

TP ARNG

Contexte À l'issue de [tp-mqtt.typ], vous disposez déjà d'une chaîne Mosquitto → capteur simulé Python → API FastAPI → InfluxDB. Ce simulateur n'abordait toutefois pas les spécificités LoRa/LoRaWAN (activation ABP/OTAA, Data Rate, Spreading Factor, contraintes radio). Ce TP capitalise sur les acquis MQTT/IP pour les transposer dans une stack LoRaWAN/ChirpStack. Nous allons basculer vers le simulateur LWN-Simulator afin d'émuler rapidement des devices longue portée, tout en réutilisant les patterns de topics, de schéma de données et de supervision vus précédemment.

Supports

- Docker compose fourni en annexe et fichiers hérités du premier TP

Transition depuis TP MQTT

Ce qui change

- La couche radio devient LoRaWAN, pilotée par ChirpStack + LWN-Simulator.

Rappel – Préparation

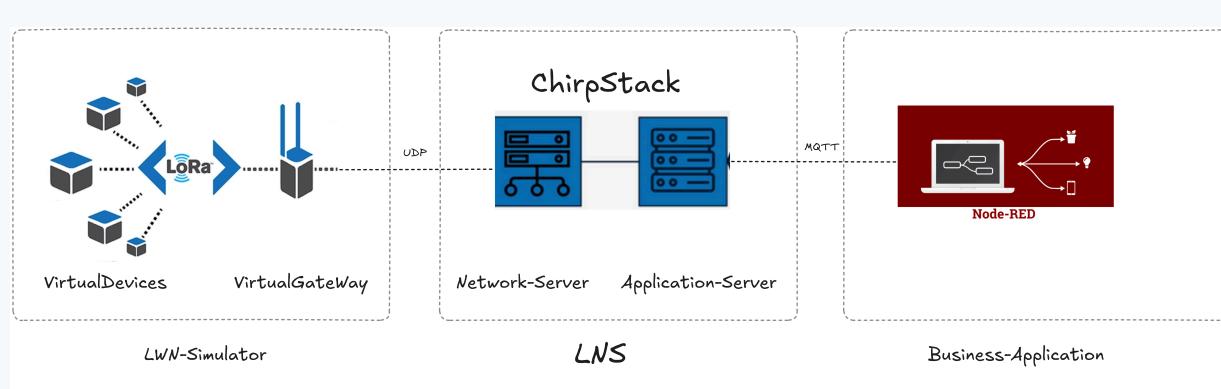
Stack fournie

- Un docker compose enrichi de celui du TP MQTT est livré (Mosquitto custom, InfluxDB) et complété par ChirpStack, LWN-Simulator, Node-RED, Grafana.
- Tous les services exposent leurs ports sur localhost (1883, 8000, 8080, 8086, ...). Lancez “docker compose up –build” depuis la racine du dépôt.

Prerequisites

- Instance ChirpStack (cloudbot ou locale) avec Gateway Bridge, Network Server et Application Server opérationnels.

Infrastructure visée



Émulation devices & gateways avec LWN-Simulator reposant sur Semtech UDP Packet Forwarder.
Possibilité d'adapter le code à une gateway physique (ex: Seeed WM1302 + Raspberry Pi4).

