

HomeSIP**INRIA****16/06/2008 au 19/09/2008**

KADIONIK Partice	Qualité : Maître de conférence ENSEIRB	
BÉRAT Frédéric	Département : Électronique	23 Octobre 2008

TABLE DES MATIERES

REMERCIEMENTS.....	3
PARTIE N°1. INTRODUCTION.....	4
PARTIE N°2. L'ENTREPRISE XYZ.....	6
PARTIE N°3. OBJECTIFS DE LA MISSION TECHNIQUE.....	7
SUJET DE STAGE.....	7
CAHIER DES CHARGES FONCTIONNEL.....	8
CRITÈRES DE VALIDATION DU PROJET.....	8
MOYENS MIS À DISPOSITION DU STAGIAIRE.....	8
PLANIFICATION DU PROJET.....	8
PARTIE N°4. RÉALISATION DU PROJET.....	9
CONCEPTION GÉNÉRALE.....	9
CONCEPTION DÉTAILLÉE.....	9
PREMIÈRE VERSION.....	9
<i>Le serveur</i>	9
<i>Le parser XML</i>	10
<i>Les périphériques</i>	10
<i>La création des réponses</i>	11
PREMIÈRE VERSION AMÉLIORÉE.....	11
DEUXIÈME VERSION.....	12
TESTS.....	14
<i>Le serveur</i>	14
<i>Le parser XML</i>	14
<i>Les périphériques</i>	14
<i>La création des réponses</i>	15
VALIDATION ET RÉSULTAT.....	15
PARTIE N°5. CONCLUSION.....	16
PARTIE N°6. GLOSSAIRE.....	17
PARTIE N°7. BIBLIOGRAPHIE.....	18
PARTIE N°8. ANNEXES.....	19

Remerciements

Je tiens bien évidemment à remercier mon maître de stage, Monsieur Kadionik, pour l'aide fourni lorsque j'ai eu quelque soucis.

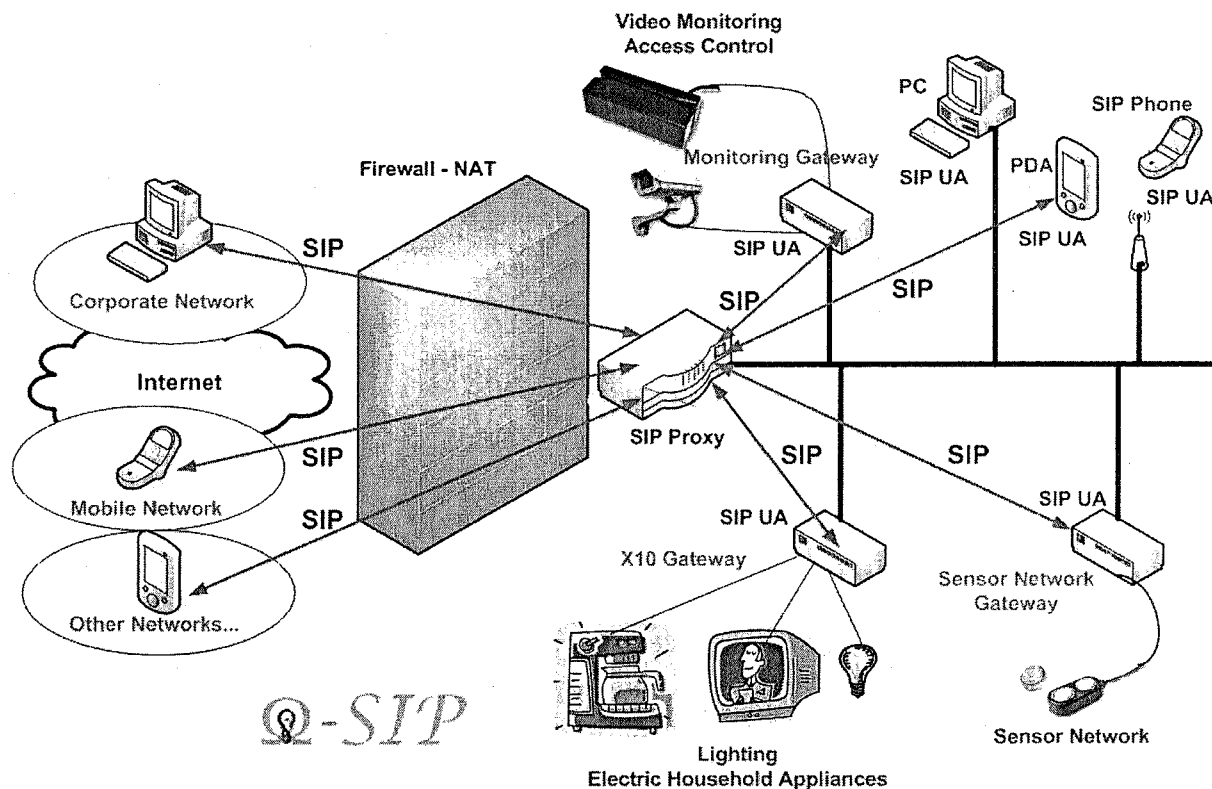
Je remercie également toute l'équipe Phoenix, pour leur accueil au sein de l'institut. Ces quelques mois passés le fur dans une très bonne entente, ce qui est d'autant plus agréable.

