République Tunisienne
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université de Tunis El Manar
École Nationale d'Ingénieurs de Tunis
Département Génie Électrique

Rapport de projet

Système de Monitoring de Capteurs Environnementaux - Dashboard IoT avec STM32, MQTT et MongoDB

Elaboré par :

Dkhil Alaa

Hentati Farah

3ème Année Génie Électrique 1

Encadré par :

M. Jelassi Khaled

Année universitaire: 2025/2026

Contents

Abstract

Ce rapport présente le développement et l'implémentation d'un système complet de monitoring en temps réel pour des capteurs environnementaux basés sur la carte B-L475E-IOT01A de STMicroelectronics. Le projet englobe l'ensemble de la chaîne de traitement des données, depuis l'acquisition par les capteurs embarqués (HTS221 pour température/humidité et LPS22HB pour pression) jusqu'à la visualisation interactive sur un dashboard moderne développé en C#.

L'architecture distribuée hybride combine le protocole MQTT pour la publication des données IoT, un broker Mosquitto pour la gestion des messages, une API REST Node.js pour l'accès aux données, une base de données MongoDB pour le stockage persistant, et une interface graphique sophistiquée offrant des visualisations dynamiques incluant trois cartes statistiques colorées, trois jauges speedometer et un graphique multi-lignes.

Ce système permet un suivi efficace et une analyse approfondie des données environnementales en temps réel avec une architecture moderne adaptée aux applications IoT professionnelles.

1 Introduction

1.1 Contexte du Projet

La surveillance environnementale joue un rôle crucial dans de nombreux domaines industriels et scientifiques. Que ce soit pour le contrôle qualité dans les environnements de production, la recherche scientifique, ou la gestion d'infrastructures sensibles, la collecte et l'analyse de données environnementales en temps réel constituent des enjeux majeurs.

Dans ce contexte, la carte de développement B-L475E-IOT01A de STMicroelectronics offre une plateforme complète et optimisée pour les applications IoT. Cette carte embarque un microcontrôleur STM32L4 et intègre nativement plusieurs capteurs MEMS (HTS221 pour température/humidité et LPS22HB pour pression), ainsi qu'un module WiFi ISM43362-M3G-L44, qui permet le développement de solutions IoT connectées.

1.2 Objectifs du Projet

Les objectifs principaux de ce projet sont:

1. **Acquisition de données embarquée** : Développer une solution d'acquisition fiable des mesures environnementales (température, humidité, pression) via les capteurs intégrés HTS221 et LPS22HB de la carte B-L475E-IOT01A.

2. **Communication IoT via MQTT** : Implémenter la transmission des données via le protocole MQTT vers un broker Mosquitto, permettant une architecture publish/subscribe adaptée aux applications IoT.

3. **Backend API REST** : Concevoir une API REST Node.js permettant au dashboard d'interroger les données historiques et d'obtenir les dernières mesures via des endpoints HTTP.

4. **Stockage persistant** : Mettre en place une base de données MongoDB pour stocker l'historique complet des mesures avec horodatage précis.

5. **Visualisation temps réel** : Créer un dashboard moderne et intuitif en C# pour une visualisation claire des données collectées.

6. **Analyse historique** : Permettre l'analyse des tendances temporelles à travers des graphiques dynamiques et un sélecteur de plages temporelles.

2 Architecture Système

2.1 Vue d'Ensemble de l'Architecture

Le système développé repose sur une architecture distribuée hybride à cinq niveaux, chaque couche ayant une responsabilité spécifique dans le traitement et la visualisation des données. Cette approche modulaire garantit une séparation claire des préoccupations, facilite la maintenance et permet une évolutivité future.

