RAPPORT DE STAGE

MESURE DE CAPTEURS DANS UN PUITS CENTRALISÉ

IMESTIR Ibrahim

Février 2020 à Mai 2020

Maître de Stage : Mr. Benjamin HANOUNE
Enseignant référant : Mr. Frédéric SERVAIS

Établissement d'origine : HE2B – ESI, Rue Royale 67, 1000 Bruxelles
Établissement d'accueil : IUT 'A' de Lille, Avenue Paul Langevin, 59653 Villeneuve-d'Ascq
Entreprise d'accueil : IRCICA, 50 Avenue Halley, 59650 Villeneuve-d'Ascq

Avant-Propos

Ce rapport est le résultat détaillant le travail que j'ai réalisé pendant ces 4 mois de stage dans le cadre du programme d'échange étudiant Erasmus+ à l'IUT 'A' de Lille qui m'a placé dans l'Institut de Recherche sur les Composants logiciels et matériels pour l'Information et la Communication Avancée (IRCICA) afin de conclure mon Bachelor en Informatiques et Système – Section Industrielle et ainsi me permettre de continuer mon cursus en Master en Informatiques.

Ayant voulu réaliser un stage loin de chez moi pour « prendre mon indépendance », pouvoir me gérer seul et me donner une expérience de travail sans aides familiales et en même temps sans trop m'éloigner de mon pays natal, la France était le pays idéal où réaliser ce stage. Je n'ai eu que très peu de soucis culturels grâce à la proximité des pays (même langue, mêmes supermarchés, peu de différences culturelles etc.). J'ai eu donc très peu de mal à m'adapter à la culture française.

J'ai choisi cette entreprise car je suis intéressé par l'aspect de l'automatisation d'actions (telle que la mesure de température) par l'utilisation d'automates ou de systèmes embarqués et d'objets connectés et j'ai voulu réaliser un projet concret et sérieux dans ce domaine.

Ce rapport a pour but de montrer comment reproduire mon travail de zéro sans avoir besoin d'effectuer des recherches approfondies sur le sujet. Il a aussi pour objectif de guider les utilisateurs sur l'utilisation du programme ainsi que des futurs développeurs sur l'aspect ajouts/modifications si les besoins de l'entreprise utilisant ce système changent.

Remerciements

Je souhaite tout d'abord remercier Mr. Rédha KASSI, ingénieur et chercheur de l'IRCICA et mon encadrant professionnel ainsi que Mr. Benjamin HANOUNE, chargé de recherches CNRS au Laboratoire de Physico-chimie des Processus de Combustion et de l'Atmosphère (PC2A) et mon maître de stage pour m'avoir guidé et aidé dans la réalisation de mon projet en m'apportant des idées, suggestions et améliorations pendant la période du stage.

Je tiens également à adresser mes remerciements à Mr. Patrick LEBÈGUE, professeur de l'IUT 'A' de Lille ainsi que Mr. Frédéric SERVAIS, professeur de l'École Supérieure d'Informatique à Bruxelles, pour m'avoir assisté pédagogiquement et m'ayant aidé administrativement au cours du stage.

Pour finir, je remercie également Mme Roxana RUSU pour m'avoir aidé dans toutes les démarches d'inscriptions, d'aide au logement et de conseils une fois arrivé à l'IUT 'A' de Lille.

Sommaire

Glossaire

- **Arduino:** Système embarqué (comme le raspberry PI) mais avec beaucoup moins de fonctions mais consommant beaucoup moins d'énergie.
- **BLE:** « Bluetooth Low Energy », nouvelle version de Bluetooth permettant une communication sans socket mais avec des « caractéristiques »
- **DB/BDD:** « DataBase » / « Base de données ».
- **DHT11:** Capteur mesurant la température et l'humidité, il peut être connecté à un Arduino.
- **Grafana:** Interface web permettant la visualisation de données présents sur une base de données influxDB.
- **I2C:** « Integrated Circuit», Protocole de communication d'un système embarqué.
- **InfluxDB:** Système de gestion temporelle de base de données.
- **Intel TinyTile:** « Variante » d'un Arduino faite par Intel, supporte le BLE.
- **Nœud:** Celui qui récolte les données.
- **Puits:** « DataPoller », celui qui va interroger les capteurs. On fait allusion au Raspberry PI lorsque ce terme est utilisé dans ce rapport.
- **Raspberry PI:** Système embarqué permettant l'automatisation et la réalisation d'une ou plusieurs actions demandées, possède un système d'exploitation (Raspbian, qui est une distribution de Linux).
- **RPI:** « Raspberry PI ».
- **RTC:** « Real Time Clock », petit mécanisme permettant à l'heure de ne pas se dérégler. Il est présent dans tous les ordinateurs basiques.
- **UUID:** « Universally Unique IDentifier », Identifiant unique permettant de différencier quelque chose d'une autre.

