# TP LORAWAN

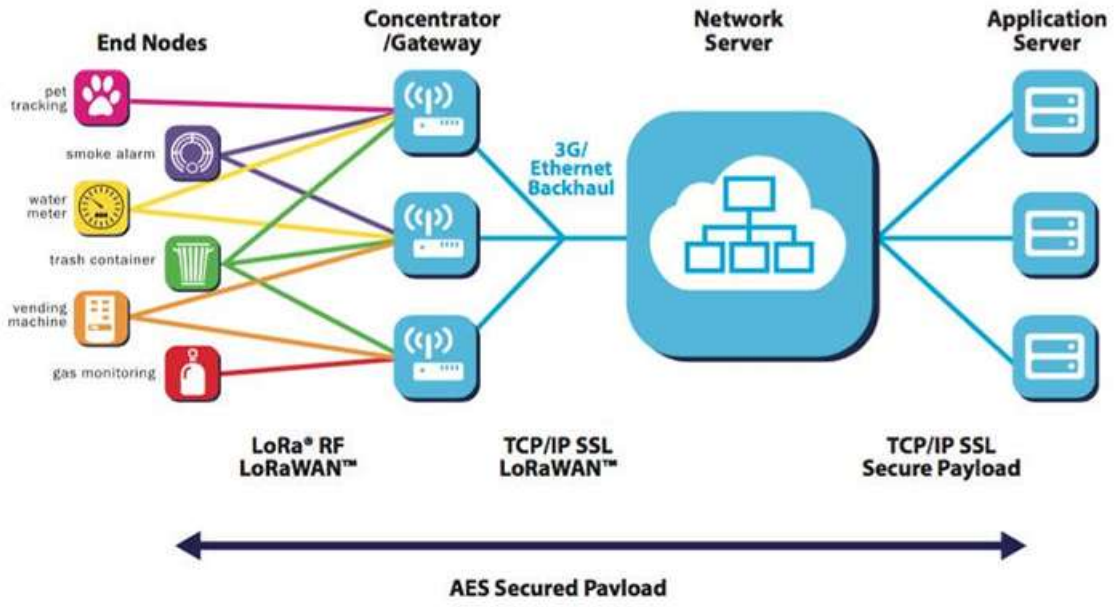
**BELOUGNE YANN**

**19/01/26**

**Introduction:**

LoRaWAN (Long Range Wide Area Network) est un protocole de communication qui utilise la technologie radio LoRa (Long Range) rendant possible la transmission de données sur de longues distances avec une faible consommation d'énergie. L'architecture LoRaWAN est constitué de end devices (appareils terminaux) qui collectent ou génèrent des données puis les envoient vers des serveurs d'applications à travers des messages appelés uplink. Les serveurs d'applications, quant à eux, décryptent et traitent ces données puis les affichent dans un tableau de bord ou les transmettent à d'autres systèmes via MQTT par exemple. Ces serveurs permettent aussi d'envoyer des commandes aux end devices, tel qu'allumer des LEDs, à travers des messages appelés downlink.

Dans la chaîne de communication entre les end devices et les serveurs d'applications, se trouve un serveur réseau et des passerelles. Le serveur réseau gère la communication LoRaWAN: Il s'assure de sa fiabilité en validant l'intégrité des messages par exemple; et dans les version antérieures à LoRaWAN 1.0.4, il s'assure aussi de la sécurité de la communication, en validant l'authenticité des end devices. Les passerelles, quant à elles, convertissent les signaux radios envoyés par les end devices en paquets IP qui seront compris par le serveur réseau; et inversement, elles convertissent les paquets IP du serveur réseau en signaux radios pour les end devices. Ainsi, la communication se fait uniquement par onde radio LoRa entre les end devices et les passerelles. Le reste de la communication se fait via Internet.




**Travail:**

**Installation de la passerelle:**

- Je suis les instructions du guide d'installation de la passerelle qui m'a été fournie
→ La passerelle est le RAK7246G

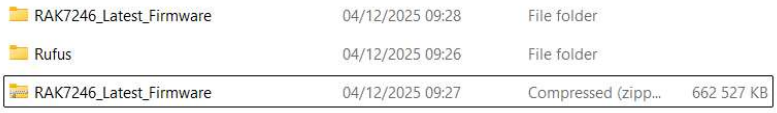
- Je télécharge le firmware permettant de faire fonctionner cette passerelle
→ Le firmware le plus récent est disponible depuis la page web de la documentation technique du RAK7246G

