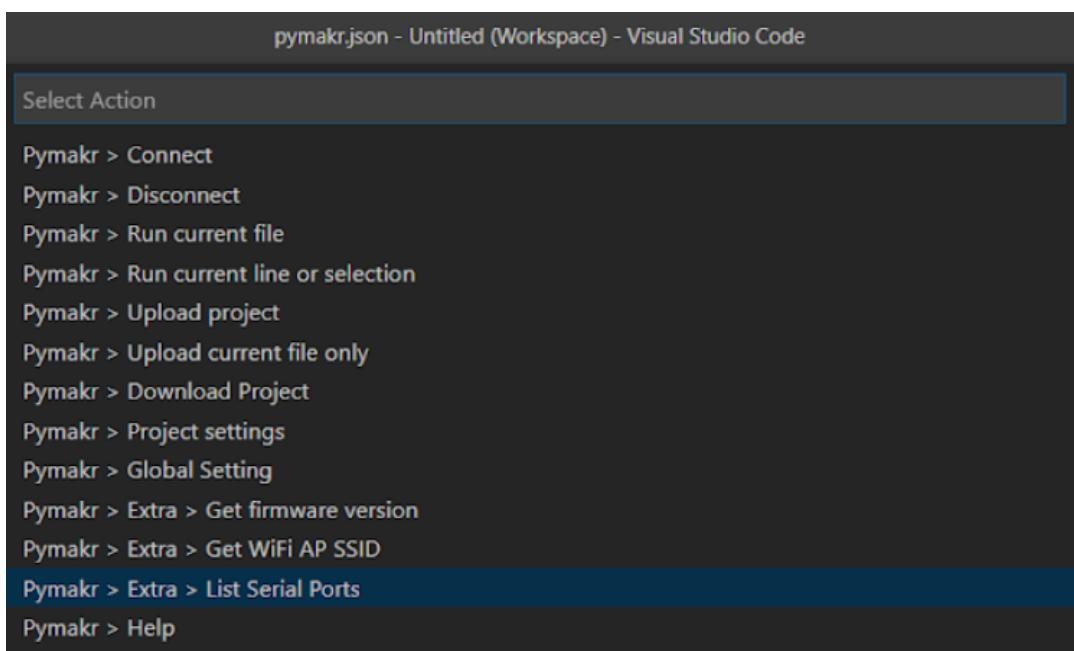
# Connexion de la carte sur Vscode

## Connexion via USB série :

1. Connectez votre appareil Pycom à votre ordinateur via USB. Si vous utilisez une Carte d'extension et venez de terminer une mise à niveau du micrologiciel, assurez-vous de supprimer le fil entre GND et G23 et réinitialisez votre appareil en appuyant sur le bouton reset. **Remarque** : Vous n'avez pas besoin du fil pour la **carte d'extension 3.0**.
2. Ouvrez Visual Studio Code et assurez-vous que le plugin Pymakr est correctement installé. (Vous devriez avoir une barre en bas dans votre Vscode)



3. Cliquez sur "**All commands**" situé en bas de la fenêtre du logiciel.
4. Dans la liste qui apparaît, cliquez sur "**Pymakr > Extra > List Serial Ports**".



- Ceci listera les ports série disponibles. Si Pymakr est capable de détecter automatiquement lesquelles utiliser, ils seront copiés dans votre presse-papiers. Sinon, veuillez copier manuellement les bons ports série.

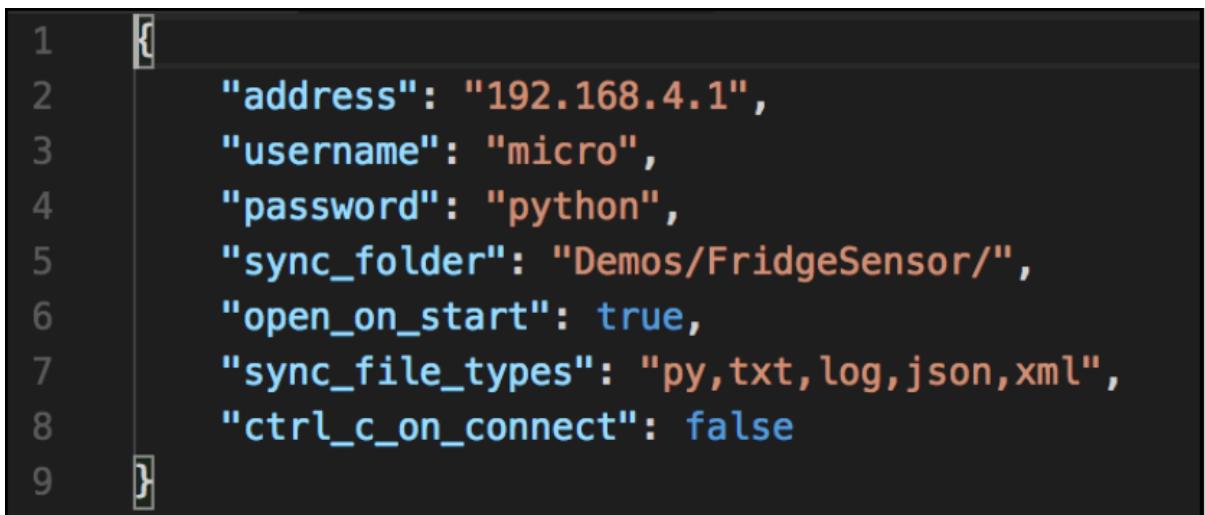


The screenshot shows a terminal window titled "Pycom Consol". The output of the command "pyserial" is displayed:

```
Connecting on 192.168.4.1...
Connection error: Login timed out. Press any key to try again.

Found 3 serialports
/dev/cu.usbserial-DQ0054ES (copied to clipboard)
/dev/cu.SparkleMotion-SPPDev
/dev/cu.SparkleMotion-SPPDev-1
```

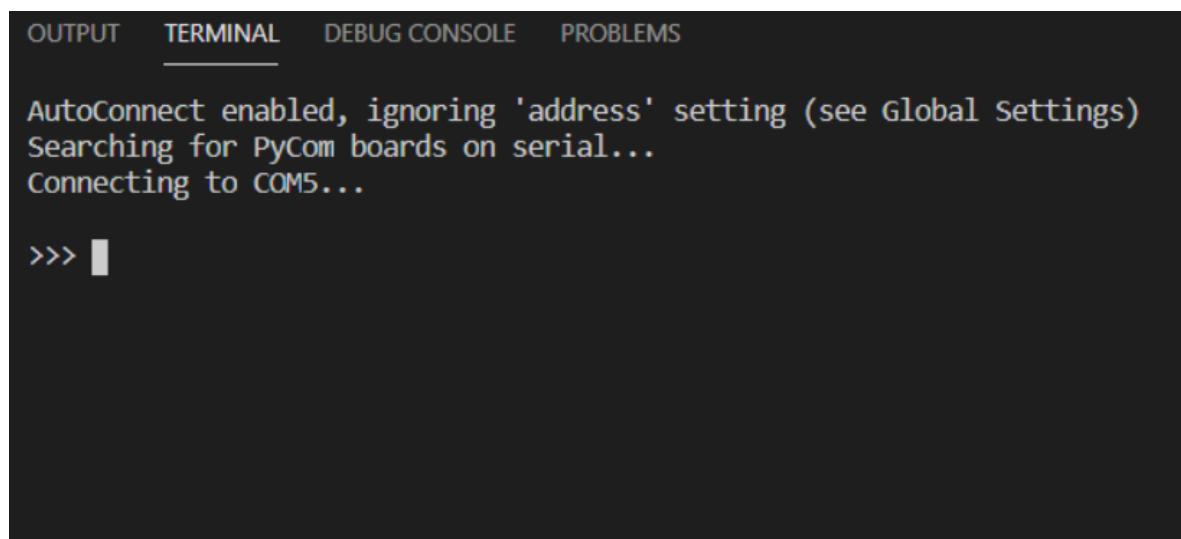
Cliquez de nouveau sur "**All commands**", puis cliquez sur "**Pymakr > Global Settings**". Cela ouvrira un fichier JSON. Ecrivez :"COMX"(voir dans le **gestionnaire de périphérique** pour savoir le numéros du com) dans le champ "**address**" et enregistrez le fichier.



The screenshot shows a JSON configuration file with the following content:

```
1 {
2   "address": "192.168.4.1",
3   "username": "micro",
4   "password": "python",
5   "sync_folder": "Demos/FridgeSensor/",
6   "open_on_start": true,
7   "sync_file_types": "py,txt,log,json,xml",
8   "ctrl_c_on_connect": false
9 }
```

- Enfin, fermez le fichier JSON, cliquez sur "**All commands**", puis sur "**Pymakr > Connect**" pour connecter votre appareil. La console Pymakr doit afficher trois flèches ">>>", indiquant que vous êtes connecté.



The screenshot shows the Pycom Console terminal with the "TERMINAL" tab selected. The output shows the connection process:

```
OUTPUT TERMINAL DEBUG CONSOLE PROBLEMS

AutoConnect enabled, ignoring 'address' setting (see Global Settings)
Searching for PyCom boards on serial...
Connecting to COM5...

>>> 
```

7. Ces paramètres peuvent également être appliqués par projet en cliquant sur "**All commands**" puis "**Pymakr > Project Settings**". Cela ouvrira un fichier JSON que vous pouvez modifier pour entrer vos paramètres souhaités pour le projet actuellement ouvert.
8. Ce processus est plus simple avec une carte d'extension Pycom ou un Pytrack / Pysense / Pyscan car les adresses sont automatiquement sélectionnées. Pour les produits externes tels que les câbles série USB FTDI, il est possible que l'adresse série doive être copiée manuellement. En outre, il peut également être nécessaire d'appuyer sur le bouton de réinitialisation de l'appareil avant qu'un message de connexion ne s'affiche.

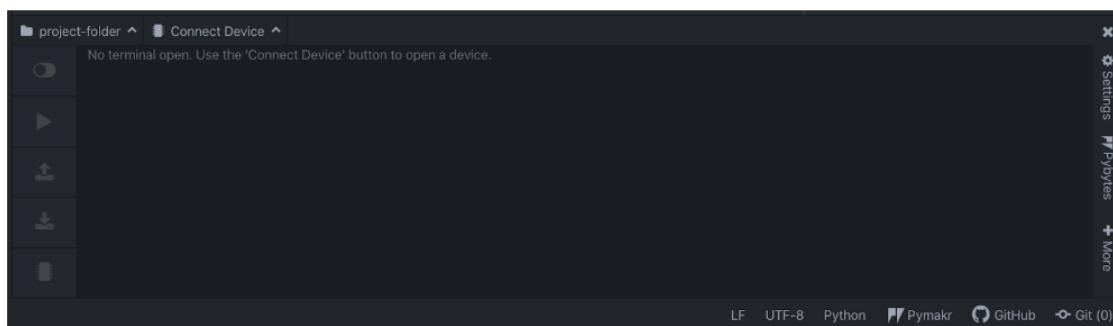
## Connexion via Telnet :

1. Assurez-vous que le périphérique Pycom est allumé.
2. Connectez l'ordinateur hôte au point d'accès WiFi nommé d'après votre carte (le SSID sera comme suit par exemple lopy-wlan-xxxx, wipy-wlan-xxx, ... ). Le mot de passe est "**www.pycom.io**".
3. Suivez les étapes ci-dessus "**Connexion via USB série**" mais entrez "**192.168.4.1**" comme adresse.
4. Le nom d'utilisateur et le mot de passe par défaut sont "**micro**" et "**python**", respectivement.
5. Enfin, fermez le fichier JSON, cliquez sur "**All commands**", puis sur "**Pymakr > Connect**", Pymakr se connectera désormais via telnet.

