**Chapitre X : Station Météo Connectée**

1. **Introduction**

Dans le cadre du projet collectif sur la **serre connectée**, notre groupe a pris en charge le développement d’une **station météo intelligente**. Cette dernière permet de **surveiller les conditions climatiques extérieures** en temps réel, et de transmettre les données au système global via une architecture LoRaWAN. Ces informations sont essentielles pour ajuster automatiquement des paramètres de la serre tels que l’irrigation, la ventilation ou le chauffage.

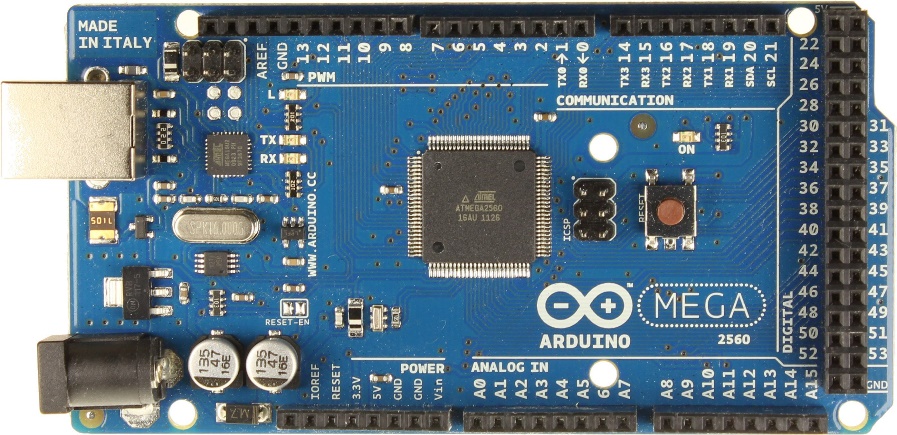
1. **Objectifs**

* Collecter des **données météorologiques complètes** :
* Température de l'air
* Humidité relative
* Pression atmosphérique
* Vitesse du vent
* Direction du vent
* Hauteur de pluie (rainfall)
* Transmettre ces données via **LoRa** en utilisant **OTAA**.
* Publier les données dans un **MQTT broker** via **ChirpStack**.
* Partager ces informations avec les autres sous-systèmes de la serre

1. **Matériel utilisé**

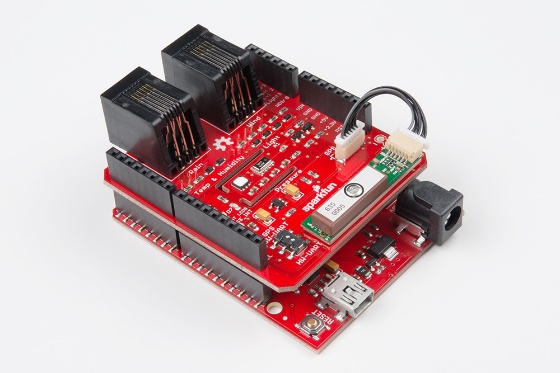
| **Composant** | **Référence** | **Fonction** |
| --- | --- | --- |
| **Carte microcontrôleur** | Arduino Mega | Contrôle des capteurs |
| **Carte météo Shield** | SparkFun Weather Shield | Mesure T°, H%, pression, vent, pluie |
| **Anémomètre + Girouette + Pluviomètre** | Intégrés via Weather Meters | Mesure vent et pluie |
| **Module LoRa** | RAk811 | Transmission LoRaWAN |
| **Passerelle LoRa** | RAK7243 | Relais vers ChirpStack |
| **Plateforme serveur** | ChirpStack | Gestion réseau et application |
| **Broker MQTT** | Mosquitto | Diffusion des données |

1. **Carte microcontrôleur – Arduino Mega 2560**



L’Arduino Mega est le cœur de notre station météo. Il joue le rôle d’unité centrale de traitement. Grâce à ses nombreux ports d’entrée/sortie et à sa mémoire étendue, il est parfaitement adapté à la lecture simultanée de plusieurs capteurs. Il récupère les données climatiques mesurées par la SparkFun Weather Shield, les traite, les organise et les prépare pour l’envoi via le module LoRa.

1. **Carte météo – SparkFun Weather Shield**



Cette carte spécialisée est conçue pour les stations météo Arduino. Elle intègre plusieurs capteurs environnementaux :

* **HTU21D** : mesure la température de l’air et l’humidité relative.
* **MPL3115A2** : capteur de pression barométrique.
* **ALS-PT19** : capteur de luminosité (optionnel selon version).

La carte possède également des connecteurs destinés à accueillir des capteurs mécaniques externes :

* **Anémomètre** : mesure la **vitesse du vent**.
* **Girouette** : mesure la **direction du vent**.
* **Pluviomètre** : mesure la **quantité de pluie tombée** grâce à un système de bascule à augets.

Elle permet donc de récupérer l’ensemble des paramètres météorologiques utiles à la gestion d’une serre.



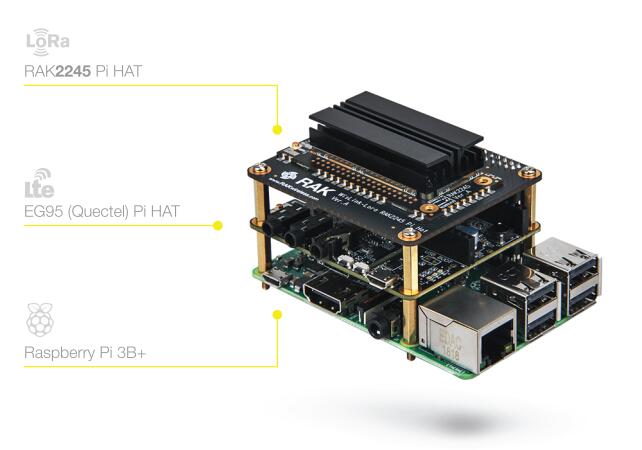
1. **Module de communication – LoRa (RA-02 / SX1276)**

Le module LoRa est utilisé pour transmettre les données météorologiques collectées vers la passerelle distante. Il fonctionne sur une fréquence de 868 MHz, adaptée à l’Europe, et permet une communication à très longue portée, tout en consommant peu d’énergie.  
Il communique avec l’Arduino via SPI et envoie les messages sous forme de paquets radio. Dans notre projet, le module est configuré pour une activation sécurisée par OTAA (Over-The-Air Activation).



1. **Passerelle LoRaWAN – RAK7243**

La passerelle RAK7243 joue un rôle fondamental dans le système. Elle reçoit les paquets radio émis par le module LoRa, les encapsule dans des paquets UDP, puis les transmet à un serveur LoRaWAN (ChirpStack).  
Elle est composée d’un concentrateur LoRa (SX1301) couplé à un Raspberry Pi 3, ce qui lui permet à la fois de gérer la partie radio et de se connecter au réseau Ethernet ou Wi-Fi pour transférer les données.

1. **Serveur réseau – ChirpStack**

ChirpStack est une plateforme serveur LoRaWAN open source. Elle est responsable de :

* La gestion des appareils connectés au réseau LoRa,
* L’authentification des nœuds via leurs identifiants OTAA (DevEUI, AppEUI, AppKey),
* Le décodage des paquets reçus (généralement encodés en base64),
* Et enfin de la transmission des données sous forme de messages MQTT.

Grâce à son interface web, ChirpStack nous permet de suivre en direct les paquets reçus, l’état des capteurs, et de configurer facilement notre station.



1. **Système de messagerie – MQTT (Mosquitto)**

Une fois les données traitées par ChirpStack, elles sont publiées sur un broker MQTT (Mosquitto dans notre cas). MQTT est un protocole de messagerie léger, idéal pour les communications entre machines.  
Les messages publiés sur un topic spécifique sont ensuite récupérables par d’autres modules de la serre (comme l’irrigation ou la ventilation), ou affichés sur des dashboards comme Node-RED ou Grafana.

