Université Abou Bekr Belkaid
Tlemcen Algérie
جامعة أبي بكر بلقايد
تلمسان الجزائر

République Algérienne Démocratique et Populaire

Université Abou Bakr Belkaid– Tlemcen

Faculté des Sciences

Département d'Informatique

Mémoire de fin d'études

pour l'obtention du diplôme de Master en Informatique

Option: *Réseaux et Systèmes Distribués (R.S.D)*

Thème

L'apport de l'agrégation de données dans les réseaux de capteurs sans fil

Réalisé par :

- **BENMAHDI Meriem Bouchra**
- **DIB Siham**

Présenté le 02 Juillet 2013 devant le jury composé de MM.

- *Mme DIDI Fadoua* *(Président)*
- *Mr LEHSAINI Mohamed* *(Encadreur)*
- *Mr BENMAMMAR Badr* *(Examinateur)*
- *Mr BELHOUCINE Amine* *(Examinateur)*

Année universitaire : 2012-2013

Remerciements

Grâce à Dieu vers lequel vont toutes les louanges, ce travail s'est accompli.

Grâce à Dieu, nous avons l'honneur d'inscrire ici un immense remerciement à nos parents dans ces moments importants.

Ces quelques lignes ne pourront jamais exprimer la reconnaissance que nous éprouvons envers tous ceux qui, de près ou de loin, ont contribué par leurs conseils, leurs encouragements ou leurs amitiés à l'aboutissement de ce travail.

Nous remercions vivement notre encadreur Monsieur LEHSAINI pour avoir accepté de nous encadrer, tout particulièrement pour l'excellence de son accompagnement ainsi que pour la confiance qu'il nous a accordé.

Nous lui sommes reconnaissantes pour sa disponibilité, son aide, pour ses conseils pertinents sur le plan technique que humain, sa présence et son soutien moral dans les moments de stress et de doutes aux cours de ces dernières années et pour avoir apporté tant de soins à la réalisation de ce projet de fin d'études.

Nos sincères remerciements vont également à tous les enseignants qui nous ont formées durant ces cinq dernières années.

Nous exprimons notre gratitude à Madame DIDI, pour l'honneur qu'elle nous fait de présider notre jury de soutenance nous lui exprimons notre gratitude profonde.

Nous remercions chaleureusement Messieurs les Examinateurs BENMAMMAR et BELHOCINE pour l'intérêt qu'ils ont témoigné à

notre modeste travail en acceptant de participer au jury de soutenance

Nous adressons également nos sincères remerciements à nos famille ; parents, frères, sœurs et Otman de nous avoir aidé à surmonter tous les obstacles et à nous forger à travers les difficultés vécues durant toute cette période de travail.

Nous aimerions également remercier tous nos amis et collègues de leur soutien et aide et qui nous ont donné la force pour continuer.

Nous ne pouvant malheureusement pas mentionner toutes les personnes que nous avons rencontrées durant notre parcouru et qui ont contribué d'une façon ou d'une autre, à l'aboutissement de ce mémoire, nous leurs disons à tous merci d'avoir été là à cet instant précis.

Nous conclurions, en remerciant vivement toutes nos familles qui nous ont toujours supportées moralement et financièr ement pendant toutes nos longues années d'études.....

Dédicaces

Merci Allah (mon dieu) de m'avoir donné la capacité d'écrire et de réfléchir, la force d'y croire, la patience d'aller jusqu'au bout de mes rêves.

Je dédie ce modeste travail à celle qui m'a donné la vie, le symbole de tendresse, qui s'est sacrifiée pour mon bonheur et ma réussite, à ma mère "Meziane Zahira".

A mon père "Benmahdi Chiheb", école de mon enfance, qui a été mon ombre durant toutes les années de mes études, et qui a veillé tout au long de ma vie à m'encourager, à me donner l'aide et à me protéger. Que dieu les gardes et les protèges.

A ma mamie "Ouali Amaria" pour son amour, sa complicité, sa tendresse, sa douceur, et sa gentillesse.

A mon futur mari "Habri Otman Zakaria" pour son soutien, sa compréhension, son attention, sa patience et ces encouragements merci pour tout.

A mon petit frère "Benmahdi Nazim".

A ma sœur "Benmahdi Imane Nazha " et a son fiancé "Sari Younes".

A toute ma famille "Benmahdi", "Meziane" et "Habri".

A mon encadreur " Mr LEHASAINI".

A mon amie et mon binôme "Siham" et à toute sa famille "Dib".

A mes chers amis partout dans le monde et particulièrement mes amies Lamia, Imane, et Ahlem pour tout leur soutien moral, merci pour m'avoir supportée et pour tous vos encouragements répétés.

A tous ceux qui me sont chers.

A tous ceux qui m'aiment.

A tous ceux que j'aime.

A tous ceux qui je compte pour eux et qui comptent pour moi…

A tous ceux qui se sentent participants dans ma réussite…

Je dédie ce travail.

Que la paix d'Allah soit avec tous…

bouchra

Je dédie ce mémoire

A Dieu Tout Puissant, qui m'a accordé santé, force et courage pour la rédaction de ce mémoire.

A mes parents pour leur amour inestimable, leur confiance, leur soutien, leurs sacrifices et toutes les valeurs qu'ils ont su m'inculquer, je ne cesse de les remercier pour tout ce qu'ils m'ont donné.

A ma grand-mère pour sa douceur et sa gentillesse.

A mes sœurs et mon frère ainsi qu'à mes beaux frères pour leur tendresse, leur complicité et leur présence malgré la distance qui nous sépare.

A mes tantes et mes oncles pour toute l'affection qu'ils m'ont donnée et pour leurs précieux encouragements.

A mon neveu, ma plus grande source de bonheur, j'espère que la vie lui réserve le meilleur.

A mon encadreur.

A mon binôme avec qui j'ai partagé de belles années d'études et avec qui j'ai eu l'honneur de les finir.

A toute ma famille ainsi qu'à mes amies.

Siham

Table des matières

Introduction générale

L'émergence des réseaux de capteurs sans fil (RCSF) ouvre la voie au déploiement de nouvelles applications dans divers domaines. Ces applications amènent de nouveaux défis scientifiques et technologiques qui ont retenu l'attention d'un très grand nombre de chercheurs au cours des dernières années.

Les RCSF représentent une révolution technologique des instruments de mesures, issue de la convergence des systèmes électroniques miniaturisés et des systèmes de communication sans fil. Il s'agit d'ensembles d'unités électroniques miniaturisées capables de mesurer certains phénomènes physiques dans l'environnement où ils sont déployés. En raison des contraintes de miniaturisation, et aussi de coût de fabrication, les *nœuds de capteurs* sont généralement dotés de ressources très limitées en termes de capacité de calcul, d'espace de stockage de données, de débit de transmission et d'énergie embarquée. Ces limitations motivent une grande partie des problématiques de recherche dans le domaine des réseaux de capteurs sans fil, en particulier la contrainte de l'énergie qui est un problème fondamental.

Il est couramment admis que la plupart de l'énergie dissipée dans un nœud concerne la transmission et la réception de données. Si l'application le permet, il est donc préférable de transmettre des mesures quand un événement est détecté dans la zone de perception du nœud capteur ou par demande directe plutôt que de transmettre les mesures périodiquement.

Au delà du mode de fonctionnement de l'application, une des techniques les plus utilisées pour diminuer l'énergie dépensée pour la transmission des données est *l'agrégation des données*. L'agrégation de données, souvent appelée *fusion de données*, consiste à combiner les données provenant de différentes sources pour éliminer les redondances, ce qui a pour effet de réduire le trafic global du réseau.

L'objectif de ce travail est de réaliser une application sur des capteurs réels qui exploite l'agrégation de données en permettant de connaître l'importance de cette dernière en termes d'économie d'énergie.

Ce manuscrit est organisé en trois chapitres. Dans le premier chapitre, nous présentons un récapitulatif sur les réseaux de capteurs sans fil. Le deuxième chapitre est une présentation des outils matériels et logiciels nécessaires à la réalisation de notre application. Le troisième chapitre constitue le cœur de notre travail. Dans ce chapitre, nous présentons une architecture complète permettant de réaliser une application de l'agrégation.

Enfin, nous concluons notre travail en présentant les résultats obtenus et en donnant quelques perspectives.

Chapitre I
Récapitulatif sur les réseaux de capteurs sans fil

I.1 Introduction

Au cours de ces dernières décennies, une nouvelle technologie a bouleversé le monde et notre manière de vivre en présentant un champ d'applications très vaste appelée "les réseaux de capteurs sans fil" (RCSF). Cette dernière a intégré le compromis entre la miniaturisation du matériel et l'embarquement de l'unité de calcul et de communication sans fil pour un coût réduit.

Dans ce chapitre, nous présentons les réseaux de capteurs sans fil avec un plan méthodologique que nous avons adopté. En commençant par une définition d'un capteur, sa composition, ses types et voir comment ces derniers sont déployés pour former un réseau de capteurs sans fil. Ensuite, l'architecture et les différents facteurs de conception ainsi que les caractéristiques des RCSF seront étudiés. Avant de conclure ce chapitre, nous faisons une étude un peu détaillée sur la consommation d'énergie d'un nœud capteur, les problématiques liées aux réseaux de capteurs, domaines d'application des réseaux de capteurs sans fil et une petite conclusion.

I.2 Qu'est-ce qu'un nœud capteur ?

Un nœud capteur appelé "mote" en anglais est un équipement électronique doté de plusieurs capteurs dont chacun d'eux est capable de mesurer une grandeur environnementale, physiologique ou physique appelée aussi mesurande.

L'élément capteur nous permet sous l'effet du mesurande d'en délivrer une image exploitable présentée sous forme de signal électrique par exemple. D'où, la grandeur physique en entrée (mesurande) se transforme en une autre grandeur de sortie ou un signal électrique [1].

Généralement, on obtient une grandeur de sortie présentée sous forme d'un signal électrique comme montre la figure I.1 et qui peut être soit:

- une charge,

- une tension,

- un courant,

- une impédance (R, L, C) [1].

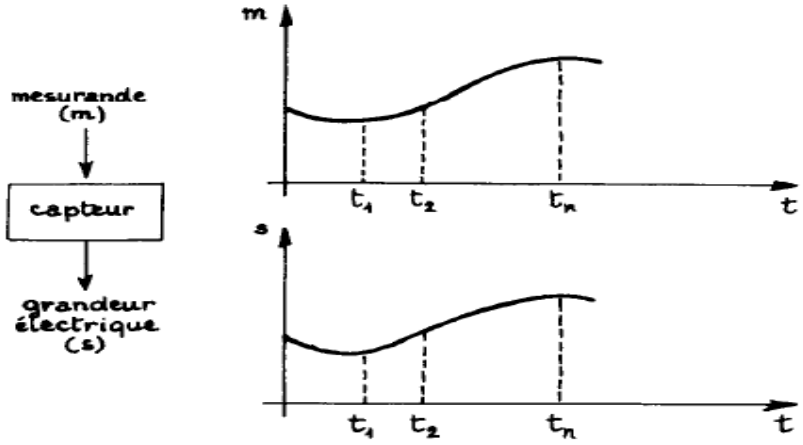


Figure I-1: Principe de fonctionnement d'un capteur

I.2.1 Capteur intelligent

Le terme capteur intelligent (smart sensor ou intelligent sensor) a été utilisé dans l'industrie des capteurs pour désigner des capteurs qui ne fournissent pas seulement des mesures, mais aussi une fonctionnalité aux mesures spécifiques [2]. Comparativement à un capteur classique, un capteur intelligent intègre de nombreux éléments électroniques additionnels, ainsi que des unités programmables et des aspects logiciels nécessaires au traitement des données, aux calculs, à la communication numérique [3]. Il est caractérisé par [2] sa capacité à effectuer une collecte des mesures, les traiter et à les communiquer au monde extérieur.

Le mot capteur est utilisé par abus de langage pour désigner un capteur intelligent. Ainsi, pour ne pas rompre avec la terminologie usuelle, dans le reste de notre travail, lorsque nous parlerons d'un capteur, nous sous-entendrons un capteur intelligent [4].

I.2.2 Architecture d'un capteur sans fil

Dans cette section, nous distinguons les deux parties qui composent un capteur

a. Architecture matérielle

Un nœud capteur est composé de quatre unités de base comme le montre la figure I.2:

- **Unité de communication (*Transceiver unit*)**: elle est composée d'un émetteur/récepteur (module radio) permettant la communication entre les différents nœuds du réseau.

- **Unité de capture (*Sensing unit*)**: elle est composée de deux sous-unités, un dispositif de capture physique qui prélève l'information de l'environnement local et un convertisseur analogique/numérique appelé ADC[1].

- **Unité de traitement (*Processing unit*)**: les données captées sont communiquées au processeur pour être traitées et par la suite elles sont stockées dans la mémoire.

