Morou Ganda Yahaya
y.morouganda.35116@gmail.com
EMIG
Génie Informatique

Travaux Dirigés/Pratiques IoT : 2025-2026

I. Node-Red

1. Présentation

Node-Red est un langage de développement graphique créé par IBM, au début de l'année 2013. Son objectif est de relier des équipements matériels, des API et des services en ligne pour faciliter l'intégration des IoT (Internet of Things). Node-Red est un outil puissant pour construire des applications de l'Internet des Objets en mettant l'accent sur la simplification de la programmation qui se fait grâce à des blocs de code prédéfinis, appelés *nodes* pour effectuer des tâches. Il utilise une approche de programmation visuelle qui permet aux développeurs de connecter les blocs de code ensemble. Les nœuds connectés, généralement une combinaison de nœuds d'entrée, de nœuds de traitement et de nœuds de sortie, lorsqu'ils sont câblés ensemble, constituent un *flow*. Node-Red est construit sur Node.js, tirant pleinement parti de son modèle non bloquant piloté par les événements. Node-Red utilise plusieurs types de blocs de nœuds :

- Les blocs **communs** sont les plus fréquemment rencontrés
- Les blocs **fonctions** sont programmables
- Les blocs **réseaux** permettent la communication avec le réseau
- Les blocs **séquences** sont des blocs qui facilitent le traitement de données (joindre, séparer, trier…)
- Les blocs **d'analyses** travaillent sur la conversion des formats de données (CSV, JSON, XML…)
- Les blocs de **stockage** servent pour la gestion des fichiers

2. Installation

L'environnement Node-Red s'appuie sur le gestionnaire de module NPM et Node.JS pour fonctionner :

- Après l'installation de Node.JS, l'installation de Node-Red se fait avec l'instruction *npm install -g --unsafe-perm node-red*.
- L'exécution de la commande Node-Red lance un serveur sur le port local 1880 : http://127.0.0.1:1880
- L'arrêt de Node-Red est commandé par la séquence de touches CTRL+C

II. TD

Exercice 1 : Message Queuing Telemetry Transport (MQTT : installation et prise en main)

MQTT est un protocole de communication léger conçu pour transmettre des données entre appareils (IoT, capteurs, objets connectés), fonctionner avec peu de bande passante (idéal pour les réseaux instables) et échanger des messages en temps réel via un système Publish/Subscribe.

Comment ça marche ?

- Publisher (Émetteur) : Envoie des messages sur un topic (ex : maison/cuisine/température).
- Subscriber (Receveur) : Reçoit les messages s'il est abonné au topic.
- Broker MQTT (Serveur) : Centralise les messages.
- Pour utiliser MQTT en local il faut installer le logiciel ***mosquito***.

L'objectif de ce TP consiste à installer MQTT, à s'y connecter et à envoyer le message « Bonjour MQTT ».

1. Installer, ajouter le dossier aux variables d'environnement et vérifier que Mosquitto est bien installé : *mosquitto -v*
2. Créer un Flow MQTT dans Node-RED avec comme objectif un étudiant (Publisher) envoie un message ("Bonjour MQTT") et un autre étudiant (Subscriber) reçoit le message.
3. Configuration :
   - Ajoutez un nœud "MQTT out" (Publisher)
     - Glissez-déposez un nœud `mqtt out` depuis la palette de gauche.
     - Double-cliquez dessus pour le configurer :
     - Server : localhost (ou `127.0.0.1`)
     - Port : `1883`
     - Topic : `tp/etudiant1` (c'est comme un "sujet de discussion")
     - Cliquez sur "Add" puis "Done".
   - Ajoutez un nœud "Inject" (pour envoyer un message)
     - Glissez-déposez un nœud `inject`.
     - Liez-le au `mqtt out` (faites un clic-glissé entre les deux).
     - Double-cliquez sur `inject` et choisissez :
     - Payload : `string` → `"Bonjour MQTT !"`
   - Ajoutez un nœud "MQTT in" (Subscriber)
     - Glissez-déposez un nœud `mqtt in`.
     - Configurez-le avec :
     - Server : Même broker (`localhost :1883`)
     - Topic : `tp/etudiant1` (doit correspondre au Publisher)
     - Liez-le à un nœud `debug` (pour voir les messages reçus).