**Firmware Specification**

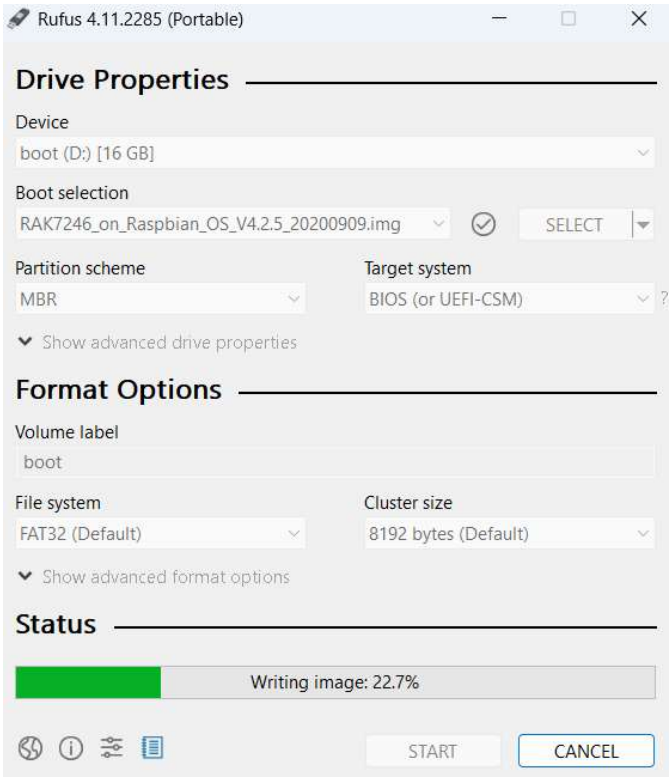
Download the latest firmware of RAK7246G in the table provided below. The supported software features are also included with the standard parameters.

| Model | Raspberry Pi Board | Firmware Version | Source |
|---|---|---|---|
| RAK7246 | Raspberry Pi Zero W | 4.2.0R | Download |

- Je unzip le fichier contenant le firmware



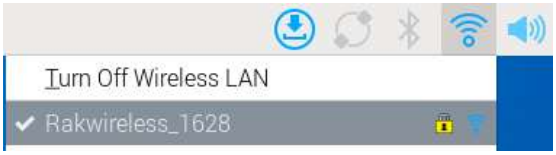

- J'installe le firmware sur une carte microSD en utilisant l'application Rufus



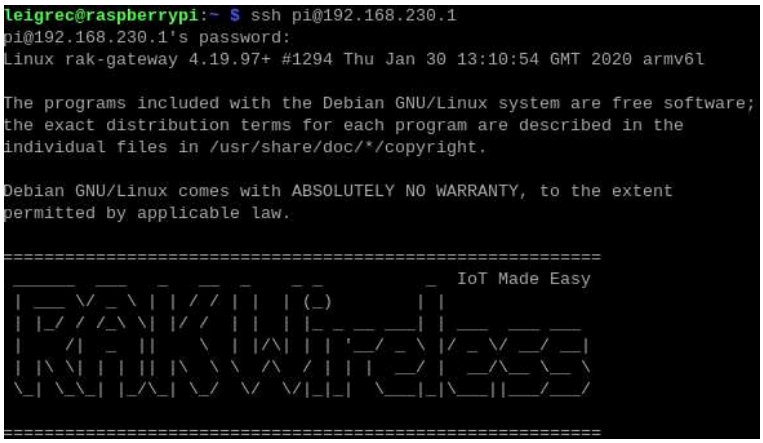

- J'insère la carte microSD dans la passerelle et j'installe l'antenne LoRa sur la passerelle

- J'alimente la passerelle
→ Une fois alimenté la passerelle se met en mode Wi-Fi AP, c'est-à-dire qu'elle émet des ondes pour créer son propre réseau Wi-Fi

- Depuis ma Raspberry Pi, je me connecte au réseau Wi-Fi de la passerelle
→ L'identifiant du réseau Wi-Fi est RakwirelessXXX et le mot de passe est rakwireless
→ L'adresse IP par défaut de la passerelle est 192.168.230.1
→ Une fois connecté l'identifiant du réseau change à Rakwireless_1628 où 1628 sont les deux derniers octets de l'adresse MAC de la passerelle

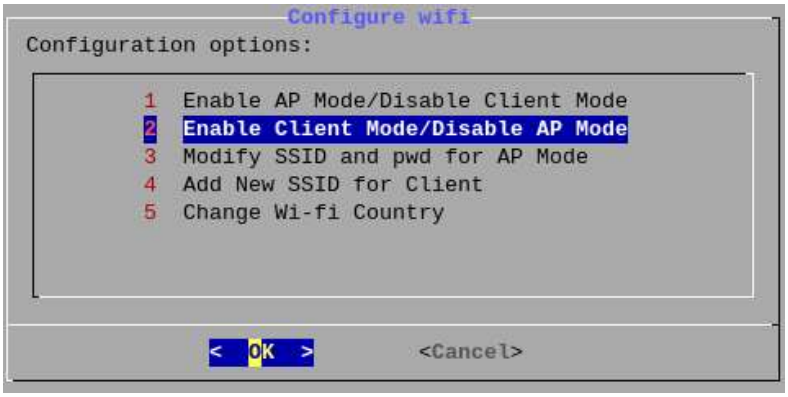



- Depuis le terminal de commande, je me connecte via SSH à la passerelle
→ J'entre la commande: ssh pi@192.168.230.1
→ Puis, j'entre la mot de passe 'raspberry'