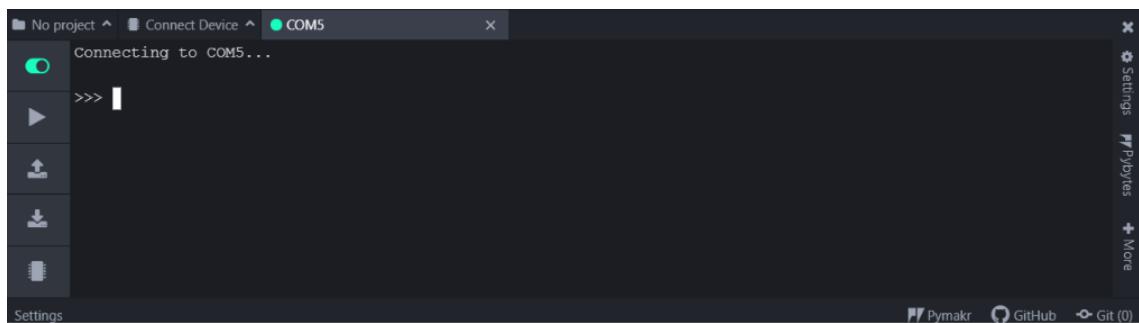
# Connexion de la carte sur Atom

## Connexion via USB série :

1. Connectez votre appareil Pycom à votre ordinateur via USB. Si vous utilisez une Carte d'extension et venez de terminer une mise à niveau du micrologiciel, assurez-vous de supprimer le fil entre GND et G23 et réinitialisez votre appareil en appuyant sur le bouton reset. **Remarque** : Vous n'avez pas besoin du fil pour la **carte d'extension 3.0**.
2. Ouvrez Atom et assurez-vous que le plug-in Pymakr est correctement installé. (Vous devriez avoir une console pymark en bas sur Atom)



3. Pymakr utilise la connexion automatique activée par défaut. Si votre appareil ne s'est pas connecté immédiatement, cliquez sur "**Connect device**" puis sur votre appareil.
4. Maintenant, il devrait montrer trois flèches ">>>", indiquant que vous êtes connecté !

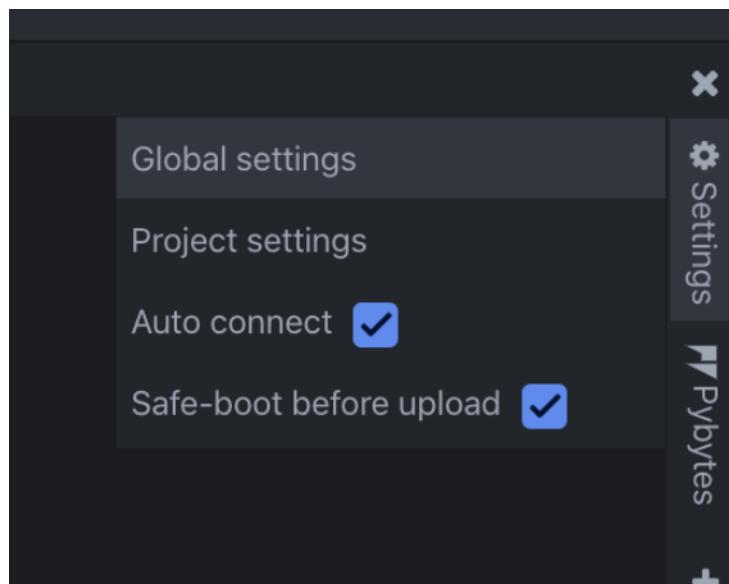


5. Ces paramètres peuvent également être appliqués par projet en cliquant sur "**Settings**" puis "**Project Settings**". Cela ouvrira un fichier JSON que vous pouvez modifier pour entrer vos paramètres souhaités pour le projet actuellement ouvert.

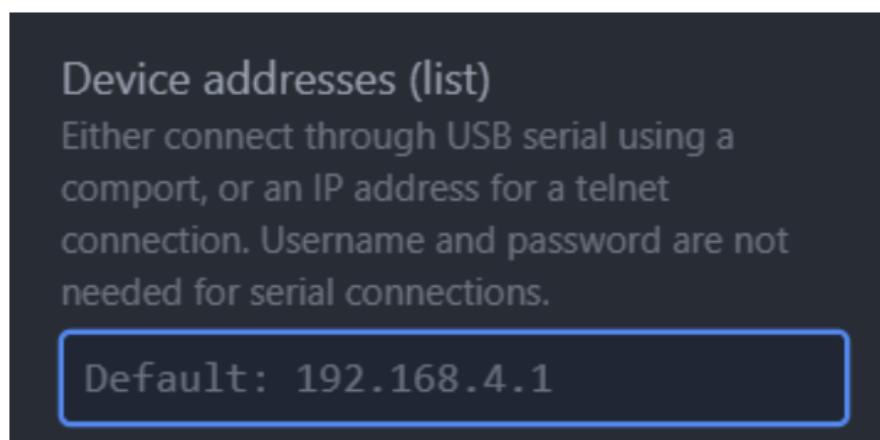
- Ce processus est plus simple avec une carte d'extension Pycom ou un Pytrack / Pysense / Pyscan car les adresses sont automatiquement sélectionnées. Pour les produits externes tels que les câbles série USB FTDI, il est possible que l'adresse série doive être copiée manuellement. En outre, il peut également être nécessaire d'appuyer sur le bouton de réinitialisation de l'appareil avant qu'un message de connexion ne s'affiche.

## Connexion via Telnet :

- Assurez-vous que le périphérique Pycom est allumé.
- Connectez l'ordinateur hôte au point d'accès WiFi nommé d'après votre carte (le SSID sera comme suit par exemple lopy-wlan-xxxx, wipy-wlan-xxx, ... ). Le mot de passe est "www.pycom.io".
- Allez à "Settings" puis sur "Global Settings".



- Dans "Devices Addresses (List)", entrez "192.168.4.1" comme adresse.



- Le nom d'utilisateur et le mot de passe par défaut sont "micro" et "python", respectivement.
- Cliquez "192.168.4.1" dans le volet "Liste des appareils", Pymakr se connectera désormais via telnet.

# Création d'un compte sur TTN (The Things Network)

Pour utiliser The Things Network, vous devez vous rendre sur leur site Web et créer un compte. Saisissez un nom d'utilisateur et une adresse e-mail auprès de leur plateforme sur : <https://account.thethingsnetwork.org/register>



## CREATE AN ACCOUNT

Create an account for The Things Network and start exploring the world of Internet of Things with us.

**USERNAME**  
This will be your username — pick a good one because you will not be able to change it.  


**EMAIL ADDRESS**  
You will occasionally receive account related emails. This email address is not public.  


**PASSWORD**  
Use at least 6 characters.  

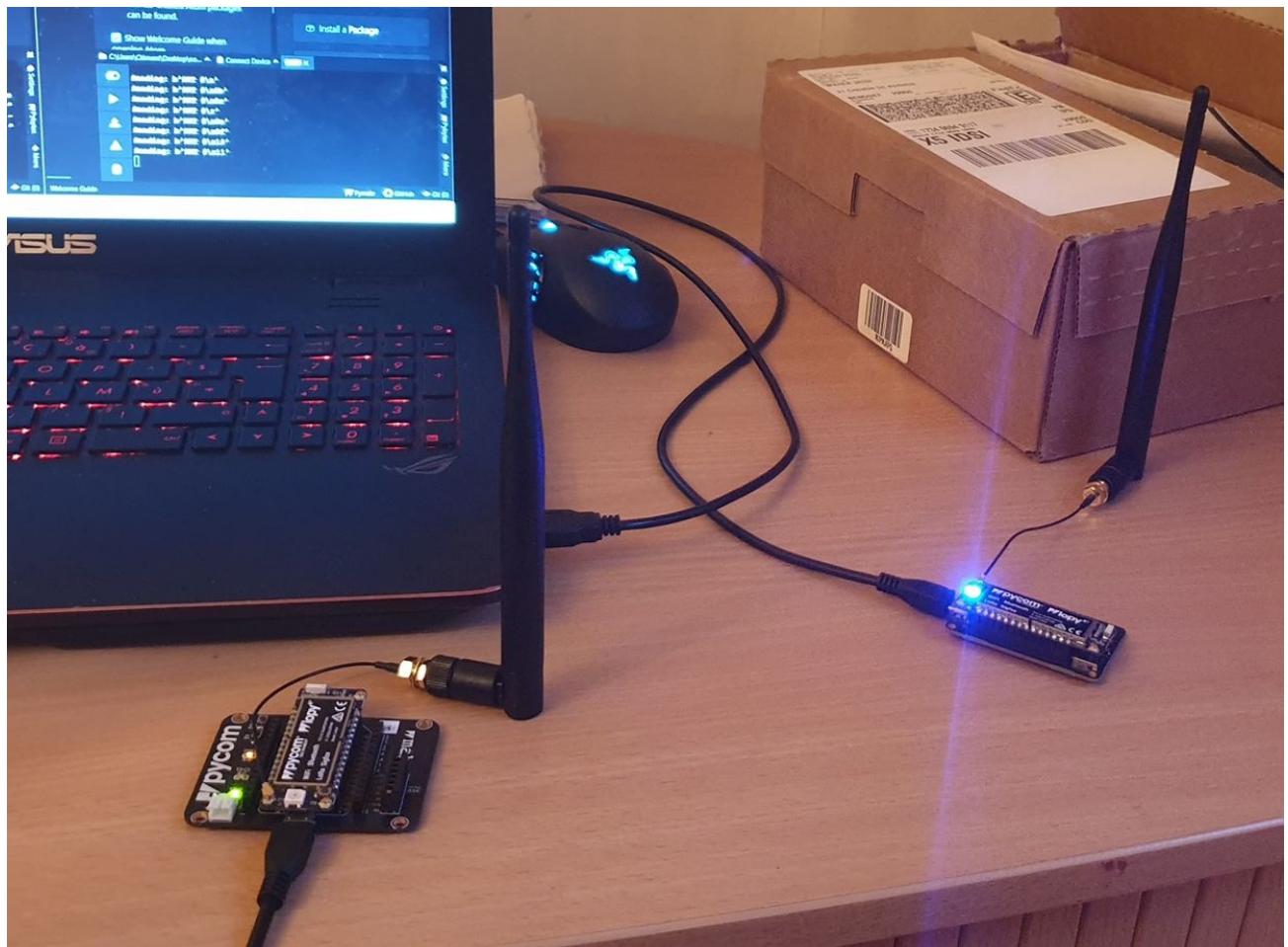

**Create account**

By registering an account you agree to our [Terms and Conditions](#) and [Privacy Policy](#).

