École Universitaire Polytechnique de Montpellier
Microélectronique et Automatique - Électronique et Informatique Industrielle

7 juin 2015

Serveur domotique

TERRIE Corentin
CROS Bastien
MONNIER Matthias

Table des matières

1 Presentation Projet **2**

2 Partie Android (client) **2**

3 Partie Raspberry (serveur) **2**
3.1 Première version du serveur : Connexion à un seul client 3
3.2 Seconde version du serveur : gestion de plusieurs clients 3
3.3 Troisième version du serveur : échanges de trame au format JSON 4
3.4 Quatrième version du serveur : lancement de tâche différentes selon le type de connexion 4

4 Protocoles domotiques **5**
4.1 Les protocoles grand public 5
4.2 Les protocoles de rupture technologique 5

5 Annexes **7**

1 Presentation Projet

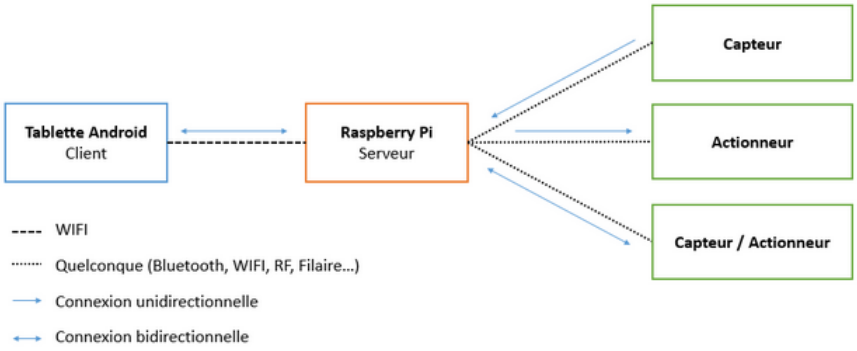


FIGURE 1 – Schéma montrant les différentes connexion du réseau.

2 Partie Android (client)

3 Partie Raspberry (serveur)

Le serveur servira a centralisé tout le réseau domotique. Il aura pour tâches :

- Possibilité de stocker les donnés des capteurs si liaison Tablette-Rasberry coupée.
- Gestion des données différentes en fonction de la connexion à un client Android ou un capteur/ actionneur.
- Si connexion à un Android :
  - Envoie des informations concernant les capteurs, les actionneurs, la métrologie.
  - Réception des ordres de la tablette : allumer tel lampe, se connecter à tel capteur, mettre en action les actionneurs.
- Si connexion à un capteur, simplement lire les informations.
- Si connexion à un actionneur, le commander (selon les ordres reçu de la tablette) et gérer son état.
- Calcul de métrologie pour chaque connexion.

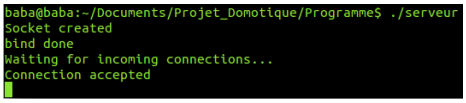
Pour implémenter le serveur, nous avons choisi d'utiliser une Raspberry B+, embarquant un système dérivé de Débian : Raspbian. Cela nous as permit d'implémenter un serveur Linux, utilisant les fonctions Unix.

3.1 Première version du serveur : Connexion à un seul client

Le premier serveur implémenté sur la Raspberry permettais seulement de gérer une connexion client/serveur la plus simple possible : le client se connecte sur le port du serveur (ici 8888) et grâce à l'adresse IP du serveur à celui-ci, et le serveur notifie sur le terminal que la connexion à bien été effectué. Ensuite le client demande à l'utilisateur de taper un message à l'écran, que le serveur lui renvoi. Finalement le client affiche dans son terminal la réponse du serveur ce qui permet de vérifier que le serveur à bien reçu le message.

Connexion au client Linux Le premier test à été effectuer en implémentant un client Linux sur la Raspberry. Ainsi, nous pouvions valider le serveur grâce au client, avant de tester la connexion avec le client Android.

Les premiers résultats ont été concluant en terme de connexion, c'est à dire que le serveur reçois la demande de connexion du client et renvois le message du client. Le client, quant à lui, reçois et affiche le message renvoyé par le serveur. Cela nous a permis de valider la création du serveur et la connexion au client. Par contre, nous avons rencontré un problème au niveau du buffer avec le client. En effet, le message affiché à l'écran étais bien celui initialement envoyé, mais le client affiche aussi tous les caractères présent dans le buffer.



(a) Connexion au client.

(b) Le client affiche la réponse du serveur.

FIGURE 2 – Serveur Linux

Comme solution nous avons pensé à gérer la longueur du buffer en fonction de la longueur du message reçu. Mais c'est une autre solution qui a été retenu : un parser. Nous avons choisi le parser JSON, que nous détaillerons plus loin.

Connexion avec le client Android Ensuite il a fallu tester cette solution avec le client sous Android : la conexion a bien marché, mais nous avons eu un premier problème au moment de recevoir le message sur le périphérique Android.

3.2 Seconde version du serveur : gestion de plusieurs clients

La deuxième version du serveur nous permet de gérer plusieurs clients à la fois. En effet, de part l'idée du projet, il nous fallait pouvoir gérer plusieurs client, d'origine différente. Pour cela, le serveur appelle un nouveau thread a

chaque demande de connexion. Le thread ainsi créé s'exécute ainsi durant la connexion au client. La connexion est quitté si le client arrête de communiquer.

Connexion au client Linux Comme précédemment, nous avons d'abord testé notre programme sur Linux avant Android. La connexion marchais très bien, hormis le problème mentionné dans la première version du serveur. Ce qui est normal car nous n'avons pas encore implémenter le parser.

Connexion avec le client Android Le serveur a bien accepté la connexion au client Android, mais nous n'arrivons pas décoder proprement les trames que nous envoyais celui-ci.