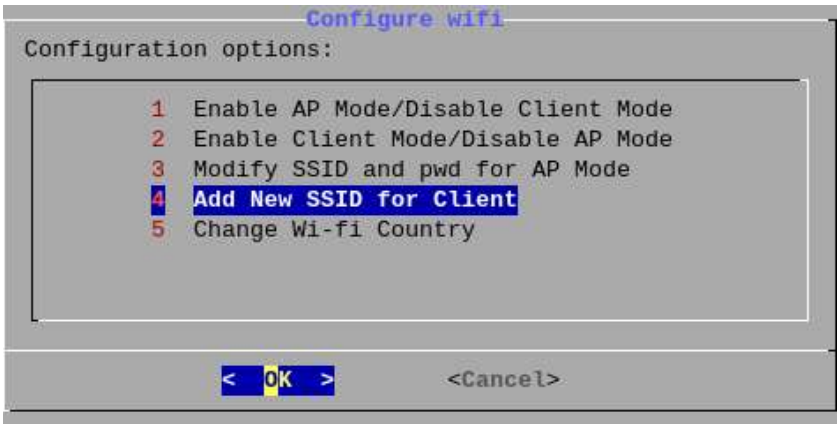

- J'ouvre l'outil de configuration de la passerelle
→ J'entre la commande: sudo gateway-config

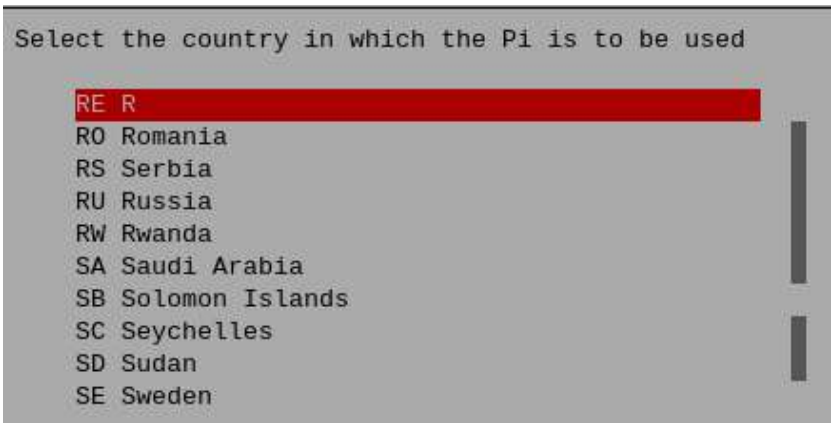

- Je configure la passerelle de sorte à ce que, après redémarrage, le mode Client s'active et le mode AP se désactive. En mode Client, la passerelle ne créé pas de réseau Wi-Fi mais peut se connecter à un réseau Wi-Fi existant
→ Je sélectionne Configure WIFI
→ Puis, je sélectionne Enable Client Mode/Disable AP Mode

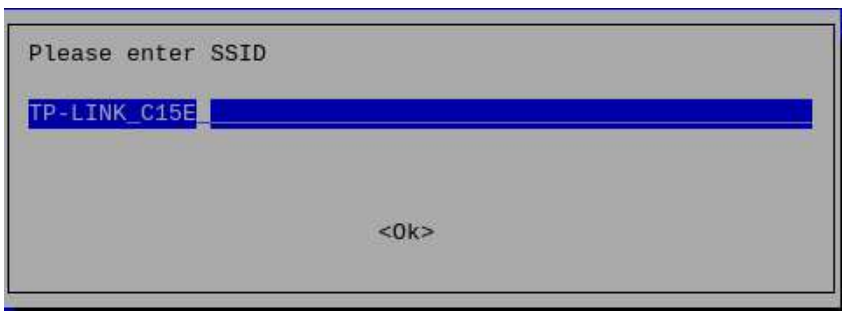



- J’entre les informations nécessaire à la connexion de la passerelle, après redémarrage, au réseau Wi-Fi de la salle

→ Dans le menu Configure WIFI, je sélectionne Add New SSID for Client
→ Ensuite, je sélectionne le pays de résidence: La Réunion
→ Puis, j’entre l’identifiant: TP-LINK_C15E
→ Finalement j’entre le mot de passe



- Je ferme l'outil de configuration puis je redémarre la passerelle depuis le terminal
→ J'entre la commande: sudo reboot
→ Le réseau Wi-Fi de la passerelle n'est maintenant plus visible depuis la Raspberry Pi et l'accès à la passerelle via SSH est maintenant perdu

```
pi@rak-gateway:~ $ sudo reboot
```

- Après redémarrage, la passerelle se connecte au réseau TP-LINK_C15E et se voit donc attribué une adresse IP différente que quand elle était en mode AP

