 **Warning:** Trying to access array offset on value of type null in **/home/clients/ee3f943e731fc0a12d3400116692186f/html/wp-content/plugins/unyson/framework/includes/option-types/typography-v2/class-fw-option-type-typography-v2.php** on line **148**

Warning: foreach() argument must be of type array|object, null given in **/home/clients/ee3f943e731fc0a12d3400116692186f/html/wp-content/plugins/unyson/framework/includes/option-types/typography-v2/class-fw-option-type-typography-v2.php** on line **148**

Warning: Trying to access array offset on value of type null in **/home/clients/ee3f943e731fc0a12d3400116692186f/html/wp-content/plugins/unyson/framework/helpers/general.php** on line **1275**

Warning: foreach() argument must be of type array|object, null given in **/home/clients/ee3f943e731fc0a12d3400116692186f/html/wp-content/plugins/unyson/framework/helpers/general.php** on line **1275**



SMART BUD

Demo (<https://bud.eco-sensors.ch>) | Tutoriels (<https://eco-sensors.ch/tutoriels/>)



NOUS SOUTENIR

Shop (<https://eco-sensors.ch/shop/>) & Sponsoring (<https://eco-sensors.ch/product-category/dons/>)



SOCIAL

Facebook (<https://www.facebook.com/ecosensors>) - **Contact** (<https://eco-sensors.ch/contact/>)

Mesure de la qualité de l'air (PM2.5/PM10) – Python3

HOME ([HTTPS://ECO-SENSORS.CH/](https://eco-sensors.ch/))
/ DIY ([HTTPS://ECO-SENSORS.CH/CATEGORY/DIY/](https://eco-sensors.ch/category/diy/))
/ MESURE DE LA QUALITÉ DE L'AIR (PM2.5/PM10) –
PYTHON3

BY ECOSENSORS ([HTTPS://ECO-SENSORS.CH/AUTHOR/](https://eco-sensors.ch/author/ecosensors/)
ECOSENSORS/

/

11 JUIN 2020 ([HTTPS://ECO-SENSORS.CH/MESURE-DE-LA-QUALITE-](https://eco-sensors.ch/mesure-de-la-qualite-de-lair-pm2-5-pm10-python3/)
DE-LAIR-PM2-5-PM10-PYTHON3/)

 0  4,198  0  

DIY ([HTTPS://ECO-SENSORS.CH/CATEGORY/DIY/](https://eco-sensors.ch/category/diy/))






Dans cet article, je vais vous montrer comment préparer un Raspberry et Python3 pour mesurer les poussières fines PM2.5 et PM10 et comment transmettre les mesures à un serveur distant grâce à LoRaWAN

J'attire votre attention que cet article n'est pas un tutoriel, mais mes notes. Il vous apportera les outils et des astuces pour le faire, mais il vous sera demandé un minimum de compétences. Cet article est donc mené à être modifier et améliorer.

*N'hésitez pas à me laisser vos commentaires
pour améliorer cet article*

Dans cet article, nous verrons comment

- Installer et configurer votre Raspberry Zero
- préparer votre script python
- installer l'écran OLED LCD
- installer du capteur SDS011

- Installation du capteur MH-Z19B
- () () () installer un serveur lighttpd et tinylora
- préparer LoRaWAN et créer votre application TTN
- envoyez les données
- sauvez les mesures dans un fichier JSON
- Installer et configurer un GPS (PG-735 (<https://cdn.sparkfun.com/datasheets/GPS/GP-735T-150203.pdf>))
- Installer une carte pour gérer la charge de la batterie avec un panneau solaire
- Economiser la consommation du Raspberry en désactivant le bluetooth, le HDMI et les LEDs de la carte