Introduction

Durant le deuxième quadrimestre de l'année 2020 (de février à mai), j'ai pu effectuer un stage à l'Institut de Recherche sur les Composants logiciels et matériels pour l'Information et la Communication Avancée (IRCICA).

L'IRCICA est une unité de Service et de Recherche basée sur une structure d'hôtels à projets. Elle développe depuis une dizaine d'années des recherches interdisciplinaires pour imaginer et créer des technologies de l'information et de la communication responsables*. J'ai été guidé par mon encadrant professionnel Mr. Rédha KASSI, ingénieur et chercheur dans cette entreprise.

Ma mission était de réaliser un système centralisé en un puits interrogeant des nœuds et utilisant les mesures acquises pour diverses utilisations (recherche, contrôle, informations etc.). Ce stage m'a permis l'utilisation de plusieurs composants et langages de programmations différents au sein d'un même système. Ce qui m'a permis d'enrichir mes connaissances dans l'utilisation de ces composants, les lier entre-eux et la programmation dans un langage auquel je n'avais pas beaucoup d'expérience.

Ce travail m'a aussi permis de gagner de l'expérience dans la collaboration et le « développement à la demande » avec les principaux utilisateurs de ce système (qui sont les employés du bâtiment de Lilliad et mon Maître de stage Mr. Benjamin HANOUNE).

L'écriture de ce rapport a pour sources les diverses recherches effectuées sur Internet pour résoudre les problématiques survenues durant le développement du système, la méthode de réalisation de ce système, le fonctionnement des divers composants utilisés ainsi que la manière d'utiliser les multiples technologies employées (telles que le Bluetooth et le BLE).

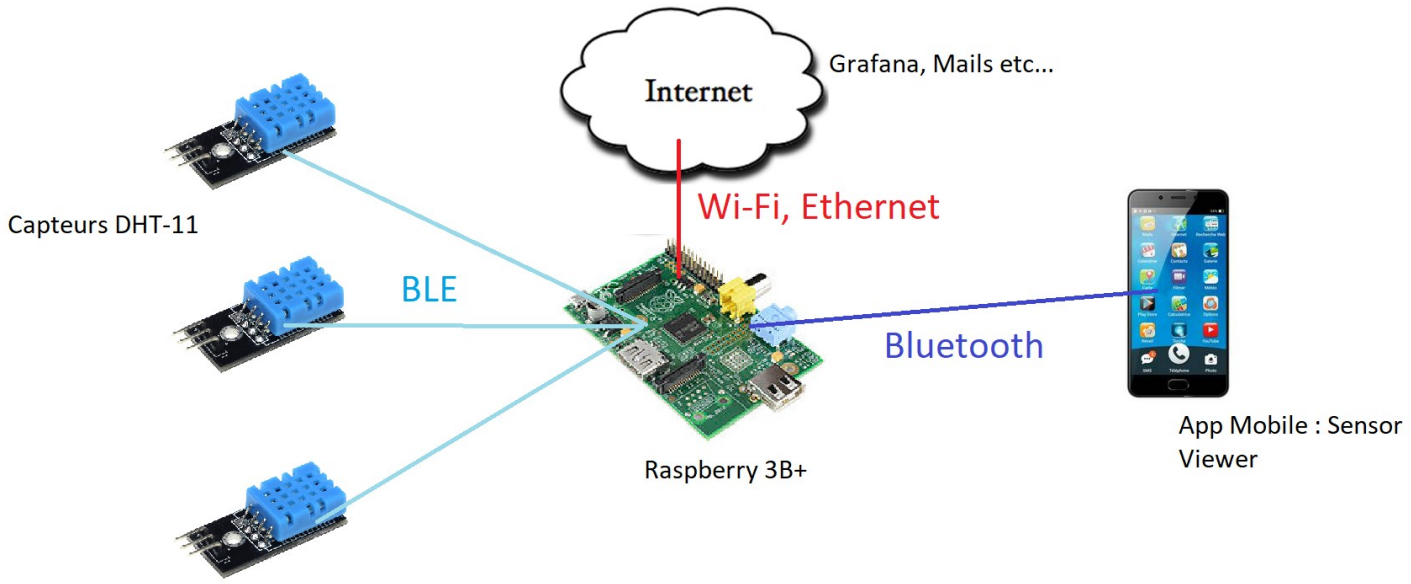
Ce rapport vous détaillera le fonctionnement, l'installation, la configuration et l'utilisation des composants et du système.

* Sources : Bibliographie

Présentation du projet

Le projet consiste à réaliser un système permettant de mesurer la température et l'humidité présentes dans les magasins (Archives se situant à Lilliad), afin de les contrôler et réguler si besoin (permettant de préserver la qualité des livres).

Le système est composé de cette manière :



Le puits (ici un RPI 3B+) est un serveur interrogeant chaque minute les capteurs (ici DHT-11) contrôlable par l'application mobile (fonctionnelle avec n'importe quel smartphone Android compatible Bluetooth).