Pont avec TP MQTT

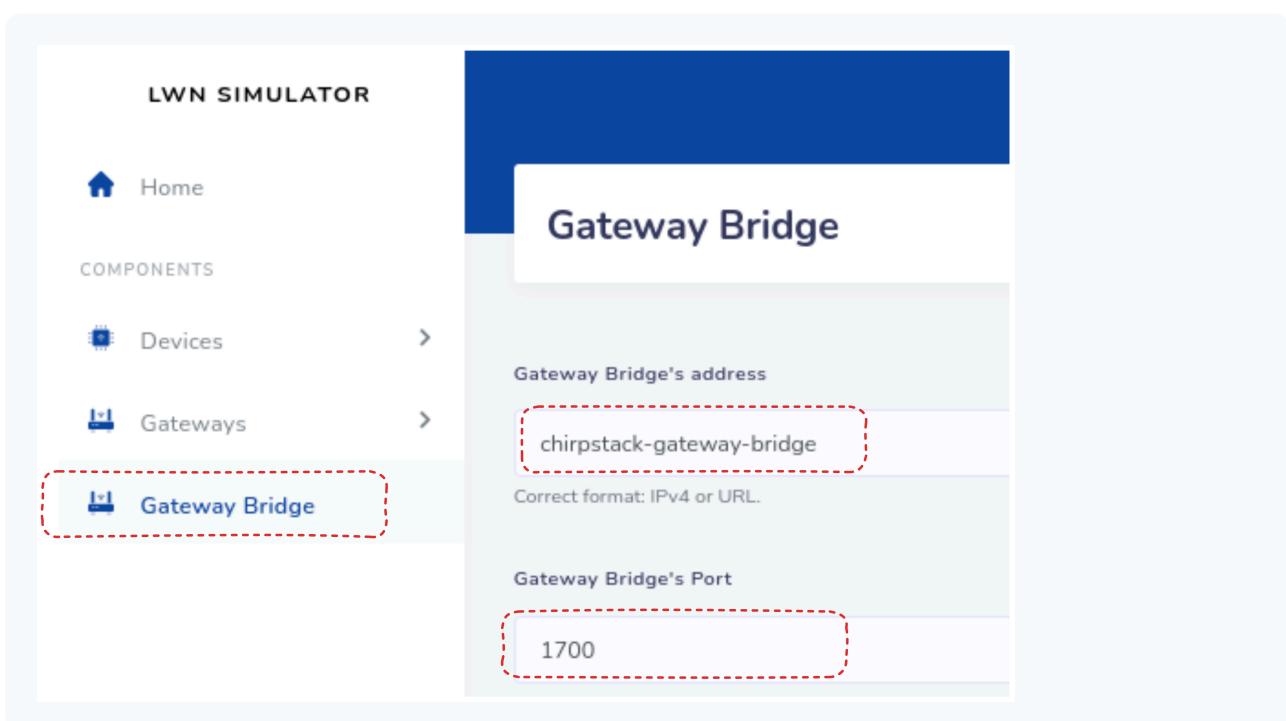
- Faites le parallèle entre le broker Mosquitto du TP précédent et le Gateway Bridge ChirpStack.

Challenge 0 – LWN-Simulator

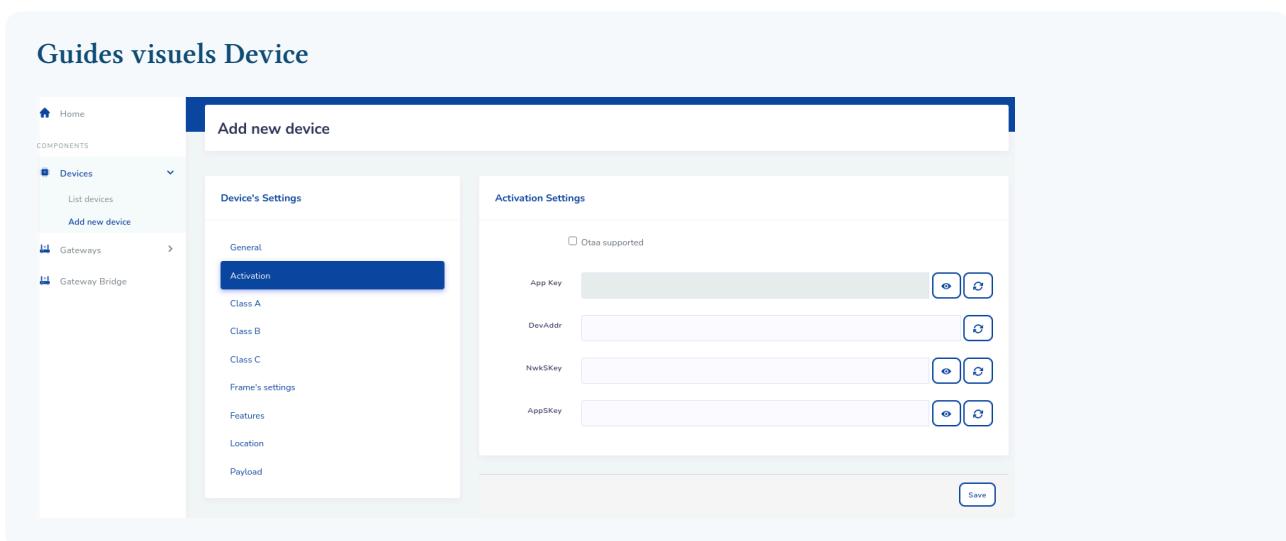
Objectifs

- Configurer un Gateway Bridge.
- Déployer trois dispositifs : dev-temp-nord, dev-temp-sud et dev-temp-centre.
- Déclarer trois passerelles virtuelles : Gw1, Gw2 et Gw3.
- Accéder à l'interface LWN-Simulator via <http://localhost:8000/> (les services sont exposés localement par Docker compose).