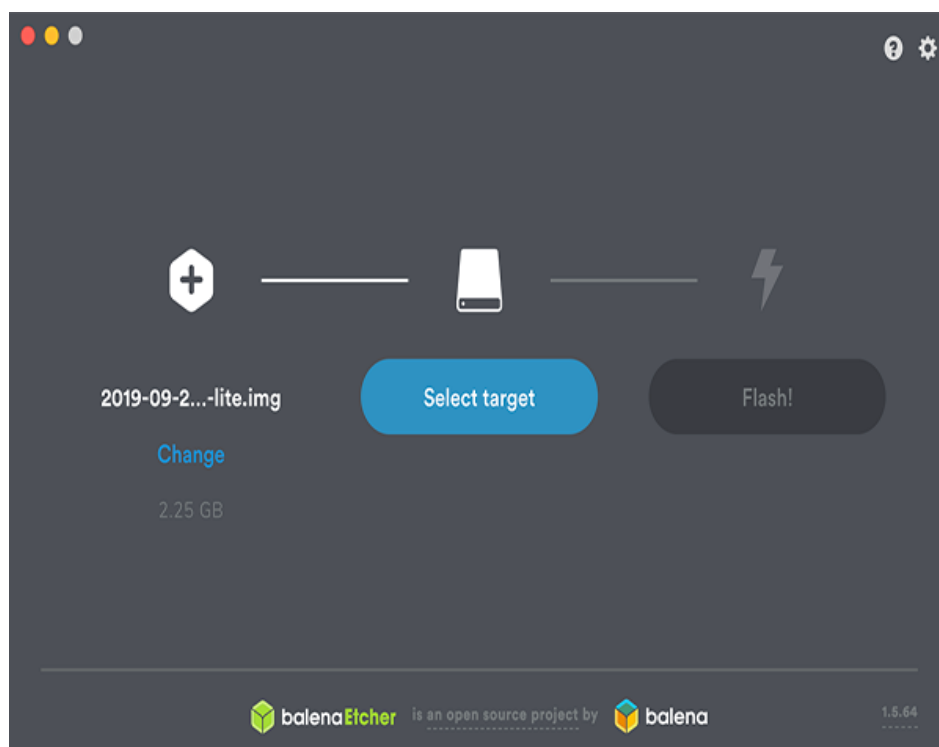
Matériel

- Raspberry Zero W (<https://www.raspberrypi.org/products/raspberry-pi-zero-w/>) (CHF 24.-)
- RFM 95 LoRaWAN/TTN hat with OLED LCD (<https://www.tindie.com/products/electronictrik/lorawanttn-kit-for-the-raspberry-pi/>) (\$40.-)
- Nova SDS011 (https://www.conrad.ch/fr/p/module-capteur-joy-it-feunstaubsensor-uart-sen-sds011-1-pc-s-1884873.html?gclid=Cj0KCQjw4dr0BRCxARIsAKUNjWT3VCLIRbORVSSGzu8LIshopping-fr&utm_medium=search&utm_campaign=shopping-online-fr&utm_content=shopping-ad_cpc&WT.srch=1&ef_id=Cj0KCQjw4dr0BRCxARIsAKUNjWT3') (CHF 38.-)
- MH-Z19D (<https://www.winsen-sensor.com/d/files/PDF/Infrared%20Gas%20Sensor/NDIR%20CO2%20SENSOR/MH-Z19%20CO2%20Ver1.0.pdf>)
- Un adaptateur USB -> MicroUSB
- GPS GP-735 (<https://www.digikey.com/catalog/en/partgroup/gps-receiver-gp-735-56-channel/66012>)
- Solar Power management (<https://www.waveshare.com/solar-power-manager.htm>)
- Panneau solaire 6V/5W (<https://www.waveshare.com/solar->

Préparation du Raspberry

Commencer par télécharger la dernière version de Rasbian Buster Desktop que vous pouvez télécharger ici (<https://www.raspberrypi.org/downloads/raspbian/>)

Téléchargez et installez la dernière version d'Etcher (<https://www.balena.io/etcher/>) et créez l'image sur la carte SD



Etcher (2019-09-2...-lite.img est représentatif, vous devriez voir quelque chose comme 2020-02-1...lite.zip)

Une fois fait, retirez et réinsérez la carte SD dans votre Mac, et créez un fichier 'ssh' dans le dossier boot, pour activer ssh lors du premier démarrage du Raspberry (L'exemple est donné depuis un Mac)

```
1 touch /Volumes/boot/ssh
```

Ceci vous permettra d'accéder à votre Raspberry en ligne de commande, avant d'avoir configuré votre Raspberry.

Ceci est particulièrement utile si vous êtes à l'aise avec les lignes

 French



de commande et si vous ne pouvez/voulez pas connecter un
écran à votre Raspberry.

```
1 ssh pi@raspberrypi.local
```

```
2 #Le mot de passe de 'pi' par défaut est 'raspl
```

Éjectez votre carte SD et insérez la dans le Raspberry.
Puis connectez votre Raspberry au routeur à l'aide du câble Ethernet. Connectez votre souris, clavier et écran à votre Raspberry.

Si vous avez créé le dossier

1 /Volumes/boot/ssh

vous pouvez vous passer de votre écran.

Mise à jour

Mettez à jour votre Raspberry et installez des packages utiles

```
1 sudo apt update && sudo apt upgrade
2 sudo apt install vim ntpdate git
```

Changez le mot de passe

```
1 sudo passwd pi
```

Créez un nouvel utilisateur avec les privilèges sudo.

