Rapport de projet de quatrième année
Smart Sensor WiFi

Thomas Maurice
Benoit Maliar

École Polytechnique Universitaire de Lille
Département IMA

Thomas.Maurice@polytech-lille.net
Benoit.Maliar@polytech-lille.net

15 avril 2014

Table des matières

Introduction

Les projets de quatrième année d'IMA sont l'occasion pour les étudiants de mener à bien un projet sur un relativement long terme en partenariat avec des industriels ou des laboratoires de recherche. Ils nous permettent de mettre en application des connaissances acquises pendant l'année et de les perfectionner autour d'un sujet complet, de l'élaboration du cahier des charges à une éventuelle réalisation finale.

Nous avons donc pu nous pencher sur un problème concret pendant trois mois. Le sujet consistait en la réalisation d'un réseau de capteurs en wifi pour surveiller de grandes structures et de faire remonter des informations d'environnement telles que la température d'une salle, son éclairement ou l'éventuelle ouverture ou fermeture d'une porte.

De telles données peuvent éventuellement être traitées en amont dans une logique d'économie d'énergie, ou tout simplement de surveillance de l'état d'un bâtiment.

1 Cahier des charges

Le cahier des charges a été formulé comme il suit : L'objectif du projet consiste en la conception et la réalisation de capteurs autonomes communiquant en WiFi afin de pouvoir remonter réguliérement des informations sur l'état des salles de cours. Les capteurs seront par exemple des détecteurs de lumières, de pression, de qualité de l'air, ...

La communication sera obligatoirement réalisée en WiFi sur le réseau de l'université en respectant les contraintes de sécurité (WPA2, ...).

Deux options sont possibles :
— Refaire complètement une carte avec un microcontroleur et une puce wifi (comme par exemple les spark)
— Réaliser un shield pour raspberry pi contenant les différents capteurs.

2 Présentation de la réalisation

2.1 Présentation de la partie web

La partie Web du projet offre une interface de supervision et de monitoring des capteurs. En effet, nous avons pense qu'il ne suffisait pas de réussir à envoyer des données en Wifi par le biais des capteurs, il fallait aussi pouvoir récupérer et stocker ces informations en vue d'un potentiel traitement.

On a donc décidé de récupérer ces informations dans une base de données qui serait gérée par une plateforme Web.

2.1.1 La base de données

La base de données que nous avons utilisée est celle offerte par Polytech'Lille, comme à chaque étudiant de l'école. Nous avons donc opté pour une base MySQL avec le panneau d'administration PHPMyAdmin. Le choix de MySQL est simplement justifié par une meilleure connaissance et habitude des requêtes et commandes SQL, contrairement à PosgreSQL (bien qu'il y ait peu de différences).

2.1.2 La plateforme Web

L'accès à la plateforme se fait via l'url suivante : Smart Sensor Wifi.
Le site n'est pas en mode sécurisé (https) pour une raison simple : le shield Wifi ne supporte pas l'https dans le firmware actuel (celui livré de base). C'est donc pour cela que nous avons choisi de le mettre en http simple (bien qu'il puisse être basculé en https très rapidement et simplement sans poser aucun souci de compatibilité).

Au niveau langage de programmation, le site est en HTML et PHP. Nous avons choisi

le PHP, plutôt que le Python ou autre, car le PHP était plus familier et offrait toutes les opérations nécessaires. De plus, une petite partie du site utilise du Javascript afin de rendre plus dynamique et interactif le site.

Au niveau design, nous avons décidé d'utiliser deux frameworks libres : Ink et Chart.js. Le premier gère le design global du site et le second est utile à la création de graphiques. Nous avons décidé d'utiliser des framework afin de rendre le site plus propre et d'offrir à l'utilisateur un outil visuellement agréable à utiliser.

Au niveau technique, la plateforme est un simple intermédiaire entre, d'une part les capteurs et la base de données, et d'autre part, les utilisateurs et la base de données. D'un côté, le site récupère les informations que les capteurs lui envoie et stocke ces données dans la base. De l'autre côté, à travers différents boutons et champs, l'utilisateur peut récupérer les informations qu'il désire.

2.2 Présentation de la partie capteur

2.2.1 Le hardware

A la fin du projet, nous avons réalisé un prototype fonctionnel de ce que pourrait être le capteur. Il est important de noter que nous n'avons pas cherché à intégrer la solution mais plutôt à la développer en terme de fonctionnalités. De ce fait le prototype est assez volumineux mais sa taille peut aisément être réduite en utilisant des solutions à base de composants CMS notamment ce qui permettrait de réduire de manière significative l'encombrement de l'appareil.

Le prototype final se compose donc de trois éléments principaux. Le coeur du système est réalisé avec un Arduino Uno. Une platine de développement sur microcontroleur basée sur un ATmega328P de chez Atmel. C'est dans cet Arduino qu'est flashé le programme du capteur. Vient en suite la partie qui prend en charge le WiFi, il s'agit d'un shield WiFi de chez GoTronic basé sur la puce WizFi210 de WizNet, cette dernière est interfacée avec l'Arduino via le port série. Enfin la dernière partie est un protoshield sur lequel est placé une breadboard sur laquelle sont enfichés les composants avec lesquels l'Arduino peut interagir, à savoir :

— Une photorésistante pour capter la luminosité
— Un capteur TMP36 de température
— Une RTC (Real Time Clock) DS1302

Notons que nous avons choisi ces capteurs pour leur facilité d'utilisation, mais dans l'absolu rien n'empêche d'utiliser n'importe quel autre capteur analogique ou TOR, ni même d'interfacer le système avec des choses plus complexes à base de SPI ou d'I^2C, étant donné que l'ATmega328P en a la possibilité.

Le principe de fonctionnement du capteur est très simple. Une fois configuré à l'aide de `configurator` (dont le fonctionnement sera détaillé ci après) il vous suffit de déployer le capteur là où vous souhaitez qu'il opère et le laisser envoyer des mesures à intervalles d'une minute. On dispose ainsi d'un suivi efficace de l'environnement. Aussitôt que l'appareil est mis sous tension il va essayer de se connecter au réseau qui lui a été spécifié en configuration et une fois que cela s'est fait correctement il va passer en mode interruptif et enverra ses mises à jour toutes les minutes. Il faut aussi noter que pendant ce temps là, le capteur écoutera les connexions entrantes sur le port 80 et pourra ainsi être reconfiguré par ce biais à l'aide d'un set de commandes AT décris et documenté en annexe.

2.2.2 Le software

La partie logicielle du projet est découpée en deux parties. La première consiste en le firmware que nous avons chargé dans l'Arduino, et qui est chargé de gérer la carte WiFi et les différents capteurs.

La seconde partie consiste en un logiciel de configuration écrit spécialement pour ce projet, il s'agit de `configurator`. C'est un script en python qui permet via une interface graphique plutôt intuitive de configurer un capteur, et ce que ça soit par le port série d'un capteur directement relié au PC en USB ou via le réseau en utilisant une socket TCP.

3 Détail de la partie web du projet

Cette partie aura pour but de présenter la partie Web du projet sous un oeil technique. On présentera donc en détail les méthodes utilisées ainsi que les choix techniques qui composent notre projet.

3.1 La base de données

La base de données, qui est une base MySQL comme dit précédemment, se compose de 3 tables. De base, nous avions opté pour 2 tables : l'une pour stocker les utilisateurs de la plateforme et leur mot de passe relatif, l'autre pour stocker les capteurs et la mise à jour des données.

Ce modèle était valide pour nos tests de début de projet mais s'est montré très limité quand nous avons décidé d'analyser les données des capteurs et spécialement quand nous avons décidé de stocker dans la durée les données. Par exemple, nous avions besoin de plusieurs données d'un même capteur pour dessiner un graphe ou plus simplement comparer différentes mesures. Comme nous stockions uniquement la dernière mesure (les informations étaient écrasées à chaque mise à jour), nous ne pouvions stocker plusieurs mesures d'un même capteur. Nous avons donc changé notre implémentation des tables dans la base.

Nous aurions pu garder seulement 2 tables, en insérant simplement les nouvelles données dans la table des capteurs au lieu de mettre à jour les champs correspondant. Cette idée n'a pas pu être mise en place car cette table possède un champ 'ID' qui est la clé primaire de la table. Or il aurait fallu avoir plusieurs entrées comportant le même 'ID', ce qui est impossible.

C'est donc à ce moment là que la table des capteurs est devenue la table répertoriant la liste des capteurs et une nouvelle table stockant toutes les données des capteurs a été crée.

