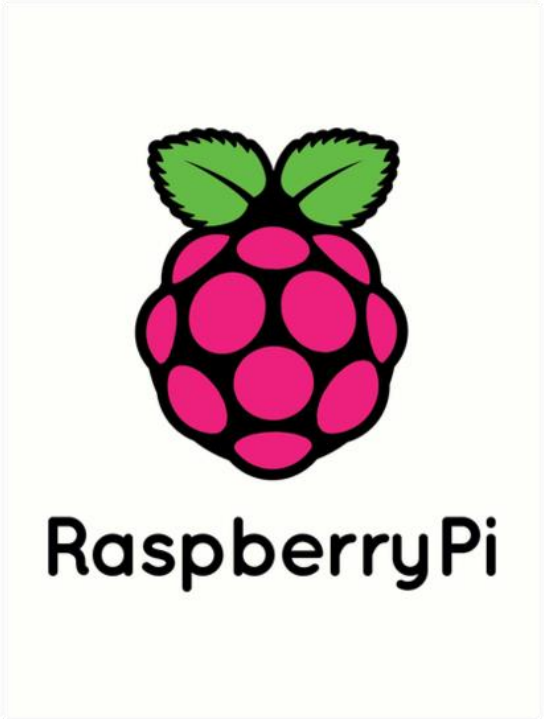


POLYTECH LILLE

Université de Lille

Projet IMA3/4

Administration système, déploiement et surveillances de logiciels dans un réseau de capteurs

Rouillé Guillaume – Capronnier Eymeric – Houssaini Ziyad

Table des matières

Introduction

Dans le cadre de recherches dans le domaine des réseaux de capteurs et des objets connectés, notre projet consiste à développer une solution de maintenance et de reconfiguration à distance d'un ensemble de nœuds déployés dans un environnement réel. Afin de faciliter la vie de ces chercheurs sur le test de leurs hypothèses, ils pourront facilement et rapidement déployer leurs créations sur tous les nœuds souhaités grâce à un système de sélection ou au téléchargement du nouveau code sur tous les nœuds du réseau.

L'objectif principal de notre projet est de créer une interface permettant de tester des programmes ou logiciels en les déployant sur le vaste réseau de capteurs. Nous devons rendre accessible chaque nœud indépendamment des autres, tout en permettant un lien entre tous, afin d'envoyer le contenu vers tout ou une partie des nœuds disponibles.

Enfin, nous allons permettre la mise à jour et l'adaptation du chemin d'envoi des données des nœuds, entre eux, et avec l'application de gestion.

Nous allons devoir répondre à plusieurs problématiques.

Tout d'abord nous devons créer un site web dynamique permettant l'envoi des données au serveur et finalement, aux nœuds de capteurs sélectionnés. Pour cela, nous allons devoir implémenter un système d'envoi de fichiers vers un serveur. Le routeur (Banana Pi) va filtrer les informations reçues, et donner un accès internet aux différents nœuds représentés par des Raspberry Pi.

Ensuite, il faudra que les Raspberry récupèrent les fichiers C sur le serveur et les envoient correctement aux différents capteurs de leur nœud et puissent récupérer les informations à retourner.

Pour se faire, on utilise l'utilitaire Python Ansible, permettant la gestion et la configuration des nœuds de capteurs. Celui-ci va récupérer les fichiers C et les distribuer sur les différents capteurs affectés. Le retour d'information se fera par le chemin inverse jusqu'au serveur web sous forme de fichiers contenant les données issues des capteurs.

Nous allons vous présenter les 4 parties principales de ce projet :

- Le site web, lieu de dépôt des codes à tester ;
- Le routeur, point d’accès internet pour les Raspberry PI ;
- La Raspberry PI, permettant le lancement du programme ;
- Le fichier C.

Le site web – Guillaume Rouillé

Ce projet nécessite la création d'une interface web, liée à un serveur de données. Nous avons utilisé le serveur Houplin mis à disposition des élèves de Polytech. Ce site doit permettre plusieurs choses :

- La connexion au serveur de données et à la base de données qui y sera créée ;
- L'interaction entre le site et le serveur afin d'envoyer des fichiers sur ce dernier ;
- La sélection du capteur sur lequel le code doit être envoyé ;
- La récupération et l'affichage des réponses des capteurs (prévu en 4ème année).