Puis quittez et authentifiez-vous avec le nouvel utilisateur

```
1 sudo adduser pierrot
2 sudo adduser pierrot sudo
3 exit
4 ssh pierrot@sds011.local
```

Supprimez l'utilisateur pi (optionnel)

```
1 sudo userdel -rf pi
```

Changez les paramètres de votre Raspberry

```
1 sudo raspi-config
```

Choisissez **2 Network Option**, puis

N1 Host name => Donnez/modifiez un nom à votre Raspberry
(j'ai donné comme nom **sds011**)

Choisissez **4 Localisation Option**, puis

I1 Change Local => adapter selon votre pays

I2 Change Timezone => Idem

I4 Change Wi-fi Country => Choisissez votre pays

 (/)   (/)

Choisissez **5 Interfacing Options**, et activez

P2 SSH

P3 VNC

P4 SPI

P5 I2C

Configurez le WiFi de votre domicile

Même si je décris comment configurer votre WiFi ici, il est préférable de la faire à la fin et de continuer l'installation avec le câble Ethernet. A vous de voir.

```
1 sudo nano /etc/wpa_supplicant/wpa_supplicant.conf
```

et ajoutez les lignes suivantes (attention aux tabatures)

```
1 network={
2   ssid="The_SSID_of_your_wifi"
3   psk="Your_wifi_password"
4 }
```


Python3

Installez ces packages

```
1 sudo apt install python3-smbus
2 sudo pip3 install psutil
```

Créez un fichier python que nous allons remplir au fur et à mesure

```
1 sudo mkdir -p /opt/sds011/  
2 sudo nano /opt/sds011/aqi-v1.py # Ne pas le n
```

et ajoutez les ligne suivantes au début du fichier

```
1 #!/usr/bin/env python3
2 import time, json
3 from datetime import datetime
4 import psutil
```

Display LCD

Sourec:

- <https://learn.adafruit.com/monochrome-oled-breakouts/python-setup> (<https://learn.adafruit.com/monochrome-oled-breakouts/python-setup>)
- https://github.com/adafruit/Adafruit_CircuitPython_SSD1306/blob/master/adafruit_ssd1306.py (https://github.com/adafruit/Adafruit_CircuitPython_SSD1306/blob/master/adafruit_ssd1306.py)

Installation des librairies SSD1306 et Pillow

```
1 sudo apt install python3-pip
2 sudo pip3 install adafruit-circuitpython-ssd1:
3 sudo apt install python3-pil
```

Editez le fichier aqi.py

```
1 sudo nano /opt/sds011/aqi-v1.py
```

ajoutez les lignes suivantes

```
1 # Import library for SSD1306
2 import adafruit_ssd1306, board, busio
3 # Create the I2C interface.
4 i2c = busio.I2C(board.SCL, board.SDA)
5 # 128x64 OLED Display
6 display = adafruit_ssd1306.SSD1306_I2C(128, 64, i2c)
7 # Clear the display
8 display.fill(0)
9 display.show()
10 # Get the LCD size
11 width = display.width
12 height = display.height
```

Il vous faudra vous assurer d'avoir la font *font5x8.bin* au même niveau que votre fichier *aqi.py*, si non utilisez la commande *wget* pour le télécharger

```
1 cd /opt/sds011/  
2 ls  
3 # ou  
4 sudo wget https://github.com/pierrot10/Raspi-
```


Exemple de code

```
1 # Efface l'écran
2 display.fill(0)
3 # Affiche
4 display.show()
5 # Met du texte dans le buffer à position x=0
6 display.text('ECO-SENSORS.CH', 0, 0, 1)
7 # Met encore du texte dans le buffer à la pos:
8 display.text('Smart Air Quality', 0, 8, 1)
9 # Affiche le texte contenu dans le buffer
10 display.show()
11 # On/Off l'écran
12 display.poweron()
13 display.poweroff()
```

SDS011

Source : <https://towardsdatascience.com/sensing-the-air-quality-5ed5320f7a56> (<https://towardsdatascience.com/sensing-the-air-quality-5ed5320f7a56>)

Les mesures seront sauvegardées dans un fichier JSON. Nous allons aussi installé un petit serveur web, pour que vous puissiez lire les données depuis un navigateur

Installation de lighttpd

```
1 sudo apt install lighttpd python-enum
```

Nous allons changer le propriétaire du dossier web

```
1 sudo chown -R pierrot:pierrot /var/www/html/  
2 # ou sudo chown -R pi:pi /var/www/html/
```

Préparation du SDS011

Nous allons maintenant importer la classe qui vous permettra de faire tourner votre capteur SDS011 et mesurer les particules

fines



(/)



(/)

```
1 cd /opt/sds011/
```

```
2 sudo wget https://github.com/pierrot10/Raspi-
```

Vous devrez encore installer cette librairie

```
1 sudo pip3 install python-aqi
```

Connectez votre capteur SDS011 au port USB.