Les relations entre ces différentes tables sont schématisées sur la figure suivante et détaillées ci dessous :

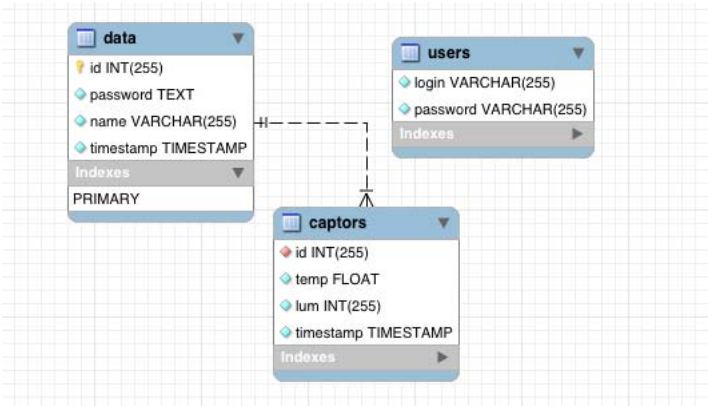


Figure 1 – diagramme des tables

La table 'users' répertorie les différents utilisateurs de la plateforme Web. C'est à travers elle qu'un utilisateur est autorisé à se connecter, sous réserve que son nom et son mot de passe sont corrects.

La table 'data' est la table répertoriant les différents capteurs. C'est à travers elle qu'un capteur est autorisé à mettre à jour ses données. Elle stocke aussi la date de dernière mise à jour du de chaque capteur.

La table 'captors' est la table stockant les données de chaque capteur. C'est en quelque sorte l'extension de la table 'data'. Chaque champ, défini par son 'ID' ou son 'Name' est relié à une entrée de la table 'data' et complète cette dernière.

3.2 La plateforme

Le site se découpe en 4 modules (3 pour un utilisateur lambda) dont un visuel se situe en annexe. Pour construire ce site, nous avons utilisé différents langages et outils qui seront développés ci dessous.

3.2.1 Les différents modules

Le site Web contient 4 modules différents qui ont chacun une fonction particulière. Le premier est celui qui apparait en premier sur le site. Il autorise la connection à la plateforme. Une fois l'utilisateur connecté, il affiche les 5 dernières mises à jour des capteurs.

Le deuxième module permet la recherche d'informations sur un capteur. La recherche s'effectue soit par 'ID' soit par 'Nom'. Une fois le capteur choisi, toutes les données disponibles le concernant s'affiche ansi que deux graphiques des 10 dernières mesures (un pour la température et un pour la luminosité).

Le troisième module regroupe l'ajout et la suppression des capteurs. Il est nécessaire de déclarer et autoriser les capteurs en vue de leur mise à jour car si les capteurs ne sont pas répertoriés dans la base, leurs informations ne seront pas prise en compte (permet de sécuriser la mise à jour d'informations au cas où un tierce voudrait compromettre la base de données). Si le capteur n'est plus utilisé, on le supprime de la base à travers cette interface.

Le dernier module n'est accessible qu'à l'administrateur de la plateforme. Il regroupe les fonctions d'administration : il permet l'ajout ou la suppression d'un utilisateur, la suppression complète des données des capteurs (en vue d'une remise à zéro) et un panneau de supervision. Ce dernier permet de repérer quels capteurs n'ont pas effectué leur mise à jour (en prenant comme référence la dernière heure).

3.2.2 Le design : Ink

Pour cette plateforme, nous avons voulu faire quelque chose de visuellement propre. C'est donc dans cette optique que nous avons décidé d'utiliser un framework css. Ink est le framework libre qui s'occupe du design de notre site. Il nous a permis de donner un style simple à notre site, une fois maitrisé.

La principale difficulté a été de comprendre comment fonctionnait Ink. En effet, en dehors des classes qui lui sont propres, il utilise un système de grille. Chaque morceau du site est

découpé et se positionne sur la grille. Ce système structure le site et offre une interface épuré avec de nombreuses possibilité déjà implanté.

L'utilisation de Ink se justifie par la qualité visuelle que l'on n'aurait pas pu atteindre par nous même, par manque de temps et certainement de talent graphique.

3.2.3 La gestion des données : PHP et Requêtes

L'interaction entre l'utilisateur et le base de données est assurée par PHP. C'est lui qui s'occupe de récupérer, mettre à jour et supprimer les informations demandées ou envoyées par l'utilisateur et le capteur. Il permet aussi d'executer du code en fonction de différents paramètres. C'est lui qui va être le moteur de l'application.

Dans un premier temps, il permet d'executer et d'afficher sur la page ce qui est demandé par l'utilisateur, par exemple si l'utilisateur n'a pas encore demandé d'informations sur un capteur, on lui propose de faire un choix entre différents capteurs. Une fois qu'il en a sélectionné un (c'est à dire après l'envoi en POST du choix), PHP récupère cette information et prépare la page pour répondre à la demande de l'utilisateur. Pour illustrer cette fonction, on prend pour exemple le code suivant :

```
/*Si on a recu une information d'ID, c'est a dire que l'utilisateur
demande une info sur une ID*/
if(isset($_POST['id'])){
        //On execute les actions appropriees
}
```

La deuxième utilité de PHP est de pouvoir interagir directement avec la BDD. En utilisant PDO, on va se connecter à la base et executer des requêtes qu'on récupérera avec PHP. PDO est une méthode orientée objet d'interaction avec différents types de BDD (Mysql, PosgreSQL, etc...) qui est destiné à être la méthode principale à utiliser dans les prochaines versions de PHP. Le choix de PDO se justifie par sa capacité à préparer les requêtes avant de les executer (on execute les requêtes en fonction de paramètres et non plus statiquement) et par l'importance qu'il prendra dans les versions futures de PHP (il est voué à être la méthode principale de connection, ce qui assure que l'application fonctionnera avec les versions futures).

PDO fonctionne de la manière suivante :

```
/*On instancie une variable PDO en PHP contenant
toutes les informations de connexion a la base de donnees*/
$bdd = new PDO('mysql:host=hostofthedatabase;dbname=nameofthedb',
'user', 'password');
/*On prepare la requete avec en option un parametre
represente par "?"*/
$check=$bdd->prepare("SELECT login, password FROM users WHERE
login=?");
/*On execute la requete avec comme parametre "Jesuisunutilisateur"*/
$check->execute(array(Jesuisunutilisateur));
/*On recupere le resultat de la requete dans une variable pour une
```

```
utilisation ulterieure*/
$data=$check->fetch();
/*Une fois fini, on ferme proprement la connection a la base de
donnees*/
$check->closeCursor();
```

Afin de gérer correctement les échanges d'informations entre le site et la base de données, quatre opérations différentes sont utilisées par PHP/PDO en utilisant le format MySQL :

— Insérer des données dans la BDD :

```
Ex: INSERT INTO captors(temp,lum,timestamp,id) VALUES(?,?,?,?)
```

— Obtenir des données de la BDD :

```
Ex : SELECT login, password FROM users
```

— Mettre à jour des données dans la BDD :

```
Ex : UPDATE data SET timestamp=? WHERE id=?
```

— Supprimer des données dans la BDD :

```
Ex : DELETE captors FROM captors WHERE timestamp < ?
```

3.2.4 Recup.php

La plus grosse partie du site consiste à faire interagir l'utilisateur avec la base de données. Cependant, il existe une page de site (recup.php) qui n'interagit pas avec l'utilisateur mais avec les capteurs. Celle ci reçoit des informations des capteurs (via POST), vérifie si le capteur a le droit de mettre à jour les données et effectue l'insertion dans la base.

Cette page n'a pas seulement l'utilité de mise à jour, elle s'occupe aussi de la conversion des données de température. On s'est aperçu que les mesures brutes des capteurs venant de l'adc devait être multipliées par un coefficient (0,647) pour obtenir une valeur cohérente en température. Cette multiplication n'étant pas faite par l'Arduino (pour le soulager un peu), on a choisi de donner cette tâche à PHP.

En dernier lieu, cette page se charge de nettoyer la table de données des capteurs en supprimer toute entrée datant de la veille via la fonction Date de PHP.

3.2.5 Du dynamisme : Javascript

Le Javascript sur le site est utilisé pour donner dynamiser l'utilisation du site. Il nous a permis de rediriger automatiquement l'utilisateur et principalement de gérer l'ajout de champ dans les pages d'ajout et de suppression de capteurs. En effet, nous l'avons utilisé pour permettre d'effectuer plusieurs actions identiques en une seule fois.

