Université Abou Bekr Belkaid جامعة ابي بكر بلقايد
تلمسان الجزائر

République Algérienne Démocratique et Populaire
Université Abou Bekr 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 (RSD)

Thème

Mise en œuvre d'un réseau de capteurs sans fil pour la détection des feux de forêt

Réalisé par :

- **ARSAOUI Omar**
- **SENOUCI BEREKSI Djazia**

Présenté le 2 Juillet 2012 devant le jury composé de MM.

- *Mr B. BENMAMMAR* *(Président)*
- *Mme N. LABRAOUI* *(Encadreur)*
- *Mr M. LEHSAINI* *(Examinateur)*
- *Mme A. BELHABI* *(Examinateur)*

Année universitaire : 2011-2012

Mise en œuvre d'un réseau de capteurs Sans fil pour la détection des feux de forêt

Auteurs : ARSAOUI Omar & SENOUCI BEREKSI Djazia

Encadreur : Mme N. LABRAOUI

Il faut savoir ce que l'on veut; quand on le sait, il faut avoir le courage de le dire ; quand on le dit, il faut avoir le courage de le faire.

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.

Nous tiendrons à remercier en premier lieu notre encadreur Mme LABRAOUI Nabila Chercheur au laboratoire STIC et maître assistant à l'université **A**. **B** de Tlemcen pour ses encouragements, sa disponibilité, ses idées, ses conseils et sa sympathie qui nous ont permis de mener à bien ce mémoire.

Nous adressons aussi nos très sincères remerciements à Mr BENMAMMAR pour l'honneur qui nous a été accordé en présidant le jury. Nous tiendrons à remercier également Mr LEHSAINI et Mme BELHABI qui ont bien voulu examiner et évaluer notre travail et qui nous font l'honneur de participer à la soutenance.

Un grand MERCI à tous les enseignants du département d'Informatique de l'université Abou Bekr Belkaid qui nous ont formées durant ces cinq dernières années.

Nos remerciements vont aussi à tous ceux qui ont contribué de près ou de loin à la concrétisation de ce travail. Qu'ils trouvent tous ici l'expression de notre gratitude et notre parfaite considération.

DEDICACES

La vie n'est qu'un éclair
Et un jour de réussite est un jour très cher.

A mes très chers parents : Pour l'éducation et l'amour dont ils m'ont entouré depuis ma naissance. Ainsi que pour leurs patience et leurs sacrifices et a qui je dois toute ma réussite.

A mon bien-aimé Mohamed pour son soutien sa compréhension et ses encouragements.

A ma chère grande sœur Imane ainsi qu'à son mari Redouane et à leur adorable petite fille Nourhane ;

A ma très chère petite sœur Yousra à qui je souhaite de la réussite dans tout ce qu'elle accomplira ;

A ma belle famille khiba yesmine yassine et sa femme pour leurs soutient.

A Mon binôme et ami Omar pour avoir établi avec moi ce projet a sa fin.

A tous mes proches et ceux qui m'aiment ;

Je dédie ce travail

Djouzia

DEDICACES

Avec l'aide du Dieu le tout puissant j'ai pu achever ce modeste travail que je dédie

A mes très chers parents qui m'ont tout appris qui m'ont aidé encouragé soutenu et surtout aimé.

A mes frères et mes soeurs

A toute la famille de prés et de loin

A tous ceux qui m'aiment;

A tous mes amis

A tous ceux que j'aime.

Et enfin à toute personne qui reconnaîtra son empreinte sur ce modeste travail

Je dédie ce travail..

Que la paix d'Allah soit avec tous.. que Dieu nous réunisse dans son vaste paradis incha Allah

Omar

Table des matières

Liste des figures

Liste des tableaux

Introduction générale

Depuis quelques décennies, le besoin d'observer, de surveiller à distance et de récupérer les données d'un environnement complexe et distribué s'accroît rapidement surtout avec les récentes avancées dans le domaine de la micro-électronique de la micromécanique, et des technologies de communication sans fil. Dans ce constat et pour répondre à ces attentes, une nouvelle branche s'est créée donnant ainsi l'apparition des réseaux de capteurs. Ces réseaux forment un type particulier des réseaux Ad Hoc, dans lesquels les nœuds sont des capteurs. Dans ce type des réseaux, les capteurs échangent l'information sur l'environnement afin d'établir une vue globale de la région surveillée.

Les capteurs apparaissent comme des systèmes autonomes miniaturisés, équipés d'une unité de traitement et de stockage de données, d'une unité de transmission sans fil et d'une batterie. Organisés sous forme de réseau, les capteurs (ou nœuds) d'un RCSF, malgré la limitation de leurs ressources (calcul, capacité de stockage et énergie), ont pour mission de détecter un événement (par exemple, un changement de température, des mouvements, des vibrations, ...). Un RCSF est capable de récolter des données relatives à son environnement, de les traiter, puis, si nécessaire, de les communiquer à des capteurs voisins via un médium sans fil.

Les caractéristiques de ces capteurs ont participé à la naissance des réseaux de capteurs Sans fil, et de favoriser leur utilisation dans une multitude d'applications. Cette nouvelle technologie exige l'auto-organisation, la collaboration et le fonctionnement autonome de capteurs. De plus, elle nécessite généralement un déploiement dense pour permettre une meilleure garantie de surveillance dans des environnements principalement dangereux et hostiles, dans lesquels les capteurs sont continuellement exposés à des menaces intentionnelles ou non intentionnelles très importantes.

Les travaux récents ont montré la nécessité d'utiliser ces réseaux pour la détection des incendies. Toutefois, pour signaler la présence de ces derniers, il faut s'assurer de mesurer les bons indices, tels que la température, l'humidité et l'intensité lumineuse. La couverture forestière partout en Algérie a été ces cinquante dernières années le théâtre d'un grand massacre avec une tendance globale d'augmentation du nombre des feux et des surfaces brûlées .Sa superficie estimée à 1.3 millions d'hectares, de vraies forêts naturelles

ont connu une régression quasi exponentielle, et se trouvent aujourd'hui dans un état de dégradation avancé. Nulle part ailleurs, la forêt n'apparaît aussi nécessaire à la protection contre les incendies. La lutte contre les feux de forêt en Algérie demande chaque année un investissement important, tant en hommes qu'en matériel et moyens financiers. Parmi ces efforts nous citons le projet de recherche national (PNR) proposé en Algérie pour la surveillance environnementale, « **Mise en œuvre d'un réseau de capteurs sans fil pour la détection des feux de forêt**» qui est sponsorisé par l'agence nationale de recherche au sein du laboratoire STIC de l'université de Tlemcen en collaboration avec des gardes forestiers. Ce projet a pour but de développer et de déployer un réseau de capteurs pour détecter les feux de forêt dans un parc d'attraction.

Pour notre part, nous nous intéressons à l'utilisation d'une nouvelle méthode pour la lutte contre les feux de forêt en Algérie, qui consiste à implémenter un réseau de capteurs sans fil capable de recueillir des données provenant de l'environnement. De manière plus particulière, nous avons choisi d'utiliser un RCSF capable de détecter la présence d'un incendie. L'objectif principal de cette recherche est de réaliser la conception d'un nouvel modèle de détection de feu de forêt, implémenter une partie de l'application et effectuer une évaluation de performance en simulant l'application avec le simulateur TOSSIM sous le système d'exploitation TinyOs.

Ce mémoire s'articule autour de quatre chapitres.

Le chapitre 1 : nous présentons une description générale des réseaux de capteurs sans fil ainsi que leurs caractéristiques, contraintes et spécificités.

Le chapitre 2 : est présenté en deux parties fondamentales, la première partie concerne le profil forestier en Algérie, les statistiques, les méthodes de lutte et les conséquences des incendies. Dans la deuxième partie on a proposé une nouvelle méthode pour la prévention contre les incendies en Algérie et qui ce base sur les réseaux de capteurs sans fil.

Le chapitre 3 : présente une description pour le système TinyOS et le langage NesC.

Le chapitre 4 : présentera la mise en place du réseau de capteur sans fil proposé. En particulier, les différentes étapes de l'implémentation et du déroulement de notre application. Par la suite nous donnons les résultats sous forme de graphes de quelques simulations, effectuées pour obtenir des mesures pour évaluer les performances de notre application.

Chapitre I

Etat de l'art sur les réseaux de capteurs sans fil

1. Introduction

Les réseaux de capteurs sans fil sont un cas particulier des réseaux sans fil sans infrastructure (réseaux ad hoc). En effet, ceux-ci sont constitués d'un ensemble de petits appareils, ou capteurs, possédant des ressources particulièrement limitées mais qui leur permettent d'acquérir des données sur leur environnement immédiat, de les traiter et de les communiquer. En effet, un réseau de capteurs peut être mis en place dans le but de surveiller une zone géographique plus ou moins étendue pour détecter l'apparition de phénomènes ou mesurer une grandeur physique (température, humidité, pression, vitesse...).

Nous présentons dans ce chapitre le concept des réseaux de capteurs sans fil ainsi que leurs applications et leurs spécificités intrinsèques.

2. Les réseaux sans fil

Un réseau sans fil (en anglais wireless network) est, comme son nom l'indique, un réseau dans lequel au moins deux terminaux peuvent communiquer sans liaison filaire (via des ondes radiofréquences ou infrarouges). Grâce aux réseaux sans fil, les utilisateurs peuvent accéder à l'information indépendamment de leurs positions géographiques. Les terminaux du réseau (appelés généralement nœuds) se déplacent librement, tandis que le système doit assurer toutes les fonctionnalités et tous les services d'un réseau classique.

Il y a deux manières pour classer les réseaux sans fil. La première est basée sur la zone de couverture du réseau. Cette classification donne lieu à quatre catégories de réseaux sans fil: les réseaux personnels, les réseaux locaux, les réseaux métropolitains et les réseaux étendus. Dans chaque catégorie sont appliquées des technologies de communication qui se démarquent par le débit, la plage de fréquence utilisée et la portée de communication. La deuxième façon de voir les réseaux sans fil est selon le modèle et l'infrastructure de communication adoptée [AH07].

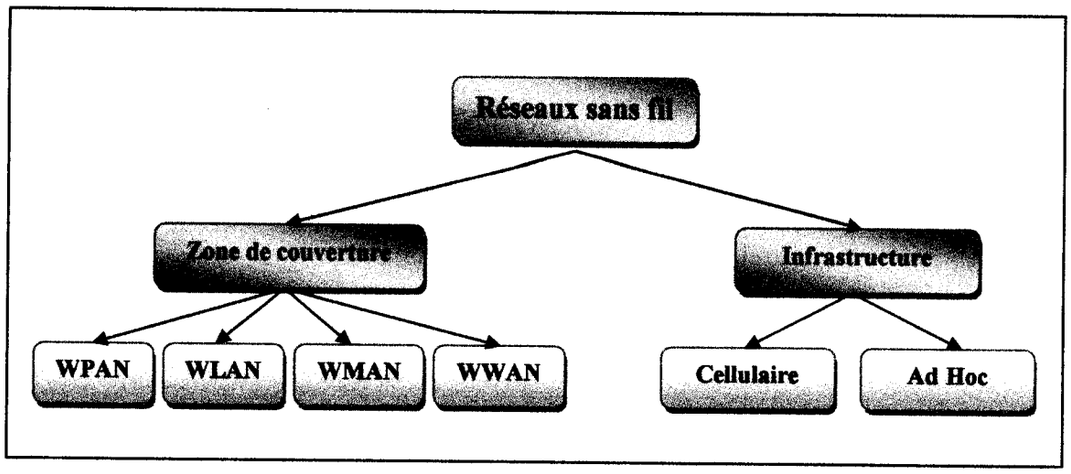


Figure 1. 1: Classification des réseaux sans fil.