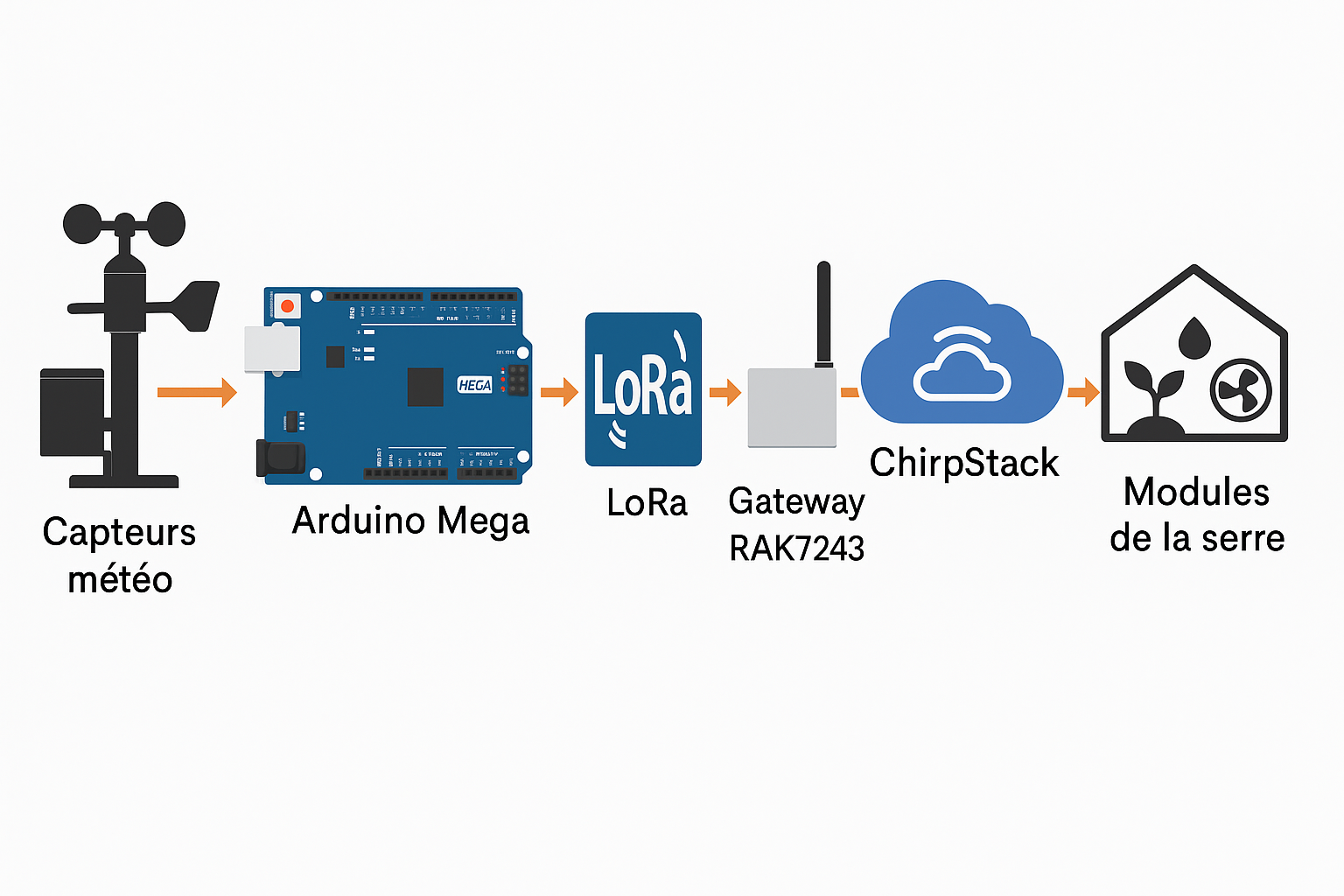


1. **Données mesurées**

La station météo mesure les paramètres suivants :

| **Donnée mesurée** | **Unité** | **Utilité dans la serre** |
| --- | --- | --- |
| Température (°C) | Celsius | Contrôle du chauffage ou des ouvertures |
| Humidité (%) | % | Gestion de l’humidité interne |
| Pression (hPa) | Hectopascals | Anticipation des changements météo |
| Direction du vent | Degrés (°) | Orientation automatique éventuelle |
| Vitesse du vent | m/s ou km/h | Sécurité (fermeture en cas de vent fort) |
| Pluie | mm | Gestion de l’irrigation |

1. **Architecture du système**

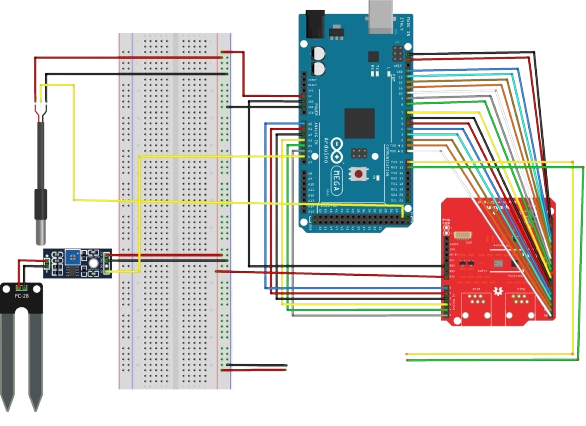
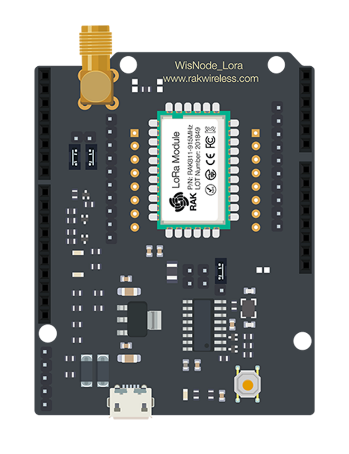


L’architecture de la station météo repose sur une chaîne de transmission structurée et optimisée pour l’environnement d’une serre connectée. Les capteurs météorologiques, reliés à la carte SparkFun Weather Shield, mesurent des paramètres tels que la température, l’humidité, la pression atmosphérique, la vitesse et la direction du vent, ainsi que la pluviométrie. Ces données sont collectées et traitées par une carte Arduino Mega, qui les transmet via un module LoRa fonctionnant à 868 MHz. La passerelle RAK7243 reçoit les paquets LoRa et les transfère au serveur réseau ChirpStack, chargé de l’authentification du dispositif, du décodage des données et de leur publication sur un broker MQTT. Grâce à ce schéma, les informations sont accessibles en temps réel par les autres modules de la serre, permettant une automatisation intelligente en fonction des conditions extérieures.

1. **Méthodologie et Implémentation**

La mise en œuvre de notre station météo connectée repose sur une architecture combinant capteurs environnementaux, traitement embarqué avec Arduino Mega, et transmission LoRaWAN via un module RAK811. Cette partie détaille le processus de configuration, de programmation, et d’intégration avec la passerelle RAK7243 et le serveur ChirpStack.

1. **Configuration de l’environnement Arduino**



Nous avons utilisé **Arduino IDE** pour le développement embarqué sur **Arduino Mega 2560**. Après installation des bibliothèques nécessaires (SparkFunMPL3115A2, SparkFun\_Si7021, RAK811, DallasTemperature, etc.), nous avons développé un programme structuré en trois grandes fonctions :

* **Acquisition des données capteurs** : température, humidité relative, pression atmosphérique, direction et vitesse du vent, pluie, luminosité, niveau de batterie et humidité du sol.
* **Affichage des mesures sur le port série**, avec conversion d’unités et interprétation des directions du vent.
* **Envoi des données formatées** via LoRaWAN, en mode OTAA, avec une trame ASCII codée en hexadécimal.