PARTIE N°1.INTRODUCTION

Ces dernières années ont vu exploser les capacités proposer par l'informatique et l'électronique. Cet engouement de la population pour ces deux sciences appliquées ont permit de faire naître ce que l'on nomme la « domotique ». La domotique est l'ensemble des technologies de l'électronique, de l'informatique et des télécommunications utilisées dans les habitations. La domotique vise à assurer des fonctions de sécurité (comme les alarmes), de confort (comme les volets roulants), de gestion d'énergie (comme la programmation du chauffage) et de communication (comme les commandes à distance) que l'on peut retrouver dans la maison. Il s'agit donc d'automatiser des tâches en les programmant ou les coordonnant entre elles.

Parmi les domaine dans lesquels peuvent s'appliquer les techniques de la domotique, nous pouvons retrouver la programmation d'appareil domestiques, gestion de l'énergie tel que pour le chauffage la climatisation ou l'éclairage, la gestion des alarmes et des interphones, ou encore l'amélioration du confort acoustique ou la gestion d'ambiance lumineuse.

On se retrouve donc avec un réseau domotique tel qu'illustré dans la figure suivante :



Ω-SIP

Dans le cadre d'une recherche pour développer ce type de système, une équipe a été mise en place à l'INRIA futur de Bordeaux. Cette équipe nommée Phoenix, à notamment pour mission de développer un certain nombre d'outil permettant la mise en place de ce système.

Le stage présenté ici a pour but de développer des passerelles SIP gérant des réseaux de capteurs ou de périphérique X10. Ce projet est par nature un projet orienté embarqué, composé de différents systèmes électroniques sous Linux embarqué, ce qui m'a motivé à le choisir.

Ce stage nécessite donc la réalisation d'une passerelle capable de communiquer avec d'autres entités et de répondre à des demandes concernant ses périphériques.

Après une présentation du Laboratoire INRIA dans lequel a été développé ce projet, nous nous attarderons plus amplement sur les objectifs de la mission technique. Après quoi nous développerons les différentes étapes de la réalisation du projet.

PARTIE N°2.L'ENTREPRISE XYZ

Les laboratoires de l'INRIA ont des centres de recherche dispersés dans toute la France. Le siège et centre principal de recherche se trouve à Paris, il y a aussi 4 centres de recherche à Rennes - Bretagne Atlantique, à Grenoble - Rhône-Alpes, à Nancy - Grand Est, à Sophia Antipolis - Méditerranée. Ces centres sont entourés de diverses équipes de recherche hors site, dont l'INRIA Bordeaux-Sud-Ouest fait parti.

La création en janvier 2008 du centre de recherche INRIA Bordeaux - Sud Ouest est basée sur les équipes-projets et services créés à Bordeaux et à Pau dans le cadre de l'unité de recherche Futurs, avec le soutien fort du Conseil Régional d'Aquitaine.

Les équipes de recherche y ont été construites en s'appuyant sur des partenariats forts avec les universités et grandes écoles des sites concernés et en collaboration avec leurs laboratoires réputés.