   - Cliquez sur "Deploy" pour activer le flow.

Exercice 2 : MQTTX

1. Faites une brève description de MQTTX
2. Installez MQTTX, créez une connexion vers localhost :1883, abonnez-vous au topic 'tp/etudiant1' et afficher le message "Bonjour MQTT !".
3. Envoyer le message « Bonjour node-red, ici MQTTX » pour qu'il s'affiche dans node-red.

Morou Ganda Yahaya
y.morouganda.35116@gmail.com

Exercice 3 : Surveillance de Température avec Dashboard

L'objectif de ce TP est de simuler un capteur de température et l'afficher dans un Dashboard qui est une interface web. Pour l'installer, dans le **menu**, cliquez sur **Manage Palette** puis recherchez **dashb** dans la catégorie de nodes à installer. Une nouvelle catégorie est ensuite disponible.

Nœuds nécessaires :

Inject (chaque 3 sec) → Function (msg.payload = {temperature: (Math.random() * 30 + 20).toFixed(1)} return msg;). Liez à un MQTT out (topic : tp3/temperature).

Pour le dashboard, ajoutez : Chart (group: "Température", label: "Température en direct"), Gauge (min: 20, max: 50).

1. Réaliser cet exercice.
2. Ajoutez un seuil d'alerte (si temperature > 30, changez la couleur du gauge en rouge)

Exercice 4 : API REST

L'objectif de ce TP est de récupérer les données météo d'une ville (température, humidité, conditions) via une API REST (OpenWeatherMap) et les afficher dans un Dashboard Node-RED.

Prérequis : node-red-dashboard (pour l'interface), node-red-contrib-http-request (pour les requêtes API) et utiliser ***open-meteo*** pour avoir la météo en temps réel.

Exercice 5 : Surveillance de la Qualité de l'Air en Ville

Créer un Dashboard surveillant la qualité de l'air (CO2, particules) via des capteurs virtuels dans une ville donnée (exemple Niamey).

Outils : Open-Meteo Air Quality ou IQAir, Simulation de capteurs avec MQTTX et utilisez un nœud ui_map pour afficher la localisation simulée.

Tâches : Récupérer les données via API REST, simuler des alertes si CO2 > 800 ppm. Afficher les données sur un Dashboard avec cartes et graphiques historiques.

Exercice 6 : Gestion Intelligente d'Éclairage Urbain

Simuler un système d'éclairage urbain ajusté à la luminosité et à l'occupation. Dans cet exercice, on essaie d'allumer des lampadaires (virtuels) si luminosité < 20 lux et mouvement détecté.

Outils : Capteur de lumière virtuel : Génération aléatoire de valeurs (0-100 lux), détection de mouvement : MQTTX publiant sur streetlight/motion, historiser les états dans InfluxDB (optionnelle). Utilisez ui_led pour les lampadaires virtuels.

Exercice 7 : Suivi de Livraisons par GPS

Développer un système de suivi en temps réel de camions de livraison en utilisant des données GPS simulées, avec alertes pour les retards et optimisation des trajets.

Outils à Utiliser

- **Module node-red-contrib-web-worldmap** (cartes interactives).
- **API OpenRouteService** (gratuite pour les itinéraires).
- **MQTTX** (simulation des données GPS des camions).
- **Fichier GeoJSON** (pour les trajets prévus).
- **Autres…**

Tâches à Réaliser

1. **Simulation des Camions** :
    - Générer des positions GPS aléatoires le long d'un trajet (ex: Niamey → Maradi).
    - Publier ces données via MQTT (camion1/gps).
2. **Intégration Cartographique** :
    - Afficher les camions comme des marqueurs sur une carte.
    - Dessiner l'itinéraire optimal (via API OpenRouteService).
3. **Détection des Retards** :
    - Comparer la position réelle avec l'itinéraire prévu.
    - Envoyer une alerte si le camion s'écarte de > 5 km.
4. **Dashboard** :
    - Carte interactive avec marqueurs colorés (vert = à l'heure, rouge = retard).
    - Tableau de bord des livraisons (temps restant, vitesse moyenne).