Already have an account? [Log in](#)

# Montage global avec les cartes Pycom

Le montage ci-dessous est celui qui doit être obtenu à la fin avec une nano-passerelle LoRa (à gauche) et un noeud Pycom (à droite) :

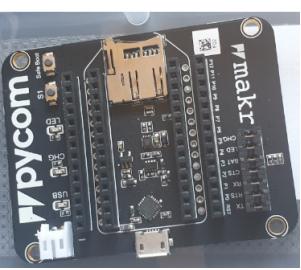


# Mettre en place une nano-passerelle Pycom

## Matériel nécessaire :

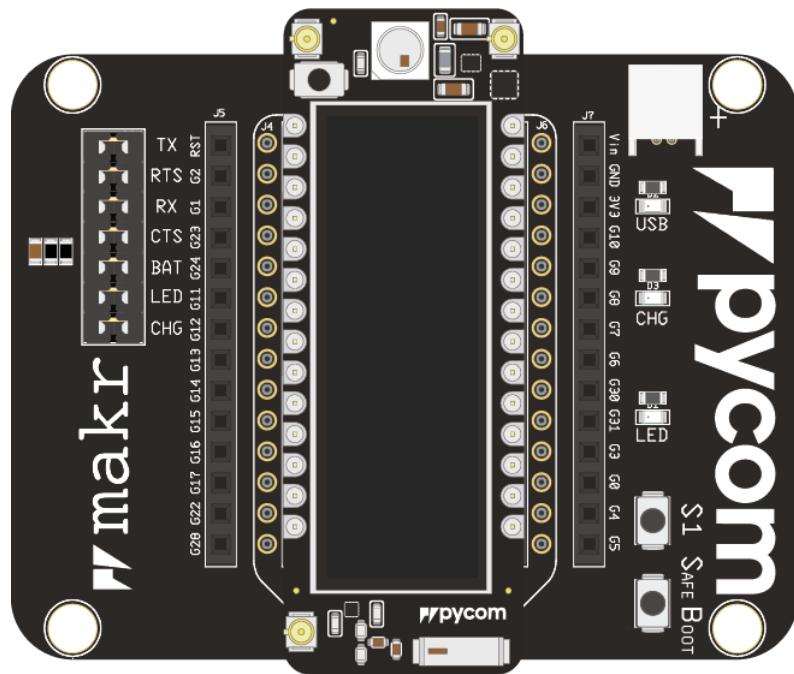
- Une carte de développement Pycom **LoPy** ou **FiPy**
- Une **carte extension 3.0** ou **Pysense** ou **Pytrack** ou **Pyscan**
- Un kit d'**antenne Pycom LoRa/Sigfox**
- Un câble micro usb
- Un pc avec le plugin **Pymark** installé dessus
- Un compte **TTN**

## Matériel utilisé pour la mise en place d'une nano-passerelle :

			
LoPy4	Carte extension 3.0	Kit d'antenne Pycom	Câble micro usb

## Montage

1. Recherchez le bouton de réinitialisation sur la carte Lopy4 (situé dans un coin de la carte, à côté de la LED).
2. Localisez le connecteur USB sur la carte d'extension.
3. Insérez le module LoPy4 sur la carte d'extension avec le bouton de réinitialisation pointant vers le connecteur USB. Il devrait s'enclencher fermement et les broches ne devraient plus être visibles.



4. Branchez le câble USB (**branchez le câble délicatement** pour ne pas abîmer la soudure)
5. Branchez l'antenne LoRa **à droite de la Led** sur la carte Lopy4

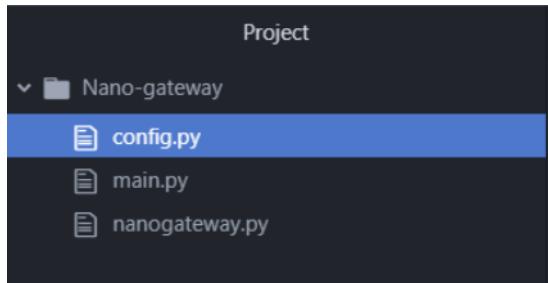
Vous devriez avoir le même rendu que sur image ci-dessous :



Si vous utilisez une autre carte extension regardez sur le lien suivant la connexion pour la carte :  
<https://docs.pycom.io/gettingstarted/connection/lopy4/>

# Mise en œuvre de la nano-passerelle

1. Créez un dossier qui sera le dossier de votre projets pour la nano-passerelle.
2. Ouvrez votre IDE, créez un projets via le dossier créé et créez les fichiers **main.py**, **config.py** et **nanogateway.py**.



3. Allez sur le lien suivant : <https://github.com/pycom/pycom-libraries/tree/master/examples/lorawan-nano-gateway>
4. Copiez coller le code pour les 3 fichiers créés ou alors allez dans le dossier **code/nano-passerelle** et récupérez le code.

## main.py

Ce fichier s'exécute au démarrage et appelle le fichier nanogateway et le fichier config.py (fichiers pour initialiser la nano-passerelle). Une fois la configuration définie, la nano-passerelle est alors démarrée.

```
1 import config
2 from nanogateway import NanoGateway
3
4 if __name__ == '__main__':
5     nanogw = NanoGateway(
6         id=config.GATEWAY_ID,
7         frequency=config.LORA_FREQUENCY,
8         datarate=config.LORA_GW_DR,
9         ssid=config.WIFI_SSID,
10        password=config.WIFI_PASS,
11        server=config.SERVER,
12        port=config.PORT,
13        ntp_server=config.NTP,
14        ntp_period=config.NTP_PERIOD_S
15    )
16
17    nanogw.start()
18    nanogw._log('You may now press ENTER to enter the REPL')
19    input()
```

## config.py

Ce fichier contient les paramètres du serveur et du réseau auxquels il se connecte. En fonction de la région de la nano-passerelle et du fournisseur (TTN, Loriot, etc.), ceux-ci varient. L'exemple fourni fonctionnera avec The Things Network (TTN) dans la région européenne de 868 Mhz. L'ID de passerelle est généré dans le script à l'aide du processus décrit ci-dessous :

```
1 from network import WLAN
2 import ubinascii
3 wl = WLAN()
4 ubinascii.hexlify(wl.mac())[:6] + 'FFFE' + ubinascii.hexlify(wl.mac())[6:]
```

Modifier les variables **WIFI\_SSID** et **WIFI\_PASS** pour qu'elles correspondent au réseau WiFi souhaité.

Vous pouvez également changer le **SF** avec **LoRa\_GW\_DR** et **LoRa\_NODE\_DR** (DR = data rate).

```
1 """ LoPy LoRaWAN Nano Gateway configuration options """
2
3 import machine
4 import ubinascii
5
6 WIFI_MAC = ubinascii.hexlify(machine.unique_id()).upper()
7 # Set the Gateway ID to be the first 3 bytes of MAC address + 'FFFE' + last
8 # 3 bytes of MAC address
9 GATEWAY_ID = WIFI_MAC[:6] + "FFFE" + WIFI_MAC[6:12]
10
11 SERVER = 'router.eu.thethings.network'
12 PORT = 1700
13
14 NTP = "pool.ntp.org"
15 NTP_PERIOD_S = 3600
16
17 WIFI_SSID = 'la SSID DE votre wifi'
18 WIFI_PASS = 'le mot de passe du wifi'
19
20 # for EU868
21 LORA_FREQUENCY = 868100000
22 LORA_GW_DR = "SF7BW125" # DR_5
23 LORA_NODE_DR = 5
```

## nanogateway.py

Ce fichier est la bibliothèque de nano-passerelles. Il contrôle toute la génération et le transfert de paquets pour les données LoRa. Cela ne nécessite aucune configuration utilisateur et la dernière version de ce code doit être téléchargée depuis le référentiel Pycom GitHub. Le protocole de communication se fait en **UDP** avec le serveur.

# Upload du project sur la carte

1. Une fois les fichiers sont configurés allez dans votre IDE, Connectez votre Appareil et appuyez sur le bouton **upload project to device**.



2. Les fichiers vont se mettre sur la carte et s'exécuter.

Si vous rencontrez quelques problèmes à ce niveau et que vous avez rempli correctement **config.py**, modifiez le code dans le fichier **nanogateway.py** :

- Modification de la fonction pour se connecter en Wifi car il peut y'avoir des problèmes de connexion (alors que le code de base aurait du fonctionner).
- Ajout de "**region = LoRa.EU868**" à 4 endroits autour des lignes 178,282,373,395 car il peut avoir un message d'erreur indiquant qu'il manque la région.

```
Upload done, resetting board...
OKets Jun  8 2016 00:22:57

rst:0x7 (TGOWDT_SYS_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
configsip: 0, SPIWP:0xee
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
mode:DIO, clock div:1
load:0x3fff8028,len:8
load:0x3fff8030,len:2156
ho 0 tail 12 room 4
load:0x4009fa00,len:19208
entry 0x400a05f4
Smart Provisioning started in the background
See https://docs.pycom.io/smart for details
[ 122.239] Starting LoRaWAN nano gateway with id: b'240AC4FFFEC74F94'
Network found!
WLAN connection succeeded!
[ 125.452] WiFi connected to: Livebox-8606
[ 125.461] wifi ok
[ 125.464] Syncing time with pool.ntp.org ...
Wifi connection established... activating device!
Unhandled exception in thread started by <bound_method>
Traceback (most recent call last):
  File "_pybytes_config.py", line 50, in smart_config
TypeError: unsupported types for __add__: 'bytes', 'NoneType'
[ 140.654] RTC NTP sync complete
[ 140.747] Opening UDP socket to router.eu.thethings.network (52.169.76.203) port 1700...
[ 140.766] Setting up the LoRa radio at 868.1 Mhz using SF7BW125
[ 140.780] LoRaWAN nano gateway online
[ 140.787] You may now press ENTER to enter the REPL
[ 140.917] Push ack
[ 165.869] Pull ack
```

Si les fichiers sont correctement configuré vous devriez avoir des message "**Push ack**" et "**Pull ack**".

```
[ 1.565] Starting LoRaWAN nano gateway with id: b'240AC4FFFEC74F94'
```

