Rapport Technique

Système IoT - Smart Multi-Node

Par Espérance AYIWAHOUN

Mai 2025

Table des matières

Introduction

Ce rapport détaille la conception et la simulation d'un système IoT distribué Smart Multi-Node. Le prototype vise à démontrer la collecte de données environnementales via plusieurs capteurs, leur traitement par un nœud central et la visualisation en temps réel sur un dashboard embarqué, le tout simulé dans Wokwi.

II. Choix techniques :

1. Architecture matérielle :

- **Environnement de simulation :** Wokwi, permettant d'émuler l'ensemble sans matériel physique.
- **Microcontrôleur :** ESP32, pour sa connectivité WiFi et ses capacités de calcul.

Capteurs et affichage :

- **DHT22** pour mesurer la température et l'humidité.
- **Écran OLED SSD1306** (128×64) pour l'affichage local des données.

2. Architecture logicielle :

Nœud capteur

- Framework : Arduino.

  Bibliothèques :

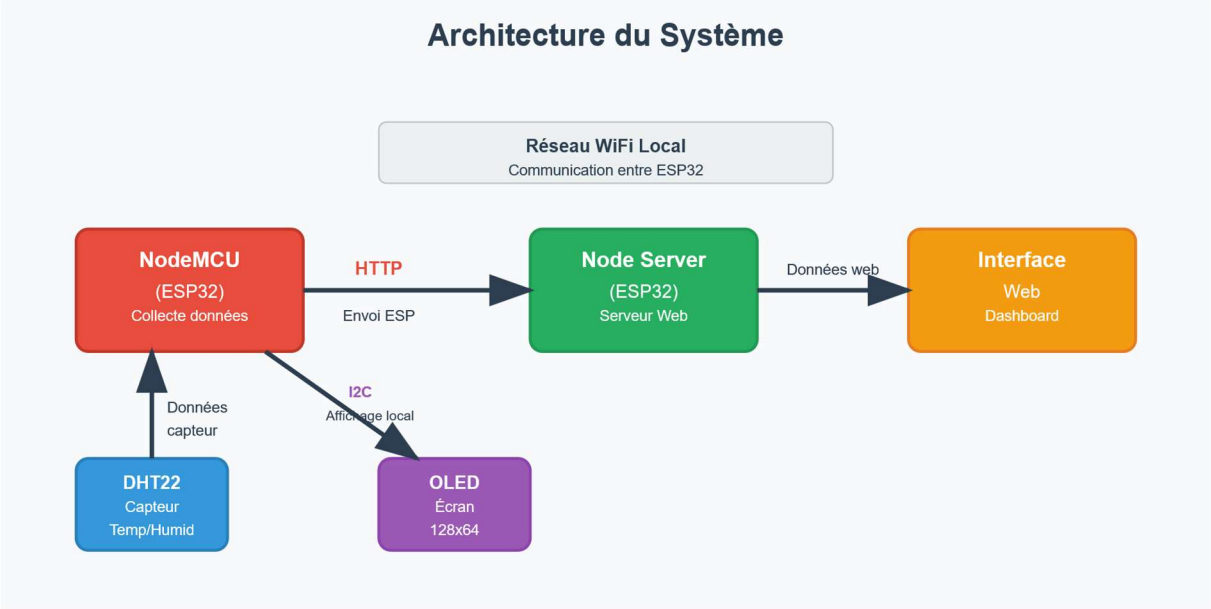
  - **DHT** pour le capteur.
  - **Wire**, **Adafruit_GFX**, **Adafruit_SSD1306** pour l'OLED.
- **Boucle principale :** lecture des capteurs toutes les 10 secondes.

- **Fonction simulateHttpPost() :** génère une trame série simulant une requête HTTP POST en JSON.

Serveur central

- **Mode WiFi** : point d'accès (*WiFi.softAP*).
- **Serveur HTTP :** léger, avec *WebServer.h.*
- **Système de fichiers :** SPIFFS, pour héberger les pages web.
- **Gestion JSON :** *ArduinoJson* pour sérialisation et désérialisation.