- **Unité d'énergie (*Power unit*)**: c'est la batterie qui n'est généralement ni rechargeable ni remplaçable. Elle permet d'alimenter les autres composants d'un capteur.

D'autres éléments optionnels peuvent être intégrés pour certaines applications spécifiques, à savoir: une unité de localisation géographique, un régénérateur d'énergie et un mobilisateur:

- **Unité de localisation:** dans plusieurs applications, les tâches de détection et les techniques de routage ont besoin de connaître la localisation géographique. Le rôle de cette unité est d'identifier leur position géographique, par exemple en utilisant un récepteur GPS ou une technique de triangulation [5].

- **Le mobilisateur:** permet à un nœud de se déplacer pour accomplir ses tâches [5]. Le support de mobilité exige des ressources énergétiques étendues qui devraient être fournies efficacement. Il peut aussi contrôler les mouvements d'un nœud en opérant dans l'interaction étroite avec l'unité de détection et le processeur [4,7].

- **Le régénérateur de l'énergie:** un générateur électrique supplémentaire peut être utilisé pour des applications conçues dans des zones d'intérêt très éloignées. Pour ce type d'applications extérieures, des piles solaires sont utilisées pour générer l'alimentation électrique. De même, des techniques de récupération d'énergie peuvent également être utilisées.

[1]Analog to Digital Converters

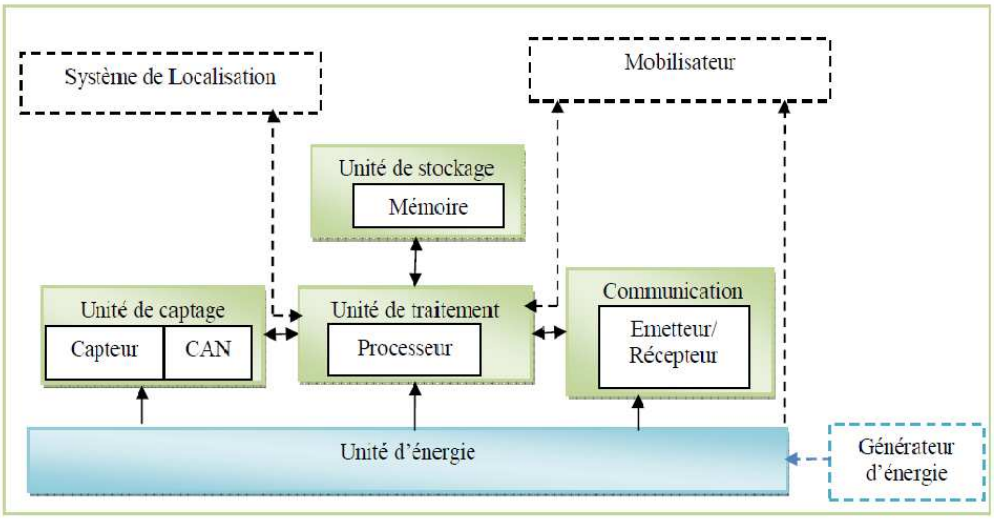


Figure I-2: Architecture d'un nœud capteur [4]

c. Architecture Logicielle

L'un des systèmes d'exploitation les plus connus dans le domaine des RCSF est «TinyOS ». Il est libre et est utilisé par une large communauté de scientifiques dans des simulations pour le développement et le test des algorithmes et protocoles réseau [8].

I.2.3 Caractéristiques d'un capteur

En analysant la gamme des composants disponibles sur le marché et les prototypes présentés dans la littérature, il est évident que la principale caractéristique d'un nœud de capteurs sans fil est sa petite taille. Depuis que les premiers nœuds de capteurs sans fil sont apparus, il y a une tendance à leur miniaturisation. Une deuxième caractéristique, évidente mais essentielle, est l'autonomie d'énergie. Ces deux premières particularités induisent plusieurs autres caractéristiques à considérer, en particulier la vitesse de calcul et la rapidité de transmission. Des performances élevées en termes de vitesse de traitement et de transmission impliquent une consommation d'énergie élevée. D'une manière générale, il est souhaitable que la durée de vie de la batterie d'un nœud capteur soit la plus grande possible, donc les différentes unités qui composent un nœud sont généralement très limitées en termes de ressources et de performances pour que leur consommation d'énergie soit extrêmement faible [5].

I.3 Réseaux de capteurs sans fil

I.3.1 Présentation des réseaux de capteurs

Un Réseau de Capteurs Sans Fil (RCSF) est un type particulier de réseaux mobiles ad hoc MANETs[2][7]. Il est composé d'un ensemble de petits dispositifs nommés nœuds capteurs, variant de quelques dizaines d'éléments à plusieurs milliers et qui sont déployés généralement d'une manière aléatoire dans une zone d'intérêt. Dans ces réseaux, chaque nœud est capable de surveiller son environnement et de réagir en cas de besoin en envoyant l'information collectée à un ou plusieurs points de collecte appelés stations de base ou nœuds puits, à l'aide d'une connexion sans fil [9]. La figure 3 illustre le déploiement d'un RCSF dans une zone d'intérêt.

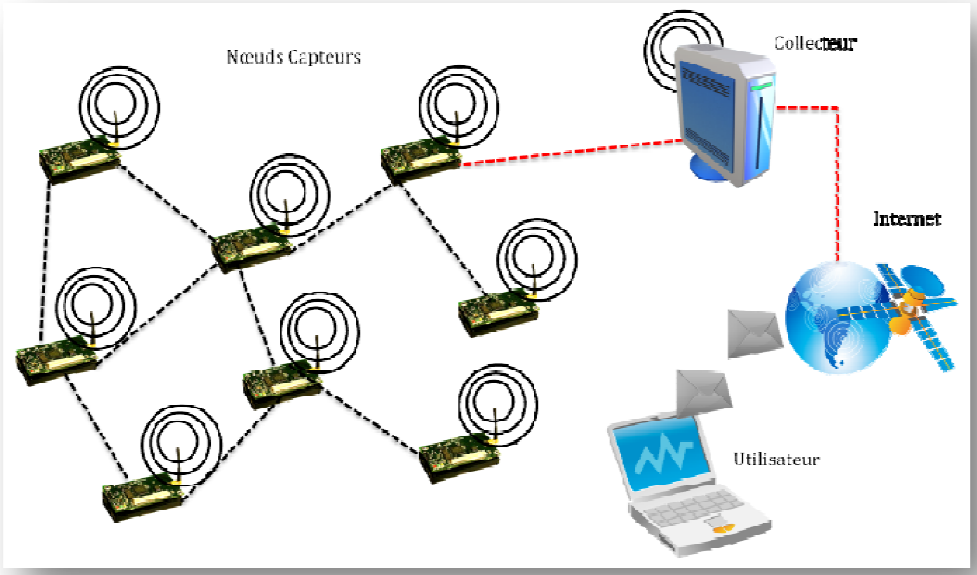


Figure I-3: Exemple des RCSF [10,11]

I.3.2 Histoire des RCSF

Les récents progrès des techniques ont provoqué un énorme intérêt dans les réseaux sans fil. Ces progrès ont rendu la technologie des RCSF une des merveilles technologiques dans le 21ième siècle puisque les RCSF ont montré leur impact sur notre vie quotidienne.

[2]Mobile Ad hoc Networks

Dans [6], les auteurs ont classifié l'évolution des RCSF en trois générations comme s'est montré dans le tableau I.1.

Tableau I-1: Evolution des RCSF

Génération	Période	Taille	Poids	Batterie
1er	Les années 80 et 90	Grande boîte à chaussures	Kilogrammes	Grosse
2e	2000-2003	Boîte de cartes	Grammes	AA
3e	2010	Particule de poussière	Négligeable	Solaire

I.3.3 Domaines d'applications

La diminution de taille et de coût des micro-capteurs, l'élargissement de la gamme des types de capteurs disponibles (thermique, optique, vibrations...) et l'évolution des supports de communication sans fil, ont élargi le champ d'applications des RCSF comme s'est illustre par la figure 4. Parmi ces applications, nous citons:

- **Applications militaires:** On peut penser à un réseau de capteurs déployé dans un endroit stratégique ou difficile d'accès, afin de surveiller toutes les activités des forces ennemies, ou d'analyser le terrain avant d'y envoyer des troupes (détection d'agents chimiques, biologiques ou de radiations).

- **Applications domestiques:** En plaçant, sur le plafond ou dans le mur, des capteurs, on peut économiser l'énergie en gérant l'éclairage ou le chauffage en fonction de la localisation des personnes. En particulier, ils sont très utiles pour assurer l'autonomie des personnes dépendantes dans le cadre de la maison intelligente.

- **Applications environnementales:** Les réseaux de capteurs sont beaucoup appliqués dans ce domaine pour détecter des incendies, surveiller des catastrophes naturelles, détecter des pollutions et suivre des écosystèmes.

- **Applications agricoles:** Dans les champs agricoles, les capteurs peuvent être semés avec les graines. Ainsi, les zones sèches seront facilement identifiées et l'irrigation sera donc plus efficace et économique.

- **Applications médicales:** Les réseaux de capteurs ont aussi des développements dans le domaine de diagnostic médical. Par exemple, des micro-caméras sont capables, sans

avoir recours à la chirurgie, de transmettre des images de l'intérieur d'un corps humain avec une autonomie de 24 heures.

- **Applications de transport:** Il est possible d'intégrer des nœuds capteurs au processus de stockage et de livraison. Le réseau ainsi formé, pourra être utilisé pour connaître la position, l'état et la direction d'un paquet ou d'une cargaison [12].

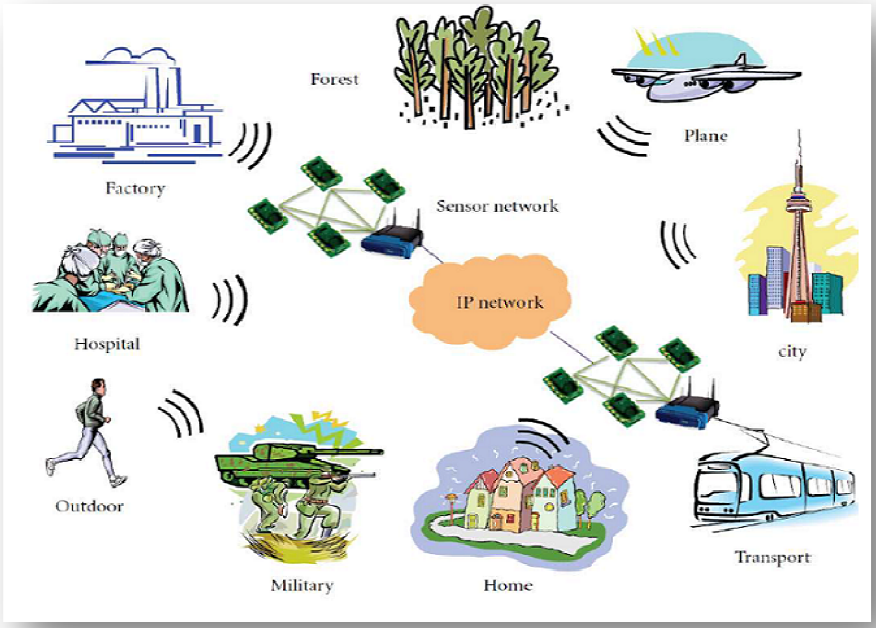


Figure I-4: Quelques domaines d'application dans les RCSF [4]

I.4 Architecture d'un RCSF

Le but d'un réseau de capteurs est de détecter, collecter et envoyer une ou plusieurs informations. Il est constitué de deux types de nœuds:

- Les capteurs qui envoient l'information
- Les puits qui récoltent les informations

Généralement, il y a deux modes pour récupérer les données:

- Un mode piloté par le puits qui envoie par broadcast une demande d'envoi d'information.
- Un envoie d'information dynamique qui réagit à un événement défini au préalable.

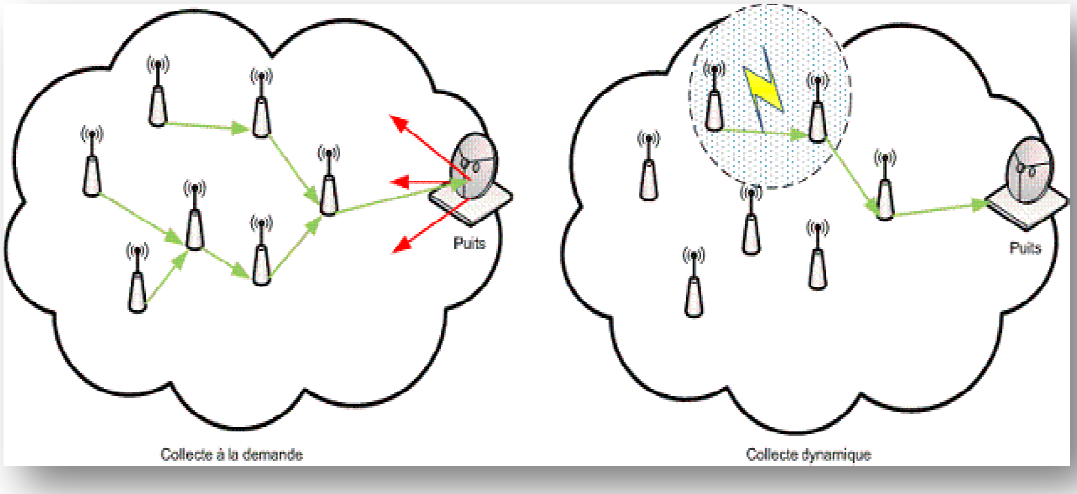


Figure I-5: Modes d'envoi d'informations dans les RCSF [13]

Il existe deux types d'architectures pour les réseaux de capteurs: les réseaux de capteurs plats et les réseaux de capteurs hiérarchiques [14].

