UNIVERSITE
VINCENNES - SAINT-DENIS

Projet Tuteuré (1ère partie)

Implémentation d'un protocole de routage hiérarchique sous TinyOS

Encadré par: Mr. Touati Youcef

Réalisé par: Maknassi Asmaa
Jaiti Sana

Année universitaire 2015-2016

Sommaire

Remerciement

Nous tenons à remercier notre encadrant Mr TOUATI Youcef pour toute l'aide apportée tout au long de ce travail, son aide permanente et ses conseils éclairés, nous a permis de mener à bien ces recherches. Nous tenons également à remercier l'université Paris 8 de nous avoir accordé les salles pour pouvoir faire notre projet.

On remercie aussi le membre de jury qui fera l'honneur d'examiner et de juger notre travail.

Le but de notre projet, Problématique et Solution

- **Le but du projet :** Il consiste à étudier les protocoles de routage dans les réseaux de capteurs sans fils (RCSF) et d'implémenter le protocole LEACH (Low-Energy Adaptive Clustering Hierarchy).

- **Problématique :** L'énergie est un grand problème dans la conception d'un RCSF ,les applications exigent que la durée de vie du réseau soit de l'ordre de plusieurs mois ou même des années et les nœuds capteurs restent en permanence en activité, leur batteries peuvent fonctionner juste pour quelques jours ou pour quelques semaines.

- **Solution :** Implémenter un protocole de routage qui a pour objectif la minimisation de la consommation d'énergie des nœuds capteurs et étendre la durée de vie du réseau.

Introduction

Dans la vie courante, l'utilisation des capteurs sans fils est de plus en plus demandée pour la supervision et la sécurité. Les industries proposent alors des capteurs sans fils qui peuvent renseigner l'utilisateur sur plusieurs données. Ces capteurs peuvent aussi être reliés ensemble pour former un réseau sans fil se basant sur des protocoles pour se communiquer et proposant des programmes et des réseaux embarqués. Les capteurs fonctionnent donc à basse tension et ceci est géré par un système d'exploitation spécialisé : TinyOS.

Les capteurs sont des dispositifs de taille extrêmement réduite avec des ressources très limitées, autonomes, capable de traiter des informations et de les transmettre, via les ondes radio, à une autre entité (capteurs, unités de traitement...) sur une distance limitée à quelques mètres. Les RCSFs (Réseaux de capteurs sans fils) sont souvent considérés comme étant les successeurs des réseaux ad hoc (réseaux sans fil capables de s'organiser sans infrastructure définie préalablement). En effet, les RCSFs partagent avec les MANET (Mobile Ad hoc Networks) plusieurs propriétés telles que l'absence d'infrastructure et les communications sans fils. Mais l'une des différences clés entre les deux architectures est le domaine d'application. Contrairement aux réseaux MANET, qui n'ont pas pu connaître un vrai succès, les RCSF ont su attirer un nombre croissant d'industriels, vu leur réalisme et leur apport concret. Notre Rapport est composé de trois chapitres :

- Dans le premier chapitre nous présentons des Généralités.
- Le deuxième chapitre présente les différents protocoles de routage hiérarchique des réseaux de capteurs sans fils.
- Le troisième chapitre présente l'environnement de travail (TinyOS,NesC) et le protocole LEACH.

1. Introduction aux Réseaux de capteurs sans fils

1.1 Qu'est ce qu'un réseau de capteur sans fil ?

Les réseaux de capteurs sans fil (Wireless Sensor Network - WSN) sont un type particulier de réseau Ad-hoc (réseaux sans fil capables de s'organiser sans infrastructure définie préalablement), dans lesquels les nœuds sont des "capteurs intelligents". Ils se composent généralement d'un grand nombre de capteurs communicants entre eux via des liens radio et capables de récolter et de transmettre des données environnementales d'une manière autonome. Dans ce type de réseau, les capteurs échangent des informations par exemple sur l'environnement pour construire une vue globale de la région contrôlée, qui est rendue accessible à l'utilisateur externe par un ou plusieurs nœuds.

1.2 Architecture des réseaux de capteurs sans fils

Un RCSF est composé d'un ensemble de nœuds capteurs. Ces nœuds capteurs sont habituellement dispersés dans une zone de capture organisé en champs "sensor fields" . Chacun de ces nœuds a la capacité de collecter des données et de les transférer au nœud passerelle (puits) par l'intermédiaire d'une architecture multi-sauts. Le puits transmet ensuite ces données par Internet ou par satellite à l'ordinateur central "Gestionnaire de tâches " (voir 1.1).