Pour se faire, nous avons utilisé les langages classiques de construction de site web, à savoir HTML5 et CSS3. Nous avons également utilisé les langages SQL et PhP afin d'interagir avec la base de données et le serveur.

Dans un premier temps, nous avons créé une page d'accueil et mis en forme le site avec un design CSS. A partir de cette page, nous avons créer les différentes parties du site, qui s'imbriquent les unes dans les autres. En effet, les parties d'entête et de barre de menu sont indépendantes et sont incluses dans chaque page. Ensuite, en fonction de l'état de la connexion, cette page d'accueil affichera différentes choses, introduites ci-dessous.

Système de connexion et inscription

Dans le but sécuriser l'interface web, nous avons créé un système de compte. Pour l'instant, tout le monde peut s'inscrire à partir du site et de la page d'inscription ; mais cela ne sera plus le cas une fois le projet terminé.

Tout d'abord, il a fallu créer la base de données comportant les informations de connexion. Pour cela, nous avons créer la base projetCapteurs sur le serveur Houplin et la table membres à l'aide de la commande suivante :

```
CREATE TABLE membres(numero int PRIMARY KEY, nom varchar(50), prenom varchar(50), pwd varchar(150), id varchar(50));
```

Cette table membres contient les champs suivants :

- Numéro du membre ;
- Nom ;
- Prénom ;
- Mot de passe ;
- Identifiant de connexion.

Ensuite, pour mettre en place ce système de compte, nous avons créé 2 formulaires HTML/PhP : un pour l'inscription et l'autre pour la connexion. Dans ces formulaires, plusieurs informations sont demandées afin d'authentifier la personne connectée. Le mot de passe sont protégés par la fonction password_hash(), ce qui ne permet pas de récupérer le mot de passe initial si la base de données est piratée.

Lorsque le formulaire d'inscription est soumis, les données sont vérifiées et envoyées dans la table membres et l'utilisateur est redirigé vers la page d'accueil.

Lorsque le formulaire de connexion est soumis, l'identifiant et le mot de passe sont vérifiés et une session est démarrée. Pour cela, on lance la fonction session_start() à chaque chargement de page, et on crée des variables de session (Login et Password) qui permettront de savoir si le visiteur est connecté ou non.

Ainsi, une fois le compte créé et l'utilisateur connecté, il peut changer de page tout en restant sur son compte, et il pourra accéder aux fonctions principales du site, présentées dans la partie suivante.

Sélection du capteur

La page principale du site une fois connectée est une page permettant la sélection d'un capteur et d'un fichier à envoyer sur le serveur. Voici comment ces 2 parties fonctionnent.

Tout d'abord, dans la base de données projetCapteurs, nous avons créé une seconde table nommée capteurs, et contenant toutes les informations nécessaires à leur identification, c'est-à-dire :

- Numéro du capteur ;
- Nom du capteur ;
- Raspberry à laquelle le capteur est connecté ;
- Type de capteur.

Sur la page principale, on affiche un tableau contenant ces informations pour tous les capteurs ainsi que des pins de sélection. L'utilisateur choisi un capteur en cliquant sur le pin correspond. Le nom et le numéro du capteur sont ensuite enregistrés dans un fichier XML, qui est envoyé sur le serveur grâce à la fonction file_put_contents().

Protocole XMLHttpRequest

Pour récupérer le fichier C sur l'ordinateur et le stocké en attendant de l'envoyer sur le serveur avec le protocole expliqué ci-dessous, on utilise la balise HTML suivante :

```
<input type="file" name="fichier" class="form-control-file" id="fichier">
```

Maintenant que le fichier est stocké sur le site dans un répertoire temporaire, nous devons le déplacer dans le dossier upload créé spécialement sur le serveur. Pour se faire, on effectue tout d'abord quelques vérifications :

- On vérifie qu'il n'y ait pas eu d'erreur lors du premier stockage ;
- On vérifie que le fichier n'a pas une taille trop importante (optionnel) ;
- On vérifie que l'extension du fichier est correcte (ici, .c).

