Rapport de projet

Station de pompage connectée

Tuteurs : Nicolas Palix - Didier Donsez - Olivier Richard

Étudiants : Qjangian Fu - Maxime Chevalier - Héloïse Fernandes
03/04/2017

Table des matières

I- Introduction

Le projet décrit dans ce rapport a été réalisé dans le cadre des projets de 4ème année en RICM de Polytech Grenoble. Il a été proposé par Nicolas Palix et a pour but principal la réalisation d'une application web permettant de vérifier et contrôler la station de pompage à distance. Cependant ce projet devait être réalisé en collaboration avec une équipe de 4ème année en IESE. Malheureusement le sujet n'a pas été choisi, ainsi toute la partie concernant la récupération de données à partir de capteurs a donc été simulée.

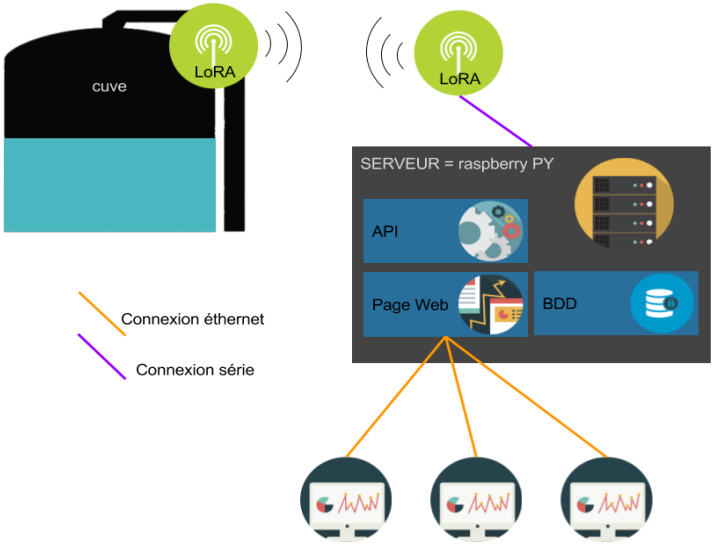
II-Objectif du projet

Pour arroser un jardin public, on utilise un arrosage automatique alimenté par une cuve, elle-même alimenté par une nappe phréatique. Donc quand le jardin est arrosé le niveau de la cuve décroît. Il est donc nécessaire de consulter régulièrement le niveau de la cuve pour la remplir.
De plus les adhérents de la MJC de Saint Martin d'Hère souhaiterais accéder au niveau de la cuve pour pouvoir l'analyser.
Ainsi l'objectif du projet se décompose en 2 grande parties :

- Permettre aux adhérents et aux jardiniers de consulter le niveau de la cuve à un instant T
- Permettre au jardinier uniquement, d'activer la pompe à distance pour remplir la cuve.

III-Technologies utilisées

Nous avions pour contrainte d'utiliser la technologie LoRa pour transmettre les données de la cuve vers une application web ou mobile tout en se servant d'une raspberry py comme serveur. Etant donné que nous devions passer par plusieurs appareils qui utilisent des langages de programmation différents nous avons choisis de construire le projet sous forme de micro-services pour une meilleure modularité du système.



LoRa

La communication LoRa est réalisée au travers de 2 cartes NUCLEO STM sur le compilateur en ligne MBED. Elles servent à faire le lien entre la cuve et le serveur :

- La cuve envoie des données (niveau, état de la cuve) au serveur
- Le serveur envoie un ordre (niveau de remplissage souhaité)

Les cartes communiquent actuellement en **865 MHz**.
La partie LoRa est séparée en deux applications :

- **SX1272Pump** : programme de la carte reliée à la pompe.
    - Lien vers le dépôt : https://developer.mbed.org/users/chevamax/code/SX1272Pump/

- **SX1276Server** : Transmet les informations reçues au programme Java, et envoie les ordres d'activation de la pompe.
    - Lien vers le dépôt : https://developer.mbed.org/users/chevamax/code/SX1276Server/