Cette application mobile permettra d'envoyer divers requêtes au puits : Envoyer un mail contenant les données récoltées des capteurs, gérer un envoi périodique et automatique (fréquence définie par l'utilisateur), éteindre proprement le puits et consulter un graphique des valeurs mesurées des divers capteurs des 6 dernières heures.

Remarques concernant ce rapport

- Toute l'installation ainsi que la configuration ont été préparées pour qu'un puits puisse fonctionner (celui des magasins situé à Lilliad), ces notes vous montrent comment ajouter un deuxième puits de zéro, installer le serveur dessus, le configurer et préparer Grafana pour afficher les données reçues des capteurs.

- Ces notes font aussi office de mode d'emploi pour utiliser l'application mobile.

- Si vous souhaitez créer une nouvelle adresse mail pour gérer l'envoi des mails, pensez à bien activer l'option « Autoriser les applications moins sécurisées » dans les paramètres du compte Google associé à cette boite mail.

- L'installation de la RTC peut différer selon le modèle choisi (ces notes ne seront donc pas fonctionnelles si vous décidez d'utiliser un autre modèle de RTC qu'utilisé ici).

- Comme la RTC, si vous décidez de choisir des matériels différents de ceux utilisés dans ces notes, l'installation pourrait différer (vous devrez donc vous référer à la notice de votre matériel).

Fonctionnement du serveur

Le serveur prend des mesures chaque minute (fréquence configurable dans le code), ces mesures sont stockées dans une base de données et envoyées à Grafana, si l'envoi des données échoue, ces données sont stockées dans une base de données temporaire (qui servira à être « vidée » dans Grafana), la base de données est supprimée si sa taille dépasse 100Mb ou lors du redémarrage du serveur après avoir édité le fichier « nodes.cfg ». La BDD temporaire est supprimée lorsque le puits arrive à envoyer ces données à Grafana ou lorsque sa taille dépasse aussi 100Mb.

Quand l'utilisateur demande à ce qu'on lui envoi un mail avec les données, le serveur convertit la BDD en fichier Excel puis l'envoie par mail à l'adresse indiquée, ce fichier excel est supprimé par la suite.

Quand l'utilisateur souhaite consulter les données des capteurs directement dans l'application mobile, le serveur enverra les données des capteurs stockées dans la BDD des 6 dernières heures (afin de ne pas surcharger le smartphone de données et pour des soucis de lisibilité).

L'envoi automatique des mails ne peut prendre qu'une fréquence allant de 1 à 60 jours inclus. Si la fréquence est réglée à moins de 1 ou plus de 60, elle sera automatiquement remise à 1 ou 60. Si la fréquence est invalide (n'est pas un nombre par exemple), elle sera réglée à 7 jours.

À noter que le décompte se fait dès le redémarrage du serveur, ou lors de l'édition de la fréquence d'envoi (le décompte est réinitialisé à chaque changement de fréquence d'envoi de mail ou redémarrage du serveur).

liliadbot@gmail.com est l'adresse mail utilisée pour envoyer les mails, voici les accès au compte :

Mot de Passe : I1n>1{^=Y6\Xq}L *Date de Naissance :* 01/01/1980

ATTENTION : Le puits a besoin d'une connexion à Internet (Wi-Fi ou Ethernet) pour pouvoir envoyer les données par mail et à Grafana.

À propos de LoRaWAN

LoRaWAN (Long Range Wide Area Network) est un protocole de communication permettant une communication entre objets (IoT en général) par radio à bas débit. Ce protocole est caractérisé par sa très faible consommation d'énergie et sa très longue portée (couvrant en général la surface d'une ville entière), le principal défaut de ce système est son faible débit qui ne permet d'envoyer que près d'un kb par minute.

Ce protocole nous permettra de faire communiquer notre système à un serveur LoRa qui couvrira entièrement le bâtiment de Lilliad, sans avoir besoin d'une connexion Internet sur le puits.

L'avantage est que l'on n'a pas à se soucier de la connectivité du puits à Internet, le désavantage c'est que plusieurs fonctionnalités du système seront hors d'usage (tel que l'envoi de mail ainsi que la restauration de la DB temporaire) qui sont onéreux en espace et donc nécessitent une connectivité.

Ne m'étant pas occupé du développement du serveur LoRa, c'est Moermans Guillaume (51561@etu.he2b.be) qui a réalisé la partie du code permettant le transfère du fichier « lora » (fichier se trouvant dans « /usr/bin/BlueServer/ », servant à transférer les données des capteurs au serveur LoRa), vous trouverez cette partie dans le dossier « lora » se trouvant dans « _SensorRPI » si vous souhaitez effectuer des modifications au code.