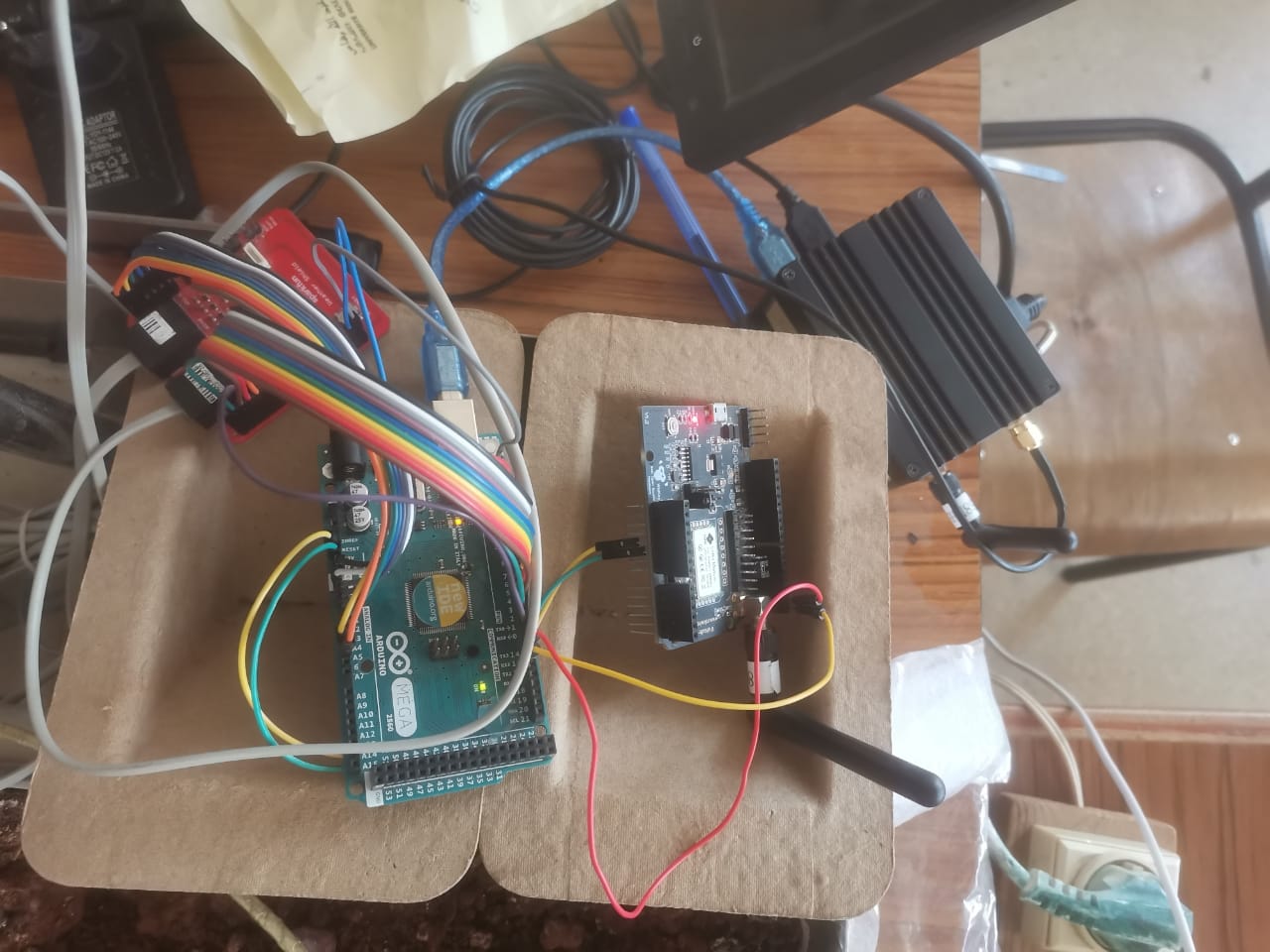
1. **Lecture et traitement des capteurs**

La station utilise une combinaison de capteurs :

* **SparkFun Weather Shield** : pour mesurer température (capteur MPL3115A2), humidité (SI7021), pression, pluie, vent.
* **FC-28** : pour l’humidité du sol.
* **Sonde Dallas DS18B20** : pour la température du sol.
* **Entrées analogiques** : pour la batterie et la luminosité.

Les interruptions matérielles sur les broches D2 et D3 permettent de détecter les impulsions du pluviomètre et de l’anémomètre avec précision.

Toutes les données sont calculées, moyennées (ex : rafale de vent sur 10 min), puis transformées en un format JSON-like avant envoi.





1. **Structure de la trame LoRa**

Le message transmis est préparé au format JSON, puis converti en hexadécimal pour être envoyé avec la commande AT :

**at+send=lora:1:<payload\_hex>**

Exemple de trame JSON générée :

*{"wd":315,"ws":85, "tp":245,"rn":12,"pr":1013,"tm":228,"hs":63}*

Chaque champ représente un paramètre environnemental (direction vent, vitesse, température, humidité, pluie, pression, etc.), avec une mise à l’échelle pour réduire la taille du message.



**Figure : Code de conversion de JSON vers Hexadécimale**

Code de décodage sur plateforme Chirpstack en Javascript

function Decode(fPort, bytes, variables) {

    try {

        // Convert bytes to string

        var payloadStr = String.fromCharCode.apply(null, bytes);

        // Handle different ports

        if (fPort === 1) {

            // Port 1: Weather data (JSON format)

            var data = JSON.parse(payloadStr);

            return {

                // Weather data

                wind\_direction: data.wd,

                wind\_speed\_kmh: data.ws / 10,

                temperature\_c: data.tp / 10,

                humidity\_percent: data.hm / 2,

                rainfall\_mm: data.rn / 10,

                pressure\_hpa: data.pr,

                Temperature\_soil: data.tm /10,

                Humdity\_soil: data.hs,

                Planet\_height: data.ht ,

                port: fPort,

            };

        } else {

            return {

                error: "Unknown port: " + fPort,

                raw\_payload: payloadStr

            };

        }

    } catch (e) {

        return {

            error: "Decoding failed: " + e.message,

            port: fPort,

            raw\_bytes: bytes,

            raw\_payload: String.fromCharCode.apply(null, bytes)

        };

    }

}

1. **Configuration du module RAK811 via RAK Serial Port Tool**

Le module RAK811 est configuré via l’outil officiel **RAK Serial Port Tool**. Ce logiciel permet d’envoyer des commandes AT au module pour définir les paramètres LoRaWAN.

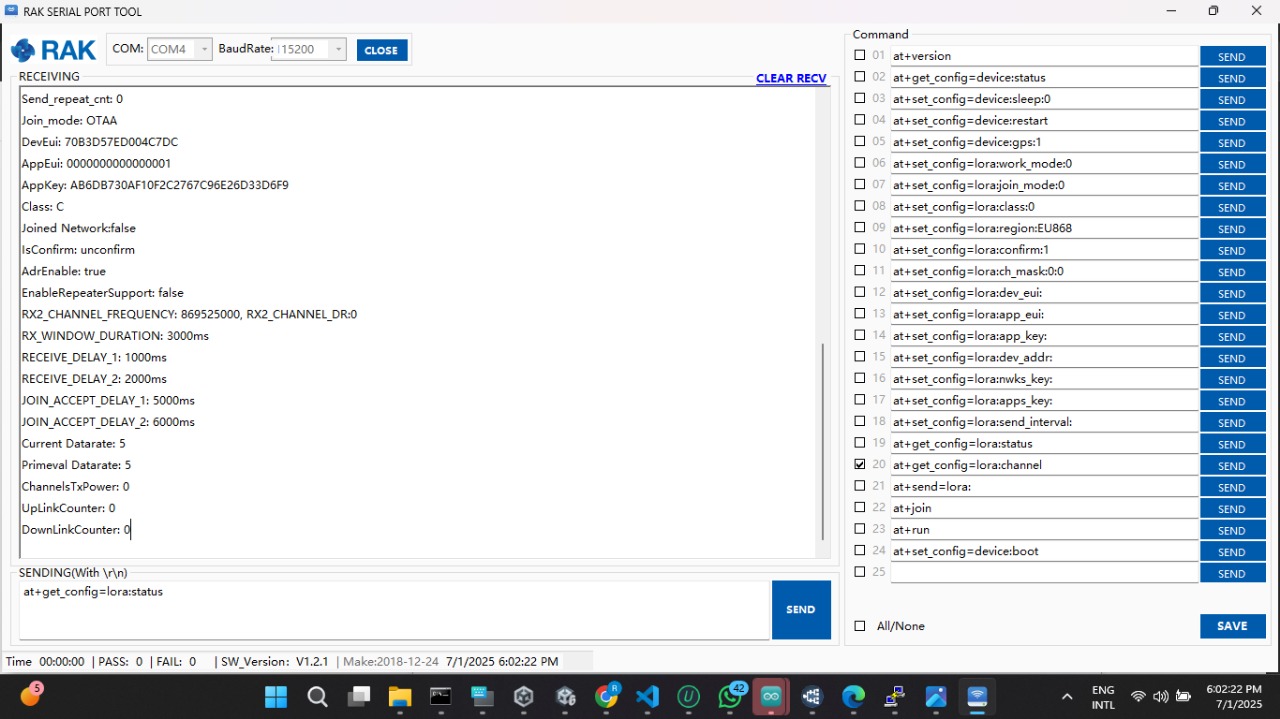
**a) Consultation des paramètres LoRa**

****Après connexion au port série, la commande suivante permet de consulter le statut du module :

**at+get\_config=lora:status**

Les paramètres affichés sont :