Application partie Pompe

- **Carte utilisée :** ST-Nucleo-F411RE
- **Extension LoRa :** SX1272MB2xAS

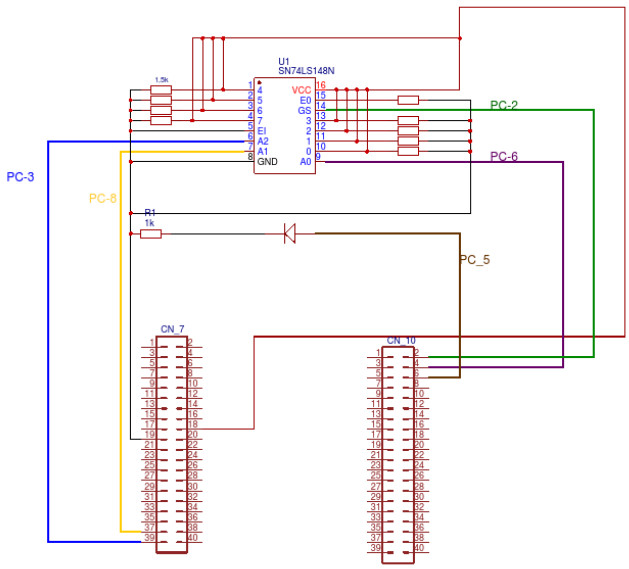
Cette application permet de communiquer avec une unique station. Pour choisir l'**ID** de la station, il faut changer **ID_STATION** dans le programme. Il faut impérativement que celui-ci tienne sur un seul octet.

Branchements pour la simulation

La simulation est réalisée grâce :

- Un convertisseur 8 vers 4 (SN74LS148N) pour simuler les différents niveaux d'eau.
- Une LED représentant l'activation de la pompe.

Vous trouverez ci-dessous le modèle de branchement pour simuler les capteurs, et la sortie d'activation de la pompe.



Une fois en route, la carte va instantanément envoyer le niveau de la cuve en LoRa. Il faut donc brancher l'alimentation en dernier.

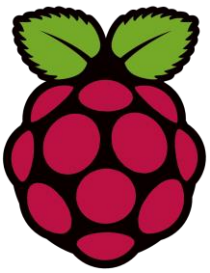
Application partie serveur

- **Carte utilisée :** ST-Nucleo-L152RE
- **Extension LoRa :** SX1276MB1MAS

Cette application permet de communiquer avec différentes cartes. Tant que le format de trame est celui demandé, et qu'**ID_STATION** de la trame correspond à l'**ID** de la station courante.
La carte communique ensuite les informations, via le port **SERIE**, au programme java avec les paramètres suivants :

- 9600 baud
- 8 bits
- 1 bit de stop
- Pas de parité

RASPBERRY



Le serveur est stocké sur une Raspberry Pi sous Raspbian JESSIE. Il contient :

- Un serveur apache2
- Une base de données MySQL qui contient nos utilisateurs et nos mesures. (**Schéma relationnel en annexes**)
- Une API en PHP permettant la communication entre l'application et la base de données
- Un programme java permettant à la base de données de communiquer avec la carte LoRa branchée en série
- Une application Oracle JET qui permet de visualiser les données

Pour mettre le serveur en ligne, nous avons utilisé une adresse Dyn DNS gratuite(http://stationpompeco2017.hopto.org/) enregistré sur http://www.noip.com/(cepandant avec l'option gratuite l'adresse doit être renouvelée chaque mois).

- Adresse serveur : http://stationpompeco2017.hopto.org/
- Adresse API : http://stationpompeco2017.hopto.org/
- Adresse application Oracle JET :http://stationpompeco2017.hopto.org/web/

ORACLE JET

L'application web est réalisée en Oracle JET en utilisant NetBeans et les données mis à notre disposition via une API. Oracle JET regroupe une collection de bibliothèques JavaScript open source populaires (par exemple jQuery, RequireJS et Knockout.js) combinés à une série de composants JavaScript. Oracle JET est un outil qui permet de créer rapidement et facilement des interfaces graphiques grâce au système de module.

Cet outil gère notamment :

- La visualisation des données
- La modularité
- Le responsive design (CSS, HTML5, SVG, etc.)
- L'accessibilité

Lors des manipulations effectuées sur la page par l'utilisateur, le programme en JavaScript envoie des requêtes au serveur Web, d'où des temps de latence plus réduits que ceux d'une application Web classique.

Pour le service REST, avec l'API de base de données, on envoie des requêtes HTTP(POST, GET, PUT) qui opèrent sur les URLs(end points). La nomination des URLs REST est basée sur leurs fonctionnalités ce qui rend leurs utilisations plus intuitive.