Matériel nécessaire à la réalisation du projet

Pour pouvoir établir ce système, il faut préparer plusieurs matériels :

1. Un ou plusieurs Capteur(s) pouvant être connecté à un Arduino
2. Un ou plusieurs arduino(s) (1 arduino par capteur)
3. Un système embarqué fonctionnant sous Linux et compatible BLE et Bluetooth
4. Une RTC compatible avec ce système embarqué (I2C)
5. Un shield permettant de réaliser une communication LoRaWAN
6. Un smartphone Android compatible Bluetooth

Pour notre cas nous avons décidé de choisir ces matériels :

1. Capteur DHT-11
2. Intel TinyTile
3. Raspberry PI 3B+ fonctionnant sous la dernière version de Raspbian
4. RTC DS3231 compatible avec le RPI 3b+
5. Un shield ArduPi du site cooking-hacks.com
6. Un smartphone Android quelconque compatible Bluetooth

Installation du Matériel

Installation des capteurs

Pour préparer un capteur, il faut tout d'abord installer les librairies « DHT sensor Library » (pour les DHT11/22), « SparkFun SCD30 Arduino Library » (pour le SCD30) et « Intel Curie Boards » (pour le curieBLE du tinyTILE, disponible dans le menu « Boards Manager » se trouvant dans : « Tools > Boards »).

Pour ce faire, aller dans « Tools > Manage Libraries... ».

Dès que c'est fait, il faut connecter l'Arduino auquel le capteur est lié à un ordinateur et lui implémenter un script en .ino pour le faire fonctionner. Trois scripts sont déjà fournis pour les capteurs DHT11, T6615 et SCD30 (se trouvant dans le dossier « _SensorArduino »).

Pour chaque valeur que le capteur mesure, on doit lui déclarer une « caractéristique ». Une caractéristique est un « endroit » où va être stockée la valeur de chaque grandeur mesurée. Par exemple, pour le capteur DHT11, il y a deux caractéristiques déclarées : Une pour la température et une autre pour l'humidité.

Chaque caractéristique est identifiée par un « UUID » qui est une chaîne de caractère simple visant à différencier la caractéristique des autes.

Voici comment sont déclarées les caractéristiques du fichier dht11.ino :

```
BLEService ser("5AF5BB12-7D2C-44E3-A1DC-000000000000");
BLEFloatCharacteristic dataTemp("5AF5BB12-7D2C-44E3-A1DC-5BE69B124FC6", BLERead);
BLEFloatCharacteristic dataHum("5AF5BB12-7D2C-44E3-A1DC-A8FFC0117A81", BLERead);

void setup() {
```

« dataTemp » et « dataHum » sont les noms des caractéristiques et la chaîne de caractère déclarée pour chaque caractéristique est son UUID associé.

Et voici comment stocker une valeur dans une caractéristique :

```
// TEMP ; HUMIDITY
dataTemp.writeValue(t);
dataHum.writeValue(h);
```

Installation du Raspberry PI

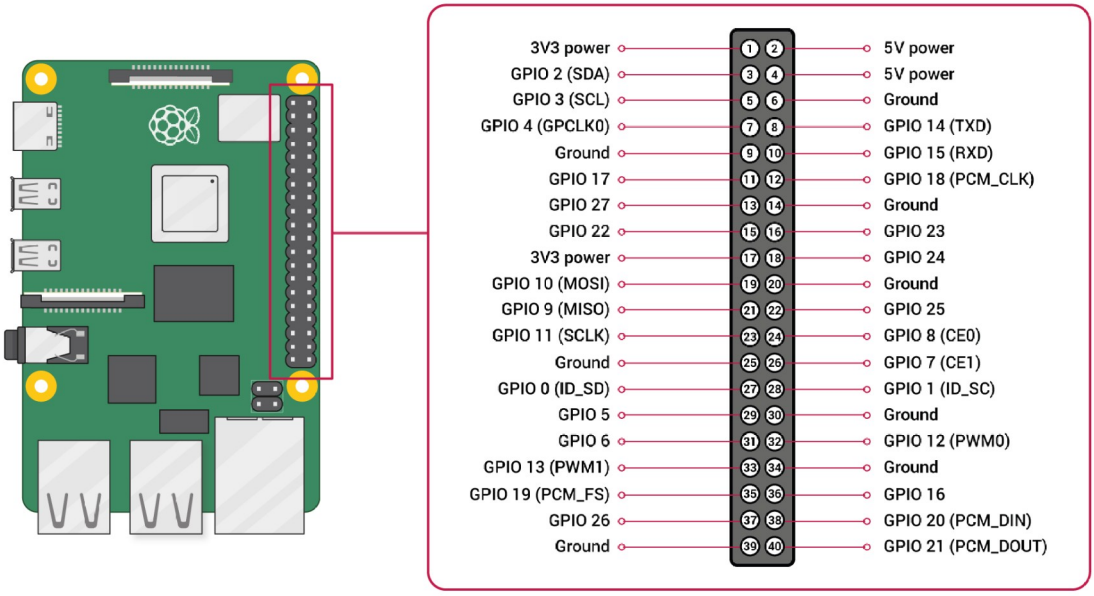
Installation de la RTC DS3231 sur le RPI

ATTENTION : Il faut que le RPI soit éteint avant de connecter les pins.