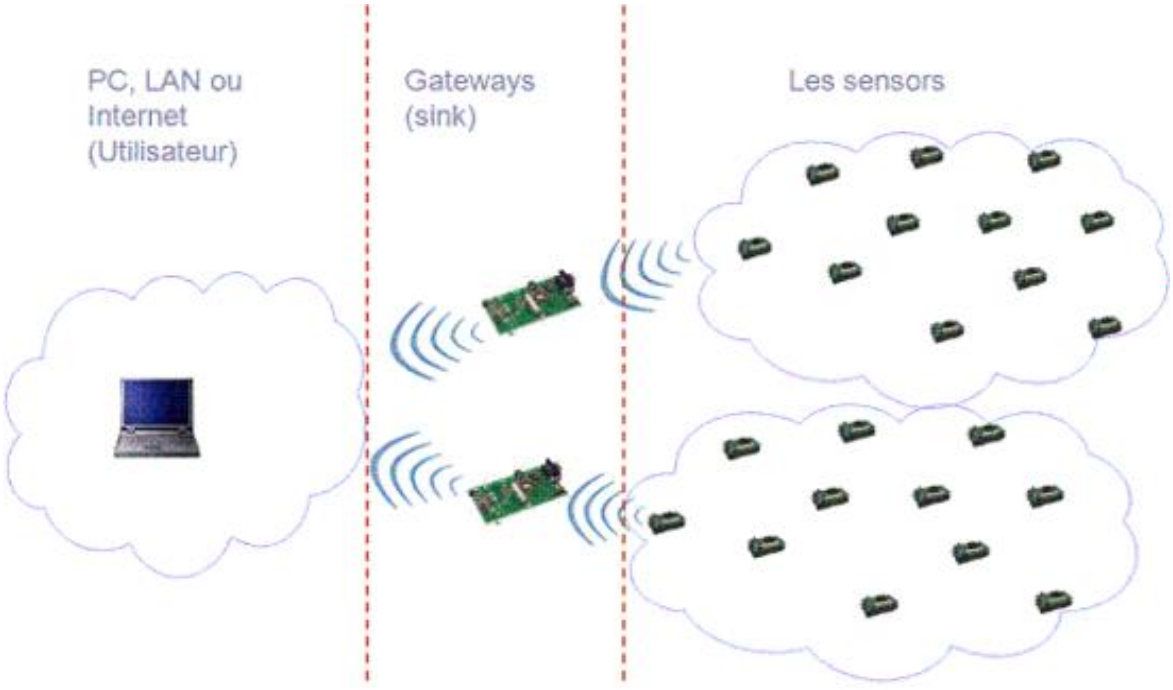


Figure 1.1 Architecture des réseaux de capteurs

1.3 Domaines d'application des RCSFs

1.3.1 Domaine médical

Les réseaux de capteurs sont largement répandus dans le domaine médical. Cette classe inclut des applications comme : fournir une interface d'aide pour les handicapés, collecter des informations physiologiques humaines de meilleure qualité, facilitant ainsi le diagnostique de certaines maladies grâce à des micro-capteurs qui pourront être ingérés ou implantés sous la peau, surveiller en permanence les malades et les médecins à l'intérieur de l'hôpital.

1.3.2 Domaine militaire

Le déploiement rapide, le coût réduit, l'auto organisation et la tolérance aux pannes des réseaux de capteurs sont des caractéristiques qui font de ce type de réseaux un outil appréciable dans un tel domaine. Actuellement, les RCSF peuvent être une partie intégrante dans le commandement, le contrôle, la communication, la surveillance. . .

1.3.3 Domaine architectural

La détection des altérations dans la structure d'un bâtiment, suite à un séisme ou au vieillissement.

1.3.4 Domaine environnemental

Dans ce domaine, les capteurs peuvent êtres exploités pour détecter les catastrophes naturelles (feux de forêts, tremblements de terre, etc.), détecter des produits toxiques (gaz, produits chimiques, pétrole, etc.) dans des sites industriels tels que les centrales nucléaires et les pétrolières.

1.4 Conclusion

Dans ce chapitre on a commencé par une introduction sur les généralités sur les protocoles de routages dans les réseaux de capteurs sans fils et on a décrit aussi le fonctionnement, l'architecture et les domaines d'application des réseaux de capteurs sans fils. Dans le prochain chapitre nous allons définir les différents protocoles de routage hiérarchique utilisés et on va finir le chapitre par une étude comparatives entre quelques protocoles de routage dans les réseaux de capteurs.