Guides visuels Bridge



The screenshot shows the LWN SIMULATOR interface. On the left, there's a sidebar with 'Home', 'Components' (Devices, Gateways, Gateway Bridge), and a 'Gateway Bridge' section highlighted with a red dashed box. The main area is titled 'Gateway Bridge' and contains fields for 'Gateway Bridge's address' (set to 'chirpstack-gateway-bridge') and 'Gateway Bridge's Port' (set to '1700'). Both fields have red dashed boxes around them.



The screenshot shows the 'Add new device' configuration page. The left sidebar shows 'Home', 'Components' (Devices selected), 'List devices', 'Add new device', 'Gateways', and 'Gateway Bridge'. The main panel has tabs for 'Device's Settings' (General, Activation selected) and 'Activation Settings'. Under 'Activation Settings', there are fields for 'Otaa supported' (unchecked), 'App Key' (with generate and copy buttons), 'DevAddr' (with copy button), 'NwkSKey' (with copy button), and 'AppSKey' (with copy button). A 'Save' button is at the bottom right.

Consignes

- Générez les DevEUI / clés ABP via les boutons prévus par le simulateur.
- Configurations attendues côté device :
 - General settings : renseigner Name, DevEUI (copié dans ChirpStack), Description optionnelle.
 - Activation : renseigner DevAddr, NwkSKey, AppSKey (boutons générateurs disponibles).
 - Frame settings : région EU868, Data Rate adapté (DR5 par ex.), Spreading Factor cohérent (SF7 / SF8) selon vos tests.
 - Payload : définir l'intervalle d'uplink (ex : 10s) et le type (temperature). Vous pouvez reprendre à l'identique le JSON produit par SensorSimulator (timestamp, temp, hum, alert_level, version) pour rester compatible avec collector.py.
- Configurez les devices en zone EU, activation ABP, payload temperature pour chacun des dev-temp-*.

- Après configuration, cochez Active puis Save.

Challenge 1 – Configuration ChirpStack

Device Profile

- Classe A, région EU868, sans OTAA.

Gateways

- Déclarer Gw1, Gw2 et Gw3 avec les paramètres générés (EUI / clés) et vérifier l'association au Gateway Bridge.

Applications & Devices

- Créer une application app-temp regroupant les trois devices (dev-temp-nord, dev-temp-sud, dev-temp-centre).
- Associer chaque device virtuel à l'application et vérifier la remontée des uplinks.

Simulation

- Démarrer la simulation dans LWN-Simulator.
- Dans ChirpStack, vérifiez la réception des uplinks (payload, paramètres radio, gateway concernée) et analyser les JSON publiés.
- Tant que Challenge 0 ou 1 n'est pas validé, n'entamez pas la suite.