- **Réseaux personnels sans fil (WPAN)**

Le réseau personnel sans fil noté WPAN pour Wireless Personal Area Network, concerne les réseaux sans fil d'une faible portée: de l'ordre de quelques dizaines de mètres. Ce type de réseau sert généralement à relier des périphériques (imprimante, téléphone portable, appareils domestiques,...) ou un assistant personnel (PDA: Personal Digital Assistant) à un ordinateur sans liaison filaire ou bien à permettre la liaison sans fil entre deux machines très peu distantes.

- **Réseaux locaux sans fils (WLAN)**

Le réseau local sans fil noté WLAN pour Wireless Local Area Network, est un réseau permettant de couvrir l'équivalent d'un réseau local d'entreprise, soit une portée d'environ une centaine de mètres. Il permet de relier les terminaux présents dans la zone de couverture entre eux.

- **Réseaux métropolitains sans fil (WMAN)**

Le réseau métropolitain sans fil noté WMAN pour Wireless Metropolitan Area Network, est connu sous le nom de Boucle Locale Radio (BLR). Les WMAN sont basés sur la norme IEEE802.16. La boucle locale radio offre un débit utile de 1 à 10 Mbit/s pour une portée de 4 à 10 kilomètres et utilise les bandes de fréquences entre 2,4 GHz et 3,5 GHz, ce qui destine principalement cette technologie aux opérateurs de télécommunication.

- **Réseaux étendus sans fil (WWAN)**

Le réseau étendu sans fil noté WWAN pour Wireless Wide Area Network, est également connu sous le nom de réseau cellulaire mobile. Il s'agit des réseaux sans fil les plus répandus puisque tous les téléphones mobiles sont connectés à un réseau étendu sans fil

2.1. Réseaux cellulaires (avec infrastructure)

Un réseau sans fil cellulaire supprime certaines liaisons filaires entre les entités du réseau et substitue le mode de connexion filaire par une technologie nouvelle basée sur les ondes radioélectriques. Ce mode impose de fixer des bornes pour délimiter une région appelée zone de couverture. La figure suivante schématise les principales caractéristiques des réseaux cellulaires (avec infrastructure).

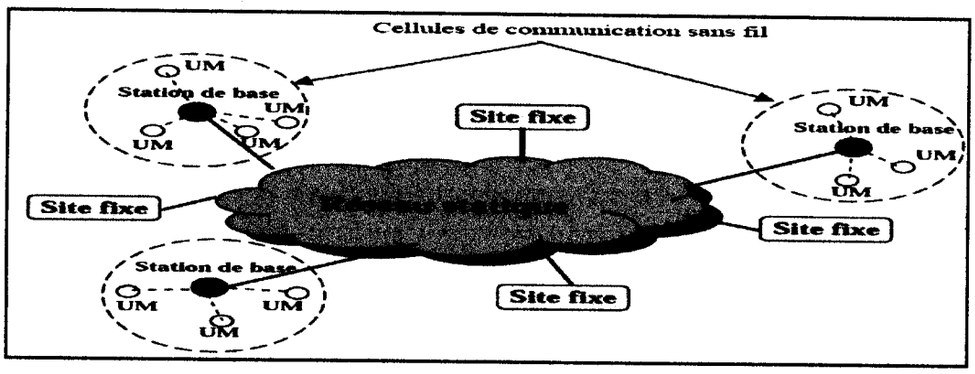


Figure 1. 2 : Le modèle des réseaux sans fil avec infrastructure [TL00].

Parmi les sites fixes, on retrouve les stations de bases (SB représentées par des cercles noirs sur la figure 1.2), appelées aussi points d'accès. Chaque station de base définit une région appelée *cellule*. Celle-ci correspond à la zone de couverture à partir de laquelle les unités mobiles (UM) peuvent émettre et recevoir des messages venants d'autres nœuds de l'intérieur ou de l'extérieur de la cellule. La communication entre deux nœuds (UM) connectés à deux points d'accès différents passe forcément par les SB correspondant aux cellules. Ces SB sont reliées entre elles par des liens filaires. La configuration standard d'un système de communication cellulaire est un maillage de cellules hexagonales. C'est-à-dire que chaque cellule couvre une zone géographique selon la portée de la SB correspondante, mais lorsque la compétition devient importante pour

L'allocation des canaux, la cellule est généralement divisée en sept cellules plus petites [TL00], comme illustré sur la figure 1.2.

2.2. Réseaux ad hoc (sans infrastructure)

Ce sont un type particulier des réseaux sans fil, qui étendent la notion de mobilité pour lui donner un sens beaucoup plus fort. La principale caractéristique qui fait la différence entre ce type de réseaux sans fil et un réseau sans fil classique est l'absence d'une infrastructure préexistante.

Une définition formelle des réseaux ad hoc MANET (Mobile Ad hoc NETwork) est donnée par la RFC 2501. Il s'agit de réseaux sans fil et sans infrastructure fixe, utilisant généralement le médium radio, où chaque nœud peut jouer le rôle du client et de routeur. Les réseaux ad hoc sont auto-organisés, ce qui implique que la connectivité doit être préservée autant que possible automatiquement lorsque la topologie du réseau change (suite à l'apparition, la disparition ou au mouvement de certains nœuds) [DD03].

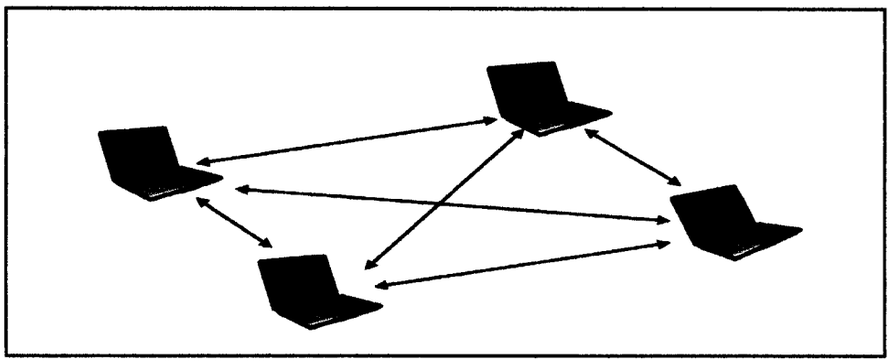


Figure 1. 3 : Le modèle des réseaux mobiles sans infrastructure.

3. Les réseaux de capteurs sans fil

Un réseau de capteurs (RCSF) est un cas particulier des réseaux ad hoc. Il existe des différences considérables entre les réseaux de capteurs et les réseaux traditionnels, donc dans la plus part des cas on ne pourra pas utiliser les mêmes protocoles. Contrairement aux réseaux traditionnels qui sont généralement conçus pour transférer des données entre les nœuds, un réseau de capteurs est étendu pour réaliser des différentes fonctions. Plusieurs nœuds dans un réseau de capteurs combinent les données capturées localement, et les communiquent à l'utilisateur. L'utilisateur ne s'intéresse pas à l'envoie

d'une requête à un nœud particulier, mais plutôt, à la récupération des informations à partir d'une région ou d'un ensemble de nœuds [MI04].

3.1 Définition d'un capteur

Les capteurs sont des petits appareils dotés d'une batterie, capables de communiquer entre eux et de détecter des évènements s'ils se trouvent à l'intérieur de leur rayon de perception. Un capteur est un petit appareil doté aussi de mécanismes lui permettant de relever des informations sur son environnement. La nature de ces informations varie très largement selon l'utilisation qui est faite du capteur : ce dernier peut tout aussi bien faire des relevés de température, d'humidité ou d'intensité lumineuse. Un capteur possède également le matériel nécessaire pour effectuer des communications sans-fil par ondes radio [MM08].

3.2 Architecture physique d'un capteur

S'il existe différents types de capteurs, leur fonctionnement reste le même. L'architecture d'un capteur, représentée en figure 1.4, se résume ainsi :

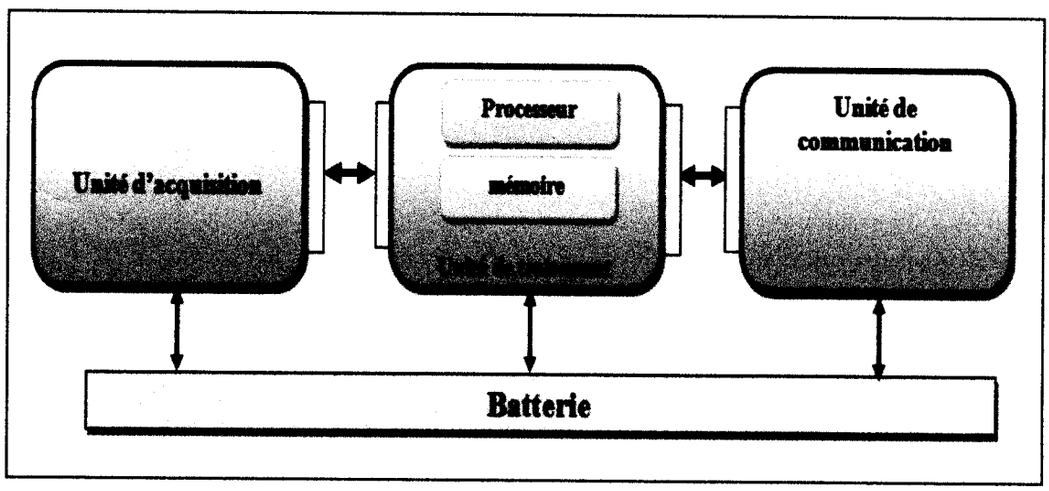


Figure 1. 4 : Architecture physique d'un capteur

- **l'unité d'acquisition** : composée d'un capteur qui obtient des mesures sur les paramètres environnementaux et d'un convertisseur Analogique/Numérique qui convertit l'information relevée et la transmet à l'unité de traitement.
- **l'unité de traitement** : composée d'un processeur et d'une mémoire intégrant un système d'exploitation spécifique. Cette unité possède deux interfaces, une interface pour

l'unité d'acquisition et une interface pour l'unité de communication. Elle acquiert les informations en provenance de l'unité d'acquisition et les envoie à l'unité de communication. Cette unité est chargée aussi d'exécuter les protocoles de communications qui permettent de faire collaborer le capteur avec d'autres capteurs. Elle peut aussi analyser les données captées.

- **l'unité de communication** : unité responsable de toutes les émissions et réceptions de données via un support de communication radio. Elle peut être de type optique, ou de type radiofréquence.
- **la batterie :** un capteur est muni d'une batterie pour alimenter tous ses composants. Cependant, à cause de sa taille réduite, la batterie dont il dispose est limitée et généralement irremplaçable. Pour cela, l'énergie est la ressource la plus précieuse puisqu'elle influe directement sur la durée de vie des capteurs.

Il existe des capteurs qui sont dotés d'autres composants additionnels comme le système de positionnement GPS (Global Positioning System) et un mobilisateur lui permettant le déplacement.