La figure 1 illustre l'architecture globale du système :

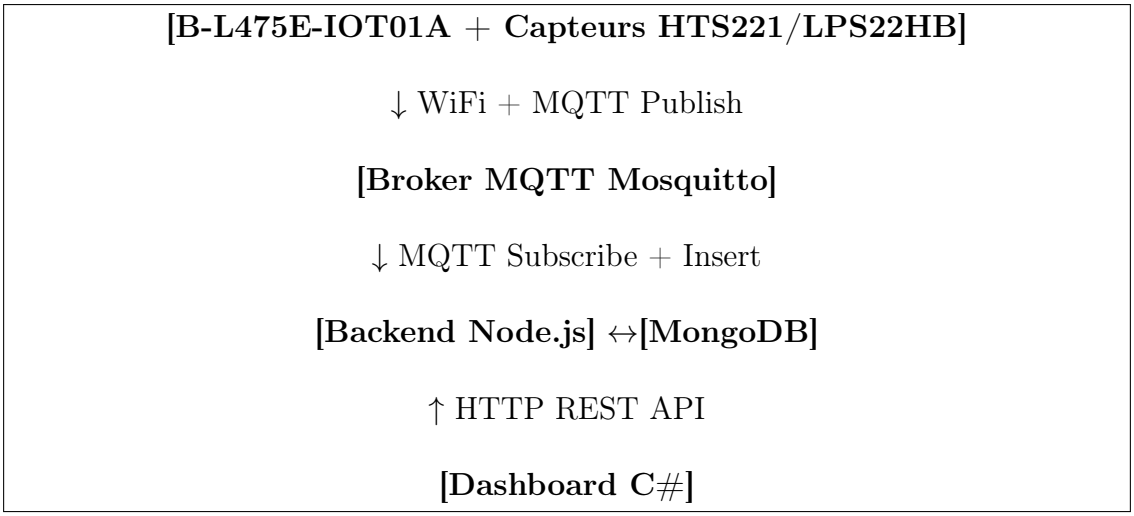

Figure 1: Architecture globale du système

Cette architecture hybride combine les avantages du protocole MQTT (publish/subscribe léger et efficace pour l'IoT) avec ceux des APIs REST (requêtes structurées pour l'interrogation de données historiques).

2.2 Couche d'Acquisition : B-L475E-IOT01A et Capteurs

La première couche du système est constituée de la carte de développement B-L475E-IOT01A qui intègre un microcontrôleur STM32L475VG et plusieurs capteurs MEMS. Cette unité embarquée assure les fonctions suivantes :

- **Acquisition des capteurs I2C** :
  - HTS221 : Capteur de température et d'humidité relative connecté via I2C2.
  - LPS22HB : Capteur de pression barométrique également sur I2C2.
- **Traitement embarqué** : Le microcontrôleur STM32L4 lit périodiquement les valeurs des capteurs (toutes les 3 secondes), effectue les conversions nécessaires et prépare les données pour la transmission.
- **Connectivité WiFi** : Module ISM43362-M3G-L44 connecté via SPI3.
- **Publication MQTT** : Envoi périodique des données vers le broker Mosquitto via trois topics distincts :
  - `sensors/temperature`
  - `sensors/humidity`
  - `sensors/pressure`

- **Feedback visuel** : LED verte (PB14) indiquant l'état de connexion WiFi et MQTT (clignotement lent quand connecté).

- **Communication série** : USART1 (PB6=TX, PB7=RX) pour le debugging et la visualisation des données via un terminal série.

Le firmware embarqué utilise les bibliothèques Arduino STM32, WiFiST pour la connectivité sans fil, PubSubClient pour MQTT, et les drivers HTS221Sensor et LPS22HBSensor pour l'interface avec les capteurs.

2.3 Couche de Messagerie : Broker MQTT Mosquitto

La deuxième couche est constituée du broker MQTT qui joue le rôle de hub de communication central pour l'architecture IoT :

- **Broker public** : Utilisation de test.mosquitto.org sur le port 1883 pour le développement et les tests.

- **Pattern Publish/Subscribe** : Le broker implémente le modèle publish/subscribe permettant une communication découplée entre la carte IoT (publisher) et les Subscribers.

- **Topics MQTT** : Organisation des données par topics thématiques (`sensors/temperature`, `sensors/humidity`, `sensors/pressure`).

