|  |
| --- |
| IUT |
| Revue technique IOTIAquarium |
| GIT : https://github.com/mapfra/LPIOTIA\_2019\_IOTIAquarium |

|  |
| --- |
| Eric Harkat, Olivier Durand, Raphaël Betti, Dieunelson Dorcelus  07/05/2020 |

Table des matières

[Avant-propos sur l’environnement technique : 2](#_Toc39918743)

[Schéma du projet : 3](#_Toc39918744)

[Fonctionnalités : 3](#_Toc39918745)

[Utilisation : 5](#_Toc39918746)

[Circuit Arduino 5](#_Toc39918747)

[Broker 5](#_Toc39918748)

[Sécurité pour mosquito 6](#_Toc39918749)

[Le Brain 6](#_Toc39918750)

[API 7](#_Toc39918751)

[InfluxDB 7](#_Toc39918752)

[Schéma de la base de données InfluxDB : 8](#_Toc39918753)

[Application 8](#_Toc39918754)

[Bilan de ce qui a été réalisé et ce qu’il reste à faire : 10](#_Toc39918755)

[Evolutions possibles : 10](#_Toc39918756)

Titre du Projet : [IOTIAquarium](http://www.larka.com)

**Projet :**

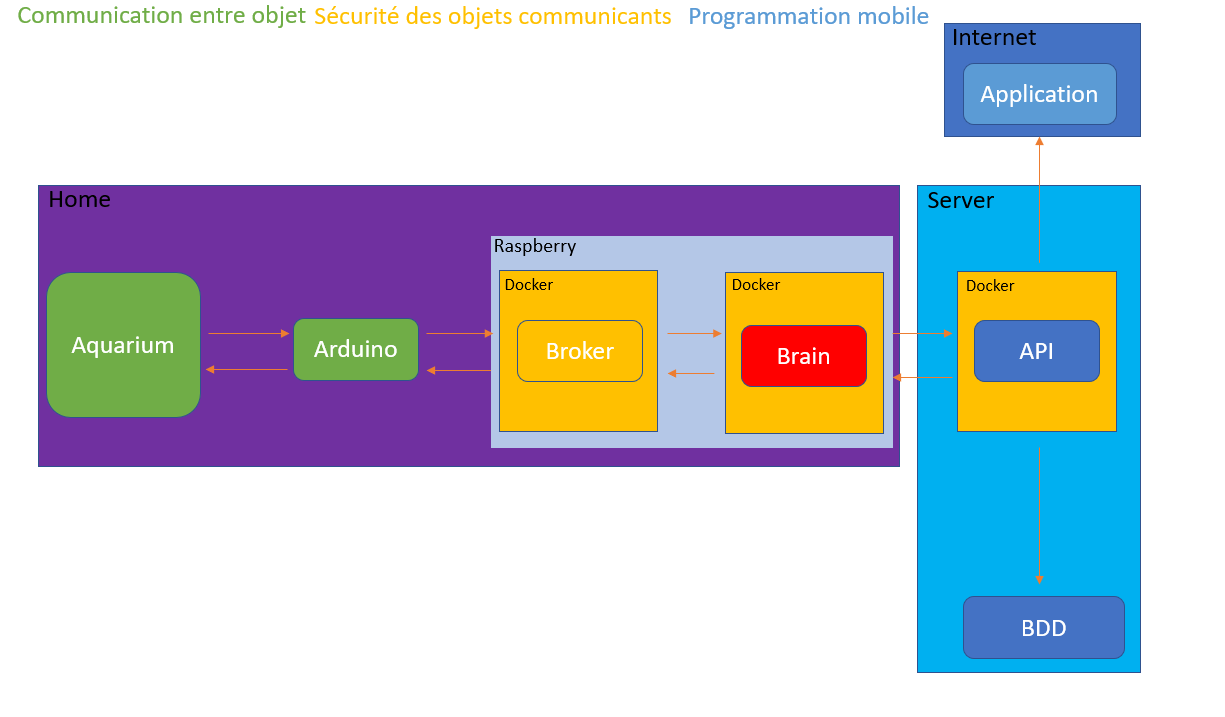
IOTIAQUARIUM a pour but d’aider à prendre soin de vos poissons en automatisant les tâches de contrôle et de maintenance. L’objectif est de communiquer des données liées à la survie de la faune et de la flore à une application qui génèrera des alertes et permettra d’intervenir sur les actionneurs. Contrairement à un écosystème naturel, un aquarium n’est pas un environnement autonome et a besoin d’une assistance constante, ces tâches sont le plus souvent assurées manuellement et donc la marge d’erreurs est plus grande qu’avec un système automatisé, le système devra donc reproduire au mieux un milieu de vie viable avec un minimum d’intervention humaine.