I.4.1 Les RCSF plats

Un RCSF plat est un réseau homogène, où tous les nœuds sont identiques en termes de batterie et des fonctions, excepté le nœud de collecte [15]. Ce dernier joue le rôle d'une passerelle et est responsable de la transmission de l'information collectée à l'utilisateur final comme le montre la figure 5.

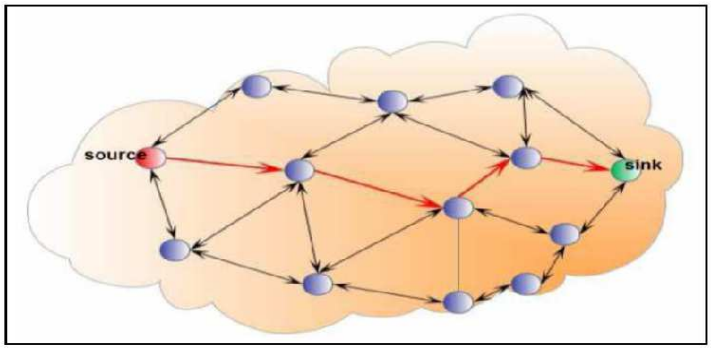


Figure I-6: Architecture d'un RCSF plat [16]

I.4.2 Les RCSF hiérarchiques

L'architecture hiérarchique a été proposée pour réduire le coût et la complexité de communications dans les RCSF.

Elle consiste à introduire un ensemble de nœuds plus coûteux et plus puissants, en créant une infrastructure qui décharge la majorité des nœuds simples à faible coût de plusieurs fonctions du réseau. Cette architecture est composée de plusieurs couches: une couche de capteur, une couche de transmission et une couche de point d'accès, la figure 6 le montre.

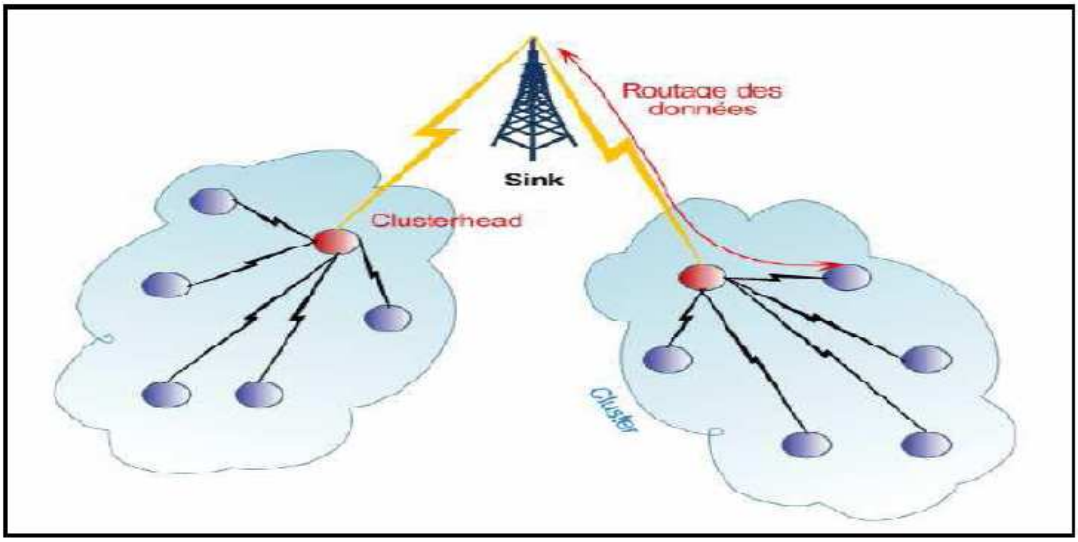


Figure I-7: Architecture d'un RCSF hiérarchique [16]

I.5 Caractéristiques d'un réseau de capteurs

Parmi les caractéristiques les plus importantes d'un réseau de capteurs, nous citons:

- La durée de vie limitée**:** Les nœuds capteurs sont très limités par la contrainte d'énergie. Ils fonctionnent habituellement sans surveillance dans des régions géographiques éloignées. Par conséquent recharger ou remplacer leurs batteries devient quasiment impossible.

- Ressources limitées**:** Habituellement les nœuds capteurs ont une taille très petite. Ce facteur de forme limite la quantité de ressources qui peuvent être mises dans ces nœuds. En conséquence, la capacité de traitement et de mémoire est très limitée.

- Topologie dynamique**:** La topologie des réseaux de capteurs change d'une manière fréquente et rapide car les nœuds capteurs peuvent être déployés dans des environnements hostiles (par exemple un champ de bataille) où la défaillance d'un nœud capteur peut donc être très probable. De plus, les nœuds capteurs peuvent être mobiles par exemple ils sont placés sur des animaux.

- Agrégation des données**:** Dans les réseaux de capteurs, les données collectées par les nœuds capteurs sont similaires, ce qui implique l'existence de redondance de données. Une approche répandue consiste à agréger les données au niveau des nœuds intermédiaires appelés nœuds agrégateurs afin de réduire la consommation d'énergie lors de la transmission de ces données.

- Passage à l'échelle**:** les réseaux de capteurs engendrent un très grand nombre de capteurs, ils peuvent atteindre des milliers voire des millions de capteurs. Le défi à relever par les RCSF est d'être capable de maintenir leurs performances avec ce grand nombre de capteurs.

- Bande passante limitée: à cause de cette contrainte, les nœuds capteurs ne peuvent pas permettre des débits élevés.

- Sécurité physique limitée**:** cela se justifie par les contraintes et limitations physiques qui minimisent le contrôle des données transmises [17].

I.6 Consommation d'énergie dans les RCSF

L'énergie consommée par un nœud capteur est due essentiellement aux opérations suivantes: la capture, le traitement et la communication de données [18].

I.6.1 Énergie de capture

L'énergie de capture est dissipée pour accomplir les tâches suivantes: échantillonnage, traitement de signal, conversion analogique/numérique et activation de la sonde du capture. En général, l'énergie de capture représente un faible pourcentage de l'énergie totale consommée par un nœud.

I.6.2 Énergie de traitement

L'énergie de traitement se divise en deux parties: l'énergie de commutation et l'énergie de fuite. L'énergie de commutation est déterminée par la tension d'alimentation et la capacité totale commutée au niveau logiciel lors de l'exécution du logiciel. Par contre, l'énergie de fuite correspond à l'énergie consommée lorsque l'unité de calcul n'effectue aucun traitement. En général, l'énergie de traitement est faible par rapport à celle nécessaire pour la communication.

I.6.3 Énergie de communication

L'énergie de communication se divise en deux parties: l'énergie de réception et l'énergie de l'émission. Cette énergie est déterminée par la quantité des données à communiquer et la distance de transmission, ainsi que par les propriétés physiques du module radio. L'émission d'un signal est caractérisée par sa puissance. Quand la puissance d'émission est élevée, le signal aura une grande portée et l'énergie consommée sera plus élevée. Notons que l'énergie de communication représente la portion la plus grande de l'énergie consommée par un nœud capteur [6].

I.7 Conclusion

Dans ce chapitre, nous avons présenté les réseaux de capteurs sans fil, leurs caractéristiques et les concepts nécessaires à la compréhension des réseaux de capteurs. En outre, on a mis le point sur quelques facteurs permettant l'économie de l'énergie tels que l'agrégation de données qui permet de minimiser le nombre de messages redondants.

Dans le chapitre qui suit, nous présentons les outils matériels et logiciels qui permettent la mise en place d'une application d'agrégation de données.

Chapitre II
Les outils de développement dans les RCSF

II.1 Introduction

La réalisation des applications dans les réseaux de capteurs nécessite des outils logiciels bien spécifiques puisque les capteurs sont considérés comme des dispositifs à ressources limitées.

Dans ce chapitre, nous allons détailler les principaux outils logiciels utilisés dans les réseaux de capteurs sans fil. Parmi ces outils, nous présentons les systèmes d'exploitation conçus spécialement pour les capteurs en particulier TinyOS et le langage de développement NesC [19].

II.2 Systèmes d'exploitation

Dans cette section, nous présentons quelques systèmes d'exploitation dédiés aux réseaux de capteurs. Nous commençons tout d'abord par TinyOS qui est le système d'exploitation le plus réputé pour les dispositifs à ressources limitées et pour la plupart des plates-formes des capteurs [20].

II.2.1 Le système d'exploitation TinyOS

a. Présentation de TinyOS

TinyOS a commencé comme projet chez Berkeley UC en tant qu'élément du programme DARPA. Dans son développement, des milliers d'universitaires, lotisseurs et utilisateurs commerciaux dans le monde entier ont été impliqués.

TinyOS est un système d'exploitation open-source spécialement conçu pour les réseaux de capteurs sans fil. Il est devenu le standard de facto lequel le propose en téléchargement sous la licence BSD qui est une licence libre utilisée pour la distribution de logiciels en assurant le suivi. Ainsi, l'ensemble des sources sont disponibles pour de nombreuses cibles matérielles.

En outre, TinyOS fonctionne sur une base d'association de composants, ce qui réduit significativement la taille du code. Il travaille aussi avec NesC, un langage dérivé du C qui a été conçu pour minimiser l'utilisation de mémoire des capteurs [12].

TinyOS fait partie des environnements exécutifs dont l'objectif est de fournir une couche d'abstraction au moment du développement, avec la possibilité d'écrire un code portable et réutilisable, sans perdre de performance au moment de l'exécution puisque l'application monolithique exploite au mieux les ressources disponibles en étant statique [21].

TinyOS a été créé pour répondre aux caractéristiques et aux nécessités des réseaux de capteurs, telles que:

- Une taille de mémoire réduite et une basse consommation d'énergie.
- Des opérations d'assistance intensive et robustes.
- Il est optimisé en termes d'usage de mémoire et d'énergie [20].

Sa conception a été entièrement réalisée en NesC [23], langage orienté composant proche du C, et la bibliothèque de composants de TinyOS est particulièrement complète puisqu'on y retrouve des protocoles réseaux, des pilotes de capteurs et des outils d'acquisition de données. Un programme s'exécutant sur TinyOS est constitué d'une sélection de composants systèmes et de composants développés spécifiquement pour l'application à laquelle il sera destiné (mesure de température, du taux d'humidité...).

TinyOS s'appuie sur un fonctionnement événementiel, c'est-à-dire qu'il ne devient actif qu'à l'apparition de certains événements, par exemple l'arrivée d'un message radio. Le reste du temps, le capteur se trouve en état de veille, garantissant une durée de vie maximale connaissant les faibles ressources énergétiques des capteurs. Ce type de fonctionnement permet une meilleure adaptation à la nature aléatoire de la communication sans fil entre capteurs [22, 23, 24].

b. Architecture générale des cibles utilisant TinyOS

TinyOS est prévu pour mettre en place des applications sur des capteurs équipés d'une radio et d'une antenne afin de se connecter à la couche physique et constituent les émissions hertziennes, on retrouve donc des équipements bardés de différents types de détecteurs et autres entrées [25]. Comme tout dispositif embarqué, ceux utilisant TinyOS sont pourvus d'une alimentation autonome telle qu'une batterie.

c. Propriétés de TinyOS

TinyOS est basé sur quatre grandes propriétés qui permettent à ce système d'exploitation de s'adapter particulièrement aux systèmes à faibles ressources:

- Événementiel**:** Le fonctionnement d'un système basé sur TinyOS s'appuie sur la gestion des événements qui se produisent. Ainsi, l'activation de tâches, leur interruption ou encore la mise en veille du capteur s'effectue à l'apparition d'événements, ceux-ci ayant la plus forte priorité. Ce fonctionnement événementiel (**event-driven**) s'oppose au fonctionnement dit temporel (**time-driven**) où les actions du système sont gérées par une horloge donnée.

- Non préemptif**:** Le caractère préemptif d'un système d'exploitation précise si celui-ci permet l'interruption d'une tâche en cours. TinyOS ne gère pas ce mécanisme de préemption entre les tâches mais donne la priorité aux interruptions matérielles. Ainsi, les tâches entre elles ne s'interrompent pas mais une interruption peut stopper l'exécution d'une tâche.