Grâce au dynamisme de ces collaborations, à l'apport de personnels ayant effectué une mobilité depuis d'autres sites de l'INRIA et en s'appuyant sur une politique de recrutement de chercheurs et de personnels de soutien à la recherche de haute qualité, le centre de recherche emploie directement, début 2008, plus de 100 personnes. Il compte 8 équipes-projets de recherche, et 6 équipes dont l'évaluation est en cours et qui devraient devenir prochainement équipes-projets. Le centre rassemble globalement plus de 250 personnes.

Dans le cadre de son plan stratégique, l'INRIA entend définir son développement à Bordeaux - Sud Ouest en liaison forte avec le pôle de compétitivité Aerospace Valley. Des partenaires industriels et académiques seront également étroitement associés. On peut citer par exemple Total, Safran/Turbomeca, Thales, Rhodia, le CEA (en particulier dans le cadre du programme de recherche autour du Laser Mégajoule), France Telecom, EDF, Airbus, SNCF, etc. directement, mais aussi au travers du pôle Aerospace Valley.

Les capacités de développement du centre et les partenariats engagés permettent d'envisager une croissance d'au moins 50% dans les quatre ans à venir.

PARTIE N°3.OBJECTIFS DE LA MISSION TECHNIQUE

Sujet de stage

Le projet HomeSIP est un projet visant à mettre en place une plateforme domotique à l'ENSEIRB. Le nom de ce projet est l'acronyme de « Home Automation with SIP » ce qui signifie qu'il s'agit d'un projet de domotique basé sur la mise en œuvre du Protocole SIP. Le principe de la domotique est de collecter, transmettre et traiter des informations afin de coordonner des actionneurs et capteurs au sein d'un local (automatisation du domicile).

L'idée est donc d'utiliser SIP dans cette optique. Pour cela, il a été mis en place une infrastructure matérielle composée de capteurs et d'actionneurs connecté à des système embarqué possédant une connectivité internet et mettre en place des logiciels dans les systèmes embarqués pour les piloter.

Le protocole SIP a été choisi pour les avantages qu'il possède par rapport à d'autres protocoles, tels que SNMP (Simple Network Management Protocol). Parmi ces avantages nous pouvons citer sa faible complexité, sa facilité d'utilisation, son aspect sécuritaire (les données peuvent être authentifiées et chiffrées), son empreinte mémoire plus faible et ses nombreuses applications clientes telles que « gaim » ou « skype ». Un autre de ses avantages est que ce protocole reprend des éléments techniques des protocoles SMTP et HTTP comme par exemple les codes d'états pour les réponses.

L'infrastructure matérielle se greffe sur la plateforme SIP actuelle de l'ENSEIRB. Les périphériques utilisés sont des PC, des téléphones SIP ainsi que des PDA. Dans le cadre du développement en cours seules deux passerelles SIP sont déployées. Le détail de ces passerelles est donné un peu plus loin.

Le choix de la pile SIP c'est porté sur sur la pile GNU oSIP ainsi que son extension eXoSIP développées par Aymeric Moizard. Cette pile est écrite en langage C et est donc fortement portable et à faible empreinte mémoire. L'extension eXoSIP est une bibliothèque de fonctions développée pour faciliter l'écriture des applications SIP.

Le SIP étant un protocole qui ne met en jeu que la gestion de session, une norme a été établie afin de pouvoir passer les commandes. Il a été choisi d'utiliser le langage XML. Le corps des messages SIP sera donc écrit avec ce langage.

Une fois ces choix fait, il faut mettre en place les logiciels permettant de piloter la passerelle. C'est sur ce point que nous allons nous arrêter par la suite.

Cahier des charges fonctionnel

Dans le cadre du projet HomeSIP, une plateforme communicante doit être développée. Cette plateforme doit pouvoir réceptionner, analyser puis répondre à une demande venant d'un autre système de manière autonome. Elle doit donc être composée d'une partie serveur capable d'établir et de maintenir cette communication, et d'une partie fonctionnelle capable d'utiliser les divers périphériques suivant la demande. S'agissant d'une passerelle, la finalité est que, du point de vue du système de commande, elle soit totalement transparente.

Critères de validation du projet

Le système doit donc suivre les contraintes suivantes :

- Être autonome et stable dans le temps.
- Répondre aux requêtes d'un moniteur
- Être capable de mettre à jour la liste des périphériques dont il dispose
- Être capable de générer des événements de type alarme

Moyens mis à disposition du stagiaire

La passerelle est composée du matériel du commerce pour ne pas avoir à le développer. Elle est composée d'une carte ARM9 de la société Eukréa qui a le grand avantage en plus de supporter Linux de posséder de nombreuses entrées/sorties pour pouvoir y connecter différents périphériques. Parmi ces E/S il y a des liaisons série, des bus I2C des bus USB etc ... Elle possède également une interface réseau.