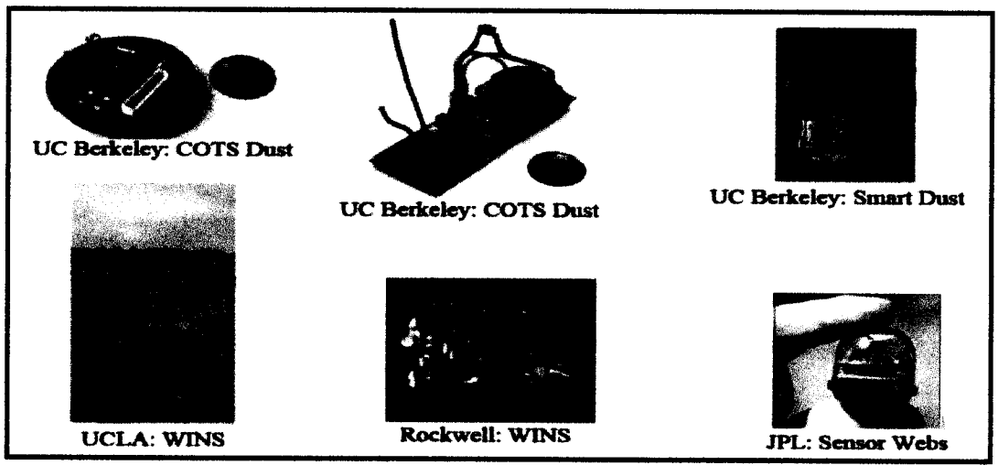


Figure 1. 5 : exemple de capteurs [BK07]

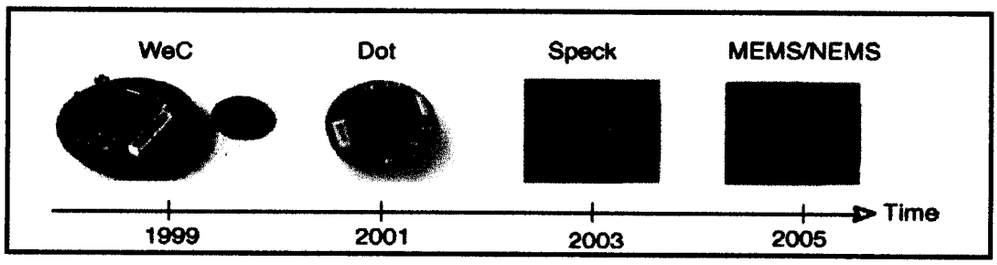


Figure 1. 6 : La taille des capteurs à partir de 1999 [KS07].

3.3 Architecture d'un réseau de capteurs

Un réseau de capteur sans fil est constitué d'un nombre plus ou moins grand de nœuds capteurs. Ces nœuds sont autonomes, distribués dans l'espace qui coopèrent pour surveiller des conditions environnementales ou physiques, telle que la *température*, le *bruit*, la *vibration*, la *pression*, le *mouvement*, etc. A l'origine, le développement des réseaux de capteur sans fil a été motivé par des applications militaires telles que la surveillance de champ de bataille. Cependant, ce type de réseau est maintenant employé dans plusieurs domaines d'application civils, comme la surveillance d'environnement, d'habitat, la surveillance médicale, l'automatisation des maisons, le contrôle du trafic [YR07]. L'architecture des réseaux de capteurs sans fil utilise beaucoup de sources. Historiquement, beaucoup de travaux relatifs ont été effectué dans le contexte des réseaux à auto-organisation, mobiles et Ad Hoc Un réseau de capteurs est constitué essentiellement de plusieurs nœuds capteurs, un nœud sink (ou station de base) et un centre de traitement des données.

- **Les nœuds :** sont des capteurs, leur type, leur architecture et leur disposition géographique dépendent de l'exigence de l'application en question. Leur énergie est souvent limitée puisqu'ils sont alimentés par des piles.

- **Le sink :** c'est un nœud particulier du réseau. Il est chargé de la collecte des données issues des différents nœuds du réseau. Il doit être toujours actif puisque l'arrivée des informations est aléatoire. C'est pourquoi son énergie doit être illimitée. Dans un réseau de capteur sans fil plus ou moins large et à charge un peu élevée, on peut trouver deux sinks ou plus pour alléger la charge.

- **Centre de traitement des données :** c'est le centre vers lequel les données collectées par le sink sont envoyées. Ce centre a le rôle de regrouper les données issues des nœuds et les traiter de façon à en extraire de l'information utile exploitable. Le centre de traitement peut être éloigné du sink[YR07], alors les données doivent être transférées à travers un autre réseau, c'est pourquoi on introduit une passerelle entre le sink et le réseau de transfert pour adapter le type de données au type du canal (comme c'est illustré dans la figure 1.7).

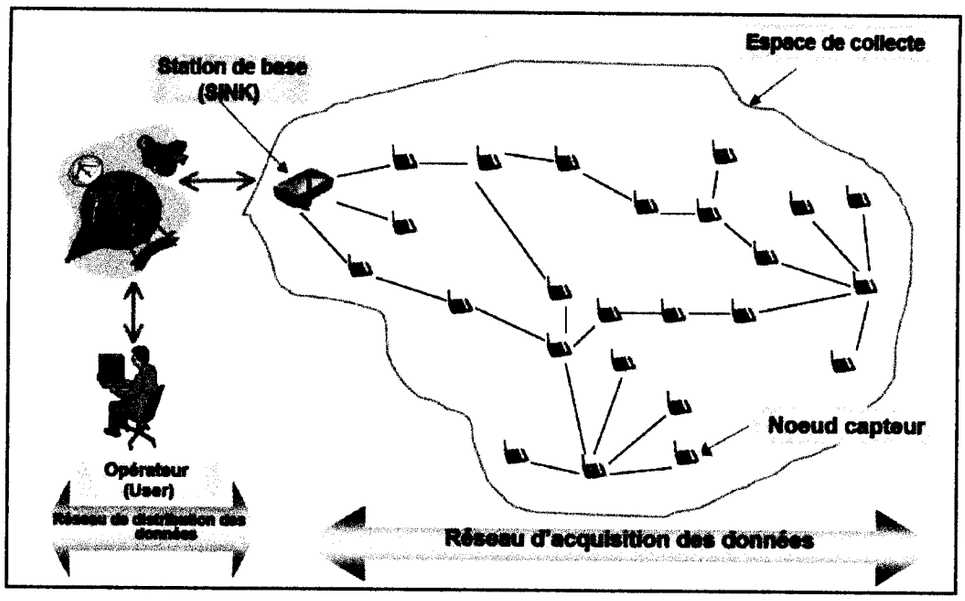


Figure 1. 7 : architecture d'un réseau de capteur [BK07].

Comme le montre la Figure I.7, un RCSF est composé d'un grand nombre de nœuds capteurs éparpillés sur le champ de captage. Quand le sink diffuse une requête, les nœuds collaborent entre eux pour lui envoyer les informations captées à travers une architecture multi-sauts. Le puits transmet ensuite ces données par Internet ou par satellite au gestionnaire de tâches pour les analyser et prendre des décisions. A un niveau plus élevé un RSCF peut être vu comme étant une combinaison de deux entités de réseau:

- **le réseau d'acquisition des données**: c'est l'union des nœuds capteurs et du sink. Son rôle consiste à collecter les données à partir de l'environnement et de les rassembler au sink.
- **le réseau de distribution des données**: son rôle est de connecter le réseau d'acquisition des données à un utilisateur [SK06].

Il existe deux types d'architecture pour les RCSF [MI05] :

3.3.1 Les réseaux de capteurs sans fil plats

Un réseau de capteurs sans fil plat est un réseau homogène, où tous les nœuds disposent des mêmes capacités et fonctionnalités concernant le captage, la communication et la complexité du matériel, seul le sink échappe à cette règle puisqu'il joue le rôle d'une

passerelle chargée de la collecte des données issues des différents nœuds capteurs pour les transmettre à l'utilisateur.

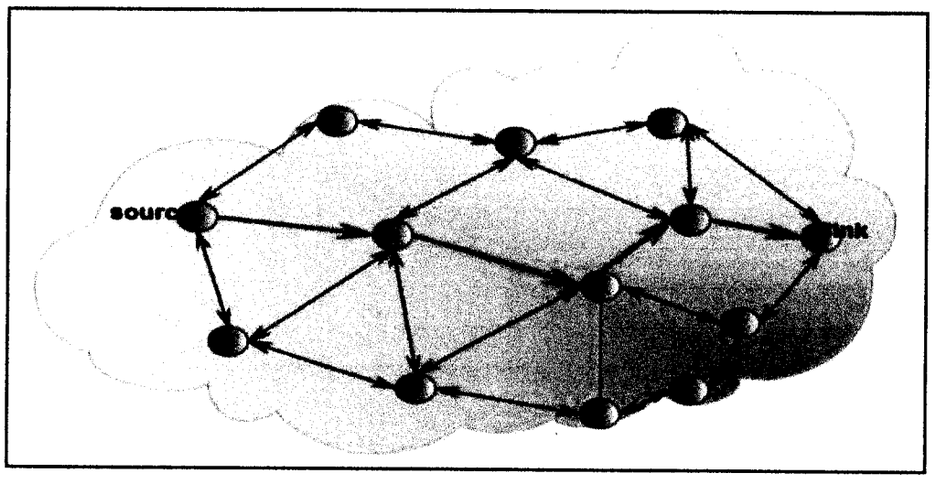


Figure 1. 8: architecture plate [YR07].

3.3.2 Les réseaux de capteurs sans fil hiérarchiques

Un réseau de capteurs hiérarchique est un réseau hétérogène où les nœuds ont des capacités différentes, par exemple certains nœuds peuvent disposer d'une source d'énergie plus importante, une plus longue portée de communication et/ou une plus grande puissance de calcul. Ceci permet de décharger la majorité des nœuds simples à faible coût de plusieurs fonctions du réseau.

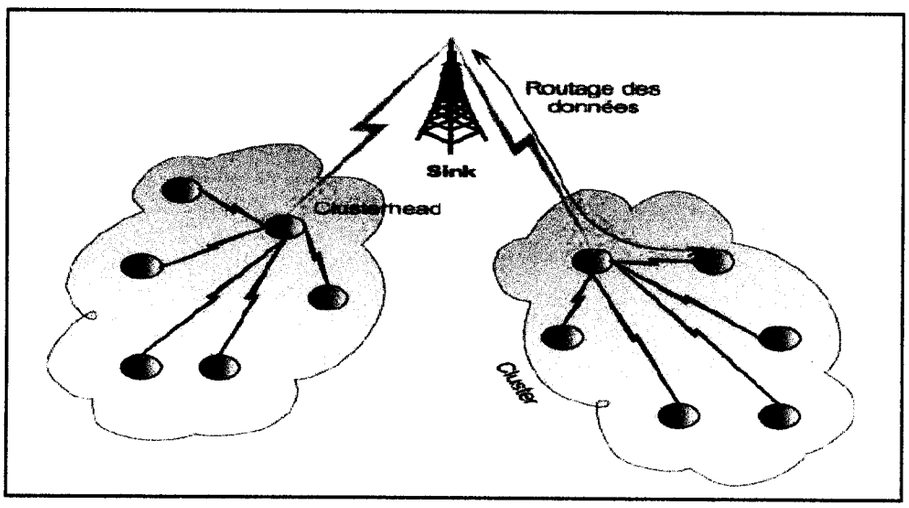


Figure 1. 9: architecture hiérarchique [YR07].

3.4 Caractéristiques des réseaux de capteurs

Les principaux caractéristiques des réseaux de capteurs se résument dans ce qui suit [DE07] :

- **Densité « importante » des nœuds**

Les réseaux de capteurs se composent généralement d'un nombre très important des nœuds pour garantir une couverture totale de la zone surveillée. Ceci engendre un niveau de surveillance élevé et assure une transmission plus fiable des données sur l'état du champ de capteur.

- **Topologie dynamique**

La topologie des réseaux de capteurs instable est le résultat des trois facteurs essentiels suivants: la mobilité des nœuds, la défaillance des nœuds et l'ajout de nouveaux nœuds.

