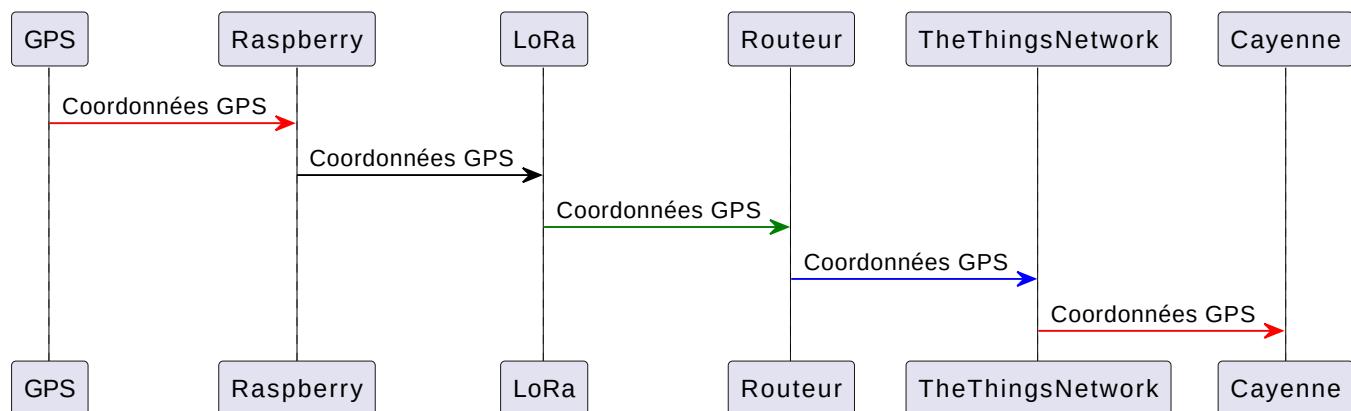


# Mise en place d'un traqueur GPS via LoRa(Wan)



## Installation du routeur sur Internet (via WiFi)

N.B. : Pourquoi via WiFi ? Dans le cas particulier de l'Université de Perpignan Via Domitia, le FireWall "n'aime" pas les connections sur le port 1700 nécessaire à l'établissement de la connection routeur -> TheThingsNetwork.

- On le branche sur secteur via USB-C 5V-2A un réseau WiFi dragino-XXXXXX apparaît.
- On se connecte à ce réseau via le mot de passe par défaut du routeur : "dragino+dragino" (sans les guillemets)
- Sur le navigateur on va sur l'IP 10.130.1.1 un couple Id/MdP est demandé par le dragino (par défaut) "root" / "dragino"
- Pour le WiFi Client (Le dragino fournit pour l'instant un serveur web mais n'est pas connecté à Internet !) On clique sur Enable WiFi-Wan Client avec le couple SSID/MdP d'un smartphone par exemple.

The screenshot shows the WiFi configuration page of the Dragino LoRa Gateway. The URL is 10.130.1.1/cgi-bin/system-wifi.has. The sidebar includes icons for File, Network, System, LogRead, Home, and Logout. The main menu has tabs for WiFi, Radio Settings, WiFi Access Point Settings, WiFi WAN Client Settings, and WiFi status. The WiFi tab is selected. In the WiFi Access Point Settings section, 'Enable WiFi Access Point' is checked, and the WiFi Name SSID is set to 'dragino-1fab40'. In the WiFi WAN Client Settings section, 'Enable WiFi WAN Client' is checked, and the Host WiFi SSID is set to 'OnePlus5'. The WiFi status message at the bottom says 'WiFi status: OK. Click Refresh to check status.'

## Routage des paquets LoRa vers TheThingsNetwork

- Créer un compte TheThingsNetwork.org (gratuit il faut fournir son courriel)
- On peut voir l'identifiant par défaut du routeur (le Gateway EUI) sur l'onglet LoRa du serveur web du routeur (10.130.1.1)
- De plus il faut choisir TheThingsNetwork v3 sur le menu déroulant en dessous (le v2 est disponible sur le routeur mais TheThingsNetwork qui maintient sa version 2 pour les anciens routeurs ne permet pas la création de nouveaux routeurs en v2)
- Il faut aussi choisir eu1.cloud.thethings.network sur le deuxième menu déroulant confère images ci dessous.