- Pas de temps réel: Un système est dit « temps réel » si celui-ci gère des niveaux de priorité dans ses tâches permettant de respecter des échéances données par son environnement. Dans le cas d'un système strict, aucune échéance ne tolère de dépassement contrairement à un système temps réel mou. TinyOS se situe au-delà de ce second type car il n'est pas prévu pour avoir un fonctionnement temps réel.

- Consommation: TinyOS a été conçu pour réduire au maximum la consommation d'énergie du capteur. Ainsi, lorsqu'aucune tâche n'est active, il se met automatiquement en veille [26].

d. Concepts de TinyOS

Le système TinyOS est un exo noyau qui utilise une approche composants et non objets comme beaucoup d'autres systèmes [27].

On distingue la différence entre un exo noyau et les noyaux plus classiques et les différences entre la programmation par composants et la programmation orientée objet [28].

Plusieurs types de noyaux existent et qui sont conçus pour des besoins différents.

- Les noyaux monolithiques**:** Dans un noyau monolithique (modulaire ou non) les pilotes matériels et les services sont intégrés dans un même bloc. L'avantage de ce type de noyaux est qu'il assure une grande rapidité d'exécution puisque les services intégrés dans le noyau ne génèrent pas des coûts d'appels système supplémentaires. Cependant, un des inconvénients de cette approche est que lorsqu'une erreur survient dans un des modules ou services du noyau elle menace la stabilité tout le système en entier.

Ce type de noyaux est le premier qui a été développé, et il est encore utilisé par les systèmes Linux, BSD et Solaris. En outre, dans ce noyau, les programmes de l'espace utilisateur font appel au noyau pour accéder au matériel.

- Les micros noyaux: Pour limiter le risque d'instabilité du système lors de l'occurrence des erreurs dans les services du noyau, les micros noyaux séparent les services du noyau pour les placer dans l'espace utilisateur. En outre, si une erreur survient dans un de ces services elle est limitée à ce service et ne compromet pas tout le système.

Dans ce type, le noyau fournit uniquement les abstractions les plus basiques nécessaires. Ainsi, l'avantage de ce noyau est d'améliorer en théorie la fiabilité du système tout en assurant sa stabilité et de permettre au programmeur de pouvoir mieux contrôler le matériel sans utiliser des abstractions de haut niveau comme dans un noyau monolithique. En contrepartie, la séparation des services engendrent de nombreux appels systèmes et les mécanismes de communication entre les services deviennent complexes et lourds en temps de traitement, ce qu'il en résulte un système aux performances diminuées.

Exemple: Mac OS X et Windows XP.

- Les exo noyaux**:** Le principe des exo noyaux est de libérer le programmeur des abstractions matérielles c'est-à-dire le système fournit uniquement des interfaces donnant au programmeur un accès direct au matériel sans rajouter de fonctionnalités. En outre, les abstractions de haut niveau sont fournies dans des librairies extérieures et par suite le programmeur devient libre de les utiliser ou non et seules les librairies vraiment nécessaires sont utilisés, contrairement aux noyaux monolithiques [29].

e. Tâches, événements et applications

TinyOs est basé sur la gestion de tâches et d'événements. Une tâche est un bloc d'instructions et un événement est l'équivalent logiciel d'une interruption matérielle qui a une priorité sur les tâches. Chaque tâche est activée ou interrompue en fonction de l'apparition d'un événement. Cependant, les tâches ne peuvent pas s'interrompre entre elles puisque TinyOs n'étant pas préemptif, mais elles peuvent l'être par un événement.

- Gestion des tâches**:** Chaque tâche activée est mise en attente dans une file d'attente de type FIFO (première arrivée première sortie), lorsque la file des tâches est vide le système se met en veille en attendant le prochain événement. Ce mécanisme de tâches a pour avantage d'empêcher une tâche d'en interrompre une autre pouvant bloquer le système, mais il a aussi pour inconvénient de ne pas permettre une gestion en temps réel. En plus, pour les tâches de longue durée TinyOs possède un mécanisme permettant de fragmenter l'exécution d'une tâche nommée split-phase qui permet de ne pas bloquer le système. Ce mécanisme est utilisé dans l'initialisation de composants qui demandent du temps au démarrage, comme la radio par exemple.

- Les événements: Lorsqu'une interruption matérielle survient, l'événement correspondant reçoit un signal et prend la main de manière asynchrone c'est-à-dire qu'il n'attend pas la fin de la tâche courante pour s'exécuter. Il existe des événements qui peuvent être signalés et ne correspondent pas forcément à une interruption matérielle. En plus, il existe également des événements synchronisés qui sont mis en attente dans la liste des tâches, avec une priorité supérieure aux tâches en attente mais ils n'interrompent pas la tâche courante comme dans le cas de certains Timers.

- Les applications**:** Les applications basées sur TinyOs sont formées de composants réutilisables et portables (comme les Timers, les convertisseurs de signal ou la radio) qui sont reliés entre eux. Ces composants peuvent être directement liés au matériel (composant gérant les LEDs par exemple) ou un regroupement de plusieurs composants de bas niveau (composants gérant les envois de données par la radio). Les composants sont implémentés en utilisant les tâches, les événements et des commandes qui permettent de faire appel aux fonctionnalités d'autres composants auxquels ils sont liés.

f. Abstraction matérielle

TinyOs utilise une architecture d'abstraction du matériel HAA[3] qui permet de satisfaire les besoins de portabilité et de réutilisabilité des applications et des composants tout en préservant l'efficacité et les performances de chaque plateforme. L'utilisation de ce type d'abstraction dans le développement de systèmes d'exploitation a permis de simplifier grandement le développement des applications en cachant le matériel aux développeurs, mais elle comporte aussi l'inconvénient de réduire les performances du système. C'est pourquoi il faut que le choix de cette architecture soit fait de manière optimale.

L'architecture d'abstraction du matériel de TinyOs est organisée en trois couches distinctes de composants. Chaque couche a un but bien précis et est dépendante des interfaces fournies par les couches inférieures. Plus on se trouve dans les couches supérieures, moins le code utilisé dépend du matériel pour finalement arriver à des applications complètement indépendantes de la plateforme sur laquelle il tourne tout en permettant leur portabilité. La figure II.1 illustre un exemple de couches pour un composant utilisant l'ADC.

- **Couche HPL**: C'est la couche de niveau le plus bas. Elle se situe juste à l'interface entre le matériel et le logiciel. Son but est de présenter les possibilités du matériel en mettant à disposition de la couche supérieure un ensemble de méthodes qui permettent de contrôler le matériel. Chaque composant de cette couche doit fournir au minimum les méthodes suivantes:
  - Des commandes d'initialisation, de démarrage et d'arrêt du composant matériel pour assurer une gestion optimale de l'énergie.
  - Des méthodes d'accès aux registres du matériel (lecture, écriture).
  - Des méthodes permettant de modifier simplement les flags les plus utilisés (sans avoir à modifier tout le registre qui les contient).
  - Des commandes permettant d'activer ou de désactiver les différentes interruptions générées par le matériel.
  - La gestion des interruptions matérielles pour les transmettre à la couche supérieure.

[3] HAA : Hardware Abstraction Architecture

- **Couche HAL**: C'est la couche centrale de l'architecture. Les composants de cette couche utilisent les interfaces fournies par la HPL pour construire des méthodes permettant d'utiliser au mieux les capacités de la plateforme pour lesquels ils sont créés. Par exemple un composant devant faire la conversion de valeurs analogiques en valeurs digitales (ADC) dépend de la plateforme sur laquelle on l'utilise. Certaines plates-formes permettent la copie des valeurs converties par un contrôleur DMA qui accélère la copie. La couche HAL du convertisseur pour ces plates-formes doit en tenir compte et utiliser ce contrôleur DMA afin de fournir un module ADC le plus performant possible.

- **Couche HIL:** Cette couche met à la disposition du développeur les composants logiciels qui doivent être utilisables quelle que soit la plateforme sur laquelle s'exécute l'application. En fonction des performances des plates-formes pour lesquelles ces composants sont développés, il faut faire un choix parmi les possibilités offertes par chacune et brider certaines possibilités au niveau de la couche HIL pour garantir l'indépendance du composant par rapport au matériel. Dans le cas des applications dédiées uniquement à une plateforme, qui demandent un grand contrôle sur le matériel nous pouvons ne pas utiliser la couche HIL.

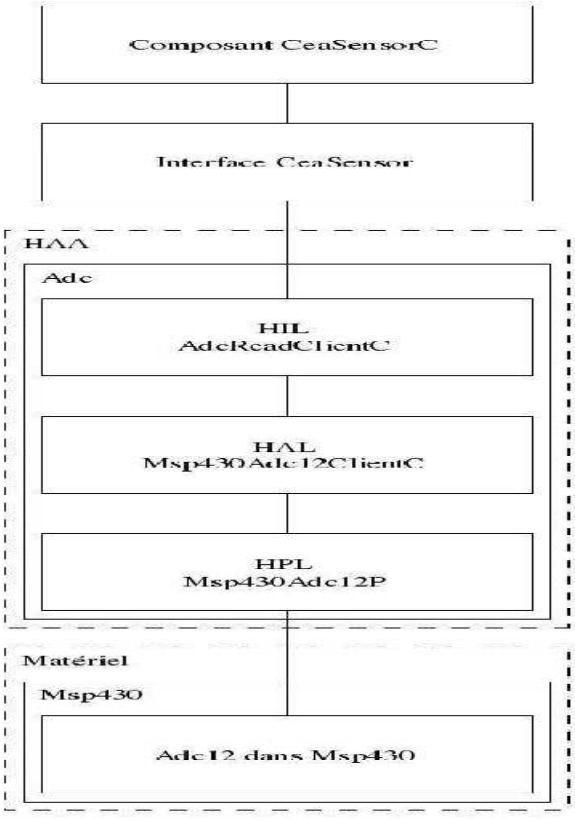


Figure II-1: Différentes couches pour un composant utilisant l'ADC [28]

II.2.2 Autres systèmes d'exploitation

Il existe d'autres systèmes d'exploitation dédiés aux réseaux de capteurs, nous citons dans ce qui suit les plus répandus:

- Contiki [33]: c'est un système d'exploitation open-source multi-tâche, développé pour les systèmes embarqués avec contraintes de mémoire.

- SOS [34]: système d'exploitation développé par l'université de Los Angeles en Californie écrit en langage C et qui reprend le système de programmation événementielle de TinyOS.

- FreeRTOS [32]: n'est pas vraiment un système d'exploitation, mais un noyau de système d'exploitation pour systèmes embarqués.

- Mantis OS [31]: système d'exploitation dédié aux réseaux de capteurs, développé par l'université du Colorado (USA) et écrit en langage C. Contrairement à TinyOS qui est basé sur un modèle de programmation événementielle, Mantis OS s'articule autour d'un modèle commandé par l'exécution de processus.

- Nut/OS [35] : Système d'exploitation multitâche pour systèmes embarqués avec une pile TCP/IP [30].

II.3 Le langage de programmation NesC

NesC est un langage conçu pour incarner les concepts structurants et le modèle d'exécution de TinyOS. C'est une extension du langage C orienté composant; il supporte alors la syntaxe du langage C et il est compilé vers le langage C avant sa compilation en binaire [20].

Les bibliothèques et les applications de TinyOS sont écrites en NesC, un nouveau langage de description pour la programmation structurée d'applications, adapté aux systèmes embarqués et donc en particulier aux réseaux de capteurs. La syntaxe de NesC ressemble à celle du C, mais elle autorise en plus l'exécution compétitive ainsi que des mécanismes pour structurer, nommer et lier des modules logiciels. Cela permet de faciliter le développement de modules logiciels qui peuvent être facilement assemblés les uns avec les autres [36].

TinyOS s'appuie sur le langage NesC. Celui-ci propose une architecture basée sur des composants [39], où chaque composant correspond à un élément matériel (LEDs,

timer,ADC ...) et peut être réutilisé dans différentes applications. Un composant est constitué de trois parties essentielles: interfaces, modules et configurations.

- **Les interfaces** permettent de spécifier des fonctions (des commandes ou des événements). Ces fonctions sont alors implémentées par le fournisseur ou l'utilisateur de l'interface, afin de distinguer les fonctions concernant les commandes de ceux concernant les événements, les fonctions sont précédées de la commande command ou event. La figure II.2 illustre un exemple de composant TinyOS.