- **Auto organisation**

L'auto organisation s'avère très nécessaire pour ce type de réseau afin de garantir sa maintenance. Vu les différentes raisons résultant une topologie instable du réseau de capteur, ce dernier devra être capable de s'auto organiser pour continuer ses applications.

- **La tolérance de fautes**

Le réseau doit être capable de maintenir ses fonctionnalités sans interruptions en cas de défaillance d'un ou plusieurs de ses capteurs. Cette défaillance peut être causée par une perte d'énergie, ou par dommage physique ou interférence de l'environnement. Le degré de tolérance dépend du degré de criticité de l'application et des données échangées .

- **Scalabilité**

Les réseaux de capteurs peuvent contenir des centaines voire des milliers de nœuds capteurs. Un nombre aussi important engendre beaucoup de transmissions inter nodales et nécessite que le nœud « *Sink* » soit équipé d'une mémoire importante pour stocker les formations reçues.

3.5 Applications des réseaux de capteurs

Les réseaux de capteurs sans fil ont trouvé un ensemble très vaste d'applications dans divers domaines, parmi lesquels on peut citer les applications militaires, environnementales, et de surveillance en général [GC10].

3.5.1 Applications militaires

Le domaine militaire a été un moteur initial pour le développement des réseaux de capteurs, comme c'est le cas pour plusieurs technologies. Un exemple typique d'application est le déploiement d'un tel réseau en un endroit stratégique ou difficile d'accès, à fin de surveiller toutes les activités des forces ennemies ou d'analyser le terrain avant d'y envoyer des troupes (par la détection d'agents chimiques, biologiques ou de radiations, par exemple).

3.5.2 Applications liées à la sécurité

L'application des réseaux de capteurs dans le domaine de la sécurité pourrait diminuer considérablement les budgets consacrés à la sécurisation des lieux et à la protection des êtres humains tout en garantissant des résultats plus fiables. Comme exemple d'applications de ce type nous pouvons citer :

- La détection des altérations dans la structure d'un bâtiment, suite à un séisme ou au vieillissement, par des capteurs intégrés dans les murs ou dans le béton,
- La surveillance des mouvements afin de constituer un système de détection d'intrusions distribué.

3.5.3 Applications environnementales

La surveillance de sites naturels ou industriels étendus utilise les réseaux de capteurs qui garantissent l'aspect distribué et une large couverture. Dans ce domaine nous pouvons mentionner des applications typiques :

- la dispersion de thermo-capteurs à partir d'un avion sur une forêt pour signaler un éventuel début d'incendie,
- le semis de nœuds capteurs en même temps que les graines dans les champs agricoles pour pouvoir identifier les zones sèches et rendre l'irrigation plus efficace.
- le déploiement de nœuds capteurs sur les sites industriels, les centrales nucléaires ou dans les raffineries de pétrole pour détecter des fuites de produits toxiques (gaz,

produits chimiques, éléments radioactifs, pétrole, etc.). Il s'agit d'alerter les utilisateurs dans un délai suffisamment court pour permettre une intervention efficace.

3.5.4 Applications médicales

L'utilisation des réseaux de capteurs dans le domaine de la médecine peut apporter une surveillance permanente des patients et une possibilité de collecter des informations physiologiques de meilleure qualité, facilitant ainsi le diagnostic de maladies grâce à des micro-capteurs qui pourront être ingérés ou implantés sous la peau. Comme applications d'avant garde de ce domaine nous pouvons énumérer :

- les micro-caméras qui peuvent être ingérées et sont capables, sans avoir recours à la chirurgie, de transmettre des images de l'intérieur d'un corps humain,
- la création d'une rétine artificielle composée d'une centaine de micro-capteurs pour améliorer la vision.

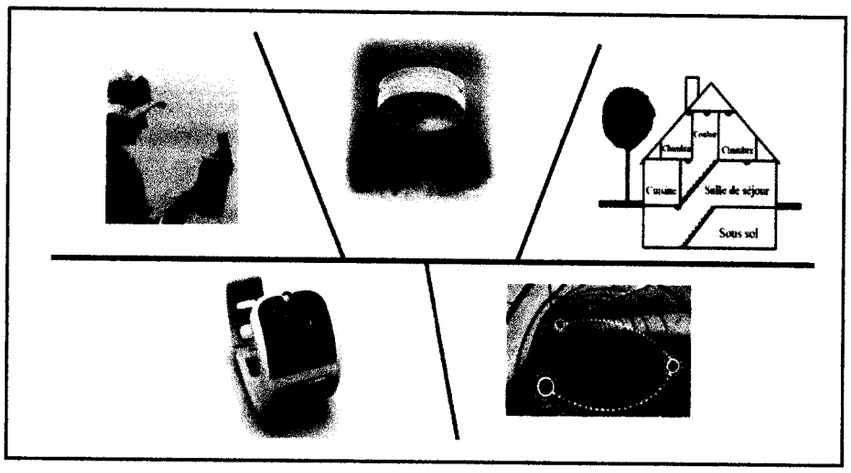


Figure 1. 10: Applications des réseaux de capteurs sans fil.

3.6. Facteurs et contraintes conceptuelles des RCSF

La conception et la réalisation des réseaux de capteurs sans fil sont influencées par plusieurs paramètres. Ces facteurs servent comme directives pour le développement des algorithmes et protocoles utilisés dans les RCSF [AC11].

Durée de vie du réseau : C'est l'intervalle de temps qui sépare l'instant de déploiement du réseau de l'instant où l'énergie du premier nœud s'épuise. Selon l'application, la durée de vie exigée pour un réseau peut varier entre quelques heures et plusieurs années.

Ressources limitées : En plus de l'énergie, les nœuds capteurs ont aussi une capacité de traitement et de mémoire limitée. En effet, les industriels veulent mettre en œuvre des capteurs simples, petits et peu coûteux.

Bande passante limitée : Afin de minimiser l'énergie consommée lors de transfert de données entre les nœuds, les capteurs opèrent à bas débit. Typiquement, le débit utilisé est de quelques dizaines de Kb/s. Un débit de transmission réduit n'est pas handicapant pour un réseau de capteurs où les fréquences de transmission ne sont pas importantes.

Facteur d'échelle : Le nombre de nœuds déployés pour une application peut atteindre des milliers. Dans ce cas, le réseau doit fonctionner avec des densités de capteurs très grandes. Un nombre aussi important de nœuds engendre beaucoup de transmissions inter nodales et nécessite que la station de base soit équipée de mémoire suffisante pour stocker les informations reçues

Topologie dynamique : La topologie des réseaux de capteurs peut changer au cours du temps pour les raisons Suivantes :

- ✓ Les nœuds capteurs peuvent être déployés dans des environnements hostiles (champ de batail par exemple), la défaillance d'un nœud capteur est, donc très probable.
- ✓ Un nœud capteur peut devenir non opérationnel à cause de l'expiration de son énergie.
- ✓ Dans certaines applications, les nœuds capteurs et les stations de base sont mobiles.

3.7. Communication dans les réseaux de capteurs

Nous verrons dans cette section, l'architecture de communication basée sur le modèle OSI ainsi que les protocoles de communications et leurs caractéristiques.

3.7.1 Architecture de communication basée sur le modèle OSI

Le modèle de communication comprend cinq couches qui ont les mêmes fonctions qui celles du modèle OSI ainsi que trois couches pour la gestion d'énergie, la gestion de la mobilité et la gestion des tâches.

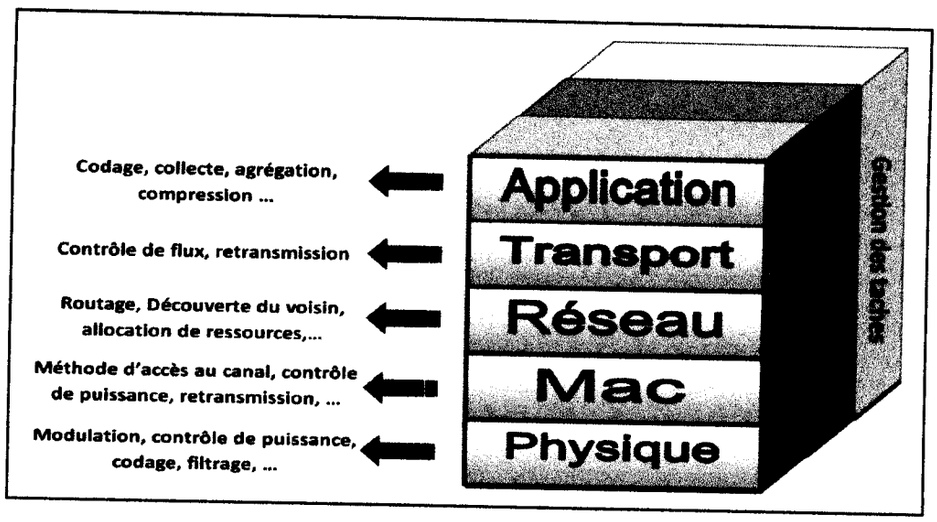


Figure 1. 11: Modèle en couches du réseau de capteurs sans fil.

Rôles des couches

- Couche physique : Matériels pour envoyer et recevoir les données
- Couche liaison de données : Gestion des liaisons entre les nœuds et les stations de base, contrôle d'erreurs
- Couche réseau : Routage et transmission des données
- Couche transport : Transport des données, contrôle de flux
- Couche application : Interface pour les applications au haut niveau
- Plan de gestion d'énergie : Contrôle l'utilisation d'énergie
- Plan de gestion de mobilité : Gestion des mouvements des nœuds
- Plan de gestion de tâche : Balance les tâches entre les nœuds afin d'économiser de l'énergie

3.7.2 Protocole de communication : Zigbee

Bluetooth et Zigbee sont les standards les plus aptes à être exploités dans les réseaux de capteurs sans-fil. Le succès de la technologie Bluetooth l'a amené à être utilisée dans le cadre des LAN sans fil. Malheureusement, cette technologie présente deux inconvénients majeurs : elle consomme une grande quantité d'énergie pour assurer les communications entre les nœuds et elle ne permet pas de connecter plus de 80 nœuds (10 piconets). Ainsi, elle ne peut pas être utilisée par des capteurs qui sont généralement déployés en grand nombre dans des zones hostiles, et qui présentent une autonomie d'énergie (batteries non rechargeables) et qui devraient idéalement fonctionner pour une

longue durée. Par contre, le standard Zigbee offre des caractéristiques qui répondent aux besoins des réseaux de capteurs puisqu'il consomme moins d'énergie que Bluetooth. Le petit débit qu'il offre, n'est pas vraiment un handicap pour un réseau de capteurs puisque la taille des paquets échangés n'est pas vraiment importante [LM09]. ZigBee a été ratifiée en Aout 2003 sous la norme IEEE 802.15.4.

Le tableau 1.1 Illustre la comparaison entre des protocoles Zigbee, Bluetooth et Wifi.

Protocole	ZigBee	Bluetooth	Wi Fi
IEEE	802.15.4	802.15.1	802.11a/b/g
Besoin mémoire	4-32Kb	250Kb+	1Mb+
Autonomie avec pile	Années	Jour	Heures
Nombre nœuds	65000+	7	Heures
Vitesse de transfert	250Kb/s	1Mb/s	11-54-108Mb/s
portée	100m	10-100m	300 m

Tableau 1. 1: protocoles :Zigbee, Bluetooth et Wifi.

3.8 Agrégation de données dans les réseaux de capteurs

Nous abordons dans cette section la technique d'agrégation des données dans les réseaux de capteurs.