Par exemple, quand on veut ajouter en chaine plusieurs capteurs, on aurait pu les ajouter un par mais cela aurait alourdi grandement l'application. La solution est simple, à chaque clic

sur un bouton (dans notre cas, un "+"), le javascript insère un nouveau champ sur la page. Cela se traduit par le code suivant :

```
// On recupere le cadre ou on va inserer le nouvel element
var div = document.getElementById('champs');

// Fonction qui va creer l'element
function addInput(name,placeholder){
        var input = document.createElement("input");
        input.name = name;
        input.placeholder = placeholder;
        div.appendChild(input);
}

// On ajoute l'element a la page
        function addField() {
        div.appendChild(document.createElement("br"));
        addInput("name[]","Nom du capteur");
        addInput("password[]","Mot de passe du capteur");
}
```

3.3 Des idées d'amélioration

Plusieurs améliorations peuvent être apportées à la plateforme Web, certaines avaient déjà été pensées au début du projet mais non implantées.

En premier lieu, le pattern MVC aurait du être celui de base mais par manque de temps et surtout par sa complexité (principalement avec l'utilisation parallèle d'Ink), il n'a pas été utilisé.

Par ailleurs, le site utilise le Javascript mais pourrait l'utiliser plus comme par exemple en rafraichissant la page de recherche d'informations pour permettre d'afficher les dernières informations et le graphique sans devoir recharger manuellement la page.

En dernier lieu, on pourrait adapter les différents champs de la base (luminosité, température, etc) ainsi que le code selon les besoins réels (capteur de luminosité, température, pression, etc). On pourrait aussi optimiser le nombre de mesures des graphiques mais cela relève plus de l'optimisation que de l'amélioration.

4 Détail de la conception de la partie microcontrôleur du projet

4.1 Le capteur

Lors de la réalisation du senseur il a fallu en premier lieu se fixer des objectifs à atteindre, afin d'être sûr d'aller dans une direction cohérente tout au long du projet. De ce fait nous avons choisi d'opter pour la réalisation d'un prototype fonctionnel et relativement simple au début. Par relativement simple, nous entendions capable de lire un ou deux capteurs analogiques simples à mettre en place de manière à nous focaliser sur l'interfaçage du dispositif avec un serveur de données. Nous nous sommes donc d'avantage focalisés sur une approche plus logicielle du projet en écartant volontairement, par manque de temps, des considérations annexes comme la gestion de l'énergie ou la mise en place immédiate de capteurs très complexes. Le but était réellement de fournir une base saine sur laquelle il serait facile de s'appuyer à l'avenir.

4.1.1 La partie électronique

La partie électronique est la plus simple de l'appareil c'est donc celle là qui sera abordée en premier. Elle se compose simplement d'une photo résistance montée en pull up sur la patte analogique 1 de l'Arduino et d'un capteur de température câblé sur la patte analogique 2. Nous avons également câblé un bouton de mise à jour (dont l'utilité sera détaillée plus tard) sur la patte PD2 ainsi que la real time clock sur trois pattes numériques de manière à pouvoir échanger des données. Tout ces composants ont été positionnés sur une breadboard de manière à ce que nous puissions facilement en modifier l'arrangement au fur et a mesure que nos besoins évoluaient dans le projet.

Nous n'avons malheureusement pas eu le temps de réaliser de carte pour fixer ce design. Et il n'aurait pas été pertinent de le faire dans la mesure ou la conception, le routage et la fabrication d'une carte sont des activités pour le moins chronophages et qu'il a effectivement été plus productif pour nous de nous concentrer sur l'ajout de nouvelles fonctionalités au capteur. Cependant nous avons édités des PCB et des schémas électroniques disponibles en annexe qui permettent de réaliser une telle carte.

4.1.2 Conception du firmware de l'Arduino

La conception du firmware de l'arduino s'est découpée en plusieurs étapes. Pour commencer nous avons séparé les différentes fonctionnalités que nous souhaitions. D'un point de vue organisation cela s'est traduit par un fractionnement du code en plusieurs fichiers, chacun regroupant les fonctions relatives à un domaine, par exemple la gestion du port série, des CAN, de l'EEPROM et autre.

Ensuite, une fois que la séparation logique des fonctionalités a été effectuée nous avons commencé à les développer une à une, en lançant à chaque fois une série de tests sur ce que nous avions créé, de manière à nous assurer que tout était bien conforme à ce que nous attendions. De cette manière nous pouvions réutiliser sans crainte le code que nous avions précédemment écrit puisqu'il avait été validé par une batterie de tests.

Une attention toute particulière a été portée à la documentation. En effet si nous souhaitons que le projet puisse être repris les années suivantes il est indispensable de fournir une documentation valable, fiable et complète de manière à ce que n'importe qui ou presque soit en mesure de comprendre ce qui a déjà été réalisé de manière logicielle. Pour ce faire nous avont eu recours au système de documentation in-code Doxygen, qui grâce à un système de commentaires au formattage particulier permet de générer automatiquement une belle documentation HTML. Cette documentation navigable est facile à utiliser, avec outils de recherche intégré et liens entre les différentes fonctions documentées. Si vous souhaitez construire la documentation du programme rendez vous dans le répertoire Arduino et tapez simplement `doxygen` un dossier contenant le nécessaire sera généré pour vous !

La seconde partie du projet, qui n'est au final même pas utilisée dans le prototype actuel, a été la réalisation d'un driver pour la real time clock de Maxim DS1302. Ce driver a été implémenté en utilisant le protocole de communication 3-Wire décrit dans la datasheet, c'est un protocole similaire au protocole SPI mais suffisamment différent de celui ci pour ne pas pouvoir utiliser le contrôleur SPI de l'Arduino. Le driver est fonctionnel et permet d'accéder aux registres de temps de la RTC ainsi qu'aux quelques octets de RAM qu'elle nous fourni en plus, ce qui peut être pratique si jamais l'arduino venait à en manquer. Toutes les informations nécessaires sont bien évidemment documentées dans le code et via Doxygen.

Nous somme partis du constat que le fonctionnement du WiFi pouvait être parfois assez instable, et que de ce fait il serait appréciable que le capteur puisse se comporter comme une sorte de datalogger dans le cas où il serait dans l'incapacité momentanée de communiquer avec le serveur de donnée, que cela soit a cause du wifi ou d'un autre problème d'ailleurs. Pour cela nous avions pensé le coupler avec une flash (voir les schémas en annexe). Cela dit comme le composant n'était disponible qu'en format SOIC8 et que nous n'avons pas réalisé la carte il nous a été impossible de l'intégrer correctement. Une amélioration possible et très facile à mettre en oeuvre pour le projet serait donc l'intégration de cette flash et l'utilisation de la RTC pour soumettre en différé des données de mesures qui n'auraient pas pu être transmises à temps.

La troisième partie du projet a été consacrée à faire en sorte que la configuration du capteur soit persistante entre les reboots. Durant les phases de développement nous ne nous étions pas vraiment occupés de cela mais dès que le programme est devenu plus compliqué il est devenu essentiel de stocker certains paramètres (d'identifications notamment) autrement qu'en dur dans le code. Il a donc fallu ajouter un module de gestion de l'EEPROM. La structure suivante a donc été choisie pour stocker des données :

— L'EEPROM est découpée en différents secteur de taille fixé à la compilation
— Chaque secteur contient un octet de taille et N octets de données ($N < 255$)
— Chaque secteur est uniquement défini par sa position en mémoire

Cette solution présente l'avantage d'être très simple à mettre en place d'un point de vue logiciel, ce système souffre cependant d'un gros problème qui est que l'on ne peut pas connaître de manière dynamique la taille maximale des données que l'on peut ranger dans un segment. Ce qui implique que le programmeur fasse très attention aux données qu'il manipule, de manière à ne pas provoquer de débordement sur un autre segment. Ceci dit, ce problème n'est pas vraiment critique puisqu'une simple réécriture de la configuration rétablira tout correctement. Ceci fait également partie des améliorations possibles du projet.

La quatrième partie du projet a sans aucun doute été la plus difficile à réaliser. En effet après s'être occupé des briques élémentaire qui compose le capteur (communiquer en série, lire des capteurs analogiques, lire la RTC). Il a fallu regrouper tout ça ensemble et l'envoyer sur internet. Pour se faire, il a d'abord fallu comprendre comment fonctionnait la carte WiFi. La

datasheet n'étant pas toujours très claire, ni très complète il a d'abord fallu passer par une phase « d'exploration » de la carte via le port série, de manière à comprendre précisément ce qu'elle attendait de nous en terme de commandes. La carte se configurait finalement par un set de commandes AT via le port série et malgré les quelques imprécisions de la datasheet nous avons finalement été capables de nous identifier sur un réseau spécialement créé pour l'occasion.