Nous devons maintenant savoir à quel port USB, le capteur est connecter. Lancer la commande suivante

```
1 dmesg | grep USB
```

et vous devriez voir une ligne finissant par ttyUSB0, comme cela est mon cas

```
ch341-uart converter now attached to ttyUSB0
```

Une fois fait, éditez votre fichier aqi-v1.py

```
1 sudo nano /opt/sds011/aqi-v1.py
```


et ajoutez les lignes suivantes

```
1 # SDS011
2 from sds011 import SDS011
3 import aqi
4 sensor = SDS011("/dev/ttyUSB0", use_query_mode=1)
5 # JSON
6 JSON_FILE = '/var/www/html/aqi.json'
```

Assurez-vous que **ttyUSB0** correspond bien à ce que vous avez trouvé avec la commande 'dmesg | grep USB'

Créez encore un fichier json, où seront sauvez plus tard, les mesures

```
1 echo [] > /var/www/html/aqi.json
```

Exemple de code

```
1 # Recevoir les mesures
2 pmt_2_5, pmt_10 = get_data()
3 # Affichage
4 print(time.strftime("%Y-%m-%d (%H:%M:%S)"), ei
5 print(f"    PM2.5: {pmt_2_5} µg/m3    ", end=
6 print(f"PM10: {pmt_10} µg/m3")
7 print(' ')
```

JSON

Pour le sauvegarder dans votre fichier JSON vous pouvez le faire ainsi

```
1 import json # A ajouter en haut du fichier
2
3 # get date and time
4 tnow = datetime.now()
5 timestamp_now = datetime.timestamp(tnow)
```

```

6
7 # open stored data
8 try:
9     with open(JSON_FILE) as json_data:
10         data = json.load(json_data)
11 except IOError as e:
12     data = []
13     print('except')
14
15 # check if length is more than 100 and delete
16 if len(data) > 100:
17     data.pop(0)
18
19 # append new values
20 jsonrow = {'pm25': pmt_2_5, 'pm10': pmt_10, '
21 data.append(jsonrow)
22
23 # save it
24 with open(JSON_FILE, 'w') as outfile:
25     json.dump(data, outfile)

```

MH-Z19B

sensor/mh-z19b-co2-ver1_0.pdf (https://www.winsensor.com/d/files/infrared-gas-sensor/mh-z19b-co2-ver1_0.pdf)

Librairie : <https://pypi.org/project/mh-z19/> (<https://pypi.org/project/mh-z19/>)

Github (forked): <https://github.com/ecosensors/mh-z19> (<https://github.com/ecosensors/mh-z19>)

Port Série (UART): <https://www.framboise314.fr/utiliser-le-port-serie-du-raspberry-pi-3-et-du-pi-zero/> (<https://www.framboise314.fr/utiliser-le-port-serie-du-raspberry-pi-3-et-du-pi-zero/>)

Préparation

Le capteur utilise les broche Tx et Rx (GPIO 14 et 15). Vous devez donc activer le port UART et désactiver la console.

Activation du Port Serial (UART)

Pour cela, ouvrez votre terminal et allez dans raspi-config

```
1 sudo raspi-config
```

Choisissez **5 Interfacing Options**, et activez :
P6 Serial

A la première question **Would you like a login shell to be accessible over Serial?**, répondez **NO**

A la deuxième question **Would you like the Serial port hardware to be enable?** répondez **YES**

Désactivation de la console

La console utilise le port Serial, vous devez donc la désactiver.

Editez le fichier suivant

```
1 sudo nano /boot/cmdline.txt
```

trouvez le texte suivant et supprimez-le

```
console=serial0, 115200
```

puis, redémarrer votre Raspberry

Assemblage



Connectez le capteur au raspberry comme ceci:

PI => MH-Z19B

5V => vin

GND => GND

Tx => Rx

Rx => Tx

Installation de la librairie MH-Z10B

Lancez la commande

```
1 sudo pip3 install mh-z19
```

Vous pouvez déjà contrôler si va fonctionne

```
1 sudo python3 -m mh_z19
```

Ce qui m'affiche

```
{"co2": 697}
```

Exemple de codes

Nous avons précédemment ajouté régulièrement des lignes dans le fichier aqi-v1.py. C'est ce que nous allons faire maintenant

```
1 sudo nano /opt/sds011/aqi-v1.py
```

et ajoutez ceci. N'oubliez pas d'ajouter le import

```
1 import mh_z19 # Vous devez ajoutez ceci au del
2
3 print('[INFO: Getting CO2]')
4 #print(mh_z19.read_all())
5 co_2 = mh_z19.read()
6 print("co2: " + str(co_2['co2']))
```

LoRaWAN

Source : <https://learn.adafruit.com/lora-and-lorawan-radio-for-raspberry-pi/sending-data-over-lorawan> (<https://learn.adafruit.com/lora-and-lorawan-radio-for-raspberry-pi/sending-data-over-lorawan>)

Rendez-vous dans votre console (<https://console.thethingsnetwork.org/>) et créez une application que je vais nommée aqi-sds011

The screenshot shows the 'ADD APPLICATION' form in the TTN console. It has four sections: 'Application ID' with the value 'aqi-sds011', 'Description' with the value 'Measuring Air Quality', 'Application EUI' with a placeholder 'EUI issued by The Things Network', and 'Handler registration' with the value 'ttn-handler-eu'. Each field has a green checkmark indicating it is valid.