3.8.1 Problématique d'agrégation

Dans un capteur, la problématique principale concerne la consommation d'énergie : en effet, ces derniers doivent restés opérationnels le plus longtemps possibles, dans des conditions parfois difficiles (ex : lâchés par avion sur les parois d'un volcan). Comme il n'est pas possible de recharger leur énergie ni changer les piles par exemple (on ne sait pas toujours où se trouvent les capteurs), il est nécessaire d'économiser au maximum l'énergie consommée par ces derniers. On estime que la transmission des données d'un capteur représente environ 70% de sa consommation d'énergie.

De plus, les réseaux de capteurs étant assez denses en général, cela signifie que des nœuds assez proches en terme de distance (voisins) peuvent capter les mêmes données (température, pression, humidité équivalentes par exemple) et donc il apparaît nécessaire

d'introduire le mécanisme d'agrégation de données afin d'éviter la duplication d'information au sein du réseau de capteurs et donc de préserver leur énergie et donc d'augmenter la durée de vie du réseau.

3.8.2 Définition

4. Conclusion

Dans ce chapitre, nous avons présenté les réseaux sans fil en général. Ils sont généralement décomposée selon deux modes : réseaux avec infrastructure ou centralisés (cellulaires) et sans infrastructure ou décentralisés (Ad Hoc).

Nous avons décrit ensuit les RCSF qui sont apparentés aux réseaux Ad Hoc. Ces réseaux connaissent un grand essor grâce a la multitude d'applications qu'ils offrent ainsi que leurs caractéristiques inhérentes telles que leur déploiement aléatoire et de faible coût, leur grand mobilité et aussi, grâce aux récents développements concernant la miniaturisation des composants électroniques (construction des capteurs de quelques millimètres cubes de volume.

Un réseau de capteurs est constitué de plusieurs nœuds capteurs chargés de collecter des informations sur l'environnement dans lequel ils sont déployés et de les transmettre vers un site donné. Il se caractérise par une grande flexibilité et tolérance aux fautes, un prix réduit et des moyens rapides de déploiement du réseau. Grâce à ces caractéristiques le champ d'application des réseaux de capteurs s'est étendu pour contenir la plupart des domaines tels que l'industrie, la recherche, l'environnement, ou la médecine. Il paraît évident qu'ils auront de nouveaux effets dans notre vie de tous les jours et peuvent changer considérablement notre vision du monde.

Nous verrons dans le chapitre suivant, le contexte de notre travail de recherche, à savoir les feux de forêt. Un problème épineux qui menace notre environnement. Nous donnerons quelques statistiques sur les feux de forêt en Algérie ainsi que leur historique et les moyens de lutte utilisés.

Chapitre II

Les feux de forêts en Algérie, une nouvelle méthode de protection

1. Introduction

L'Algérie constitue une entité écologique exceptionnelle dans la biosphère. Rares sont les autres pays biogéographiques présentant une telle étendue et possédant une telle surface constituée par des écosystèmes de types méditerranéen, steppique et saharien. Elle est toutefois loin de disposer de tous les atouts que laisserait supposer sa dimension territoriale. Ses atouts naturels sont certes conséquents, tant en ressources de surface qu'en ressources de sous-sol, mais il faut tempérer les simples estimations quantitatives et les idées reçues que l'on a pu fonder sur elles, car dans une vision de développement durable, les ressources de l'Algérie apparaissent des plus limitées lorsqu'on les confronte à la croissance démographique enregistrée depuis l'indépendance et que l'on complète cette comparaison par les menaces de plus en plus inquiétantes que fait peser sur ces ressources, leur exploitation ou mise en valeur inconsidérée. La forêt algérienne, actuellement fragile, a besoin d'être protégée car la déforestation ne cesse de s'accentuer en raison des incendies de forêts répétés.

Ce deuxième chapitre détaille une étude générale des feux de forêts en Algérie le but de cette dernière est d'établir un constat de l'état actuel du patrimoine forestier Algérien. La fin de ce chapitre sera consacrée à la présentation d'une nouvelle méthode de lutte contre les incendies en Algérie.

2. Profil forestier de l'Algérie

En Algérie les forêts, les reboisements, les maquis et les garrigues occupent une superficie d'environ 4100000 ha, néanmoins chaque année environ 36 000 ha sont parcourus par les incendies. La forêt algérienne est directement liée au climat méditerranéen qui caractérise tout le nord de l'Algérie. Les caractères du milieu confèrent à la forêt une vulnérabilité et une fragilité accentuées par une exploitation qui dure depuis quelques millénaires. Les forêts climaciques sont assez réduites, de grandes superficies sont remplacées par des formations de dégradation telles que les maquis, les garrigues, les broussailles et les pelouses. La dégradation ancienne de la forêt a entraîné un déséquilibre important entre les superficies existantes et les superficies potentielles [MATE03].

Essences	Superficie (ha)	Taux %
Pin d'Alep	881000	21.5%
Chêne liège	230000	5,6%
Chêne vert	108000	2,6%
Chêne Zeen et Chêne Afares	48000	1,2%
Eucalyptus	43000	1%
Pin maritime	31000	0,8%
Cèdre de l'Atlas	16000	0,4%
Autres (Thuya + Genévrier + Frêne)	124000	3%
Reboisement et protection	717000	17,5%
Maquis et broussailles + vides	1902000	46,4%
Total	**4100000**	**100%**

Tableau 2. 1: Principales essences des forêts algériennes (2007).

Sur les 48 wilayas que compte l'Algérie, 40 disposent d'une couverture forestière, les huit wilayas du Sud sont dépourvues de forêts. La wilaya d'El Tarf dispose du taux de couverture forestière le plus élevé (57,51%), alors que pour la wilaya de Naama le taux de couverture n'est que de 0,36%. En ce qui concerne la superficie forestière c'est la wilaya de Batna qui dispose de la plus grande superficie avec 314 565 ha, la plus petite superficie revient à la wilaya d'Alger (5000 ha). Cette répartition s'explique en grande partie par le climat, en effet les massifs littoraux du nord-est, les plus humides, sont aussi les régions les plus forestières. Les 4,1 millions d'hectares de couverture forestière ne représentent qu'un taux de boisement de 10,89% en ne considérant que le nord du pays, et seulement 1,72% si l'on prend en ligne de compte tout le territoire national. Dans les deux cas, cette couverture forestière est nettement insuffisante en comparaison au taux de 25%, mondialement admis.

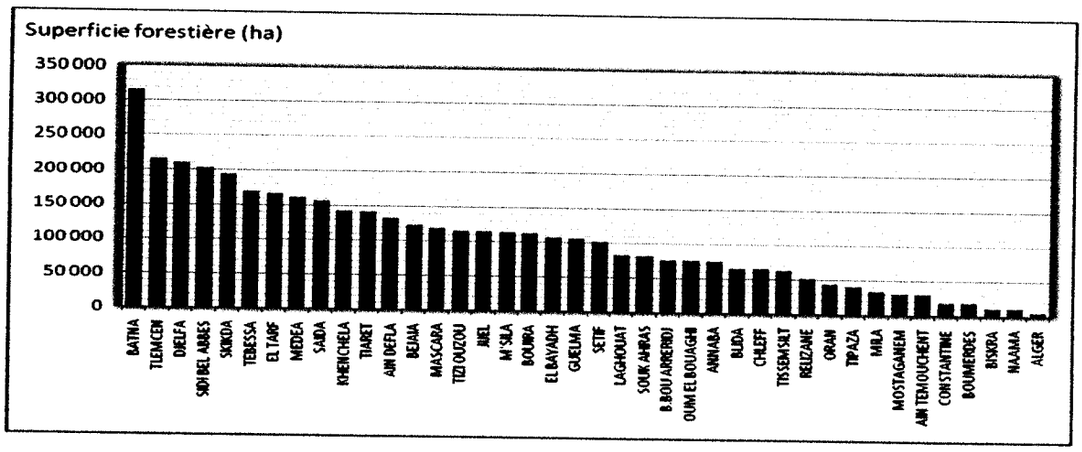


Figure 2. 1 : Répartition de la superficie forestière par wilaya en Algérie (2007).

3. L'historique des feux de forêt en Algérie

De tous les facteurs de dégradation de la forêt algérienne, les incendies sont les plus dévastateurs. Ils détruisent en moyenne, en l'espace de quelques mois seulement (juin à octobre), plus de 36 000 ha de formations ligneuses par an. La moyenne des différents programmes de reboisement depuis 1963 qui est de 26 000 ha/an ne peut équilibrer ces pertes, même si le taux de réussite de ces actions est de 100 %, ce qui n'est malheureusement pas le cas. Entre 1881 et 2006, trois décennies ont été particulièrement désastreuses pour la forêt algérienne, la décennie 1911-1920 qui coïncide avec la première guerre mondiale, la décennie 1951-1960 et la décennie 1991-2000. Deux causes principales sont à l'origine des incendies de grande ampleur que connaissent nos forêts :

- le climat, c'est durant les années particulièrement sèches (1983) que les incendies ont été les plus dévastateurs ;
- la deuxième cause est liée au trouble social, en particulier lors des guerres et des révoltes, en raison notamment, de la conjoncture sécuritaire difficile qu'a traversé l'Algérie durant la décennie 1990-2000. C'est l'année 1994 qui a été la plus destructrice pour la forêt algérienne avec une superficie de 271 598 ha soit 6,6% de la superficie forestière totale. Entre 1881 et 2006, 4 834 874 ha, soit 118% du domaine forestier algérien à brûlé en 126 ans [AA08].

Années	Forêts (Ha)	Hors Forêts (Ha)	nb F	Total (Ha)
2000	35617,52	20164,07	1910	55781,59
2001	9066,40	5311,29	1327	14377,69
2002	6959,95	5257,52	1008	12217,47
2003	5743,79	12451,71	1233	18195,50
2004	7010,59	24988,18	1463	31998,77
2005	14283,03	14097,03	2013	28380,06
2006	8610,29	8306,07	2029	16916,36
2007	23451,31	24487,26	2026	47938,57
2008	10577,53	15437,06	2378	26014,59
2009	11769,85	14413,07	2358	26182,92
2010	11008,31	19620,91	3439	30629,22

Tableau 2. 2: Surfaces forestières incendiées en Algérie entre 2000 et 2010.

4. Fréquence annuelle des incendies de forêt

Entre 1985 et 2006, la moyenne annuelle des superficies incendiées se chiffre à 35 448,73 ha. Cependant, la superficie brûlée fluctue d'une année à une autre. Durant les années 1993, 1994, 1999 et 2000 la superficie incendiée est supérieure à cette moyenne. C'est l'année 1994 qui a été la plus destructrice pour la forêt algérienne, où pas moins de 271 598 ha ont brûlé, ce qui représente 34,83%, soit un tiers de la superficie totale incendiée durant les deux décennies (figure II.3). La cause de ces incendies exceptionnels durant cette année est liée au trouble social, en raison notamment, de la conjoncture sécuritaire difficile qu'a traversé le pays durant la décennie 1990-2000.

Par ailleurs, il y a lieu de signaler que durant les années qui suivent celles où l'on a enregistré d'importantes superficies incendiées, nous constatons une baisse substantielle des effets des incendies de forêt, ceci serait due vraisemblablement à la vigilance observée après avoir vécu des catastrophes, notamment au niveau des zones fortement boisées et densément peuplées, où nous avions constaté la menace qui a pesé sur des villages entiers enclavés en milieu forestier. Quant au nombre d'incendie, la moyenne annuelle est de 1470,64 foyers. Même si l'on remarque des différences d'une année à une autre, elle n'est pas aussi importante que celle qui se rapporte aux superficies (voir figure 2.2) [AA08].

