POLYTECH GRENOBLE

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

INRIA

centre de recherche GRENOBLE - RHÔNE-ALPES

Gaté Jocelyn

3I-5

Rapport de stage 5ème année.

Réalisation de tests sur un réseau de capteurs : la plateforme SENSLAB

Tome principal

Année universitaire 2009 – 2010

6 avril 2010 – 17 septembre 2010

Table des matières

Table des illustrations

1 INTRODUCTION

Les progrès conjoints de la microélectronique, de la microtechnologie, des technologies de transmission sans fil et des applications logicielles ont permis de produire à coût raisonnable des micro-capteurs de faible volume, susceptibles de fonctionner en réseaux.

Le principal enjeu d'un réseau de capteurs est la gestion de l'énergie, en effet, l'idéal serait d'avoir une faible consommation, ceci assurerait une grande autonomie et ainsi moins d'interventions au niveau humain (remplacement des batteries).

Durant ce stage, le travail qui m'a été proposé, est la réalisation d'une batterie de test sur un réseau de capteurs, la plateforme SENSLAB. Plus particulièrement, sur la partie radio, qui joue un rôle prépondérant dans la consommation d'énergie et qui nécessite d'être caractérisée.

Dans ce document, je vais vous décrire l'institut de l'INRIA, ses différentes activités. Je vous présenterai aussi le SED, service dans lequel j'ai effectué mon stage de fin d'études. Enfin, je présenterai mon projet et tout son déroulement. Puis pour finir, je dresserai le bilan de ce stage.

2 PRÉSENTATION DE L'INRIA

2.1 INTRODUCTION

Fondé en 1967 pour doter la France d'une recherche et d'une industrie en informatique, l'institut a largement contribué à l'essor des sciences et technologies de l'information et de la communication.

L'INRIA, institut national de recherche en informatique et en automatique est placé sous la double tutelle des ministères de la recherche et de l'industrie. Depuis plus de quarante ans, il accompagne les mutations économiques et sociales liées à la diffusion des technologies numériques. L'institut mène au plus haut niveau international, avec ses partenaires académiques et industriels, une activité de recherche fondamentale et de développement technologique toujours plus rayonnante.

2.2 LES INSTITUTS EN FRANCE

L'INRIA se compose de 4 100 personnes réparties entre le siège et ses 8 centres de recherche situés à Rocquencourt, Rennes, Sophia Antipolis, Grenoble, Nancy, Bordeaux, Lille et Saclay. 3150 scientifiques de l'INRIA et d'organismes partenaires travaillent dans plus de 174 équipes-projets de recherche. Un grand nombre de chercheurs de l'INRIA sont également enseignants, et leurs étudiants préparent leur thèse au sein de l'institut.

Le budget de l'INRIA s'élève à 217 millions d'euros, dont 21 % de ressources propres.

L'INRIA développe de nombreux partenariats avec le monde industriel et favorise le transfert technologique et la création d'entreprises dans le domaine des STIC (Service des Technologies de l'Information et de la Communication). Plus de 98 entreprises ont été créées grâce au soutien de sa filiale INRIA-Transfert, spécialisée dans l'accompagnement, l'évaluation, la qualification et le financement des jeunes entreprises innovantes de haute technologie informatique. L'INRIA est actif au sein d'instances de normalisation comme l'IETF[1], l'ISO[2] ou le W3C[3] dont elle a été le pilote européen de 1995 à 2002.

Enfin, l'institut entretient d'importantes relations internationales : en Europe, l'INRIA est membre du consortium ERCIM (European Research Consortium for Informatics and Mathematics), qui regroupe des instituts de recherche de 20 pays européens. L'INRIA participe à une centaine d'actions dans le cadre du 7e PCRD (Programme cadre de recherche et de développement). À l'international, l'institut collabore avec de nombreuses institutions scientifiques et universitaires.

La stratégie de l'institut repose sur la combinaison étroite de l'excellence scientifique et du transfert technologique. Pour les années 2008-2012 l'INRIA se fixe de nouveaux défis à la fois scientifiques et technologiques. Les sept priorités scientifiques pour répondre aux enjeux de demain sont les suivants :

- La modélisation
- La programmation
- La communication et les réseaux
- L'interaction entre réel et virtuel
- Les sciences numériques
- L'ingénierie numérique
- La médecine numérique

2.3 L'INRIA GRENOBLE – RHONE-ALPES

Figure 1 : Le bâtiment INRIA Grenoble Rhones-Alpes (www.inria.fr)

[1] Internet Engineering Task Force
[2] http://fr.wikipedia.org/wiki/Liste_de_normes_ISO
[3] http://fr.wikipedia.org/wiki/World_Wide_Web_Consortium (World Wide Web Consortium)

C'est dans un contexte régional particulièrement porteur que s'est développée l'INRIA Grenoble - Rhône-Alpes, créé en 1992.
Quelques chiffres donnent la mesure de la croissance de l'institut depuis sa création : il comporte aujourd'hui près de 640 personnes, dont environ 190 chercheurs et enseignants-chercheurs, 200 ingénieurs, techniciens et administratifs et 220 doctorants. Ses activités sont conduites dans une trentaine d'équipes de recherche. Et, depuis 1998, 17 sociétés de technologie ont été créées à partir des recherches de l'INRIA et de ses partenaires.

L'INRIA Grenoble - Rhône-Alpes développe une politique de collaboration extrêmement étroite avec les institutions majeures de la région, par constitution d'équipes ou de laboratoires communs, souvent avec le CNRS.

Citons ainsi à Grenoble l'Université Joseph Fourier, l'Institut National Polytechnique, le CEA-Leti. Mais l'INRIA Grenoble - Rhône-Alpes est aussi implanté à Lyon, cité dont le tissu universitaire est également remarquable, et dont les activités en biologie et en médecine sont mondialement connues. Ses partenaires sont l'Ecole Normale Supérieure, l'Institut National des sciences appliquées et l'Université Claude Bernard.

2.4 QUELQUES CHIFFRES

l'INRIA Grenoble – Rhône-Alpes est composée de:

- 33 équipes de recherche

Son budget 2009 est de :

- 10,5 M€ dont 6.4 M€ en ressources contractuelles

Il est réparti sur 4 sites:

- Inovallée Meylan-Montbonnot
- Campus à Grenoble
- Domaine scientifique de la Doua
- Technopole Gerland à Lyon

2.5 LE SED

Mon stage se déroule dans le SED, je vais donc vous le présenter.
Les services d'expérimentation et de développement, appelés SED, sont présents dans tous les centres de recherche INRIA. Leurs missions sont identiques et se déclinent selon 3 axes :

- Maintenir un réseau d'expertises pour diffuser les bonnes pratiques de développement logiciel et l'utilisation d'outils communautaires au sein des équipes-projets.
- Mettre en place, développer et maintenir les plates-formes expérimentales avec les équipes-projets.
- Participer aux développements logiciels, notamment aux ADT (Actions de Développement Technologique) au sein des équipes-projets.

Au niveau national :

Pour mutualiser les efforts, les SED sont organisés en réseau national animés par la D2T (Direction du Développement Technologique).

3 LES RÉSEAUX DE CAPTEURS

3.1 QU'EST-CE QU'UN RÉSEAU DE CAPTEURS?

Un réseau de capteurs sans fil est un réseau ad hoc avec un grand nombre de nœuds qui sont des micro-capteurs capables de récolter et de transmettre des données d'une manière autonome. La position de ces nœuds n'est pas obligatoirement prédéterminée[4]. Ils peuvent être aléatoirement dispersés dans une zone géographique, appelée « champ de captage » correspondant au terrain d'intérêt pour le phénomène capté.

Chaque nœud intègre :

- Une unité de captage chargée de capter des grandeurs physiques (chaleur, humidité, vibrations, rayonnement...) et de les transformer en grandeurs numériques.
- Une unité de traitement informatique.
- Une unité de stockage de données.
- Un module de transmission sans fil.

Ces micro-capteurs sont donc de véritables systèmes embarqués.

Selon un magasine du MIT (Technology Review[5]), le réseau de capteurs sans fil est l'une des dix nouvelles technologies qui bouleverseront le monde et notre manière de vivre et de travailler. Il répond à l'émergence de ces dernières décennies, au besoin accru d'observation et de contrôle des phénomènes physiques et biologiques dans différents domaines :

- Industriels, techniques et scientifique (monitoring de la température, la pression, l'hygrométrie, la luminosité...).
- Ecologie et environnement (surveillance des UV, de la radioactivité, de polluants tels que les HAP[6], les métaux lourds)
- Santé (suivi des malades, veille éco-épidémiologique et épidémiologique.
- Sécurité.
- Transports (automatisations diverses, prévention des accidents...).
- L'automatisation des bâtiments avec la domotique.
- Etc...

Les réseaux de capteurs peuvent avoir un large champ d'action, en effet, ils peuvent être utilisés pour :

- La prévention des feux de forêt
- La mesure d'hydrométrie pour les agriculteurs (économie d'eau)
- La capture de mouvement, de données physiologiques (cf. marathon des sables ci-dessous)

[4] Réseau mobile dynamique (cf. marathon des sables)
[5] http://www.technologyreview.com/
[6] http://fr.wikipedia.org/wiki/Hydrocarbure_aromatique_polycyclique (Hydrocarbure aromatique polycyclique)

3.2 EXEMPLE DE RÉSEAU DE CAPTEURS, PROJET MARATHON DES SABLES

Cette année, le 25ème Marathon des Sables a accueilli un coureur d'un nouveau genre. Guillaume Chelius, chargé de recherche au sein de l'équipe D-Net à l'INRIA et ultra-marathonien, il a couru 250 kms, 7 jours durant, équipé de 16 capteurs qui ont enregistré en temps réel, sans interruption et sans aucune intervention humaine, de nombreuses données physiologiques, biomécaniques et environnementales. Il finit d'ailleurs classé 66ème sur un peu plus de 1000 participants.

A l'origine, il s'agit d'un défi humain par les conditions de course en environnement extrême. C'est le 25ème Marathon des Sables, dans le désert marocain, pendant 7 jours, en auto-suffisance.

En plus de cela, le défi est technologique par l'enregistrement temps-réel inédit de la position, des mouvements, de la dynamique et de la physiologie du coureur tout au long de sa course, avec des contraintes sur l'autonomie, l'ergonomie, le poids et l'encombrement du système embarqué. Les mesures sont ensuite collectées et traitées après la course pour rejouer, en virtuel, le parcours et la dynamique du coureur.

Pour finir, il s'agit aussi d'un défi scientifique, en effet ce projet a nécessité la mise au point de méthodes de traitement et d'analyse automatique de grandes quantités de données. Les données acquises durant les 7 jours de course constitueront une source d'informations inédite. Le choix du type et du positionnement des capteurs doit permettre de recueillir des éléments pertinents en termes d'étude du mouvement humain, de performances sportives et d'adaptations à l'environnement.

Le réseau de capteurs qui équipe le coureur enregistre en temps-réel :

- des données environnementales, telles que la température, l'humidité ou la luminosité
- des données physiologiques comme la fréquence cardiaque, la température sur-cutanée
- des données de macro-mobilité : position du coureur dans le référentiel terrestre, ex. GPS
- des données de micro-mobilité : la capture de mouvement de différentes parties du corps (tronc, tête, pieds, tibia, cuisse, bras) afin de pouvoir reconstruire précisément sa dynamique.

4 LE SUJET D'ETUDE

Comme on peut l'imaginer, déployer un réseau de capteurs pouvant atteindre une centaine voire un millier de nœuds capteurs peut s'avérer difficile, surtout lorsqu'il s'agit de tester et débugger un programme sur une telle quantité de nœuds. Pour faciliter cette tâche, l'INRIA, avec ses partenaires a mis en place une plateforme expérimentale permettant de tester facilement et rapidement une application pour réseau de capteurs à large échelle.