Vous allez devoir maintenant créer un Device. Retourner sur vos applications et cliquez sur l'application que vous venez de créer et cliquez sur l'onglet **Devices**

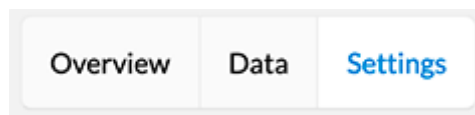
The screenshot shows the 'APPLICATIONS' list in the TTN console. It contains one application: 'aqi-sds011' with the description 'Measuring Air Quality MP25 MP10' and the handler 'ttn-handler-eu'.

et finalement sur le lien **register device**

The screenshot shows the 'REGISTER DEVICE' form in the TTN console. It has four sections: 'Device ID' with the value 'firstdevice', 'Device EUI' with a placeholder 'this field will be generated', 'App Key' with a placeholder 'this field will be generated', and 'App EUI' with the value '70 B3 D5 7E D0 02 DC DB'. Each field has a green checkmark indicating it is valid.

Sous le champs **Device EUI**, assurez-vous qu'il y a bien le texte *this field will be generated*, si non, donnez un identifiant unique, ou cliquez sur les deux flèches qui se croisent, à côté du champs. Cliquez sur **Register**.

Pour terminer, vous devez encore modifier les paramètres de ce Device. Vous verrez, en haut à droite l'onglet **Settings**.



Cliquez dessus, et modifiez les paramètres suivants

Activation Method

☐ OTAA ☒ ABP

Device Address

The device address will be assigned by the network server

Network Session Key

Network Session Key will be generated

App Session Key

App Session Key will be generated

Frame Counter Width

☒ 16 bit ☐ 32 bit

☐ Frame Counter Checks

Disabling frame counter checks drastically reduces security and should only be used for development purposes

Activation Method => ABP

Frame Counter Width => 16 bit

Frame Counter Check => Délectionnez-le

et cliquez sur **Save**

Le méthode OTAA est souvent recommandée et vous êtes libre de faire comme vous le souhaitez.




Ultérieurement vous allez avoir besoin des clés **App Session Key**, **Network Session Key** et **Devise address** que vous pourriez déjà relever.

TinyLora

Pour envoyer les données avec LoRaWAN aux serveur TTN (The Things Network (<https://www.thethingsnetwork.org/>)), nous allons



devoir installer une librairie d'Adafruit. Retourner sur votre

 (/)   (/)
Raspebrry

```
1 ssh pierrot@sds011.local # remplacer ssds011 |
```


et installez la librairie tinylora

```
1 sudo pip3 install adafruit-circuitpython-tiny
```

Editez encore le fichier aqi.py

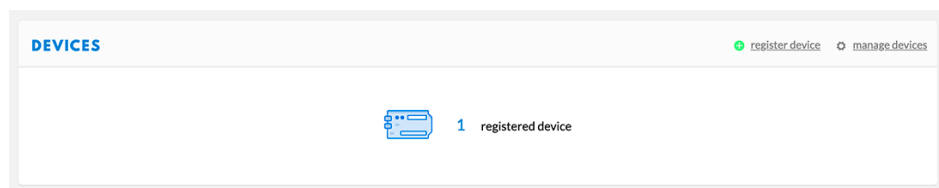
```
1 cd /opt/sds011/  
2 sudo nano aqi-v1.py
```

et ajoutez les lignes suivantes en haut du fichier (je les ai mise en dessous de *import busio*




```
1 from digitalio import DigitalInOut, Direction
2 from adafruit_tinylora.adafruit_tinylora import
3 # importer encore board, si vous n'utilisez pas
4 # import board
5
6 # TinyLoRa Configuration
7 ## Attention, cette configuration dépendra du
8 ## bien pour le module d'adafruit RFM95W LoRa
9 spi = busio.SPI(board.SCK, MOSI=board.MOSI, MISO=board.MISO)
10 cs = DigitalInOut(board.CE1)
11 irq = DigitalInOut(board.D5)
12 rst = DigitalInOut(board.D25)
13
14 # TTN Device Address, 4 Bytes, MSB
15 devaddr = bytearray([0x00, 0x00, 0x00, 0x00])
16 # TTN Network Key, 16 Bytes, MSB
17 nwkey = bytearray([0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00])
18 # TTN Application Key, 16 Bytes, MSB
19 appkey = bytearray([0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00])
20
21 # Initialize ThingsNetwork configuration
22 ttn_config = TTN(devaddr, nwkey, appkey, country="FR")
23 lora = TinyLoRa(spi, cs, irq, rst, ttn_config)
24 # 2b array to store sensor data
25 data_pkt = bytearray(2)
26
```

Vous allez devoir informer les clés (key) que vous trouverez dans votre console TTN (allez dans Application) (<https://console.thethingsnetwork.org/applications>), et éditez l'application que vous avez créé plus tôt.