Le broker Mosquitto reçoit donc les messages publiés par la carte B-L475E-IOT01A et les redistribue aux clients abonnés, notamment le backend Node.js qui persiste les données en base.

2.4 Couche Backend : API REST Node.js

La couche backend constitue le cœur logique du système. Elle remplit un double rôle :

Rôle 1 : Abonné MQTT et Persistance

- **Souscription MQTT** : Le backend s'abonne aux topics MQTT pour recevoir les données en temps réel de la carte IoT.

- **Traitement des messages** : Réception et parsing des valeurs numériques transmises via MQTT.

- **Insertion en base** : Stockage immédiat des mesures dans MongoDB avec horodatage automatique et métadonnées (type de capteur, valeur, timestamp).

Rôle 2 : API REST pour le Dashboard Le backend expose plusieurs endpoints HTTP permettant au dashboard de récupérer les données. Il suit les principes RESTful standards, utilisant les méthodes HTTP appropriées et retournant les données au format JSON pour une interopérabilité maximale.
Cette approche hybride combine les avantages de MQTT pour l'ingestion temps réel et de REST pour les requêtes structurées du dashboard.

2.5 Couche de Stockage : MongoDB

MongoDB a été choisi comme système de gestion de base de données pour plusieurs raisons :

- **Flexibilité du schéma** : La nature schemaless de MongoDB permet une évolution facile de la structure des données.
- **Performance** : L'architecture orientée documents offre d'excellentes performances pour les opérations d'insertion massive.
- **Requêtes temporelles efficaces** : Les index sur les champs de timestamp permettent des requêtes rapides sur les plages temporelles.

2.6 Couche de Présentation : Dashboard C#

La couche de présentation offre une interface graphique riche et interactive développée en C#. Cette couche communique avec l'API backend pour récupérer et afficher les données en temps réel.

Les principales caractéristiques de cette couche sont :

- **Design moderne** : Interface avec des cartes colorées pour une esthétique pratique.
- **Interactivité** : Sélecteur de plage temporelle permettant d'explorer l'historique des données.
- **Rafraîchissement automatique** : Mise à jour périodique des affichages pour un monitoring en temps réel.

2.7 Flux de Données

Le flux de données à travers le système suit le chemin suivant :

1. Les capteurs HTS221 et LPS22HB mesurent les paramètres environnementaux (température, humidité, pression) toutes les 3 secondes.
2. Le microcontrôleur STM32L4 lit ces valeurs via I2C et les convertit en chaînes de caractères formatées.
3. La carte B-L475E-IOT01A publie les données via WiFi vers le broker MQTT Mosquitto sur trois topics distincts (`sensors/temperature`, `sensors/humidity`, `sensors/pressure`).

4. Le backend Node.js, abonné à ces topics MQTT, reçoit les messages en temps réel.
5. Le backend insère immédiatement chaque mesure dans MongoDB avec un timestamp précis.
6. Le dashboard C# interroge périodiquement (toutes les 3 secondes) le backend via l'API REST.
7. Les données JSON reçues sont désérialisées et affichées dans l'interface graphique sous forme de cartes statistiques (avec calcul des moyennes, min, max, peak), de jauges speedometer (avec aiguilles animées).

3 Résultats

3.1 Captures d'Écran du Dashboard

Cette section présente les captures d'écran du dashboard fonctionnel, démontrant l'interface utilisateur finale et ses différentes fonctionnalités.