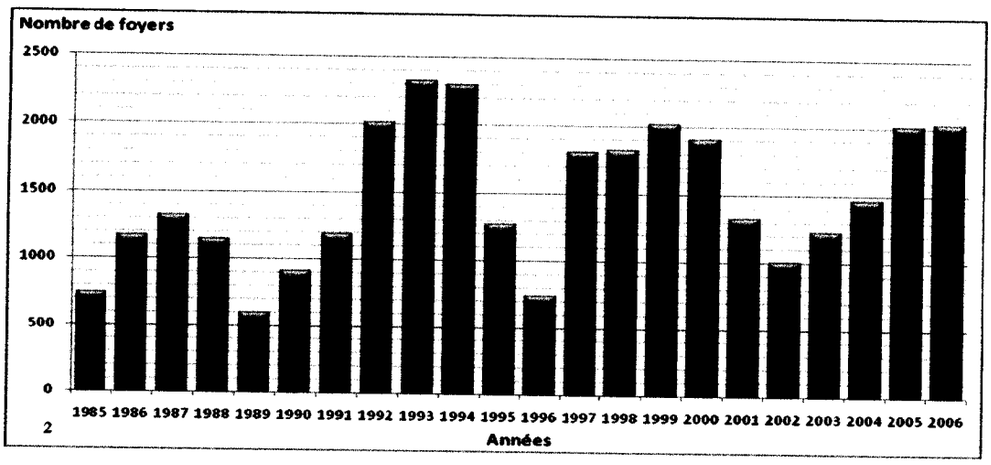


Figure 2. 2: Fréquence annuelle des départs de feux en Algérie

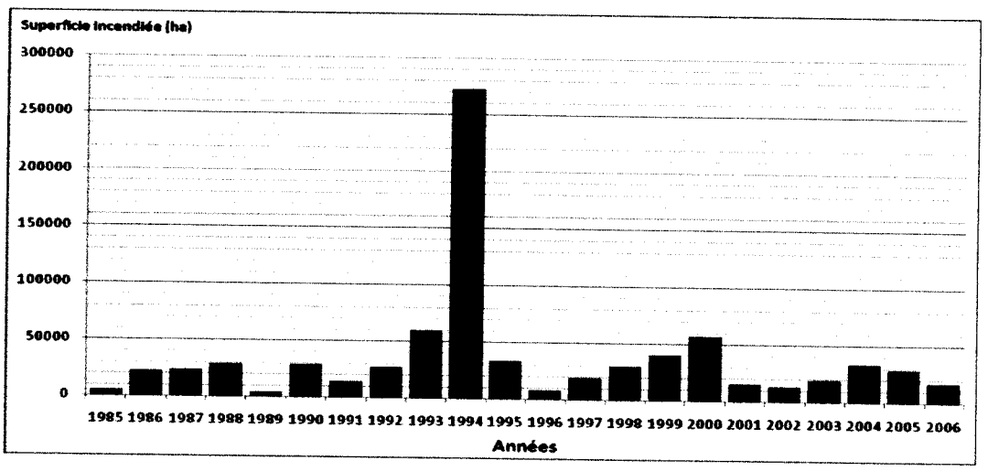


Figure 2. 3 : Fréquence annuelle des superficies incendiées en Algérie

5. La recherche des causes d'incendies de forêt

Les statistiques sur les causes des incendies de forêt en Algérie sont loin d'être complètes, mais il est évident que la plupart des incendies sont déclenchés par l'homme. Les causes exactes des feux de forêts sont variables d'une région à l'autre et sont difficiles à établir avec exactitude. Au niveau local, les rapports d'incendies actuels désignent dans 86% des cas, la formule : "cause volontaire, auteur inconnu". Les analyses ont déjà mis en évidence la question des feux pastoraux, dont l'ampleur échappe à leurs auteurs et qui sont la cause principale des incendies dans les régions où le feu est le moyen le plus économique de "régénérer" des pâturages envahis par des ligneux et d'ouvrir des maquis impénétrables.

Une autre cause, non moins importante de développement des incendies est la culture sur brûlis, notamment, pour les arachides, l'extension des vides labourables (clairières réservées à l'agriculture) et la mise à nue par le feu des terres à vocation forestière pour des cultures spéculatives. Cette pratique a été favorisée par des textes permettant l'accès à la propriété foncière par la mise en valeur des terres, conçues à l'origine pour le Sahara et qui a été étendue illégalement à l'ensemble du territoire national.

Les populations urbaines sont particulièrement insensibles au danger des incendies et à leurs conséquences potentiellement dangereuses. De nombreux citadins ne considèrent pas les feux de forêt comme une menace, même au cœur de l'été. L'inconscience des fumeurs et des touristes qui font du feu pour cuire leurs aliments est la source de beaucoup d'incendies. Une cause de plus en plus importante est le brûlage de grandes quantités de déchets solides laissés par les touristes ou produits par les autres utilisations des forêts à des fins récréatives, souvent effectué sans prendre les précautions nécessaires.

Des incendies, en nombre croissant, sont allumés non à des fins utilitaires mais dans le seul but de détruire pour diverses raisons, y compris la vengeance privée et les conflits que soulèvent le droit de propriété, les droits de chasse et même les politiques forestières gouvernementales. Les incendies ont aussi pour but d'essayer de modifier la classification de l'utilisation des terres. Dans certaines parties de l'Algérie, de vastes zones forestières ont été détruites par des promoteurs immobiliers peu scrupuleux [AA08].

6. Les moyens de lutte contre les incendies de forêt en Algérie

L'aménagement forestier est l'un des moyens le plus adéquat pour lutter contre les incendies de forêt. Les plans d'aménagement intègrent toutes les infrastructures nécessaires en matière de défense des forêts contre les incendies à savoir :

- L'ouverture et l'entretien de pistes ;
- L'ouverture et l'entretien de T.P.F. (Tranchées Pare-Feu) ;
- L'installation de poste de vigie ;
- La réalisation et l'aménagement de points d'eau ;
- Par ailleurs une surveillance des massifs forestiers par des brigades mobiles et les gardes forestiers doit être prévue particulièrement en été.

7. Les conséquences des incendies de forêt en Algérie

Les grands effets de destruction et de régression forestière, sont l'œuvre des incendies de forêts. Les incendies représentent une importante cause de destruction tant des écosystèmes climaciques ou de ceux caractérisant les formations végétales ligneuses dégradées (maquis, garrigue, matorral, etc.). En effet, lorsque l'incendie devient trop fréquent, les forêts n'ont plus le temps de se régénérer et sont tout d'abord remplacées par des formations végétales dégradées : boisements ouverts puis formations de type arbustif. Progressivement, s'installe une succession régressive d'écosystèmes pouvant atteindre le stade ultime de pelouses squelettiques, dépourvues de végétation ligneuse et laissant le sol à nu par renouvellement systématique du feu [DGF02].

Les pertes économiques dans le secteur forestier générées par les incendies de forêt en Algérie entre 1985 et 2006 se chiffrent à plus de 113 milliards de dinars algériens. Cette évaluation financière ne prend en compte que la valeur marchande des produits perdus (bois, liège, broussailles, alfa, arboriculture...) sans tenir compte des dépenses annuelles pour la lutte contre les incendies de forêt (matériels, véhicules, main-d'œuvre...). De plus il faut ajouter à cela une perte à long terme de la biodiversité et de l'équilibre des écosystèmes forestiers, qui reste très difficile à chiffrer. Surtout si on sait que le reboisement et l'entretient d'un hectare coûte en moyenne 80.000 DA. L'une des conséquences indirectes des incendies est l'érosion des sols et l'envasement des barrages qui représentent une perte économique importante. Donc, ces chiffres sont bien en dessous de la réalité, mais, ils permettent d'avoir au moins une idée de l'impact économique des incendies sur la forêt algérienne.

8. Proposition Pour une meilleure prévention et gestion des incendies de forêt en Algérie

Le gouvernement algérien a une politique de défense contre les incendies depuis plusieurs années déjà. Mais, le nombre des incendies continue d'augmenter. Reste à savoir si toutes les techniques disponibles sont appliquées et si la politique de prévention est menée jusqu'au bout. L'analyse des bilans des campagnes de prévention et de lutte contre les incendies de forêts, atteste des manques sur le plan organisationnel, réglementaire et législatif, dans ce sens, nous avons proposés une nouvelle méthode de lutte contre les

incendies de forêt qui consiste à utiliser les réseaux de capteurs sans fil à la détection des incendies de forêts, le rôle de ces applications est de fournir des données exactes et réelles sur la propagation des incendies dans les forêts.

8.1. Principe

Le principe de base pour détecter les incendies en utilisant les RCSF, consiste à déployer des capteurs pour couvrir une zone cible. Ces capteurs devraient être capables de recueillir plusieurs types de données, comme la température, l'humidité. Généralement, trois types de capteurs sont utilisés. Le premier type s'occupe de recueillir les données concernant le milieu ou ces capteurs sont déployés. Ces données sont transmises via des liens radio aux capteurs du deuxième type. Le deuxième type de capteurs sont les sinks et ont pour rôle de traiter les données qui proviennent des capteurs du premier type. Ce type de senseurs transmet les données traitées à la passerelle via des liens radio. Leur nombre est beaucoup moins élevé que le nombre de capteurs déployés. Le troisième type de capteurs regroupe les passerelles. Ces dernières sont connectées directement à un ordinateur via un port USB (Universal Serial Bus). Leur rôle est de transmettre les données reçues des sinks directement à l'ordinateur. Dans ce cas, une seule passerelle est suffisante pour transmettre les données.

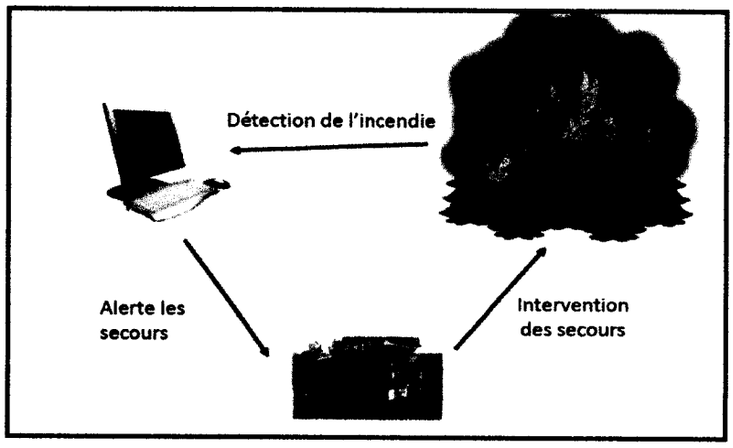


Figure 2. 4 : la détection d'un feu de forêt (le principe générale).

En plus des capteurs, il faut avoir un équipement qui reçoit les données et qui les affiche aux utilisateurs. Généralement, les utilisateurs utilisent des ordinateurs pour afficher ces données. On peut aussi utiliser d'autres équipements pour avoir beaucoup de détails concernant les incendies et pour mieux les localiser, comme le GPS (Global

Positioning System), des caméras et des stations de base. Par exemple, Lloret et al [JL,MG] ont utilisé des caméras dans un réseau de senseurs sans fil pour avoir des images et des vidéos réels sur l'incendie. L'utilisation des caméras est surtout utile pour valider la présence d'incendie et éviter les fausses alarmes. [YK12]

8.2. Modèles de propagation du feu

Plusieurs types de modèles de propagation de feu ont été développés, classés par ordre de complexité croissante, ce sont : [CP,FG01]

- Modèles Statistiques (type automates cellulaires): Les états de la forêt sont modifiés de façon probabiliste. Ces modèles fournissent des codes extrêmement rapides, mais il est assez difficile de relier les transitions de probabilité en fonction de paramètres physiques (vent, humidité, pente etc...).