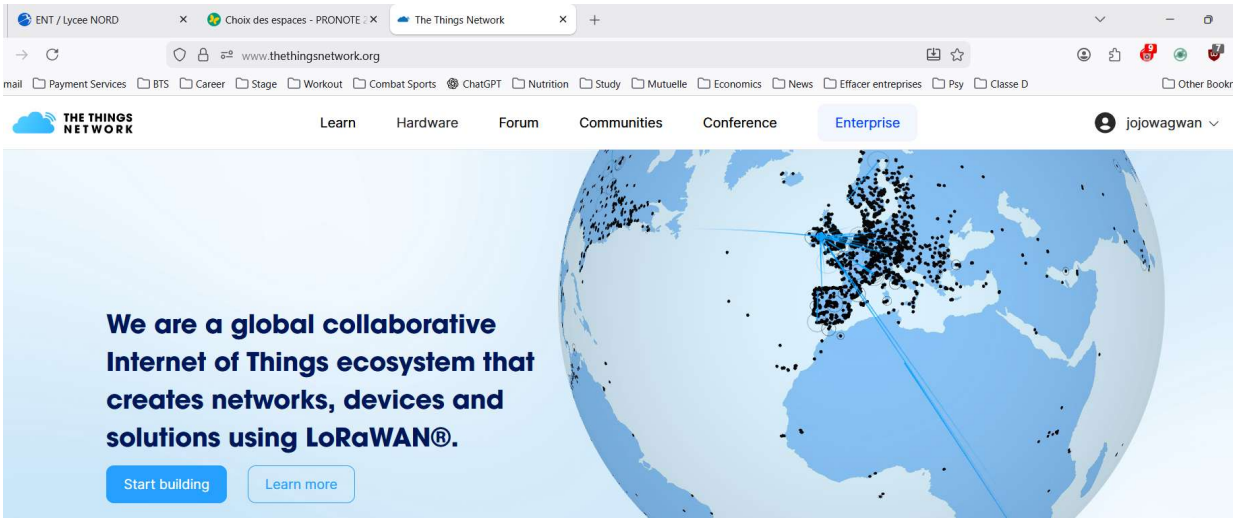
- J'obtiens son adresse IP avec le logiciel Advanced IP Scanner
→ Le logiciel affiche les adresses IP et adresses MAC de l'ensemble des appareils connectés au réseau TP-LINK_C15E

- Je connecte maintenant ma Raspberry Pi au réseau TP-LINK_C15E

- Je me connecte à nouveau via SSH à la passerelle

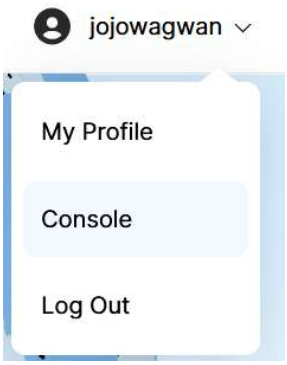
- Je récupère le EUI (Extended Unique Identifier) de la passerelle. C'est l'identifiant unique de la passerelle
→ J'entre la commande: gateway-version

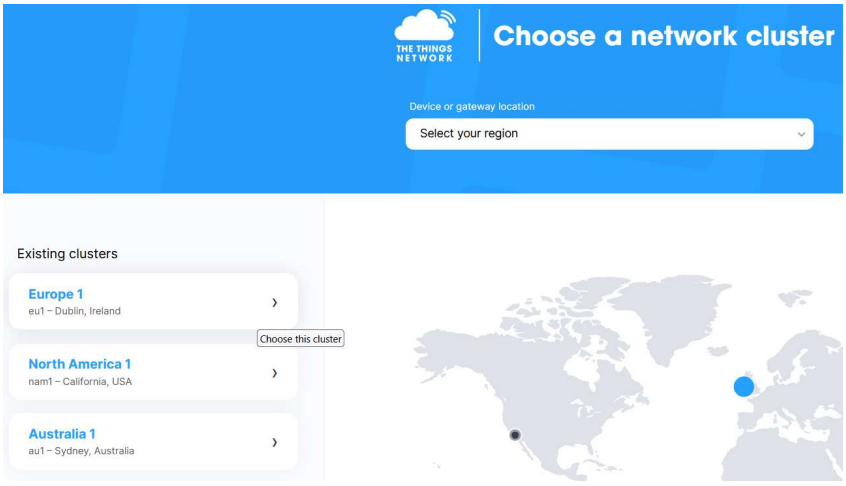
**<u>Installation du serveur réseau LoRaWAN:</u>**
- J'accède à la page web TTN (The Things Network)
→ TTN est une plateforme accessible en ligne pouvant fournir des serveurs réseaux et d'applications LoRaWAN