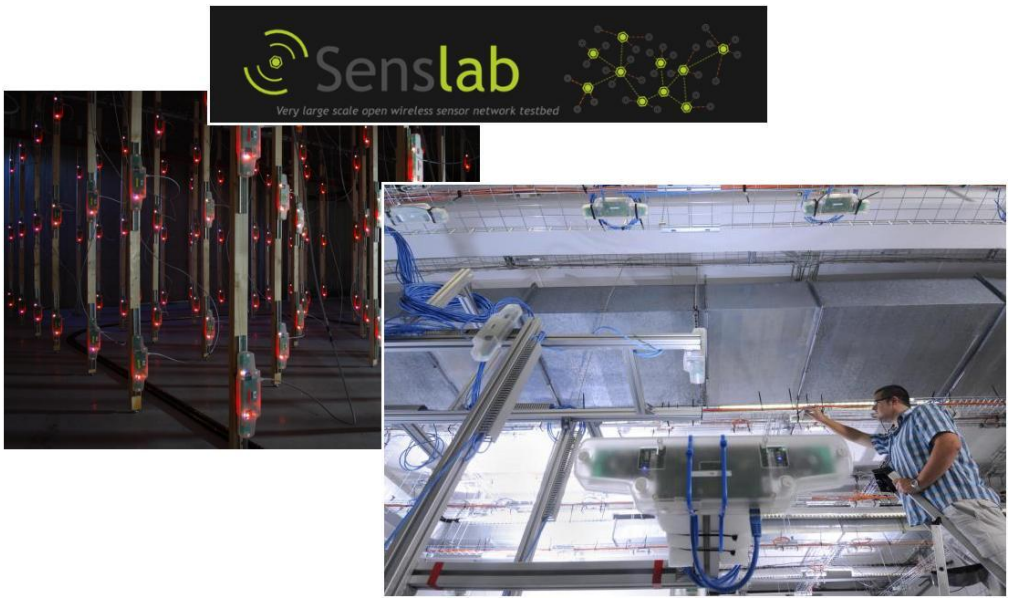


Figure 2 : Plateforme SENSLAB et son logo

SENSLAB est donc un projet hors norme puisqu'il fait partie d'une plateforme ouverte d'expérimentation nationale de réseaux de capteurs sans fil large échelle qui associent 4 sites en France (Rennes, Strasbourg, Grenoble et Lille). Sur ces 4 sites sont déployés 1024 capteurs appelés aussi nœuds. Cette plateforme permet une expérimentation réelle complètement automatisée. Les capteurs n'ont plus besoin d'être manipulés un à un. L'ensemble de l'expérience est lancé et chargé sans manipulation particulière de l'utilisateur.

Cette plateforme SENSLAB lève tous les verrous techniques qui bloquaient jusqu'à maintenant les expérimentations à grande échelle. Le fruit de ses travaux fera l'actualité dans les années à venir.

Chacune des 4 plateformes SENSLAB est composée de 256 nœuds ayant une architecture comme ci-dessous.

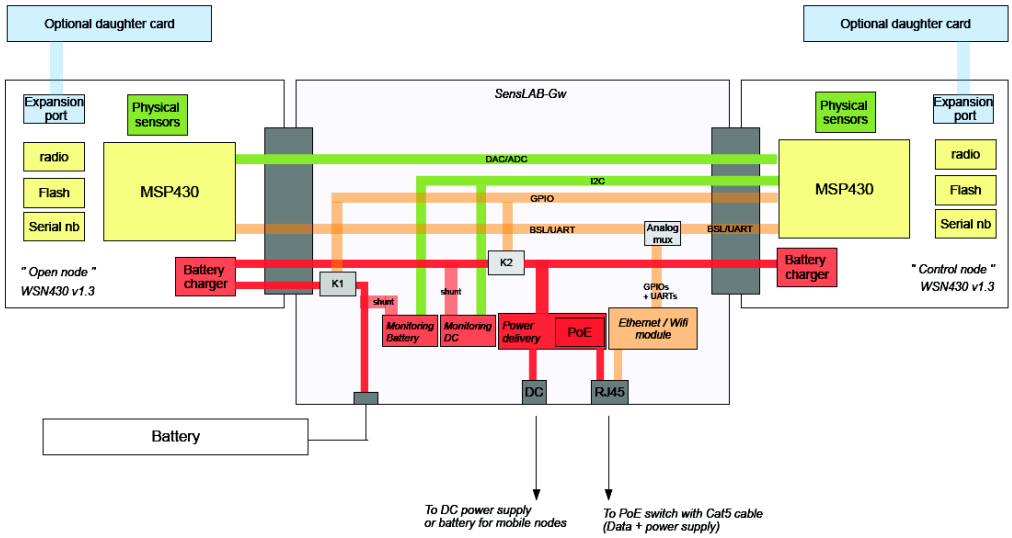


Figure 3 : Architecture d'un nœud

- ***Le nœud ouvert :***

  C'est le nœud accessible par l'utilisateur. C'est ce nœud que l'utilisateur peut programmer, et c'est grâce à celui-ci qu'il peut tester son application réseau de capteurs.

- ***La carte SENSLAB Gateway :***

  Cette carte est une passerelle qui a pour rôle de faire le lien entre le nœud de contrôle, le nœud ouvert puis le serveur.
  Les données des liaisons série de chacun des nœuds sont encapsulées et transmises au serveur via une connexion Ethernet TCP/IP[7]. L'utilisateur peut donc facilement récupérer les données provenant des liaisons séries.
  Cette carte sert aussi à la programmation du nœud ouvert.

- ***Le nœud de contrôle :***

  Le nœud de contrôle a plus un rôle de monitoring, il n'est pas accessible à l'utilisateur. C'est ce nœud qui spécifie le type d'alimentation (batterie ou secteur). En fonction des paramètres saisis lors de la création de l'expérience, on peut décider d'effectuer des mesures de consommation, de luminosité, de température, le tout échantillonné à une certaine période.

Sur chacun des nœuds peut se rajouter une carte fille sur laquelle on peut imaginer d'autre types de capteurs (accéléromètre, gyromètre, GPS, 3G...) De base, les capteurs présents sont un micro, un capteur de luminosité ainsi qu'un capteur de température.

[7] http://fr.wikipedia.org/wiki/Transmission_Control_Protocol

Ces trois cartes forment un ensemble que l'on appelle couramment nœud. Voici une image d'un nœud:

Figure 4 : Un nœud de la plateforme SENSLAB

Chaque nœud est connecté au serveur via un câble RJ45. Ce câble assure l'alimentation (recharge des batteries) ainsi que la remontée des liaisons série via le protocole de communication Ethernet TCP

En se connectant au serveur en utilisant un protocole de communication sécurisé (ssh)[8] sur une machine virtuelle[9], on peut facilement créer une expérience. Ensuite, on peut programmer et dialoguer avec chacun des nœuds de l'expérience.

Figure 5 : Le serveur et les nœuds de la plateforme SENSLAB Grenoble

[8] http://fr.wikipedia.org/wiki/Secure_Shell
[9] http://fr.wikipedia.org/wiki/Machine_virtuelle

5 LE DÉROULEMENT DU STAGE

Je vais maintenant vous détailler le déroulement de mon stage. Je commencerai par faire une présentation du matériel que j'ai utilisé, je vous présenterai par la suite le fonctionnement de la plateforme plus en détail, j'identifierai le cahier des charges, viendra le planning prévisionnel puis ensuite le vif du sujet qui recensera les différents points développés tout au long de mon stage. Enfin j'évoquerai les différents problèmes rencontrés avant de finir par un bilan.

5.1 DESCRIPTION DU MATERIEL UTILISÉ

5.1.1 Le nœud ouvert

Voici une photo du nœud ouvert. C'est principalement ce nœud qui sera utilisé lors de toutes les manipulations.

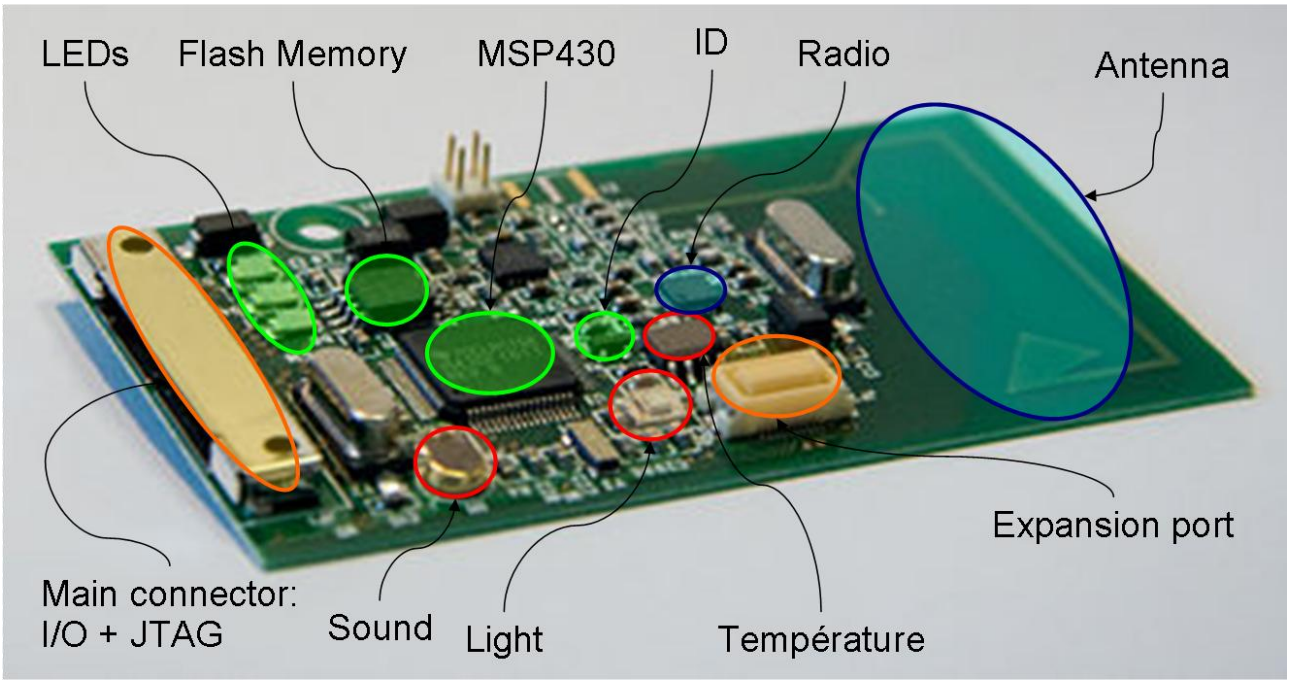


Figure 6 : Les principaux composants du nœud ouvert (www.grenoble.senslab.info/)

Dans cette partie je vais vous présenter les composants de la carte nœud ouvert qui m'ont été les plus utiles. Je commencerai par le microcontrôleur de chez Texas Instrument MSP430, ensuite le chip radio du même constructeur puis pour finir le chip identifiant unique.

5.1.2 Le micro contrôleur (Texas Instrument MSP430)

Construit autour d'un CPU 16 bits, le MSP430 a été conçu pour des applications embarquées à basse consommation et à faible coût. Il est particulièrement adapté aux applications sans-fil fonctionnant sur batteries.

Figure 7 : Le chip MSP430 (www.focus.ti.com)

Le composant est décliné en une série de configurations comprenant les périphériques usuels :

- convertisseurs A/D 10/12/14/16 bits,
- convertisseurs D/A 12 bits, comparateurs,
- interfaces USART, SPI, I2C, pilote LCD, watchdog, multiplicateur hardware, DMA, oscillateur interne, etc.

Tous ces circuits sont programmables par JTAG ou BSL.

Le MSP430 est une option très répandue pour les appareils de mesure à basse consommation alimentés par batterie. Son mode de fonctionnement en attente consomme moins de 1 microampère.

Il a toutefois des limitations qui l'empêchent d'être utilisé dans des systèmes embarqués plus complexes. Par exemple, il n'a pas de bus mémoire externe et sa capacité mémoire peut se révéler insuffisante pour des applications qui demandent de grandes tables de données.

5.1.3 Le chip radio

Le chip CC1100 de chez Texas Instrument est un émetteur/récepteur radio pour des fréquences inférieures à 1GHz bon marché. Il est connu pour sa faible consommation.

Ces bandes de fréquences sont :

- 315 MHz / 433 MHz / 868 MHz / 915 MHz

Il est contrôlé en SPI et supporte de nombreuses modulations (2-FSK, GFSK, MSK, OOK, ASK)

La transmission de données peut se faire jusqu'à 500 kBaud.

Figure 8 : Le chip cc1100 (www.focus.ti.com)

5.1.4 Le chip qui assure un numéro de série unique (ds2411)

Le chip DS2411 fournit une identité absolument unique. Ce numéro de série peut être déterminé avec une interface électronique minimale (un seul fil de donnée et un protocole de communication propriétaire).

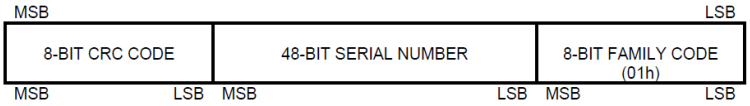


Figure 9 : Identifiant donné par le chip ds2411

L'immatriculation du chip est codée sur 64 bits. Elle inclut un numéro de série de 48 bits, un CRC de 8 bits ainsi que 8 autres bits correspondant à sa famille.

Après avoir présenté les différents composants de la carte nœud ouvert, je vais vous présenter le fonctionnement de la plateforme. Je ferais un rapide descriptif de celle-ci et ensuite, je vous expliquerais son fonctionnement.

5.2 DESCRIPTION DU FONCTIONNEMENT DE LA PLATEFORME

La plateforme est composée d'un grand nombre de ressources (1024 nœuds) et est multiutilisateur. Il y a donc la notion d'expérience à retenir, on appellera expérience, la réservation d'un nombre N de nœuds pendant une durée T avec différents paramètres utilisateur. Il faut donc un système de réservation et d'allocation des ressources. En plus de cela, il faut un système de communication qui établisse un lien entre les nœuds et l'utilisateur puis un système qui permette la programmation à distance des nœuds réservés.
Dans cette partie, je vais décrire le fonctionnement du système de réservation des nœuds, la façon de programmer les nœuds puis le fonctionnement de la communication nœud-utilisateur.