2. Les protocoles de routage dans les réseaux de capteurs

2.1 Définition des protocoles de routage dans les réseaux de capteurs

Le routage est une méthode d'acheminement des informations vers une destination donnée dans un réseau de connexion. Il est fondamental dans les réseaux de capteurs sans fils, vu qu'il n'existe pas d'infrastructure qui gère les informations échangées entre les différents nœuds du réseau. En effet, c'est à chaque nœud du réseau de jouer le rôle d'un routeur. Comme nous l'avons déjà vu, l'architecture des réseaux Ad-hoc est caractérisée par l'absence d'infrastructure fixe préexistante, à l'inverse des réseaux de télécommunication classiques. Un réseau Ad-hoc doit s'organiser automatiquement de façon à être déployé rapidement et pouvoir s'adapter aux conditions du trafic et aux différents mouvements pouvant intervenir au sein des nœuds mobiles.

2.2 Contraintes de conception des protocoles de routage pour RCSF

- Offrir un support pour pouvoir effectuer des communications multi-sauts fiables.
- Assurer un routage optimal si possible.
- Offrir une bonne qualité concernant les temps de latence.
- Auto-organiser le réseau.

2.3 Les différents protocoles de routage dans les réseaux de capteurs

2.3.1 LEACH

LEACH (Low-Energy Adaptive Clustering Hierarchy) est un protocole qui choisit aléatoirement les nœuds cluster-Head et attribue ce rôle aux différents nœuds selon la politique de gestion Round-Robin (c'est-à-dire tourniquet) pour garantir une dissipation équitable d'énergie entre les nœuds. Dans le but de réduire la quantité d'informations transmises à la station de base, les cluster-Head agrègent les données capturées par les nœuds membres qui appartiennent à leur propre cluster, et envoient un paquet agrégé à la station de base (voir 2.1).

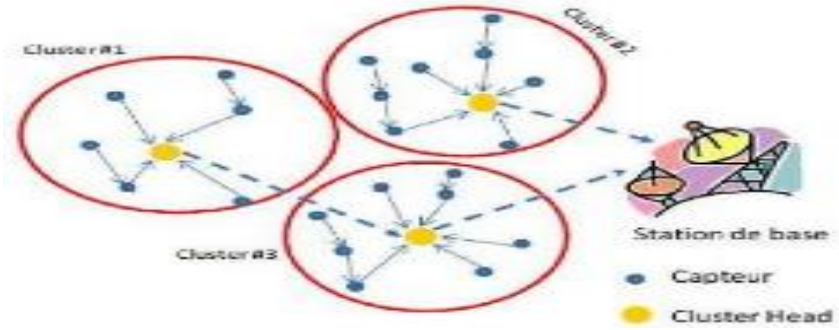


Figure 2.1 Le Clustering dans un RCSF

LEACH est exécuté en deux phases : la phase « set-up » et la phase « steady-state » suivant la figure 2.2. Dans la première phase, les clusters Head sont sélectionnés et les clusters sont formés, et dans la seconde phase, le transfert de données vers la station de base aura lieu. Durant la première phase, le processus d'élection des clusters Head est déclenché pour choisir le futur cluster Head. Ainsi, une fraction prédéterminée de nœuds s'élisent comme cluster Head selon le schéma d'exécution suivant : durant une période T, un nœud n choisit un nombre aléatoire nb dont la valeur est comprise entre 0 et 1 (0 < nb < 1). Si nb est inférieure à une valeur seuil alors le nœud n deviendra cluster Head durant la période courante, sinon le nœud n devrait rejoindre le cluster Head le plus proche dans son voisinage.