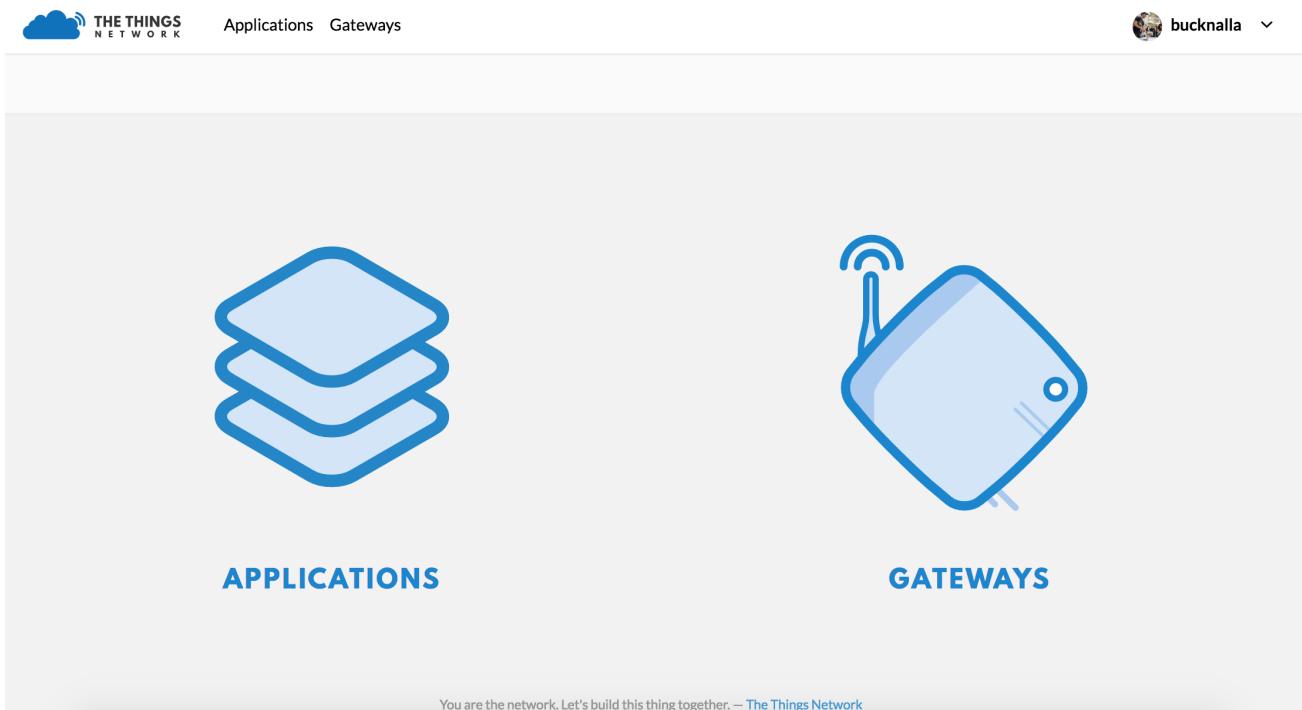
Gardez ID de la nano gateway pour la suite des événements. (ici ID est "**24 0A C4 FF FE C7 4F 94**" ).

# Enregistrement de la passerelle

Une fois le matériels configurer ry brancher au pc allez sur :

<https://console.thethingsnetwork.org/>

À l'intérieur de la console TTN, il y a deux options, applications et gateways. Sélectionnez **gateways** puis cliquez sur **register gateway**. Cela permettra la mise en place et l'enregistrement d'une nouvelle passerelle.



Sur la page Register Gateway, vous devrez définir les paramètres suivants :

Gateways > Register

## REGISTER GATEWAY

**Gateway ID**  
A unique, human-readable identifier for your gateway. It can be anything so be creative!

**I'm using the legacy packet forwarder**  
Select this if you are using the legacy [Semaphore packet forwarder](#).

**Description**  
A human-readable description of the gateway

**Frequency Plan**  
The [frequency plan](#) this gateway will use

**Router**  
The router this gateway will connect to. To reduce latency, pick a router that is in a region which is close to the location of the gateway.

**Location**  
The exact location of your gateway. This will be used if your gateway cannot determine its location by itself. Set a location by clicking on the map.

**Antenna Placement**  
The placement of the gateway antenna

Indoor    outdoor

[Cancel](#) [Register Gateway](#)

1. Cochez la case "**I'm using the legacy packet forwarder**"
2. Mettez ID de la passerelle
3. Ecrivez une description, exemple : "**Lopy4 passerelle**"
4. Mettez la fréquence de la passerelle (**Europe 868 MHZ**)
5. Le routeur devrait se mettre tout seul sinon mettez : "**ttn-router-eu**"
6. Mettez sa localisation ainsi que son placement (**intérieur ou extérieur**)

**GATEWAY OVERVIEW**

[settings](#)

**Gateway ID** eui-240ac4fffec74f94

**Description** LoPy4 Gateway student project

**Owner**  **Mythael**  [Transfer ownership](#)

**Status**  connected

**Frequency Plan** Europe 868MHz

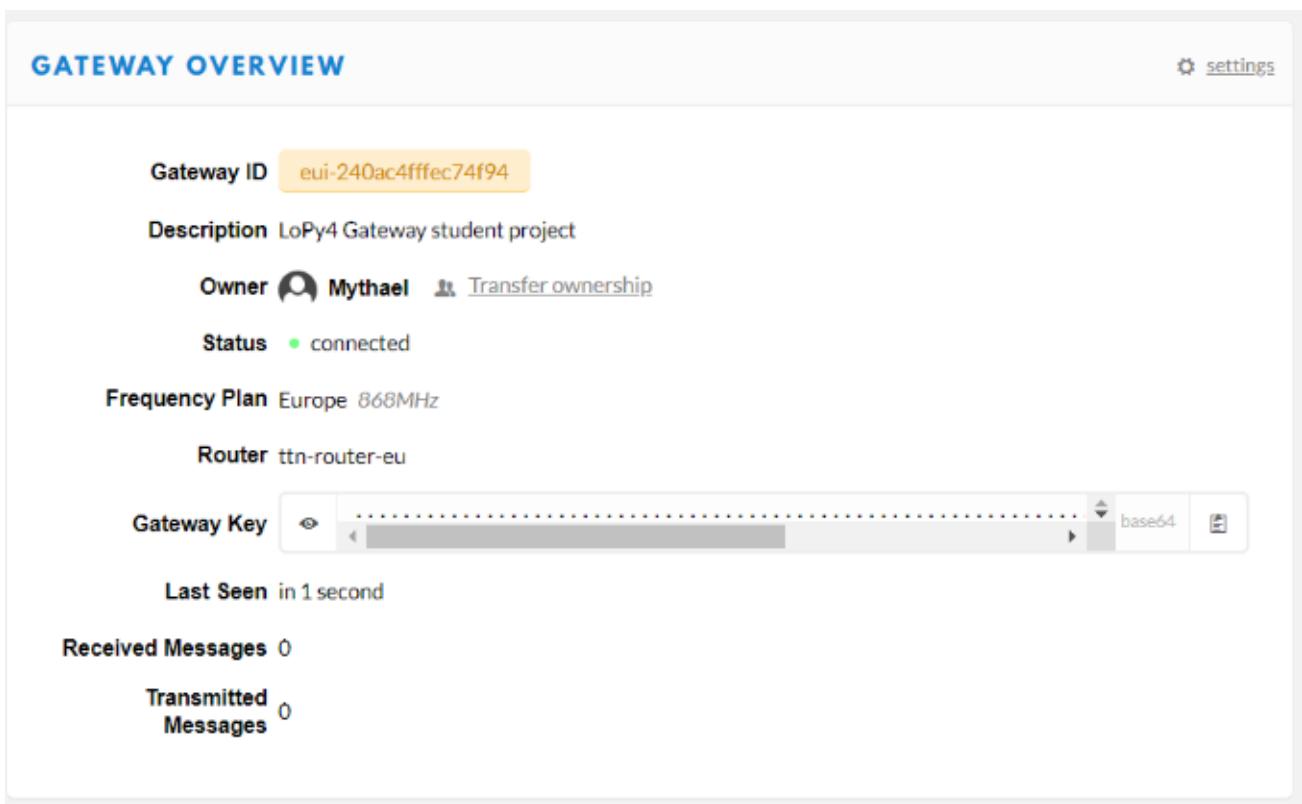
**Router** ttn-router-eu

**Gateway Key**       

**Last Seen** in 1 second

**Received Messages** 0

**Transmitted Messages** 0



Si vous avez correctement rentrez les informations vous devriez voir votre gateway connecté.

L'installation de la nano-gateway est fini.

# Noeud LoRa pour tester la nano-passerelle LoRa

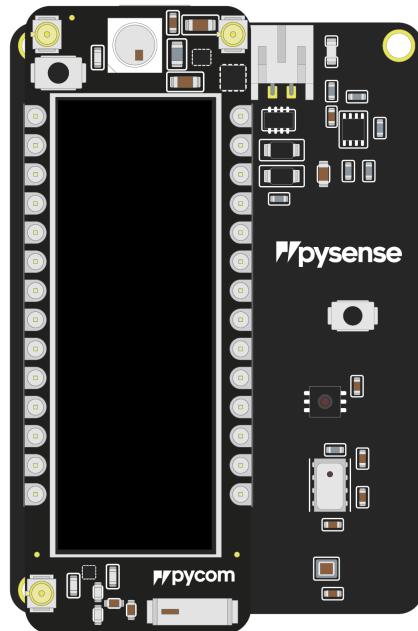
## Matériel nécessaire :

- Nano-passerelle Lora
- Une carte de développement Pycom **LoPy** ou **FiPy**
- Une **carte extension 3.0** ou **Pysense** ou **Pytrack** ou **Pyscan**
- Un kit d'**antenne Pycom LoRa/Sigfox**
- Un câble micro usb
- Un pc avec le plugin **Pymark** installé dessus
- Un compte **TTN**

			
LoPy4	Carte Pysense	Kit d'antenne Pycom	Câble micro usb

## Montage

1. Recherchez le bouton de réinitialisation sur la carte Lopy4 (situé dans un coin de la carte, à côté de la LED).
2. Localisez le connecteur USB sur la carte Pysense.
3. Insérez le module LoPy4 sur la carte pysense avec le bouton de réinitialisation pointant vers le connecteur USB. Il devrait s'enclencher fermement et les broches ne devraient plus être visibles.