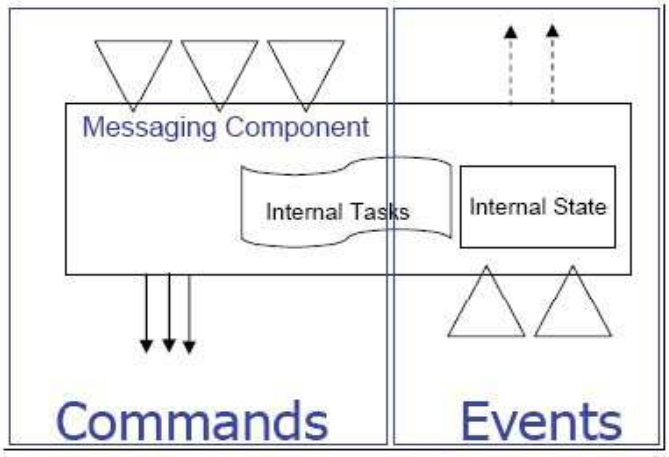


Figure II-2: Exemple de composant

- **Les modules** sont les éléments de base de la programmation. Ils permettent d'implémenter les composants et ils sont stockés dans un fichier qui possède la structure présentée par la figure II.3.

```
1  Timer.nc
2  interface Timer {
3   command result_t start(char type,
        uint32_t interval);
4   command result_t stop();
5   event result_t fired();
6  }
```

Figure II-3: Exemple de module

- **Les configurations** permettent de décrire l'architecture. Une configuration est donc constituée de modules et/ou d'interfaces ainsi que de la description des liaisons entre ces composants [37].

II.4 Plateforme TelosB

La plateforme TelosB a été élaborée et publiée à la communauté scientifique par l'université de Berkeley. Cette plateforme offre une faible consommation d'énergie permettant une longue autonomie de la batterie ainsi qu'un éveil rapide de l'état de veille. Le microcontrôleur TPR2420 utilisé dans TelosB, est compatible avec la distribution open-source de TinyOS. Ce type de nœud peut être utilisé dans les applications suivantes:

- Plate-forme à faible puissance pour le développement de la recherche
- Expérimentation des réseaux de capteurs sans fil [38]

Un nœud capteur de type TelosB est caractérisé par:

- Micro-contrôleur MSP430 de Texas Instrument cadencé à 8MHz, avec 10Kb de RAM et 48Kb de flash.
- Radio Chipcon Wireless Transceiver CC2420 2.4GHz à 250kbps respectant la norme IEEE 802.15.4.
- Convertisseurs analogique-digital et contrôleur DMA intégrés.
- Liaison USB.
- Très faible consommation électrique.
- Temps de réveil inférieur à 6us.
- Un support d'expansion 16 ports.
- Encombrement réduit.
- Supporté par TinyOS.

II.5 Conclusion

Dans ce chapitre, nous avons décrit les outils logiciels nécessaires pour le développement d'applications dans les RCSF. Puis, nous avons présenté la plateforme TelosB que nous avons utilisée pour développer une application qui permet d'agréger des données avant de les envoyer à la station de base.

Dans le chapitre suivant, nous allons aborder la partie implémentation de notre application dans ces deux versions. La première consiste à utiliser un nœud relais et la deuxième implique un nœud agrégateur.

Chapitre III
Implémentation de l'application

III.1 Introduction

La minimisation de la consommation d'énergie est vue comme un défi dans les RCSF puisque les capteurs disposent d'une quantité d'énergie limitée et ils sont généralement déployés dans des zones à accès difficile où le remplacement et le rechargement des batteries est une tâche difficile voire impossible. Dans cette optique, plusieurs travaux ont visé à minimiser la consommation d'énergie pour garantir une longue durée de vie dans ce type de réseaux. L'agrégation de données est l'une des solutions proposée dans ces travaux. Dans cette solution, les auteurs ont essayé de réduire les messages de données redondants puisque la consommation d'énergie lors de la transmission des données représente environ 70% de la consommation d'énergie au niveau d'un capteur.

Dans ce chapitre, nous avons développé une application sur des capteurs réels pour illustrer l'apport de l'agrégation de données dans les RCSF. Pour concrétiser cet apport, nous avons évalué un réseau de capteurs dans deux scénarios. Le premier est une architecture à plat dans laquelle il y a un nœud relai qui renvoie toute donnée reçue de la part des capteurs déployés. Le deuxième scénario implique un nœud agrégateur qui reçoit un ensemble de données des capteurs déployés dans son voisinage, agrège ces données en une seule et la retransmet à la station de base.

Dans ce qui suit, nous détaillons les éléments matériels et logiciels qui ont une relation avec notre application.

III.2 Partie matérielle

Dans cette section, nous présentons la plateforme des capteurs sur laquelle nous avons développé notre application.

III.2.1 Matériel utilisé

Nous avons développé notre application en utilisant: des capteurs de type TelosB, un PC portable de type Dell, des piles de type AA pour alimenter les capteurs, et un câble filaire qui relie les ports USB.

III.2.2 Plateforme matérielle

L'architecture de base de notre application contient un ensemble de capteurs des Senders, un Receiver, une station de base et un centre de contrôle. La figure III.1 représente la plateforme matérielle utilisée.

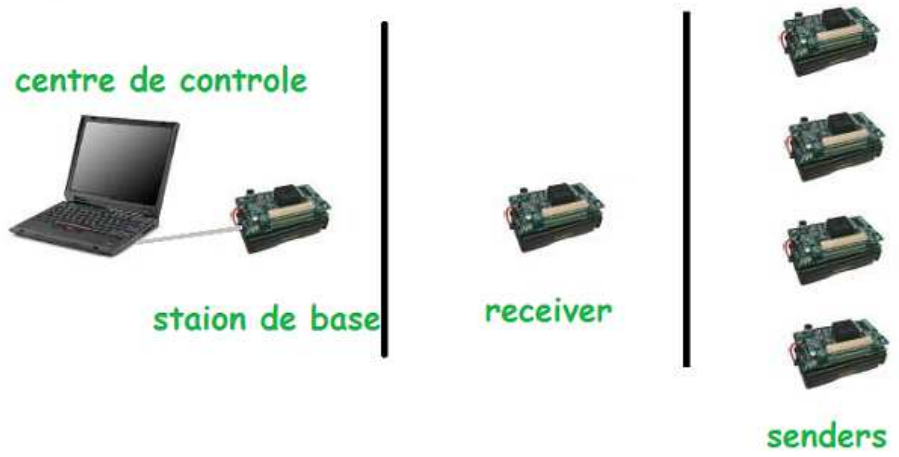


Figure III-1: Plateforme matérielle

Un ensemble de Senders est reliés à un Receiver via un lien radio. Le Receiver peut jouer le rôle d'un nœud agrégateur ou un nœud relai en fonction de son mode de déploiement. Dans le cas où la zone de couverture est large, plusieurs nœuds de type Receiver peuvent être déployés. Ces nœuds sont reliés directement à la station de base via un lien radio et la station de base est connectée à un centre de contrôle via un port série.

III.3 Partie logicielle

Dans cette partie, nous présentons les outils logiciels nécessaires pour le développement de notre application.

Pour réaliser cette application nous avons utilisé Ubuntu12.04, TinyOS2.1.1, NesC et NetBeans IDE 6.9.1.

Les capteurs utilisent TinyOS comme système d'exploitation et nous avons installé ce dernier sous Ubuntu. En outre, pour l'implémentation de l'application nous avons adopté deux langages de programmations NesC et java. Le premier est un langage orienté composant pour programmer les dispositifs à ressources limitées tels que les capteurs et le deuxième est un langage orienté objet utilisé pour interpréter et bien exploiter ce qui se passe au centre de contrôle et pour communiquer les données envoyées par les capteurs tels que SerialForwarder et MIG[4].

D'autre part, nous utilisons le langage Java qui contient les librairies nécessaires au développement d'interfaces homme/machine par exemple la création d'interfaces graphiques a besoin de la librairie Swing. La caractéristique principale de ce langage c'est la portabilité de son code exécutable. C'est ce langage qui a été choisi pour cette application.

Pour l'environnement de développement, nous avons utilisé NetBeans qui est un IDE (Integrated Development Environment) modulable et open-source, et qui offre de nombreuses fonctionnalités:

- Il contient un éditeur avec coloration syntaxique suivant le langage choisi.
- Il permet la navigation et la gestion des différents projets et de leurs classes.
- Il contient un debugger et un compilateur.
- Il renomme automatiquement les fonctions/classes/variables dans l'ensemble du code.
- Il permet la complétion lors de la saisie du code.
- Il est compatible avec Windows, Unix et Mac.

Etant simple d'utilisation, très complète et gratuite, le choix de ce logiciel s'est donc fait naturellement.

III.4 Etapes de développement

Dans cette partie nous allons présenter la partie installation logicielle, l'installation matérielle et l'architecture de l'application.

[4] MIG: Message Interface Generator

III.4.1 Installation logicielle

La première étape a été de prendre en main les capteurs, en se familiarisant avec le langage NesC et le système d'exploitation TinyOs. Nous avons choisi l'environnement Linux (Ubuntu) pour la mise en place de l'application de l'agrégation des données. Il est possible de l'exploiter sous cygwin (Windows) mais il y a des soucis avec la deuxième version de TinyOs (TinyOs-2.x) sous cet environnement. La phase d'installation a été la plus délicate. En effet, nous avons commencé par installer TinyOS 2.1.1, NesC et le micro-contrôleur MSP430 pour les capteurs de type TelosB. La procédure d'installation de TinyOs se déroule en plusieurs étapes et elle est décrite dans l'annexe. Puis nous avons installé NetBeans 6.9.1 au niveau du poste de contrôle pour communiquer avec les capteurs et faire visualiser les valeurs des grandeurs remontées à la station de base.

III.4.2 Installation matérielle

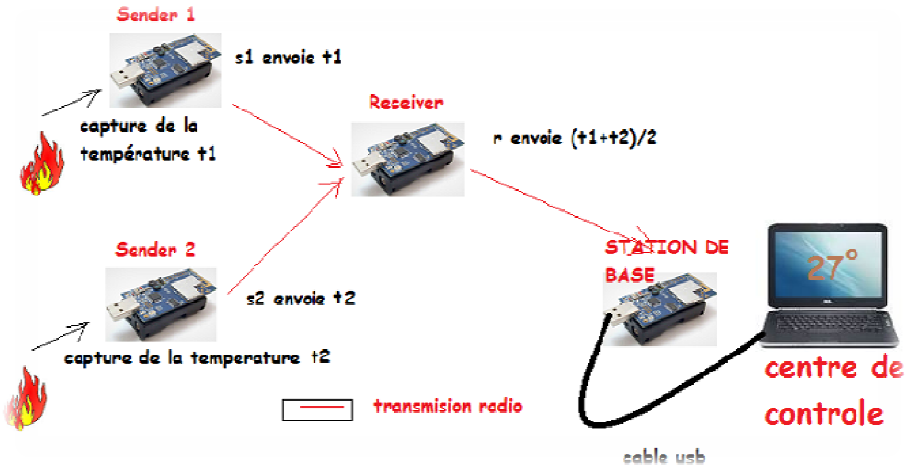
Une fois l'installation logicielle terminée, il a fallu installer le matériel : une station de base reliée à l'ordinateur via un câble USB, trois capteurs TelosB pour la maquette comme s'est illustré dans la figure III.2. Chacun des TelosB communique avec la station de base via une liaison sans fil et la station de base communique avec l'ordinateur via le câble USB**.**

Figure III-2: Environnement du travail

L'architecture de l'application s'articule autour de quatre capteurs TelosB : deux Senders, un Receiver, une station de base et aussi un centre de contrôle représenté par un ordinateur.

Nous avons travaillé avec deux scenarios:

a. Premier scénario



III-3 Scénario du nœud agrégateur.

Les étapes suivantes expliquent le fonctionnement de l'architecture proposée :

- Les Senders mesurent périodiquement les températures dans leurs zones de couverture.
- Ils transmettent les mesures et leurs identifiants au Receiver qui jouera le rôle d'un nœud agrégateur dans l'architecture.
- Le Receiver effectue la moyenne des mesures qu'il a reçu, à son tour il retransmet la valeur de la moyenne à la station de base rattachée au centre de contrôle où réside l'application.
- L'application affiche les températures reçues des Senders et la moyenne reçue du Receiver à chaque arrivée d'un nouveau Sender.

b. Deuxième scénario

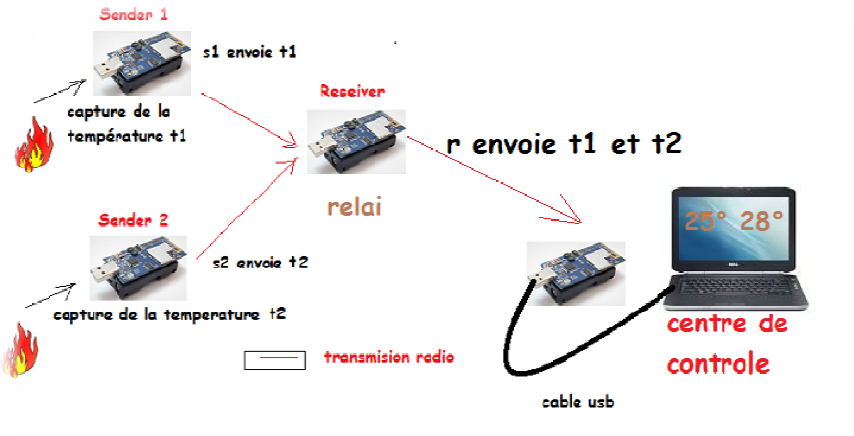


Figure III-4: Scénario du nœud relai

Les étapes suivantes expliquent le fonctionnement de l'architecture proposée:

- Les Senders mesurent périodiquement les températures dans son voisinage.
- Ils transmettent les mesures et leurs identifiants au Receiver qui jouera le rôle d'un nœud relai.
- Le Receiver retransmet chaque mesure reçue à la station de base qui rattachée au centre de contrôle ou réside l'application.
- L'application affiche les températures relayées par le Receiver à chaque fois qu'il reçoit une valeur d'un Sender.