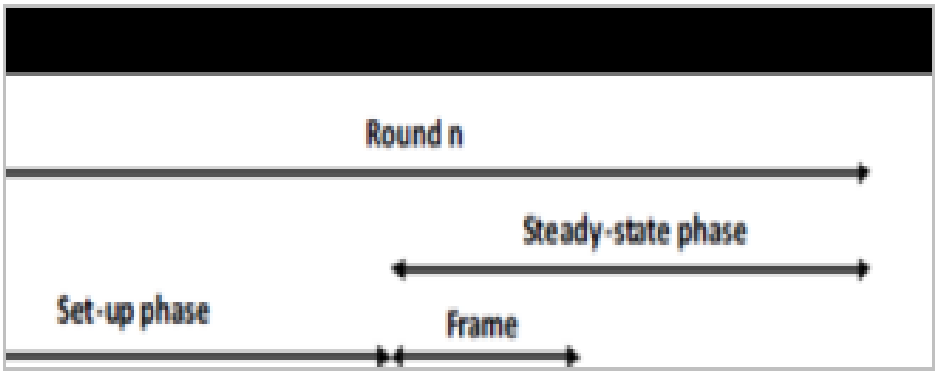


Figure 2.2 Illustration des deux fonctions de LEACH

2.3.2 PEGASIS

Power-Efficient Gathering in Sensor Information Systems (PEGASIS) est une version améliorée du protocole LEACH. PEGASIS forme des chaînes plutôt que des clusters de nœuds de capteurs afin que chaque nœud transmette et reçoive uniquement des données d'un voisin. Un seul nœud est sélectionné à partir de cette chaîne pour transmettre à la station de base. L'idée de PEGASIS est qu'il utilise tous les nœuds pour transmettre ou recevoir des données avec ses plus proches voisins. Il déplace les données reçues de nœud à nœud, puis les données seront agrégées jusqu'à ce qu'elles atteignent tous la station de base. Donc, chaque nœud du réseau est tour à tour un chef de file de la chaîne, ainsi que responsable pour transmettre l'ensemble des données recueillies et fusionnées par la chaîne de nœuds au niveau de la station de base.

Une variante de PEGASIS appelée Hierarchical PEGASIS a été conçue afin d'améliorer PEGASIS.

2.3.3 TEEN

TEEN est conçu pour être sensible à des changements soudains des attributs tels que la température. La réactivité est importante pour les applications critiques dont le réseau fonctionne dans un mode réactif. L'architecture du réseau de capteurs est basée sur un groupement hiérarchique où les nœuds forment des clusters et ce processus va se répéter jusqu'à ce que la station de base soit atteinte.

Au début, les nœuds écoutent le médium continuellement et lorsque la valeur captée du paramètre contrôlé dépasse le seuil Hard, le nœud transmet l'information. La valeur captée est stockée dans une variable interne appelée SV. Puis, les nœuds ne transmettront des données que si la valeur courante du paramètre contrôlé est supérieure au seuil hard HT ou diffère du SV d'une quantité égale ou plus grande que la valeur du seuil Soft ST.

2.3.4 APTEEN

APTEEN est une extension du TEEN basée sur la capture périodique des données et réaction aux événements temps-réel. Quand la station de base forme les clusters, les cluster-Head diffusent les attributs, les seuils et le plan de transmission à tous les nœuds et effectuent également l'agrégation des données afin d'économiser de l'énergie.

2.3.5 HEED

Younes et Fahmy ont proposé un algorithme de clustering distribué appelé HEED pour les réseaux de capteurs. HEED ne fait aucune restriction sur la distribution et la densité des nœuds. Il ne dépend pas de la topologie du réseau ni de sa taille mais il suppose que les capteurs ont la possibilité de modifier leur puissance de transmission. HEED sélectionne les cluster-heads selon un critère hybride regroupant l'énergie restante des nœuds et un second paramètre tel que le degré des nœuds. Il vise à réaliser une distribution uniforme des clusters Head dans le réseau et à générer des clusters équilibrés en taille. Son problème réside dans la détermination du nombre optimal des clusters. De plus, HEED ne précise pas de protocole particulier à utiliser pour la communication entre les clusters Head et le sink. A l'intérieur du cluster, le problème ne se pose pas car la communication entre les membres du cluster et le cluster head est directe (à un saut). D'autre part, avec HEED, la topologie en clusters ne réalise pas de consommation minimale d'énergie dans les communications intra-cluster et les clusters générés ne sont pas équilibrés en taille.

2.4 Conclusion

Dans ce chapitre on a décrit les différents protocoles de routage hiérarchique dans les réseaux de capteurs sans fils. Dans le prochain chapitre nous allons définir le système d'exploitation TinyOS ainsi que le langage de programmation utilisé dans les réseaux de capteurs NesC et nous allons procéder à l'étape d'implémentation du protocole de routage LEACH.