4. Branchez le câble USB (**branchez le câble délicatement** pour ne pas abîmer la soudure)
5. Branchez l'antenne LoRa **à droite de la Led** sur la carte Lopy4

Vous devriez avoir le même rendu que sur image ci-dessous :

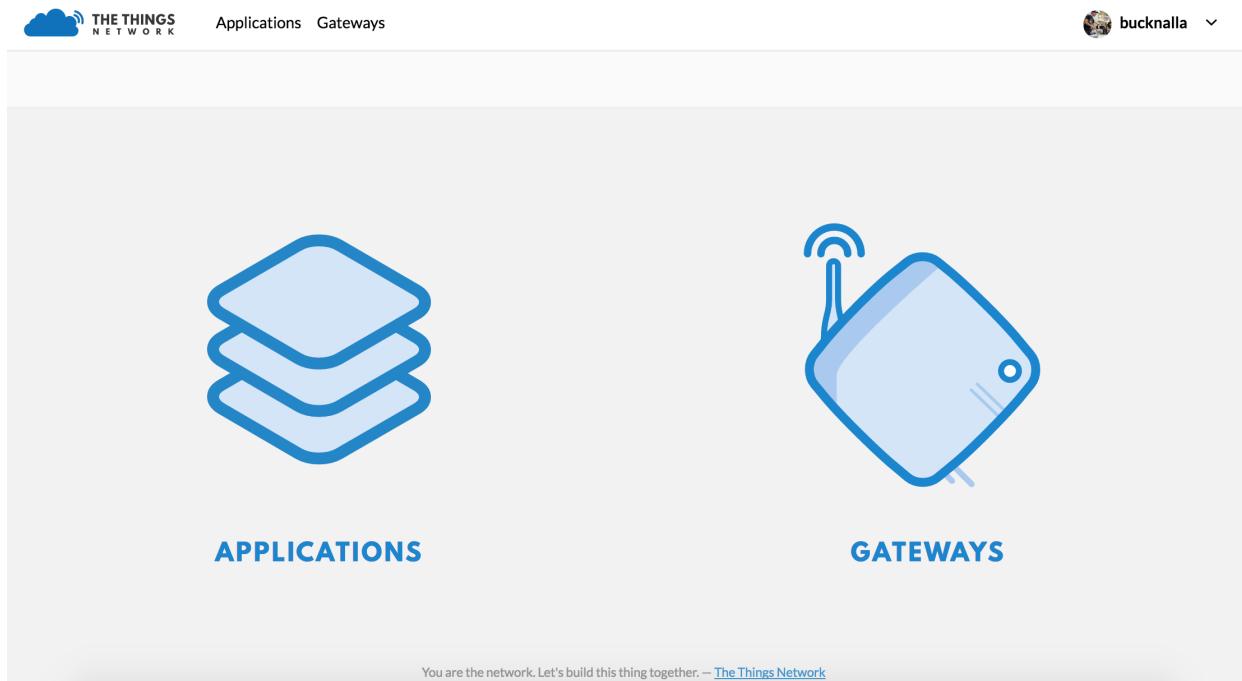


Si vous utilisez une autre carte extension regardez sur le lien suivant la connexion pour la carte :  
<https://docs.pycom.io/gettingstarted/connection/lopy4/>

# Mise en œuvre du nœud LoRa

Il faut créer une application sur **TTN** pour tester la passerelle.

1. Allez sur le lien suivant : <https://console.thethingsnetwork.org/>



2. Cliquez sur "**application**" puis sur "**add application**"



3. Sur la page register application, vous devrez définir les paramètres suivants :

A detailed screenshot of the 'ADD APPLICATION' form. It has several input fields:

- Application ID:** A placeholder text field with the instruction "The unique identifier of your application on the network".
- Description:** A placeholder text field with the instruction "A human readable description of your new app" and an example "Eg. My sensor network application".
- Application EUI:** A placeholder text field with the instruction "An application EUI will be issued for The Things Network block for convenience, you can add your own in the application settings page." and a note "EUI Issued by The Things Network".
- Handler registration:** A dropdown menu with the selected option "ttn-handler-eu".

At the bottom right of the form, there are "Cancel" and "Add application" buttons.

4. Rentrez Application ID que vous voulez (Il doit être unique dans vos applications existante)
5. Rentrez une description et appuyez sur le bouton "**Add Application**"

- Une fois application créez vous pourrez rajouter un appareil en cliquant sur "register device".

The screenshot shows the TTN Application Overview interface. At the top, it displays the Application ID as "test\_nano\_gateway", a Description of "test", and was Created "1 minute ago" by the Handler "ttn-handler-eu (current handler)". Below this, there are sections for Application EUIS (with a manage\_euis link) and Devices (with a register\_device and manage\_devices link). The Devices section shows 0 registered devices.

- Sur la page register device, vous devrez définir les paramètres suivants :

The screenshot shows the Register Device form. It includes fields for Device ID (set to "nboud\_test"), Device EUI (set to "24 0A C4 FF FE C7 6E 60"), App Key (a generated key), and App EUI (set to "70 B3 D5 7E D0 02 D9 37"). At the bottom, there are "Cancel" and "Register" buttons.

- Rentrez identifiant que vous voulez dans Device ID (Il doit être unique pour votre application).
- Rentrez le Device EUI (Il doit être unique sur le réseau, vous pouvez récupérer identifiant de votre appareil grâce au code utilisé dans la partie "Mise en place d'une nano-passerelle").

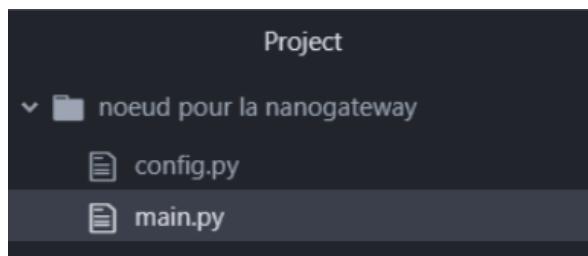
Une fois l'appareil ajouté, changez la méthode d'activation entre **OTAA** et en **ABP** fonction des préférences de l'utilisateur. Cette option se trouve sous l'onglet Paramètres.

# Configuration de la carte

1. Créez un dossier pour le noeud LoRa.
2. Créez un projet dans votre IDE a partir du dossier créez puis suivez la partie OTAA ou ABP selon vos préférences pour activation du noeud.

## OTAA

1. Allez sur le lien et copier coller le fichier **otaa\_node.py** et **config.py** dans votre dossier (le fichier config.py est le même que pour votre projet nano passerelle) : <https://github.com/pycom/pycom-libraries/tree/master/examples/lorawan-nano-gateway>
2. Renommez **otaa\_node.py** en **main.py**



3. Configurez le fichier config.py si besoin.
4. Remplissez les lignes avec les information que vous trouvez sur l'application créée précédemment :

```
dev_eui = binascii.unhexlify('dev-eui-rempli-sur-TTN')
app_eui = binascii.unhexlify('app-eui-donné-sur-TTN')
app_key = binascii.unhexlify('app-key-donné-sur-TTN-apres-avoir-rajouter-un-device')
```
5. Une fois le fichier modifié et renommé uploader les fichiers sur votre carte.

```
Load:0x4009fa00,len:19208
entry 0x400a05f4
Smart Provisioning started in the background
See https://docs.pycom.io/smart for details
id de l'appareil : () b'240AC4FFFE76E60'
Not joined yet...
joined !!!
Pending: b'PKT #\x00'
Pending: b'PKT #\x01'
Pending: b'PKT #\x02'
```

6. Si vous avez correctement rempli le fichier vous devriez vous connecter à l'application.

**DEVICE OVERVIEW**

Application ID: test\_nano\_gateway  
 Device ID: noeud\_test  
 Activation Method: OTAA

Device EUI: 24 0A C4 FF FE C7 6E 60  
 Application EUI: 78 B3 D5 7E D8 B2 D9 37  
 App Key: (redacted)  
 Device Address: 26 01 2A F7  
 Network Session Key: (redacted)  
 App Session Key: (redacted)

Status: 3 seconds ago  
 Frames up: 13 [reset frame counters](#)  
 Frames down: 0

- Sur le site de TTN vous pouvez voir les échanges du noeud et de la carte.

**GATEWAY OVERVIEW**

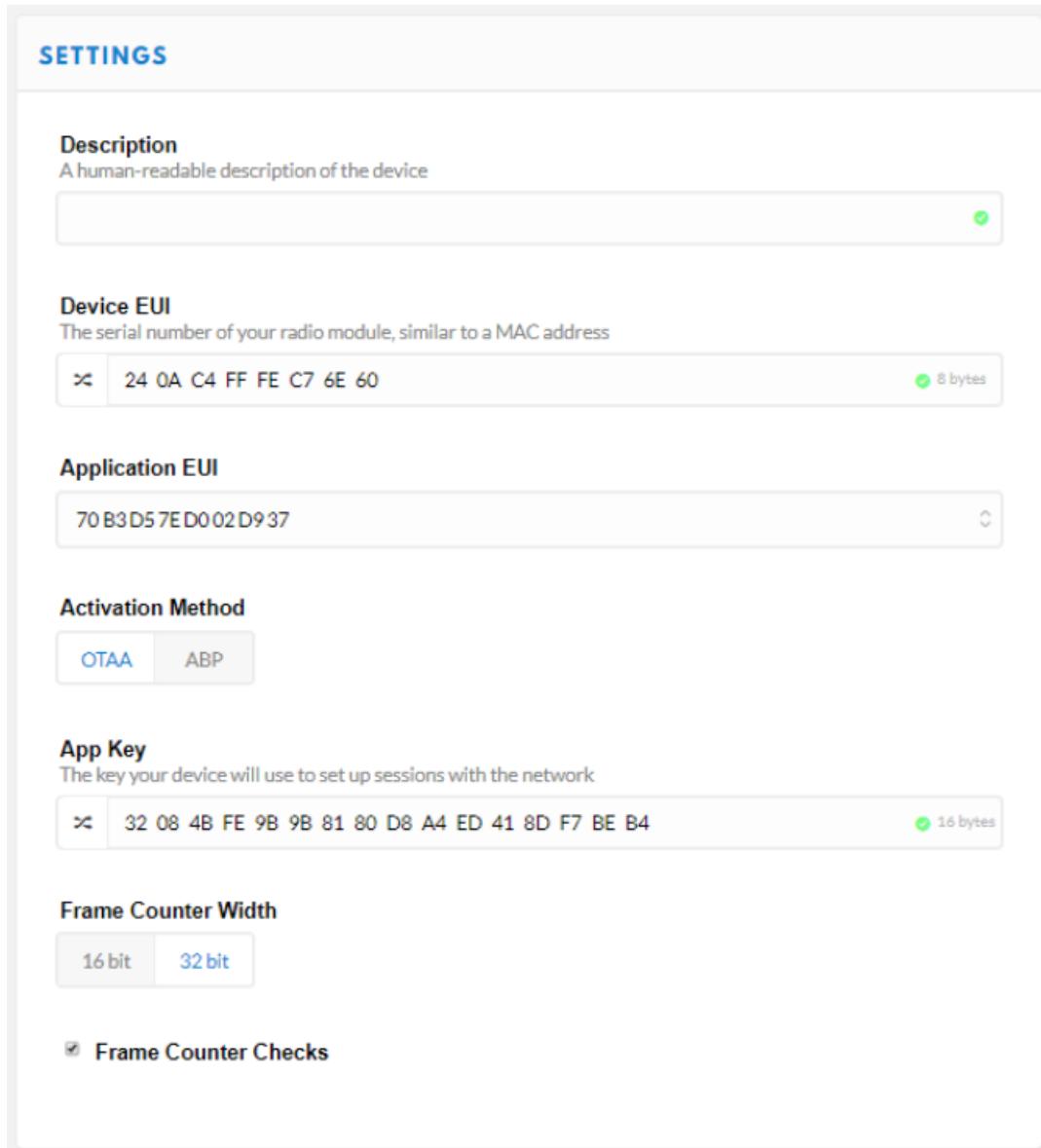
Gateway ID: eui-240ac4fffc74f94  
 Description: LoPy4 Gateway student project  
 Owner: Mythael [Transfer ownership](#)  
 Status: connected  
 Frequency Plan: Europe 868MHz  
 Router: ttu-router-eu  
 Gateway Key: (redacted)  
 Last Seen: 7 seconds ago  
 Received Messages: 220  
 Transmitted Messages: 7

- Vous pouvez également voir les messages reçus et transmis sur votre gateway (ici il y a beaucoup de messages reçus et pas transmis car l'application n'était pas bien configuré au début des tests)
- Si vous configurez mal votre noeud vous devriez voir le compteur de received message augmenté.