A partir de là a été développé une petite bibliothèque qui intègre les fonctions réseau de base, à savoir s'identifier sur un réseau avec des paramètres donnés, soumettre des requêtes HTTP et vérifier le retour des commandes AT de manière à réitérer plusieurs fois une tentative de connexion qui aurait échoué. Cette bibliothèque fonctionne sur le port série via une scrutation, elle n'utilise pas d'interruptions pour avoir plus de contrôle et intercepter tout ce qui circule sur le port série et le conserver au sein du même contexte (il est préférable que le retour d'une commande reste interceptée par la fonction qui l'a émise plutôt que de devoir passer par une interruption globale et deviner quelle fonction a besoin des données reçues).

4.1.3 Communication avec l'application web

Une fois la carte configurée et connectée au réseau, il a fallu envoyer des données de mise à jour au site de Benoit. Nous avions au départ pensé à faire ça via post en JSON, simplement ça aurait été trop long et gourmand à mettre en place niveau mémoire (les données post doivent avoir une longueur définie et il faut encoder les caractères " :," qui sont très utilisées dans une chaine JSON ce qui aurait rendu l'envoi des données affreuse). Du coup nous avons opté pour un format d'envoi en post classique. Voici par exemple la requête POST que le capteur "cap1" doté du mot de passe "coucoutuveuxvoirmadata" aurait envoyé :

```
POST /recup.php HTTP/1.0
Host: smartsensorwifi.plil.net
Content-type: application/x-www-form-urlencoded
Content-length: 100

temp=140&lum=17&mid=cap1&mpass=coucoutuveuxvoirmadata
```

Notons que les valeurs envoyées peuvent paraître étranges, mais elle ne sont que la lecture directe du CAN de l'ATmega, la serveur les traitera pour les transformer en température ou déterminer le seuil de luminosité. Ainsi on décharge le microcontrôleur d'une partie du travail.

4.1.4 Reconfiguration du capteur

Une fois que le projet est capable de communiquer avec l'appli web, de soumettre des données, de se connecter à un réseau et ce genre de chose il ne lui manque pas beaucoup de fonctionnalités. Ceci dit une fonctionnalité indispensable s'est vite faite ressentir : le besoin de pouvoir reconfigurer le capteur à distance. En effet, le seul moyen de reconfigurer le capteur était en modifiant en dur dans le code les identifiants et de reflasher le programme ensuite, ce qui n'est pas pratique.

Pour palier à ce problème nous avons donc créé un set relativement complet de commandes AT, documentées en annexe. L'idée est que le capteur écoute ce qu'il se passe sur son port série

et dès qu'une commande AT est reconnue elle est exécutée pour peu que l'on soit authentifié sur le capteur. Pour se faire nous avons établi deux modes de configuration, une configuration "locale" via le port série, et une reconfiguration à distance via le réseau.

4.1.5 Principales difficultés rencontrées

4.2 Configurator.py

Afin que l'utilisateur n'ai pas a se munir de minicom ou de netcat pour reconfigurer ses capteurs, nous avons choisi de développer un logiciel dont le rôle est de se charger de la configuration à la place de l'utilisateur. Qu'il s'agisse d'une configuration locale via le port série ou à distance via le réseau, configurator fait le lien entre l'utilisateur et son capteur. Le logiciel dispose d'une interface très simple composée de quelques champs et de deux boutons, un pour rapatrier la configuration d'un capteur et l'autre pour envoyer une configuration à un capteur. De cette manière il n'est plus utile de connaître par coeur les commandes AT nécessaires à la configuration de l'appareil. Ce logiciel a été réalisé en python avec pygtk, glade2 et pyserial.

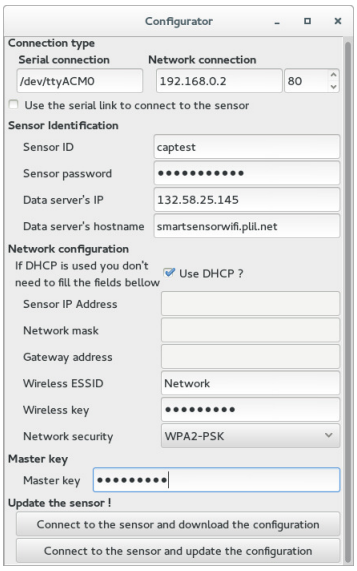


Figure 2 – Interface de configurator

4.3 Tentative de substitution par une solution a base de RaspberryPi

Comme la carte WizFi tardait à arriver, nous avons pensé opter pour une solution alternative pour réaliser le capteur, à base d'une RaspberryPi. Certes la solution était un peu surdimensionnée pour ce que l'on voulait en faire, mais comme cela on aurait eu quelque chose de fonctionnel à montrer lors de la soutenance. L'idée était d'utiliser les GPIO de la Raspberry pour l'interfacer avec le monde exterieur.

Malheureusement la RaspberryPi est un appareil qui ne dispose pas d'entrées analogiques, ce qui est assez peu pratique pour lire une température ou une luminosité de la même manière que sur un ATmega. Nous avons donc dû recourir à un ADC externe fonctionnant suivant le protocole SPI, ce qui tombait plutôt bien puisque la Raspberry est équipée d'une inetrface SPI lui permettant de contrôler jusqu'à deux esclaves. Pour notre problème nous avons opté pour l'ADC MCP3208 de MicroChip. Pourquoi celui là ? Tout simplement parce qu'il dispose de huit canaux analogiques ce qui permet une grande variété de capteurs, mais également parce qu'il peut fonctionner en 3v3, tension à laquelle fonctionne également la Raspberry, ainsi on évite de la griller.

J'ai donc réalisé une carte et un PCB pour réaliser cette alternative, mais également la couche logicielle qui va avec. J'ai en effet développé une classe en C++ qui encapsule tout les appels systèmes permettant d'écrire et de lire sur le port SPI de la raspberry, en full duplex. J'ai en réalité réalisé deux classes, une classe SPI qui est la classe mère, comprenant seulement les routines d'initialisation et d'écriture du port SPI, et une SPI_MCP3208 qui comprend les commandes spécifiées dans la datasheet du composant.

Pour information, cette classe et le programme d'exemples sont disponibles sur GitHub et la documentation est disponible à l'addresse spécifiée dans le wiki du projet.

Cela dit, il y a eu un problème, il était que les données renvoyées par l'ADC étaient totalement incohérentes. Le fait est qu'à luminosité égale, si on lançait 10 acquisitions nous pouvions obtenir des écarts de plus de 3000 (sur 4095 d'amplitude) entre deux valeurs, quand bien même la luminosité ne variait pas. Le plus déroutant dans tout cela c'est que le problème n'avait pas l'air de venir de notre logiciel, dans la mesure ou nous étions parfaitement capables, avec la même couche logicielle, de piloter d'autre appareils SPI fonctionnant en esclave (comme un afficheur 4x7 segments de chez SparkFun par exemple). La seconde chose la plus dérourante était que les données étaient cohérentes du point de vue du fonctionnement du composant tel que décrit dans la datasheet. Je m'explique, une trame classique de communication avec l'ADC se compose comme il suit (schématiquement) :

```
4 bits de commande | deux bits de sample | 12 bits de résultat MSB first
```

Le fait est que d'après la datasheet en continuant à envoyer des coups d'horloge au composant après le dernier bit de donnée, le composant va nous renvoyer le résultat de la conversion mais LSB first. On récupère donc le symétrique des données par rapport au LSB. Après avoir analysé plusieurs séquences nous avons pu constater que l'erreur ne venait pas de la manière dont nous récupérions nos données puisqu'elles étaient correctes du point de vue de la datasheet mais totalement aberrantes d'un point de vue physique.

L'erreur ne venait pas non plus du hardware puisqu'un rapide coup d'oeil à l'oscilloscope nous a permis de constater que la tension sur la patte de l'ADC évoluait bien comme elle le devait. Le mystère est resté entier puisqu'après nous avons reçu la carte WizFi et nous avons pu commencer à travailler dessus, cette solution a donc été abandonnée. Cela dit, nous avions déjà commencé à réfléchir aux technologies que nous aurions pu mettre en place pour réaliser la solution, qui auraient été :

— Un programme de lecture des données appelé par cron à intervalle régulier (un bête script bash qui lit le port SPI et conditionne les données en POST par exemple pour les envoyer au site de Benoît)
— Une application Web similaire à celle de Benoît mais ne concernant que les données locales de l'appareil, qui aurait permis de modifier les paramètres de mises à jour/sécurité tout simplement depuis un navigateur. Nous avions pensés à la réaliser en Python avec CherryPy un excellent framework python très léger, et évidemment Ink pour le design.