3.Environnement de travail&implémentation du protocole

3.1 Le système d'exploitation TinyOS

TinyOS est un système d'exploitation intégré, modulaire, destiné aux réseaux de capteurs miniatures. Cette plate-forme logicielle ouverte et une série d'outils développés par l'Université de Berkeley est enrichie par une multitude d'utilisateurs. En effet, TinyOS est le plus répandu des systèmes d'exploitation pour les réseaux de capteurs sans fils. Il est utilisé dans les plus grands projets de recherches sur le sujet (plus de 10.000 téléchargements de la nouvelle version). Un grand nombre de ces groupes de recherches ou entreprises participent activement au développement de cet OS (Operating System) en fournissant de nouveaux modules, de nouvelles applications,... Cet OS est capable d'intégrer très rapidement les innovations en relation avec l'avancement des applications et des réseaux eux même tout en minimisant la taille du code source en raison des problèmes inhérents de mémoire dans les réseaux de capteurs. La librairie TinyOS comprend les protocoles réseaux, les services de distribution, les drivers pour capteurs et les outils d'acquisition de données. TinyOS est en grande partie écrit en C mais on peut très facilement créer des applications personnalisées en langages C, NesC, et Java.

3.2 Le langage de programmation de TinyOS: NesC

Une application NesC consiste en un ou plusieurs composants assemblés pour former un exécutable. Un composant, du point de vue programmation, est composé de plusieurs sections, il offre et utilise des interfaces qui sont son unique point d'accès. Il existe deux types de composants : modules et configurations.

- **Modules :** sont des composants qui définissent le code de l'application en implémentant une ou plusieurs interfaces.
- **Configurations :** sont des composants qui assemblent les composants de l'application, en reliant les interfaces utilisées par des composants aux interfaces offertes par d'autres composants. Les éléments connectés doivent être compatibles : "Interface" à "interface", "command" à "command", "event" à "event" (Voir 3.1).

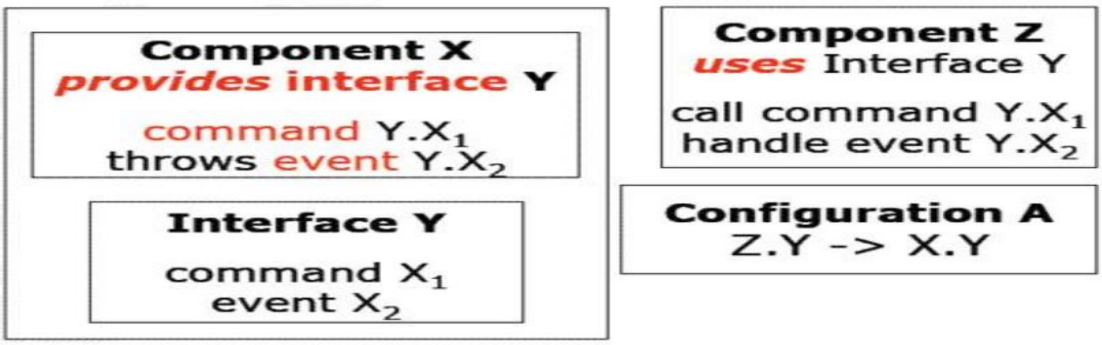


Figure 3.1 Application TinyOS

3.3 Interface

- Une interface définie les interactions entre deux composants.
- Elle définie un ensemble de fonctions appelées commandes (qui doivent êtres implémentées par le composant qui offre cette interface) et un autre ensemble de fonctions appelées événements (qui doivent êtres implémentées par le composant utilisant cette interface). Les interfaces sont utilisées pour les opérations qui décrivent des interactions bidirectionnelles.
- Pour qu'un composant puisse appeler les commandes d'une interface, il doit implémenter les événements qui y sont définis. Un même composant pourrait offrir et utiliser plusieurs interfaces différentes ou plusieurs instances d'une même interface.

3.4 Modèle de concurrence

TinyOS exécute seulement un programme consistant d'un ensemble de composants requis par l'application. Il existe deux chemins d'exécution les tâches et les handlers.

3.4.1 Les tâches

Sont des fonctions dont l'exécution est différée. Une fois lancée, une tâche s'exécute jusqu'à sa fin et ne peut pas interrompre une autre tâche.

- Une tâche est un élément de contrôle indépendant défini par une fonction retournant void et sans arguments : task void myTask() ...

3.4.2 Les handlers d'événements matériels

Sont exécutés en réponse à des interruptions matérielles (pression d'un bouton, changement d'état d'une entrée,...) et ils permettent de faire le lien entre ces dernières et les couches logicielles que constituent les tâches. Ils sont prioritaires par rapport aux tâches et peuvent interrompre la tâche en cours d'exécution ou un autre "handler".