III.4.3 Eléments de l'application

Dans les deux types de capteurs (Sender et Receiver), nous avons la représentation de l'information (température+identifiant) sous forme d'une structure de données qui figure dans le fichier **"Message.h"** comme montre la figure III.5. Ce dernier est incorporé en entête et il contient la structure des messages à envoyer**.**

```
1)#ifndef MESSAGE_H
2)#define MESSAGE_H

3)typedef nx_struct Temperature_Msg{
 4) nx_uint16_t temperature;
5)  nx_uint16_t nodeid;  /* ajouter */
6)} Temperature_Msg;

7)enum {AM_TEMPERATURE_MSG = 6,
    };

8)#endif
```

Figure III-5: Exemple d'un fichier d'entête

Lignes 3-6: La définition de la structure du message que nous allons envoyer. Les types de données sont précédés d'un préfixe nx_. Ce sont des types que nous utilisons uniquement en communication radio et série.

Ligne 7: Type AM de paquet. Nous pouvons demander à un capteur d'envoyer plusieurs types de paquets : paquet de température, paquet d'humidité etc. Pour que le récepteur puisse les différencier et les traiter différemment, il faut donner à chaque paquet une étiquette (le type AM). Le type AM est toujours <AM_> + <NOM DE LA STRUCTURE EN MAJUSCULE>.

1) Sender

Le Sender a les fonctions suivantes:

- Lecture de la température dans son entourage
- L'envoi de cette lecture et son identifiant à la station de base et au Receiver via un lien radio.

2) Receiver

- **Scénario 1**

Le Receiver joue le rôle d'un nœud agrégateur.

- Il reçoit les températures des Senders, les agrège (c'est-à-dire il fait la moyenne des valeurs qu'il a reçue).

- Il retransmet le résultat de l'agrégation à la station de base.

- **Scénario 2**

Le Receiver joue le rôle d'un nœud relai.

- Il reçoit les températures des Senders.

- Il retransmet ces températures à la station de base.

3) Station de base

Reliée au centre de contrôle, contenant un programme prédéfini dans TinyOs.

4) Centre de contrôle

Cet élément exécute une application développée en Java au niveau du centre de contrôle. Nous avons un programme Java qui reçoit et affiche les paquets reçus, mais pour que l'application puisse interpréter les informations reçues, il faut qu'il sache la structure du contenu dans le fichier "**Message.h"** dont nous avons parlé précédemment.

TinyOs fournit l'outil MIG, qui est capable d'analyser un **"fichier.h"** et de convertir toutes les structures en classes Java disposant des méthodes nécessaires qui nous permettent d'accéder aux valeurs des attributs. Nous faisons appel à cet outil depuis le fichier Makefile, dans notre application coté Receiver**.** Son appel prend la syntaxe suivante:

mig java -target=telosb $(CFLAGS) -java-classname=TemperatureMsg /opt/tinyos-2.1.1/apps/essai2/Receiver4/Message.h Temperature_Msg -o $(@)

Maintenant pour afficher les valeurs reçues, nous lançons le programme Java en exécutant la commande suivante en mode console:

Java MsgReader TemperatureMsg –comm serial@/dev/ttyUSB0 :telosb

III.4.4 Programmes en NesC

L'extension d'un fichier **NesC** est "**.nc ".**

Pour le Receiver et le Sender, il existe un module et une configuration, nommés respectivement **ReceiverC** et **ReceiverAppC, SenderC** et **SenderAppC** implémentent les différents composants:

- Main: cœur de l'application.
- ActiveMessage: permet l'accès à la liaison sans fil et l'encapsulation des messages qui pourront être ensuite envoyés via la liaison sans fil.
- SerialActiveMessageC: permet l'accès à la liaison filaire et l'encapsulation des messages qui pourront être ensuite envoyés via la liaison série.

- Leds : permettent de suivre le déroulement du programme en allumant les Leds du capteur à des moments pertinents.
- TimerMilli : permet le déclenchement d'évènements périodiques.
- SensirionSht11 : lit la température et l'humidité.
- AMSender : permet l'envoi des messages.
- Le composant TimerMilli déclenche chaque seconde un évènement. Cet événement demande au composant SensirionSht11 de lire et calculer la température puis d'encapsuler cette valeur à l'aide du composant ActiveMessage et de l'envoyer via le composant.

III.5 Exemples d'exécution

Dans ce qui suit, nous présentons les résultats obtenus en illustrant d'une part l'agrégation de données et d'autre part l'évaluation des performances dans les deux contextes : contexte à nœud agrégateur et le contexte à nœud relai.

III.5.1 Première partie

Dans cette partie, l'application consiste à afficher les valeurs des températures reçues des Senders et la valeur agrégée de ces températures. Après l'exécution nous avons obtenu l'affichage de la température captée (29°) par le seul Sender qui est déployé et qui a 2 comme identifiant. Cette valeur est la même envoyée par le nœud agrégateur comme le montre la figure III.6.

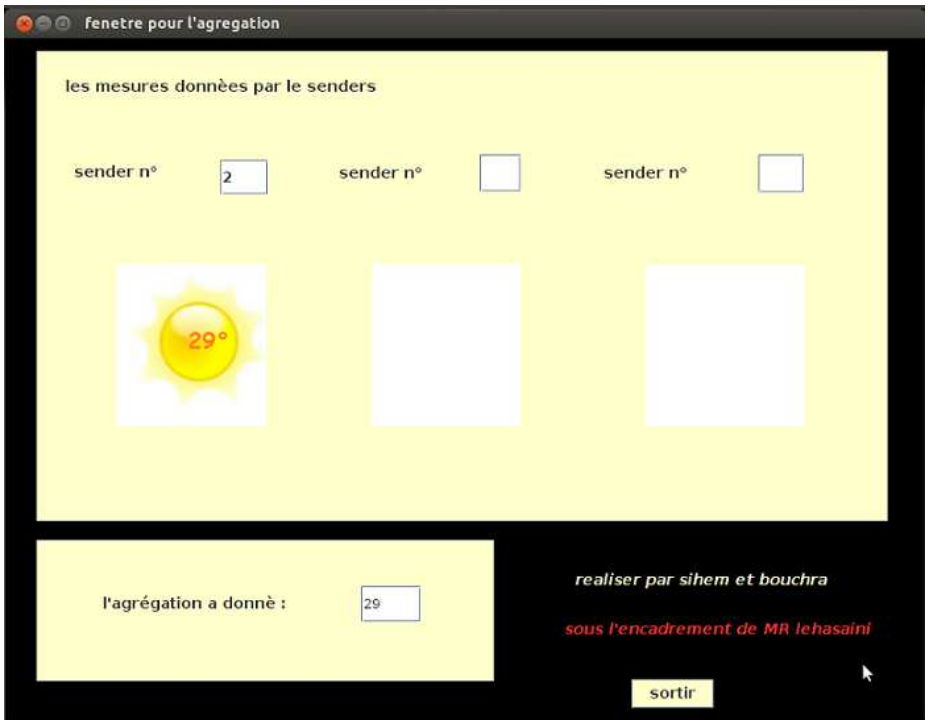


Figure III-6: Déploiement d'un seul Sender

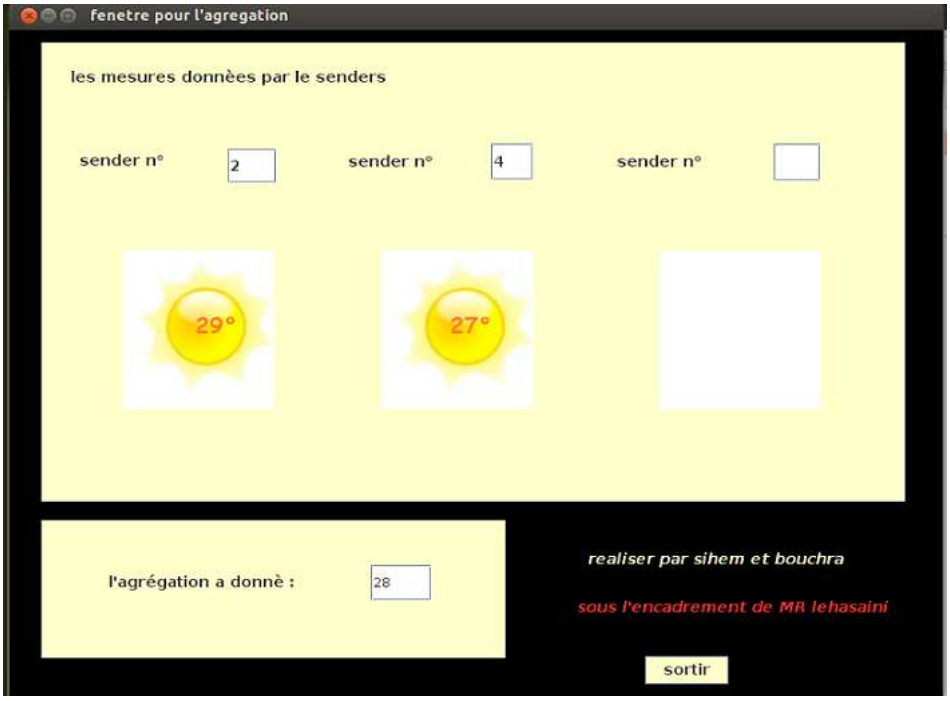


Figure III-7: Déploiement de deux Senders

Dans la figure III.7 un deuxième Sender dont l'identifiant 4 est devenu actif. De ce fait, nous assistons à l'envoi d'une valeur agrégée résultante de la moyenne des deux valeurs de températures captées par les deux Senders.

Valeur agrégée = (29°+27°) /2 qui donne 28° comme le montre la figure III.7.

Figure III-8: Déploiement du troisième Sender

Pour la figure III.8, un troisième Sender dont l'identifiant 1 devient actif. Ce dernier détecte la température 26°, par suite nous assistons à un nouvel envoi d'une valeur agrégée résultante de la moyenne des trois valeurs de températures captées par les trois senders.

Valeur agrégée = (29°+27°+26°) /3 qui donne 27° comme le montre la figure III.8.

III.5.2 Deuxième partie

Pour illustrer l'apport de l'agrégation des données en termes de consommation d'énergie, nous avons mis en place deux scénarios. Dans le premier, le Receiver joue le rôle d'un nœud agrégateur et dans le second scénario le nœud Receiver relaye tout message reçu de la part des Senders déployés dans son voisinage. Or, dans les RCSF, la transmission et la réception sont les opérations qui consomment plus d'énergie. De ce fait, pour évaluer la quantité d'énergie consommée dans les deux scénarios, il suffit de compter le nombre de paquets émis et reçus durant la durée de vie du réseau.

Dans les deux scénarios, nous avons utilisé deux Senders qui envoient des paquets au Receiver après chaque seconde.

a. Scénario 1

Comme nous avons parlé précédemment, ce scénario décrit le fonctionnement du nœud agrégateur.