Catégories pour ce projet : IOT - Création d’un projet d’automatisation d’aquarium.

# Avant-propos sur l’environnement technique :

* **Architecture physique** : Utilisation d’un Arduino Mega wifi lié à six capteurs (ph, température de l’eau, niveau de l’eau) et deux actionneurs (nourriture et lumière) positionnés sur un aquarium, l’Arduino communique en wifi avec un Rapsberry-Pi 3+. Le Rapsberry abrite un conteneur (docker) contenant un Broker Mosquitto MQTT (protocole) permettant l’écoute et le relais des données envoyé par l’Arduino, un conteneur contenant la base de données InfluxDB ainsi qu’un conteneur contenant le Brain. Le Rapsberry communique avec une API, et l’API communique avec InfluxDB et une Application motorisé par Cordova.
* **Architecture Technique** : Le sketch Arduino est programmé en C. Le Brain utilise le langage JAVA pour communiquer avec le broker et l’API. Il est pensé dans la logique d’un schéma neuronale inspiré du design pattern « command ». L’API utilise NODE JS et communique avec InfluxDB grâce à des requêtes REST. Quant à elle, l’application Mobile utilise le Framework Cordova ainsi que les langages JS, HTML et CSS. Toutes ces fonctionnalités sont détaillées dans les parties suivantes.

# Schéma du projet :



# Fonctionnalités :

|  |  |
| --- | --- |
| Fonction : Alimenter les poissons | |
| Objectif | Alimenter les poissons de manière automatique. |
| Description | Un moteur associé à un distributeur de nourriture, permet de gérer l’ouverture et la fermeture de la boite pendant une durée donnée qui permette de délivrer la quantité nécessaire de nourriture ( de l’ordre de la seconde) |
| Contraintes / règles de gestion | Donner la quantité nécessaire de nourriture aux poissons. |
| Niveau de priorité | Priorité haute |

|  |  |
| --- | --- |
| Fonction : Allumer/Eteindre la lumière | |
| Objectif | Allumer ou éteindre la lumière selon le niveau de luminosité |
| Description | Un capteur de lumière permet de déterminer si l’aquarium a besoin de lumière ou pas. |
| Contraintes / règles de gestion | Allumer la lumière quand cela est nécessaire. |
| Niveau de priorité | Priorité moyenne |

|  |  |
| --- | --- |
| Fonction : Entretien de l’eau | |
| Objectif | S’assurer que le ph et que le niveau d’eau sont correctes. |
| Description | Un capteur de niveau d’eau et un capteur de Ph, permettent de s’assurer que ces mesures soient correctes. |
| Contraintes / règles de gestion | Rajouter automatiquement de l’eau si le niveau d’eau est incorrect, sans dépasser le seuil maximum.  Si le PH varie dans l’intervalle entre 0.2 autour de 7, on déclenche une demande d’entretien de l’aquarium ->Test de safety : Ajuster le ph en ajoutant de l’acide ou de la soude. |
| Niveau de priorité | Priorité haute |

|  |  |
| --- | --- |
| Fonction : Hypervision de l’environnement | |
| Objectif | Manager l’aquarium via l’application. |
| Description | Une application permet de superviser l’aquarium à distance |
| Contraintes / règles de gestion | L’application doit pourvoir afficher toutes les données des capteurs (courbes de PH température, luminosité) et proposer des aménagements en rapport à des consignes et aux espèces de poissons |
| Niveau de priorité | Priorité moyenne |

# Utilisation :

## Circuit Arduino

Les composants connectés à l’Arduino communiquent en mode stream, chaque trame est délimitée par des points virgule.

Les données périodiques sont envoyées chaque demi-heure.

Les données sont également envoyées en cas de dépassement de seuil d’un des capteurs

## Broker

Pour ce qui est de l’écoute et du relais de la trame, le broker Mosquitto est installé dans un conteneur Docker pour faciliter l’installation. L’outil docker-compose permet grâce à un fichier YAML de configurer et de lancer les services par l’intermédiaire d’une seule commande.