3.5 L'implémentation du protocole LEACH

Comme cité précédemment ,LEACH est un protocole de routage hiérarchique, employant un procédé de clustering qui divise le réseau en deux niveaux : les cluster-heads et les nœuds membres. Le protocole se déroule en rounds. Chaque round se compose de deux phases : construction et communication (voir annexe 2).

3.5.1 Débogage du programme

Il existe de nombreuses façons de débugger un programme et notamment pour NesC, on pourra utiliser Gdb qui est un debugger s'exécutant en mode console ou Eclipse plugin. Eclipse IDE est conçu pour appuyer sur Gdb pour examiner l'état des programmes en cours d'exécution .

Gdb gère quatre missions principales :

- Lancement de programmes précisant les paramètres susceptibles d'influer sur son comportement .
- Arrêter le programme sur les conditions prédéfinies (points d'arrêt, points d'observation) .
- Examen de l'état du programme quand il est arrêté (Stack trace, examiner les variables).
- Changer l'état du programme (pas à pas unique dans le code) .

3.6 Problèmes rencontrés

- L'installation de TinyOS nous a pris beaucoup de temps (voir annexe 1).
- Ecriture du premier essai du code LEACH (voir annexe 2).
- La plupart de la documentation sur les réseaux de capteurs sans fils est en anglais.

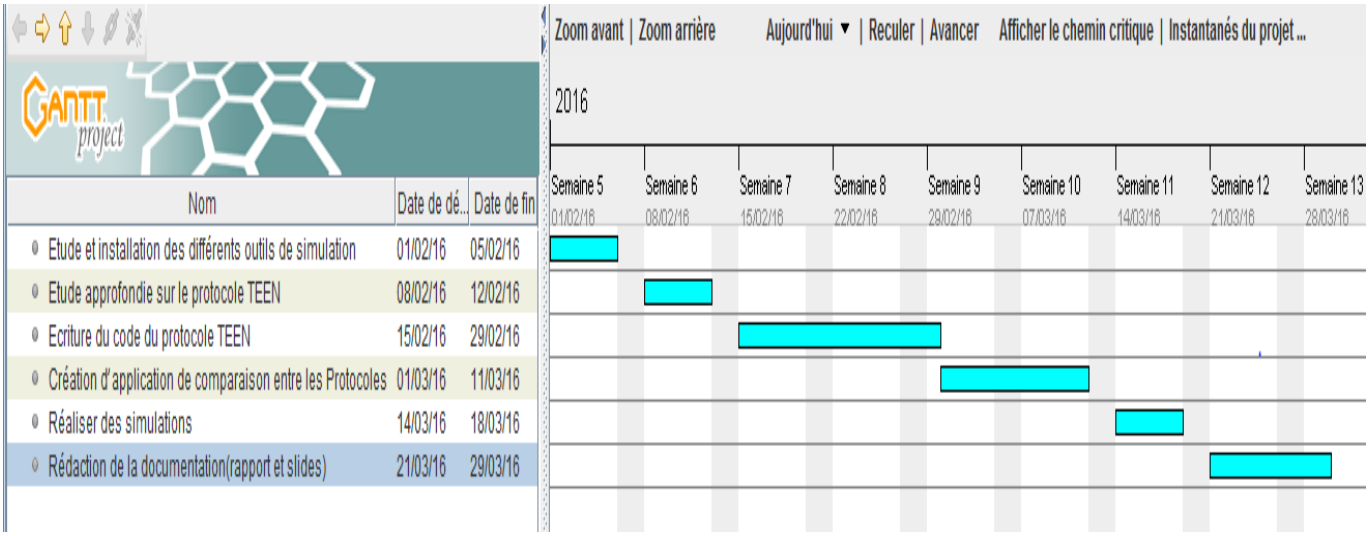
Conclusion Générale

Les réseaux de capteurs sans fils nous intéresse déjà bien avant de choisir ce sujet et maintenant notre passion ne cesse d'augmenter ,la première partie de ce projet nous a permis d'apprendre plus de choses et d'acquérir de nouvelles connaissances théoriques et pratiques dans les réseaux de capteurs sans fils et de travailler en équipe puisque on devait partager les tâches pour avancer le plus vite possible.

La compréhension du fonctionnement des modèles TinyOS ainsi que le développement sous NesC, nous a permis de comprendre de manière plus efficace la mise en œuvre d'un RCSF.