Parmi les périphériques, il y a des capteurs de températures, les DS1920 de Dallas Semiconductor possède une interface permettant de les connecter via la liaison Série. Nous avons également à disposition des dispositifs X10 permettant la commande de lampes électriques en mode tout ou rien ou en mode graduel. Enfin, des périphériques ZigBee sont également connectés à la passerelle.

Planification du projet

La réalisation du projet c'est faite en plusieurs parties. La première a consisté simplement à reprendre le code existant pour obtenir quelque chose de fonctionnel, puis l'améliorer en conservant un processus unique. Dans cette optique, et après une prise en main des bibliothèques disponibles, il a fallu analyser et déterminer la provenance des erreurs générés par le programme. Le code a également été divisé en parties indépendantes afin de plus facilement retrouver ces erreurs.

La seconde partie a consisté à diviser la tâche en différents processus fonctionnant en parallèle. Cette partie est la plus complexe, l'existence des processus multiple rend plus difficile la tâche de recherche et de découverte des erreurs.

Enfin la dernière partie a consisté à modifier la vision qu'a l'utilisateur des différents périphériques du système. En effet, dans les deux premières parties, le moniteur voyait et communiquait directement avec la passerelle. Le but ici est donc d'avoir une passerelle plus transparente, le moniteur communiquant avec chaque périphérique par son intermédiaire sans s'en rendre compte. Il faut bien sûr reprendre un maximum des fonctions précédemment développées.

PARTIE N°4.RÉALISATION DU PROJET

Conception générale

Dans la première version, le système était conçu de la manière suivante :

- Une partie serveur gérant la réception et l'envoi de message en utilisant le protocole SIP
- Un parser XML permettant d'extraire les informations issues des message SIP
- Des fonctions de commande des périphériques.

Le code a donc été découpé en plusieurs parties. Chacune de ses parties pouvant être développées et testées de manière indépendante. Parmi ces parties on distingue :

- la partie serveur, qui permet l'enregistrement et la réception les requêtes,
- le parser qui permet d'analyser et de récupérer les informations disponibles dans ces requêtes,
- le bloc périphérique, qui permet d'établir la liste des périphériques, de récupérer les valeurs des capteurs ou d'effectuer une commande,
- Le bloc réponse, qui génère la réponse qui va être envoyé au moniteur grâce à toutes les informations récupérés précédemment.

Conception détaillée

Première version

Le serveur

La partie serveur du système a pour mission d'enregistrer la plateforme au près du serveur du laboratoire ainsi qu'au près du framework. La plateforme se voit donc attribuer une adresse IP fixe ainsi qu'un nom permettant au moniteur de communiquer avec elle via le framework.

Cette partie serveur permet également de traiter les messages en fonction de leur type. Dans ce projet nous nous sommes intéressés aux messages de types « New Message » qui incluent les message de demande d'option de la part du framework ainsi que les messages de commande de la part du moniteur.

La demande d'option permet au framework de savoir que l'entité tentant de s'enregistrer au prêt de lui est de type « Passerelle ». De cette manière, notre système sera inclus dans la liste des passerelles lorsque le moniteur la demandera.

S'il ne s'agit pas d'une demande d'option, le message est traité par le module servant à parser le corps du message. Il est ensuite traité en fonction des informations récupérées, la réponse est créée par le module correspondant puis envoyée par le serveur.

Une fois tout cela fait, le serveur se met dans l'attente de la réception d'un nouveau message.

Le parser XML

Lorsqu'un message différent d'une demande d'option est reçu, il possède une architecture prédéfinie. De part cette architecture et grâce aux balises XML dont il est composé, il est facile d'extraire les informations nécessaires afin d'exécuter une commande et répondre au moniteur. Les informations récupérées sont stockées dans une variable globale dont la structure a été prédéfinie. Il s'agit de la structure UserData.