5.2.1 Le système de réservation des nœuds (OAR)

OAR est un système batch basé sur une base de données, un langage de script (Perl) et un outil administratif évolutif. Il est composé de modules qui interagissent seulement avec la base de données. Cette approche facilite le développement de modules spécifiques. En effet, chaque module peut être développé dans n'importe quel langage ayant une bibliothèque d'accès de base de données.
Caractéristiques principales :

- Règles d'admission
- Correspondance de ressources
- Files d'attente multiples avec priorité
- Outils de visualisation d'activités
- Réservation avancée

Ainsi, les nœuds défectueux ou ayant des anomalies peuvent être décelés et déclarés non accessibles à la réservation. De plus lors de la création d'une nouvelle expérience, s'il y a déjà une expérience en cours et qu'il n'y à pas assez de ressources disponibles, l'expérience est mise en attente jusqu'à ce que les ressources soient libérées. Elle pourra ensuite débuter.

Une des perspectives à la suite de mon stage pourrait être la création d'une règle d'admission qui diviserait la plateforme en différents clusters. Ceci permettrait par exemple à l'utilisateur de choisir un ensemble de nœuds dont l'interconnexion est fiable, ou encore, un ensemble de nœuds qui ne peuvent pas communiquer directement entre eux. L'utilisateur pourrait ainsi tester des protocoles de communication du type routage à saut multiple. Ce type de routage permet d'utiliser le chip radio à moindre puissance et ainsi minimiser la consommation de chacun des nœuds.

5.2.2 La programmation de chacun des nœuds

Pour commencer, il faut réaliser le code source du firmware que l'on enverra par la suite dans les nœuds. Le langage utilisé pour la création de la source du firmware est le langage C. Une fois les sources développées, il faut générer le binaire par cross-compilation puis ensuite l'envoyer dans les nœuds désirés. Je ne vais pas détailler la création du code source, ici on s'intéresse plutôt a la création du binaire puis à l'envoi de celui-ci dans chacun des nœuds.

5.2.2.1 Génération du binaire par cross-compilation

Un compilateur croisé (en anglais Cross Compiler) est un programme capable de traduire un code source en code objet ayant un environnement d'exécution différent de celui où la compilation est effectuée. Ces compilateurs sont principalement utilisés en informatique industrielle.

Le monde du logiciel libre offre une solution MSPGCC utilisable sur plusieurs plateformes. Cet environnement est basé sur le compilateur et débogueur GNU. C'est cette solution que l'INRIA a décidé d'utiliser.

5.2.2.2 Envoi du binaire dans le micro contrôleur du ou des nœuds désirés

Pour la création de l'expérience, un portail web est disponible, il est complet et très simple d'utilisation, voici à quoi il ressemble:

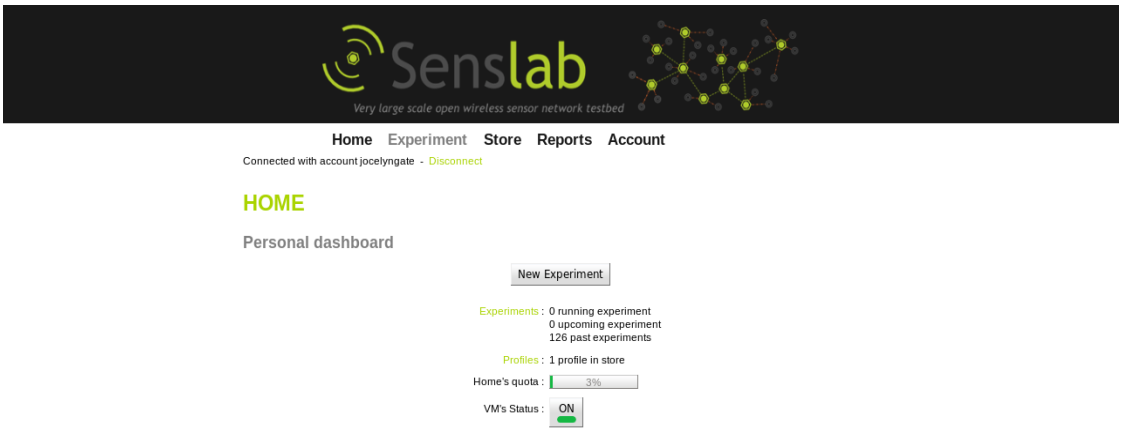


Figure 10 : Aperçu du portail web SENSLAB (www.grenoble.senslab.info/)

L'utilisateur choisit le nombre de ressources dont il a besoin lors de son expérience, il affecte ensuite le firmware à tel ou tel nœud, il spécifie la durée de l'expérience. Il a la possibilité dans une même expérience d'utiliser plusieurs firmwares différents. Il peut aussi demander l'acquisition de différentes mesures à des échantillonnages donnés (consommation, luminosité, température…), il choisit la durée de l'expérience puis, pour finir demande le démarrage de celle-ci.

Après avoir expliqué le mécanisme de la création d'une expérience, je vais expliquer le fonctionnement de la communication entre les nœuds et l'utilisateur.

5.2.3 Le dialogue avec chacun des nœuds

Dès que l'on a créé une expérience, on a la possibilité de communiquer avec chacun des nœuds. Pour cela, il suffit, à partir de sa machine virtuelle, d'ouvrir les sockets TCP, on a ainsi accès aux liaisons série de chacun des nœuds. Une connexion sur le port de base + i établira un lien entre le nœud i de l'expérience et l'utilisateur. Il faudra autant de connexions que de nœuds dans l'expérience pour pouvoir communiquer avec tous les nœuds. Le fonctionnement des communications TCP est similaire à celui des communications séries du point de vue de la programmation PC haut niveau.

On retrouve les notions :

- D'ouverture
- De connexion
- De lecture
- D'écriture
- De fermeture

5.3 LE CAHIER DES CHARGES

L'objectif du stage est de fournir un ensemble d'outils de caractérisation et de test de la plateforme. La plateforme SENSLAB étant en cours de déploiement, il est important de pouvoir

chiffrer un certain nombre de paramètres comme la couverture radio du réseau de capteurs, la qualité des liens entre nœuds capteurs, le débit maximal de données autorisé.

Le cahier des charges peut donc se diviser en plusieurs points :

- Réalisations de différentes mesures relatives à l'utilisation du chip radio embarqué (cc1100)
  - Puissances des signaux radio reçus
  - Nœuds voisins (réussite de communication supérieure à un seuil)
  - Débit maximal de données

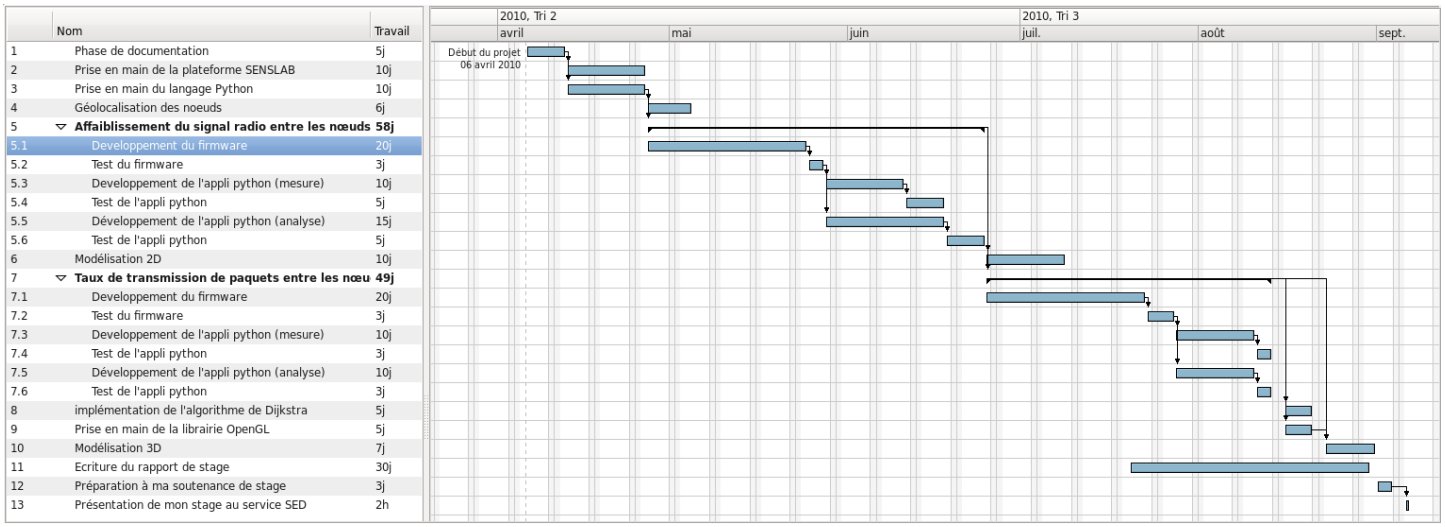
- Développement d'un outil permettant de fournir des informations de caractérisation radio de la plateforme.
  - Courbes
  - Histogrammes
  - Séries de données
  - Modélisations

- L'outil doit être utilisable sur les autres plateformes
  - Windows, linux,
  - SENSLAB GRENOBLE, LILLE, RENNES, STRASBOURG

5.4 LE PLANNING PRÉVISIONNEL

Dès le début du stage, j'ai réalisé un planning prévisionnel pour avoir une idée de mon avancement.
Les treize tâches les plus importantes à mes yeux sont :

- Documentation
- Prise en main de la plateforme SENSLAB
- Prise en main du langage Python
- Développement de la solution permettant la localisation des nœuds
- Mesure/analyse/affichage de l'affaiblissement du signal radio entre les nœuds
- Modélisation 2D de la plateforme
- Mesure/analyse/affichage du taux de transmission de paquets entre les nœuds
- Implémentation de l'algorithme de Dijkstra
- Prise en main de la librairie OpenGL
- Modélisation 3D de la plateforme
- Écriture du rapport de stage
- Préparation de ma soutenance
- Présentation de mon stage au service SED



	Nom	Travail
1	Phase de documentation	5j
2	Prise en main de la plateforme SENSLAB	10j
3	Prise en main du langage Python	10j
4	Géolocalisation des noeuds	6j
5	**Affaiblissement du signal radio entre les nœuds**	**58j**
5.1	Developpement du firmware	20j
5.2	Test du firmware	3j
5.3	Developpement de l'appli python (mesure)	10j
5.4	Test de l'appli python	5j
5.5	Développement de l'appli python (analyse)	15j
5.6	Test de l'appli python	5j
6	Modélisation 2D	10j
7	**Taux de transmission de paquets entre les nœu**	**49j**
7.1	Developpement du firmware	20j
7.2	Test du firmware	3j
7.3	Developpement de l'appli python (mesure)	10j
7.4	Test de l'appli python	3j
7.5	Développement de l'appli python (analyse)	10j
7.6	Test de l'appli python	3j
8	implémentation de l'algorithme de Dijkstra	5j
9	Prise en main de la librairie OpenGL	5j
10	Modélisation 3D	7j
11	Ecriture du rapport de stage	30j
12	Préparation à ma soutenance de stage	3j
13	Présentation de mon stage au service SED	2h

Figure 11 : Diagramme de Gantt

5.5 LES DIFFÉRENTS POINTS À DÉVELOPPER

5.5.1 La localisation des nœuds

A mon arrivée à l'INRIA, la localisation géographique des nœuds de l'expérience était impossible. En effet, seulement une seule partie des nœuds avait été repérée sur la plateforme. Un fichier contenant les coordonnées des différents nœuds existe mais, seulement 100 nœuds sur 256 sont renseignés. De plus, même en connaissant les positions de chacun des nœuds, on ne peut pas savoir quel nœud correspond à quelle connexion. En effet, le numéro de la connexion d'un nœud donné est différent suivant les expériences.
Le seul moyen d'avoir une correspondance entre le nœud de connexion i et le nœud réel i est de réserver toute la plateforme. Or une telle réservation empêche les autres utilisateurs de faire d'autres tests en même temps et vu le temps que dure les mesures, c'est à éviter!

L'idée est donc d'utiliser le chip ds2411 qui fournit à chaque nœud une immatriculation unique sur 64 bits. On peut ainsi créer un allias qui associe l'identifiant au numéro sous lequel est repéré le nœud.