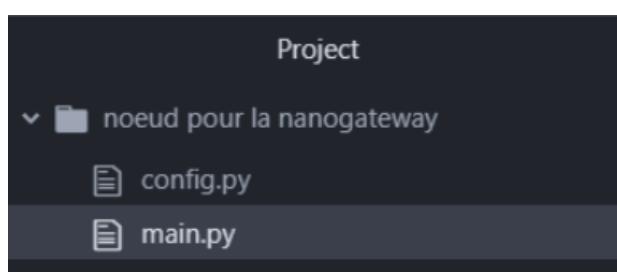
**Vous avez terminé le test de la nano-gateway en OTAA.**

# ABP

1. Allez sur le site TTN et changer dans les settings le mode activation



2. Allez sur le lien et copier coller le fichier **abp\_node.py** et **config.py** dans votre dossier (le fichier config.py est le même que pour votre projet nano passerelle) : <https://github.com/pycom/pycom-libraries/tree/master/examples/lorawan-nano-gateway>
3. Renommez **abp\_node.py** en **main.py**



4. Configurez le fichier config.py si besoin.

## DEVICE OVERVIEW

Application ID	test_nano_gateway
Device ID	noeud_test
Activation Method	ABP
Device EUI	24 0A C4 FF FE C7 6E 60
Application EUI	70 B3 D5 7E D0 02 D9 37
Device Address	26 01 1A 72
Network Session Key	E7 8B F1 26 8B FB 36 C6 E1 D4 37 5F 8E 5B A1 C7
App Session Key	E3 BF BD F3 B1 08 A5 18 44 FF E6 43 87 48 C6 C9

5. Remplissez les lignes avec les information que vous trouvez sur l'application créée précédemment :

```
dev_addr = struct.unpack(">I", binascii.unhexlify("26011A72"))[0]
nwk_swkey = binascii.unhexlify(E78BF1268BFB36C6E1D4375F8E5BA1C7)
app_swkey = binascii.unhexlify(E3BFBDFF3B108A51844FFE6438748C6C9)
```

6. Une fois le fichier renommer et modifier uploader vos fichiers sur la carte.

```
rst:0x7 (TG0WDT_SYS_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
configsip: 0, SPIWP:0xee
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
node:DIO, clock div:1
load:0x3fff8028,len:8
load:0x3fff8030,len:2156
no 0 tail 12 room 4
load:0x4009fa00,len:19208
entry 0x400a05f4
Smart Provisioning started in the background
See https://docs.pycom.io/smart for details
Sending: b'PKT #\x00'
]
```

7. Le noeud met beaucoup moins de temps qu'en OTAA à envoyer des données sur la passerelle.

8. vous pourrez voir si l'appareil est bien connecté en ABP sur le site TTN.

The screenshot shows the 'DEVICE OVERVIEW' section of the TTN website. It displays the following device information:

- Application ID:** test\_nano\_gateway
- Device ID:** noeud\_test
- Activation Method:** ABP
- Device EUI:** 24 0A C4 FF FE C7 6E 60
- Application EUI:** 70 B3 05 7E D0 02 D9 37
- Device Address:** 26 01 1A 72
- Network Session Key:** (redacted)
- App Session Key:** (redacted)
- Status:** 10 seconds ago
- Frames up:** 26 ([reset frame counters](#))
- Frames down:** 4

9. Vous pouvez également envoyer des messages pour voir si l'appareil reçoit bien les messages de la passerelle.

**Vous avez terminé le test de la nano-gateway en ABP.**

# Mettre en place une communication TCP entre la nano-passerelle et le serveur OVH

Le code dans les fichiers **config.py** et **nanogateway.py** de la passerelle LoRa ont été modifié afin d'envoyer les données LoRa directement au serveur OVH via le protocole **TCP**.

## config.py

Un serveur TCP a été mis en place sur le serveur OVH à l'adresse **51.83.79.109** sur le port **9999**.

Il faut donc remplacer le serveur TTN par le serveur OVH :

```
1 SERVER = '51.83.79.109'  
2 PORT = 9999
```

## nanogateway.py

Le code a été réécrit pour communiquer en TCP et contrairement à la communication UDP qui avait été mis en place on se déconnecte du serveur après chaque envoi de donnée.

La fonction pour envoyer les données en TCP :

```
1 def _push_data(self, data):  
2     try:  
3         self._log('Opening TCP socket to OVH server {} port {}...',  
4         , self.server_ip[0], self.server_ip[1])  
5         self.sock = usocket.socket(usocket.AF_INET, usocket.  
6             SOCK_STREAM) # AF_INET : IPv4 ; SOCK_STREAM : TCP  
7         self.sock.connect(self.server_ip)  
8         self.sock.send(data)  
9         request = self.sock.recv(1024) # taille du buffer, doit etre  
10        une puissance de 2  
11        print ("[*] Received: {}", request)  
12        self.sock.close()  
13    except Exception as ex:  
14        self._log('Failed to push uplink packet to server: {}', ex)
```

Dans cette fonction on se connecte au serveur TCP, on envoie la donnée puis on attend de recevoir un message de confirmation avant de stopper la communication si l'envoi ne marche pas on écrit un message dans les logs pour informer que le paquet de donné n'a pas été envoyé.

# Mise en œuvre

1. Récupérez les fichiers dans le dossier code puis nanopasserelle\_OVH.
2. Remplacez les fichiers de la passerelle par les nouveau fichiers et si besoin configurer la WiFi dans le fichier config.py.
3. Uploadez les fichiers sur la carte.
4. Lors de l'exécution du code on envoie des données vide avant de vérifier la communication.
5. Si la configuration est correctement rentrée vous devriez avoir l'image ci-dessous :

```
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
mode:DIO, clock div:1
load:0x3fff8028,len:8
load:0x3fff8030,len:2156
ho 0 tail 12 room 4
load:0x4009fa00,len:19208
entry 0x400a05f4
Smart Provisioning started in the background
See https://docs.pycom.io/smart for details
[ 112.269] Starting LoRaWAN nano gateway with id: b'240AC4FFFEC74F94'
Network found!
WLAN connection succeeded!
[ 116.396] WiFi connected to: Livebox-8606
[ 116.405] wifi ok
[ 116.407] Syncing time with pool.ntp.org ...
[ 116.516] RTC NTP sync complete
[ 116.527] Opening tCP socket to OVH server (192.168.1.25) port 9999...
Wifi connection established... activating device!
Unhandled exception in thread started by <bound_method>
Traceback (most recent call last):
  File "_pybytes_config.py", line 50, in smart_config
TypeError: unsupported types for __add__: 'bytes', 'NoneType'
[*] Received: {} b'ACK'
[ 116.863] Setting up the LoRa radio at 868.1 Mhz using SF7BW125
[ 116.876] LoRaWAN nano gateway online
[ 116.884] You may now press ENTER to enter the REPL
[]
```

Le serveur envoie ACK pour validé l'envoie de la donnée.

6. Vous pouvez vérifier la connexion en simulant un serveur TCP sans le serveur OVH sur votre PC. En utilisant par exemple le script suivant en python3 :

```

1 import socket
2 import threading
3
4 bind_ip = "IP-de-votre-machine"
5 bind_port = 9999
6
7 server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
8
9 server.bind((bind_ip,bind_port))
10
11 server.listen(5) #backlog : nombre de connexions non-acceptees avant de
12     refuser de nouvelles connexions
13 print ("[*] Listening on {}:{}".format(bind_ip,bind_port))
14 # thread de gestion des clients
15
16 def handle_client(client_socket):
17
18     # ce qu'on a recu
19     request = client_socket.recv(1024)
20
21     print ("[*] Received: {}".format(request))
22
23     # envoie de la reponse
24     client_socket.send('ACK'.encode())
25     client_socket.close()
26     print("socket ferme")
27 while True:
28     client,addr = server.accept()
29     print ("[*] Accepted connection from: {}:{}".format(addr[0],addr[1]))
30
31     # lancement du thread client
32     client_handler = threading.Thread(target=handle_client,args=(client,))
33         # si tu sais a quoi sert la virgule toute seule a la fin...
34     client_handler.start()

```

```

[*] Listening on {}:{} 192.168.1.25 9999
[*] Accepted connection from: {}:{} 192.168.1.44 64222
[*] Received: {} b'{"stat": {"alti": 0, "rxok": 0, "rxfw": 0, "ackr": 100.0, "dwnb": 0, "long": 0, "txnb": 0, "rxnb": 0, "time": "2020-04-19 13:43:29 GMT", "lati": 0}}'
socket fermé

```

**Vous avez terminé la mise en place de la communication entre la passerelle et le serveur en TCP.**

# Utiliser les capteurs de la carte Pysense

Dans cette partie nous allons voir comment exploiter les capteurs de la carte Pysense et les envoyer à la passerelle LoRa.

## Matériel nécessaire :

- Nano-passerelle Lora
- Une carte de développement Pycom **LoPy** ou **FiPy**
- Une carte **Pysense**
- Un kit d'antenne Pycom **LoRa/Sigfox**
- Un câble micro usb
- Un pc avec le plugin **Pymark** installé dessus

## Montage

Si vous ne l'avez pas encore fait suivez les étapes de la partie **Noeud LoRa pour tester la nano-passerelle** jusqu'à la partie **Montage** puis revenez sur cette partie une fois le matériel monté.

Les capteurs de la carte Pysense :

1. Accéléromètre à 3 axes (LIS2HH12)
2. Capteur numérique de lumière ambiante (LTR-329ALS-01)
3. Capteur d'humidité et de température (SI7006A20)
4. Capteur de pression barométrique avec altimètre (MPL3115A2)

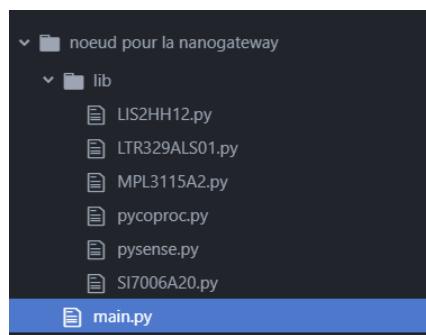
Pour pouvoir utiliser les différents capteurs il faudra récupérer les bibliothèques des capteurs sur le github de Pycom. Il faudra également la bibliothèque de **Pycoproc** également sur le github.

Pycoproc permet de gérer l'interaction avec le MCU PIC.

