Guide d'Installation et de Démarrage (Windows)

Jumeau Numérique IoT avec Diagnostic IA - Pompe Grundfos CR

Système de Monitoring Temps Réel

19 décembre 2025

Table des matières

1 Introduction

Ce document présente le guide complet d'installation et de configuration du système de jumeau numérique IoT pour le monitoring et le diagnostic intelligent d'une pompe Grundfos CR.

1.1 Présentation du Projet

Le système combine plusieurs technologies avancées :

- **Frontend :** Interface React avec Vite pour la visualisation 3D et les graphiques temps réel
- **Backend :** API FastAPI avec communication WebSocket
- **Simulation :** MATLAB/Simulink ou simulateur Python
- **Protocole IoT :** MQTT (Mosquitto broker)
- **Intelligence Artificielle :** Google Gemini 2.5 Flash avec RAG
- **Base de données vectorielle :** ChromaDB + LangChain

1.2 Fonctionnalités Principales

- Télémétrie MQTT temps réel depuis MATLAB/Simulink
- Tableau de bord React avec graphiques dynamiques et visualisation 3D
- Diagnostic IA avec RAG (Retrieval-Augmented Generation)
- Checklists de dépannage dynamiques
- Chatbot de maintenance technique

2 Architecture du Système

2.1 Schéma de Flux

```
MATLAB/Simulink → MQTT Broker → FastAPI Backend → WebSocket → React Frontend
                                        ↓
                                  RAG Engine → ChromaDB
                                        ↓
                                  Gemini AI → Diagnostics
```

2.2 Structure des Composants

- **RAG Engine :** Recherche sémantique dans le manuel de la pompe
- **AI Agent :** Diagnostics et chat alimentés par Gemini
- **Simulator :** 6 types de pannes (cavitation, usure des roulements, etc.)
- **MQTT Bridge :** Relais de télémétrie
- **Frontend :** Graphiques temps réel et modèle 3D

3 Prérequis Système

3.1 Logiciels Requis

1. **Python 3.9+** - Langage principal pour le backend
2. **Node.js 16+** - Environnement d'exécution pour le frontend
3. **Clé API Google Gemini** - Disponible sur https://makersuite.google.com/app/apikey
4. **Broker MQTT (Mosquitto)** - Gestion de la communication IoT
5. **MATLAB R2020b+** (optionnel) - Pour la simulation avancée

3.2 Configuration Minimale

- Système d'exploitation : Windows 10 ou supérieur
- RAM : 4 Go minimum (8 Go recommandé)
- Espace disque : 2 Go disponibles
- Connexion Internet pour l'installation des dépendances

4 Installation

4.1 Étape 1 : Installation du Broker MQTT

```
# Installation via Chocolatey
choco install mosquitto

# Demarrage du service
net start mosquitto
```

Note : Si Chocolatey n'est pas installé, vous pouvez télécharger Mosquitto directement depuis https://mosquitto.org/download/

4.2 Étape 2 : Clonage du Projet

```
# Cloner le depot GitHub
git clone https://github.com/6ym6n/digital_twin.git

# Acceder au repertoire du projet
cd digital_twin
```

4.3 Étape 3 : Configuration du Backend Python

4.3.1 Création de l'Environnement Virtuel

```
# Creer un environnement virtuel
python -m venv venv

# Activation
venv\Scripts\activate
```

4.3.2 Installation des Dépendances

```
# Installer les packages Python
pip install -r requirements.txt
```

Note : Assurez-vous que l'environnement virtuel est activé avant d'installer les dépendances.

4.4 Étape 4 : Configuration du Frontend React

```
# Acceder au repertoire frontend
cd frontend

# Installer les dependances Node.js
npm install

# Retourner au repertoire racine
cd ..
```

4.5 Étape 5 : Configuration des Variables d'Environnement

```
# Copier le fichier d'exemple
cp .env.example .env
```

IMPORTANT : Éditer le fichier `.env` et ajouter votre clé API Google Gemini :

```
GOOGLE_API_KEY=votre_cle_api_ici
```

5 Démarrage du Système

5.1 Méthode Automatique (Scripts Batch)

Le projet fournit des scripts de démarrage automatique pour Windows :

```
# Terminal 1 : Backend
.\start_backend.bat

# Terminal 2 : Frontend
.\start_frontend.bat

# Terminal 3 : Simulation (optionnel)
.\start_matlab_simulation.bat
```

Important : Ouvrez 3 terminaux PowerShell séparés et exécutez chaque script dans un terminal différent.

5.2 Méthode Manuelle

5.2.1 Terminal 1 : Démarrage du Backend

```
# Activer l'environnement virtuel
venv\Scripts\activate

# Demarrer FastAPI
uvicorn backend.api:app --reload --port 8000
```

Le backend sera accessible sur : http://localhost:8000

5.2.2 Terminal 2 : Démarrage du Frontend

```
# Acceder au repertoire frontend
cd frontend

# Demarrer le serveur de developpement Vite
npm run dev
```

Le frontend sera accessible sur : http://localhost:5173

5.2.3 Terminal 3 : Démarrage de la Simulation

Option A : Simulateur Python (si MATLAB non disponible)

```
# Activer l'environnement virtuel
venv\Scripts\activate

# Lancer le simulateur Python
python src/simulator.py
```

Option B : Simulation MATLAB

```
# Windows : Auto-demarrage
start_matlab_simulation.bat

# Manuel : Dans MATLAB
addpath('matlab');
mqtt_digital_twin;
```

Le simulateur MATLAB publie les données des capteurs sur le topic MQTT `pump/telemetry` toutes les 2 secondes.