Le parser permet donc d'extraire la commande parmi la liste suivante (par avance désolé pour le français j'ai par défaut conservé les noms des fonctions jusqu'à la version 2) :

- AppelcommandeX10
- GetListeCapteur
- GetValeurCapteur

Pour l'appel de la commande X10, il y a plusieurs informations supplémentaires, l'identifiant du module X10 dans la liste des périphériques de la plateforme, le code site ainsi que le code dispositif.

La méthode de récupération de la liste des capteurs quant à elle ne nécessite pas d'argument supplémentaire.

Enfin, la méthode de récupération de la valeur d'un capteur est suivi de l'identifiant du périphérique concerné ainsi que l'unité de retour (Celsius ou Fahrenheit pour la température par exemple).

Une fois la récupération de ces valeurs effectuée la demande est traitée par le bloc gérant la création de la réponse.

Les périphériques

Différentes éléments ont été développées afin de répondre aux demandes du moniteur.

Tout d'abord, il a fallu créer une liste de périphériques, dans lequel on pouvait retrouver tout ce qui les caractérisait. Il fallait donc leur type (ici Xbee, Filaire ou X10), un identifiant qui peut être le code dispositif pour les X10 ou l'id propre des capteurs Xbee, le code site dans le cas des périphériques X10.

Il faut également pouvoir initialiser, compléter et réinitialiser cette liste. De cette manière la liste peut évoluer au cours du temps, ce qui permet par la suite de la mettre à jour régulièrement pendant que le serveur est dans l'attente de l'arrivée d'un nouveau message, ou encore de faire une mise à jour à la demande.

Ainsi donc, lorsqu'un appel est fait à la commande « GetListeCapteur », Une série de fonction est lancée afin de déterminer le nombre de périphérique ainsi que le port sur lequel ils se trouvent. Toute les informations requises dans la liste de périphériques sont alors complétées au fur et à mesure. Cette liste est ensuite renvoyé au moniteur, et utilisée pour vérifier qu'un périphérique existe bien lorsqu'une demande de valeur est faite par exemple.

La création des réponses

Le corps des réponses SIP est écrit en XML. Lorsqu'une commande est reçue on entre dans une fonction dans laquelle la réponse est pré-écrite. Le but est donc de compléter avec les informations manquantes. Ces informations sont l'Id du périphérique, sa valeur, son unité de mesure ainsi que la date à laquelle la valeur a été relevée dans le cas d'un capteur. Dans le cas où il s'agit de retourner la liste des périphériques, le type est envoyé et suivant celui-ci, l'id, le code site et le code dispositif.

The screenshot shows the Wireshark interface with a filter set to 'sip'. The packet list pane displays two packets:

No.	Time	Source	Destination	Protocol	Info
17	3.347087	194.199.1.201	194.199.1.203	SIP/XML	Request: MESSAGE sip:root@arm04
18	3.451718	194.199.1.203	194.199.1.201	SIP/XML	Status: 200 OK

The packet details pane for the selected packet (No. 18) shows the following XML structure:

```

<n0:getListeDispositifX10Response
  xmlns:n0="urn://phoenix.labri.fr/"
  <response
    i:type="d:framework.datatype.ListeCapteurs">
    <?xml
    <liste-dispositifs>
      <dispositifs>
        <code_site>
          A
        </code_site>
        <code_dispositif>
          1
        </code_dispositif>
      </dispositifs>
    </liste-dispositifs>
  </response>
  
```

Exemple d'un message envoyé pour la liste des dispositifs X10

Première version améliorée

Les améliorations réalisées sur la première versions ont consisté en une automatisation de certaines tâches. Parmi elles, on retrouve la recherche de périphérique, la prise de valeur à intervalle régulier, et surtout l'envoi de notification.

En effet, dans la version précédente, aucune vérification n'était faite sur les valeurs des périphériques. Il n'était donc pas possible de créer des événements dans le cas où par exemple la température dépassait un seuil prédéfini.

Dans un premier temps, il n'y avait qu'un processus unique. Les recherches des nouveaux périphériques ainsi que la mise à jour des valeurs des capteurs se faisait pendant la période d'attente du serveur. Cette solution n'était pas satisfaisante, elle ne permet en effet pas au serveur de traiter une demande pendant l'exécution des tâches de recherche et de mise à jour.