* **Mode** : LoRaWAN
* **Join Mode** : OTAA
* **DevEUI / AppEUI / AppKey** : identifiants uniques du nœud
* **Fréquence** : EU868
* **Classe** : C
* **Statut Join** : false (en attente de commande at+join)
* **IsConfirm** : unconfirm (transmission non confirmée)



**Figure : Détails supplémentaires RAKwireless : RX windows, datarate, délais, compteur uplink/downlink**

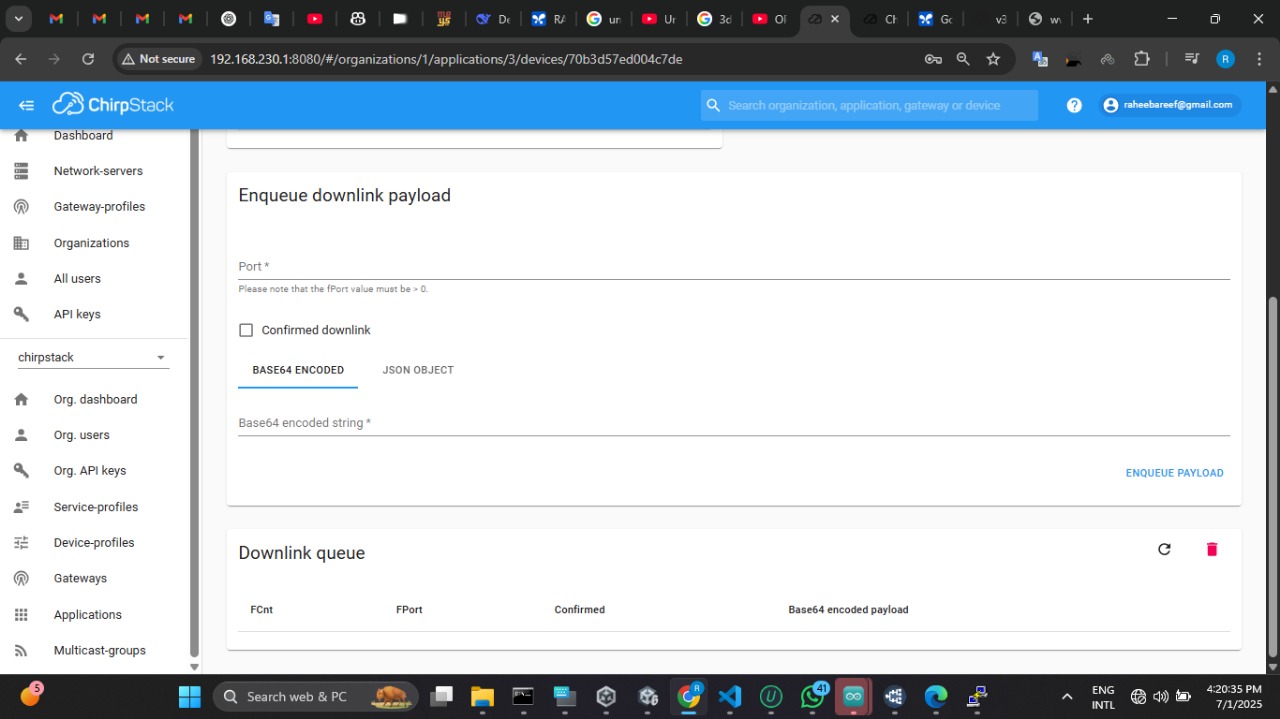
1. **Envoi de la commande at+join**

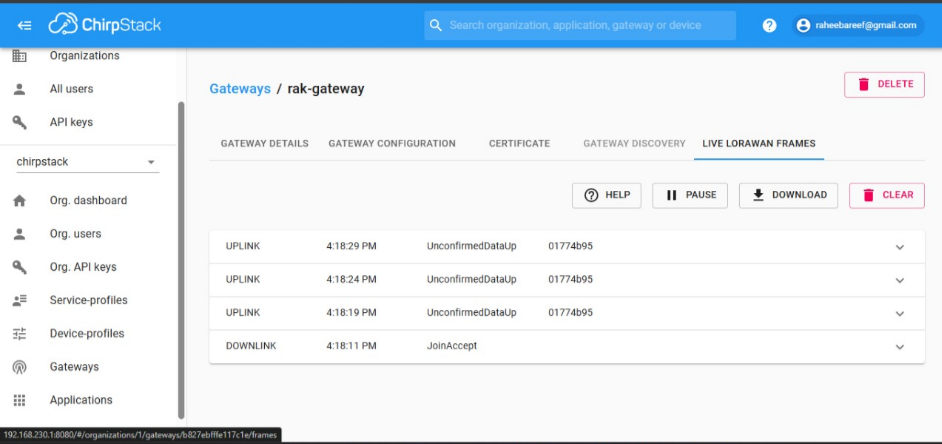
Une fois les paramètres vérifiés, la commande suivante est envoyée depuis l’Arduino (ou directement via l’outil RAK) :

**at+join**

Le module tente alors de se connecter au réseau LoRaWAN via la passerelle RAK7243. Une réponse OK Join Success valide l’enregistrement du nœud sur le serveur ChirpStack.

On voit sur chipstack un downlink dont « joinAccept » notre join est accepter





**Figure : test de message d’envoi sur Gateway depuis RAK8111**

1. **Objectif de la suite**

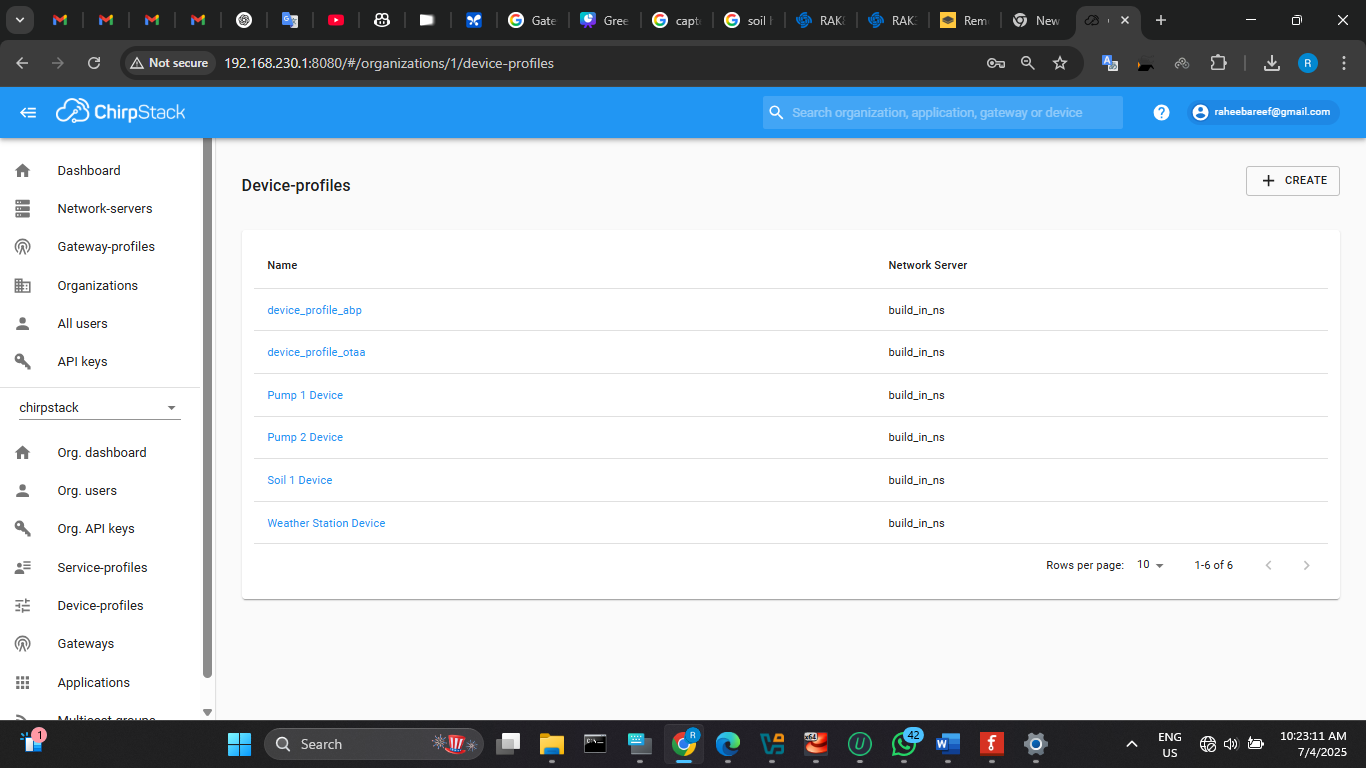
Une fois connecté, le module transmet les trames météo à la passerelle, qui les relaie vers ChirpStack. Dans la suite du rapport, nous détaillerons :