The screenshot shows the configuration interface for a Dragino LoRa Gateway. The main section is titled "LoRaWAN Configuration". Under "General Settings", the "Email" field is set to "dragino-1fab40@dragino.com" and the "Gateway ID" field is set to "AB40411FAB404151". In the "Primary LoRaWAN Server" section, the "Service Provider" is set to "The Things Network V3" and the "Server Address" is set to "eu1.cloud.thethings.network". The "Uplink Port" and "Downlink Port" are both set to 1700. Below this, there is a "Packet Filter" section with fields for "Fport Filter" (set to 0) and "DevAddr Filter" (set to 0). At the bottom of the configuration page, it says "Current Mode: LoRaWAN Semtech UDP" with buttons for "Save&Apply" and "Cancel".

L'identifiant du routeur (Gateway EUI) doit être le même que dans la configuration précédente. Le GatewayID est libre mais doit être unique sur ttn donc disponible. Le Gateway Name est quant à lui totalement libre. Enfin les Gateway Server Address doit correspondre au précédent soit pour l'Europe : eu1.cloud.thethings.network

Le reste des options peut être laissé par défaut ou changé (si on sait pourquoi 😊)

Ca y est vous avez votre routeur connecté sur le LoRaWan.

The screenshot shows a Firefox browser window with several tabs open. The active tab is titled "Add gateway - Console" and has the URL <https://eu1.cloud.thethings.network/console/gateways/add>. The page content is a form for adding a new gateway. It includes fields for "Owner" (set to "bastienbaranoff"), "Gateway ID" (set to "mon-super-routeur"), "Gateway EUI" (set to "18 40 41 1F AB 40 41 51"), "Gateway name" (set to "Routeur Dragino"), "Gateway description" (with placeholder text "Description for my new gateway"), and "Gateway Server address" (set to "eu1.cloud.thethings.network"). The browser's sidebar shows various desktop icons, and the top bar displays the date and time as "9 mars 17:46".

## Préparation du RaspberryPi (l'objet connecté)

Un raspberry est un micro-ordinateur à peu près de la taille d'une CB avec la puissance d'un smartphone et des broches d'entrées-sortie électriques. Le système d'exploitation de ce matériel se trouve (en général et dans cette étude, sinon il peut être via NetBoot, USB, HDD, emmc) sur une carte SD préparée par exemple de la manière suivante :

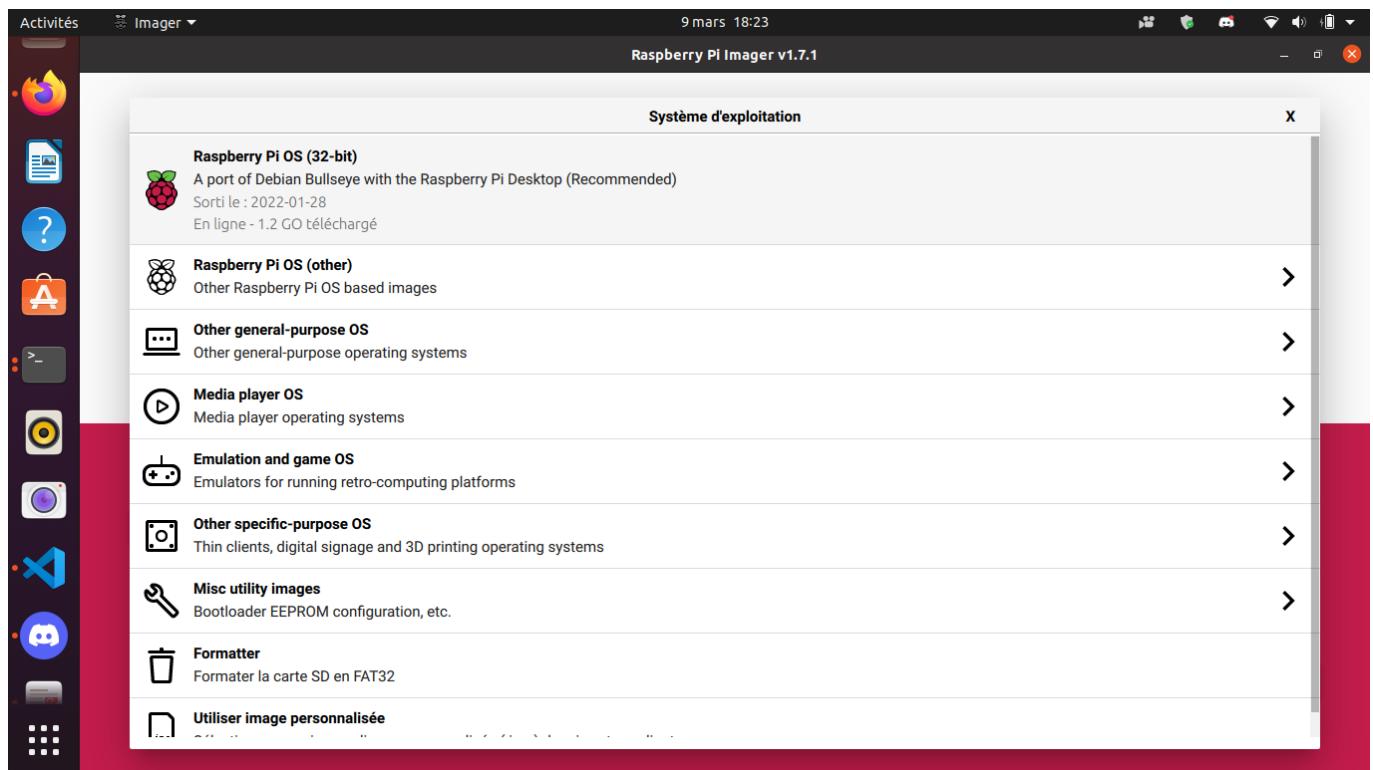
### La carte SD :