J'ai ensuite effectué une expérience sur toute la plateforme sur une durée réduite (moins d'une heure) pour pouvoir créer la table d'alias. Voici à quoi elle ressemble.

```
node2id = {   1:"b2ce",6:"bfc6",5:"b27c",2:"bdc0",3:"cdf2",7:"b39e",4:"c6c0",12:"c1fe",
              9:"c7e6",8:"b07f",14:"b2ca",15:"b020",11:"bb40",10:"beed",16:"b6d8",13:"b807",
              19:"b03d",17:"c631",26:"b1cb",18:"cc8b",25:"bed2",20:"c24c",27:"b94f",24:"c13d",
              23:"bc97",21:"cc0d",22:"b047",28:"c33e",29:"1cbe",34:"b902",37:"cf33",30:"ccc8",
              etc ...
          }
id2node = {   "b2ce":1,"bfc6":6,"b27c":5,"bdc0":2,"cdf2":3,"b39e":7,"c6c0":4,"c1fe":12,
              "c7e6":9,"b07f":8,"b2ca":14,"b020":15,"bb40":11,"beed":10,"b6d8":16,"b807":13,
              "b03d":19,"c631":17,"b1cb":26,"cc8b":18,"bed2":25,"c24c":20,"b94f":27,"c13d":24,
              "bc97":23,"cc0d":21,"b047":22,"c33e":28,"1cbe":29,"b902":34,"cf33":37,"ccc8":30,
              etc ...
          }
```

Figure 12 : Table des alias (bidirectionnelle)

64 bits pour identifier 256 capteurs, c'est immense, j'ai donc limité la longueur de l'identifiant à 16 bits en tronquant l'identifiant de 64 bits et en m'assurant de l'absence de doublons.

Cette table d'alias est sauvegardée sous forme d'objet python. Il s'agit en fait de deux 'dictionnaires'. Ce type d'objet est très intéressant, en effet, grâce à celui-ci on peut indexer un tableau avec à peu près n'importe quel type de données (entiers, chaîne de caractère, ...)

Ainsi lorsque le nœud i nous renvoie son unique identifiant, un simple « id2node[identifiant] » nous permet de connaitre son réel numéro et ainsi sa localisation. On peut donc créer à chaque début d'expérience un autre alias qui est la correspondance entre l'offset à la connexion et le numéro réel du nœud.

Après avoir implémenté la localisation des nœuds, je me suis attaqué à la mesure de l'affaiblissement du signal radio entre les nœuds.

5.5.2 Affaiblissement du signal radio entre les nœuds

La mesure de l'affaiblissement du signal radio passe par trois importantes étapes, la première est le développement du firmware embarqué, ensuite il faut réaliser l'application coté PC qui contrôle et récupère les informations provenant des différents nœuds de l'expérience en cours puis stocke les données utiles dans un fichier. Enfin, il faut créer une application encore coté PC qui exploite ces données, trace des courbes, des histogrammes…
Coté nœud, le firmware est développé en langage C-ANSI, coté PC, les applications sont développées en Python, un langage objet très haut niveau. Il permet le développement rapide d'application avant de les optimiser avec un langage de plus bas niveau si besoin est.
Pour la mesure des affaiblissements en puissance, on ne va pas pour l'instant envoyer de paquets, on va plutôt utiliser le fait que le chip radio puisse envoyer indéfiniment un préambule[10]. Le chip offre aussi la possibilité de lecture de la puissance du signal reçu instantanément et cela, même s'il n'y a pas de signal émis. Ce sont ces deux propriétés que l'on va utiliser dans cette partie.

5.5.2.1 Développement du firmware (embarqué, coté microcontrôleur)

Voici le principe de fonctionnement du firmware embarqué sur chacun des nœuds dans l'expérience. Le principe est assez simple, à chaque fois que le nœud reçoit un octet, une interruption se déclenche. Il s'agit de « UART0 CALLBACK ».

Ensuite, si l'octet reçu correspond à:

- Une commande de lecture de RSSI, le flag rssi est levé (rssi :=1)
- Une commande d'initialisation de la radio, le flag reset_RX est levé (reset_RX :=1)
- Une commande de changement de puissance d'émission, le flag Switch_power est levé et la valeur de l'octet reçu est stockée dans la variable globale POWER.

Pour finir, le programme principal est une boucle infinie qui scrute en permanence l'état des flags et s'ils sont levés, il effectue la tâche et abaisse le flag en question.

[10] Suite de zéros et de uns « 0101010101010101010101….0101 »

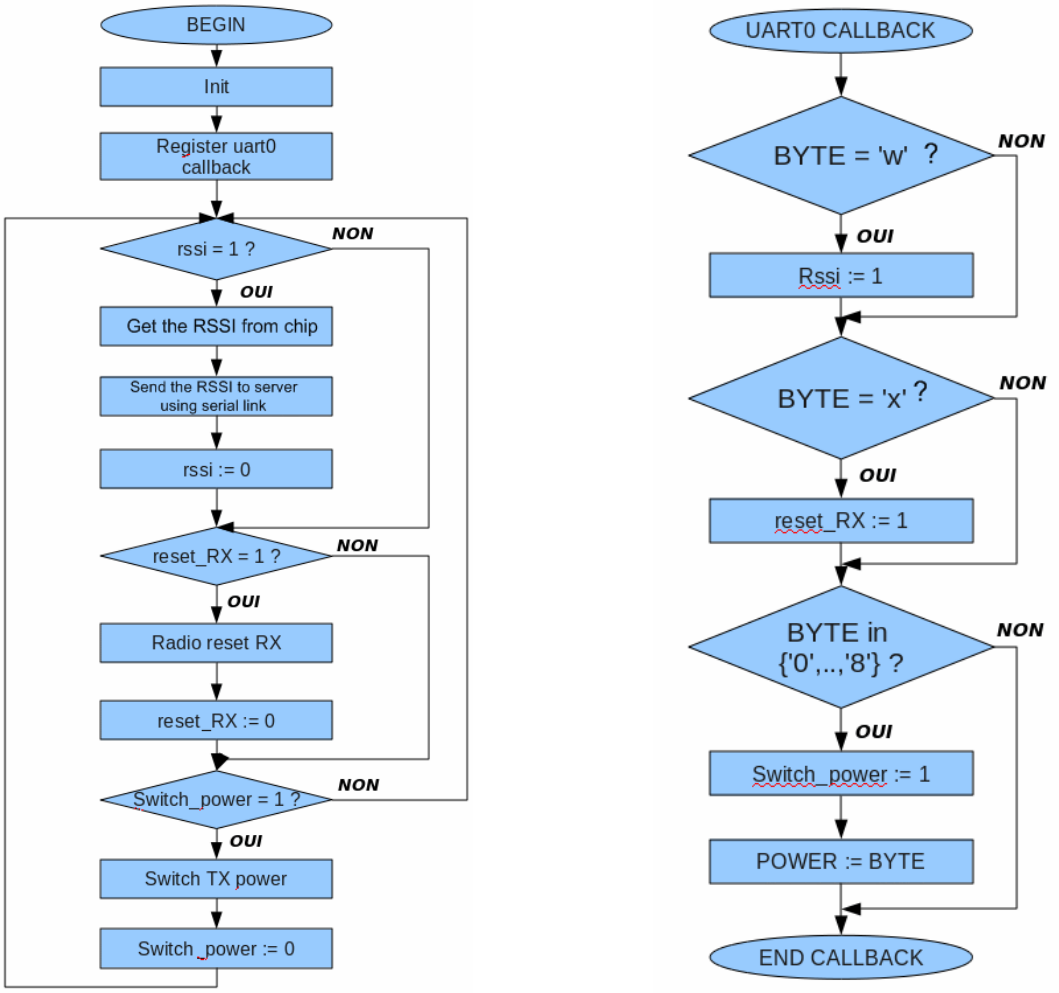


Figure 13 : Boucle principale du firmware embarqué et le Callback UART0 (interruption)

5.5.2.2 Développement de la partie récupération des données (coté PC)

Voici le principe de fonctionnement de la partie logicielle qui se charge de récupérer les données.

On commence par lancer une recherche des connexions TCP possibles, dès qu'une connexion réussie et que le nœud nous renvoie un identifiant valide. On utilise la table de localisation puis on affecte à cette connexion le numéro réel du capteur grâce à un allias.

Ensuite, on crée une matrice vide de taille 256x256. Dans cette matrice seront stockées les valeurs mesurées des RSSIs sur les récepteurs pour chaque émetteur.

On effectuera ce mécanisme pour les différentes puissances d'émission possibles et on sauvegardera les résultats dans un fichier « .csv » directement exploitable dans n'importe quel tableur.

Une fois terminé, on ferme les connexions TCP et on quitte le programme.

Nous avons donc 8 fichiers « .csv » dans lesquels sont sauvegardées toutes les valeurs de RSSI pour le récepteur i quand c'est l'émetteur j qui émet à une puissance de X dBm.

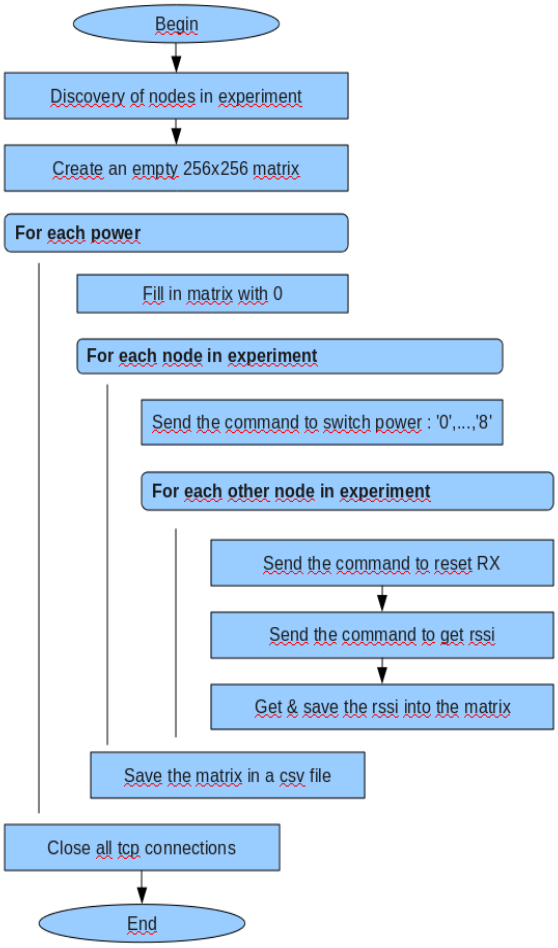


Figure 14 : Diagramme du fonctionnement de la partie mesure

RSSIs		RECEPTEUR					
		1	2	3	...	255	256
EMETTEUR	1	0	-67	-56	...	-70	-71
	2	-66	0	-61	...	-65	-70
	3	-58	-60	0	...	-59	-65
	...	...	...	...	0	...	...
	255	-60	-62	-60	...	0	-67
	256	-72	-71	-56	...	-72	0

Figure 15 : Matrice contenant les RSSIs mesurés

Avec cette application, on peut également créer le vecteur du bruit radio ambiant. Pour cela, on s'assure qu'aucun nœud n'est en émission radio et on lance les mesures. Si l'on a ce vecteur, on pourra par la suite, récupérer la valeur maximale du bruit radio ambiant et ainsi avoir un seuil maximum pour se mettre dans le cas le plus défavorable lors de l'analyse des données, ou encore tracer le rapport signal bruit en fonction de la distance pour avoir une idée de la qualité des liens radios.

5.5.2.3 Développement de la partie analyse des données (coté PC)

- **Partie distribution des affaiblissements**

Voici le principe de fonctionnement de la partie logicielle qui se charge d'analyser les données précédemment récupérées.

Tout d'abord, on doit saisir 2 paramètres:

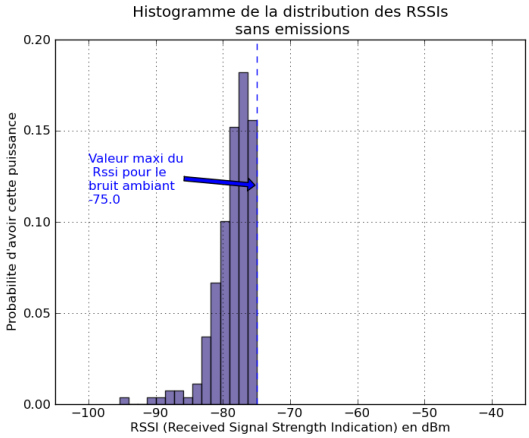
- La puissance d'émission
- Le numéro du nœud émetteur

Ensuite le programme parcourt la matrice, récupère les valeurs de RSSI.

Après, plusieurs approches sont possibles, soit on raisonne en terme de puissance en gardant en vue la valeur maximale du bruit radio ambiant, soit on fait ressortir le rapport signal à bruit.

Dans chacun des cas une liste avec les valeurs utiles est remplie.

Pour finir, cette liste est passée en paramètres à la fonction qui trace l'histogramme de la distribution des puissances reçues ou bien du rapport signal à bruit.



Ici, le graphe de la distribution des puissances reçues sans aucuns émetteurs. Il s'agit de la distribution du bruit radio ambiant aux alentours de la plateforme SENSLAB.

La valeur maximale du bruit radio ambiant est de -75 dBm. On utilisera cette même valeur lors de la création des prochains graphes comme seuil critique.

Voici les graphes de la distribution des puissances du signal reçu pour le nœud 56 à la puissance de +10 dBm puis de 0 dBm.