Il faut tout d'abord connecter la RTC sur le RPI :

- Connecter le pin SDA vers SDA
- Connecter le pin SCL vers SCL
- Connecter le pin VCC sur un pin 3.3V du RPI
- Connecter le pin GND sur un pin Ground du RPI

Voici un schéma pour vous aider à connecter le tout:



ATTENTION : une connexion Internet est REQUISE pour l'installation de la RTC et du serveur.

Maintenant que le tout est connecté, il faut installer les utilitaires suivants :

- `sudo apt-get install python-smbus`
- `sudo apt-get install i2c-tools`

Une fois cette étape réalisée, il faut activer l'interface I2C dans le menu de configuration du RPI (« sudo raspi-config » > Interfacing Options > I2C).

Ajoutez maintenant dans le fichier modules (« sudo nano /etc/modules ») les lignes suivantes :

- `i2c-bcm2708`
- `i2c-dev`
- `rtc-ds1307`

Il faut maintenant synchroniser la date de la RTC à celle du système (à chaque redémarrage), pour ce faire, éditer le fichier /etc/rc.local (« sudo nano /etc/rc.local ») en lui ajoutant les lignes ci-dessous (avant « exit 0 ») :

- `echo ds1307 0x68 > /sys/class/i2c-adapter/i2c-1/new_device`
- `hwclock -s`

Vous pouvez maintenant redémarrer le RPI, si en redémarrant, l'heure de la RTC est déréglée, vous devez (en étant connecté à Internet) d'abord définir le fuseau horaire (si ce n'est pas déjà fait) en allant dans :

- `« sudo raspi-config > Localisation > TimeZone »`

puis exécutez le commande suivante :

- `sudo hwclock -w`

Changer le nom du bluetooth du RPI

Exécutez la commande ci-dessous dans un terminal :

« sudo nano /etc/machine-info » en ajoutant/modifiant la ligne :
« PRETTY_HOSTNAME=MON_NOM » dans l'éditeur nano. *« MON_NOM » est le nom que vous voulez attribuer au RPI (n'ajoutez pas de guillemets).*

Exécutez maintenant la commande suivante :« sudo service bluetooth restart »

puis redémarrez le serveur s'il est déjà installé sur le RPI.

Il est important de réaliser cette étape avant d'installer le serveur car l'identification du puits par Grafana et l'application mobile se fait grâce à ce nom.

Préparation de la communication LoRaWAN

Il faut tout d'abord placer le shield cooking-hacks sur le RPI.

Ensuite, il suffit d'éditer le fichier « /boot/config.txt » (« sudo nano /boot/config.txt) en lui ajoutant les lignes suivantes :

- `dtoverlay=pi3-miniuart-bt`
- `enable_uart=1`
- `dtparam=spi=on`
- `dtparam=i2c_arm=on`

Il faut maintenant éditer le fichier « /boot/cmdline.txt » (« sudo nano /boot/cmdline.txt ») en remplaçant la ligne par :

- `dwc_otg.lpm_enable=0 root=/dev/mmcblk0p2 rootfstype=ext4 elevator=deadline rootwait ip=192.168.1.160::255.255.255.0`

Installation du serveur sur le RPI

Il faut tout d'abord déplacer le dossier d'installation (« _SensorRPI ») dans le bureau du RPI et d'ensuite lancer « install.py » s'y trouvant (tous les fichiers de ce dossier doivent être présents avant de lancer le script), ce script permettra l'installation et la configuration automatique des packages nécessaires au fonctionnement du serveur. Le Raspberry redémarrera automatiquement lorsque les dépendances ont fini d'être installées et configurées.

Le lancement du script python sur Linux se fait par la commande « sudo python install.py » dans le terminal *en étant situé dans le dossier d'installation*.

Après l'installation, le serveur démarrera automatiquement à chaque redémarrage du puits.

À noter qu'il faut par la suite modifier le fichier « keys » dans le dossier du serveur par l'ensemble de clés fournies par le serveur LoRa.

Installation de l'application mobile

Pour pouvoir installer l'application mobile sur votre téléphone, il suffit simplement de lancer le fichier d'installation « SensorViewer.apk » se trouvant dans le dossier « _AndroidProject » via votre téléphone.

Configuration du serveur du Puits

Pour pouvoir faire toute modification ou configuration sur le serveur, il faut se rendre dans le dossier « /usr/bin/BlueServer/ » et modifier le fichier « config.cfg ».

Il faut d'abord stopper le serveur avant de faire toute modification, pour ce faire : aller dans le terminal et taper la commande :

« sudo systemctl stop BlueServer.service »

Pour démarrer le serveur après l'avoir arrêté :

« sudo systemctl start BlueServer.service »

Un fichier « log » se trouve dans ce répertoire, ce qui permet d'avoir des informations sur l'état du serveur et donc de pouvoir déboguer. Ce fichier est supprimé à chaque redémarrage du serveur (pour des soucis de gestion de mémoire).