Les différentes tâches ont alors été distribuées sur différents processus indépendant les uns des autres, ce qui permet d'une part de mettre à jour la liste de périphérique indépendamment d'une demande du moniteur, de relever et enregistrer les valeurs des capteurs et donc de gagner en moyenne du temps sur la réponse fournie au moniteur et enfin de relever la variation des valeurs ou encore le dépassement d'un seuil de celle-ci.

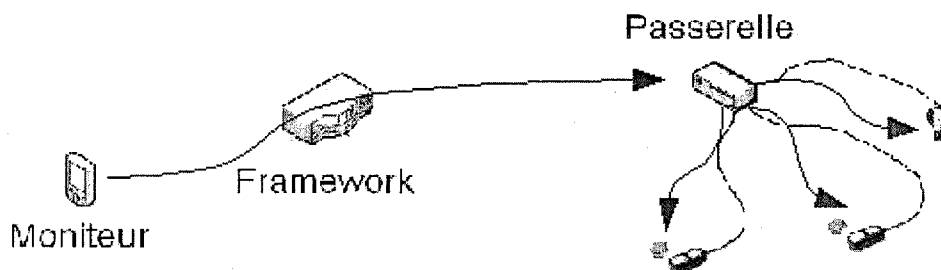
Il suffit alors au moniteur de souscrire à un évènement venant de la plateforme pour recevoir un message à chaque fois qu'un dépassement de seuil à lieu par exemple.

Cette version laisse également la possibilité d'avoir un processus attendant des évènements de manière asynchrone (alarme par exemple).

Deuxième version

Cette version est caractérisé comme seconde version en raison d'un changement majeur dans la gestion des périphériques et de l'enregistrement.

Dans la première version le réseau était constitué de la manière suivante :

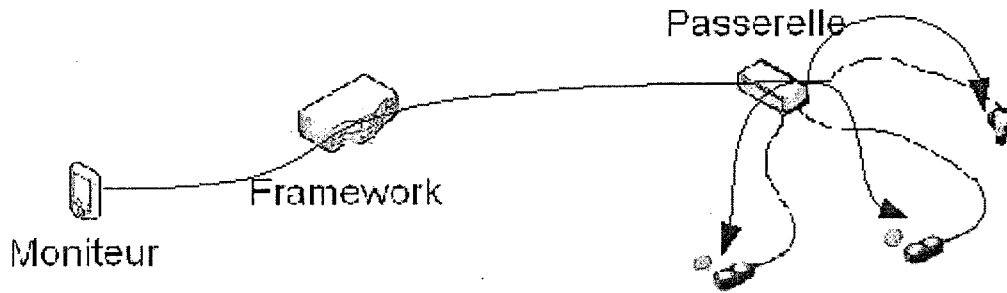


Description de la communication vue du moniteur

On voit donc que la passerelle n'en est pas vraiment une, puisqu'il n'y a aucune transparence vis-à-vis du moniteur. Le moniteur communique avec la passerelle qui elle même communique ensuite avec les périphériques.

Cette manière pose un problème par rapport notamment à la gestion des évènements. En effet, si le moniteur souscrit à un évènement il souscrit aux évènements de ce type pour l'ensemble des capteurs et non pour un seul d'entre eux.

L'idée est donc de rendre la passerelle invisible. Le moniteur va alors communiquer avec chaque périphérique par l'intermédiaire de la passerelle de manière transparente.



Description recherchée de la communication vue du moniteur

Pour cela il faut modifier entièrement la manière dont elle s'enregistre. En fait, la passerelle va enregistrer indépendamment l'ensemble des périphériques. L'adresse IP étant fixe, c'est le nom qui va changer. Chaque périphérique est enregistré de la manière suivante :

TypeId@adresse_IP:port

Une fois cela effectué il faut revoir la manière dont son gère les commandes du moniteur et les réponses fournies. Pour cela, la majorité du code déjà créé à été réutilisé et des fonctions ont été ajoutée pour récupérer les données issue des messages du moniteur.

Le parser ne sert plus alors qu'a récupérer la méthode, il n'y a plus d'arguments. Le périphérique auquel est associé cette méthode est alors identifié grâce à son nom. Une recherche exhaustive est faite dans la liste des périphériques en fonction de son type, de son id et enfin de son code site le cas échéant (pour les dispositifs X10).