Ensuite, on utilise la fonction move_uploaded_file() pour déplacer le fichier du répertoire temporaire vers le répertoire upload après l'avoir renommé en binaire.c. On affiche ensuite les informations contenues dans la variable $_FILES. C'est-à-dire le nom, le répertoire, le nombre d'erreurs, le type et la taille.

Les fichiers binaire.c et sensor.xml sont maintenant disponibles sur le serveur. Il reste un dernier fichier à envoyer, contenant le numéro de la demande, utile pour la partie Raspberry.

Le routeur – Ziyad Houssaini

Pour réaliser ce projet dans sa globalité, nous avons besoin d'un routeur wifi donnant l'accès à internet aux différentes Raspberrys. Il nous était recommandé d'utiliser une carte Banana PI R1, très pratique pour ce genre de chose. Notre objectif était donc de passer la carte en point d'accès afin de s'y connecter par liaison filaire depuis une Raspberry.

Pour le moment, nous n'avons pas en tête d'utiliser la Banana PI pour faire un premier filtrage, c'est pourquoi elle servira uniquement de routeur. Le fichier C sera récupéré directement sur la Raspberry. Pour ce premier objectif, il nous fallait du matériel :

- Un câble d'alimentation de moins de 1.6A ;
- Un câble HDMI/VGA pour connecter la Banana PI à un écran ;
- Une carte micro-SD avec son lecteur.

La première étape était d'installer l'OS permettant le fonctionnement de la Banana PI. Dans les directives du projet, il nous été recommandé d'utiliser OpenWRT. Nous avons donc installé ce dernier sur la Banana PI.

Après plusieurs tentatives d'interaction avec la Banana (via le clavier), rien ne paraissait fonctionner avec OpenWRT. C'est pourquoi nos tuteurs nous ont proposé une nouvelle carte SD contenant une image de Debian. Après installation de celle-ci, nous avions accès à la Banana PI, le problème était donc résolu.

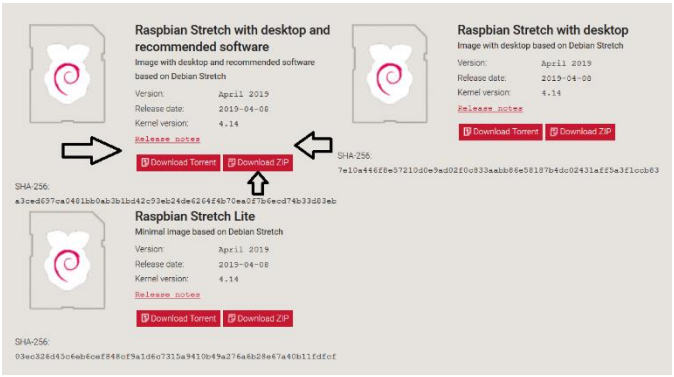
Au cours de la séance suivante, au moment du branchement de la carte, rien ne se passait. Nous n'avions plus accès à Debian, rien ne s'affichait à l'écran. Après plusieurs tentatives de redémarrage et de mise hors tension, il n'y avait aucune évolution. Nous avons donc décidé de charger une nouvelle image Armbian sur la Banana PI. Après son installation, l'interface était de nouveau accessible. A ce stade, il fallait donc configurer la carte pour qu'elle remplisse son rôle.

Après configuration du SSID, du mot de passe et d'autres paramètres permettant de rendre la carte visible en tant que point d'accès, nous avons réussi à la faire détecter par un téléphone portable. Cependant, la connexion à celle-ci était impossible même avec le bon mot de passe.

C'est à ce moment que nous nous sommes rendu compte que le noyau de la carte était probablement détruit à la suite d'une mauvaise manipulation ou installation. Celle-ci ne supportait pas l'image Armbian installée. Nous avons donc décidé de laisser de côté la partie routeur pour cette année, car elle n'est pas indispensable pour le bon fonctionnement des 2 autres parties du projet.

Configuration

Dans cette partie, il a fallu tout d'abord configurer la Raspberry afin qu'elle soit utilisable dans notre projet. Il a fallu télécharger Raspbian sur la carte SD et pour une utilisation simplifiée, nous avons choisi de télécharger et d'utiliser Raspbian le plus complet et le plus récent possible avec toute l'interface permettant de se faciliter la vie.