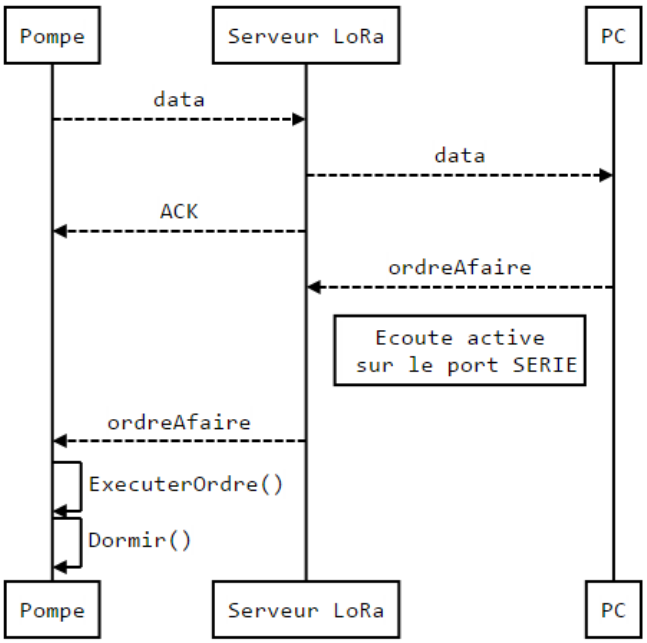
IV-Objectifs réalisés

Ensemble des fonctionnalités disponibles pour ce projet :

- Les données générées par un capteur fictif sont reçues et traité par la carte LoRa qui est branchée dessus
- Les deux cartes LoRa communiquent l'une avec l'autre grâce à un système de trame
- La carte LoRa peut envoyer des données vers le serveur qui les enregistres dans la base de données. Quant au serveur il peut envoyer un ordre lu dans la base de donnée vers la carte LoRa (via un programme java)
- Un ensemble de page php permet de récupérer les données contenues dans la base de données au format JSON
- Une application Oracle JET sur le serveur Raspberry pour l'affichage des données

V-Fonctionnement

Communication capteur-LoRa :

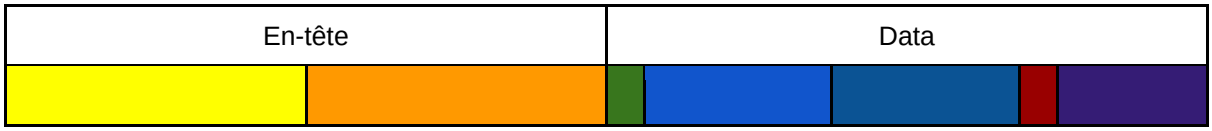


La pompe est en mode basse consommation la plupart du temps. Toutes les 30 minutes, elle se réveille et envoie ses données. La carte passe alors en réception, et un timer se déclenche. Il y a réémission des données si l'ACK n'a pas été reçu.
Lorsque l'ACK est reçu, un autre timer se déclenche pour la réception d'ordre. Si la carte reçoit une trame d'ordre, elle l'exécute et se met ensuite en consommation basse énergie. Si le timer sonne, la carte s'endort.

La carte coté serveur est en écoute permanente. Lors d'une réception, elle envoie l'ACK, puis transmet les données sur le port série. Elle écoute ensuite le port série jusqu'à réception d'un ordre. Cela permet d'avoir soit l'ordre effectif, soit un ordre vide qui représente un ACK. Si l'ordre est effectif, il est envoyé, sinon, la carte repasse en réception.

Communication LoRa :

- **Trame de mise à jour des données** : 4 octets



Identifiant de la station, 1 octet

Identifiant de la carte LoRa émettrice, 1 octet

Etat de la pompe, 1 bits (1 allumé, 0 sinon)

Nombre de niveau maximum (5 bits), nombre de niveaux maximum = 31, dans notre cas nous n'en avons que 8.

Niveau de la pompe 5 bits

Bit inutilisé laissé à 0

Niveau de la batterie qui va de 0 à 10(4 bits)

Cette trame est construite par la carte côté cuve, et ensuite transmise jusqu'au programme java qui la décompose et la traite.

- **Trame d'envoi d'ordre** : 3 octets

Identifiant de la carte LoRa réceptrice (1 octet)

Identifiant de carte émettrice (1 octet)

Niveau maximum de remplissage. La cuve se remplit jusqu'au niveau indiqué puis s'éteint (5 bits)

1 = il y a un ordre, 0 sinon

Bits inutilisé(2bits)

Cette trame est construite par l'application java, et est ensuite transmise à la carte LoRa serveur et sert d'ACK si le bit d'ordre est à 0. S'il est positionné à 1, elle sert d'ACK pour la carte LoRa côté serveur et est transmise à la carte LoRa côté cuve. Cette dernière la traite.

API JAVA :

L'api côté JAVA consiste en un programme java qui écoute un port COM. Quand le port reçoit des données le programme les lit et les enregistre dans la base de données. Ensuite il récupère dans la même base de données un ordre pour la “device” qui lui a envoyer les données et met à jour cet ordre.