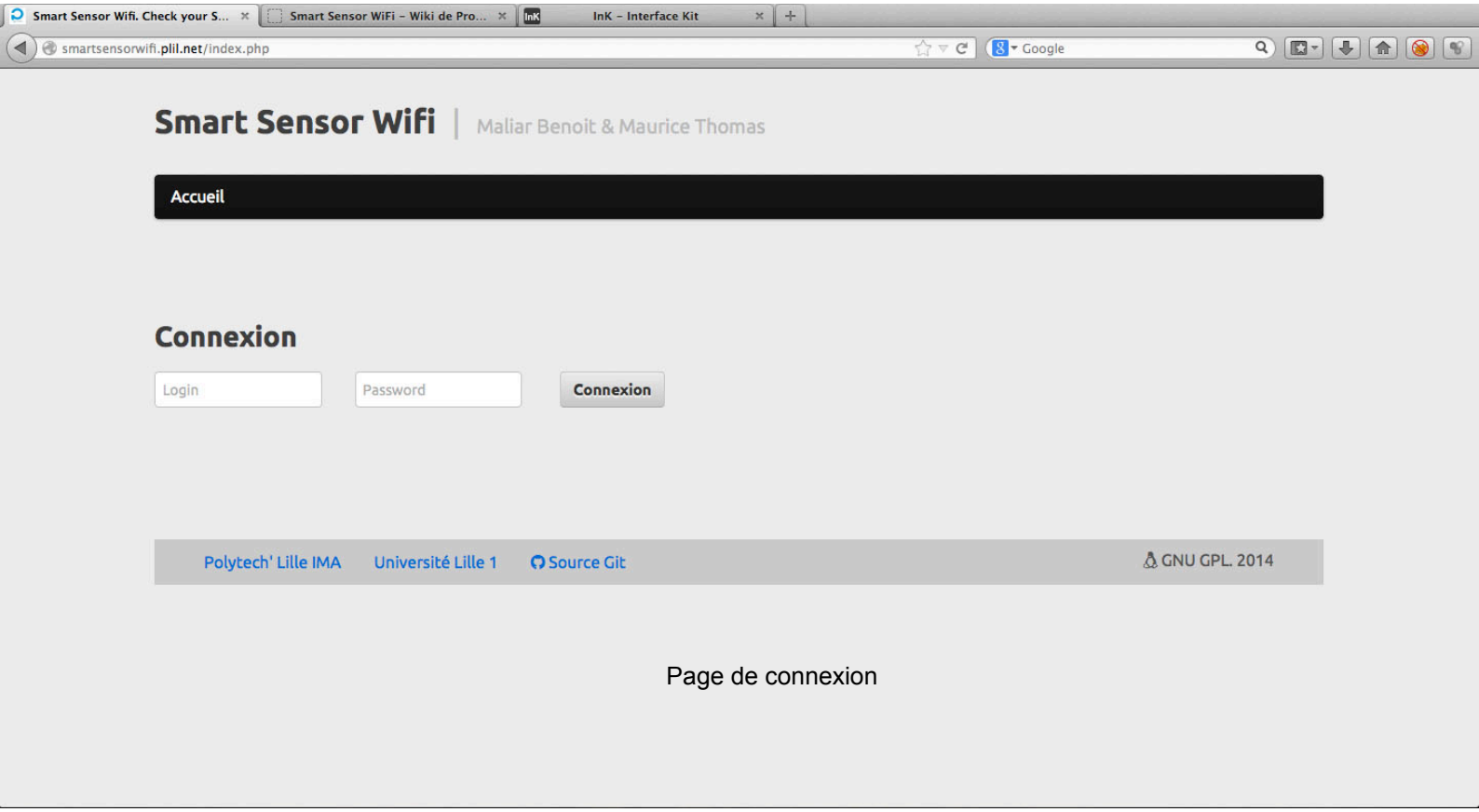
Conclusion

En conclusion, nous avons répondu à la problématique qui nous a été posée en fournissant un prototype de capteur qui envoie des données via le réseau Wifi. De plus, une plate-forme permettant de monitorer ces capteurs a été conçu.

Un des points auquel nous n'avons pas totalement répondu est la sécurité du réseau. A la base, nous devions fournir un capteur capable d'envoyer ses données via le réseau de l'université (PEAP) mais cela n'a pas été possible avec le firmware fourni de base mais qui est possible en obtenant le firmware auprès du concepteur du shield Wifi. Néanmoins, le prototype final est tout de même capable de fonctionner sous WPA2.

La façon dont nous avons réalisé le projet permet de continuer et d'améliorer assez aisément le projet en vu d'une évolution ou d'une mise en production.

Annexes



Page de connexion

Smart Sensor Wifi. Check your Sensor !

Smart Sensor Wifi. Check your S... | Smart Sensor WiFi – Wiki de Pro... | InK – Interface Kit

smartsensorwifi.plil.net/index.php

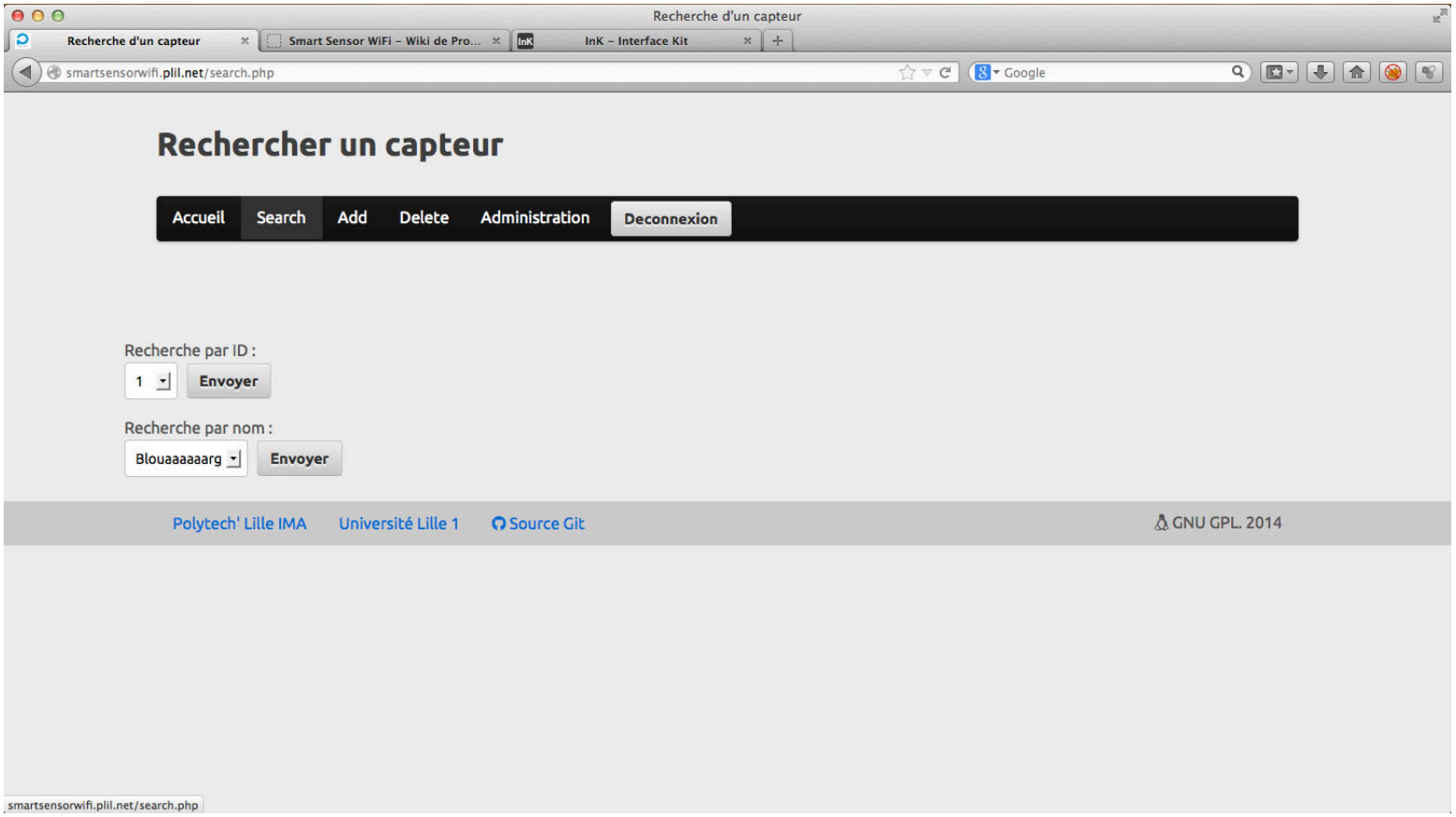
Smart Sensor Wifi | Maliar Benoit & Maurice Thomas