5.3 Vérification du Démarrage

1. **Backend** : Accédez à http://localhost:8000/docs pour voir la documentation Swagger de l'API
2. **Frontend** : Ouvrez http://localhost:5173 dans votre navigateur
3. **MQTT** : Vérifiez que le broker Mosquitto est actif
4. **Simulation** : Vérifiez que les données s'affichent sur le tableau de bord

6 Utilisation du Système

6.1 Procédure d'Utilisation Standard

1. Démarrer tous les services (backend, frontend, broker MQTT)
2. Ouvrir le navigateur sur http://localhost:5173
3. Observer les données des capteurs en temps réel :
   - Courants triphasés (Phase A, B, C)
   - Tensions triphasées
   - Vibrations (mm/s RMS)
   - Pression de sortie (bar)
   - Température (°C)
   - État de panne et durée
4. Cliquer sur le bouton **"Diagnose"** pour obtenir une analyse IA
5. Utiliser le chatbot pour poser des questions sur la maintenance

6.2 Fonctionnalités Avancées

6.2.1 Diagnostic IA

— Analyse automatique des anomalies détectées
— Recommandations basées sur le manuel Grundfos
— Génération de checklists de dépannage contextuelles

6.2.2 Chatbot de Maintenance

— Questions techniques sur la pompe
— Recherche dans la documentation technique
— Conseils de maintenance préventive

7 Résolution des Problèmes

7.1 Problèmes MQTT

7.1.1 Le Broker ne Démarre Pas

```
# Verifier l'etat du broker
mosquitto -v

# Redemarrer le service
net stop mosquitto
net start mosquitto
```

7.2 Problèmes Frontend

7.2.1 Le Frontend ne se Charge Pas

— Vérifier que le backend est actif : http://localhost:8000/docs
— Consulter la console du navigateur (F12) pour les erreurs JavaScript
— S'assurer que le broker MQTT est en cours d'exécution
— Vérifier que le port 5173 n'est pas utilisé par une autre application

7.3 Problèmes de Données

7.3.1 Aucune Donnée Affichée

1. Démarrer le simulateur MATLAB ou Python
2. Vérifier les logs du backend pour la connexion MQTT
3. Confirmer que le broker MQTT écoute sur `localhost:1883`
4. Tester la publication MQTT manuellement :

```
# Test de publication MQTT
mosquitto_pub -h localhost -t pump/telemetry -m "test"
```

7.4 Problèmes ChromaDB

7.4.1 Erreurs de Base de Données Vectorielle

```
# Supprimer la base de donnees existante
rm -rf chroma_db

# Reconstruire ChromaDB
python src/rag_engine.py
```

7.5 Problèmes API Gemini

- Vérifier que la clé API est correctement configurée dans le fichier `.env`
- S'assurer que la clé API est active sur https://makersuite.google.com
- Vérifier les quotas d'utilisation de l'API
- Consulter les logs du backend pour les messages d'erreur détaillés

8 Structure du Projet

```
digital_twin/
 backend/                # Backend FastAPI
    api.py              # Endpoints REST + WebSocket
    mqtt_bridge.py      # Abonne MQTT
 frontend/              # Dashboard React
    src/
        App.jsx         # Composant UI principal
 src/                   # Modules principaux
    rag_engine.py       # Moteur de recherche vectorielle
    ai_agent.py         # Integration Gemini AI
    simulator.py        # Simulateur Python de secours
 matlab/                # Simulation MATLAB
    mqtt_digital_twin.m  # Editeur de telemetrie
 data/                  # Base de connaissances
    grundfos-cr-pump-troubleshooting.pdf
 start_backend.bat      # Scripts de lancement
 start_frontend.bat
 start_matlab_simulation.bat
```

9 Conseils et Bonnes Pratiques

9.1 Développement

- Toujours activer l'environnement virtuel Python avant de travailler sur le backend
- Utiliser les scripts batch fournis pour un démarrage rapide
- Consulter les logs des terminaux pour identifier rapidement les erreurs
- Sauvegarder régulièrement votre clé API dans un gestionnaire de mots de passe

9.2 Performance

- Le simulateur Python consomme moins de ressources que MATLAB
- ChromaDB peut prendre quelques secondes pour s'initialiser au premier démarrage

— WebSocket assure une latence minimale pour les données temps réel
— La visualisation 3D peut nécessiter un GPU pour des performances optimales

9.3 Sécurité

— Ne jamais committer le fichier `.env` contenant votre clé API
— Utiliser un broker MQTT avec authentification en production
— Activer HTTPS pour les déploiements publics
— Restreindre l'accès réseau au broker MQTT

10 Documentation Additionnelle

Pour plus d'informations détaillées, consultez les documents suivants dans le répertoire `docs/` :

— `INSTALLATION.md` - Guide d'installation complet
— `PIPELINE.md` - Détails de l'architecture système
— `SIMULATION.md` - Guide de configuration de la simulation
— `slideready.md` - Présentation du projet
— `matlab/README.md` - Configuration MATLAB spécifique

11 Support et Ressources

11.1 Liens Utiles

— Dépôt GitHub : https://github.com/6ym6n/digital_twin
— API Gemini : https://makersuite.google.com/app/apikey
— Documentation FastAPI : https://fastapi.tiangolo.com/
— Documentation React : https://react.dev/
— Mosquitto MQTT : https://mosquitto.org/

11.2 Contact

Pour toute question ou assistance, veuillez consulter la documentation du projet ou créer une issue sur le dépôt GitHub.

12 Conclusion

Ce guide vous a présenté l'installation et la configuration complète du système de jumeau numérique IoT. Le projet démontre l'intégration de plusieurs technologies modernes (IoT, IA, temps réel) dans une architecture cohérente et fonctionnelle.

Note : Ce projet est à usage éducatif et de démonstration. Pour un déploiement en environnement de production, des configurations supplémentaires de sécurité et de performance sont nécessaires.