Pour la persistance des données, les certificats, clés et fichiers de configuration vont être stockés sur la machine hôte et synchroniser avec le conteneur à chaque redémarrage, à noter que la synchronisation va dans les deux sens hôte<-->container.

Exemple 1 : Mosquitto gère lui-même son fichier de mot de passe mais avec la synchronisation nous pouvons consulter l'état de ce fichier dans le répertoire data de l'hôte.

Exemple 2 : toutes les modifications du fichier Mosquitto.conf seront répercuté sur le container après son redémarrage.

### Sécurité pour mosquito

Le port 1883 sera fermé vers l'extérieur pour permettre une identification par mot de passe sur le réseau local. L'identification par certificat et mot de passe sera réservée au port 8883 ouvert vers l'extérieur. Pour la connexion au Brain nous utilisons la technologie des websocket sur le port 8080 avec la même sécurité que pour la connexion TLS (certificat + identifiant/mot de passe).

Toute la configuration du broker est transparente dans le processus d’installation car un effort d’automatisation a été fourni pour simplifier la tâche au maximum.

**Sécurité pour InfluxDB**

Le port 8086 sera fermé vers l'extérieur pour permettre une identification par mot de passe sur le réseau local. Pas d'accès extérieur.

## Le Brain

Le Brain est le module qui permet de traiter les informations échangées entre le broker et l’Api. Pour sa création nous nous sommes inspiré des systèmes de réseau neuronaux. Chaque neurone aura une tache et communiquera avec un autre neurone via une synapse.

N

N

N

N

N

N

N

Neurone composant

Neurone de fonctionnalité

N

N

Thread d’écoute

Entrée

Neurone Bridge

N

Le neurone qui fait office de bridge récupère et découpe la trame en morceau, puis les envoient vers les neurones composants concernés. Pour finir les neurones composants transmettent l’information aux neurones de fonctionnalité selon l’opération à effectuer.

Lorsque les neurones de fonctionnalités ont terminé leur traitement, ils envoient leurs données à l’API.

## API

L’api est le module qui sert d’interface entre le Brain, l’application et InfluxDB, grâce à des requêtes REST (POST, GET) elle est capable de lire et d’écrire dans la base de données. Elle a été développée avec le Framework Express qui permet de mettre en place une infrastructure d’application Web Node.js. Il est important de noter que L’API est complétement modulable et s’adapte à tout type de configuration d’aquarium afin d’accepter l’ajout de nouveaux capteurs sur l’Arduino.