Puis cliquez sur le **Device** et puis sur le « device » que vous avez aussi déjà créé



et relever les clés Device Address (**devaddr**), Network Session Key (**nwkey**) et App Session Key (**app**)

Device Address	<>	=	msb	{ 0x26, 0x21, 0x20, 0x1F }	
Network Session Key	<>	=	msb	{ 0x9C, 0x9B, 0x9A, 0x99, 0x98, 0x97, 0x96, 0x95, 0x94, 0x93, 0x92, 0x91, 0x90, 0x8F, 0x8E, 0x8D, 0x8C, 0x8B, 0x8A, 0x89, 0x88, 0x87, 0x86, 0x85, 0x84, 0x83, 0x82, 0x81, 0x80 }	
App Session Key	<>	=	msb	{ 0x81, 0x80, 0x7F, 0x7E, 0x7D, 0x7C, 0x7B, 0x7A, 0x79, 0x78, 0x77, 0x76, 0x75, 0x74, 0x73, 0x72, 0x71, 0x70, 0x6F, 0x6E, 0x6D, 0x6C, 0x6B, 0x6A, 0x69, 0x68, 0x67, 0x66, 0x65, 0x64, 0x63, 0x62, 0x61, 0x60, 0x5F, 0x5E, 0x5D, 0x5C, 0x5B, 0x5A, 0x59, 0x58, 0x57, 0x56, 0x55, 0x54, 0x53, 0x52, 0x51, 0x50, 0x4F, 0x4E, 0x4D, 0x4C, 0x4B, 0x4A, 0x49, 0x48, 0x47, 0x46, 0x45, 0x44, 0x43, 0x42, 0x41, 0x40, 0x3F, 0x3E, 0x3D, 0x3C, 0x3B, 0x3A, 0x39, 0x38, 0x37, 0x36, 0x35, 0x34, 0x33, 0x32, 0x31, 0x30, 0x2F, 0x2E, 0x2D, 0x2C, 0x2B, 0x2A, 0x29, 0x28, 0x27, 0x26, 0x25, 0x24, 0x23, 0x22, 0x21, 0x20, 0x1F, 0x1E, 0x1D, 0x1C, 0x1B, 0x1A, 0x19, 0x18, 0x17, 0x16, 0x15, 0x14, 0x13, 0x12, 0x11, 0x10, 0x0F, 0x0E, 0x0D, 0x0C, 0x0B, 0x0A, 0x09, 0x08, 0x07, 0x06, 0x05, 0x04, 0x03, 0x02, 0x01, 0x00 }	

Note:

- Cliquez sur <> pour avoir le format {0x9c, 0x91..., 0xBC}
- Cliquez sur les deux flèches inversées pour avoir le format msb
- Cliquez sur l'icône de droite, pour copier la clé dans votre presse-papier

1 # TTN Device Address, 4 Bytes, MSB
2 devaddr = bytearray([0x00, 0x00, 0x00, 0x00])
3 # TTN Network Key, 16 Bytes, MSB
4 nwkey = bytearray([0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00])
5 # TTN Application Key, 16 Bytes, MSB
6 app = bytearray([0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00])

Attention, remplace les {} par []

PycURL à la place de LoRaWAN

Dans certains cas il peut être nécessaire d'envoyer des données via un routeur WiFi à la place de passer par le protocole et une passerelle LoRaWAN. Pour une station, cette situation s'est présentée car il n'y avait pas de passerelle LoRaWAN dans le périmètre et il me fallait prendre des mesures dans cette location. Il était aussi trop onéreux de mettre en place une passerelle, que je n'avais pas en stock d'ailleurs. La solution PycURL s'est montré très utile puisqu'un routeur WiFi était à disposition.

Pour commencer, installez la librairie

Info : <https://stackabuse.com/using-curl-in-python-with-pycurl/>
(<https://stackabuse.com/using-curl-in-python-with-pycurl/>)

```
1 sudo pip3 install pycurl
```

GPS

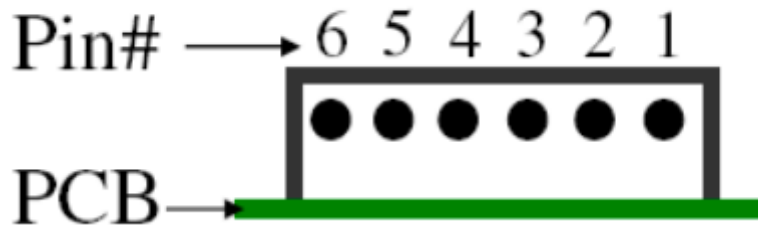
Source:

1. <https://ozzmaker.com/berrygps-setup-guide-raspberry-pi/>
(<https://ozzmaker.com/berrygps-setup-guide-raspberry-pi/>)
2. <https://ozzmaker.com/using-python-with-a-gps-receiver-on-a-raspberry-pi/> (<https://ozzmaker.com/using-python-with-a-gps-receiver-on-a-raspberry-pi/>)
3. <https://fishandwhistle.net/post/2016/using-pyserial-pynmea2-and-raspberry-pi-to-log-nmea-output/> (<https://fishandwhistle.net/post/2016/using-pyserial-pynmea2-and-raspberry-pi-to-log-nmea-output/>)

On va maintenant connecter le GPS GP-735. (<https://fr.pyboard.org/2016/07/01/using-gps-on-raspberry-pi/>)



Observez bien les broches du GPS dans la datasheet (<https://cdn.sparkfun.com/datasheets/GPS/GP-735T-150203.pdf>), page 8 et 9



6-pin Interface, pitch 1.0mm, UART TTL

Pin	Name	Function	I/O
1	GND	Ground	Input
2	VCC	3.1 ~ 5.5 V power supply	Input

4

ADH-tech

Data Sheet - GP-735

3	TXA	Port A serial data output (from GPS)	Output
4	RXA	Port A serial data input (to GPS)	Input
5	V_BAT	Backup power (1.5 ~ 3.5V)	Input
6	PWR_CTRL	Module power control High or disconnected: power ON Low: power OFF	Input

gp-735 pinout

Connexion

- la broche 1 sur un GND du Raspberry
- la broche 2 sur le 5V du Raspberry
- la broche 3 (TXa) sur la broche Rx du Raspberry
- la broche 4 (RXb) sur la broche Tx du Raspberry
- la broche 6 sur la broche 18 du Raspberry

Il va falloir encore modifier la configuration de votre Raspbbery

1 sudo raspi-config
🗑️ (/) 🔍 👤 (/)

Sélectionnez **5. Interfacing Options**, puis **P6 Serial**, répondez **NON** à la prochaine question, puis **OUI** à la dernière question.

Vous allez devoir redémarrer votre Raspberry.

Vous pouvez déjà voir si ça fonctionne en lisant les données du GPS, de cette manière

- 1 `sudo cat /dev/ttyS0` # Pour un Raspbbery 3
- 2 `sudo cat /dev/ttyAMA0` # Pour les autres

Parser les données de GPS

Installons maintenant la librairie pynmea2

```
1 sudo pip3 install pynmea2
```

Ouvrez votre fichier

```
1 sudo nano /opt/sds011/aqi-v1.py
```

et ajoutez en haut du fichier


```
1 import pynmea2, serial
```

Autres liens utiles

- <https://www.sparkfun.com/tutorials/403> (<https://www.sparkfun.com/tutorials/403>)
- <https://github.com/Knio/pynmea2/blob/master/NMEA0183.pdf> (<https://github.com/Knio/pynmea2/blob/master/NMEA0183.pdf>)
- <https://circuitpython.readthedocs.io/en/1.0.0/docs/esp8266/tutorial/pins.html> (<https://circuitpython.readthedocs.io/en/1.0.0/docs/esp8266/tutorial/pins.html>)

zero2go

Économiser la consommation

Source : <https://learn.pi-supply.com/make/how-to-save-power-on-your-raspberry-pi/> (<https://learn.pi-supply.com/make/how-to->  French)

Désactivation du bluetooth

Afficher le status

```
1 systemctl status bluetooth
```

Désactivation

```
1 echo " " | sudo tee -a /boot/config.txt
2 echo "# Diable Bluetooth" | sudo tee -a /boot,
3 echo "dtoverlay=pi3-disable-bt" | sudo tee -a
4 sudo systemctl disable hciuart
5 sudo reboot
```


Après avoir désactivé le Bluetooth, mon Raspberry n'était plus capable de lire sur le port ttyS0. Le GPS communiquait via le port ttyAMA0. J'ai du adapter mon code en conséquence

Désactivation des LED

```
1 echo " " | sudo tee -a /boot/config.txt
2 echo "# Diable On-board LEDs" | sudo tee -a /l
3 echo "dtparam=act_led_trigger=none" | sudo tee
4 echo "dtparam=act_led_activelow=on" | sudo tee
5 sudo reboot
```

Désactivation du port HDMI

Le port HDMI ne sera pas utiliser, donc pas d'écran connecté. Ce qui permettrait de sauver jusqu'à 30mA

Désactivation




```
1 sudo /opt/vc/bin/tvservice -o
```

Activation

```
1 sudo /opt/vc/bin/tvservice -p
```

DYNU

ddclient

<https://www.dynu.com/DynamicDNS/IPUpdateClient/DDClient>
 [\(1\)](#)   [\(1\)](#)
(<https://www.dynu.com/DynamicDNS/IPUpdateClient/DDClient>)

Vous devez d'abord créer un host chez dyndns

```
1 sudo apt install ddclient
```