3. **Stockage des données**
   - **Buffer circulaire** de 50 entrées, optimisant l'usage mémoire.
   - **Calcul de statistiques** en temps réel : minimum, maximum et moyenne.
4. **Interface utilisateur**
   - **Dashboard web responsive** en HTML/CSS/JS minimaliste.
   - **Mise à jour dynamique** toutes les 2 secondes.



Architecture du système

III. Logique de communication simulée

La communication entre capteur et serveur se fait via le moniteur série, sans réseau réel.

1. **Côté nœud capteur**

   La fonction simulateHttpPost() émet :

```
POST /api/data HTTP/1.1
Host: 192.168.4.1
Content-Type: application/json
Content-Length: XX
{"temp":XX.X,"hum":XX.X,"timestamp":XXXXX}
HTTP/1.1 200 OK
```

   Les données JSON contiennent les champs : temp, hum, timestamp.

2. **Côté serveur central**

   - **Endpoint** : /api/data en POST.
   - **Validation JSON** via ArduinoJson.
   - **Insertion** dans le buffer circulaire.
   - **Mise à jour** des statistiques.
   - **Vérification d'activité** : expiration après 30 s sans nouvelles données.

3. **Communication avec le client web**

   - **Endpoint :** /api/data en GET pour servir les données.
   - **Réponse JSON** structurée :
     - Valeurs actuelles (temp, hum, timestamp).
     - Statistiques (min, max, moyenne).
     - État du système (actif/inactif, dernière mise à jour, nombre d'entrées).
   - **AJAX** interrogeant l'API toutes les 2 s pour rafraîchir le dashboard.

IV. Tests

1. **Tests effectués**
   - **Lecture capteur :** acquisition périodique du DHT22.
   - **Affichage OLED :** écran d'accueil et actualisation des valeurs.
   - **Simulation HTTP :** format des requêtes POST correct.
   - **Serveur web :**
     - Routes pour les fichiers statiques.
     - API POST/GET fonctionnelle.
     - Indicateur d'état actif/inactif.
   - **Traitement des données :** buffer circulaire et calcul des statistiques.
2. **Tests non effectués :**
   - **Robustesse réseau :** comportement en cas de perte temporaire de connexion.
   - **Persistance :** sauvegarde des données au redémarrage.
   - **Performance** sous forte charge.
   - **Sécurité** : authentification et chiffrement non implémentés.
   - **Multi-nœuds :** prototype limité à un seul capteur.

V. Suggestions d'amélioration

1. Améliorations fonctionnelles

- **Multi-nœuds :** identification et gestion de plusieurs capteurs.
- **Persistance** : stockage sur flash ou carte SD.
- **Graphiques évolutifs** : intégration de Chart.js pour visualiser les tendances.
- **Configuration à distance** via l'interface web.

2. Améliorations techniques

- **Sécurité :**
  - Authentification basique pour API et dashboard.

    - HTTPS pour chiffrer les échanges.
    - Protection contre les injections.
- **Efficacité énergétique :** deep sleep pour le capteur.
- **Robustesse :**
    - Gestion avancée des erreurs et reconnexion automatique.
    - Watchdog pour éviter les blocages.
- **Communication avancée :**
    - MQTT pour échanges plus légers.
    - WebSockets ou encore SSE pour des mises à jour en temps réel.
- Tests supplémentaires :
    - Tests unitaires
    - Tsets de stress

3. **Évolutions futures :**
    - **Intégration cloud** (AWS IoT, Azure IoT, ThingSpeak).
    - **Alertes et notifications** basées sur des seuils.
    - **IA prédictive** pour anticiper les variations.
    - **Nouveaux capteurs** (luminosité, qualité de l'air).
    - **Application mobile** pour monitoring distant.

Conclusion

Le prototype Smart Multi-Node IoT démontre une architecture cohérente exploitant l'ESP32 et Wokwi pour simuler un système de surveillance environnementale. Les choix techniques équilibrent performances embarquées et expérience utilisateur. La simulation via moniteur série illustre efficacement la logique de communication IoT. Pour un déploiement réel, il faudra implémenter une communication réseau réelle et renforcer la sécurité, la persistance des données et la robustesse globale.