Télécharger Raspi-Imager sur le site officiel de Raspberry <https://www.raspberrypi.com/software/>

Pour installer raspi-imager sur un ordinateur hôte ubuntu récent il suffit d'ouvrir une fenêtre de commandes (Ctrl-Alt-T) et de taper

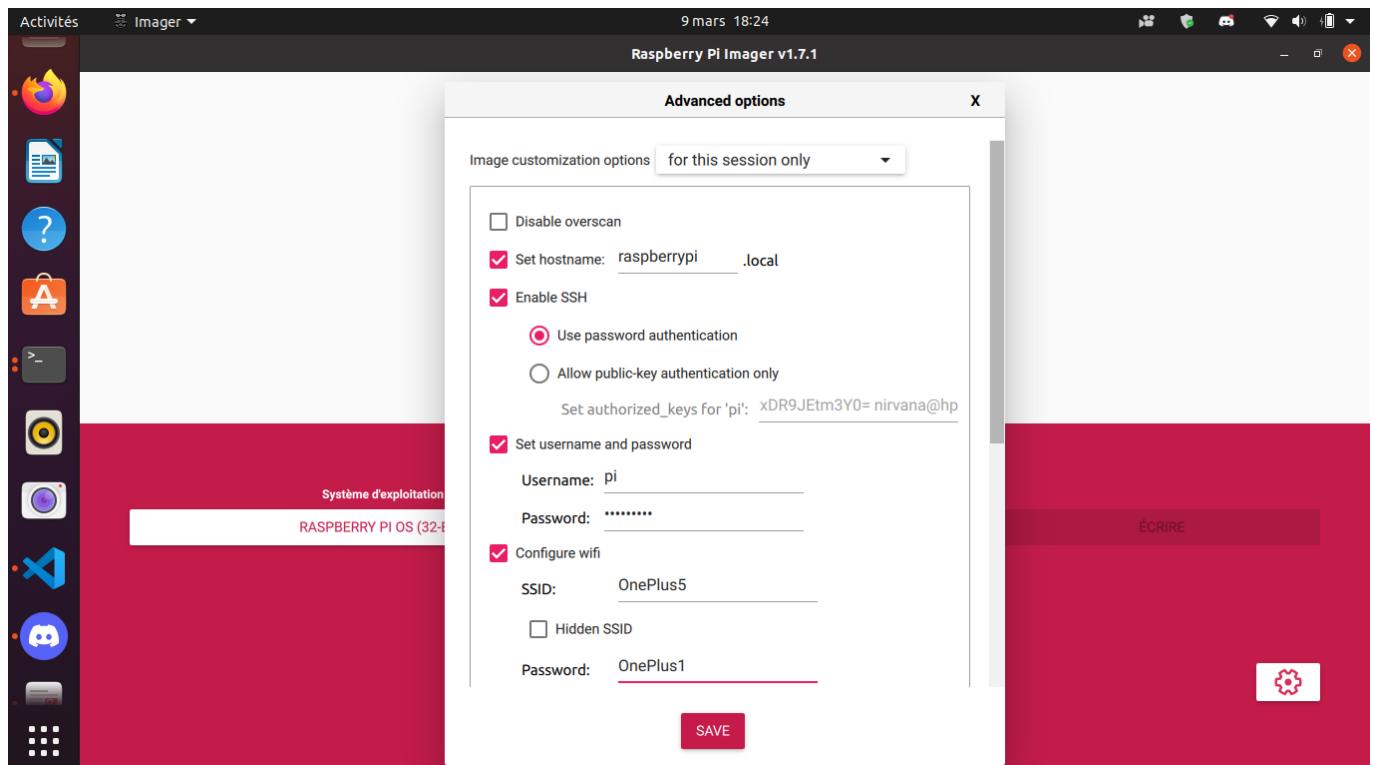
```
sudo snap install rpi-imager
```

Ensuite on sélectionne l'OS suivant (Debian Bullseye le premier de la liste et le dernier existant au moment où sont écrits ces mots)



et on sélectionne les options suivantes

ssh : username/password (conseil : "pi"/"raspberry") Wifi : le WiFi du téléphone ou n'importe lequel auquel on a accès optionnel : set hostname = raspberry.local



Ensuite on sélectionne le media sur lequel on va écrire et on choisi enfin écrire.

On met la carte une fois terminé ça y est le raspberry devrait tourner avec un OS. On peut vérifier via HDMI sur un écran. Ou si on veut on peut accéder en ssh si l'ordinateur se trouve sur le même réseau local que le raspberry. Si le réseau est en 192.168.1.0/24 il faut faire

```
nmap 192.168.1.1-254 -p 22
```

pour connaître l'adresse du rpi. Ou encore si on ne connaît pas les octets de l'ip du gateway

```