De même pour la partie gérant les périphériques il a fallu ajouter un module d'enregistrement et de résiliation. Pour ce faire une copie de la liste est créée avant la nouvelle recherche des périphériques. Une fois la recherche effectuée, les deux listes sont comparées de manière a trouver quels éléments sont présent ou non dans les deux listes le tout en considérant que les positions des éléments ne sont pas nécessairement identiques.

Tests

Chaque partie précédemment détaillée a été testé indépendamment. L'objectif primaire des test était de vérifier le bon fonctionnement de chaque fonction, ce qui implique la récupération ou l'ajout de valeurs de manière correcte et répétée à l'ensemble des variables globales.

Le serveur

Les tests au niveau de la partie « serveur » sont relativement simples, il doit tout simplement s'enregistrer correctement au prêt du serveur du Laboratoire et au prêt du framework.

Dans la première version un seul enregistrement est fait, celui de la passerelle. Les vérifications du bon enregistrement se fait dans un premier temps par la réception d'une demande d'option effectuée par le framework, puis dans un second temps par la réception de messages provenant du moniteur.

La vérification correcte de la réception des messages du côté du framework est réalisée grâce au logiciel « WireSharks » qui permet de visualiser les échanges ayant lieu avec la carte réseau de l'ordinateur.

Du côté de la passerelle les vérifications peuvent être faite en redirigeant le buffer dans lequel ce trouve le corps du message reçu vers un fichier. Le fichier peut ensuite être examiné afin de vérifier notamment que toutes les informations requise y sont et qu'il n'y a pas d'erreur lors de la récupération puis du traitement du message.

Le parser XML

Pour vérifier le bon fonctionnement du parser il y a plusieurs choses à faire. La première est de récupérer le corps du message reçu, la seconde est de renvoyer sur la sortie standard les informations qui ont été récupérées puis enregistrées dans les variables globale.

Les périphériques

Pour cette partie, c'est l'ajout et le retrait d'éléments dans la liste des périphériques qu'il faut tester. Cette vérification est effectuée par le renvoi sur la sortie standard de toute les informations contenue dans la liste ainsi que l'obtention des valeurs des capteurs le cas échéant.

Il faut également procéder à une série d'ajout et de réinitialisation de cette liste de périphériques, ce qui implique une nouvelle recherche de périphérique.

Pour la seconde version les mêmes vérifications ont été faites ainsi que des vérifications de la copie multiple de liste. S'ajoute également à ça la vérification de l'enregistrement de chaque périphériques, la réception et la réponse à la demande d'option.

La création des réponses

Les tests concernant la création des réponses ont été effectués grâce à Wiresharks. Il s'agit simplement d'une visualisation de la réponse reçue du côté du framework quelle que soit la demande effectuée par le moniteur.

Validation et résultat

Afin de valider le projet, un dernier test doit être effectué. En effet, le projet repose sur le fait que le système doit être autonome et durer dans le temps. Il faut donc tester l'endurance du programme, et notamment l'absence de « fuites mémoires ». Ces fuites sont caractérisées par une augmentation plus ou moins rapide de l'utilisation de l'espace mémoire pendant le fonctionnement du système. Il faut donc faire fonctionner le système pendant une longue durée pendant laquelle de nombreux échanges sont effectués avec le moniteur.

Ainsi, le programme a fonctionné pendant 4h d'affilé sans interruption, répondant à une requête toutes les 3secondes, renouvelant la liste des périphériques toutes les 5 minutes, et mettant à jour les valeurs des capteurs de manière à ce qu'elles soient globalement toutes renouvelées dans un intervalle de 30 secondes. Durant cette période, des périphériques ont été ajoutés et retirés de la plateforme de manière à vérifier que l'enregistrement et le renouvellement se faisaient bien correctement.

Le bilan est positif dans les cas testés.

A noter que suivant la fréquence des requêtes l'utilisation de la mémoire par le programme augmente. Cette augmentation est probablement due à la mémorisation des requêtes récentes par la pile SIP.

PARTIE N°5.CONCLUSION

Avec le grand progrès des technologies de la communication, de l'électronique et de l'informatique se sont réunies pour former un seul domaine : la domotique. Cet union permet aujourd'hui à l'utilisateur un meilleur contrôle de son environnement proche et de son confort.

Dans cet optique, nous avons développé un système qui permet aux utilisateurs de piloter et de surveiller les dispositifs domestiques localement ou à distance à travers une infrastructure SIP.