* L’enregistrement du device dans ChirpStack
* La configuration du payload decoder
* La publication dans MQTT et affichage via dashboard

1. **Configuration ChirpStack**

**Partie 1 : Gestion des Device-profiles dans ChirpStack**

Les **Device-profiles** dans ChirpStack sont des modèles de configuration prédéfinis permettant de standardiser les paramètres des appareils LoRaWAN. Ils simplifient le déploiement en évitant une configuration manuelle pour chaque device.



**Figure : Gestion des Device-profiles dans ChirpStack**

* 1. **Structure des Device-profiles**

Le tableau extrait de l’interface ChirpStack présente deux colonnes principales :

* **Name** : Identifie le profil (ex: device\_profile\_abp, RAK8111-Profile).
* **Network Server** : Indique le serveur LoRaWAN associé (ici, build\_in\_ns pour tous).
  1. **Types de Profils et Cas d’Usage**

1. **Profils ABP (Activation By Personalization)**
   * Utilisés pour une activation sécurisée hors réseau (clés pré-configurées).
   * Exemple : device\_profile\_abp.
2. **Profils OTAA (Over-The-Air Activation)**
   * Permettent une activation dynamique via le réseau (méthode plus flexible).
   * Exemple : device\_profile\_otaa.
3. **Profils Spécifiques (ex: RAK8111)**
   * Optimisés pour des modules particuliers (fréquences, puissances, etc.).

Pour ajouter un appareil RAK8111 :

1. Sélectionner RAK8111-Profile.
2. Le device hérite automatiquement des paramètres (fréquence, méthode d’activation, etc.).
3. Le trafic est géré par le build\_in\_ns (serveur intégré).
   1. **Configuration des périphériques, applications et analyse des trames montantes dans ChirpStack**

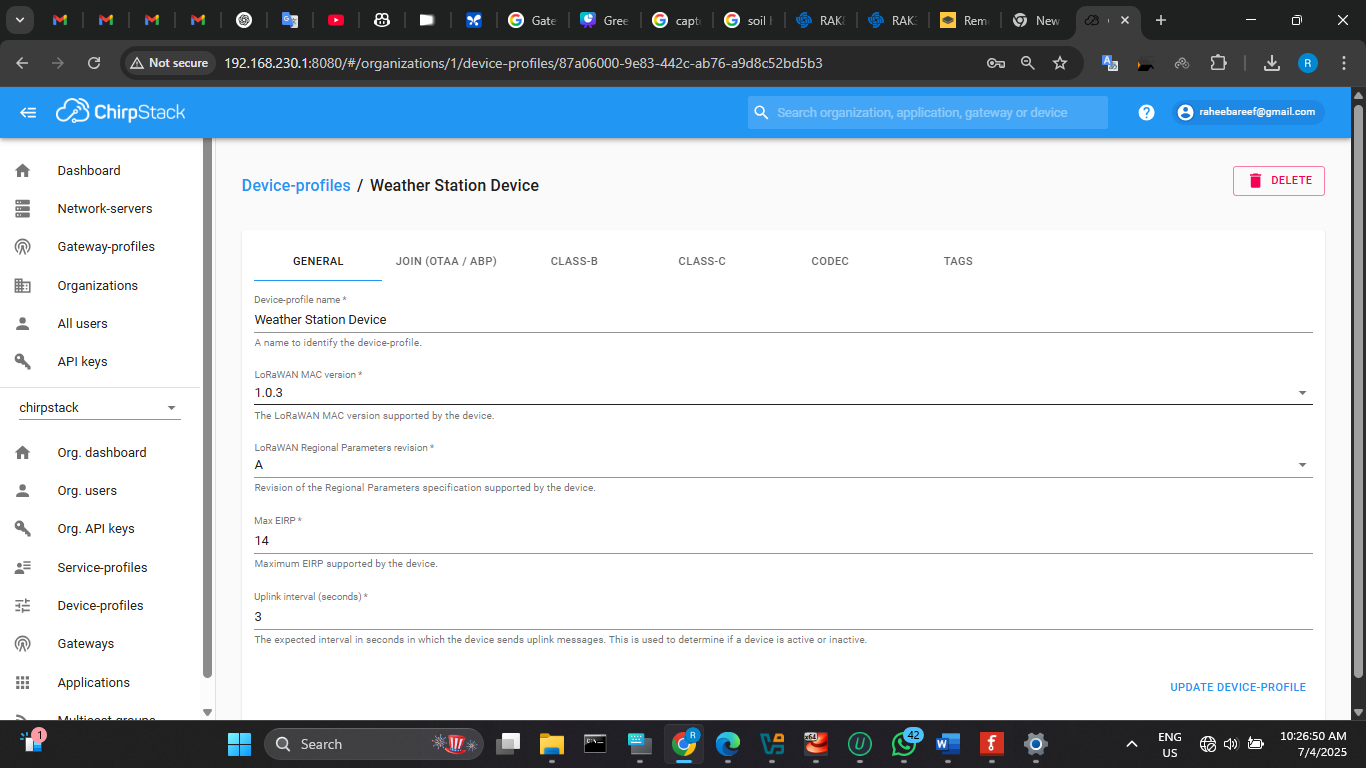
Cette section présente la configuration principale des périphériques LoRaWAN dans l’interface ChirpStack. Elle inclut également une analyse succincte des échanges de données entre les capteurs et le réseau.