Les BDD utilisées par le système se trouvent aussi dans ce dossier.

ATTENTION : Il est important de ne pas modifier le reste du fichier en dehors du label CONFIGURATION pour ne pas provoquer des dysfonctionnements du serveur.

Ajouter/Retirer des capteurs

Il suffit de modifier le fichier « nodes.cfg » se trouvant dans le dossier du serveur (« /usr/bin/BlueServer/ »), en ajoutant une entrée comme ceci (entre les deux « # ») :

```
1   [Sensors]
2   #
3   comment = "Capteur DHT11 - 1"
4   mac = 84:68:3E:06:78:BE
5   labels = temperature, humidity
6   uuids = 5AF5BB12-7D2C-44E3-A1DC-5BE69B124FC6, 5AF5BB12-7D2C-44E3-A1DC-A8FFC0117A81
7   #
```

- **comment :** une brève description/nom du capteur
- **mac :** l'adresse mac du capteur
- **labels :** le nom des grandeurs mesurées (seront réutilisées dans grafana)
- **uuids :** les UUIDs liés à ces labels (le premier uuid correspond au premier label, le 2e uuid au 2e label etc...)

Toutes les infos doivent être sur la même ligne.

ATTENTION : Il faut absolument que l'entrée « labels » et l'entrée « uuids » aient le même nombre d'informations sous peine de faire planter le serveur.

Les labels ne doivent pas contenir de caractère spécial.

Modifier la fréquence d'acquisition des données

Il suffit de modifier la variable « sensor_freq » se trouvant dans « config.cfg » en nombre de minute(s) que vous voulez.

Exemple : Si sensor_freq = 3, il y aura 1 mesure chaque 3 minutes.

```
37   sensor_freq = 1
```

Configuration de Grafana

Pour que le puits puisse envoyer les données des capteurs à Grafana, il faut d'abord avoir créé une base de données influxdb où stocker ces valeurs (nécessite un administrateur. Pour ma part, j'ai contacté Mr. Degrande Samuel à l'adresse : samuel.degrande@univ-lille.fr).

Une fois cette étape réalisée, il faut modifier la configuration du serveur du puits pour que les données soient envoyées à cette BDD :

```
34  influx_host = "lilliad.apisense.io"
35  influx_username = "lilliad"
36  influx_password = "lilliad"
37  influx_db = "warehouse"
```

Maintenant que le serveur et la BDD sont configurés, il faut créer un affichage pour ces données dans Grafana.

Rendez-vous à l'adresse de Grafana (ici lilliad.apisense.io), et cliquez dans « Magasins » (vous pouvez aussi créer un nouveau dashboard si vous le souhaitez).

Créez un panel pour chaque grandeur mesurée par les capteur du puits, il faut absolument nommer son RPI avant (cf. « Changer le nom du Bluetooth du RPI »).

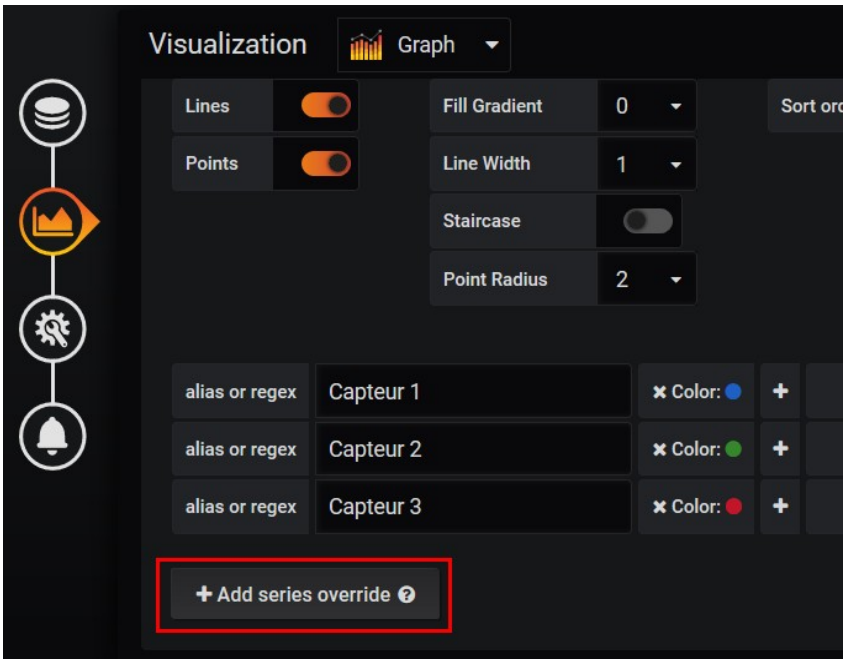
Une fois dans la configuration du Panel, réglez les données reçues que vous souhaitez visualiser dans l'onglet « Queries » comme ci-dessous :

Lier une couleur par capteur

Pour ce faire, il faut d'aller dans l'onglet "Visualization" (dans le menu Edit de votre Panel) et d'appuyer sur le bouton « Add series override ».