Nous avons vu au travers de la démarche de conception, l'ensemble des structures et logiciels nécessaires pour permettre la bonne communication entre un moniteur et une série de capteurs et actionneurs situé dans une autre partie du réseau.

Nous nous sommes donc attaché à développer un serveur utilisant le protocole SIP, ainsi que des fonctions permettant l'analyse de la demande, la gestions des périphériques ainsi que la production de réponse adéquat au moniteur.

Ce stage fut un bel aperçu d'une partie de mon futur travail qui me plait le plus. Bien sur il m'a également permis d'apprendre des notions qui n'ont pas été abordé au cours de mes études. D'un point de vue technique, la grande difficulté fut de debugger un programme « à distance ». En fait il n'y avait pas réellement moyen de voir l'état de la mémoire directement lors d'un plantage, aucun moyen de savoir à quel moment de l'exécution l'erreur se trouvait. Je marchais un peu dans le brouillard, avec une idée vague d'où venait le problème.

Concernant le point de vue relationnelle ça donne une petite idée de comment sont les relations d'équipes, la plus ou moins bonne entente entre les membre de cette équipe, les railleries etc ...

PARTIE N°6.GLOSSAIRE

GNU : GNU est un projet de système d'exploitation composé exclusivement de logiciels libres.

HTTP : Le HyperText Transfer Protocol, plus connu sous l'abréviation HTTP, littéralement le « protocole de transfert hypertexte », est un protocole de communication client-serveur développé pour le World Wide Web. HTTPS (avec S pour secured, soit « sécurisé ») est la variante du HTTP sécurisée par l'usage des protocoles SSL ou TLS.

SIP : Session Initiation Protocol (SIP) est un protocole standard ouvert de gestion de sessions souvent utilisé dans les télécommunications multimédia (son, image, etc.). Il est depuis 2007 le plus courant pour la téléphonie par internet (la VoIP).

SMTP : Le Simple Mail Transfer Protocol (littéralement « Protocole simple de transfert de courrier »), généralement abrégé SMTP, est un protocole de communication utilisé pour transférer le courrier électronique vers les serveurs de messagerie électronique.

SNMP : Simple Network Management Protocol (SNMP), protocole simple de gestion de réseau en Français, est un protocole de communication qui permet aux administrateurs réseau de gérer les équipements du réseau, superviser et de diagnostiquer des problèmes réseaux, matériels à distance.

Xbee : Module utilisant le protocole ZigBee.

ZigBee : ZigBee est un protocole de haut niveau permettant la communication de petites radios, à consommation réduite, basée sur la norme IEEE 802.15.4 pour les réseaux à dimension personnelle (Wireless Personal Area Networks : WPANs).

PARTIE N°7.BIBLIOGRAPHIE

Voici la liste des documents, ou plutôt adresses des documents qui m'ont servit tout au long de mon stage.

Documentation en ligne sur les librairies oSIP et eXoSIP :

- <http://www.antisip.com/doc/osip2/modules.html>
- <http://www.antisip.com/doc/exosip2/modules.html>

Documentation en ligne sur l'utilisation de Pthread (Librairie C)

- <https://computing.llnl.gov/tutorials/pthreads/>
- <http://franckh.developpez.com/tutoriels/posix/pthreads/>

PARTIE N°8.ANNEXES

En annexe, le code source du programme.

Annexe 1 : Lisez-moi

Pour compiler les différent fichier il faut procéder de la manière suivante :

- Se placer dans le répertoire "V2"
- Lancer la commande : "make allobj" qui crée l'ensemble des objet nécessaire pour faire le programme. Il s'agit ici de cross compilation, les objets sont compilés pour que le programme soit utilisé directement sur la plateforme ARM9.
- Lancer la commande au choix :
 - "make serv" qui compile le serveur test
 - "make serveur" qui compile serveur.c

La commande make all fait en réalité un make allobj et un make serv.

Pour compiler les fichier de test il faut les placer dans le répertoire V2, faire un make allobj (même si pas toujours nécessaire) et la commande correspondant au fichier de test. "make periph" pour le fichier test de la structure periph par exemple.

Le fichier une fois créé doit bien évidemment être placé sur la carte puis exécuté.

##Note :##

La commande heyu dans réponse.c (L.237) est désactivée pour les tests.