Une fois téléchargé et installé sur la carte SD, nous avons pu effectuer nos premiers pas sur la Raspberry Pi 3. Une fois le clavier, la souris, l'écran et la carte SD branchés et connectés à la Raspberry, nous avons pu connecter l'alimentation de la Raspberry. Le premier démarrage fut un peu long, car lors de celui-ci, la Raspberry a installé le système Raspbian. Lors de ce premier démarrage, des premières configurations ont été effectué qui ont été très simplifié car nous avons une version de Raspbian avec une interface graphique. Ces quelques réglages consistaient principalement à régler la langue, l'heure, la connexion internet et le nouveau mot de passe. L'identifiant et le mot de passe ont été laissé comme initialement c'est à dire le login « pi », et le password « raspberry ».

Réception du fichier

Dans cette partie, nous nous sommes tout d'abord intéressé à la récupération des informations envoyées par le serveur. Nous avons choisi simplement d'utiliser un wget afin de récupérer le fichier sensor.xml contenant les informations sur le capteur sélectionné et le fichier binaire.c contenant le code que le chercheur a envoyé afin d'être testé sur les différents capteurs (les capteurs étant remplacés pour cette fin d'année par des LEDs pour simuler pour l'instant, seulement l'envoie de l'information vers l'endroit souhaité).

```
wget -N -P ./ http://houplin.studserv.deule.net/~grouille/PROJET/upload/binaire.c
wget -N -P ./ http://houplin.studserv.deule.net/~grouille/PROJET/upload/sensor.xml
```

Pour l'instant, le numéro de PIN est stocké directement dans sensor.xml c'est à dire que dans ce fichier, l'information est stockée de façon à avoir le nom du capteur suivi d'un tiret puis le numéro de PIN à activer pour faire s'allumer la LED correspondante. Cependant par la suite le numéro de PIN ne sera pas directement stocké dans le fichier sensor.xml, chaque Raspberry possédera un fichier .txt où chaque capteur associé à cette Raspberry sera notifié dans ce fichier .txt par le nom du capteur et le numéro de PIN de la Raspberry étant connecté au capteur. Il faudra donc vérifier qu'il y est bel et bien un capteur associé à la Raspberry de même nom que le capteur demandé par le serveur afin d'effectuer un test. Tout cela car notre projet s'inscrit dans un contexte à grand échelle avec plusieurs Raspberry gérant plusieurs capteurs différents. Mais pour cette fin d'année, nous nous intéressons à simuler seulement le cas d'une Raspberry.

Il fallait donc pouvoir récupérer ce numéro de PIN stocké dans ce fichier afin que le code envoyé binaire.c puisse se lancer sur la LED sélectionnée. Nous avons donc créé la fonction pin_num() récupérant le numéro de pin dans le fichier sensor.xml. Le fichier xml étant sous la forme :

```
<?xml version='1.0' encoding='ISO-8859-1'?>
<sensor>nom_num</sensor>
```

Nous avons donc seulement parcouru ce fichier jusqu'à arriver au début du nom du capteur (dès que l'on a vu passer deux fois le caractère « > » alors on est au début du nom de capteur). Puis nous avons stocké le nom et le numéro de capteur dans un tableau. Enfin, pour accéder à ce numéro nous prenons la dernière valeur stockée dans le tableau.

Nous avons ajouté par la suite la création par le serveur d'un fichier demande.txt contenant le numéro de la demande ce qui sera utile pour l'automatisation du code afin de ne pas répéter une demande indéfiniment jusqu'à la prochaine demande. Un fichier verif.txt est stocké dans la Raspberry, lorsqu'une nouvelle demande est envoyée, il y a un nouveau numéro de demande, le fichier verif.txt qui stockait le numéro de l'ancienne demande est donc différent de demande.txt donc on peut lancer le code puis le fichier verif.txt prend la valeur de demande.txt ce qui empêche le code de se relancer tant qu'il n'y a pas de nouvelle demande.