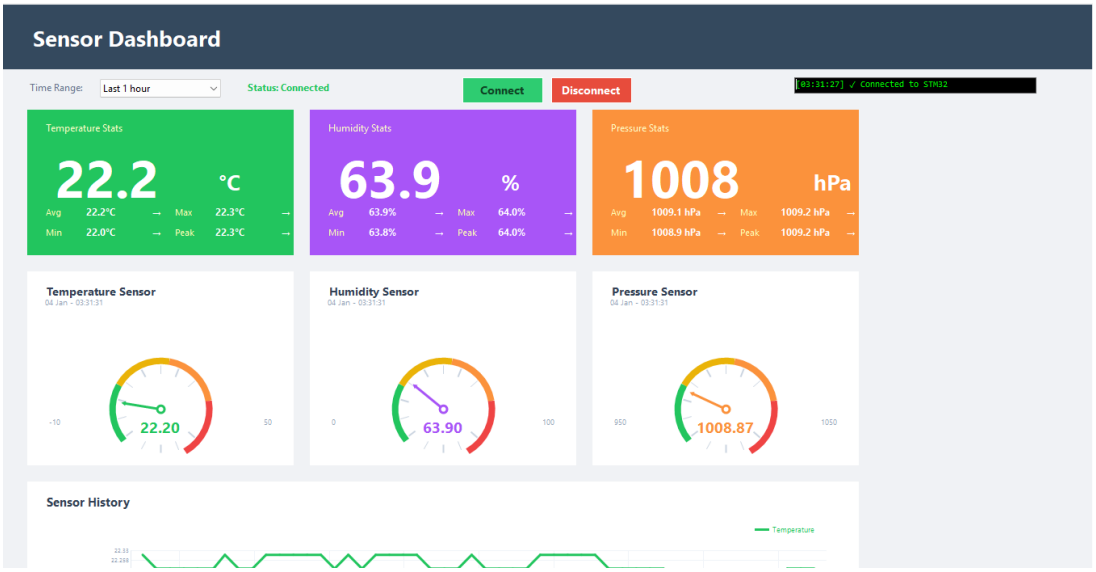

Figure 2: Vue d'ensemble du dashboard

3.1.1 Cartes de Statistiques

Les trois cartes colorées affichent les statistiques principales pour chaque type de capteur.

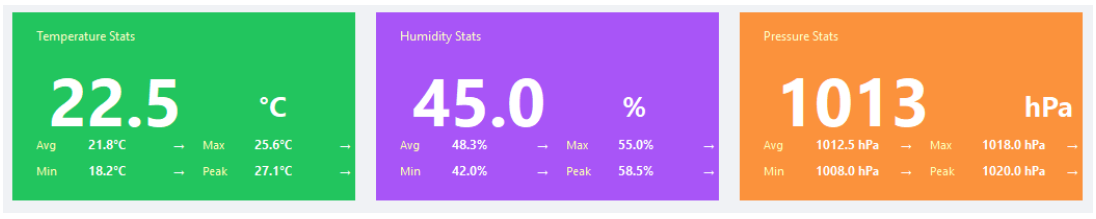

Figure 3: Cartes de statistiques colorées

3.1.2 Jauges Speedometer

Les trois jauges analogiques offrent une visualisation des valeurs actuelles avec des aiguilles animées.

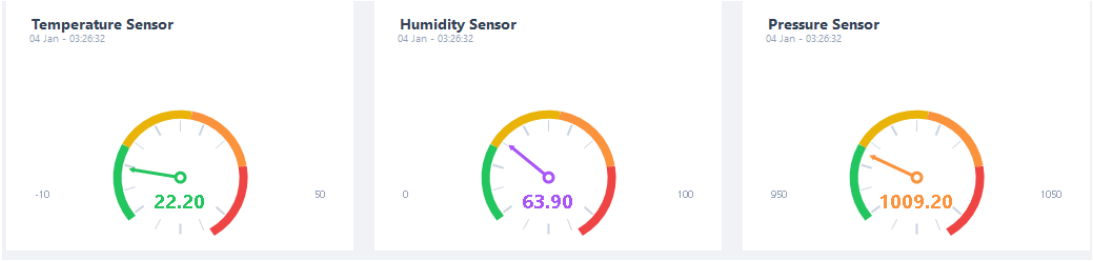

Figure 4: Jauges speedometer animées

3.1.3 Graphique d'Historique

Le graphique multi-lignes permet d'analyser l'évolution temporelle des trois paramètres simultanément.