Vous allez devoir répondre à plusieurs questions

Fournisseur de service DNS dynamique: autre

Serveur de DNS dynamique: api.dynu.com

Protocole de mise à jour du DNS dynamique: dyndns2

Identifiant pour les service... : Votre identifiant

Mots de passe pour ...: Votre mots de passe

Interface utilisée par ...? : wwan0

Nom de domaine de DNS ..: *host.ddnsfree.com*

```
1 sudo ddclient -daemon=0 -debug -verbose -noqu:
```

```
sudo nano /etc/ddclient.conf
```

🔖 AIR QUALITY ([HTTPS://ECO-SENSORS.CH/TAG/AIR-QUALITY/](https://eco-sensors.ch/tag/air-quality/))
AQI ([HTTPS://ECO-SENSORS.CH/TAG/AQI/](https://eco-sensors.ch/tag/aqi/)) BUSIO ([HTTPS://ECO-SENSORS.CH/TAG/BUSIO/](https://eco-sensors.ch/tag/busio/)) GP-735 ([HTTPS://ECO-SENSORS.CH/TAG/GP-735/](https://eco-sensors.ch/tag/gp-735/)) GPIO ([HTTPS://ECO-SENSORS.CH/TAG/GPIO/](https://eco-sensors.ch/tag/gpio/)) GPS ([HTTPS://ECO-SENSORS.CH/TAG/GPS/](https://eco-sensors.ch/tag/gps/)) JSON ([HTTPS://ECO-SENSORS.CH/TAG/JSON/](https://eco-sensors.ch/tag/json/)) LIGHTTPD ([HTTPS://ECO-SENSORS.CH/TAG/LIGHTTPD/](https://eco-sensors.ch/tag/lighttpd/)) LORAWAN ([HTTPS://ECO-SENSORS.CH/TAG/LORAWAN/](https://eco-sensors.ch/tag/lorawan/)) MH-Z19B ([HTTPS://ECO-SENSORS.CH/TAG/MH-Z19B/](https://eco-sensors.ch/tag/mh-z19b/)) NOVA ([HTTPS://ECO-SENSORS.CH/TAG/NOVA/](https://eco-sensors.ch/tag/nova/)) PILLOW ([HTTPS://ECO-SENSORS.CH/TAG/PILLOW/](https://eco-sensors.ch/tag/pillow/)) PM10 ([HTTPS://ECO-SENSORS.CH/TAG/PM10/](https://eco-sensors.ch/tag/pm10/)) PM2.5 ([HTTPS://ECO-SENSORS.CH/TAG/PM2-5/](https://eco-sensors.ch/tag/pm2-5/)) PYTHON3 ([HTTPS://ECO-SENSORS.CH/TAG/PYTHON3/](https://eco-sensors.ch/tag/python3/)) RASPBERRY ([HTTPS://ECO-SENSORS.CH/TAG/RASPBERRY/](https://eco-sensors.ch/tag/raspberry/)) SDS011 ([HTTPS://ECO-SENSORS.CH/TAG/SDS011/](https://eco-sensors.ch/tag/sds011/)) SSD1306 ([HTTPS://ECO-SENSORS.CH/TAG/SSD1306/](https://eco-sensors.ch/tag/ssd1306/)) TINYLORA ([HTTPS://ECO-SENSORS.CH/TAG/TINYLORA/](https://eco-sensors.ch/tag/tinylora/)) TTN ([HTTPS://ECO-SENSORS.CH/TAG/TTN/](https://eco-sensors.ch/tag/ttn/))

PREV POST

NEXT POST

(<https://eco-sensors.ch/mesure-de-la-qualite-de-lair-pm2-5-pm10/>) (<https://eco-sensors.ch/passerelle-lorawan-avec-deux-raspberry-pi3/>)

Leave **Comment:**

FULL NAME

EMAIL ADDRESS

PHONE NUMBER

YOUR COMMENT

SUBMIT NOW

Nous **soutenir**

STM32 LORA DISCOVERY KIT

(<https://eco-sensors.ch/product/stm32-lora-discovery-kit/>)

CHF30.00

ST-LINK

(<https://eco-sensors.ch/product/st-link/>) CHF59.00



Newsletter

Inscrivez-vous à notre newsletter

Prénom

Nom de famille

Email

Votre secteur

Je souhaite juste vous suivre



Votre fonction

 French



 Je souhaite juste vous suivre 


Votre exploitation/association (facultatif)

Indiquez-nous pour qui vous euvrez

Vos cultures, seront-elle sensibles aux dommages causés par le gel (facultatif)

non 

Sponsor/donateur potentiel

peut-être 

Commentaire (facultatif)

Que pouvons-nous vous apporter?

S'ABONNER

Suivez-nous sur



 info@eco-sensors.ch (mailto:info@eco-sensors.ch)

© Copyright 2020 EcoSensors. - Tous droits réservés.