# Mise en œuvre des capteurs

## Installation des bibliothèques

1. Créez un dossier pour votre noeud et créez un dossier lib dans votre dossier.
2. Allez sur le lien suivant pour récupérer les bibliothèque des capteurs :  
<https://github.com/pycom/pycom-libraries/tree/master/pysense>
3. Récupérez les fichiers **LIS2HH12.py**, **LTR329ALS01.py**, **MPL3115A2.py**, **SI7006A20.py** et **pysense.py** et placez les dans le fichier **lib** que vous avez créé.
4. Récupérez également le fichier **main.py** et placez le dans votre dossier.
5. Allez sur le lien suivant pour récupérer la bibliothèque de Pycoproc :  
<https://github.com/pycom/pycom-libraries/tree/master/lib/pycoproc>
6. Récupérez le fichiers **pycoproc.py** et placez le dans le fichier **lib** que vous avez créé.



7. Ouvrez le dossier comme un projet sur votre IDE
8. Uploadez vos fichiers sur le noeud LoRa
9. Si vous avez bien mis en place les fichiers vous devriez avoir l'affichage suivant :

```
C:\Users\Clément\Desktop\noeud pour la nanogateway ▾ Connect Device ▾ COM6 ×
configsip: 0, SPIWP:0xee
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
mode:DIO, clock div:1
load:0x3fff8028, len:8
load:0x3fff8030, len:2156
ho 0 tail 12 room 4
load:0x4009fa00, len:19208
entry 0x400a05f4
Smart Provisioning started in the background
See https://docs.pycom.io/smart for details
MPL3115A2 temperature: 19.9375
Altitude: 33.8125
Pressure: 100942.5
Temperature: 20.70739 deg C and Relative Humidity: 58.38065 %RH
Dew point: 12.24542 deg C
Humidity Ambient for 24.4 deg C is 46.70204%RH
Light (channel Blue lux, channel Red lux): (0, 0)
Acceleration: (0.0, 0.0, 0.0)
Roll: 1.978917
Pitch: -0.7478532
Battery voltage: 4.616452
```

# Explication des fonctions disponibles

Vous pouvez maintenant utiliser les capteurs de la carte Pycom voici les fonctions disponibles grâce aux bibliothèques installées que vous pourrez exécuter dans votre programme :

## Accéléromètre à 3 axes (LIS2HH12)

Pysense dispose d'un accéléromètre à 3 axes qui fournit des sorties pour l'accélération ainsi que le roulis, le tangage et le lacet.

```
class LIS2HH12(pysense = None, sda = 'P22', scl = 'P21')
```

Il faut en premier lieu créer un objet **LIS2HH12** qui renverra des valeurs d'accélération, de roulis, de tangage et de lacet. Le constructeur doit recevoir un objet Pysense ou I2C pour réussir la construction.

**LIS2HH12.acceleration ()** : Renvoie **un tuple**(une liste non modifiable) avec les 3 valeurs d'accélération (G).

**LIS2HH12.roll ()** : Renvoie **un float** en degrés compris entre -180 et 180.

**LIS2HH12.pitch ()** : Renvoie **un float** en degré compris entre -90 et 90. Une fois que la carte a basculé au-delà de cette plage, les valeurs se répéteront. Cela est dû à un manque de mesure de lacet, ce qui ne permet pas de connaître l'orientation exacte de la planche.

## Capteur numérique de lumière ambiante (LTR-329ALS-01)

Pysense possède un capteur de lumière double qui fournit des sorties pour les niveaux de lumière externe en lux.

```
class LTR329ALS01(pysense = None, sda = 'P22', scl = 'P21', gain = ALS_GAIN_1X, integration = ALS_INT_100, rate = ALS_RATE_500)
```

Il faut en premier lieu créer un objet **LTR329ALS01** qui renverra les valeurs de lumière en lux. Le constructeur doit recevoir un objet Pysense ou I2C pour réussir la construction.

**LTR329ALS01.light ()** : Renvoie **un tuple**(une liste non modifiable) avec deux valeurs pour les niveaux d'éclairage en lux

Argument pour le constructeur modifiable :

**Gain** : ALS\_GAIN\_1X, ALS\_GAIN\_2X, ALS\_GAIN\_4X, ALS\_GAIN\_8X, ALS\_GAIN\_48X, ALS\_GAIN\_96X

**Integration** : ALS\_INT\_50, ALS\_INT\_100, ALS\_INT\_150, ALS\_INT\_200, ALS\_INT\_250, ALS\_INT\_300, ALS\_INT\_350, ALS\_INT\_400

**Rate** : ALS\_RATE\_50, ALS\_RATE\_100, ALS\_RATE\_200, ALS\_RATE\_500, ALS\_RATE\_1000, ALS\_RATE\_2000

## Capteur d'humidité et de température (SI7006A20)

Pysense dispose d'un capteur d'humidité et de température qui fournit des valeurs d'humidité relative et de température extérieure.

```
class SI7006A20 (pysense = None, sda = 'P22', scl = 'P21')
```

Il faut en premier lieu créer un objet **SI7006A20** qui renverra des valeurs d'humidité (%) et de température ('C). Le constructeur doit recevoir un objet Pysense ou I2C pour réussir la construction.

**SI7006A20.humidity ()** : Renvoie **un float** avec le pourcentage d'humidité relative.

**SI7006A20.température ()** : Renvoie **un float** avec la température.

## Capteur de pression barométrique avec altimètre (MPL3115A2)

Pysense possède un capteur de pression barométrique qui fournit des relevés de pression, d'altitude ainsi qu'un capteur de température supplémentaire.

```
class MPL3115A2 (pysense = None, sda = 'P22', scl = 'P21', mode = PRESSURE)
```

Il faut en premier lieu créer un objet **MPL3115A2** qui renverra des valeurs de pression (Pa), d'altitude (m) et de température ('C). Le constructeur doit recevoir un objet Pysense ou I2C pour réussir la construction.

**MPL3115A2.pressure ()** : Renvoie **un float** avec la pression en (Pa).

**MPL3115A2.altitude ()** : Renvoie **un float** avec l'altitude en (m).

**MPL3115A2.température ()** : Renvoie **un float** avec la température en ('C).

Argument pour le constructeur modifiable :

**Mode** : PRESSURE,ALTITUDE

# Envoyer les données des capteurs en LoRa

Maintenant que nous savons récupérer les valeurs des capteurs nous allons les envoyés sur la passerelle LoRa.

1. Tout d'abord il faudra importer les bibliothèques pour la communication LoRa.

```
1 #!/usr/bin/env python
2 import socket
3 import time
4 import pycom
5 from network import LoRa
6 from pysense import Pysense
7 from LIS2HH12 import LIS2HH12
8 from SI7006A20 import SI7006A20
9 from LTR329ALS01 import LTR329ALS01
10 from MPL3115A2 import MPL3115A2, ALTITUDE, PRESSURE
```

2. Nous allons ensuite initialiser une communication LoRa en brute (Il n'aura pas d'identification ABP ou OTAA) :

```
1 # initialize LoRa in LORA mode.
2 lora = LoRa(mode=LoRa.LORA, region=LoRa.EU868)
3
4 # for EU868
5 LORA_FREQUENCY = 868100000
6
7 # # remove all the non-default channels
8 for i in range(3, 16):
9     lora.remove_channel(i)
10
11 # set the 3 default channels to the same frequency
12 lora.add_channel(0, frequency=LORA_FREQUENCY, dr_min=0, dr_max=5)
13 lora.add_channel(1, frequency=LORA_FREQUENCY, dr_min=0, dr_max=5)
14 lora.add_channel(2, frequency=LORA_FREQUENCY, dr_min=0, dr_max=5)
15
16 # create a LoRa socket
17 s = socket.socket(socket.AF_LORA, socket.SOCK_RAW)
18
19 # # make the socket non-blocking
20 s.setblocking(False)
```

3. On initialiser la carte avec une led de couleur blanche pour récupérer les données des capteurs :

```
1 pycom.heartbeat(False)
2 pycom.rgbled(0x0A0A08) # white
3
4 py = Pysense()
```

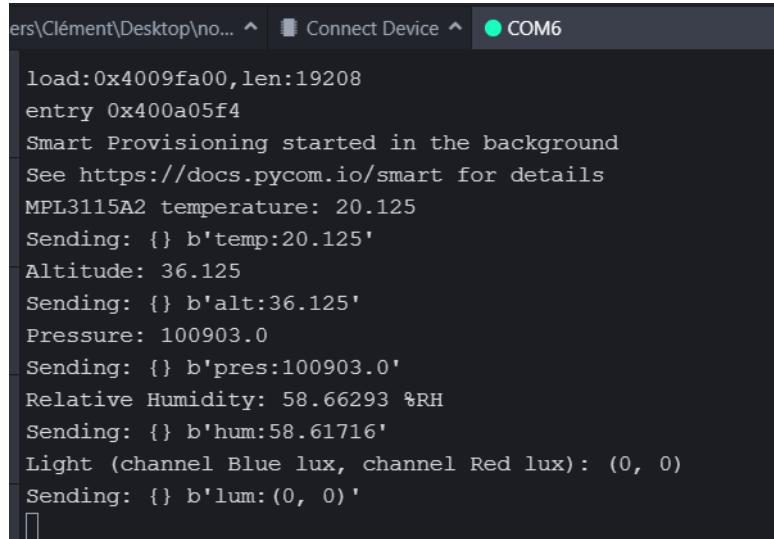
4. On va envoyer chaque donnée qui nous intéresse (ici j'ai récupéré la valeur de température du capteur de pression barométrique)

```
1 #Temperature
2 mp = MPL3115A2(py, mode=ALTITUDE) # Returns height in meters. Mode may
   also be set to PRESSURE, returning a value in Pascals
3
4 print("MPL3115A2 temperature: " + str(mp.temperature()))
5 temperature= b'temp:' + str(mp.temperature())
6 print('Sending: {}', temperature)
7 s.send(temperature)
8 time.sleep(2)
```

5. quand on aura terminer envoie des données des capteurs on mettra la carte en mode "**SLEEP**".

```
1 py.setup_sleep(3600) #time in sec
2 py.go_to_sleep()
```