3.3 Troisième version du serveur : échanges de trame au format JSON

Pour pouvoir parser/déparser nos trames, nous avons choisi d'utiliser le format de trames JSON. Il présente plusieurs avantage, dont les principaux sont sont son coté générique et abstraits d'une part. Il est donc facilement implémentable sur un langage C (serveur) et JAVA (client). De plus, des parsers existe à la fois sous Linux et Android et ce format de donnée et facilement lisible. Il se presente sous la forme suivante :

```
{"name":"JackHack","nickname":"Hacker", "score":123,"current":152.3}
```

Nous avons choisi le parser suivant : parser JSON jsmn, qui a l'avantage d'être facilement adaptable avec notre programme. Un parser JSON est nativement implémenter dans le système Android, ce qui nous a permis directement tester la communication Android-Linux.
Le périphérique Android envoie une trame JSON (comme présenté plus haut) et le serveur la décode et l'affiche :

```
tD{"current":152.32,"score":123,"nickname":"Hacker","name":"JackHack"}start Json
 unparse

Object expected
length message : 1
message client : t
```

Le parser jsmn, qui marchait très bien pour une communication Linux-Linux, pose des problèmes dans ce cas ici. En effet, des caractères (ici "tD") sont parsé avant d'arriver à la trame au bon format. La solution envisagée est de coder une fonction qui prends le string reçu qu'a partir de la première accolade.

3.4 Quatrième version du serveur : lancement de tâche différentes selon le type de connexion

La version précédentes permettant de gérer plusieurs clients, mais pas de trier la fonction de ceux-ci. Nous avons donc envisagé plusieurs solutions pour résoudre le problème.

Différents ports Au moment de créer le serveur, plusieurs ports sont ouvert. Ainsi, chaque périphérique se connecte au numéro de port correspondant au service requis. Ensuite, le serveur lance le thread correspondant.

Un identifiant dans la trame JSON Comme il est facile de modifié les trames sous ce format, il est possible de rajouter un champ pour indiquer quel est le service demandé.

Un message comme identifiant Le premier message de la connexion peut servir à donner l'identifiant du périphérique.

4 Protocoles domotiques

4.1 Les protocoles grand public

X10 Le X10 est un vieux protocole (développé en 1975) par courant porteurs. Les modules X10 peuvent être piloté par des télécommandes radio (433MHz).

Les plus :

- Protocole le moins cher dans le domaine des automatismes résidentiels
- Communauté d'utilisateurs très active
- Bonne distribution des produits

Les moins :

- Gros problème de sécurité (toute personne possédant l'accès à une partie de l'installation électrique peut envoyer des ordres X10)
- Incompatibilité entre les réseaux électriques des différents pays
- Pas de retour d'état des modules et collisions non gérées

Le X10 est vieillissant et par conséquent de nombres constructeurs le délaissent. Il comporte comme seul avantage d'être simple et n'est donc plus conseillé.

OREGON

OREGON

4.2 Les protocoles de rupture technologique

Deux protocoles seront détaillés dans cette partie : Z-Wave et EnOcean. Ils partagent de nombreuses similitudes, ils sont tous deux :

- des protocoles domotiques
- des technologies sans fil
- des technologies disponibles via plusieurs centrales domotiques
- des technologies mises en œuvre par plusieurs constructeurs de périphériques.

Z-Wave Le Z-Wave est un protocole radio conçu pour la domotique. Il utilise la fréquence 848.42MHz. Ses caractéristiques sont qu'il est relativement sécurisé, à double sens (chaque composant est à la fois récepteur et émetteur) et qu'il utilise un réseau maillé.

Les plus :

- Grande richesse fonctionnelle.
- Topologie maillée du réseau sans fil (contrainte de portée levée car les périphériques relayent les informations entre eux).
- Retour d'état des périphérique (acquittement de commande, retour d'état de la batterie etc...).
- Grande liberté de choix sur les périphériques.

Les moins :

- Complexité de la mise en place.
- Prix élevé des modules.
- Consommateur de pile.

Le Z-Wave est un protocole de choix pour les actionneurs mais moins pour les capteurs. Voici le résumé graphique :

EnOcean Le Enocean est également un protocole radio utilisant la fréquence 848.42MHz. Ce protocole à l'avantage d'être sans fils et sans piles. En effet les périphériques puisent leur source d'énergie de leur environnement direct. Il utilise notamment l'effet photovoltaïque (transforme la lumière en électricité), l'effet piezo-électrique (transforme un choc ou une forte pression en électricité) ainsi que l'effet Peltier ou effet thermoélectrique (transforme une différence de température constaté à un instant donné en électricité).

Lorsqu'un interrupteur EnOcean utilise des cristaux piezo-électrique pour produire son électricité, il communique son ordre ON/OFF par voie hertzienne en utilisant l'énergie fournie par la pression mécanique de celui qui actionne l'interrupteur. Il peut ainsi générer le peu de courant nécessaire pour envoyer l'information à la centrale.

Les plus :

- Flexibilité élevée lors de la mise en oeuvre mais également en cas de modification de l'installation.
- Pas besoin de changer les piles.
- Pas de coût de consommables.

Les moins :

- Coût des modules plus important que certaines autres technologies.
- Design peut attrayant de certains modules.
- Retour d'état non disponible pour certains périphériques (seuls ceux ayant une batterie de stockage de l'énergie le peuvent).
- Le réseaux n'est pas maillé. Mais il existe des relayeurs graces auxquels le réseaux peut rayonner sur une portée de 200m.

Voici le résumé graphique :

Les deux graphiques précédent sont issues du site `www.abavala.com`. Elles ne sont pas issues de mesures scientifiques mais d'avis personnels exprimés librement. Voici le graphique qui regroupes ces deux protocoles :

5 Annexes