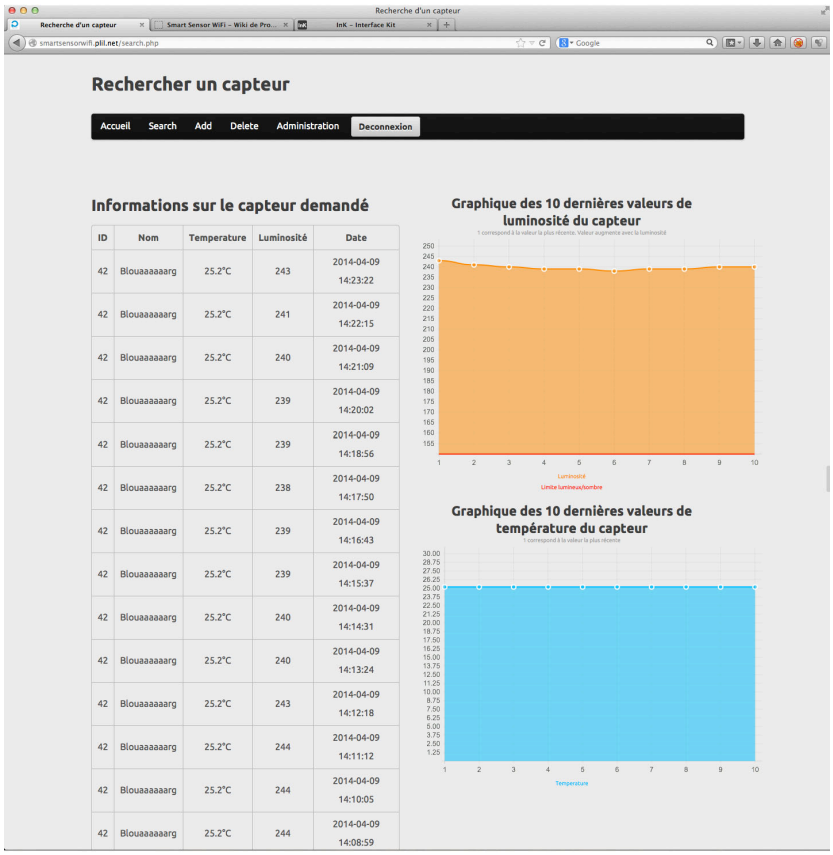
Une fois connecté...

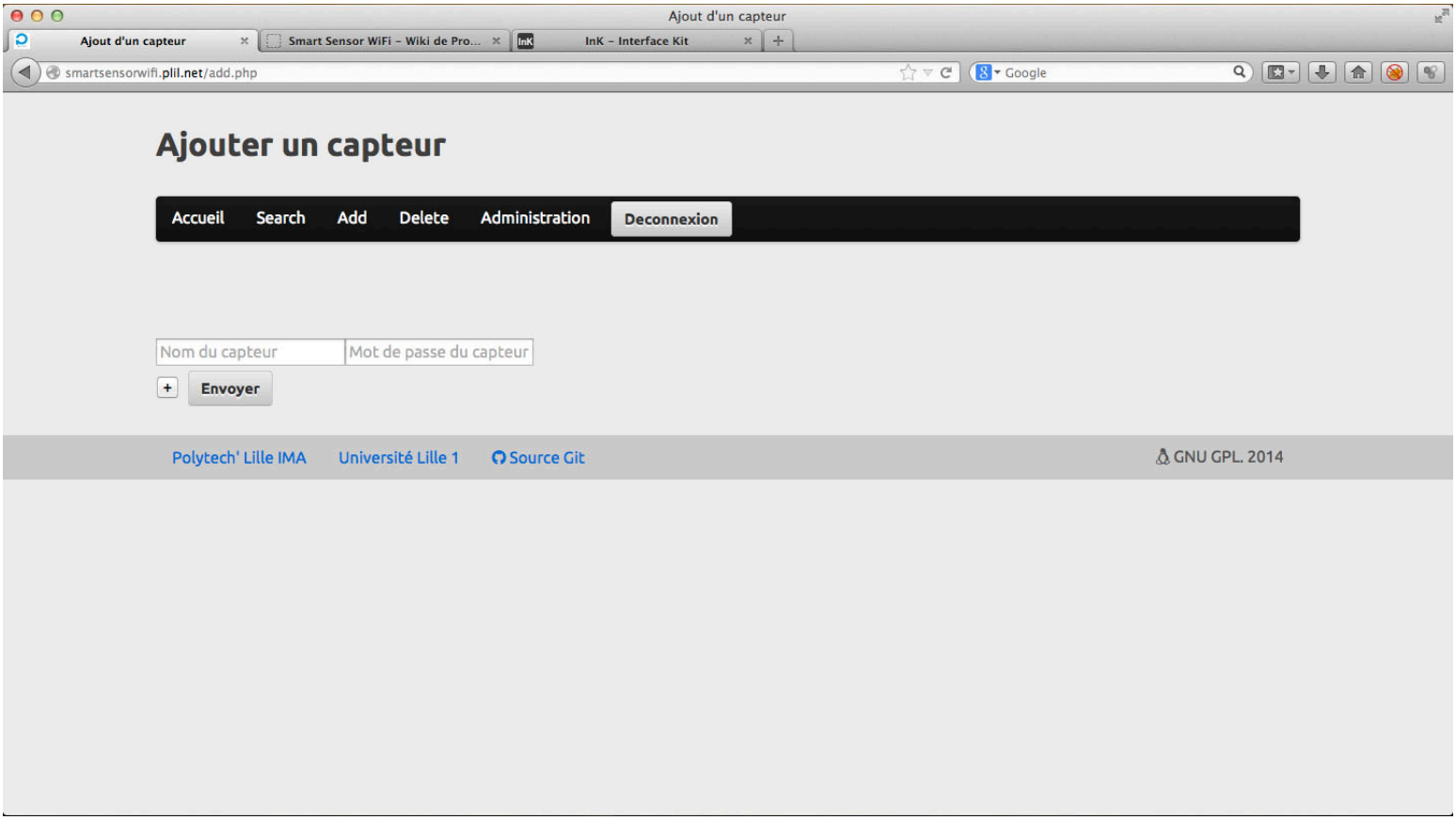
Accueil | Search | Add | Delete | Administration | Deconnexion