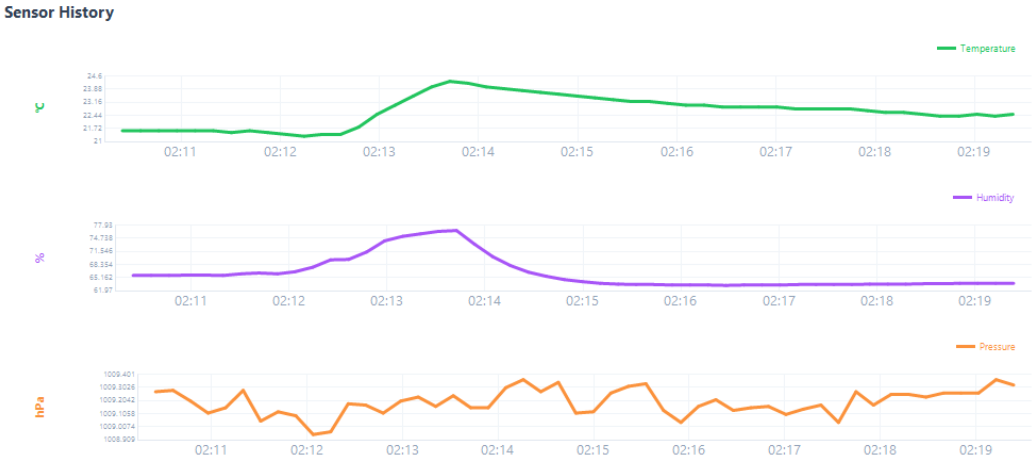

Figure 5: Graphique d'historique multi-lignes

3.1.4 Sélecteur Temporel et Console

L'interface inclut également des outils de contrôle et de monitoring.

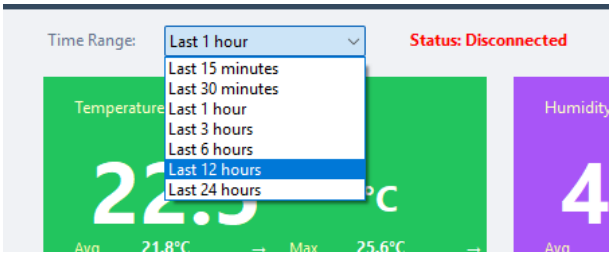

Figure 6: Outils de contrôle et console de logs

4 Conclusion

4.1 Accomplissements

Ce projet a permis de développer avec succès un système complet de monitoring environnemental, de l'acquisition des données jusqu'à leur visualisation interactive. Les objectifs initiaux ont été atteints :

1. **Architecture distribuée fonctionnelle** : La chaîne complète STM32 → API → MongoDB → Dashboard est opérationnelle et robuste.
2. **Interface moderne et intuitive** : Le dashboard offre une expérience utilisateur de qualité avec des visualisations variées et esthétiques.
3. **Monitoring temps réel efficace** : Le rafraîchissement automatique permet un suivi continu des conditions environnementales.
4. **Analyse historique** : Le sélecteur temporel et le graphique multi-lignes facilitent l'analyse des tendances.

4.2 Améliorations Futures

Bien que le système actuel soit fonctionnel et réponde aux objectifs fixés, plusieurs améliorations pourraient être envisagées pour étendre ses capacités :

- **Système d'alertes** : Notifications automatiques par email ou SMS lorsque les mesures dépassent des seuils prédéfinis.
- **Exportation de données** : Génération de rapports PDF ou Excel pour l'archivage et l'analyse externe.
- **Authentification utilisateur** : Système de login pour sécuriser l'accès et permettre des configurations personnalisées.
- **Configuration dynamique** : Interface permettant de modifier les paramètres (intervalles de rafraîchissement, seuils d'alerte) sans recompilation.

4.3 Conclusion Générale

Ce projet a atteint ses objectifs en fournissant un système complet, fonctionnel et moderne de monitoring environnemental. L'interface utilisateur développée offre une expérience de qualité professionnelle, rendant les données complexes accessibles et compréhensibles. Le système est prêt pour un déploiement en conditions réelles et constitue une excellente base pour des évolutions futures.