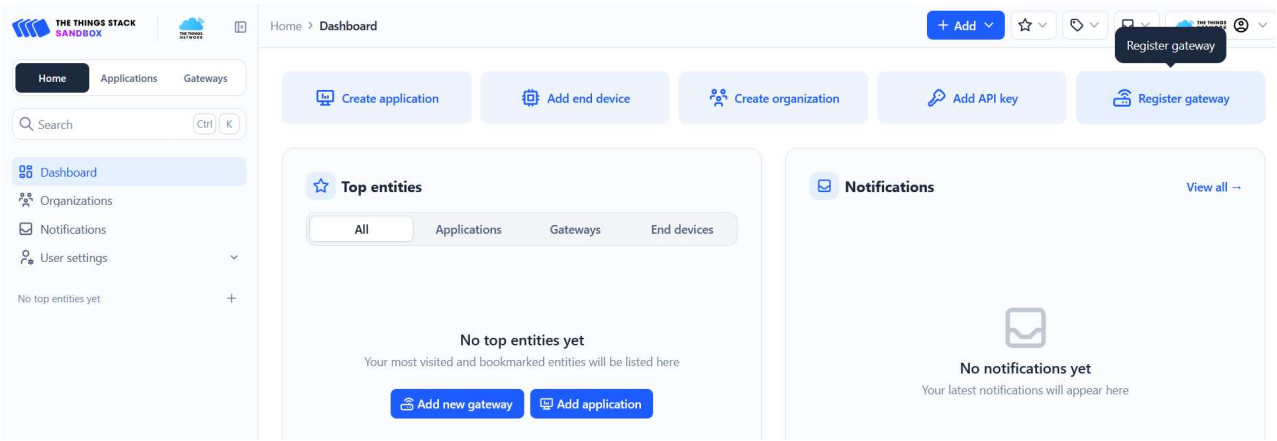

- Je créé un compte

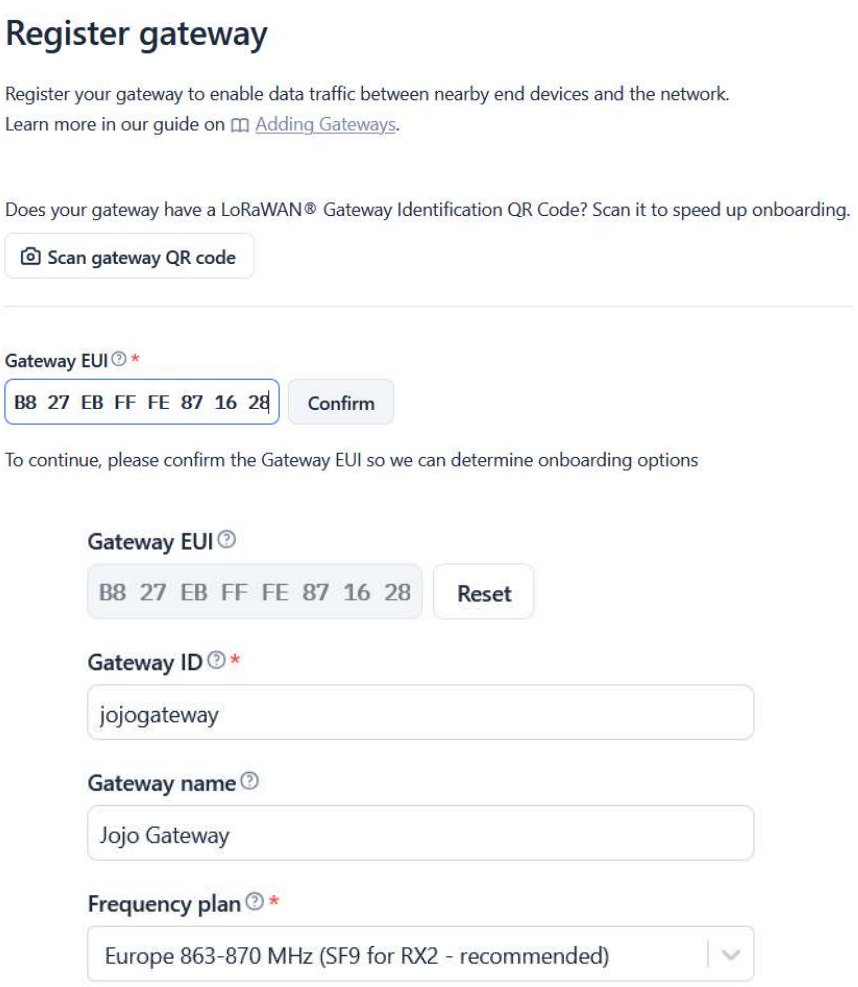
- J'accède à la console permettant de gérer mon réseau LoRaWAN
→ Je sélectionne console en haut à droite de la fenêtre
→ Puis je sélectionne la région Europe 1




- J’ajoute ma passerelle au réseau
→ Je sélectionne Register Gateway
→ J’entre le EUI
→ J’entre l’identifiant qui sera utilisé au sein du réseau, un nom ainsi que le plan de fréquence
→ Je sélectionne Register Gateway






- Le plan de fréquence est la plage de fréquences radio qui sera utilisé pour la communication entre les end devices et la passerelle
→ Cette plage de fréquence dépend principalement de la réglementation radio locale de chaque pays ou région. Par exemple la plage de fréquence autorisée en Europe est 863-870MHz tandis qu'aux États-Unis elle est de 902-928MHz

- Ma passerelle n'est pas visible depuis ma console donc je dois faire des configurations sur la passerelle

- J'accède au tableau de bord
→ Je sélectionne Home en haut à gauche de la fenêtre

- Je télécharge le fichier global_conf.json
→ Je sélectionne les trois tirets en haut à droite de la fenêtre
→ Puis je sélectionne Download global_conf.json