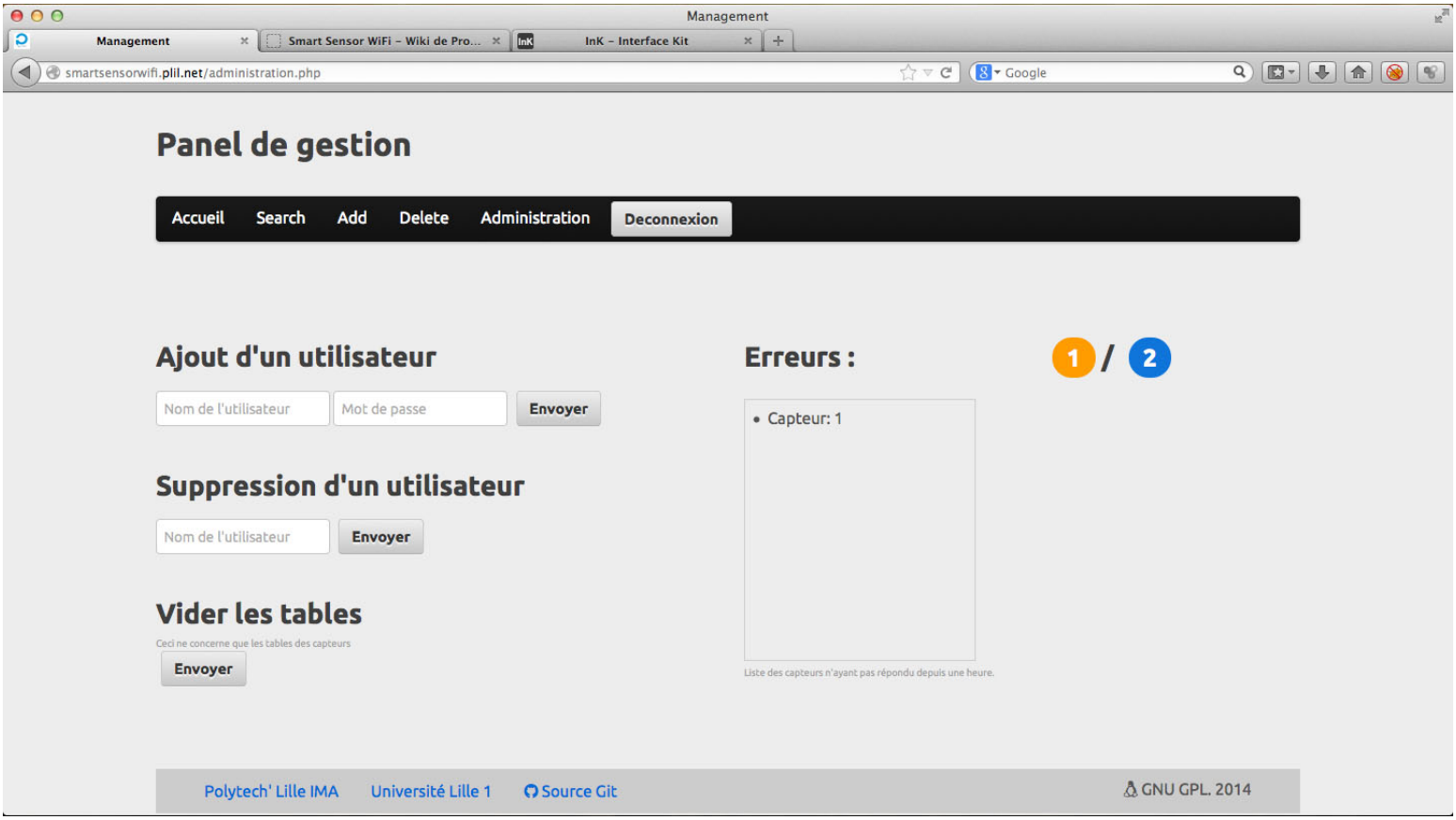
5 Derniers ajouts

ID	Nom	Temperature	Luminosité	Date
42	Blouaaaaaarg	25.2 °C	Lumineux (243)	2014-04-09 14:23:22
42	Blouaaaaaarg	25.2 °C	Lumineux (241)	2014-04-09 14:22:15
42	Blouaaaaaarg	25.2 °C	Lumineux (240)	2014-04-09 14:21:09
42	Blouaaaaaarg	25.2 °C	Lumineux (239)	2014-04-09 14:20:02
42	Blouaaaaaarg	25.2 °C	Lumineux (239)	2014-04-09 14:18:56


Recherche d'un capteur
Recherche d'un capteur
Smart Sensor WiFi – Wiki de Pro...
InK – Interface Kit
smartsensorwifi.plil.net/search.php
Google
Rechercher un capteur
Accueil
Search
Add
Delete
Administration
Deconnexion
Recherche par ID :
1
Envoyer
Recherche par nom :
Blouaaaaaarg
Envoyer
Polytech' Lille IMA
Université Lille 1
Source Git
GNU GPL. 2014
smartsensorwifi.plil.net/search.php


Recherche d'un capteur
Recherche d'un capteur
Smart Sensor WiFi – Wiki de Pro...
InK – Interface Kit
smartsensorwifi.plil.net/search.php
Google
Rechercher un capteur
Accueil
Search
Add
Delete
Administration
Deconnexion
Informations sur le capteur demandé
ID
Nom
Temperature
Luminosité
Date
42 Blouaaaaaarg 25.2°C 243 2014-04-09 14:23:22
42 Blouaaaaaarg 25.2°C 241 2014-04-09 14:22:15
42 Blouaaaaaarg 25.2°C 240 2014-04-09 14:21:09
42 Blouaaaaaarg 25.2°C 239 2014-04-09 14:20:02
42 Blouaaaaaarg 25.2°C 239 2014-04-09 14:18:56
42 Blouaaaaaarg 25.2°C 238 2014-04-09 14:17:50
42 Blouaaaaaarg 25.2°C 239 2014-04-09 14:16:43
42 Blouaaaaaarg 25.2°C 239 2014-04-09 14:15:37
42 Blouaaaaaarg 25.2°C 240 2014-04-09 14:14:31
42 Blouaaaaaarg 25.2°C 240 2014-04-09 14:13:24
42 Blouaaaaaarg 25.2°C 243 2014-04-09 14:12:18
42 Blouaaaaaarg 25.2°C 244 2014-04-09 14:11:12
42 Blouaaaaaarg 25.2°C 244 2014-04-09 14:10:05
42 Blouaaaaaarg 25.2°C 244 2014-04-09 14:08:59
Graphique des 10 dernières valeurs de luminosité du capteur
1 correspond à la valeur la plus récente. Valeur augmente avec la luminosité
Luminosité
Limite lumineux/sombre
Graphique des 10 dernières valeurs de température du capteur
1 correspond à la valeur la plus récente
Temperature


Ajout d'un capteur
Ajout d'un capteur
Smart Sensor WiFi – Wiki de Pro...
InK – Interface Kit
smartsensorwifi.plil.net/add.php
Google
Ajouter un capteur
Accueil
Search
Add
Delete
Administration
Deconnexion
Nom du capteur
Mot de passe du capteur
Envoyer
Polytech' Lille IMA
Université Lille 1
Source Git
GNU GPL. 2014


Management
Management
Smart Sensor WiFi – Wiki de Pro...
InK – Interface Kit
smartsensorwifi.plil.net/administration.php
Google
Panel de gestion
Accueil
Search
Add
Delete
Administration
Deconnexion
Ajout d'un utilisateur
Nom de l'utilisateur
Mot de passe
Envoyer
Suppression d'un utilisateur
Nom de l'utilisateur
Envoyer
Vider les tables
Ceci ne concerne que les tables des capteurs
Envoyer
Erreurs :
1 / 2
Capteur: 1
Liste des capteurs n'ayant pas répondu depuis une heure.
Polytech' Lille IMA
Université Lille 1
Source Git
GNU GPL. 2014

Communicate with the SmartSensors via AT-commands

Introduction

If you are familiar with the use of microcontroller devices such as XBee chips or so you migh know what the AT commands ar for : they enables you to configure your device via a textual mode. You are very lucky since the SmartSensors are shipped with a set of a dozen AT commands which enable you to configure it quite easily.

The main advantage of these AT commands is that they are accessible both in serial mode *as well as* in WiFi mode. That is to say that you can reconfigure your sensors remotely via the wifi, provided the sensor has already access to a network. Cool isn't it ?

To do so, the `util/` folder contains the **configurator.py** software wich can connect and update/download a sensor configuration. Just run it the fields/buttons are quite self explanatory.

Identification

To identify on the sensor, you have to know something called the **master key** of the sensor. This is a code that you must enter before trying to do anything on the sensor. The command for that is `AT+IDENT=masterkey`.

The commands

The generic syntax for an AT command is the following : `AT+COMMAND=ARG`. If you are coding something about it, don't forget to add an `\n` or an `\r` after the command.

Sensor configuration

These commands are used to configure the sensor itself and it's identification onto your database server.

- `AT+ID` : Changes the ID of the sensor to access the website
- `AT+PW` : Changes it's password
- `AT+IPDB` : Changes the IP of the data server
- `AT+HOSTDB` : Changes the hostname of the data server

Network configuration

- `AT+DHCP` : 1 if DHCP is used, 0 if not. If DHCP is used you don't need further config.
- `AT+NIP` : IP address of the sensor
- `AT+NGW` : Gateway IP
- `AT+NMSK` : Netmask

Wireless configuration

- `AT+ESSID` : Obviously sets the ESSID of the network. Does not matter if visible or not
- `AT+WKEY` : Sets the passphrase for the sensor
- `AT+NETTYPE` : Sets the type of network according to the following values :
  - 0 : Auto
  - 1 : Open authentication
  - 2 : WEP authentication
  - 4 : WPA PSK
  - 8 : WPA2 PSK

Security

- `AT+MASTER` : Changes the Master key.
- `AT+IDENT` : Identify on the sensor. The parameter is the master key.

End of sequence

- `AT+EXIT` : This command shall be issued at the end of every command sequence end.

Configurator

one software to conf them all

Introduction

If you succeeded to build a sensor and to upload a firmware onto it (GG) you are probably wondering "How the hell am I supposed to confiigure it in order to make it work with my network and with my server" aka "WTF?".

Well **configurator.py** is the answer ! This software is designed to allow you to configure a sensor device either from your computer through the network or through a serial link to perform the initial configuration.

How do I launch it ?