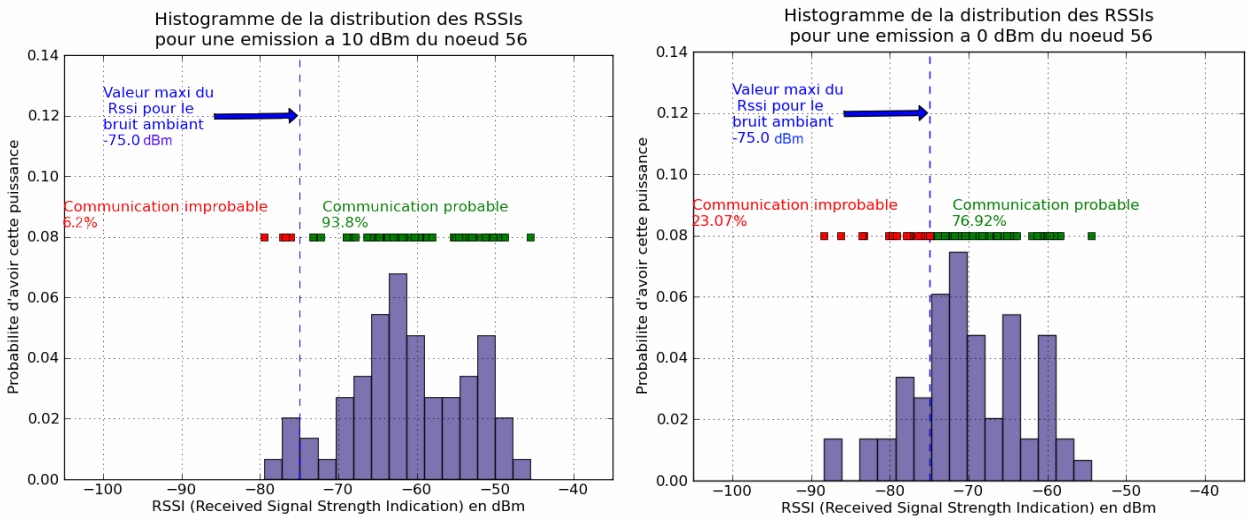


Figure 16 : Graphe des distributions des RSSIs

Sur ces graphes sont affichés le pourcentage de nœuds ayant une puissance de réception inférieure/supérieure à la valeur maximale du bruit ambiant, les pointillé verticaux représentent cette valeur maximale.

On notera que plus la puissance d'émission diminue, plus le signal reçu a tendance à être noyé dans le bruit ambiant.

On peut donc en déduire que plus la puissance diminue, plus le réseau de nœuds communiquant devient petit et que plus les liens inter-nœuds perdent en qualité.

- **Partie affaiblissements en fonction de la distance**

Voici le principe de fonctionnement de la partie logicielle qui se charge d'analyser les données précédemment récupérées.

Tout d'abord, on doit saisir 2 paramètres:

- La puissance d'émission
- Le numéro du nœud émetteur

Ensuite le programme parcourt la matrice, récupère les valeurs de RSSIs.

Ensuite, deux approches sont possibles, soit on raisonne en terme de puissance en gardant en vue la valeur maximale du bruit radio ambiant, soit on raisonne en terme de rapport signal a bruit.

On calcule les distances émetteur-récepteur grâce au théorème de Pythagore.

On ajoute toutes les données utiles à une liste.

Pour finir, cette liste est passée en paramètres à la fonction qui trace le nuage de points représentant la puissance du signal reçu en fonction de la distance ou encore le rapport signal à bruit en fonction de la distance (plutôt en fonction du logarithme de la distance)[11]

[11] La théorie prévoit une décroissance en r^2

Voici les graphes générés automatiquement de la puissance de réceptions (RSSI) en fonction de la distance (log10 de la distance) pour le nœud émetteur n°94 à la puissance de +10 dBm puis de 0 dBm.

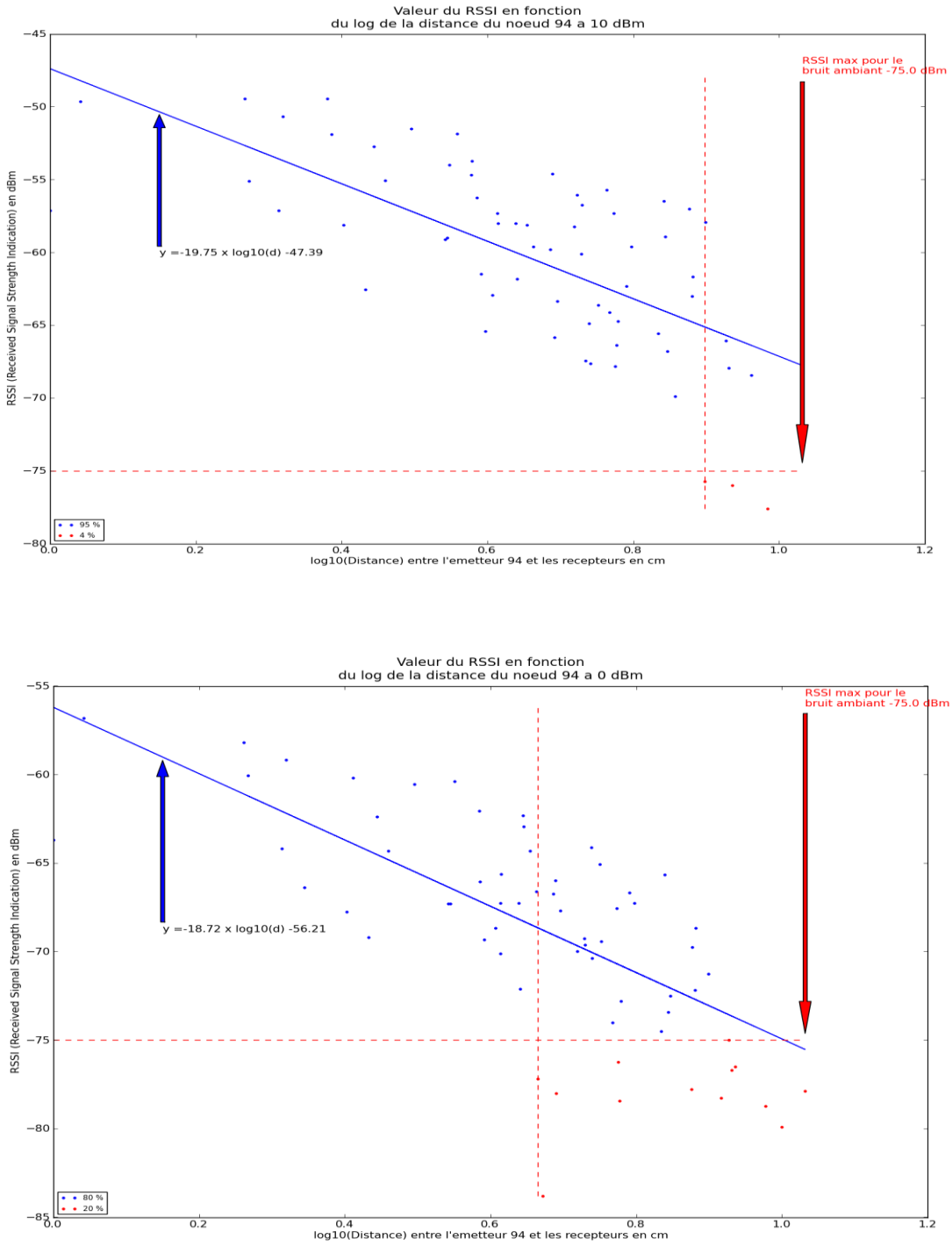


Figure 17 : Puissance reçue en fonction de la distance

Les pointillés horizontaux représentent la valeur maximale du RSSI mesuré pour le bruit ambiant. Les pointillés verticaux représentent l'image de la distance à partir de laquelle au moins un nœud capte une puissance inférieure à -75dBm (le signal reçu est noyé dans le bruit).

On peut dire que pour le nœud 94 et seulement celui-ci et à une puissance d'émission de +10dBm que la distance est discriminante jusqu'à 7m90[12] alors qu'à une puissance d'émission de 0dBm la distance est discriminante jusqu'à 4m70[13].

Il faut tout de même noter que pour les graphes précédents, on se place à chaque fois dans le pire des cas!

J'ai fait quelques calculs de puissances reçues en fonction de la puissance émise et en me basant sur la formule de Friis[14] en espace libre:

$$P_r = G_e.G_r.P_e.\left[\frac{\lambda}{4.\pi d}\right]^2$$

avec : Pr, Pe les puissances de réception et d'émission en W

Ge, Gr les gains des antennes (sans unité)

D la distance entre émetteur et récepteur en m

λ la longueur d'onde de l'onde radio en m

On commence tout d'abord par passer en log de manière à avoir une équation du premier ordre :

$$P_r = G_e.G_r.P_e.\left[\frac{\lambda}{4.\pi d}\right]^2 \text{ est équivalente à : } P_r = P_e + G_r + G_e - 20\log(d) - 20\log(f) + 20\log(4\pi c)$$

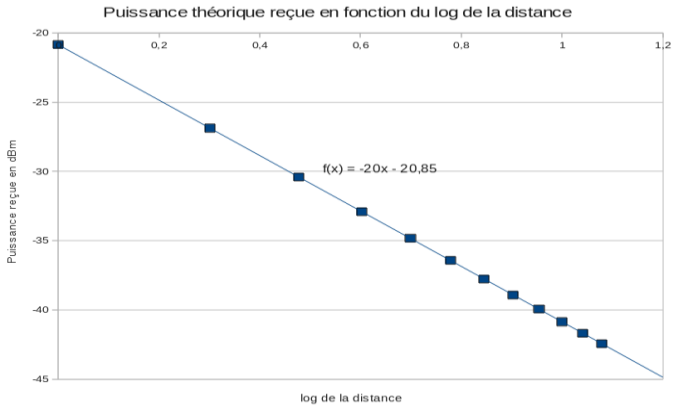
avec : Pr, Pe les puissances de réception et d'émission en dBm

Ge, Gr les gains des antennes en dBi

D la distance entre émetteur et récepteur en m

f la fréquence de l'onde radio en Hz

Si l'on trace pour une puissance donnée la puissance reçue en fonction du log de la distance entre les deux antennes, on obtient une droite. Le coefficient de la courbe est de -20. On retrouve à peu près ce coefficient sur les deux courbes précédentes. Les mesures sont donc en accord avec la théorie et les écarts sont dus à différents phénomènes que je cite dans le prochain paragraphe.



[12] Log10(distance) = 0,90 donne : distance = 7,9m
[13] Log10(distance) = 0,67 donne : distance = 4,7m
[14] http://fr.wikipedia.org/wiki/Équation_des_télécommunications

5.5.2.4 Analyse des données

Une fois la localisation des nœuds implémentée, j'ai pu croiser les données de puissance du signal reçu avec les distances.

Il est clair que l'affaiblissement augmente avec la distance mais on ne peut pas dire que la distance soit un réel facteur discriminant sur l'ensemble des nœuds, en effet, de nombreux phénomènes font apparaitre des affaiblissements entre les différents nœuds.

On peut retrouver des phénomènes de :

- Réflexion (réflexion sur une surface comme de l'eau, le sol, une voiture)
- Réfraction (changement de direction du au passage de l'onde milieu d'un indice n1 à n2)
- Diffraction (rencontre d'obstacles grand par rapport à la longueur d'onde)
- D'absorption (une partie de l'énergie de l'onde est absorbée et transformée en énergie)

On peut donc dire que l'affaiblissement du signal est lié au fait que l'on ne soit pas en extérieur et que de nombreux obstacles métalliques sont présents à proximité des nœuds.

5.5.3 Taux de transmission de paquets entre les nœuds

Après avoir récupéré les données concernant les affaiblissements des signaux, je me suis attaqué à la détermination des voisins. Par voisin j'entends réussite de communication supérieure à un seuil donné. Par exemple, on considère que A est voisin avec B si B reçoit plus de 90% de ce que A envoie.

Cette partie s'est elle aussi déroulée en quatre étapes que je vais vous détailler par la suite.

5.5.3.1 Développement du firmware (embarqué, coté microcontrôleur)

Dans cette partie, le firmware est plus compliqué, en effet, la partie communication est bien plus sophistiquée. Toutes les données envoyées et reçues sont encapsulées, il y a même un contrôle de redondance cyclique le CRC[15].

De nombreuses autres fonctionnalités ont été ajoutées, modification du type de modulation, reset du nœud, envoi de paquets en mode BURST[16], notification quand le nœud démarre ou redémarre, notification quand le nœud reçoit un paquet, etc...

Dans le premier firmware, il n'y avait qu'une interruption, maintenant une deuxième est présente, il s'agit du callback radio, à chaque fois qu'un paquet est reçu via le chip radio, la fonction callback est appelée et effectue sa tâche.

[15] Certifie qu'il n'y a pas eu d'erreurs de transmission
[16] Envoie consécutif de paquets

Chaque fois qu'un paquet est reçu par la radio, il est dés-encapsulé puis notifié par l'envoi d'une trame sur la liaison série (UART0) contenant différentes informations.

