Université Mohammed Premier
Faculté des Sciences Oujda

Rapport de Mini-Projet IoT

Système de Surveillance Sanitaire Multi-Patients

Développement d'une plateforme de monitoring temps réel basée sur une architecture micro-services

Réalisé par :
Anass El Amrany

Encadré par :
Prof. Belouch Mustapha

Année Universitaire 2024 - 2025

Table des matières

1 Introduction Générale **3**
1.1 Contexte du Projet 3
1.2 Objectifs du Mini-Projet 3

2 Analyse et Architecture **4**
2.1 Architecture Technique 4
2.2 Les Outils Utilisés 4
2.2.1 Docker et Docker Compose 4
2.2.2 MQTT et Mosquitto 5
2.2.3 Node-RED 5
2.2.4 InfluxDB et Grafana 5

3 Réalisation et Développement **6**
3.1 Mise en place de l'Infrastructure 6
3.2 Simulation des Patients (Python) 8
3.3 Traitement des Données avec Node-RED 9

4 Résultats et Dashboard **10**
4.1 Présentation du Tableau de Bord 10
4.2 Le Système d'Alertes 11

Conclusion Générale **12**

Table des figures

2.1 Schéma global de l'architecture IoT 4

3.1 Flux de traitement Node-RED complet 9

4.1 Vue globale du Dashboard Médecin 10
4.2 Tableau d'historique des alertes critiques 11

Chapitre 1

Introduction Générale

1.1 Contexte du Projet

Aujourd'hui, l'Internet des Objets (IoT) est très important dans le domaine médical. On parle souvent de "e-Santé". Le problème principal dans les hôpitaux est le manque de personnel pour surveiller tous les patients en même temps.

L'idée de ce projet est de créer un système automatique. Ce système permet aux médecins de surveiller les signes vitaux (comme le cœur, la tension ou la température) à distance. Cela aide à détecter les problèmes graves très vite, sans être physiquement à côté du patient.

1.2 Objectifs du Mini-Projet

Notre travail consiste à développer une solution complète, du capteur jusqu'à l'écran de contrôle. Les objectifs principaux sont :

1. **La Simulation :** Créer un programme capable d'imiter le comportement de plusieurs patients (3 à 5) avec des données réalistes incluant la tension artérielle.
2. **La Communication :** Utiliser un protocole rapide et léger pour envoyer les données. Nous avons choisi MQTT.
3. **Le Stockage :** Garder un historique des mesures dans une base de données spéciale pour le temps (Time-Series).
4. **La Visualisation :** Créer un tableau de bord (Dashboard) facile à lire pour le médecin, avec des alertes visuelles.

Chapitre 2

Analyse et Architecture

2.1 Architecture Technique

Pour réaliser ce projet, nous avons utilisé une architecture en "micro-services". Cela veut dire que chaque partie du projet est indépendante. Nous avons utilisé **Docker** pour isoler ces parties.

Le schéma ci-dessous montre comment les données circulent :

1. Le **Simulateur Python** génère les données (BPM, Température, SpO2, Tension).
2. Il les envoie au **Broker Mosquitto** via MQTT.
3. **Node-RED** récupère ces données, les traite et les formate.
4. Il les enregistre dans **InfluxDB**.
5. **Grafana** lit la base de données et affiche les courbes.

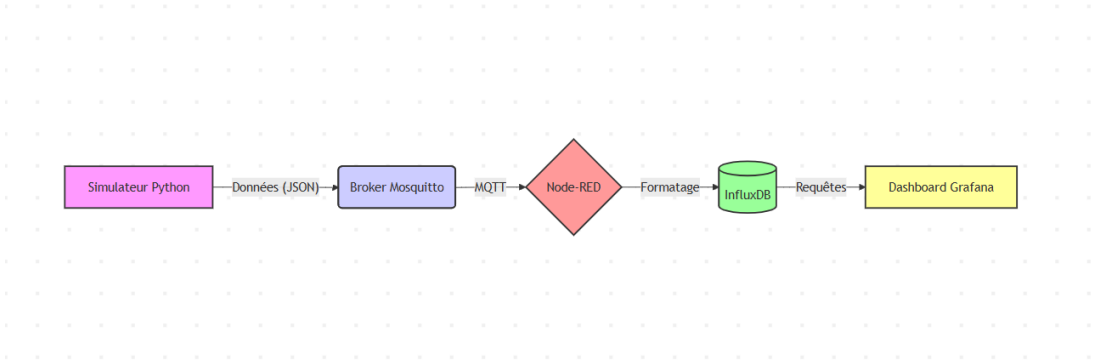

FIGURE 2.1 – Schéma global de l'architecture IoT

2.2 Les Outils Utilisés

Nous avons choisi des technologies modernes et open-source :