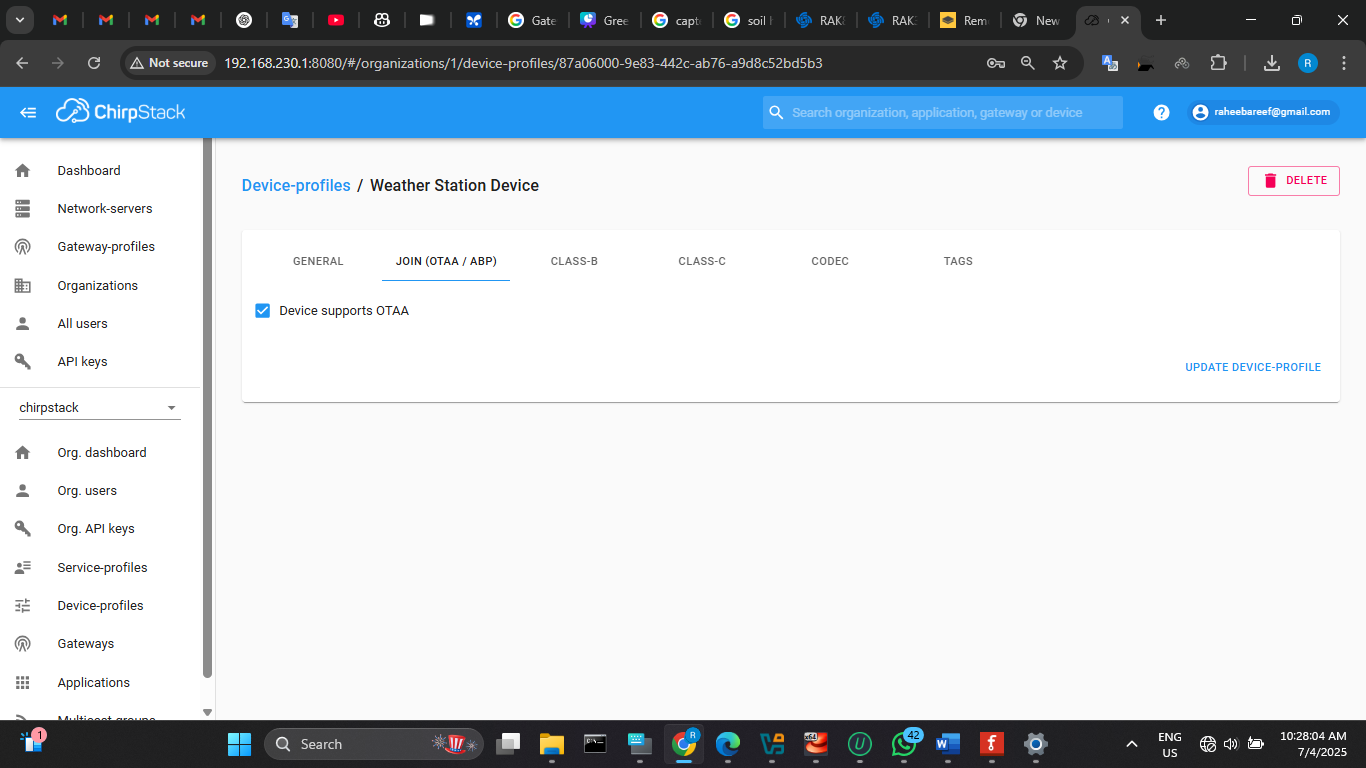
* 1. **Profil du périphérique (Device-profile)**

Le profil RAK8111-Profile définit les paramètres généraux utilisés par les périphériques RAK811 dans le réseau. Ces paramètres incluent :

* Version LoRaWAN : 1.0.3
* Puissance maximale d’émission : 14 dBm
* Intervalle entre deux messages montants : 3 secondes
* Mode d’activation : OTAA activé
* Classe C : activée

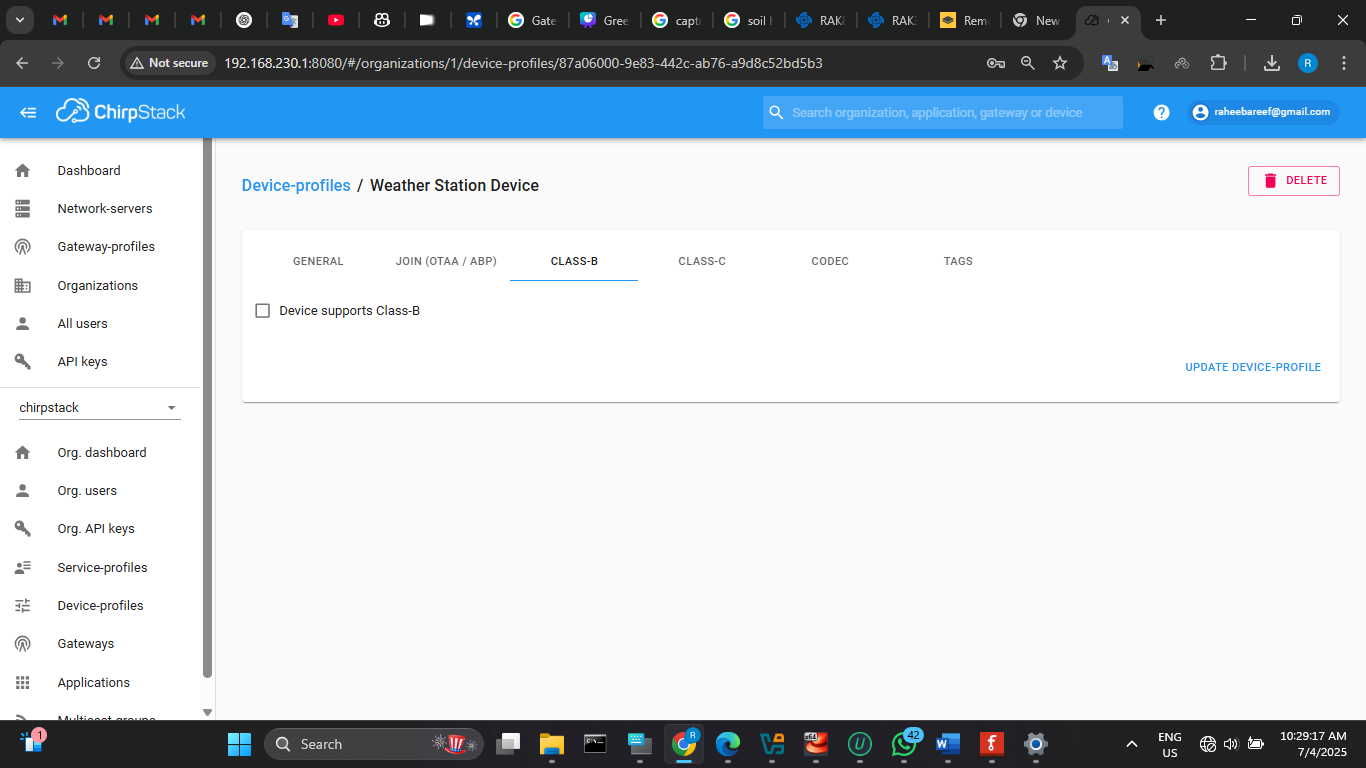
Ces réglages permettent au périphérique de fonctionner conformément aux normes LoRaWAN.





**Figure : Activation de OTAA**





**FIGURE : Activation de Classe C et désactivation de classe B**

* 1. **Périphérique (Device)**

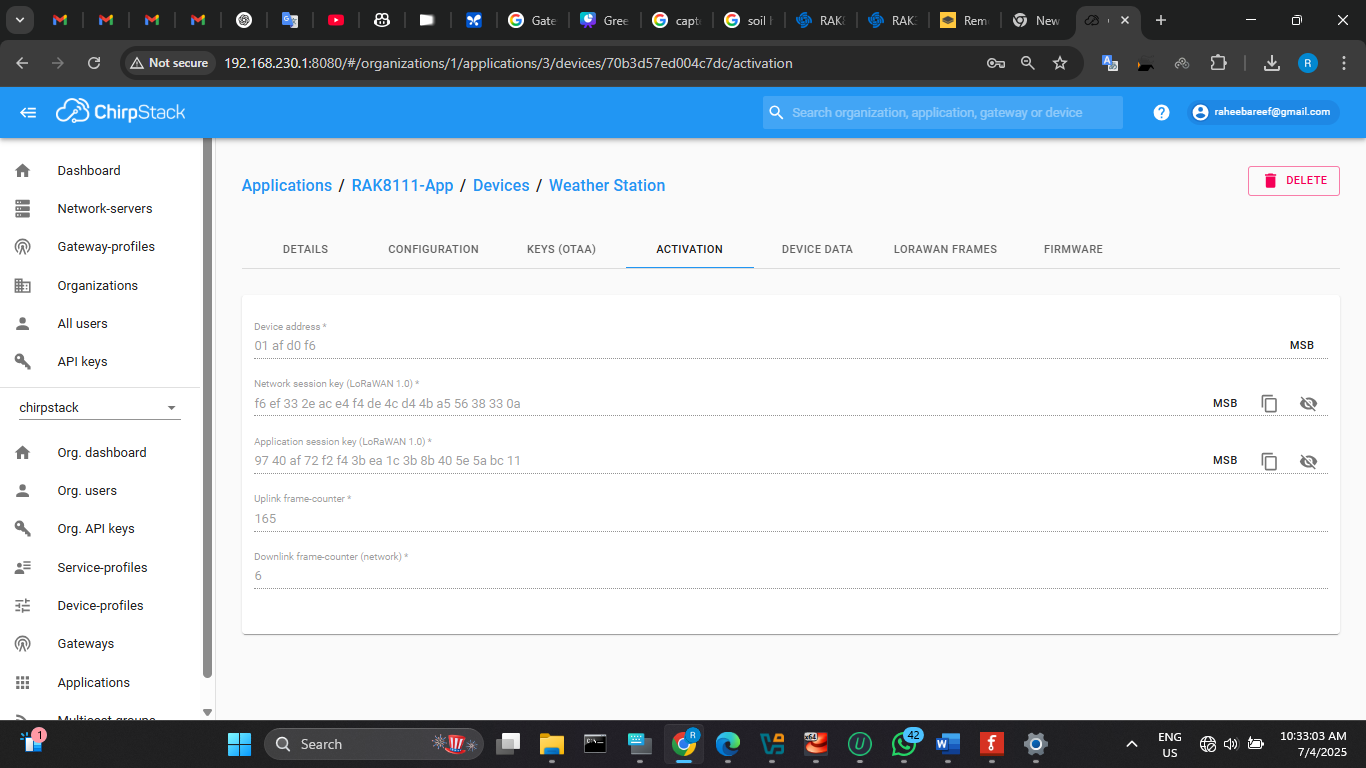
Le périphérique étudié porte le nom RAK8111-Node-01. Il utilise le profil décrit ci-dessus et est associé à l’application RAK8111-App.

