M2 IoT enoncé projet2

SmartAquarium – Monitoring et contrôle d’un aquarium connecté

Date / Ecole / Auteur

- du 26 au 30 Janvier 2026
- H3 Hitema groupe M2 Web
- Malik HARRIZ - m.harriz@h3hitema.fr

Objectif du projet

Les étudiants devront concevoir et implémenter un système permettant de :

- Collecter les données d’un aquarium via des capteurs Arduino
- Transmettre et stocker ces données dans une base de données
- Visualiser les données sur un tableau de bord web
- Contrôler certains actionneurs (pompe, chauffage, éclairage) via Arduino
- Déployer le projet en utilisant Docker

⚠️ Un des couche doit être implémentée en Python ou via des frameworks Python.

Le travail est à réaliser en groupe de 4 étudiants, sur une durée de 5 jours ouvrés.

Technologies

Partie embarquée – Arduino

- Carte Arduino (ESP32, ESP8266 ou Arduino classique) https://wokwi.com/
- Capteurs recommandés :
  - Température de l’eau
  - Niveau d’eau

    - pH ou conductivité
- Actionneurs simulés ou réels :
    - Pompe à eau
    - Chauffage
    - Éclairage LED
- Communication avec le serveur via HTTP REST ou MQTT

Back-end

- Langage : Python / PHP / Javascript
- Framework :
    - Flask
    - FastAPI
    - Django
    - NodeJS
    - PHP / Symfony
- API pour réception et traitement des données IoT
- Gestion des logs et validation des données

Base de données

PostgreSQL / MySQL / Elastic Search

Tables recommandées :

- Capteurs
- Mesures
- Actionneurs (état)
- Historique de commandes
- Stockage horodaté

Front-end – Site web de monitoring

Tableau de bord affichant :

- Température de l’eau, niveau, pH

- Historique des mesures
- État des actionneurs
- Possibilité de contrôler les actionneurs (ON/OFF)

Technologies :

- HTML / CSS
- Templates Python (Jinja2 / Django Templates)
- Angular
- VueJS

Docker

Conteneurisation obligatoire pour :

- Back-end
- Base de données

Fichiers Docker :

- Dockerfile
- docker-compose.yml
- Lancement global via docker-compose up

Fonctionnalités attendues

Collecte et stockage

- Réception des données Arduino
- Horodatage serveur
- Gestion d’erreurs simples (données invalides)

Monitoring et contrôle

- Tableau de bord web en temps réel ou quasi temps réel
- Historique par capteur
- Contrôle des actionneurs via le front-end
- Visualisation des alertes si seuils dépassés (optionnel)

Organisation du travail

Répartition recommandée :

- Étudiant 1 : Arduino & capteurs / actionneurs
- Étudiant 2 : Back-end Python / API
- Étudiant 3 : Base de données & Docker
- Étudiant 4 : Front-end web / intégration

Planning indicatif (5 jours)

Jour	Activités
Jour 1	Conception, architecture, MCD
Jour 2	Arduino & communication
Jour 3	Base de données & API
Jour 4	Front-end web & Docker
Jour 5	Tests, documentation, soutenance

Livrables attendus

- Code Arduino
- Code (API + front-end)
- Base de données fonctionnelle
- Interface web de monitoring et contrôle
- Fichiers Docker (Dockerfile, docker-compose.yml)
- Documentation technique
- Démonstration opérationnelle

Critères d’évaluation

Critère	Pondération
Fonctionnement Arduino	20 %

Critère	Pondération
API	20 %
Base de données	15 %
Front-end / tableau de bord	15 %
Docker	15 %
Documentation et présentation	15 %

Bonus (facultatif)

- Multi-capteurs et multi-actionneurs
- Alertes automatiques (ex : température trop haute/basse)
- Graphiques dynamiques en temps réel
- Simulations de panne et reprise automatique