- Un packet forwarder est le logiciel d'une passerelle permettant de transmettre les paquets LoRa entre la puce LoRa et le serveur réseau
→ Il permet également de gérer les canaux et les fréquences de transmission
→ Les fichiers local_conf.json et global_conf.json sont des fichiers de configuration du packet forwarder

- Depuis mon terminal de commande, je me connecte à ma passerelle via SSH

- Je modifie le fichier local_conf.json de ma passerelle
→ J'entre dans le répertoire /opt/ttn-gateway/packet_forwarder/lora_pkt_fwd
→ Puis je remplace le contenu du fichier local_conf.json avec le contenu du fichier global_conf.json que j'ai téléchargé précédemment

- Je redémarre le packet forwarder
→ J'entre la commande: gateway-config
→ Puis je sélectionne Restart packet-forwarder

- Maintenant je peux apercevoir ma passerelle depuis ma console TTN

**Envoi d'une donnée depuis le nœud vers le serveur**

- On ma fourni la version shield du Dragino LA66
→ C'est un appareil conçu pour être un end device LoRaWAN
→ La version shield permet une utilisation simple et pratique avec une carte Arduino
→ Elle peut être contrôlé avec un microcontrôleur via des commandes AT qui sont répertoriés dans un fichier disponible depuis la page officiel du produit

- Depuis le site TTN, je mets en place un serveur d'application
→ Depuis mon tableau de bord, je sélectionne Create Application
→ Puis j'entre un identifiant et un nom pour l'application

- J'ajoute mon Dragino à l'application
→ Depuis la section Applications je sélectionne Register end device
→ Ensuite, je sélectionne Enter end device specifics manually
→ Puis, je sélectionne le même plan de fréquence que pour la passerelle ainsi que les premières versions du protocole LoRaWAN et des paramètres régionaux à utiliser
→ Finalement, j'entre le JoinEUI/AppEUI (identifiant jouant un rôle dans la sécurité de la communication), le DevEUI (Identifiant unique à l'appareil), le AppKey (clé de 128 bits utilisé pour la sécurité) et un identifiant pour le end device

- J'installe le Shield sur l'Arduino

- Je télécharge les exemples de programmes Arduino pour l'utilisation du Dragino, disponible sur la page web officielle de l'appareil

- Je lance le programme LA66-LoRaWAN-shield-AT-command-via-Arduino-UNO

- Il lit les caractères depuis le port série LA66 puis les envoie au moniteur série

```
while ( Serial.available()) {
// get the new byte:
char inChar = (char) Serial.read();
// add it to the inputString:
inputString += inChar;
// if the incoming character is a newlin
// do something about it:
if (inChar == '\n' || inChar == '\r') {
  LA66_serial_port.print(inputString);
  inputString = "\0";
```

- Il Lit la chaîne de caractère entrée dans le moniteur série puis l'envoi au Dragino

```
while ( LA66_serial_port.available()) {
// get the new byte:
char inChar = (char) LA66_serial_port.read();
// add it to the inputString:
inputString += inChar;
// if the incoming character is a newline, se
// do something about it:
if (inChar == '\n' || inChar == '\r') {
  stringComplete = true;

      if (stringComplete) {
      Serial.print(inputString);
```

- La commande AT+CFG commande au Dragino de me renvoyer toutes ses configuration

```
AT+RECV=
AT+VER=EU868 v1.3
AT+CFM=0,7,0
AT+SNR=14
AT+RSSI=-32
AT+PORT=2
AT+CHS=0
AT+RX1WTO=24
AT+RX2WTO=9
AT+DECRYPT=0
AT+SLEEP=0
AT+BAT=3332
AT+RJTDC=20
AT+RPL=0
AT+TIMESTAMP=systime= 1970/1/1 00:48:46 (2926)
AT+LEAPSEC=18
AT+SYNCMOD=0
AT+SYNCTDC=10
AT+DDETECT=0,1440,2880
AT+SETMAXNBTRANS=1,0
AT+DISFCNTCHECK=0
AT+DISMACANS=0
Start Tx events

OK
```

- AT+SEND et AT+SENDB sont des commandes qui permettent d'envoyer des uplinks

- Je veux créer une interface web qui va afficher les données envoyé depuis le nœud vers le serveur
→ Pour faire ceci je vais créer une connexion MQTT

- Sur le serveur TTN je vais dans applications
→ Je vais dans réseau LoRaWAN (nom de mon réseau)
→ Je vais dans other intergrations puis MQTT
→ Je génère new API key

- Je créé un programme python qui se subscribe au topic MQTT du serveur TTN pour ensuite l'envoyer à une page HTML via un serveur web