- Modèles Empiriques, Rothermal (1972): La vitesse de propagation est déterminée en laboratoire en fonction de paramètres (vent, humidité, pente etc...), par un bilan local d'énergie. Ces modèles fournissent aussi des simulations rapides. Pour des raisons de similitude physique, les expériences de laboratoire ne représentent pas nécessairement les feux réels, les corrélations obtenues sont donc difficilement utilisables.

- Modèles Physiques: Les modèles physiques se décomposent en deux sous classes : les modèles de propagation ou modèles physiques et les modèles de combustion dits complètement physiques. Dans les modèles complètement physiques la végétation est considérée comme un milieu poreux. Les relations de conservation de la masse, de la quantité de mouvement et de l'énergie sont écrites sur les grandeurs moyennes et des relations de fermeture sont utilisées. La combustion d'un tel milieu est alors modélisée. [KC,OG].

8.3. Méthodes utilisées

Dans le but de prévenir les feux de forêts, plusieurs projets ont eu lieu, Chaque projet dispose de sa propre méthode afin d'étudier les comportements des incendies pour garantir une meilleure prévention de ce phénomène [YK12].

- **Moderate-resolution Imaging Spectroradiometer (MO-DIS)** : MODIS est parmi les premiers projets réalisés dans ce domaine de la prévention des incendies. Ce système ainsi que certains autres, se basent sur des images satellites pour donner des rapports concernant le feu. Ce genre de système est extrêmement couteux et dépend des conditions météorologiques. Par exemple, dans le cas ou le ciel est nuageux, les données des images satellites ne sont pas exactes, ce qui influence la performance de tels systèmes. Donc, il est clair que trouver d'autres systèmes moins couteux et beaucoup plus fiable est nécessaire. Et aux vues de la fiabilité des senseurs, de leur moindre couteux, ainsi que leur adaptation avec le milieu ou ils sont déployés, de nouveaux projets qui se basent sur la performance des senseurs pour détecter et étudier le comportement des feux de forêts ont vu le jour.

- **Réseaux de capteurs** Liyang Yu et al [LY,NW] ont proposé des méthodes pour détecter le déclenchement des feux de forêts en utilisant les RCSFs. Le but est de détecter ou de prévoir le feu de forêt le plus rapidement possible. Dans le paradigme proposé, un grand nombre de capteurs ont été déployés dans une forêt. La figure 3.5 présente le déploiement de ces capteurs.

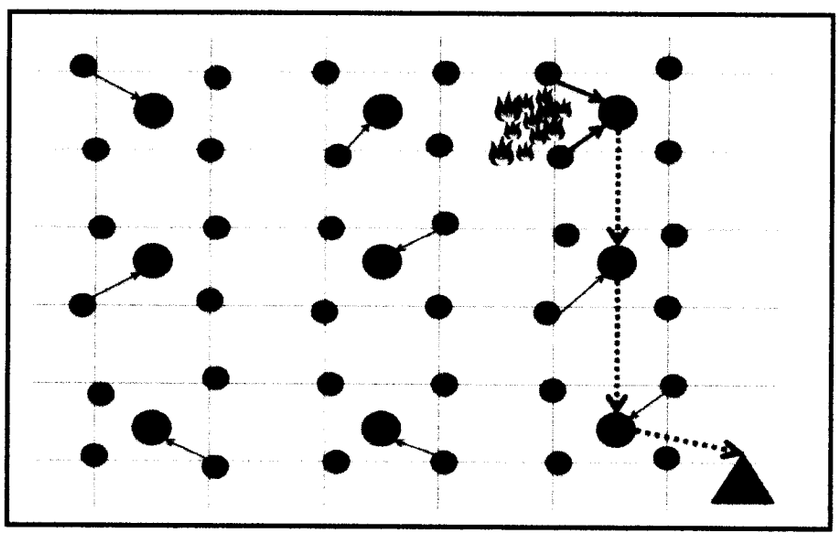


Figure 2. 5 : Réseau de capteur sans fil pour la détection d'un feu de forêt [YE10].

Cette figure montre un grand nombre de capteurs déployé dans une forêt. Ces capteurs sont organisés dans des clusters. Chaque capteur a un chef de cluster pour correspondre.

Les capteurs recueillent les données mesurées, telles que la température, l'humidité relative et la vitesse du vent, tout en étant capables de savoir leur localisation, et les

envoient à leurs capteurs chef du cluster qui traitent les données collectivement, en construisant un réseau de neurones. Ce dernier prend les données mesurées en entrée pour produire des "indices météos" (weather indexes) qui mesurent la probabilité qu'un incendie se déclenche. Les chefs de cluster envoient ces indices à un capteur gérant via le sink. Ainsi, le senseur gérant fournit deux types d'informations aux utilisateurs en se basant sur les indices météos et sur d'autres facteurs qui sont : le rapport d'apparition des événements anormaux (par exemple : fumée ou détection d'une température extrêmement élevée), le taux de risque d'incendie pour chaque cluster.

9. Projet national de recherche proposé en Algérie

Parmi les projets de recherche proposés en Algérie pour la surveillance environnementale, nous citons le projet PNR « **Mise en œuvre d'un réseau de capteurs sans fil pour la détection des feux de forêt**» qui est parrainé par l'agence nationale de recherche au sein du laboratoire STIC de l'université de Tlemcen. Le but de ce projet est de développer une stratégie de déploiement des réseaux de capteurs à grande échelle et cela par le développement d'un ensemble d'algorithmes et méthodes de routage, de sécurité et de localisation pour permettre la mise en place d'un tel réseau. Un autre but de ce projet est le travail en collaboration avec des gardes forestiers pour déployer un réseau de capteurs pour détecter les feux de forêt dans un parc d'attraction.

La finalité de cette application est d'intégrer les gardes forestiers à ce projet pour leur permettre la maîtrise de cette nouvelle technologie à savoir les réseaux de capteurs sans fil, qui leur facilitera le travail par une diminution du nombre d'agents utilisés pour la surveillance des feux de forêts, et d'autre part permettre l'accélération des interventions des gardes forestier pour cerner les feux à temps.

En tant qu'étudiants en fin de projet, et encadrés par Mme Labraoui Nabila, membre du laboratoire STIC et membre de ce projet, notre contribution dans ce projet se limite à la conception d'un modèle de détection de feu de forêt et à l'implémentation d'une partie de l'application et à sa simulation. L'expérimentation aurait donné plus de valeur à notre travail afin de valider le prototype proposé mais hélas, le matériel n'est pas encore disponible.

10. Conclusion

Les incendies seront toujours présents dans nos forêts. Des perturbations, telles que le feu, ont façonné les paysages de l'Algérie les rendant tels que nous les aimons aujourd'hui. Il nous faut reconnaître l'importance du rôle du feu dans nos écosystèmes et en tenir compte dans la planification de la gestion de nos terres. Pour lutter contre la dégradation des forêts, il y a lieu tout d'abord de procéder à l'aménagement et à la réalisation des travaux sylvicoles, à l'ouverture et l'entretien des infrastructures et équipements, pour pouvoir prétendre à une gestion rationnelle et durable, de mettre en place des dispositifs de prévention et de lutte contre les feux de forêts, les maladies et parasites et toute autre forme d'atteinte au patrimoine forestier. L'ensemble de ces actions qui s'inscrivent dans le cadre de la politique forestière et de protection, doivent intégrer les préoccupations essentielles d'ordres écologiques, économiques et sociales, et s'inspirer également du respect des conventions et des accords internationaux que l'Algérie a ratifiés.

Ce chapitre a été présenté en deux parties fondamentales, la première partie concerne le profil forestier en Algérie , l'historique des feux de forêt en Algérie avec quelque statistiques ,les méthodes de lutte contre les incendies et les conséquence de ces feux de forêts , la deuxième partie on a proposé une nouvelle méthode pour la prévention contre les incendies en Algérie et qui ce base sur les réseaux de capteur sans fil.

Dans le chapitre suivant, nous allons présenter le système d'exploitation TinyOs, un système open source embarqué pour les RCSF afin de pouvoir comprendre son fonctionnement et présenter le langage NesC pour l'implémentation de notre application sur la détection des feux de forêts.

Chapitre III

Le système d'exploitation pour les réseaux de capteurs: TinyOS

1. Introduction

TinyOS est un système d'exploitation open-source spécialement conçu pour les applications embarquées fonctionnant en réseaux et, en particulier, pour les réseaux de capteurs sans fil. Initialement développé au sein de l'université de Berkeley en Californie, TinyOS est devenu le standard de facto. En effet, celui-ci est installé sur tous les motes disponibles actuellement et de nombreux groupes de recherche ainsi que des industriels s'en servent afin de développer et tester des protocoles, différents algorithmes, des scénarios de déploiement.

Cette popularité est liée au fait que TinyOS fournit non seulement des outils de développement et de simulation aux développeurs mais également une solution permettant de développer rapidement des applications répondant à la diversité des caractéristiques existantes d'un réseau à l'autre telles que le besoin de fiabilité dans les informations récoltées, la qualité de service du réseau ou encore aux capteurs mis en jeu. En outre, le domaine de l'embarqué impose de sévères contraintes notamment en ce qui concerne l'espace de stockage et l'espace mémoire alloués au système d'exploitation et aux applications tournant dessus. TinyOS répond à ce problème en générant une très petite empreinte mémoire, celle-ci correspondant à la fusion du système d'exploitation et de l'application exécutée. Enfin, en ce qui concerne plus particulièrement le domaine des réseaux de capteurs, les capteurs sont souvent à l'origine de la détection d'un phénomène et informe le réseau de son existence.

TinyOS propose ainsi un modèle de programmation orienté évènement. Cette section décrit TinyOS en commençant dans un premier temps par la description de son modèle et des principales notions associées à celui-ci. Ensuite, nous détaillerons le langage NesC utilisé pour l'implémentation de TinyOS. Celui-ci est un langage orienté composant s'inspirant du C classique [KM10].

Dans ce chapitre, on va introduire et développer le mode de fonctionnement de la plateforme TinyOS ainsi que le langage NesC. Cette description va nous permettre par la suite (chapitre IV) de valider notre application pour la détection des feux de forêt sur la plateforme TinyOS .

2. Système Embarqué TinyOS

2.1 Présentation

TinyOS est un système d'exploitation open source conçu pour des réseaux de capteurs sans fil. Il respecte une architecture basée sur une association de composants, réduisant la taille du code nécessaire à sa mise en place. Cela s'inscrit dans le respect des contraintes de mémoires qu'observent les réseaux de capteurs.

Pour autant, la bibliothèque de composant de TinyOS est particulièrement complète puisqu'on y retrouve des protocoles réseaux, des pilotes de capteurs et des outils d'acquisition de données. L'ensemble de ces composants peut être utilisé tel quel ou bien il peut être aussi adapté à une application précise [WZ06].

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

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

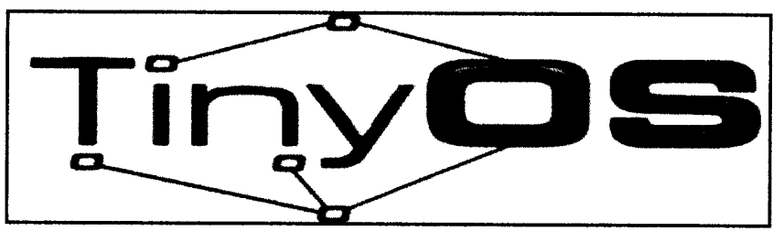


Figure 3. 1: Symbole de system TinyOS

2.2 Objectifs

L'équipe en charge de TinyOs à l'université é de Brekeley lui a fixé les objectifs suivant :

- **Permettre un haut niveau de concurrence** : dans un réseau de capteurs sans fil, les différents éléments sont en compétition pour émettre .Or, lorsque les données sont

récoltées fréquemment et régulièrement, il faut être en mesure de les transmettre en conséquence. La conception du système doit donc prendre en compte ces deux problématiques.