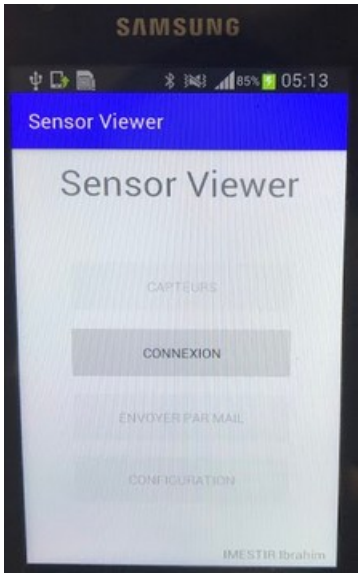
Il ne suffit plus qu'à entrer le nom du capteur et la couleur associée.

(Ceci est à faire pour tous les panels).



Utilisation de l’application Mobile

Connexion au puits



Pour pouvoir vous connecter au puits via l’application mobile, il suffit d’appuyer sur le bouton « CONNEXION » et de sélectionner dans la liste de périphériques (qui se remplit au fur et à mesure) le puits (dans notre cas, le nom du puits était « raspberrypi »), vous pouvez choisir de vous connecter automatiquement au puits lors du démarrage de l’application. La localisation du périphérique et le Bluetooth doivent être activés pour que l’application fonctionne (la localisation est automatiquement demandée dans le menu connexion).

À noter que les autres fonctions du programme se débloquent une fois que le smartphone est connecté au puits.

Si un échec de connexion survient, cela est peut-être dû à un problème suivant :

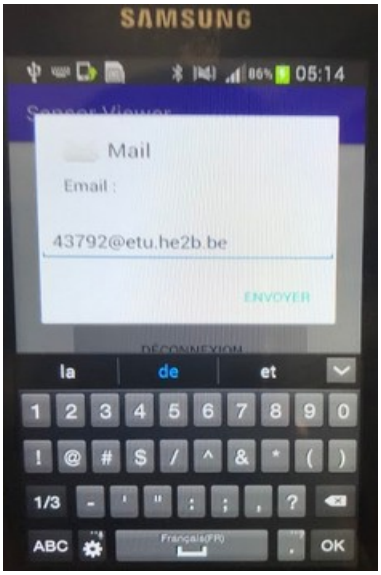
- Êtes-vous assez proche du puits ?
- S’agit-il vraiment du puits ?
- Est-ce que le serveur est installé sur le puits ?
- Est-ce que le puits est correctement configuré ?
- Est-ce qu’un deuxième smartphone est connecté au puits ?

Si les échecs persistent, essayez de d’abord redémarrer votre smartphone, si ça n’a aucun effet, redémarrez le puits (en débranchant le câble).

Visualiser les données des capteurs

Il suffit d'appuyer sur le bouton « CAPTEURS », les données peuvent prendre un certain nombre de temps avant de s'afficher (dépend de la connexion avec le puits et du nombre de valeurs reçues).

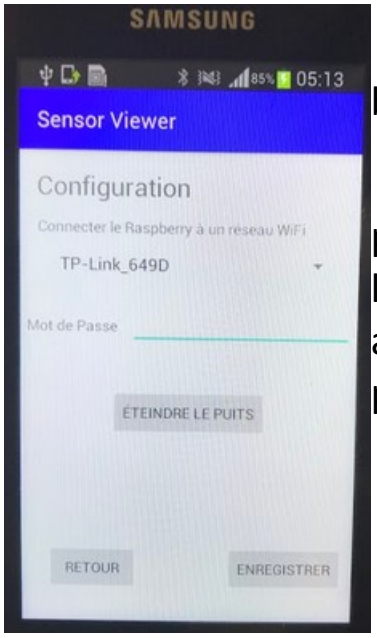
Envoyer par mail les données des capteurs



Il faut appuyer sur le bouton « ENVOYER PAR MAIL » et de suivre les instructions, ce bouton permet également de configurer l'envoi périodique et automatique des mails. L'envoi manuel des données envoie systématiquement les données des 60 derniers jours, c'est pour cela que je vous recommande l'envoi automatique.

Dans le menu « Envoi automatique », le bouton « IGNORER » permet de conserver la configuration du serveur, le bouton « NON » permet de désactiver l'envoi automatique, le bouton « OUI » permet d'activer cette fonction.

Éteindre le puits via l’app et gérer son redémarrage



Appuyer sur le bouton « CONFIGURATION » puis sur le bouton « ÉTEINDRE LE PUITS ».

Après avoir appuyé sur ce bouton, le puits s’éteindra. Pour pouvoir le rallumer, il suffit juste de débrancher et rebrancher le câble d’alimentation. Puisque le serveur se lance automatiquement à chaque démarrage du puits, il n’y aura pas plus de manipulation à réaliser.