2.2.1 Docker et Docker Compose

Docker permet de créer des conteneurs virtuels. C'est très utile car nous n'avons pas besoin d'installer Mosquitto ou InfluxDB directement sur Windows. Avec un seul fichier `docker-compose.yml`, on lance toute l'infrastructure.

2.2.2 MQTT et Mosquitto

MQTT est le protocole standard pour l'IoT. Il est léger et consomme peu d'énergie. Mosquitto est le logiciel "serveur" (Broker) qui reçoit les messages et les distribue.

2.2.3 Node-RED

C'est un outil de programmation visuelle. Il permet de relier les services entre eux sans écrire beaucoup de code complexe. Il joue le rôle de "cerveau" dans notre système.

2.2.4 InfluxDB et Grafana

InfluxDB est parfait pour stocker des données horodatées. Grafana est l'outil le plus populaire pour créer des tableaux de bord de monitoring.

Chapitre 3

Réalisation et Développement

3.1 Mise en place de l'Infrastructure

La première étape a été de configurer le fichier `docker-compose.yml`. Ce fichier définit nos 4 services. Nous avons configuré un réseau privé nommé `iot-network` pour que les conteneurs puissent communiquer entre eux de manière sécurisée.

```yaml
version: '3.8'

services:
  # === 1. Mosquitto (Broker) ===
  mosquitto:
    image: eclipse-mosquitto:2.0
    container_name: mosquitto
    restart: unless-stopped
    ports:
      - "1883:1883"
      - "9001:9001"
    volumes:
      - ./mosquitto/config:/mosquitto/config
      - ./mosquitto/data:/mosquitto/data
      - ./mosquitto/log:/mosquitto/log
    networks:
      - iot-network

  # === 2. InfluxDB (Base de donn es) ===
  influxdb:
    image: influxdb:1.8
    container_name: influxdb
    restart: unless-stopped
    ports:
      - "8086:8086"
    environment:
      - INFLUXDB_DB=hopital_db
    volumes:
      - ./influxdb:/var/lib/influxdb
    networks:
      - iot-network

  # === 3. Node-RED (Cerveau) ===
  nodered:
    image: nodered/node-red:latest
    container_name: nodered
```

```
    restart: unless-stopped
    ports:
      - "1880:1880"
    volumes:
      - ./nodered:/data
    environment:
      - TZ=Africa/Casablanca
    networks:
      - iot-network
    depends_on:
      - mosquitto
      - influxdb

  # === 4. Grafana (Visualisation) ===
  grafana:
    image: grafana/grafana
    container_name: grafana
    restart: unless-stopped
    ports:
      - "3000:3000"
    volumes:
      - ./grafana:/var/lib/grafana
    networks:
      - iot-network
    depends_on:
      - influxdb

# === R seau D di   ===
networks:
  iot-network:
    driver: bridge
```

Listing 3.1 – Configuration Docker Compose Complète

3.2 Simulation des Patients (Python)

N'ayant pas de vrais capteurs physiques, j'ai développé un script en Python. Ce script utilise la bibliothèque `paho-mqtt`.