Challenge 2 – Node-RED, souscription MQTT

Objectif

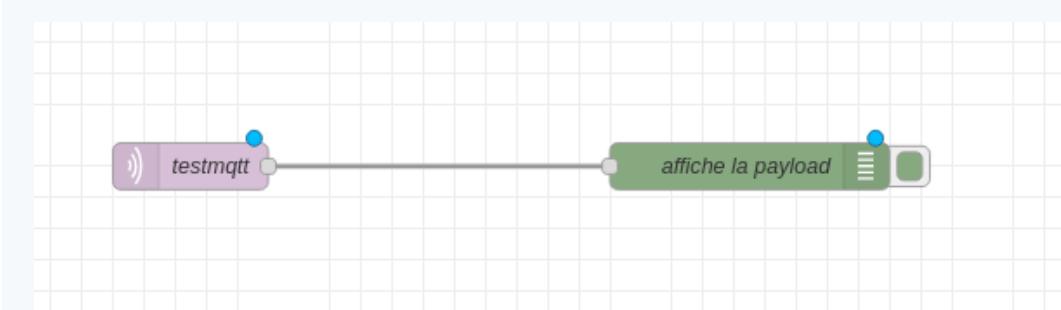
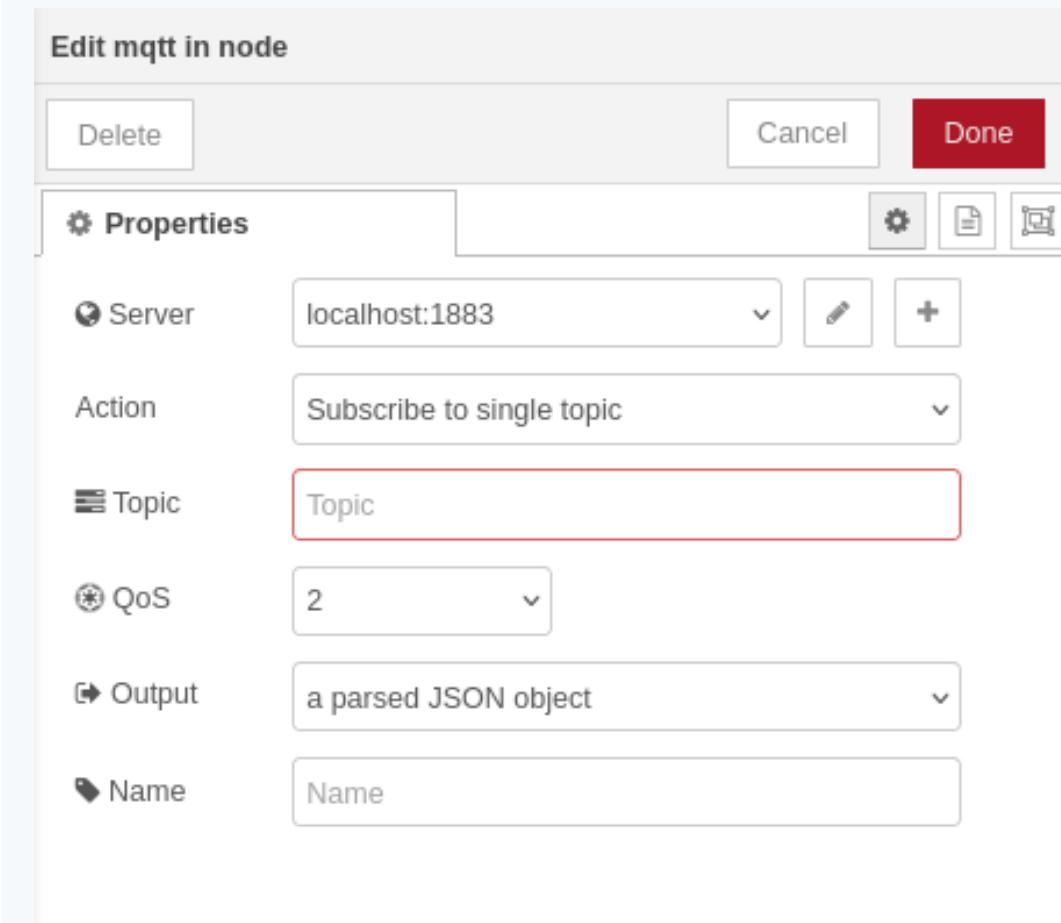
- Souscrire aux uplinks (topics MQTT) à partir du broker utilisé par ChirpStack.
- Visualiser les messages via un flow Node-RED minimal, identique à celui employé dans le TP MQTT (mqtt in → json → debug).

Étapes

1. Ajouter un mqtt in dans Node-RED et configurer le broker (IP + port).
2. Repérer les topics via docker compose logs -f <stack> et utiliser les jokers (+) pour capter plusieurs devices.
3. Réimporter/adapter votre noeud debug existant ou un sous-flow exporté depuis le TP précédent.

4. Observer les messages dans la console debug après quelques secondes.

Illustrations configuration mqtt et flow de debug



Challenge 3 – Décodage température

Tâche Transformer le message MQTT pour n'afficher que la valeur de température dans Node-RED, comme vous l'avez fait pour SensorSimulator.

Indications

- Utilisez un function node pour parser le JSON, extraire temperature
- Redirigez vers un debug node configuré sur complete msg.

Flow attendu