- J'utilise la bibliothèque paho.mqtt pout communiquer avec le protocole MQTT

- La fonction on_connect() commande la souscription au topic du serveur TTN

- La fonction on_message() définie que quand un message MQTT est reçu, les données sont converties en hexadécimale et en ASCII puis sont stockés dans la variable last_payload

```python
def on_message(client, userdata, msg):
    global last_payload
    try:
        data = json.loads(msg.payload.decode())
        uplink = data.get("uplink_message", {})

        raw_bytes = uplink.get("decoded_payload", {}).get("raw_bytes", [])

        if not raw_bytes:
            frm = uplink.get("frm_payload", "")
            raw_bytes = list(base64.b64decode(frm)) if frm else []

        ascii_text = "".join(chr(b) for b in raw_bytes if 32 <= b <= 126)

        hex_text = " ".join(f"{b:02X}" for b in raw_bytes)

        last_payload = {
            "raw_bytes": raw_bytes,    # décimal
            "hex": hex_text,           # hexadécimal
            "ascii": ascii_text        # texte lisible

        }

        print("Uplink reçu :", last_payload)
```

- J'utilise la bibliothèque flask pour créer mon serveur web

- La fonction index() indique d'ouvrir la page index5.html lorsque le lien http://localhost:5000/ est ouvert sur un navigateur web

- La fonction uplink() indique de renvoyer la variable last_payload lorsque le lien /uplink est ouvert

- Je créé la page index5.html

- La fonction refreshUplink() ouvre le lien /uplink, récupère les données de la variable last_payload puis l'affiche dans un conteneur
→ Cette fonction est appelée toute les 2 secondes

```javascript
function refreshUplink() {
    fetch("/uplink")
        .then(r => r.json())
        .then(d => {
            document.getElementById("uplink").innerText =
                "Raw bytes : " + JSON.stringify(d.raw_bytes) + "\n" +
                "ASCII     : " + d.ascii + "\n" +
                "HEX       : " + d.hex;
        });
}
```

- Je lance le programme LA66-LoRaWAN-shield-AT-command-via-Arduino-UNO

- J'entre la commande AT+SENDB=0,2,2,4D4D depuis le moniteur série
→ Je reçois bien la valeur sur l'interface web

**Dernier uplink reçu**

```
Raw bytes : [77,77]
ASCII     : MM
HEX       : 4D 4D
```

- J'entre la commande AT+SEND=0,2,8,mongasur

→ Ça fonctionne aussi


**Envoi d'une donnée depuis le serveur vers le nœud:**

- Maintenant je modifie la page HTML pour qu'on puisse commander un envoi de downlink

- Dans la page HTML j'ai mis en place un conteneur ou il est possible d'entrer du texte et de sélectionner un numéro de port

```
<div class="box">
    <h2>Envoyer un downlink</h2>
    <label>Texte ASCII à envoyer :</label><br>
    <textarea id="ascii"></textarea><br>
    <label>FPort :</label>
    <input type="number" id="fport" value="1"><br>
    <button onclick="sendDownlink()">Envoyer</button>
```

- Lorsque le bouton Envoyer est appuyé, la fonction sendDownlink() est appelée

- La fonction sendDownlink() ouvre le lien /downlink puis envoie le texte entré dans le conteneur ainsi que le numéro de port

```
function sendDownlink() {
    const asciiText = document.getElementById("ascii").value;
    const fport = document.getElementById("fport").value;

    fetch("/downlink", {
        method: "POST",
        headers: { "Content-Type": "application/json" },
        body: JSON.stringify({
            ascii: asciiText,
            f_port: fport
```

- Maintenant je modifie le programme python

- la fonction downlink() indique que lorsque le lien /downlink est ouvert, le texte et le numéro de port envoyé depuis l'interface web sont récupéré puis la fonction send_downlink est appelée

```python
def downlink():
    data = request.json

    # Texte ASCII reçu depuis la page HTML
    ascii_text = data.get("ascii", "")

    # FPort LoRaWAN
    fport = int(data.get("f_port", 10))

    # Conversion ASCII → bytes décimaux
    raw_bytes = [ord(c) for c in ascii_text]

    # Envoi du downlink
    send_downlink(raw_bytes, fport)

    return jsonify({
        "status": "OK",
        "ascii_sent": ascii_text,
        "raw_bytes": raw_bytes
```

- la fonction send_downlink() commande d'envoyer via MQTT au serveur TTN le texte à envoyer et le numéro de port dans le format nécessaire qui va indiqué au serveur TTN d'envoyer le downlink

```python
def send_downlink(raw_bytes, fport):
    payload = {
        "downlinks": [
            {
                "f_port": fport,
                "frm_payload": base64.b64encode(bytes(raw_bytes)).decode(),
                "confirmed": False,
                "priority": "NORMAL"
            }
        ]
    }
    mqtt_client.publish(topic_downlink, json.dumps(payload))
    print("Downlink envoyé :", payload)
```

- L'architecture LoRaWAN fait que la réception d'un downlink ne peut que se faire lors d'un laps de temps préci après qu'un uplink soit envoyée depuis le noeud
→ https://www.thethingsnetwork.org/docs/lorawan/classes/

- Voilà pourquoi mon programme Join-TTN-networkLEDONOFFASCII envoie des uplink tout les 10 secondes

```
if((new_time-old_time>=uplink_interval)&&(network_joined_status==1)){
  old_time = new_time;
  get_LA66_data_status=false;
  ss.println("AT+SENDB=0,2,4,4A4F4A4F");//confirm status,Fport,payload
}
```

```
long uplink_interval=10000;
```