Calendrier prévisionnel de la deuxième partie

Ce planning a été réalisé avec le logiciel Gantt Project qui est un outil permettant de gérer les projets sur les modèles des diagrammes de Gantt. Cette application permet de décomposer le projet en arborescence et d'assigner des ressources à chacune des tâches prévues au planning.



Références

- UC Berkeley, “Smart dust : Autonomous sensing and communication in a cubic millimeter.” robotics.eecs.berkeley.edu/ pister/SmartDust.

- P. Levis, N. Lee, M. Welsh, and D. Culler, “Tossim : accurate and scalable simulation of entire tinyos applications,” in SenSys ’03 : Proceedings of the 1st international conference on Embedded networked sensor systems. New York, NY, USA : ACM Press, pp. 126–137.

- TinyOS Team, “Tinyos.” www.tinyos.net

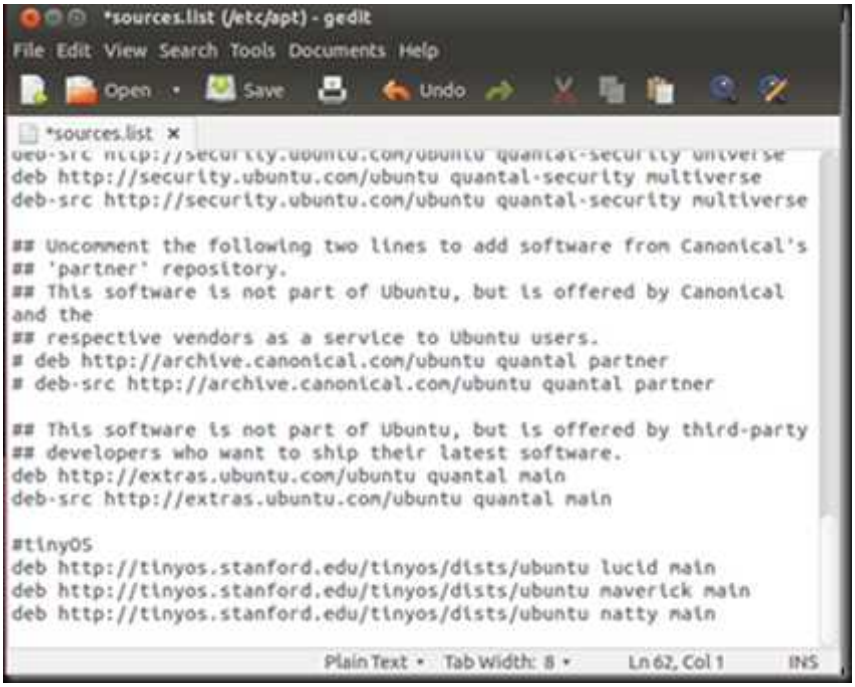
Annexe 1: Installation de TinyOS-2.1.2

Installation des paquets Ubuntu

1-Avant d'installer Tinos sur ubuntu 14.04 il faut inclure les adresses suivantes dans le fichier source référentiel **sources.list**.Cela en éditant le fichier **/ etc / apt /sources.list** comme suit:

```
sudo gedit / etc / apt /sources.list
```

Ensuite, vous allez obtenir une boîte de dialogue comme ci-dessous comme celle présentée ci-dessous:



2-Inclure les lignes suivantes à la fin du script :

```
#tinyOS
deb http://tinyos.stanford.edu/tinyos/dists/ubuntu lucid main
deb http://tinyos.stanford.edu/tinyos/dists/ubuntu maverick main
deb http://tinyos.stanford.edu/tinyos/dists/ubuntu natty main
```

3-Enregistrer et fermer l'application "gedit".

4-Ouvrir la **console de terminal** et exécutez les commandes suivantes:

```
sudo apt-get update
sudo apt-get install tinyos
```

Ensuite, vous aurez à une liste des versions de Tinos disponibles tel que présenté ci-dessous:

5-Exécuter les commandes suivantes pour installer Tinos 2.1.2. Lors de cette étape peut-être vous serez invité à entrer le mot de passe d'administration.

```
sudo apt-get install tinyos-2.1.2
sudo apt-get install subversion
```

6- Exécuter les commandes suivantes, ceci permettra l'installation des derniers outils de MSPGCC, Cela va ajouter la dernière Tinos ® code, qui comprennent les nouveaux pilotes de puces msp430x afin d'être en mesure de compiler les iris des plateformes comme suit:

```
sudo svn checkout http://tinyos-main.googlecode.com/svn/trunk/ tinyos-main-read-only
sudo cp -R /opt/tinyos-main-read-only /opt/tinyos-2.x
```