Tâches à réaliser par la suite

Les tâches que vous trouverez ci dessous sont les tâches que je n'ai pas pu réaliser dû aux événements liés au COVID-19 (manque de matériel) ou la difficulté de la tâche.

- ➔ *Système de connexion à un wifi via l'app mobile :* ce système permet d'éviter de reconfigurer manuellement le puits (en lui branchant un écran, clavier etc.) en lui envoyant directement par Bluetooth les informations nécessaires pour se connecter à un Wi-Fi.
  L'envoi de ces données est déjà fonctionnel mais la connexion au réseau est plus compliquée à réaliser.
  Si une prochaine personne se penche sur ce travail, veuillez noter que la fonction qui est appelée quand l'utilisateur envoie des données Wi-Fi est « wifiConnect(SSID, pwd) » (se trouvant dans le fichier bs_socket.py). Les informations sont transmises en clair, il faudrait les chiffrer grâce à une clef publique.

```
232  def wifiConnect(SSID, pwd):
233      # TODO : Wifi Connection
234      return
```

Je recommande quand même d'utiliser un câble Ethernet plutôt qu'un réseau Wi-Fi dû à l'instabilité et à la configuration laborieuse du réseau eduroam.

Conclusion

En conclusion, ce projet m'a permis, d'une part, de travailler et d'apprendre à utiliser des technologies comme le Bluetooth ou le BLE ainsi que de développer sur un environnement Android, d'autre part, de composer un système de plusieurs technologies très différentes pour réaliser un but unique (ici de mesurer les températures et humidités des magasins).

Ensuite, ayant développé le projet en Python, un langage auquel je n'ai pas beaucoup de connaissances, j'ai pu enrichir ma compréhension de ce langage et gagner de l'expérience dans son utilisation.

Ce projet m'a aussi permis de « prendre mon indépendance » car ce fut la première fois que je vis seul dans un autre pays pour une période de plusieurs mois grâce au programme d'échange étudiant Erasmus+.

J'ai aussi pu expérimenter le fait de devoir faire du télétravail (dû au confinement total à cause des événements du COVID-19), ce qui implique de devoir s'imposer un rythme de travail quotidien chez soi et de travailler presque entièrement seul (faire des recherches, tester des librairies, résoudre des problèmes etc.).

Ce programme d'échange étudiant était une excellente expérience que ce soit en terme d'apprentissage ou tout simplement humainement et je recommande tout étudiant voulant réaliser un stage de partir le faire à l'étranger.

Pour toute question ou requête concernant le projet, n'hésitez pas à me contacter via mon adresse mail.

Bibliographie

- https://ircica.univ-lille.fr/fr/linstitut

Capteur Arduino

- https://www.electronicproducts.com/Board_Level_Products/Communication_Boards/Hands_on_review_getting_started_with_the_Intel_tinyTILE.aspx
- https://www.arduino.cc/en/Tutorial/Genuino101CurieBLELED
- https://forum.arduino.cc/index.php?topic=386417.0
- nagashur.com/blog/2013/06/18/lire-la-valeur-dune-sonde-de-temperature-et-d'hygrometrie-dht11/
- http://co2meters.com/Documentation/AppNotes/AN157-T6613-Arduino-uart.pdf

Raspberry PI

- https://elinux.org/RPi_Bluetooth_LE
- http://pages.iu.edu/~rwisman/c490/html/pythonandbluetooth.htm
- https://wiki.python.org/moin/TcpCommunication
- https://www.tutorialspoint.com/send-mail-from-your-gmail-account-using-python
- https://stackoverflow.com/questions/20470626/python-script-for-raspberrypi-to-connect-wifi-automatically
- https://www.influxdata.com/blog/getting-started-python-influxdb/
- http://nagashur.com/blog/2015/08/31/ds3231-raspberry-pi-rtc-ajouter-une-horloge-temps-reel-i2c-alitest/
- http://www.intellamech.com/RaspberryPi-projects/rpi_RTCds3231
- https://stackoverflow.com/questions/26299053/changing-raspberry-pi-bluetooth-device-name

Application Mobile

- https://medium.com/@onyekweretrust/how-to-create-a-simple-graph-in-android-6c484324a4c1
- https://stackoverflow.com/questions/10795424/how-to-get-the-bluetooth-devices-as-a-list
- https://solidgeargroup.com/discovery-bluetooth-devices-android/
- https://stackoverflow.com/questions/21947834/how-to-test-android-apps-in-a-real-device-with-android-studio
- https://www.zoftino.com/saving-files-to-internal-storage-&-external-storage-in-android
- https://stackoverflow.com/questions/5185015/updating-android-ui-using-threads
- https://developer.android.com/guide/topics/connectivity/bluetooth
- https://stackoverflow.com/questions/22899475/android-sample-bluetooth-code-to-send-a-simple-string-via-bluetooth
- https://stackoverflow.com/questions/20921375/cant-discover-service-created-with-pybluez
- https://github.com/jjoe64/GraphView/wiki/Simple-graph