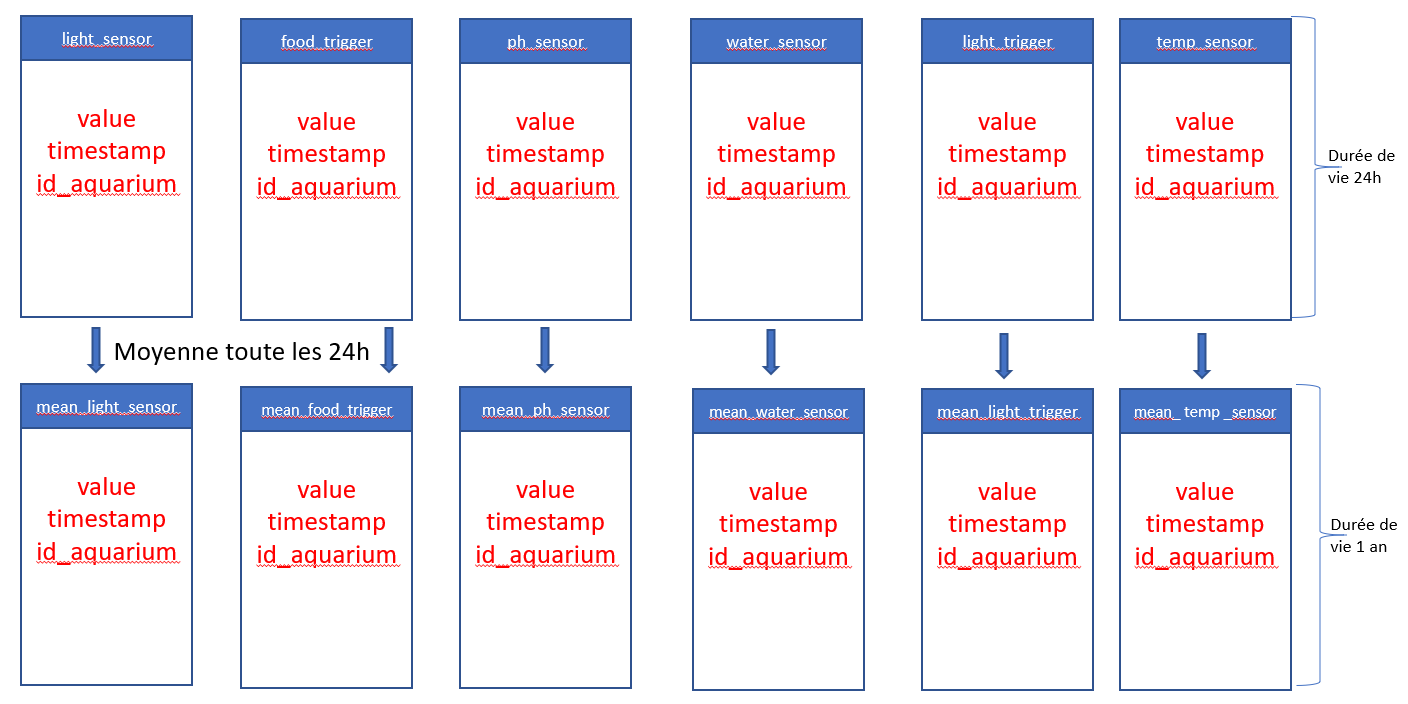
## InfluxDB

Influx DB est une base de données qui a été optimisée pour les données de séries temporelles (ce sont des suites de valeurs numériques représentant l'évolution d'une quantité spécifique au cours du temps). Ces données proviennent généralement de sources telles que des groupes de capteurs IOT.

Ces séries de données ont en commun qu'elles sont plus utiles dans leur ensemble. Il est difficile de tirer des conclusions à partir d'une valeur indiquant que l'utilisation du processeur est de 12% à 12 :38 :35. Elle devient plus utile lorsqu'elle est combinée avec le reste de la série et visualisée (dans un graphique par exemple). C'est là que les tendances commencent à se manifester, et qu'il est possible de tirer des conclusions. En outre, les données des séries temporelles sont généralement créées une seule fois et rarement mises à jour.

Le résultat est qu'InfluxDB n'est pas une base de données CRUD (Create Read Update Delete) complète mais plutôt une CR, qui donne la priorité aux performances de création et de lecture des données par rapport à la mise à jour et à la suppression.

## Schéma de la base de données InfluxDB :



**Value** correspond à la donnée envoyé par le capteur.

**Timstamp** correspond à la date de reception et sert de clé de primaire.

**Id\_aquarium** correspond à l’identifiant de l’aquarium.

Ces trois champs sont contenu dans une table (appelé shard dans InfluxDB).

Il y a donc six shards contenant les series temporelles liées aux capteurs et aux actionneurs, à cela s’ajoute cinq autres tables permettant l’archivage des données sous forme de sous échantillonage. (il s’agit d’une méthode consistant à prélévé la moyenne des valeurs d’une serie temporelle, afin de réduire le volume de donnée).