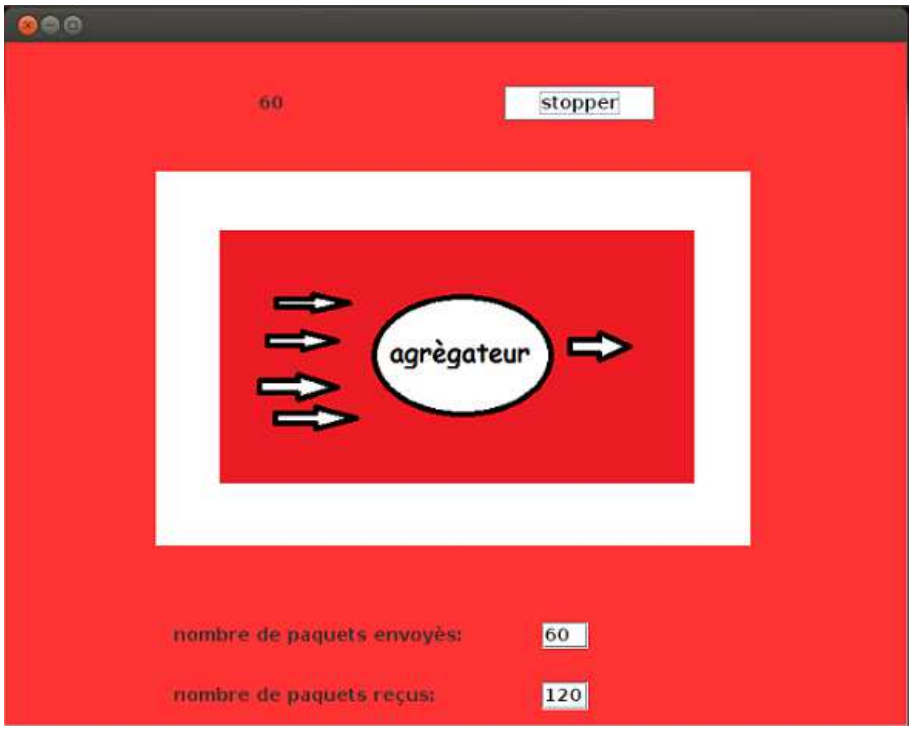


Figure III-9: Agrégateur état 1

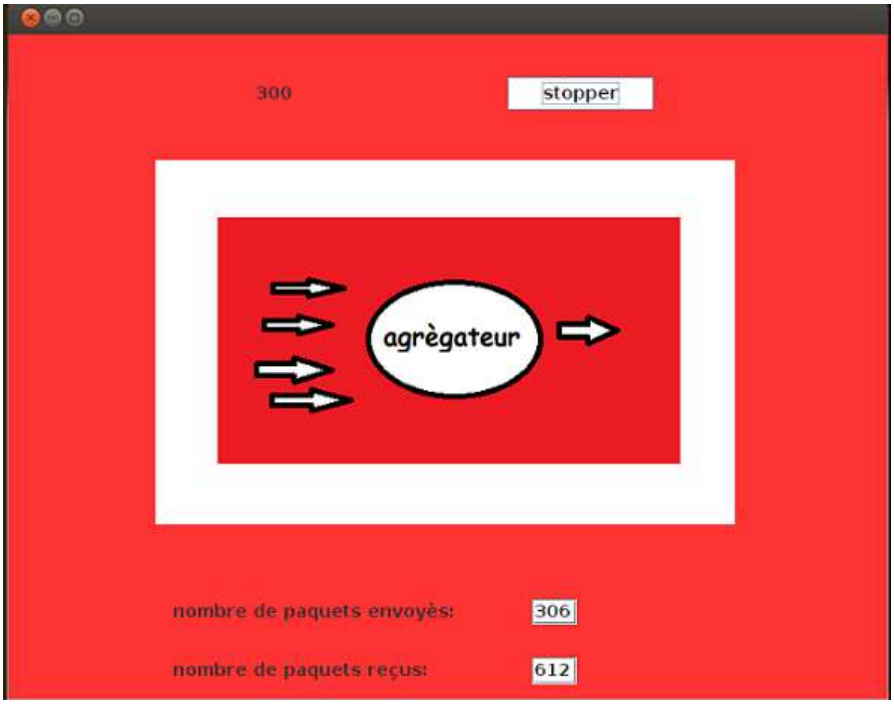


Figure III-10: Agrégateur état 5

Les figures III.9 et III.10 représentent respectivement l'état de l'exécution après 60 secondes, et 5 minutes.

Le tableau III.1 résume les résultats obtenus dans le premier scénario.

Tableau III-1 : Résultats du premier scénario

Temps (Minutes)	1	2	3	4	5
Nbre de paquets	60	121	183	244	306
Nbre de paquets reçus	120	243	366	489	612

La première ligne du tableau III.1 représente le temps durant lequel nous avons supervisé la zone d'intérêt, la deuxième le nombre de paquets transmis des Senders au nœud agrégateur, et la troisième le nombre de paquets transmis du nœud agrégateur à la station de base.

Nous remarquons que le nombre de paquets reçus est le double du nombre de paquets envoyés cela mène à éviter la redondance des informations et la minimisation de l'énergie au niveau du nœud agrégateur. En outre, si nous faisons référence à l'idée citée dans l'introduction qui dit que la transmission des données représente environ 70% de la consommation d'énergie au niveau d'un capteur nous trouvons que l'agrégation permet de minimiser l'énergie.

b. Scénario 2

Ce scénario décrit le fonctionnement du nœud relai. La figure III.11 représente l'état de l'exécution après 1 minute.

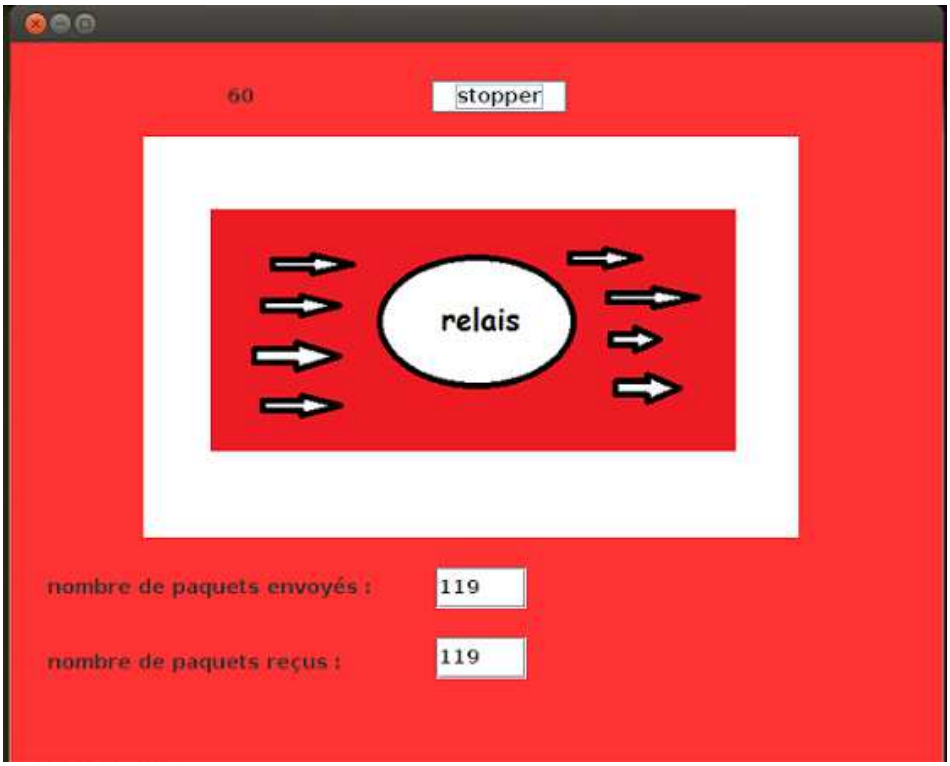


Figure III-11: Relais état 1

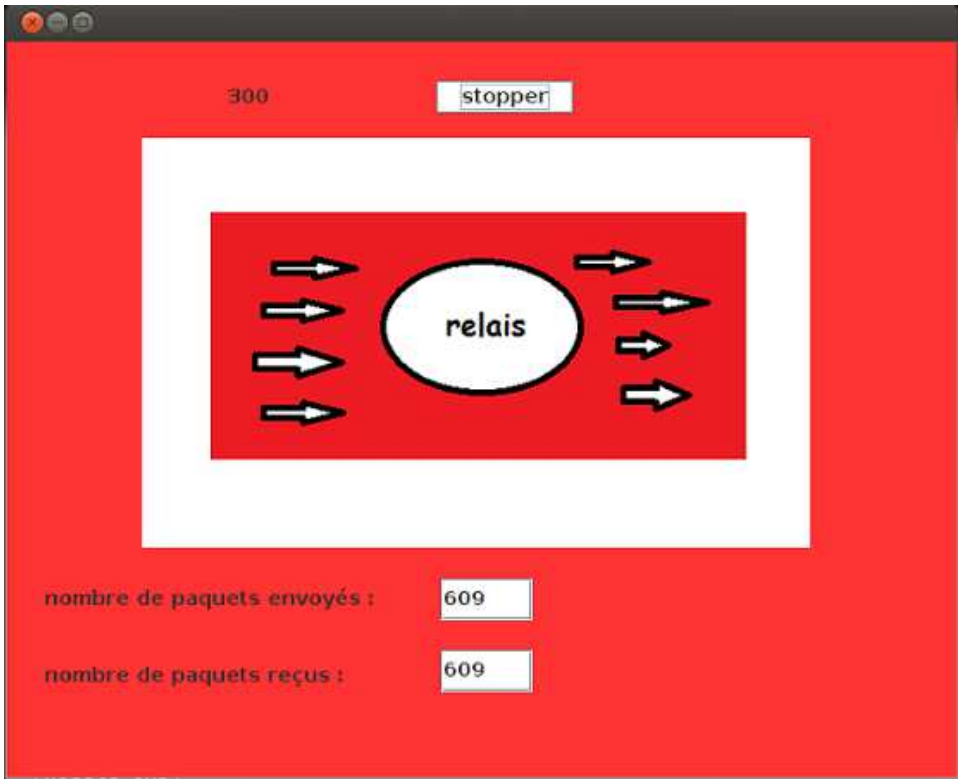


Figure III-12: Relais état 5

La figure III.12 représente l'état de l'exécution après 5 minutes.

Le tableau III.2 illustre les résultats obtenus dans le deuxième scénario.

Tableau III-2: Résultats du scénario 2

Temps (Minutes)	1	2	3	4	5
Nbre de paquets	119	241	363	487	609
Nbre de paquets reçus	119	241	363	487	609

La première ligne de ce tableau représente le temps, la deuxième le nombre de paquets transmis des Senders au nœud relai, et la troisième le nombre de paquets transmis du nœud relai à la station de base.

Nous remarquons que le nombre de paquets reçus est égal au nombre de paquets envoyés.

c. Interprétation

Dans cette partie, nous illustrons la différence entre les deux scénarios par un graphe en prenant les résultats présentés dans les tableaux III.1 et III.2.

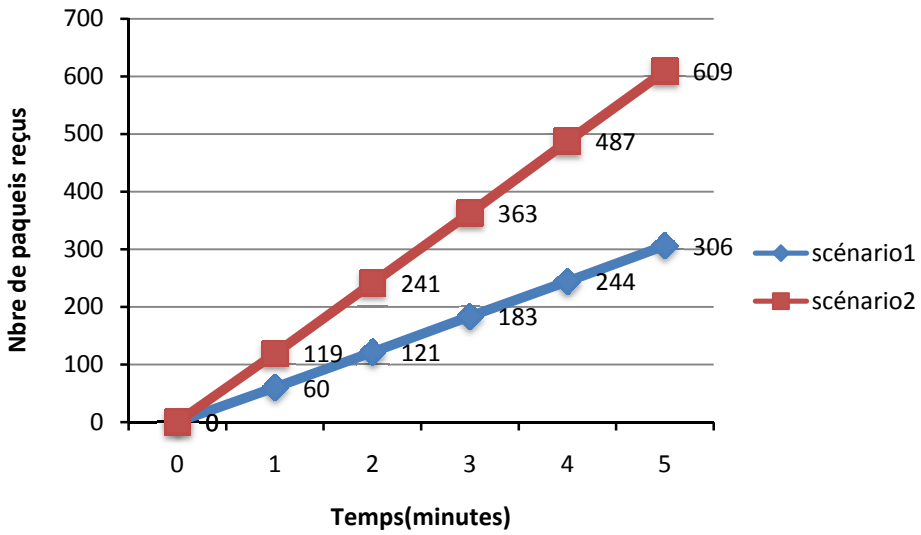


Figure III-13: Nbre de paquets reçus en fonction du temps

La figure III.13 illustre l'apport de l'agrégation de données dans les RCSF. Nous remarquons que le nombre de paquets est réduit quand nous impliquons des nœuds agrégateurs dans un RCSF comparativement au scénario dans lequel nous avons utilisé des nœuds relais.

Ces résultats symbolisent que la consommation d'énergie est minimisée dans le cas où des nœuds agrégateurs sont déployés. Cette constatation est faite sur la base que

la consommation d’énergie est calculée en fonction de la quantité d’informations échangées.

III.6 Conclusion

Dans ce chapitre, nous avons présenté la démarche à suivre pour réaliser une application qui permet de mettre en valeur l’agrégation des données dans les RCSF. Le développement de cette application nécessite des outils logiciels bien particuliers tels qu’un système d’exploitation léger "TinyOs" et un langage orienté composant "NesC".

Pour montrer cet apport, nous avons utilisé deux scénarios différents. Dans le premier, nous avons impliqué des nœuds agrégateurs alors que dans le deuxième nous avons intégré des nœuds relais.