Le script crée 3 patients virtuels. Pour chaque patient, il génère des valeurs aléatoires pour le rythme cardiaque (BPM), la température et l'oxygène (SpO2). Nous avons ajouté la **tension artérielle** (Systolique/Diastolique) et une logique de fièvre aléatoire.

```python
import paho.mqtt.client as mqtt
import json
import time
import random

# Configuration
BROKER = "localhost"
PORT = 1883
TOPIC_BASE = "hopital/service_A/patient_"

def get_vital_signs(patient_id):
    """G  n  re  des donn es  m dicales  simul es"""
    fievre = random.randint(0, 20) == 0

    # G  n  ration  de la tension (Systolique / Diastolique)
    sys = random.randint(110, 140)
    dia = random.randint(70, 90)
    tension_str = f"{sys}/{dia}"

    data = {
        "id": patient_id,
        "bpm": random.randint(110, 140) if fievre else random.randint(60, 90),
        "temp": round(random.uniform(38.5, 40.0), 1) if fievre else round(random.uniform(36.5, 37.5), 1),
        "spo2": random.randint(94, 100),
        "tension": tension_str,
        "timestamp": int(time.time())
    }
    return data

def main():
    client = mqtt.Client()
    print("Connexion au Broker Mosquitto...")
    try:
        client.connect(BROKER, PORT, 60)
        client.loop_start()
        print("Connect  !")
    except Exception as e:
        print(f"Erreur : {e}")
        return

    patients = ["001", "002", "003"]
    # ... Boucle d'envoi ...
```

Listing 3.2 – Code de la simulation (patient_simulator.py)

3.3 Traitement des Données avec Node-RED

Node-RED est l'intermédiaire. Il reçoit le message JSON envoyé par Python. Une étape difficile a été de formater correctement les données pour InfluxDB. Nous avons dû séparer les **Tags** (pour le menu déroulant Grafana) et les **Fields** (les valeurs).

Voici le code JavaScript final utilisé dans le nœud "Function" :

```
var rawData = msg.payload;

// Conversion du texte en objet JSON si n cessaire
if (typeof rawData === 'string') {
    rawData = JSON.parse(rawData);
}

// Pr paration des champs (Valeurs pour les graphiques)
var fields = {
    bpm: parseInt(rawData.bpm),
    temp: parseFloat(rawData.temp),
    spo2: parseInt(rawData.spo2),
    tension: rawData.tension
};

// Pr paration des tags (Pour le filtrage par ID)
var tags = {
    patient_id: rawData.id
};

// Envoi vers la base de donn es au format [fields, tags]
msg.payload = [fields, tags];
return msg;
```

Listing 3.3 – Fonction Node-RED pour le formatage des données

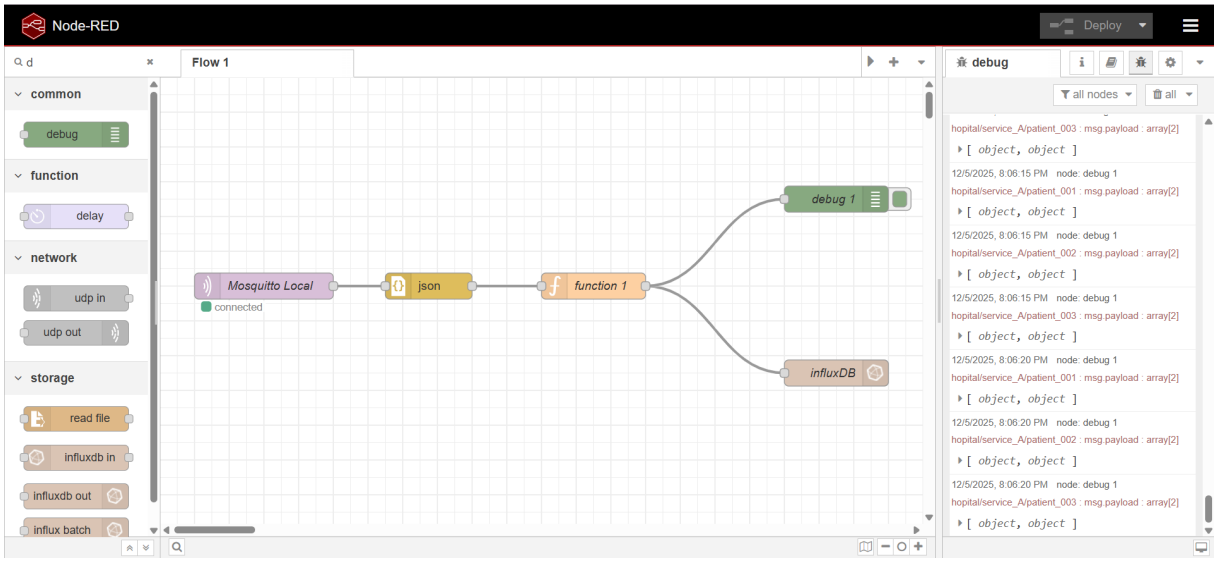

FIGURE 3.1 – Flux de traitement Node-RED complet

Chapitre 4

Résultats et Dashboard

4.1 Présentation du Tableau de Bord

L'interface finale a été réalisée avec Grafana. C'est l'outil que le médecin utilisera. Nous avons configuré une variable dynamique `$patient_id` qui permet de créer un menu déroulant. Le médecin peut choisir de voir "Tous les patients" ou un patient précis (ex : Patient 001).