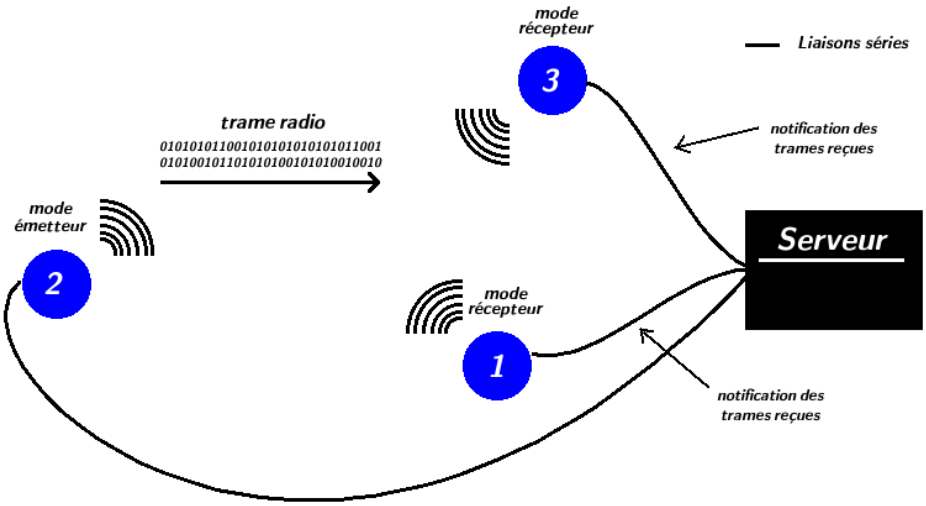


Figure 18 : Illustration des fonctionnalités du firmware

5.5.3.2 Développement de la partie récupération des données (coté PC)

Voici le principe de fonctionnement de la partie logicielle qui se charge de récupérer les données.

On commence par lancer une recherche des connexions TCP possibles, dès qu'une connexion réussie et que le nœud nous renvoie un identifiant valide. On utilise la table de localisation puis on affecte à cette connexion le numéro réel du capteur grâce à un allias.

Ensuite il s'agit de créer un thread par nœud, en effet il faut pouvoir paralléliser la lecture et l'écriture sur les ports séries.

Chaque thread permet l'envoi de commande à un nœud mais aussi la sauvegarde dans un fichier '.csv' de tout ce que ce nœud a reçu par la liaison radio avec de nombreuses autre valeurs telles que:

- la source de la trame radio
- l'heure
- les données encapsulées
- la puissance du signal

Une fois tous ces threads créés, on peut commencer la série de mesure. On envoie la commande utile pour mettre le premier nœud en émission à la puissance voulue puis on déclenche l'envoie de N paquets. On effectue la même manipulation pour tous les nœuds et pour toutes les puissances désirées.

On obtient ainsi autant de fichier qu'il y avait de nœuds dans la série de mesure.

Les fichiers sont formatés comme cela :

n1.csv

```
time,               src,    burstid,    burstsize,  extralen,   rssi
1280418827.185469,  2,      512,        3,          0,          -62.5
1280418840.422183,  3,      512,        3,          0,          -60
1280418853.058864,  4,      512,        3,          0,          -70.5
1280418866.551591,  5,      512,        3,          0,          -63.5
1280418880.304332,  6,      512,        3,          0,          -68
1280418893.417039,  7,      512,        3,          0,          -57.5
1280418907.221782,  8,      512,        3,          0,          -56.5
1280418921.226537,  9,      512,        3,          0,          -71.5
1280418935.063282,  15,     512,        3,          0,          -70
1280418947.899974,  16,     512,        3,          0,          -57
1280418960.308643,  17,     512,        3,          0,          -58
1280418973.741366,  18,     512,        3,          0,          -65
...
```

Figure 19 : Aperçu du fichier '.csv' généré et relatif au nœud 1

5.5.3.3 Développement de la partie analyse des données (coté PC)

- Affichage des voisins d'un nœud donné

L'utilisateur commence par choisir le nœud pour lequel il veut visualiser les résultats. Pour cela il le sélectionne dans la liste de gauche. Dans ce cas, c'est le nœud 51 qui est sélectionne.

Ensuite, libre à lui de saisir un seuil différent, le seuil est en fait un pourcentage qui traduit le succès de transmission de paquets, la valeur par défaut est de 90%.

Après, l'utilisateur choisit la puissance d'émission de l'émetteur, ici une faible puissance est choisie (-15dBm), on pourra s'attendre à un faible nombre de voisins

Enfin, pour visualiser les résultats, il suffit d'appuyer sur le bouton 'Voisins'.

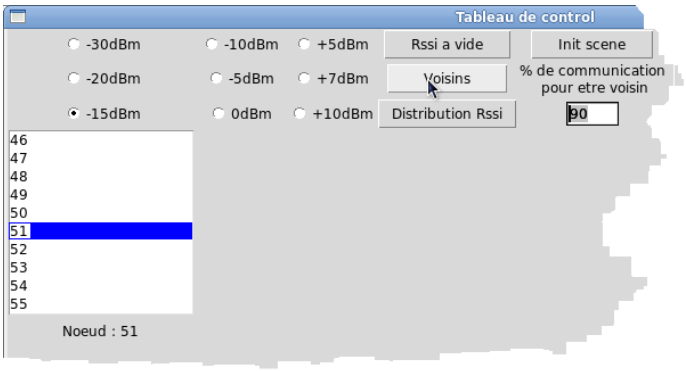


Figure 20 : IHM : saisie des paramètres pour l'affichage des voisins

- Le nœud en gris est l'émetteur.
- Les nœuds en vert sont les nœuds pour lesquels la réussite d'envoi de paquets est supérieure à 90% (valeur personnalisable par l'utilisateur).
- Les nœuds en jaune sont les nœuds pour lesquels la réussite d'envoi de paquets est supérieure à 75% mais inférieure au seuil saisi par l'utilisateur (ici 90 %).

- Les nœuds en orange sont les nœuds pour lesquels la réussite d'envoi de paquets est inférieure à 75%.
- Les nœuds en rouge sont les nœuds pour lesquels la réussite d'envoi de paquets est égale 0% ou qu'il n'y a pas de données relatives à ce nœud.

Voici un aperçu de l'affichage des voisins du nœud 51 pour une émission à 15dBm et avec un seuil de 90%.

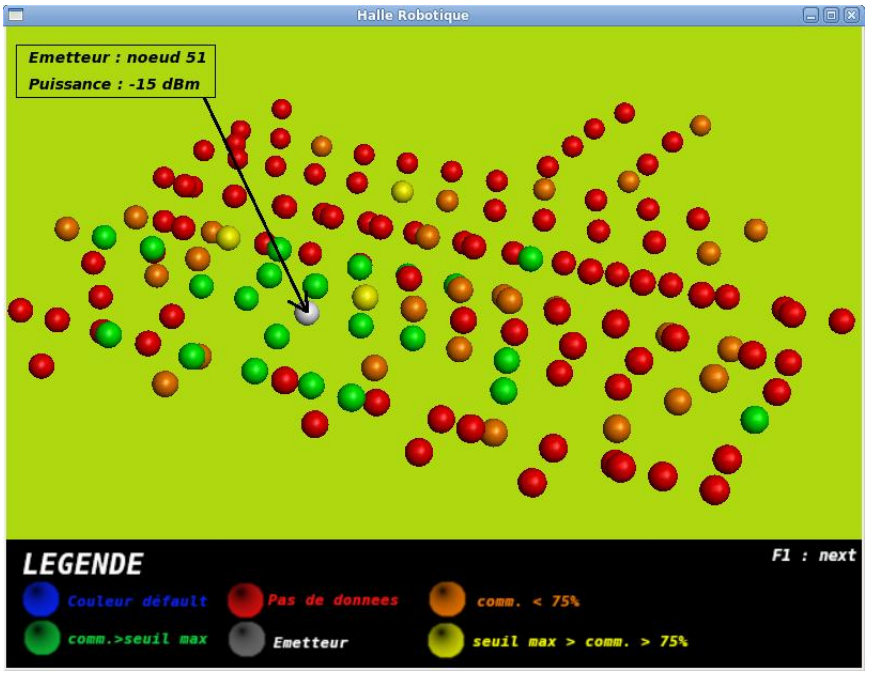


Figure 21 : Affichage des voisins du nœud 51 à -15dBm et avec un seuil de 90%

- **Création des courbes de la distribution du nombre de voisins pour la puissance souhaitée**

L'utilisateur commence par choisir la puissance d'émission de l'émetteur, il lui suffit ensuite de cliquer sur le bouton 'distrib. nb voisins'. Un histogramme est alors calculé et affiché, il représente la distribution du nombre de voisins des nœuds à la puissance sélectionnée.

Voici un aperçu des l'histogramme pour les puissances de +7 dBm puis de -15 dBm.

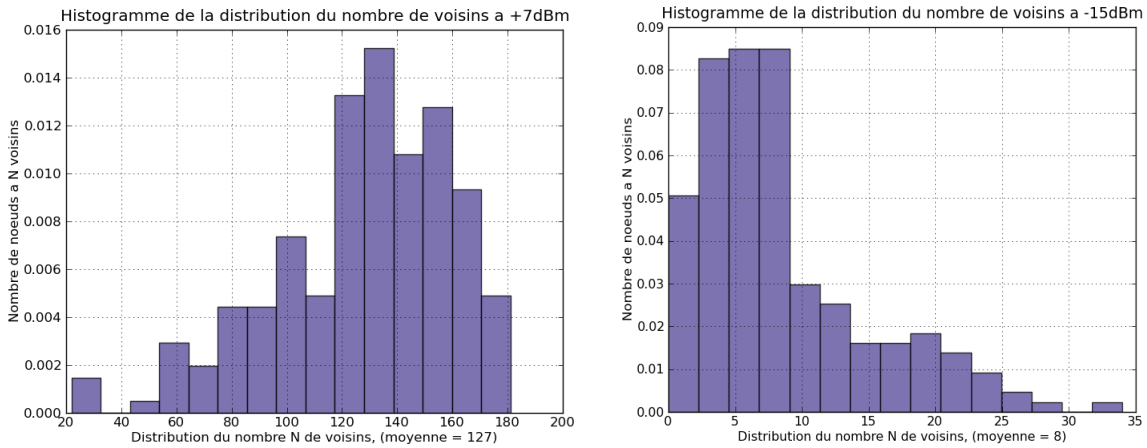


Figure 22 : Distribution du nombre de voisins à +7dBm et -15dBm

5.5.3.4 Analyse des données

Il est clair que l'affaiblissement et le taux de transmission de paquets sont deux grandeurs liées. Ici encore le taux de transmission diminue en fonction de la distance mais on ne peut pas dire que la distance soit un réel facteur discriminant sur l'ensemble des nœuds, en effet, de nombreux phénomènes augmentent l'affaiblissement entre les différents nœuds. Ils sont principalement dus au fait que l'on ne soit pas en extérieur et que de nombreuses pièces métalliques sont à proximité des nœuds.

J'ai lu sur le site d'Audio-Technica, un fabriquant de casques audio haut de gamme sans fil utilisant des fréquences UHF (Ultra high frequency : 300MHz -> 3GHz) et VHF (very high frequency : 30MHz -> 300MHz):

« Peu importe l'antenne utilisée, il ne faut absolument aucun obstacle entre l'antenne de l'émetteur et celle du récepteur. Les objets métalliques bloquent les signaux RF et causent des affaiblissement ainsi qu'une transmission bruyante »

5.5.4 Modélisation de la plateforme

5.5.4.1 Modélisation en 2D

Un fichier avec la localisation de la moitié des nœuds existait déjà à mon arrivée. (Environ 90 sur les 256). Je l'ai donc récupéré et formaté de telle sorte que l'on puisse récupérer facilement les coordonnées de chaque nœud.

J'ai ensuite créé une carte 2D représentant la salle en 2D. La carte à été réalisé en python à l'aide de la librairie graphique de base Tkinter.

Une telle modélisation regroupe de nombreux avantages mais aussi des inconvénients.

- ***Avantage:***

Grâce à celle-ci la visualisation des différents états des capteurs est bien meilleure, et on peut s'assurer que les mesures ne sont pas incohérentes.

On peut sélectionner directement grâce à la souris le capteur sur lequel on veut visualiser les données.

- ***Inconvénient:***

  La modélisation 2D à ses limites, en effet, les coordonnées des nœuds présentes dans le fichier sont toutes sur le même plan, mais en réalité il y a 4 ou 5 plans. Ainsi, si l'on compte rajouter les autres plans, la visualisation en sera altéré et deviendra un véritable « fouillis ».

Pour valider les premiers tests, la modélisation 2D a suffit mais il faut songer à une autre représentation pour la modélisation finale.