sudo arp -a
```

Enfin pour accéder à un shell sur ce même rpi

```
ssh pi@ip_du_pi_trouvée_précédemment
```

ou

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

## Installation et configuration du Hat Dragino (GPS/LoRa) sur le raspberry

Une fois sur le shell du rpi comme toujours :

```
sudo apt update && sudo apt upgrade
```

Ensuite on installe les paquets nécessaires :

```
sudo apt install git device-tree-compiler git python3-crypto python3-nmea2
python3-rpi.gpio python3-serial python3-spidev python3-configobj gpsd
libgps-dev gpsd-clients libgps23 python3-pip
pip3 install simplecayennelpp
```

Ensuite on rajoute au fichier /boot/config.txt les lignes suivantes :

```
enable_uart=1
dtoverlay=miniuart-bt
dtoverlay=spi-gpio-cs
```

On modifie le fichier /boot/cmdline.txt de façon à ce qu'il devienne

```
dwc_otg.lpm_enable=0 console=tty1 root=/dev/mmcblk0p2 rootfstype=ext4
elevator=deadline fsck.repair=yes rootwait
```

Ensuite dans le /home/pi

```
git clone https://github.com/computenodes/dragino
cd dragino/overlay
dtc -@ -I dts -O dtb -o spi-gpio-cs.dtbo spi-gpio-cs-overlay.dts
sudo cp spi-gpio-cs.dtbo /boot/overlays/
sudo reboot
```