That's rather easy. You just fave to go into the util/configurator folder then run `python configurator.py`. Note that you have some dependencies to satisfy in order to run the software, namely :

- python (you don't say)
- python-serial
- python-gtk, python-gtk2 or pygtk depending on how the package is called in your distro

How do I use it ?

Well, the interface is quite simple actually. And the field are quite self explanatories. All you need to do is to fill in all the forms and hit the "Connect and upload" button. If some important fields are not properly filled then an error shall be raised and nothing will be uploaded. You will not break your config :)

If you just want to edit it, you can alternatively load the config from a sensor to your interface. Then identically just hit the "Connect and download" button.

Let's do that !

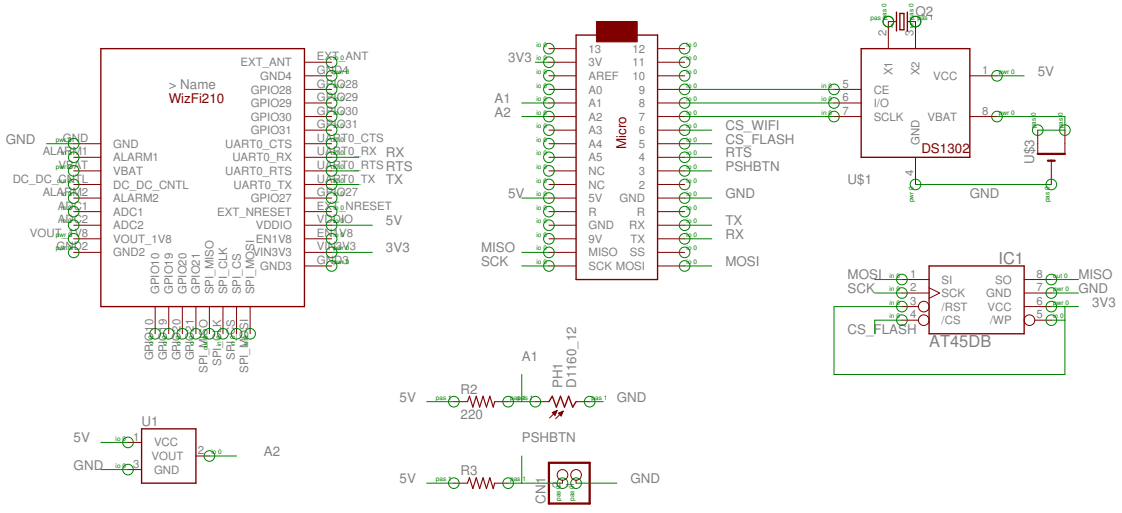
Yep, but before you get frustrated because "stupid software won't work" you have to know that an initial serial configuration requires a slight physical interraction with the device. That is to say you have to do two things : firstly you have to change the switch of the wifi shield from the "Run" to the "Prog" mode. Then while pressing the "Download" or "Upload" button you have to press the

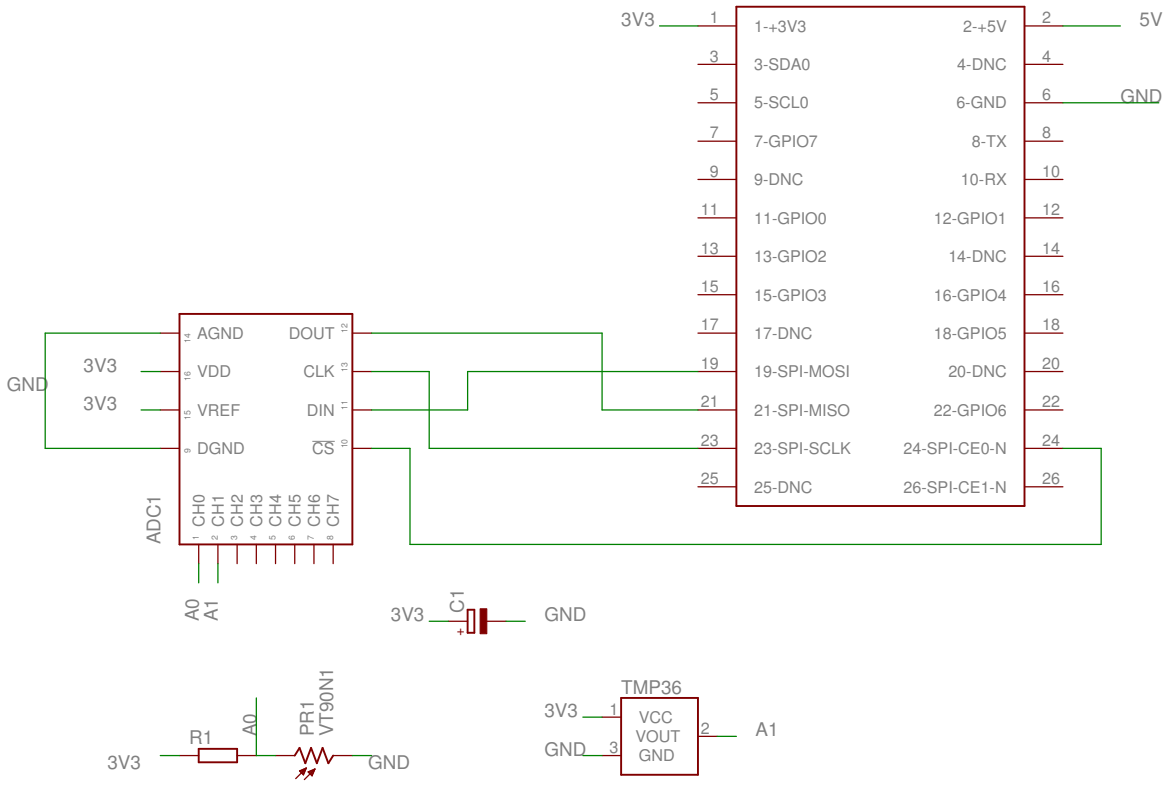
pushbutton located on the right part of the shield. This will allow the device to be serially configured.

To do this wirelessly you don't need such manipulations. Just fill in the address and here you go !

Master key

Note that you don't need to provide the master key if you are reprograming the software via the serial link.


> Name
WizFi210
EXT_ANT
GND4
GPIO28
GPIO29
GPIO30
GPIO31
UART0_CTS
UART0_RX
UART0_RTS
UART0_TX
GPIO27
EXT_NRESET
VDDIO
EN1V8
VIN3V3
GND3
GND
ALARM1
VBAT
DC_DC_CNTL
ALARM2
ADC1
ADC2
VOUT_1V8
GND2
GPIO10
GPIO19
GPIO20
GPIO21
SPI_MISO
SPI_CLK
SPI_CS
SPI_MOSI
RX
RTS
TX
5V
3V3
Micro
13
3V
AREF
A0
A1
A2
A3
A4
A5
NC
NC
5V
R
GND
9V
MISO
SCK
12
11
10
9
8
7
6
5
4
3
2
GND
RX
TX
SS
MOSI
3V3
A1
A2
5V
MISO
SCK
CS_WIFI
CS_FLASH
RTS
PSHBTN
GND
TX
RX
MOSI
Q2
X1
X2
CE
I/O
SCLK
VCC
VBAT
GND
DS1302
U$1
5V
GND
U$3
IC1
SI
SCK
/RST
/CS
SO
GND
VCC
/WP
MOSI
SCK
CS_FLASH
MISO
GND
3V3
AT45DB
U1
VCC
VOUT
GND
5V
GND
A2
A1
R2
220
PH1
D1160_12
5V
GND
PSHBTN
R3
5V
CNT
GND


3V3
1
1-+3V3
2-+5V
2
5V
3
3-SDA0
4-DNC
4
5
5-SCL0
6-GND
6
GND
7
7-GPIO7
8-TX
8
9
9-DNC
10-RX
10
11
11-GPIO0
12-GPIO1
12
13
13-GPIO2
14-DNC
14
15
15-GPIO3
16-GPIO4
16
17
17-DNC
18-GPIO5
18
19
19-SPI-MOSI
20-DNC
20
21
21-SPI-MISO
22-GPIO6
22
23
23-SPI-SCLK
24-SPI-CE0-N
24
25
25-DNC
26-SPI-CE1-N
26
GND
3V3
3V3
AGND
VDD
VREF
DGND
DOUT
CLK
DIN
CS
ADC1
CH0
CH1
CH2
CH3
CH4
CH5
CH6
CH7
A0
A1
3V3
C1
GND
R1
A0
PR1
VT90N1
3V3
GND
TMP36
3V3
GND
VCC
VOUT
GND
A1