La classe ThreadCommunicationJava qui correspond au programme principal peut communiquer avec la base de données par deux moyens :

- En exécutant directement des requêtes SQL (fonctionnalité uniquement disponible en local sur le serveur)
- En passant par l'API PHP.

Le protocole de communication utilisé entre le programme et la carte LoRa est le même que celui entre les 2 cartes LoRa.

API PHP :

Il s’agit d’un ensemble de lien qui renvoie des données sous format JSON.
Fonctionnalité de l’api :

- Obtenir des données
- S’inscrire (adhérent seulement)
- Ajouter ordre (jardinier seulement)
    - ATTENTION : s’il existe déjà un ordre non exécuté il est remplacé par le nouveau
- Obtenir ordre
- Vérification connexion
- Obtenir la liste des cartes connectées aux cuves
- Mettre à jour un ordre

Application oracle jet :

Fonctionnalité de l’application :

- Inscription d'adhérents
- Pour les adhérents
    - Connexion
    - Consulter le niveau courant pour la device i
    - Consulter le niveau de batterie pour la device i
    - Consulter le niveau sur les 24 derniers enregistrements pour la device i
    - Consulter le niveau au selon un mois et une année pour la device i
    - Consulter ordres en cours pour la device i
- Pour les jardiniers
    - Possibilité d'ajouter/modifier un ordre pour la device i

VI-Amélioration et correction

Corrections :

- Corriger erreur CRC pour les cartes LoRa

Améliorations :

- Ajouter la possibilité d'inscription pour les jardiniers
- Sécuriser le serveur et l'api
- Mise à jour automatique de l'application Oracle JET (lorsqu'une nouvelle donnée est reçue)
- Faire un thème personnalisé pour l'application
- Faire évoluer l'application web, en application Cordova
- Permettre l'annulation d'un ordre
- Ne pas faire d'attente active sur les cartes LoRa
- Faire fonctionner le deepSleep avec un timer
- Une Raspberry Pi avec plus de ports USB et un processeur légèrement plus performant
- Exporter les données via l'application Oracle JET.

VII-Conclusion

Nous avons beaucoup aimé travailler sur ce projet, malgré la déception qu'aucun groupe d'IESE ne l'ait choisi. Nous avons pu travailler sur beaucoup de technologies (C++, LoRa, JavaScript, Oracle JET, knocout.js, JQuery, Java, PHP) et appris comment mettre en ligne un serveur web avec un nom de domaine. Travailler sur un projet nécessitant toutes ces compétences a été éprouvant et difficile, mais nous sommes heureux du résultat et espérons que ce projet puisse, un jour, aboutir.

VIII-Annexe

Application oracle jet

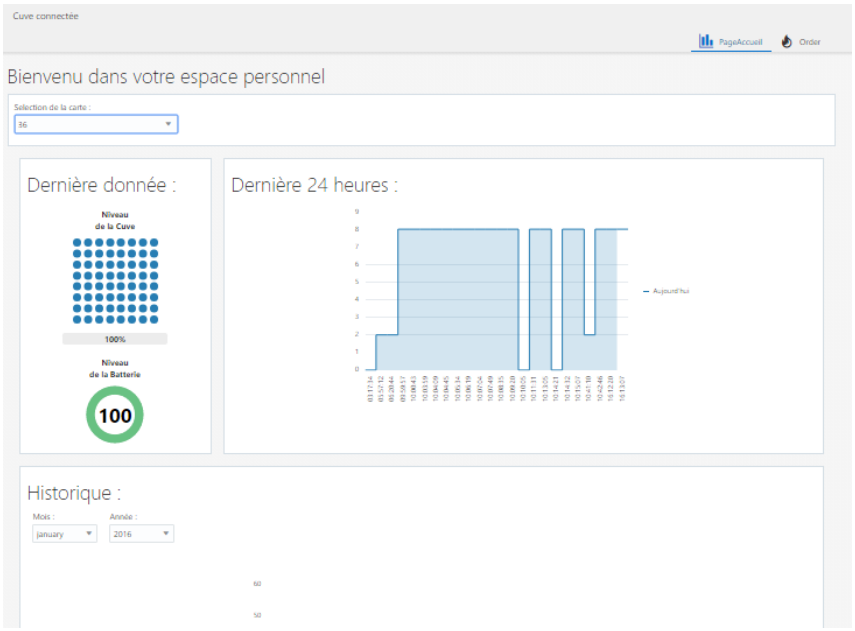


Figure 1 Page d'accueil d'un utilisateur connecté

Figure 2 Formulaire d'envoi d'ordre