Ensuite dans /home/pi on crée le fichier gpscron tel que :

```
#!/bin/bash
sudo python3 /home/pi/dragino/test_cayenne.py
```

dans /home/pi/dragino on écrit le fichier test\_cayenne.py tel que :

```
#!/usr/bin/env python3
"""
    Test harness for dragino module - sends hello world out over LoRaWAN 5
times
"""

import logging
from datetime import datetime
from time import sleep
import RPi.GPIO as GPIO
from dragino import Dragino
# import subprocess
import gpsd
from simplecayennelpp import CayenneLPP # import the module required to
pack th$
import binascii
# importing the module
# Connect to the local gpsd
gpsd.connect()
packet = gpsd.get_current()
# See the inline docs for GpsResponse for the available data
print(packet.position())
lat = packet.lat
lon = packet.lon
alt = packet.alt

print (lat, lon, alt)
lpp = CayenneLPP()
lpp.addGPS( 1, lat, lon, alt)
text=binascii.hexlify(lpp.getBuffer()).decode()
sent=list(binascii.unhexlify(text))
print(text)
logLevel=logging.DEBUG
```

```

logging.basicConfig(filename="test.log", format='%(asctime)s - %(funcName)s
- %(lineno)d - %(levelname)s - %(message)s', level=logLevel)
D = Dragino("/home/pi/dragino/dragino.ini", logging_level=logLevel)
D.join()
while not D.registered():
    print("Waiting for JOIN ACCEPT")
    sleep(2)
for i in range(0, 2):
    D.send_bytes(sent)
    start = datetime.utcnow()
    while D.transmitting:
        pass
    end = datetime.utcnow()
    print("Sent GPS coordinates {}".format(end-start))
    sleep(1)

```

On prend le fichier /home/pi/dragino/dragino.ini.default et on le réécrit sous /home/pi/dragino/dragino.ini de la manière suivante

```

gps_baud_rate = 9600
gps_serial_port = /dev/ttys0
gps_serial_timeout = 1
gps_wait_period = 10

#LoRaWAN configuration
spreading_factor = 7
max_power = 0x0F
output_power = 0x0E
sync_word = 0x34
rx_crc = True
#Where to store the frame count
fcount_filename = .lora_fcount

##Valid auth modes are ABP or OTAA
##All values are hex arrays eg devaddr = 0x01, 0x02, 0x03, 0x04
#auth_mode = "abp"
#devaddr =
#nwskey =
#appskey =

auth_mode = otaa
deveui = 0xFF, 0xFE, 0xFD, 0xFC, 0xFD, 0xFE, 0xFF
appeui = 0x70, 0xB3, 0xD5, 0x00, 0x00, 0xD5, 0xB3, 0x70
appkey = 0x3D, 0x83, 0xC3, 0x16, 0x2C, 0xAD, 0x44, 0xB7, 0xB0, 0x50, 0x6C,
0x3C, 0xA1, 0x54, 0x36, 0xB7

```

En choisissant les deveui, appeui de façons à ce qu'ils soient uniques sur ttn. Et l'appkey avec suffisamment d'entropie pour ne pas qu'on puisse la brute-forcer.

Enfin pour executer le script python toutes les minutes :

```
sudo crontab -e
```

On sélectionne son éditeur préféré et on ajoute la ligne

```
* * * * * /home/pi/gpscron
```

à la fin du fichier. Du côté du raspberry tout doit être prêt maintenant

## Connection de l'objet au LoRaWan (thethingsnetwork)

On va dans applications on crée une application ensuite on va dans enddevices et on choisi + Add Enddevice

ID	Name	DevEUI	JoinEUI	Last activity
eui-0001020303020100		00 01 02 03 03 02 01...	00 01 02 03 03 02 01...	21 hr. ago •
eui-0100000000000000		00 00 00 00 00 00 00...	00 00 00 00 00 D5 B3...	24 hr. ago •
eui-fffffdfcfcfdfeff		FF FE FD FC FC FD FE...	70 B3 05 00 00 05 B3...	20 sec. ago •

Ensuite on choisit les paramètres de l'objet (AppEUI, DevEUI, AppKey) pour qu'ils correspondent à ceux établis précédemment dans /home/pi/dragino/dragino.ini

soit dans l'exemple de cette étude :

```
deveui = 0xFF, 0xFE, 0xFD, 0xFC, 0xFC, 0xFD, 0xFE, 0xFF
appeui = 0x70, 0xB3, 0xD5, 0x00, 0x00, 0xD5, 0xB3, 0x70
appkey = 0x3D, 0x83, 0xC3, 0x16, 0x2C, 0xAD, 0x44, 0xB7, 0xB0, 0x50, 0x6C,
0x3C, 0xA1, 0x54, 0x36, 0xB7
```