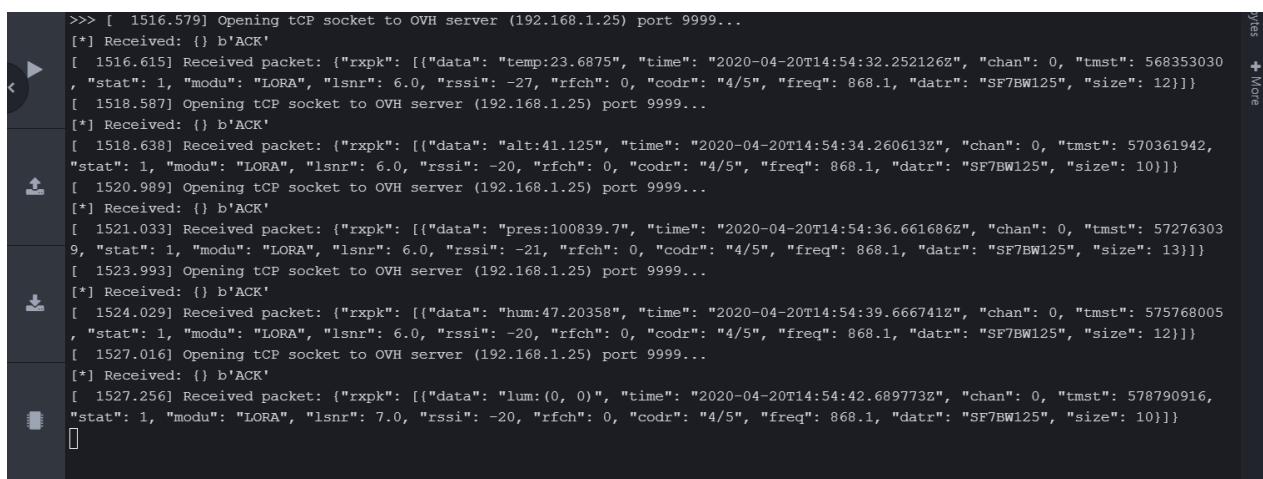
Vous pouvez aussi récupérer dans le dossier code le dossier **nœud\_pysense\_OVH** pour utiliser le capteurs. Il envoie les données suivantes :



```
load:0x4009fa00,len:19208
entry 0x400a05f4
Smart Provisioning started in the background
See https://docs.pycom.io/smart for details
MPL3115A2 temperature: 20.125
Sending: {} b'temp:20.125'
Altitude: 36.125
Sending: {} b'alt:36.125'
Pressure: 100903.0
Sending: {} b'pres:100903.0'
Relative Humidity: 58.66293 %RH
Sending: {} b'hum:58.61716'
Light (channel Blue lux, channel Red lux): (0, 0)
Sending: {} b'lum:(0, 0)'
[]
```

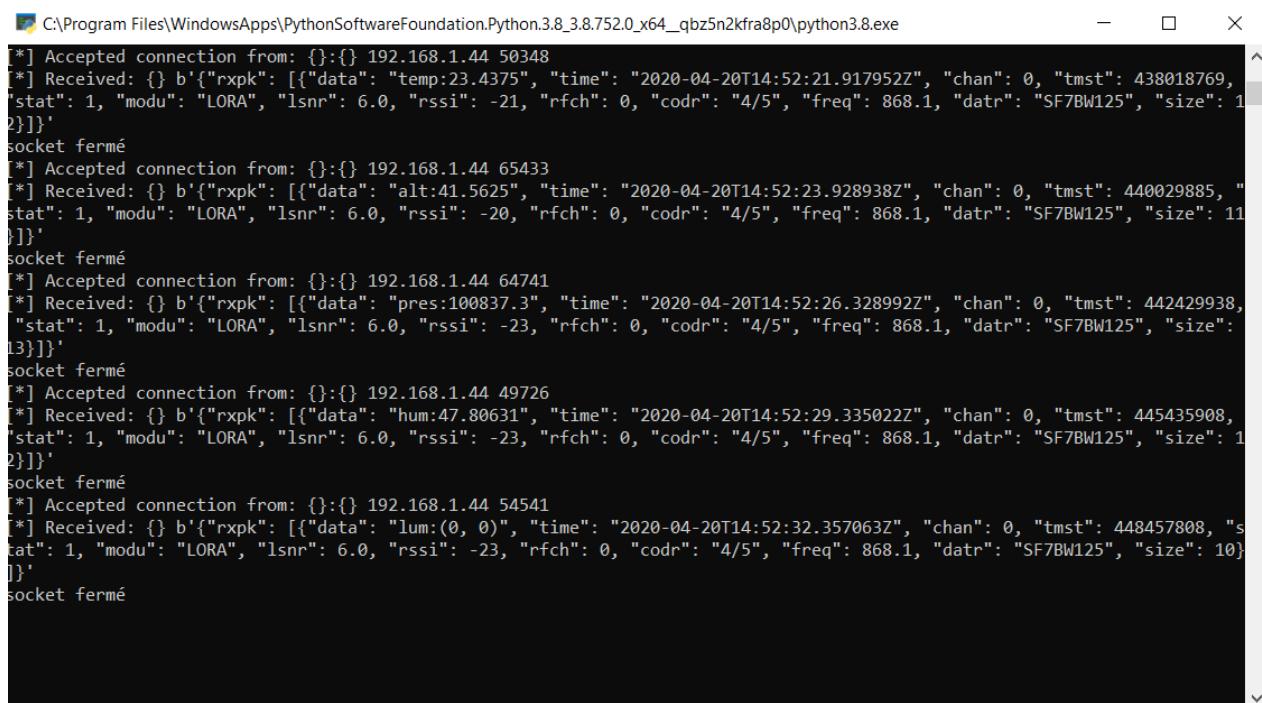
Vous pouvez voir si l'envoie est bien réalisé sur votre nano-passerelle Pycom (il faudra également que la communication soit configurer en brute sur la passerelle) :

```
1     # initialize the LoRa radio in LORA mode
2     self._log('Setting up the LoRa radio at {} Mhz using {}'.format(self.
3         freq_to_float(self.frequency), self.datarate)
4         self.lora = LoRa(
5             mode=LORA.LORA,
6             region = LoRa.EU868,
```



```
>>> [ 1516.579] Opening tCP socket to OVH server (192.168.1.25) port 9999...
[*] Received: {} b'ACK'
[ 1516.615] Received packet: {"rxpk": [{"data": "temp:23.6875", "time": "2020-04-20T14:54:32.252126Z", "chan": 0, "tmst": 568353030
, "stat": 1, "modu": "LORA", "lsnr": 6.0, "rss": -27, "rfch": 0, "codr": "4/5", "freq": 868.1, "datr": "SF7BW125", "size": 12}]}
[ 1518.587] Opening tCP socket to OVH server (192.168.1.25) port 9999...
[*] Received: {} b'ACK'
[ 1518.638] Received packet: {"rxpk": [{"data": "alt:41.125", "time": "2020-04-20T14:54:34.260613Z", "chan": 0, "tmst": 570361942
, "stat": 1, "modu": "LORA", "lsnr": 6.0, "rss": -20, "rfch": 0, "codr": "4/5", "freq": 868.1, "datr": "SF7BW125", "size": 10}]}
[ 1520.989] Opening tCP socket to OVH server (192.168.1.25) port 9999...
[*] Received: {} b'ACK'
[ 1521.033] Received packet: {"rxpk": [{"data": "pres:100839.7", "time": "2020-04-20T14:54:36.661686Z", "chan": 0, "tmst": 57276303
, "stat": 1, "modu": "LORA", "lsnr": 6.0, "rss": -21, "rfch": 0, "codr": "4/5", "freq": 868.1, "datr": "SF7BW125", "size": 13}]}
[ 1523.993] Opening tCP socket to OVH server (192.168.1.25) port 9999...
[*] Received: {} b'ACK'
[ 1524.029] Received packet: {"rxpk": [{"data": "hum:47.20358", "time": "2020-04-20T14:54:39.666741Z", "chan": 0, "tmst": 575768005
, "stat": 1, "modu": "LORA", "lsnr": 6.0, "rss": -20, "rfch": 0, "codr": "4/5", "freq": 868.1, "datr": "SF7BW125", "size": 12}]}
[ 1527.016] Opening tTCP socket to OVH server (192.168.1.25) port 9999...
[*] Received: {} b'ACK'
[ 1527.256] Received packet: {"rxpk": [{"data": "lum:(0, 0)", "time": "2020-04-20T14:54:42.689773Z", "chan": 0, "tmst": 578790916
, "stat": 1, "modu": "LORA", "lsnr": 7.0, "rss": -20, "rfch": 0, "codr": "4/5", "freq": 868.1, "datr": "SF7BW125", "size": 10}]}]
```

En simulant un serveur TCP sur votre PC vous pouvez également voir que les données sont bien envoyé en brute.



```
C:\Program Files\WindowsApps\PythonSoftwareFoundation.Python.3.8_3.8.752.0_x64_qbz5n2kfra8p0\python3.8.exe
[*] Accepted connection from: {}:{} 192.168.1.44 50348
[*] Received: {} b'{"rxpk": [{"data": "temp:23.4375", "time": "2020-04-20T14:52:21.917952Z", "chan": 0, "tmst": 438018769, "stat": 1, "modu": "LORA", "lsnr": 6.0, "rss": -21, "rfch": 0, "codr": "4/5", "freq": 868.1, "datr": "SF7BW125", "size": 12}]}'
socket fermé
[*] Accepted connection from: {}:{} 192.168.1.44 65433
[*] Received: {} b'{"rxpk": [{"data": "alt:41.5625", "time": "2020-04-20T14:52:23.928938Z", "chan": 0, "tmst": 440029885, "stat": 1, "modu": "LORA", "lsnr": 6.0, "rss": -20, "rfch": 0, "codr": "4/5", "freq": 868.1, "datr": "SF7BW125", "size": 11}]}'
socket fermé
[*] Accepted connection from: {}:{} 192.168.1.44 64741
[*] Received: {} b'{"rxpk": [{"data": "pres:100837.3", "time": "2020-04-20T14:52:26.328992Z", "chan": 0, "tmst": 442429938, "stat": 1, "modu": "LORA", "lsnr": 6.0, "rss": -23, "rfch": 0, "codr": "4/5", "freq": 868.1, "datr": "SF7BW125", "size": 13}]}'
socket fermé
[*] Accepted connection from: {}:{} 192.168.1.44 49726
[*] Received: {} b'{"rxpk": [{"data": "hum:47.80631", "time": "2020-04-20T14:52:29.335022Z", "chan": 0, "tmst": 445435908, "stat": 1, "modu": "LORA", "lsnr": 6.0, "rss": -23, "rfch": 0, "codr": "4/5", "freq": 868.1, "datr": "SF7BW125", "size": 12}]}'
socket fermé
[*] Accepted connection from: {}:{} 192.168.1.44 54541
[*] Received: {} b'{"rxpk": [{"data": "lum:(0, 0)", "time": "2020-04-20T14:52:32.357063Z", "chan": 0, "tmst": 448457808, "stat": 1, "modu": "LORA", "lsnr": 6.0, "rss": -23, "rfch": 0, "codr": "4/5", "freq": 868.1, "datr": "SF7BW125", "size": 10}]}'
socket fermé
```

#### Remarque :

Vous pouvez également récupérez sur votre carte la tension de la batterie. (`py.read_battery_voltage()`)

**Vous savez désormais comment envoyer les données des capteurs de la carte Pysense via une communication LoRa en brute.**

# Références

- [1] <https://docs.pycom.io/>
- [2] <https://code.visualstudio.com/>
- [3] <https://nodejs.org/en/>
- [4] <https://atom.io/>
- [5] <http://dfu-util.sourceforge.net/releases/dfu-util-0.9-win64.zip>
- [6] <http://zadig.akeo.ie/>
- [7] <https://console.thethingsnetwork.org/>
- [8] <https://github.com/pycom/pycom-libraries/tree/master/examples/lorawan-nano-gateway>
- [9] <https://core-electronics.com.au/tutorials/building-a-lorawan-nano-gateway-to-the-things-network.html>
- [10] <https://python.doctor/page-reseaux-sockets-python-port>