Voici un aperçu de la modélisation 2D de la plateforme SENSLAB sur un seul plan:

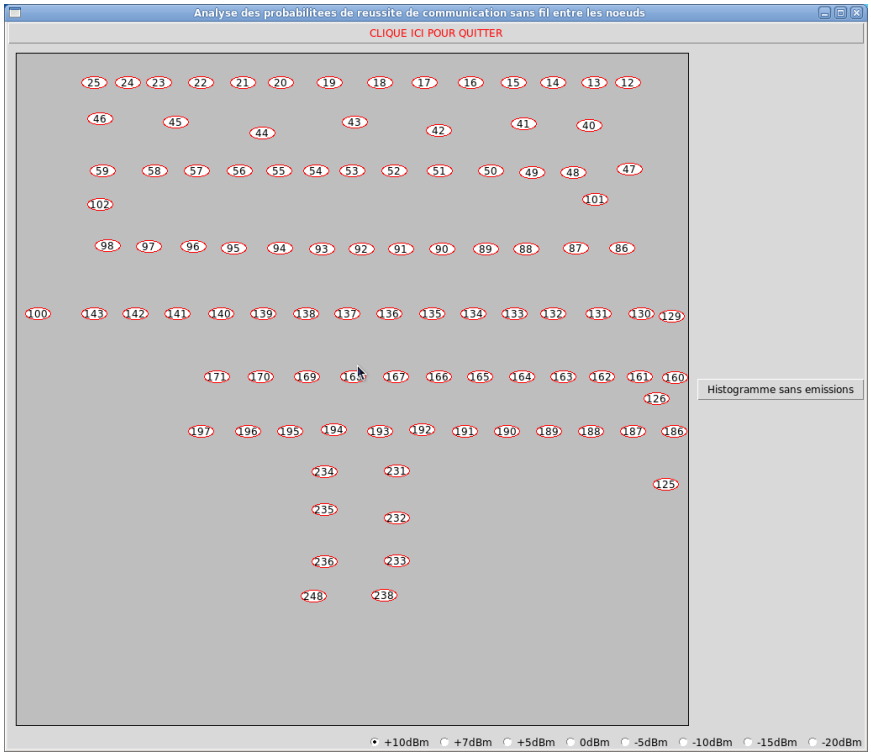


Figure 23 : Modélisation 2D de la plateforme SENSLAB

5.5.4.2 Modélisation en 3D

Vu les limites de la modélisation 2D, je me suis donc documenté sur les différentes librairies permettant de faire de la 3D en python. La plus répandue et la plus puissante étant pyOpenGL, je l'ai donc choisie (une version adaptée au python de la librairie très connue OpenGL).

Cette librairie a fait ses preuves, elle est utilisée dans la plupart des logiciels de DAO (Dessin assisté par ordinateur), CAO (Conception assistée par ordinateur)...

L'idée est donc de faire un rendu de la plateforme sur lequel l'utilisateur peut interagir (rotation, translation, zoom, sélection du nœud...)

Voici un aperçu de la modélisation 3D :

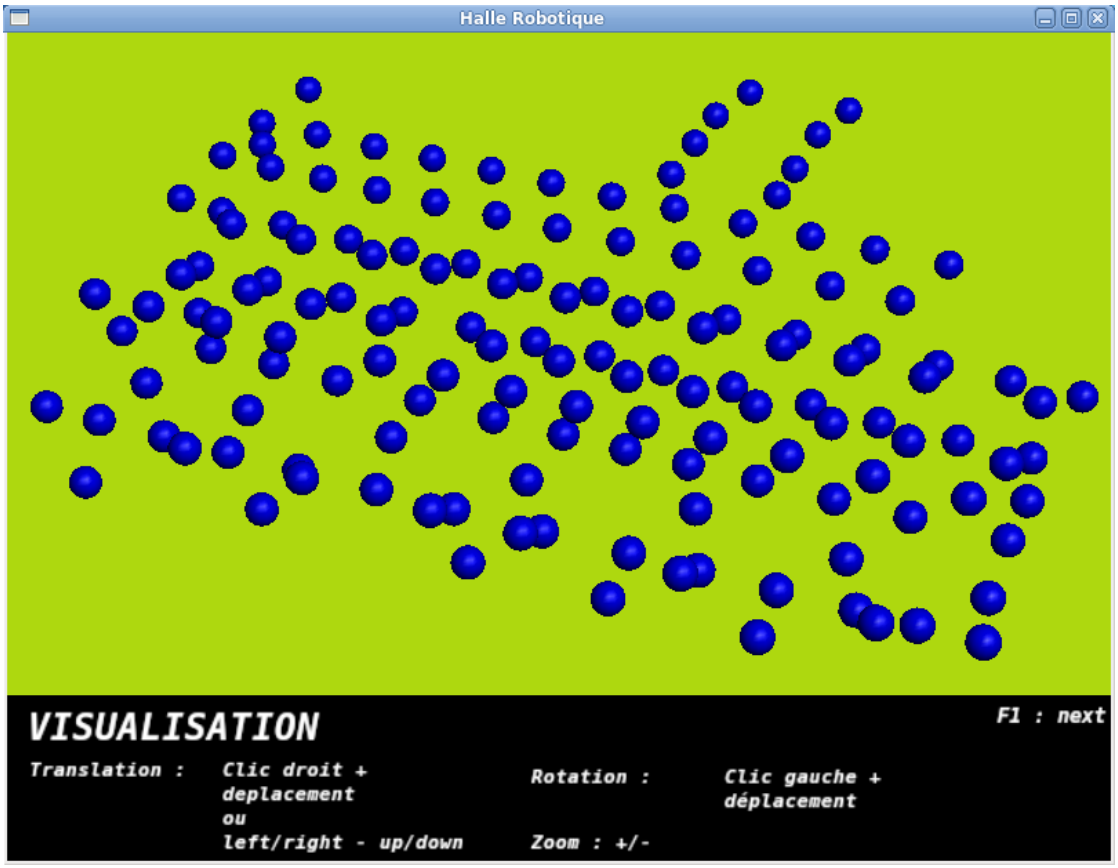


Figure 24 : Modélisation 3D de la plateforme SENSLAB

L'utilisateur peut interagir sur la modélisation. En effet, il effectuer des rotations, des zooms, des translations comme il le souhaite pour ainsi observer la plateforme sous l'angle qu'il désire. La sélection du nœud se fait grâce à une liste dans une autre fenêtre appelée « tableau de contrôle », le nœud sélectionné dans la liste change de couleur.

Certes la sélection du nœud est moins intuitive qu'avec le modèle 2D mais le rendu est bien meilleur.

5.5.5 Implémentation de l'algorithme de Dijkstra

L'algorithme de Dijkstra sert à résoudre le problème du plus court chemin. Il permet, par exemple, de déterminer le plus court chemin pour se rendre d'une ville à une autre connaissant le réseau routier d'une région. Il s'applique à un graphe connexe dont le poids lié aux arêtes est positif ou nul.

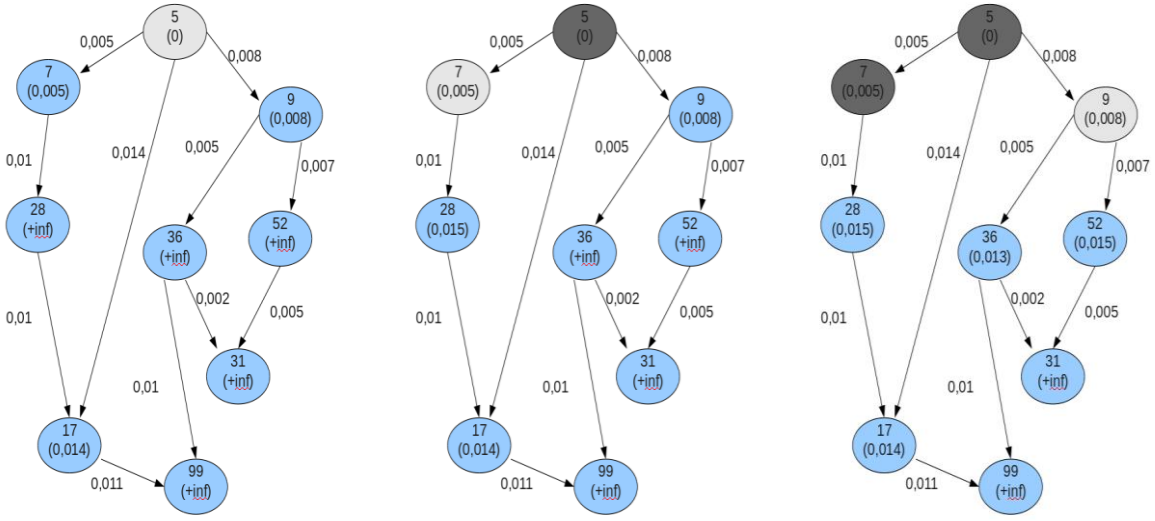
L'algorithme porte le nom de son inventeur, l'informaticien néerlandais Edsger Dijkstra et a été publié en 1959.

Ici, nous connaissons les voisins de chacun des nœuds à une puissance donnée. L'idée est donc de créer un graphe connexe[17] à partir de ces données et pouvoir ainsi calculer le meilleur chemin pour aller d'un nœud initial à un nœud final en maximisant la probabilité de réussite de communication.

Voici un exemple simplifié de la résolution du meilleur chemin (maximisation de la probabilité de réussite de communication) :

La valeur inscrite sur chaque arc est log10(1/Pr(communication)) avec 0 < Pr(communication) <= 1

Ainsi, en minimisant la somme des log10 on maximise aussi la probabilité de réussite de communication.



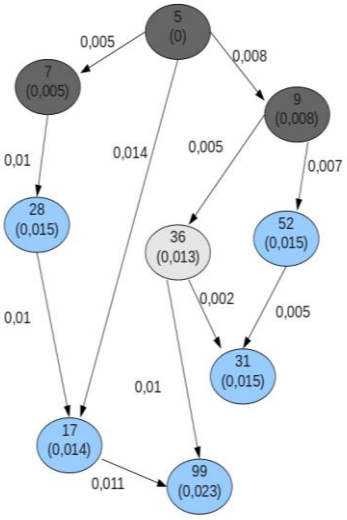
Étape 1 **Étape 2** **Étape 3**

Étape 1 : à partir du nœud '5', 3 nœuds sont accessibles, '7', '17', et '9' qui se voient donc affectées des poids respectifs de 0.005, 0.014, 0.008, tandis que les autres sont affectés d'un poids infini.

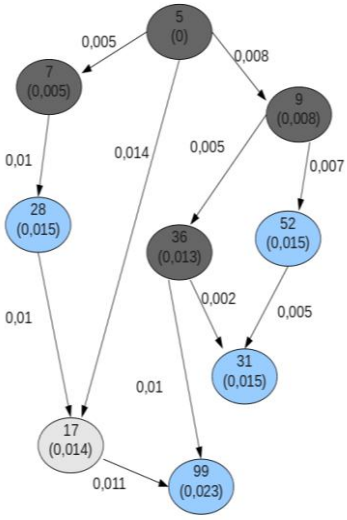
Étape 2 : le poids le plus petit est celui menant au nœud '7'B. Le passage par le nœud '7' ouvre la route au nœud '28' (0.005 + 0.01 = 0.015).

Étape 3 : Le poids le plus petit suivant est celui de l'arc menant au nœud '9'. Le passage par le nœud '9' ouvre une route vers les nœuds '36' et '52'.

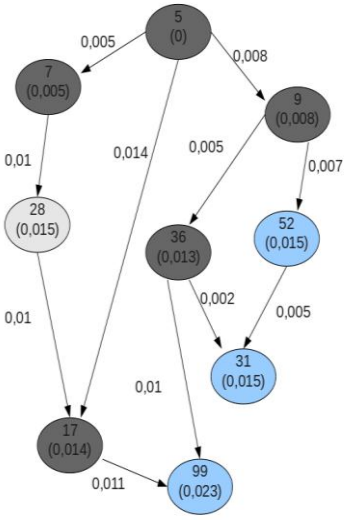
[17] Tous les sommets du graphe sont accessible (remarque si le graphe créé n'est pas connexe, ça ne perturbera pas le fonctionnement de l'algorithme, il n'y aura juste pas de chemin de trouvés pour les sommets non accessibles)



Étape 4



Étape 5

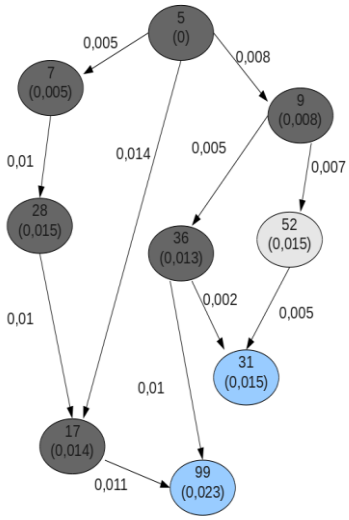


Étape 6

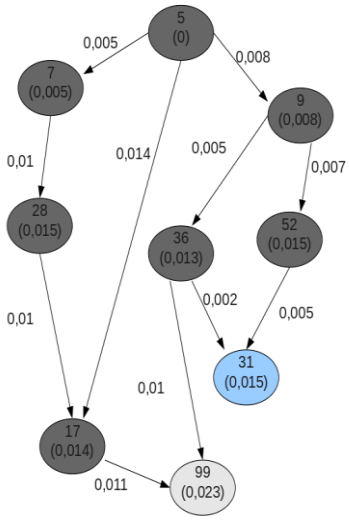
Étape 4 : Le poids le plus petit suivant est alors celui menant au nœud '36'. Le passage par le nœud '36' ouvre une route vers les nœuds '31' puis '99' avec un premier poids de 0.023 pourra-t-on le minimiser.