Conclusion générale

Ce projet nous a appris la manière de mener les projets et les différentes façons de traiter les problèmes rencontrés. De plus nous avons acquis des connaissances dans les domaines techniques et nous avons appris à tirer le meilleur de la théorie qui existe afin d'atteindre notre but. En plus de tout ce que nous avons appris, nous avons développé notre sens du travail collectif et le développement des idées ce qui nous facilitera l'immersion dans le domaine professionnel.

Ce projet nous a permis d'acquérir des connaissances en programmation évènementielle. Il nous a aussi fait découvrir un nouveau langage de programmation, le NesC ainsi que la plateforme de programmation adéquate qui est TinyOs.

Dans notre projet, nous nous sommes intéressés à mettre en place une application de l'agrégation de données consistant à calculer la moyenne, ce qui nous a permis d'élargir nos idées pour évaluer deux scénarios en terme de la consommation d'énergie qui est proportionnelle au nombre de paquets au niveau des nœuds intermédiaires.

En perspectives, nous proposons de mettre en place une application d'agrégation consistant à compresser les métadonnées ce qui permet d'optimiser l'allocation de la mémoire. Nous pouvons aussi mettre en place une application de fusion des données à grande échelle permettant une tolérance aux pannes au niveau des nœuds capteurs.

Références bibliographiques

[1] Olivier Français, "Capteurs et électronique associée", Cours présenté à l'école d'ingénieur ESIEE, 2000.

[2] Vernon S. Somerset, "Intelligent and Biosensors, Edited by Vernon S. Somerset", Intech, January 2010.

[3] F. Brissaud, D. Charpentier, A. Barros et C. Bérenguer, "Capteurs intelligents : nouvelles technologies et nouvelles problématiques pour la sûreté de fonctionnement", Maîtrise des Risques et de Sûreté de Fonctionnement, Lambda-Mu 16, Avignon : France (2008).

[4] Yacine Younes, "Minimisation d'énergie dans un réseau de capteurs", mémoire de magistère, université Mouloud Mammeri de Tizi-Ouzou, Septembre 2012.

[5] Cristian Duran-Faundez, "Transmission d'images sur les réseaux de capteurs sans fil sous la contrainte de l'énergie", Université Henri Poincaré Nancy 1, Juin 2009.

[6] Moad Sofiane, "Optimisation de la consommation d'énergie dans les réseaux de capteurs sans fil", Master: Recherche 2 en Informatique, Université : IFSIC-Rennes 1 , Année universitaire : 2007/2008 .

[7] Khaled Bouchakour, "Routage hiérarchique sur les réseaux de capteurs sans fil: Protocole KhLCH (K-hop Layered Clustering Hierarchy)", Mémoire de Magistère, ESI Algérie, 2012.

[8] www.techno-science.net/?onglet=glossaire&definition=3690, Juin 2013.

[9] M. Badet, W. Bonneau, "Mise en place d'une plateforme de test et d'expérimentation", Projet de Master Technologie de l'Internet 1ere année, Université Pau et des pays de l'Adour, 2006.

[10] Mohamed Lehsaini, "Diffusion et couverture basées sur le clustering dans les réseaux de capteurs : application à la domotique", Thèse de Doctorat, Université A.B Tlemcen && Université de Franche-Comté, 2009.

[11] Mohamed Aissani, "Optimisation du routage dans les routage réseaux de capteurs pour les applications temps réel", Thèse de Doctorat, Université des Sciences et de la Technologie Houari Boumediene && Université Paris-Est, Mars 2011.

[12] Xue Yong, Gonzalez Andres, Aguilar Andres, Barroux Mickaël, "Agrégation de données dans les réseaux de capteurs", Rapport final, Université de Technologie Compiègne, Automne 2010.

[13]wapiti.telecom.lille1.eu/commun/ens/peda/options/st/rio/pub/exposes/exposesrio2007-ttnfa2008/Maurin-Guillory/capteurs.htm, Réalisation Sébastien Guillory et Aurélien Maurin, Juin 2013.

[14] Dhib Aya, "Routage avec QOS temps réel dans les réseaux de capteurs", Master, Ecole supérieure de communication de TUNIS, 2006-2007.

[15] A.Bharathidasan, V.Anad Sau Ponduru, "Sensor networks : An overview", département d'informatique de Californie.

[16] Yasser Romdhane, "Evaluation des performances des protocols S-MAC et directed diffusion dans les réseaux de capteurs", rapport de projet de fin d'études, école supérieure des communications de Tunis, 2007.

[17] A. Delye, V. Gauthier, M. Marot, and M. Becker. "Etat de l'art sur les réseaux de capteurs", Rapport de Recherche INT N-05001RST GET-INT, UMR5157 SAMOVAR, Institut National des Télécommunications, Evry, France, 2005.

[18] W. Heinzelman, A. Chandrakasan, H. Balakrishnan, "Energy-Efficient Communication Protocol for Wireless Micro sensor Networks", In proc of the Hawaii International Conference on Systems Science, vol. 8, pp. 8020, January 2000.

[19] blog.memsic.com/TelosB/, Juin 2013.

[20] Bouzidi Zeyneb, Benameur Amina, "Mise en place d'un réseau de capteurs sans fil pour l'irrigation intelligente", Master en Informatique, Université Abou Bakr Belkaid– Tlemcen, 2012.

[21] Gwenhael Goavec, "Etude d'un système d'exploitation pour microcontrôleur à faible consommation", juin 2010.

[22] TinyOS Team, "Tinyos." [Online]. Available: www.tinyos.net , Juin 2013.

[23] UC Berkeley, "nesc : A programming language for deeply networked systems" [Online]. Available: nescc.sourceforge.net, Juin 2013.

[24] EL Mehdi Damou, "Simulation d'un réseau de capteurs avec TinyOS", article, France.

[25] Douma Nabila, "la localisation dans un réseau de capteurs", projet de première année master RTM, 2010.

[26] Douma Nabila, Elhammouti Mohamed, "La localisation dans un réseau de capteurs", projet, Université d'Avignon et des Pays de Vaucluse .

[27] www.tinyos.net/tinyos-1.x/doc/ , mai 2013.

[28] Selatna Abbes, "Implémentation d'une application orientée surveillance pour les réseaux de capteurs", Master en Informatique, Université Abou Bakr Belkaid– Tlemcen, Juillet 2012.

[29] Nicolas Esteves, "Capture de Mouvement par Réseau de Capteurs sans Fil", rapport de stage, Juillet 2007.

[30] David Martins, "Sécurité dans les réseaux de capteurs sans fil Stéganographie et réseaux de confiance", Thèse de Doctorat, Université de Franche-Comté, Novembre 2010.

[31] Shah Bhatti, James Carlson, Hui Dai, Jing Deng, Je_ Rose, Anmol Sheth,Brian Shucker, Charles Gruenwald, Adam Torgerson, and Richard Han. "Mantis os: an embedded multithreaded operating system for wireless micro sensor platforms", Mob. Netw. Appl., 10(4) :563_579, 2005.

[32] FreeRTOS. In Http: //www.freertos.org/ ,Juin 2013.

[33] Adam Dunkels, Bjorn Gronvall, and Thiemo Voigt. "Contiki - a lightweight and flexible operating system for tiny networked sensors". In LCN '04: Proceedings of the 29th Annual IEEE International Conference on Local Computer Networks, pages 455-462, Washington, DC, USA, 2004. IEEE Computer Society.

[34] Chih-Chieh Han, Ram Kumar, Roy Shea, Eddie Kohler, and Mani Srivastava. "A dynamic operating system for sensor nodes". In MobiSys '05: Proceedings of the 3rd international conference on Mobile systems, applications, and services, pages 163_176, New York, NY, USA, 2005. ACM.

[35] Nut/OS. In Http: //www.ethernut.de/en/software/index.html, Juin 2013.

[36] Mickael Cartron, "Vers une plate-forme efficace en énergie pour les réseaux de capteurs sans fil", Thèse de Doctorat, Université de Rennes 1, Décembre 2006.

[37] Mathieu Badet, Willy Bonneau, "Mise en place d'une plateforme de test et d'expérimentation", Master Technologie de l'Internet, Université de Pau et des pays de l'Adour, 2005/2006.

[38] Noureddine Lasla, "La gestion de clés dans les réseaux de capteurs sans-fil", Magistère, Oued-Smar, Alger, 2006 – 2007.

[39] Yacine Challal, Hatem Bettahar, Abdelmadjid Bouabdallah, "Les Réseaux de capteurs (WSN: Wireless Sensor Networks)", Cours1, Université de Technologie de Compiègne, FRANCE.

Annexe

Installation de TinyOs 2.1.1 sous Linux

- Pour pouvoir modifier le fichier sources.list ouvrez le terminal et exécutez :

$ sudo gedit /etc/apt/sources.list

- Ajoutez cette ligne à la fin du fichier

deb http://tinyos.stanford.edu/tinyos/dists/ubuntu karmic main

karmic pour toute les distribution d'ubuntu

- ouvrez le fichier **.bashrc**

$ gedit ~/.bashrc

- Et copiez ces linges dans **.bashrc**

```
export TOSROOT=/opt/tinyos-2.1.1
export TOSDIR=$TOSROOT/tos
export CLASSPATH=$TOSROOT/support/sdk/java/tinyos.jar:.:$CLASSPATH
export MAKERULES=$TOSROOT/support/make/Makerules
export PATH=/opt/msp430/bin:$PATH
#Sourcing the tinyos environment variable setup script
source /opt/tinyos-2.1.1/tinyos.sh
```

Programmation

Branchez le capteur TelosB à votre ordinateur avec un câble USB et ouvrez une console sous UBUNTU. Avant de compiler et d'installer le programme, il faut tester si le capteur a été bien détecté et reconnu par le système. Entrez la commande suivante motelist pour lister tous les capteurs branchés :

Liste figures

Liste des tableaux

Liste des abréviations

RCSF : Réseaux de Capteurs Sans Fil.

GPS : Global Positioning System.

CAN : Convertisseur Analogique-Numérique.

TinyOS : Tiny Operating System.

NESC : National Electrical Safety Code.

LED : Light-Emitting Diode.

TCP : Transmission Control Protocol.

IP : Internet Protocol.

RAM : Random Access Memory.

UC Berkeley : University of California Berkeley.

IEEE : Institute of Electrical and Electronics Engineers.

DARPA : Defense Advanced Research Projects Agency.

DMA : Direct Memory Access .

BSD : Berkeley Software Distribution.

FIFO : First In First Out.

HPL : Hardware Presentation Layer.

HAL : Hardware Adaptation Layer.

HIL : Hardware Interface Layer.

SOS : Sensor node Operating System .

IDE : Integrated Development Environment

Résumé **L'apport de l'agrégation de données dans les réseaux de capteurs**

Dans le cadre de notre projet, nous avons implémenté une application sur des capteurs réels permettant l'agrégation des données. Pour se faire, nous avons testé notre application sur une plateforme contenant un ensemble de Senders, un Receiver et une station de base. Le nœud Sender mesure la température et l'envoie périodiquement au Receiver qui jouera le rôle d'un nœud agrégateur. En outre, nous avons mis en place une autre plateforme dans laquelle est intégré un nœud relai. Puis, nous avons évalué le nombre de paquets dans les deux scénarios pour illustrer l'apport de l'agrégation de données.

Mots clés : RCSF, Agrégation des données, Nœud agrégateur, Nœud relai, TinyOS, NesC.

Abstract **Contribution of data aggregation in wireless sensor networks**

In our project, we implemented an application on real sensors for data aggregation. We tested our application on a platform containing a set of Sender nodes, a Receiver node and a base station. Sender node measures temperature and periodically transmits it to the receiver node which is an aggregator node. Moreover, we have implemented another platform in which we involve a relay node. Then, we evaluated the number of packets in the two scenarios to illustrate the contribution of data aggregation.

Keywords: WSN, Data aggregation, Aggregator node, Relay node, TinyOS, NesC.

تلخيص **قيمة تجميع البيانات في أجهزة الاستشعار**

في إطار مشروعنا، قمنا بتنفذ تطبيق على أجهزة استشعار حقيقية لتجميع البيانات. وللقيام بذلك، اختبرنا التطبيق باستعمال مجموعة من أجهزة استشعارتقوم بإرسال المعلومات (درجة الحرارة) جهاز استقبال ومحطة قاعدة. يقيس جهاز الاستشعار درجة الحرارة ويرسلها بشكل دوري إلى جهاز الاستقبال الذي يقوم بتجميعها، وبالإضافة إلى ذلك، قمنا بتنفيذ تطبيق آ خر يحتوي على جهاز يقوم بتوصيل المعلومة إلى محطة المراقبة . ثم، قمنا بتقييم عدد الحزم في االسيناريوهات اثنين لتوضيح قيمة تجميع البيانات.

الكلمات المفتاحية: أجهزة الاستشعار, تجميع المعلومات ,جهاز تجميع ,جهاز توصيل TinyOs,Nesc,