- Il lit les données envoyées par le Dragino (Le Dragino envoie automatiquement les données qu'il reçoit en downlink)

```
while ( ss.available()) {
// get the new byte:
char inChar = (char) ss.read();
// add it to the inputString:
inputString += inChar;
```

- Si les données commencent par AT+RECVB= alors les données sont affichés dans le moniteur série en HEX et en ASCII

```
if (strncmp(rxbuff, "AT+RECVB=", 9) == 0) {
  stringComplete = false;
  inputString = "\0";

  char* data = &rxbuff[9];   // après AT+RECVB=
  Serial.print("\nFPort & Payload (HEX): ");
  Serial.println(data);
  // Séparer FPort et payload
  char* payloadHex = strchr(data, ':');
  if (payloadHex != NULL) {
    payloadHex++;  // sauter le ':'

    char payloadAscii[50];
    hexToAscii(payloadHex, payloadAscii);

    Serial.print("Payload HEX   : ");
    Serial.println(payloadHex);
    Serial.print("Payload ASCII : ");
    Serial.println(payloadAscii);
  }
```

**<u>Contrôle d'une LED depuis l'interface WEB:</u>**

- Maintenant je veux pouvoir contrôler une LED depuis la page web

- J'ajoute du code dans mon programme HTML

- J'ajoute un bouton LIGHT ON et un bouton LIGHT OFF
→ Lorsque le bouton LIGHT ON est appuyé, le texte LIGHTON est envoyé en downlink; et lorsque le bouton LIGHT OFF est appuyé, le texte LIGHTOFF est envoyé en downlink

- J'ajoute du code dans le programme arduino

- Lorsque la valeur hexadécimale correspondant au texte LIGHTON est reçu alors la LED interne de l'Arduino est allumé

- Lorsque la valeur hexadécimale correspondant au texte LIGHTOFF est reçu alors la LED interne de l'Arduino est éteinte

```
// LIGHTON -> LED ON
if (strstr(&rxbuff[9], "4c494748544f4e") != NULL) {
  digitalWrite(13, HIGH);
  Serial.println("LED ON");
}

// LIGHTOFF\n -> LED OFF
if (strstr(&rxbuff[9], "4c494748544f4646") != NULL) {
  digitalWrite(13, LOW);
  Serial.println("LED OFF");
}
```

**Matériels utilisés:**
- Arduino Mega 2560
→ https://docs.arduino.cc/hardware/mega-2560/
- Shield Dragino LA66 LoRaWAN
→ https://www.dragino.com/products/lora/item/231-la66-lorawan-shield.html
- Raspberry Pi 5
→ https://datasheets.raspberrypi.com/rpi5/raspberry-pi-5-product-brief.pdf
- Passerelle RAK7246G
→ https://docs.rakwireless.com/product-categories/wisgate/rak7246g/datasheet/
- Carte microSD

**Logiciels utilisés:**
- Arduino

**Liens visités:**
- Datasheet Shield Dragino LA66 LoRaWAN
→ https://www.dropbox.com/scl/fo/2i1w7s2769os6n98ntbh7/AIXWoxTZhl92ryP4OGoHt_Q/Datasheet_LA66%20LoRaWAN%20Shield.pdf?rlkey=284yfbo3pr86k4ef1esstl094&e=1&dl=0
- Datasheet Dragino LA66 LoRaWAN
→ https://www.dropbox.com/scl/fo/2i1w7s2769os6n98ntbh7/AFdD0FD1xiE-AA9bAWENToQ/Datasheet_LA66%20LoRaWAN%20Module.pdf?rlkey=284yfbo3pr86k4ef1esstl094&e=1&dl=0
- Manuel d'utilisation Shield Dragino LA66 LoRaWAN
→ https://wiki.dragino.com/xwiki/bin/view/Main/User%20Manual%20for%20LoRaWAN%20End%20Nodes/LA66%20LoRaWAN%20Shield%20User%20Manual/
- Commandes AT Dragino LA66 LoRaWAN
→ https://www.dropbox.com/scl/fo/2i1w7s2769os6n98ntbh7/AFkqQforcfAuSNeTl4vomIE/LA66%20AT%20commands.pdf?rlkey=284yfbo3pr86k4ef1esstl094&e=1&dl=0
- Exemples de programmation Shield Dragino LA66 LoRaWAN
→ https://www.dropbox.com/scl/fo/9tboozbcyrr3xy4dwcsov/AJnDhdVzzfQYCtzwimvpoLY?rlkey=081i0yz6x2elkyu7wvl1og03t&e=1&dl=0
- Guide installation passerelle RAK7246G
→ https://docs.rakwireless.com/product-categories/wisgate/rak7246g/quickstart/
- Guide installation serveur TTN
→ https://docs.rakwireless.com/product-categories/wisgate/rak7246g/lorawan-network-server-guide/
- Page web Rufus
→ https://rufus.ie/en/
- Page web The Things Network
→ https://www.thethingsnetwork.org/