**Clés de sécurité (OTAA) :**

* DevEUI : 01 77 4b 95
* NwkSKey et AppSKey : clés générées après l’activation OTAA
* Le compteur de trames montantes indique 20

La validation des compteurs de trames est active, ce qui protège contre les attaques de type "rejeu".



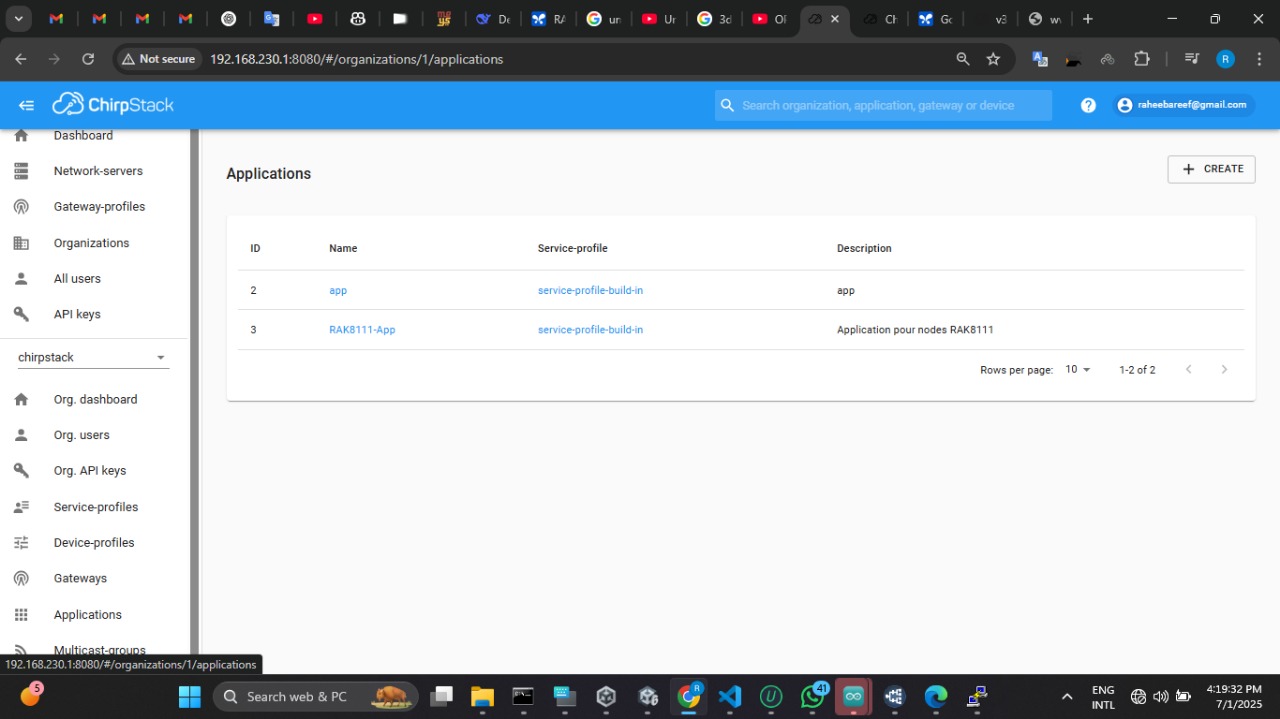


* 1. **Application**

Deux applications sont configurées :

* app : application générique
* RAK8111-App : dédiée aux périphériques RAK811

L’application RAK8111-App est liée au périphérique RAK8111-Node-01. Cela permet de regrouper les données provenant de ce capteur et de les traiter de manière organisée.



* 1. **Analyse des trames montantes (Uplink)**

L'analyse des trames montantes permet de comprendre les paramètres techniques et les données transmises par le périphérique vers la passerelle. Voici les détails observés :

a. Informations générales sur la trame montante

* Type de message : UnconfirmedDataUp
* Adresse du périphérique (DevAddr) : 01774b95
* Numéro de compteur de trame montante (fCntUp) : 3
* Port : 1
* Statut de CRC : CRC\_OK (indique que la trame a été correctement reçue sans erreur)

b. Données brutes (Payload)

* Données brutes (bytes) : EREHzaiivrsDUwAuK3CK6pNgEzM4hPdSMKozzbhkNeXDXQHJUpo64N5T2UTcOc5IJJRj
* Code MIC (Message Integrity Check) : 849b1c07

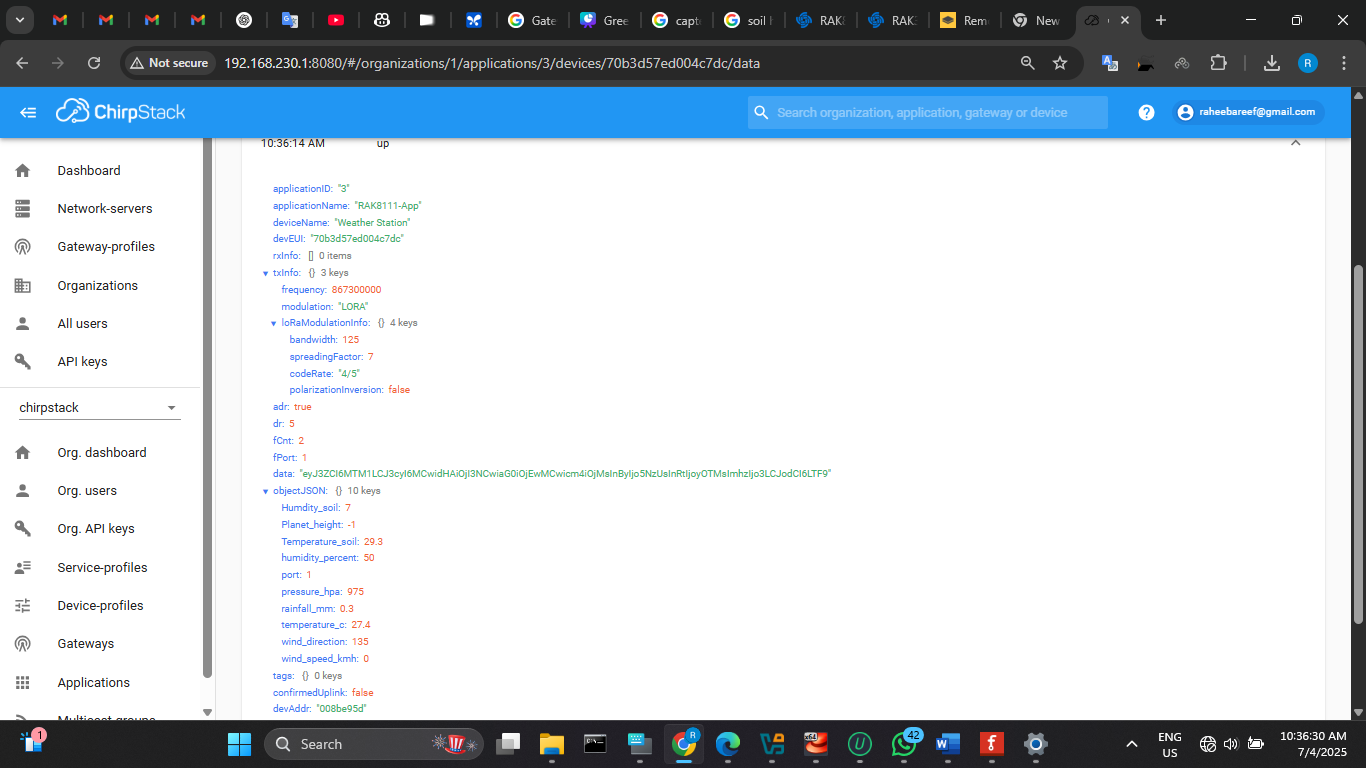
c. Informations sur la réception (rxInfo)