The screenshot shows a Firefox browser window with the URL <https://eu1.cloud.thethings.network/console/applications/0109/devices/add/manual>. The page title is "Register end device - GPS". The sidebar on the left contains various application icons. The main content area is titled "From The LoRaWAN Device Repository" and "Manually". It includes fields for Frequency plan (Europe 863-870 MHz), LoRaWAN version (LoRaWAN Specification 1.0.1), Regional Parameters version (TS001 Technical Specification 1.0.1), DevEUI (generated as :F FE FD FC FC FD FE FF), AppEUI (70 B3 D5 00 00 D5 B3 70), AppKey (3D 83 C3 16 2C AD 44 B7 B0 50 6C 3C A1 54 36 B1), and End device ID (eui-Off0fe0fd0fc0fc0). A note at the bottom states "This value is automatically prefilled using the DevEUI".

Démarrer le pi (truc et astuces pour le GPS !!!!!)

Sur le shell du pi :

```
sudo ntpdate fr.pool.ntp.org
```

Mettre le RPi en extérieur Débrancher le jumper GPS Tx du Hat dragino alimenter le RPi attendre le 3D fix (la diode verte du dragino, pas du RPi) et brancher (à chaud) le jumper Tx.

Ca devrait y être vous avez votre premier (?) objet connecté (au LoRaWan)

## Format du message

Enfin dans le cas de cette étude nous avons choisi de mettre le payload sous la forme CayenneLPP on verra pourquoi par la suite. Pour que TheThingsNetwork puisse interpréter le payload il faut le lui dire

The screenshot shows the 'Default uplink payload formatter' configuration page. The sidebar on the left is collapsed. The main content area shows the breadcrumb path: Applications > GPS-tracker > Payload formatters > Uplink. The title is 'Default uplink payload formatter'. A note says: 'You can use the "Payload formatter" tab of individual end devices to test uplink payload formatters and to define individual payload formatter settings per end device.' Below is a 'Setup' section with a 'Formatter type\*' dropdown set to 'CayenneLPP'. There is a 'Save changes' button.

Pour voir l'objet sur ttn allez dans l'application que vous venez de créer sélectionner votre enddevice et live data vous devriez voir quelquechose comme

The screenshot shows the 'Live data' section for the GPS-tracker application. The sidebar is collapsed. The main content area shows the breadcrumb path: Applications > GPS-tracker > Live data. A table lists events with columns: Time, Entity ID, Type, and Event details. One event is selected: '19:52:02 eui-ffffefdfcfcf... Forward uplink data message'. A modal window displays the detailed message content:

```

"received_at": "2022-03-09T18:52:03.672803539Z",
"uplink_message": {
  "session_key_id": "AX9q2y+ye1b96x01hgKccw==",
  "f_port": 1,
  "f_cnt": 492,
  "frm_payload": "CYgGe4IAa8sAQPY=",
  "decoded_payload": {
    "gps_9": {
      "altitude": 166.3,
      "latitude": 42.4834,
      "longitude": 2.7595
    }
  },
  "rx_metadata": [
    {
      "gateway_ids": {
        "gateway_id": "a840411fab40",
        "eui": "A840411FAB404150"
      },
      "time": "2022-03-09T18:52:03.373684Z",
      "timestamp": 3415008451,
      "rssi": -22,
      "channel_rssi": -22,
      "snr": 9.5,
      "uplink_token": "ChoKGaMYTg0MDQzMZhYjQWEgioQEEfq0BBUBD"
    }
  ]
},