7-Maintenant, utilisez **la console du terminal** et exécutez la commande suivante, allé à l'intérieur de l'installation de TinyOS, comme suit:

```
cd /opt/tinyos-2.1.2
```

8-Maintenant, voir la liste des fichiers en utilisant la commande **ls** et vérifier si il ya un fichier nommé ' **tinyos.sh** '. Si ce n'est pas le cas il faut le créer. Ceci peut être accompli en utilisant gedit, il faut s'assurer que le contenu de ' **tinyos.sh**'correspond avec le contenu ci-dessous:

```
#! /usr/bin/env bash
# Here we setup the environment
# variables needed by the tinyos
# make system
echo "Setting up for TinyOS 2.1.2 Repository Version"
export TOSROOT=
export TOSDIR=
export MAKERULES=
TOSROOT="/opt/tinyos-2.1.2"
TOSDIR="$TOSROOT/tos"
CLASSPATH=$CLASSPATH:$TOSROOT/support/sdk/java:.:$TOSROOT/support/sdk/java/t
inyos.jar
MAKERULES="$TOSROOT/support/make/Makerules"
export TOSROOT
export TOSDIR
export CLASSPATH
export MAKERULES
```

9-Enregistrez le fichier et quitter

10-Maintenant, utilisez **la console du terminal** et exécutez la commande suivante, allé à l'intérieur de l'installation de TinyOS, comme suit:

```
cd /opt/tinyos-2.x
```

11-Maintenant, la liste des fichiers en utilisant la commande **ls** et vérifier si il ya un fichier nommé ' **tinyos.sh** '. Si ce n'est pas le cas il faut le créer. Ceci peut être accompli en utilisant gedit, il faut s'assurer que le contenu de ' **tinyos.sh**'correspond avec le contenu ci-dessous:

```
#! / Usr / bin / env bash
# Ici nous configurer l'environnement
# variables requises par les TinyOS
Système d'appoint d'#
echo "Configuration pour TinyOS 2.x référentiel Version"
l'exportation TOSROOT =
l'exportation TOSDIR =
l'exportation MAKERULES =
TOSROOT = "/ opt / TinyOS-2.x"
TOSDIR = "$ TOSROOT / tos"
CLASSPATH = $ CLASSPATH: $ TOSROOT / support / sdk / java:.: $ TOSROOT / support / sdk / java /
tinyos.jar
MAKERULES = "$ TOSROOT / support / faire / Makerules"
l'exportation TOSROOT
l'exportation TOSDIR
export CLASSPATH
MAKERULES d'exportation
```

12-Enregistrez le fichier et quitter

13- Maintenant, utilisez **la console du terminal** et exécutez les commandes suivantes

```
sudo cp /opt/tinyos-2.1.2/tinyos.sh /opt/tinyos-2.x/tinyos.sh
sudo chmod +x /opt/tinyos-2.x/tinyos.sh
sudo gedit /opt/tinyos-2.x/tinyos.sh
```

14-Afin d'utiliser ce dossier comme chemin de TinyOS ROOT, ouvrir la **console de terminal** et exécuter la commande suivante:

```
sudo gedit ~/.bashrc
```

15-Ajouter les lignes suivantes à la fin du document qui s'ouvre:

```
#Sourcing Le script de configuration variable TinyOS de l'environnement
la source /opt/tinyos-2.1.2/ tinyos.sh
#Sourcing Le script de configuration variable TinyOS de l'environnement
la source /opt/tinyos-2.x/ tinyos.sh
```

17 Enregistrez le fichier et fermez l'éditeur. Fermez la **console de terminal** et redémarrez le PC.

Annexe 2: Ecriture d'un premier essai d'une partie du fichier module

```
module MHLeachPSM
{
        provides
        {
                interface StdControl;

        }
        uses
{
interface ReceiveMsg as LEACH_ReceiveMsg;
interface SendMsg as LEACH_SendMsg;
interface ReceiveMsg as ANNONCE_ReceiveMsg;  //utiliser par les CH pour annoncer  aux nœuds qu'ils sont les chefs
interface SendMsg as ANNONCE_SendMsg; //réception des nœuds membre des nouveau CH
interface ReceiveMsg as ORGANISATION_ReceiveMsg;  //réception des CH les msg de ses membres
interface SendMsg as ORGANISATION_SendMsg; //envoie des nœuds aux CH auxquels ils compte appartenir
```