Le tableau de bord contient :

- **Une jauge de température :** Elle change de couleur (devient rouge) si la température dépasse 38°C.
- **Un graphique historique :** Pour voir l'évolution du rythme cardiaque sur la dernière heure.
- **Un indicateur SpO2 :** Pour vérifier le taux d'oxygène.
- **Un tableau Tension :** Affichant la dernière valeur systolique/diastolique.

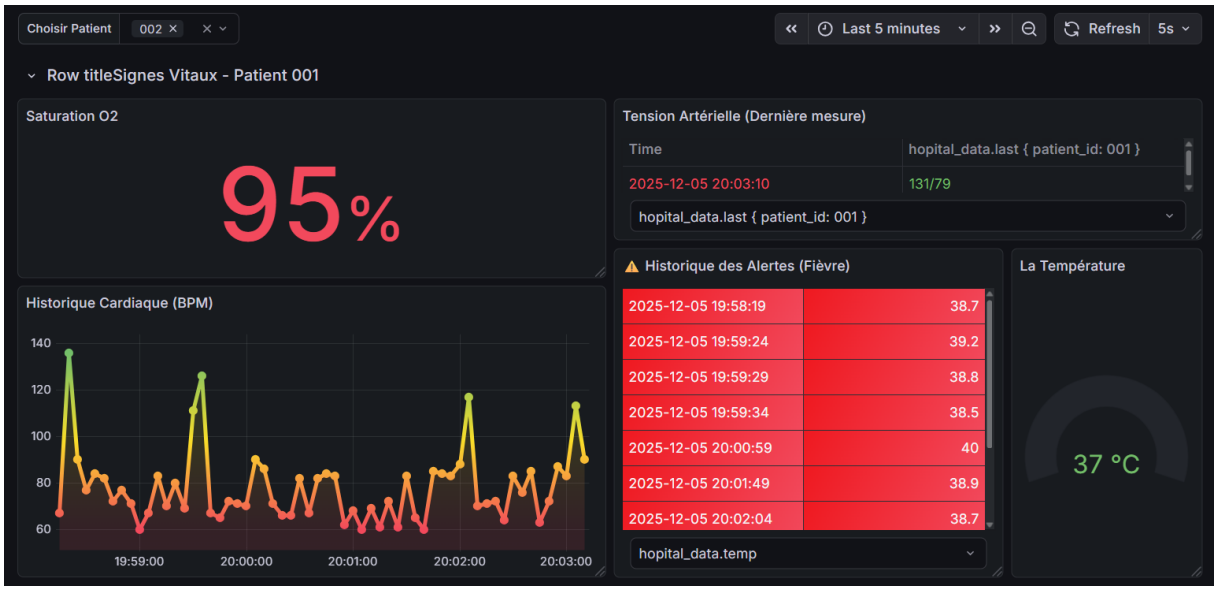

FIGURE 4.1 – Vue globale du Dashboard Médecin

4.2 Le Système d'Alertes

Un point important du cahier des charges était l'historique des alertes. Nous avons ajouté un tableau en bas de l'écran. Ce tableau affiche automatiquement une ligne rouge quand une température anormale est détectée. Cela permet de voir rapidement quel patient a eu de la fièvre et à quelle heure.