* ID de la passerelle : b827ebfffe117c1e
* SNR (Signal-to-Noise Ratio) : 9.2
* Canal utilisé : 1
* Fréquence : 868300000 Hz
* Modulation : LoRa
* Largeur de bande (bandwidth) : 125 kHz
* Facteur de diffusion (spreadingFactor) : 7
* Taux de codage (codeRate) : 4/5

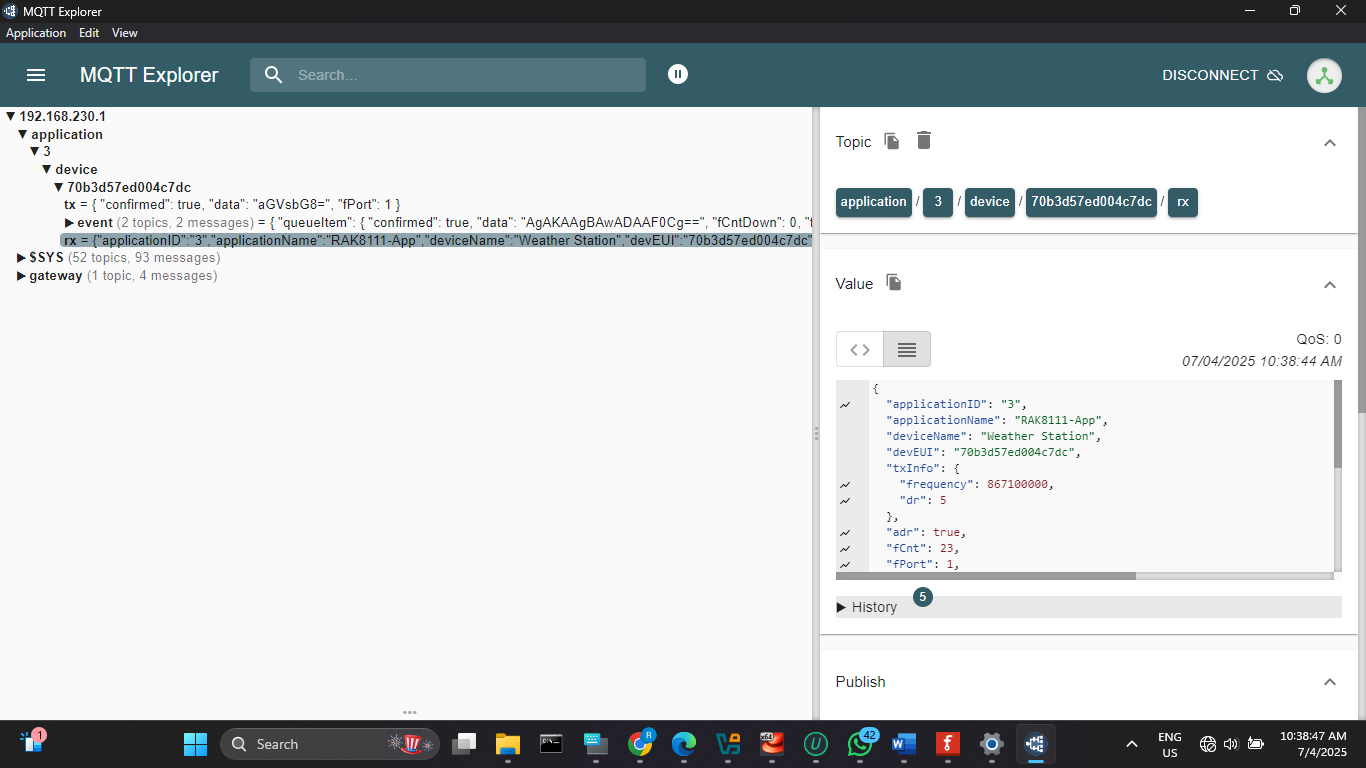
d. Informations sur la transmission (txInfo)

* Antenne utilisée : 0
* Position GPS de la passerelle :
  + Latitude : 34.03508
  + Longitude : -4.97324
  + Altitude : 619

Ces informations fournissent un aperçu complet des conditions de transmission et de réception de la trame montante, ce qui est essentiel pour diagnostiquer les problèmes éventuels liés à la couverture ou à la qualité du signal.



1. **Résultats**

recevoir les données à logiciel MQTT explorer

la formate de json que nous avons reçu sur MQTT explore

{

"applicationID": "3",

"applicationName": "RAK8111-App",

"deviceName": "Weather Station",

"devEUI": "70b3d57ed004c7dc",

"txInfo": {

"frequency": 868100000,

"dr": 5

},

"adr": true,

"fCnt": 40,

"fPort": 1,

"data": "eyJ3ZCI6MTM1LCJ3cyI6MCwidHAiOjI3MiwiaG0iOjEwMCwicm4iOjEwNywicHIiOjk3NSwidG0iOjI5NywiaHMiOjcsImh0IjotMX0=",

"object": {

"Humdity\_soil": 7,

"Planet\_height": -1,

"Temperature\_soil": 29.7,

"humidity\_percent": 50,

"port": 1,

"pressure\_hpa": 975,

"rainfall\_mm": 10.7,

"temperature\_c": 27.2,

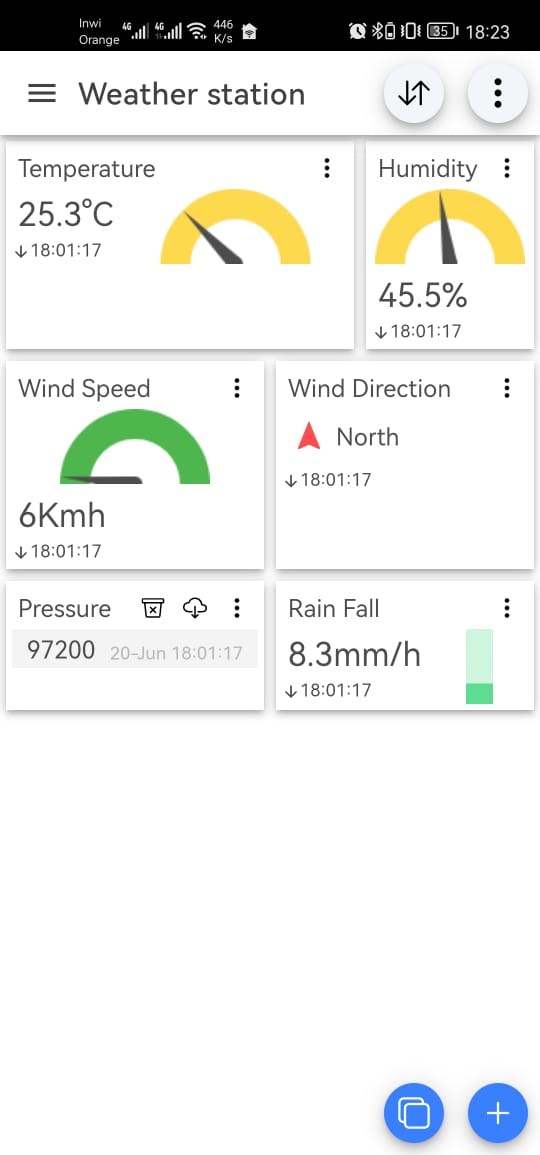
"wind\_direction": 135,

"wind\_speed\_kmh": 0

}

}

▶ History32

Affichage de résultat sur application mobile.