```

## Gestion des données (Intégration à Cayenne)

Aller sur <https://mydevices.com/>

Créer un compte Cayenne

Sélectionner TheThingsNetwork

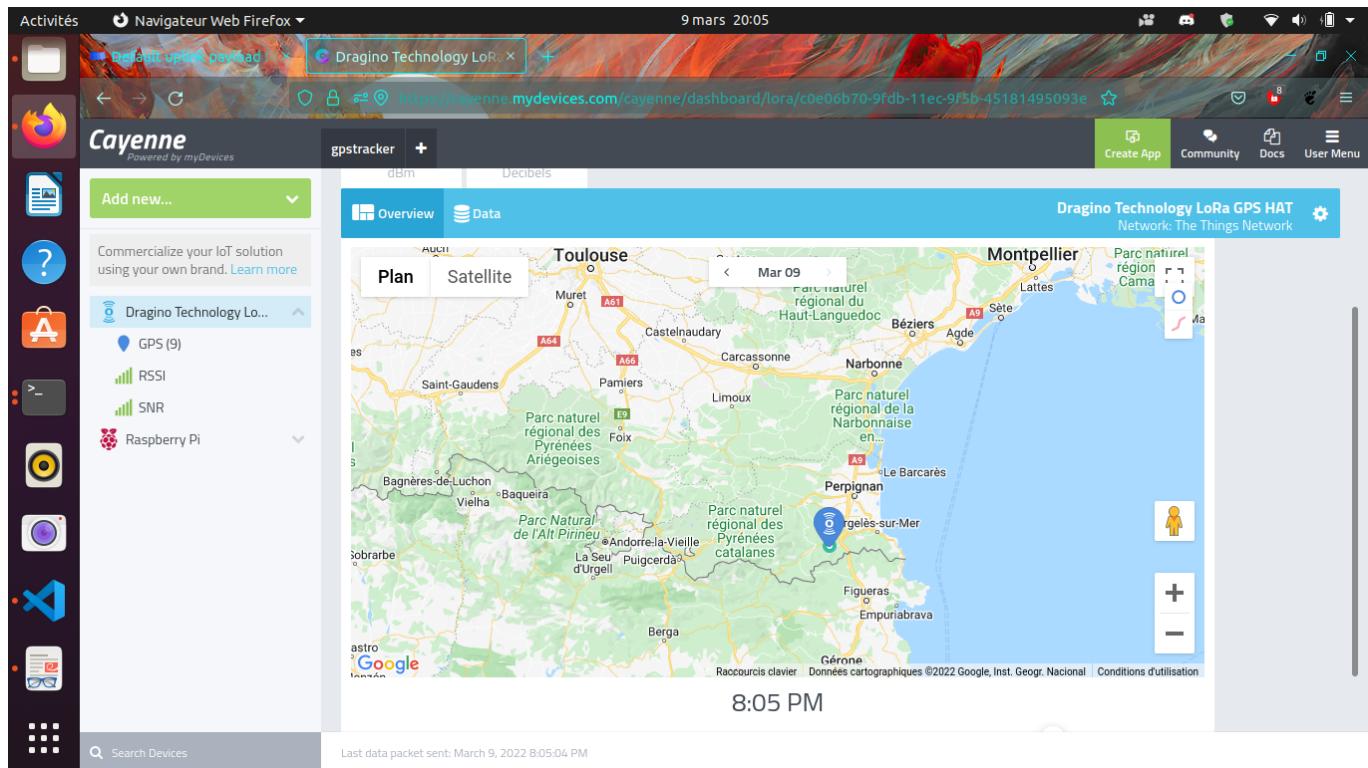
The screenshot shows the Cayenne IoT dashboard interface. At the top, there's a navigation bar with icons for file operations, a search bar, and user account information. Below the navigation bar is a header with the text "Cayenne Powered by myDevices" and a "gpstracker" tab. A sidebar on the left contains icons for file management, help, and a Raspberry Pi logo, along with a "Search Devices" bar.

The main content area displays a grid of network provider logos and names. The first row includes Helium, Kerlink, Loriot, Objenious, and OrbiWise. The second row includes Pixel Networks, Sagemcom, Semtech, Senet, and SenRa. The third row includes Spark, Stream, Swisscom, The Things Network, and X-Telia. Below this grid is a section titled "Custom Widgets" featuring five different data visualization cards: a temperature gauge showing 74°, a line graph, a red progress bar, a green square with the number 1, and a lightbulb icon.

## Sélection Dragino RPi Hat et mettre le DevEUI

This screenshot shows the Cayenne IoT dashboard with a similar layout to the previous one, but the central content area is now displaying a list of Dragino Technology products. The list includes:

- Dragino Technology LoRa GPS HAT (LoRa GPS expansion for RaspberryPi)
- Dragino Technology LoRa Sensor Node-LSN50 (Long Range LoRa Sensor Node)
- Dragino Technology Lora Shield (Long range transceiver)
- Dragino Technology Mini (Long distance wireless transceiver)
- Dragino Technology Mini Dev (Long distance LoRa wireless transceiver)



Données en live du traqueur GPS !!!!!!!!