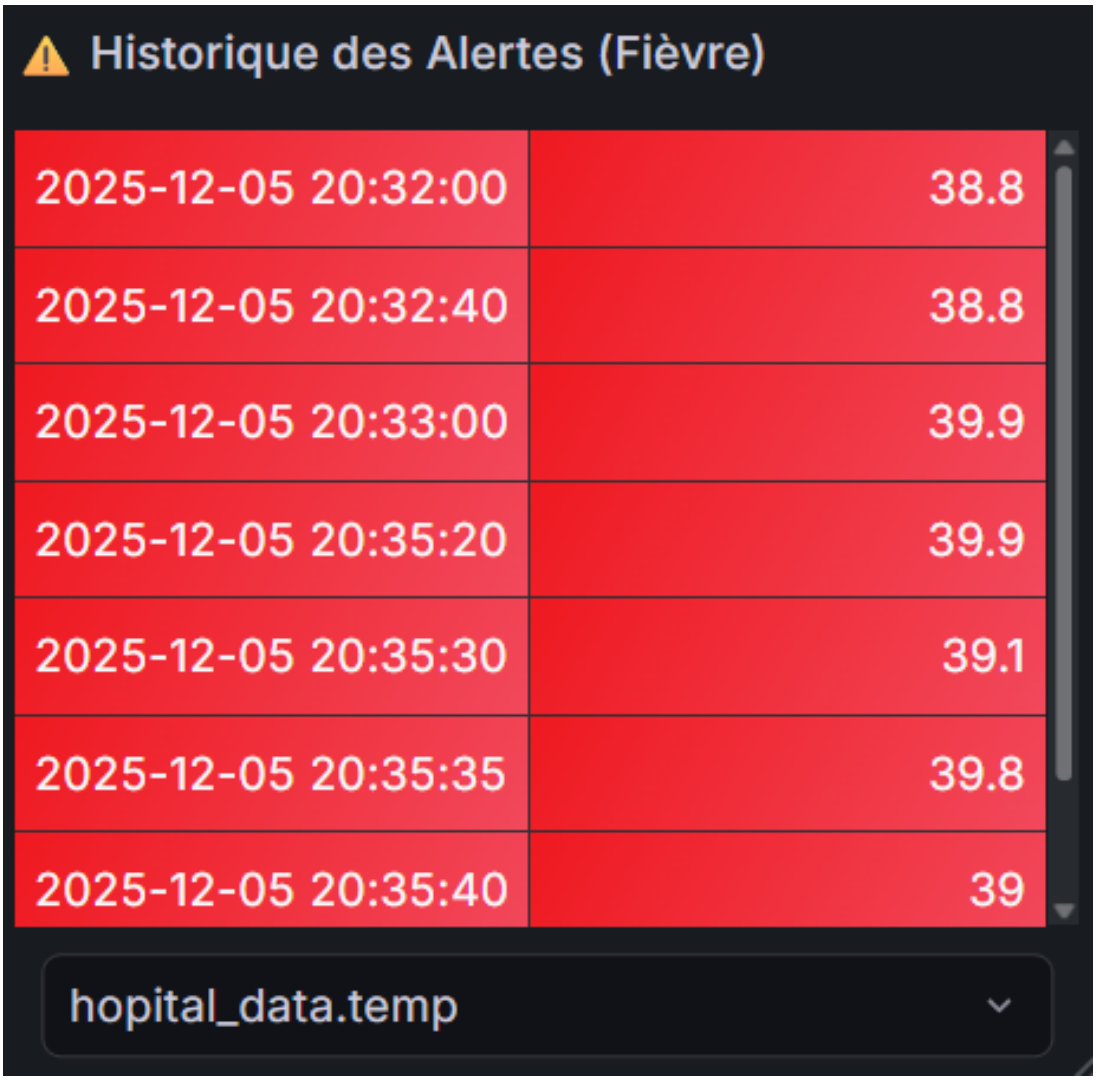

FIGURE 4.2 – Tableau d'historique des alertes critiques

Conclusion Générale

Ce mini-projet a été une excellente occasion de mettre en pratique les concepts de l'Internet des Objets (IoT) et des micro-services. J'ai réussi à construire une chaîne complète, partant de la simulation d'un capteur jusqu'à la visualisation graphique.

J'ai rencontré plusieurs difficultés techniques, notamment :

— La configuration du réseau Docker sous Windows.

— La gestion des types de données (String vs Int) dans InfluxDB qui causait des erreurs.

— La création des variables dynamiques dans Grafana pour le menu déroulant.

Cependant, j'ai pu résoudre ces problèmes et le système final est fonctionnel et stable. Il respecte toutes les contraintes du cahier des charges.

Pour améliorer ce projet dans le futur, on pourrait ajouter un système de notification par SMS pour alerter les médecins directement sur leur téléphone, ou sécuriser la connexion avec des mots de passe et du cryptage SSL.