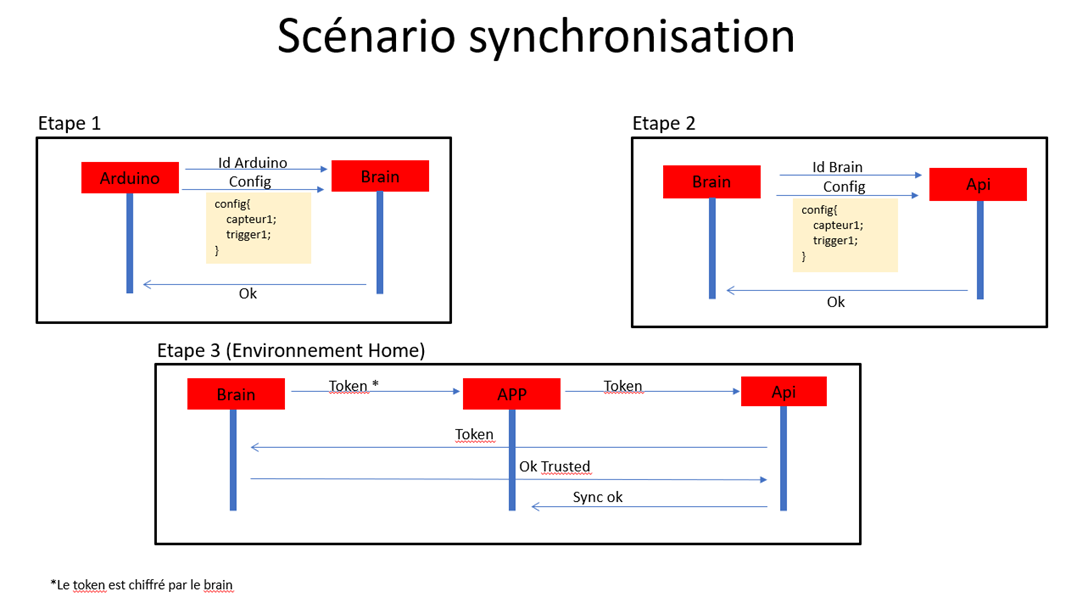
## Application

L’application motorisé par Cordova permet une utilisation web et mobile à la fois. L’utilisateur se connecte via ses identifiants, il peut ajouter ou supprimer des aquariums, contrôler les données de chacun d’eux et définir des horaires d’allumage et d’extinction de la lumière ainsi que l’heure à laquelle la nourriture doit être distribué.

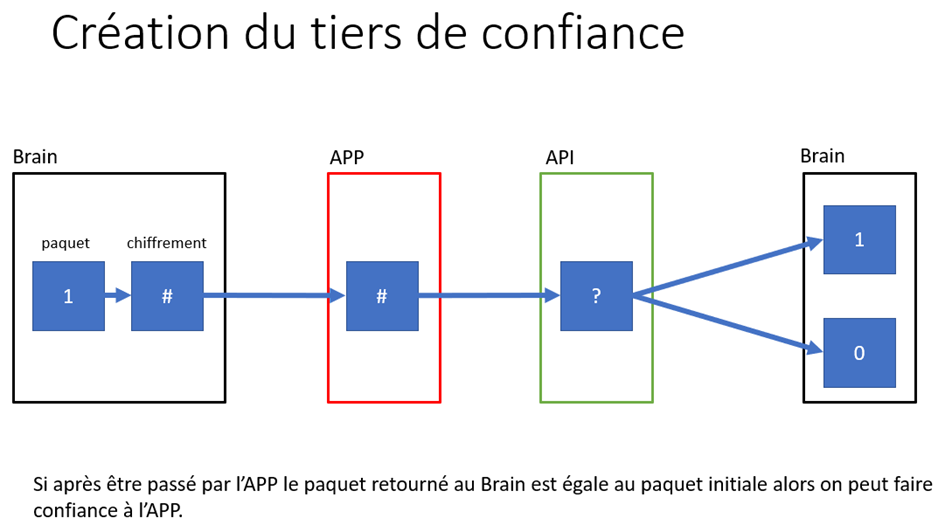
Une Alerte permet d’informer si un capteur a dépassé le seuil autorisé.

L’application communique ave l’API afin de récupérer les données et interagir avec l’aquarium.

Lorsqu’un utilisateur veut ajouter un aquarium dans l’application, il doit procéder à sa synchronisation.



Durant l’étape 3 de la synchronisation, on détermine si l’utilisateur est un tier de confiance.



# Bilan de ce qui a été réalisé :

* Récupération des données envoyé par les capteurs via un broker MQTT
* Automatisation complète du déploiement des conteneurs Mosquitto et InfluxDb
* Traitement des données à travers le réseau neuronal
* Gestion des données grâce à l’Api
* Monitoring des données à travers une application mobile et web

# Ce qu’il reste à faire :

* Finir l’interconnexion des parties de l’infrastructure
* La gestion des alertes
* Hypervision sur une plage de données
* Conteneurisation du Brain et de l’API

# Evolutions possibles :

* Pouvoir paramétrer depuis l’application les seuils des capteurs.
* Scalabilité du projet. (Capacité d’un projet à s’adapter à la demande en gardant ses fonctionnalités).
* Possibilité d’afficher des graphs dans l’application
* Archiver les logs de connections et d’erreurs pour pouvoir monitorer plus précisément l’activité des différents éléments du projet.
* Utiliser les données de sous-échantillonnées pour dégager des tendances générales du comportement de l’aquarium.