Lancement du code

Enfin dans cette dernière partie, nous nous intéressons à l'automatisation du code.
Nous avons créé le script de récupération de tous les fichiers nécessaires, à savoir :

- Le fichier binaire.c ;
- Le fichier sensor.xml ;
- Le fichier demande.txt.

Ce script (load.sh) permet aussi la compilation du fichier C et le lancement des exécutables du programme.

```
wget -N -P ./ http://houplin.studserv.deule.net/~grouille/PROJET/upload/binaire.c
wget -N -P ./ http://houplin.studserv.deule.net/~grouille/PROJET/upload/sensor.xml
wget -N -P ./ http://houplin.studserv.deule.net/~grouille/PROJET/upload/demande.txt
gcc -o binaire binaire.c -lwiringPi
./blink
./binaire
```

Initialement, nous voulions utiliser une crontab afin de lancer ce script toutes les minutes par exemple, cependant nous n'avons pas réussi à faire marcher la crontab sur la Raspberry. Nous avons donc créé un autre script permettant l'exécution périodique (toutes les 15s) du premier script load.sh vu précédemment.

```
#!/bin/sh
while true
do
    ./load.sh
    sleep 15
done
```

Le fichier C

Pour cette première ébauche au S6, nous avons créer un script c permettant d'allumer une LED pendant 10 secondes en la faisant clignoter.

Le principe est simple, il suffit de passer le pin lié à la LED sélectionnée (c'est-à-dire au capteur sélectionné sur le site) en mode sortie (OUTPUT). Ensuite, on la fait clignoter en envoyant successivement un état LOW (0V) et un état HIGH (3.3V).

Ce fichier sera bien sûr modifié lorsque l'on considérera des capteurs et non des LEDs. Nous avons, par ailleurs, choisi de séparer ce fichier C du fichier C présent sur la Raspberry pour être cohérent vis-à-vis de ce qui nous était demandé. On a donc la possibilité de tester les fichiers C envoyés sur le serveur indépendamment de son traitement sur la Raspberry.

Conclusion

Au cours de ce premier semestre de projet, nous nous sommes beaucoup documentés sur nos différentes parties et nous avons élaboré une première ébauche considérant les capteurs comme de simples LEDs.

Nous avons conçu un site web avec un système de connexion et une interface de sélection de capteur. Sur ce site, nous pouvons également sélectionner un fichier sur l'ordinateur et l'envoyer sur le serveur.

- Lien vers le site web :

http://houplin.studserv.deule.net/~grouille/PROJET/

- Lien vers le Gitlab contenant les fichiers du site :

https://archives.plil.fr/grouille/IMA3_P10

Nous avons tenté de mettre en place un routeur à partir d'une carte Banana PI. Cependant, après une erreur lors du chargement d'un OS sur celle-ci, le noyau de la carte a été détruit. Cette partie n'étant pas indispensable pour le bon fonctionnement des autres parties, nous la laissons de côté pour l'année prochaine.

Finalement, nous avons réussi à récupérer automatiquement les fichiers déposés sur le serveur à partir de la Raspberry PI, et nous avons créé des scripts permettant leur mise à jour régulière et le lancement des programmes à chaque demande faite depuis le site web. Notre programme active un des pins de la Raspberry et fait clignoter pendant 10 secondes la LED qui lui est associée.

Webographie

Site Web

- Protocole XMLHttpRequest

https://openclassrooms.com/fr/courses/1085676-upload-de-fichiers-par-formulaire

Banana PI

- DataSheet

https://www.electronicsdatasheets.com/datasheet/BPI-R1.pdf

- OpenWRT

https://openwrt.org/

- Wiki

http://wiki.banana-pi.org/Banana_Pi_BPI-R1

Raspberry PI

- Pin Raspberry

https://pi4j.com/1.2/pins/model-3b-plus-rev1.html

- Configuration

https://peip-ima.plil.fr/mediawiki/index.php/BE_2017-2018

- Clignotement d'une LED

https://www.framboise314.fr/la-saga-blink-un-raspberry-pour-faire-clignoter-une-led/

http://nagashur.com/blog/2013/01/01/controler-une-led-depuis-les-ports-gpio-du-raspberry-pi/

- Installation Raspbian

https://raspbian-france.fr/installer-raspbian-premier-demarrage-configuration/