Étape 5 : Le poids le plus petit suivant est alors celui menant au nœud '17'. Le passage par le nœud '17' ouvre une route vers le nœud final, le nœud '99' mais le poids engendré n'est pas minimisé (0.025 > 0.023).

Étape 6 : Le poids le plus petit suivant est alors celui menant au nœud '28'. Le passage par le nœud '28' ouvre une route vers les nœuds '17' avec un poids de 0.025 or ce nœud est déjà marqué d'un poids de 0.014, on ne le marque donc pas de ce nouveau poids.



Étape 7



Étape 8

Étape 7 : Le poids le plus petit suivant est alors celui menant au nœud '52'. Le passage par le nœud '52' nous mène à une impasse.

Étape 8 : Le poids le plus petit suivante est alors celui menant au nœud '99'. Le passage par le nœud '99' est l'arrivée à destination. Le poids engendré par ce parcours est de 0.23. C'est le poids minimal que l'on peut trouver.

On connait ainsi le chemin qui minimise le poids des arcs (c'est à dire qui maximise la probabilité de réussite de communication) entre les nœuds '5' et '99'

J'ai développé cet algorithme en python et pour trouver un plus court chemin dans un graphe de 256 nœuds qui ont chacun en moyenne 131 voisins, le temps d'éxécution (création du graphe + calcul du plus court chemin + affichage du chemin en 3D) est de 100ms ce qui est très raisonnable

La performance de cet algorithme n'est plus à remettre en cause, en effet, l'algorithme de Dijkstra est utilisé dans plusieurs applications informatiques telles que les GPS. En outre, Google par exemple a pu introduire cet algorithme pour améliorer plusieurs fonctionnalités sur Google Maps ainsi que Google Earth

Voici un exemple de l'utilisation de l'algorithme de Dijkstra sur la plateforme SENSLAB, on cherche à trouver une route qui maximise la probabilité de réussite de communication entre les nœuds '4' et '231' à une puissance de -15dBm.

Pour cela, il suffit de sélectionner le nœud de départ puis le nœud d'arrivée dans la liste, la puissance désirée et ensuite d'appuyer sur le bouton 'Dijkstra'

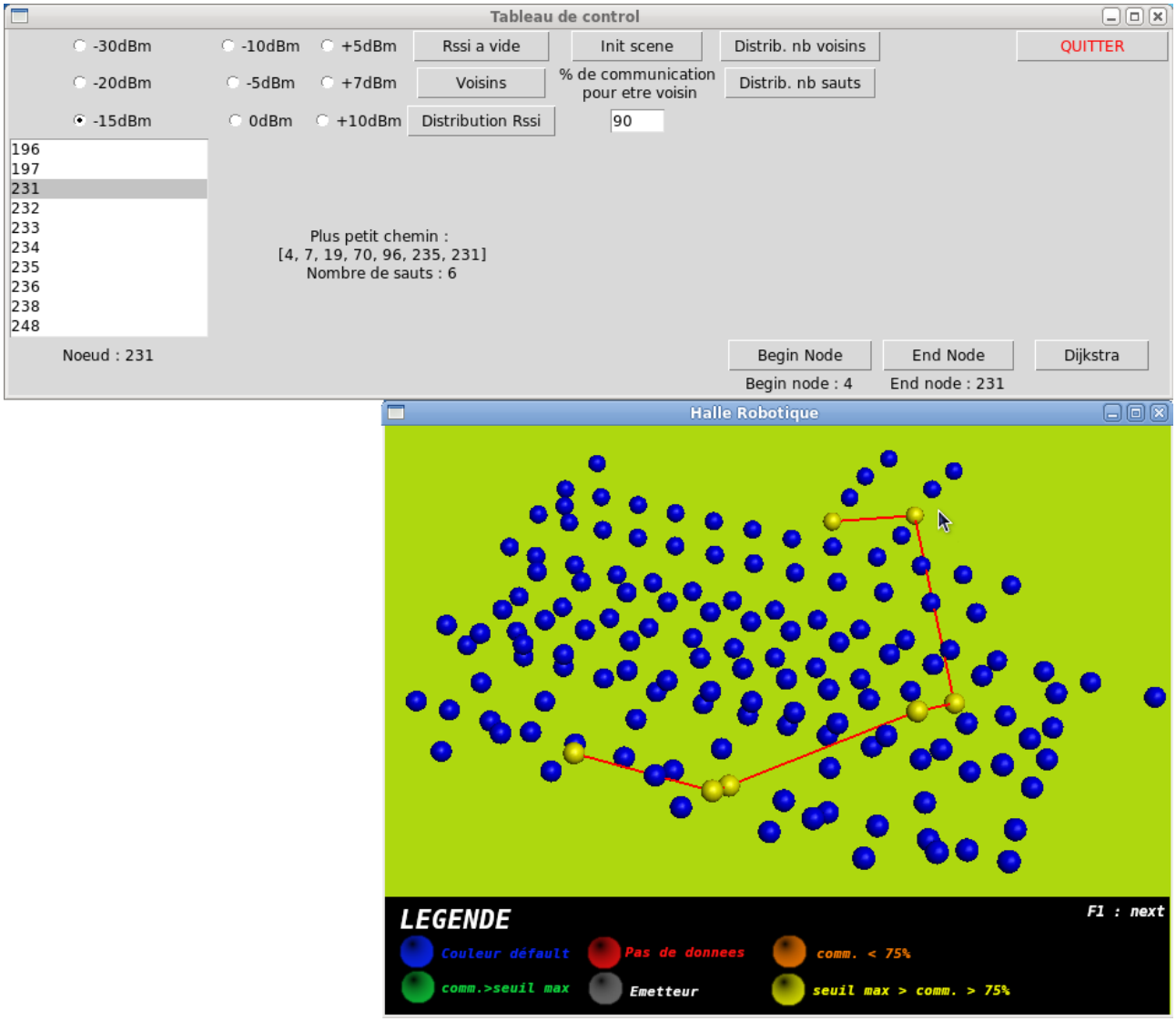


Figure 25 : Algorithme de Dijkstra entre les nœuds 4 et 231 à -15 dBm

5.6 LES DIFFÉRENTS PROBLÈMES RENCONTRÉS

5.6.1 Prise en main du langage objet Python

Python est un langage de programmation interprété multi-paradigme. Il favorise la programmation impérative structurée, et orientée objet. Il est doté d'un typage dynamique fort, d'une gestion automatique de la mémoire par ramasse-miettes et d'un système de gestion d'exceptions. Il est ainsi similaire à Perl, Ruby, Scheme, Smalltalk et Tcl.

Aucun de ces langages ne nous a été enseigné durant notre formation, il a donc fallut apprendre sur le tas. Fini les points-virgules, fini les pointeurs, fini les déclarations de types... La syntaxe est assez déroutante et au fur et à mesure on y prend gout. Par son très haut niveau d'abstraction, il

permet de réaliser des tâches assez complexes sans écrire trop de ligne de code contrairement au langage C.

En à peine deux semaines, je maîtrisais les bases de ce langage orienté objet.

5.6.2 Prise en main de OpenGL

OpenGL est une spécification qui définit une API multi-plateforme pour la conception d'applications générant des images 3D. Elle utilise en interne les représentations de la géométrie projective pour éviter toute situation faisant intervenir des infinis.

L'interface regroupe environ 250 fonctions différentes qui peuvent être utilisées pour afficher des scènes tridimensionnelles complexes à partir de simples primitives géométriques.

Cette bibliothèque est également utilisée dans l'industrie du jeu vidéo.

J'ai du me familiariser avec cette API pour pouvoir réaliser la modélisation 3D de la plateforme SENSLAB.

Très compliquée au premier abord, j'ai commencé par étudier différents exemples utilisant cette API. Et, de fil en aiguille, j'ai compris son fonctionnement.

En à peu près une semaine, j'avais les éléments de bases nécessaires à la modélisation en main, j'ai donc pu me rassurer quant à la faisabilité de cette modélisation.

5.7 LE BILAN DU PROJET

Comme dans tout projet, on rencontre des problèmes et des difficultés, et il faut à tout pris y faire face. Tout au long du stage, j'ai cherché à éviter ces aléas, ceci en commençant par réfléchir en écrivant sur papier au lieu de partir tête baissée sur la programmation pure et dure, et je me rappelle qu'au cours de l'année certains enseignants nous l'ont dit et redit. De plus, lorsque l'on mène un projet qui commence à prendre un certain volume, il vaut mieux être sûr de ce que l'on compte faire et de la manière dont on va le réaliser. En effet, les programmes n'ont pas la même taille que ceux réalisés en TP.
Le résultat à mes yeux est globalement satisfaisant. A l'heure ou j'écris le rapport, le développement n'est pas totalement fini, il reste encore quelques petites améliorations à faire, mais je suis en avance par rapport au diagramme de gant.

6 CONCLUSION

A l'issue de ce stage au sein de l'INRIA, je suis sorti grandi. En effet l'expérience professionnelle est primordiale aux cotés de la théorie que l'on nous enseigne tout au long de l'année.

Je trouve même dommage ne pas en avoir pris conscience plus tôt car j'aurais peut-être préféré faire un parcours par alternance. J'ai bien pris conscience des différences qu'il y a entre la théorie et la pratique, et c'est je pense primordial pour s'intégrer sans trop de soucis dans la vie active.

Ce stage m'a appris à travailler avec rigueur, à m'organiser et j'ai découvert de nouvelles méthodes de travail. Maintenant, la programmation objet qui ne m'était vraiment pas familière début avril, l'est beaucoup plus.

Ensuite, je pense que travailler dans ce genre d'institut me plairait bien, et si l'opportunité se présentait, je pense que je la saisirai. Pour ce qui est de mon avenir, je ne suis pas encore prêt à faire de la programmation pendant quarante ans. L'idée de me reconvertir dans les années qui viennent n'est donc pas à exclure.

Pour terminer, je tiens à remercier l'INRIA pour son accueil, la bonne ambiance de travail ou règne l'entraide et la bonne humeur. Je remercie aussi le service SED en général et plus particulièrement Christophe BRAILLON et Clément Burin des Roziers qui m'ont suivi tout au long de ce stage de fin d'étude.

Etudiant : GATE Jocelyn 3I-5 ISA

Entreprise : INRIA Grenoble – Rhône-Alpes
Adresse complète : 655 Avenue de l'Europe 38330 Montbonnot Saint Martin
Téléphone: 04 76 90 20 62

Responsable administratif : BRAILLON Christophe
Téléphone : 0476615377
Mél : Christophe.Braillon@inrialpes.fr

Maître de stage : BURIN DES ROZIERS Clement
Téléphone : 0456527111
Mél : clement.burin-des-roziers@inrialpes.fr

Tuteur enseignant : CAUFFET Gilles
Téléphone : 0476826365
Mél : gilles.cauffet@g2elab.inpg.fr

Titre : Réalisation de tests sur un réseau de capteurs déployé sur 4 sites en France : la plateforme Senslab.

Résumé :

Les progrès conjoints de la microélectronique, de la microtechnologie, des technologies de transmission sans fil et des applications logicielles ont permis de produire à coût raisonnable des micro-capteurs de faible volume, susceptibles de fonctionner en réseaux.

Un réseau de capteur est alors un ensemble d'objets communiquant en autonomie capables de remonter des informations physiques. Chaque objet du réseau est couramment appelé nœud et est composé de :

- Une unité de captage chargée de capter des grandeurs physiques (chaleur, humidité, vibrations, rayonnement...) et de les transformer en grandeurs numériques.
- Une unité de traitement informatique.
- Une unité de stockage de données.
- Un module de transmission sans fil.

Le principal enjeu d'un réseau de capteurs est la gestion de l'énergie, en effet, l'idéal serait d'avoir une faible consommation, ceci assurerait une grande autonomie et ainsi moins d'interventions au niveau humain (remplacement des batteries).

Durant ce stage, le travail qui m'a été proposé, est la réalisation d'une batterie de test sur un réseau de capteurs, la plateforme SENSLAB. Plus particulièrement, sur la partie radio, qui joue un rôle prépondérant dans la consommation d'énergie et qui nécessite d'être caractérisée. La caractérisation passera par les points suivants :

- Développement de la solution permettant la localisation des nœuds
- Mesure/analyse/affichage de l'affaiblissement du signal radio entre les nœuds
- Modélisation 2D de la plateforme
- Mesure/analyse/affichage du taux de transmission de paquets entre les nœuds
- Implémentation de l'algorithme de Dijkstra
- Modélisation 3D de la plateforme