- **Fonctionnement avec des ressources limitées :** cette contrainte prend en compte deux problèmes distincts : d'une part, un capteur est petit donc les composants qu'il embarque sont peu puissants et peuvent stocker extrêmement peu de données. D'autre part, il ya l'énergie, un nœud capteur disposant d'une pile ou d'une batterie au lithium pour fonctionner.

- **Supporter un maximum de matériels possible** : cet objectif, lié au précédent, a besoin d'être réalisé pour que TinyOS soit adopté par un maximum d'acteurs du domaine des capteurs sans fil. En effet, les différents modèles de capteurs embarquent des matériels qui sont susceptibles de varier. Par exemple, il existe plusieurs modules radio : nous pouvons citer le MPR400CB utilise sur les cartes Mica2, Alors le CC240 est celui des carte Micaz et TelosB.

- **La robustesse** : un RCSF est souvent amené à être laissé tel quel, sans intervention extérieure pendant plusieurs semaines, voire des années. La conception du système d'exploitation des capteurs doit en conséquence être robuste pour éviter le moindre bug qui paralyserait le réseau ou ferait consommer inutilement de l'énergie aux nœuds [BA11].

2.3 Propriétés principales

TinyOS a été conçu pour la programmation des réseaux de capteurs et il est caractérisé par :

- **Disponibilité et sources** : TinyOS est un système principalement développé et soutenu par l'université américaine de Berkeley, qui le propose en téléchargement sous la licence BSD et en assure le suivi.

- **Event-driven** : Le fonctionnement d'un système basé sur TinyOS s'appuie sur la gestion des évènements qui se produisent instantanément. Ainsi, l'activation de tâches, leur interruption ou encore la mise en veille du capteur s'effectue à l'apparition d'évènements, ceux-ci ayant la plus forte priorité. Ce fonctionnement basé sur les

évènements (Event-driven) s'oppose au fonctionnement dit temporel (time-driven) où les actions du système sont gérées par une horloge donnée.

- **Langage** : TinyOS a été programmé en langage NesC que nous allons détailler plus tard.

- **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 (sous forme d'un évènement) peut stopper l'exécution d'une tâche ;

- **Temps réel** : Lorsqu'un système est dit « temps réel » celui-ci gère des niveaux de priorité dans ses tâches permettant de respecter des échéances données par son environnement.

- **Consommation** : TinyOS a été conçu pour réduire au maximum la consommation en énergie d'un nœud capteur. Ainsi, lorsqu'aucune tâche n'est active, il se met automatiquement en mode veille [CB08].

Le Tableau 3.1 ci-dessous résume ses propriétés :

Propriété	Valeur pour
Type	Event-driven
Disponibilité	Open source
Langage	NesC
Préemptif	Non
Temps réel	Non
Source	Fournies

Tableau 3. 1: Propriété de TinyOS.

2.4 Primitives de TinyOS

TinyOS est un système d'exploitation dédié pour les réseaux de capture qui sont caractérisés par une limitation considérable des ressources c'est pour quoi il est impératif qu'il implémente des primitives aussi fines pour pouvoir gérer et fonctionner correctement sur ces réseaux

2.4.1 Gestion des processus

A. Ordonnancement : Nous allons détailler le fonctionnement précis de l'ordonnanceur TinyOS qui est au cœur de la gestion des tâches et des évènements du système. Le choix d'un ordonnanceur déterminera le fonctionnement global du système et le dotera de propriétés précises telles que la capacité à fonctionner en évènementiel. Un système d'exploitation doit percevoir l'écoulement du temps pour prendre ses décisions d'ordonnancement. Pour fournir au système d'exploitation une base de temps élémentaire, le système se base sur une horloge matérielle générant des événements Ticks à des intervalles réguliers. Ainsi, la résolution de l'échelle du temps est fonction de la fréquence de l'horloge. Comme la prise en compte de cette horloge est couteuse en traitements, la fréquence n'est, en pratique, pas très élevée (de l'ordre de dixièmes de milliseconde). Dans certains cas, le système utilise un compteur externe ou interne au processeur avec une résolution plus élevée (allant jusqu'à l'ordre des fractions de nanoseconde selon le type du capteur). Cette fonctionnalité permet une prise en compte plus fine des contraintes temporelles.

B. Communication et Synchronisation : Les notions de synchronisation et de communication sont étroitement liées et l'existence de l'une ne peut s'envisager sans faire référence à l'autre. En général, les processus d'une application n'évoluent pas de façon indépendante. La spécification de l'application fixe les relations logiques et temporelles entre ses processus.

Dans le but de synchroniser ces processus, on fait communiquer ces processus à l'aide de mécanismes offerts par le système. Parmi les mécanismes existants pour les systèmes Linuxien et spécialement TinyOS, on peut citer:

- **Communications par Messages:** Pour éviter les problèmes liés à l'exécution en parallèle des processus et résoudre les problèmes que pose la synchronisation, le mécanisme le plus simple est celui de la boîte aux lettres ou BAL (généralement implémentée avec une seule case mémoire ou avec plusieurs sous forme de file de message ou "queue"). Les échanges s'effectuent de façon indirecte, le message est placé en mémoire par l'émetteur et il est récupéré depuis cette case par le récepteur.

- **Communication par zone commune:** La façon la plus simple et la plus rapide de transmettre des données est de ne pas en transmettre. Il suffit de disposer d'une zone de

mémoire commune dans laquelle seront disposées les données en libre accès. Le problème que pose ce type de communication est surtout la synchronisation entre les processus.

Concernant la synchronisation, le système TinyOS fournit plusieurs outils, parmi lesquels on cite:

- **Synchronisation par événements:** le système TinyOS utilisent le mécanisme de drapeaux événements (ou event flags) pour signaler aux processus qu'un événement asynchrone (event) s'est produit. Le principe d'attente sur un événement est relativement simple. Si le drapeau correspondant est positionné lorsqu'arrive la requête d'attente, la tâche continue son exécution. Si le drapeau est à zéro, le processus est suspendu jusqu'à ce que le drapeau soit positionné. Un processus peut attendre sur plusieurs événements. Pour cela, il existe deux manières d'attendre: une synchronisation conjonctive fait reprendre le processus lorsque tous les événements signalés ont eu lieu; une synchronisation disjonctive fait reprendre le processus lorsque n'importe lequel des événements signalés a eu lieu.

- **Exclusion mutuelle** : utilisée pour le contrôle des ressources partagées du système. L'exclusion mutuelle empêche l'accès à une donnée, à un composant, ou à un périphérique si celui-ci est utilisé par un autre processus. Il existe plusieurs méthodes pour assurer l'exclusion mutuelle: Masquage des interruptions, Variables verrous, Sémaphores, etc... [KM10].

2.4.2 Gestion de la mémoire

Il est important de préciser de quelle façon un système d'exploitation aborde la gestion de la mémoire, d'autant plus lorsque ce système travaille dans un environnement aussi restreint.

Dans un système embarqué et/ou Temps Réel, l'allocation mémoire est généralement statique (une fois pour toute, au début du programme), cela à l'avantage de ne pas avoir des mauvaises surprises sur la disponibilité mémoire *Memory Overflow* et surtout, ne nécessite pas d'implémenter des algorithmes d'allocation mémoire compliqués nécessitant beaucoup de temps de calcul. Et ceci pour éviter le décalage temporel dû à l'allocation mémoire autant que possible, mais ceci n'est pas évident vue que les systèmes

sont embarqués et dotés d'une mémoire réduite qui ne peut pas accueillir le système et tous les programmes en même temps [KM10].

TinyOS occupe un espace mémoire très faible puisqu'il ne prend que 300 à 400 octets dans sa distribution minimale. En plus de cela, il est nécessaire d'avoir 4 Ko de mémoire libre qui se répartissent de la façon suivante:

- **La pile** : sert de mémoire temporaire au fonctionnement du système notamment pour l'empilement et le dépilement des variables locales.
- **Les variables globales** : réservent un espace mémoire pour le stockage de valeurs pouvant être accessible depuis des applications différentes.
- **La mémoire libre** : pour le reste du stockage temporaire.

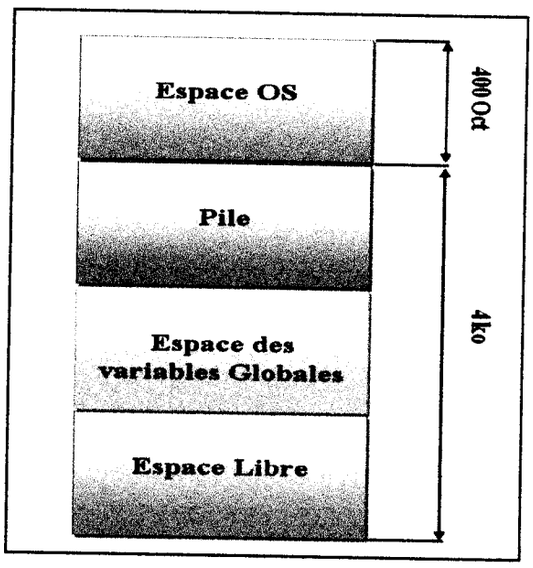


Figure 3. 2 : Structuration de la mémoire sous TinyOS

La gestion de la mémoire dans TinyOS possède de plus quelques propriétés. Le système opte pour la *flat memory* (Considérer l'espace mémoire comme une seule et même zone) au détriment de la protection vue que cette dernière engendre un surcoût lié à la vérification de chaque adresse accédée par le programme. La solution dans un système centralisé était de doter le matériel d'un composant MMU (*Memory Mangement Unit*) qui gère la protection mais comme il est gourmand en ressources, le système TinyOS en est dépourvu.

2.4.3 Gestion des Entrées/Sorties

Dans un système embarqué, le matériel et le logiciel sont intiment liés, ainsi le matériel et le logiciel ne sont pas aussi facilement discernables. Comme le système embarqué TinyOS est conçu de façon à fonctionner sur un grand nombre d'architectures, on a conçu ce qu'on appelle la HAL (*Hardware Adaptation Layer*). Cette couche encapsule toutes les fonctions dépendantes de l'architecture.

2.4.4 Gestion de la connectivité (Networking)

Le support de la connectivité dans les systèmes embarqués est très important car il leur permet de communiquer facilement avec l'extérieur mais aussi de les gérer, les mettre à jour, etc. à distance! Sans oublier que parfois, la connectivité (communication) est la fonction principale du système tel que les réseaux de capteurs, routeur, téléphone portable, PDA, etc.

TinyOS implémente de nombreux protocoles pour la communication réseaux et offre aux utilisateurs une palette de protocole de routage (protocole de routage proactif, réactif et hybride). Car le plus 70% des opérations exécuté par un capteur sont des routages de paquets.

2.5 Structure logicielle

Le système d'exploitation TinyOS s'appuie sur le langage NesC. Celui-ci propose une architecture basée sur des composants, permettant de réduire considérablement la taille mémoire du système et de ses applications. Chaque composant correspond à un élément matériel (LEDs, timer,...) et peut être réutilise dans différentes applications. Ces applications sont des ensembles de composants réutilisables et portables associes dans un but précis. Les composants peuvent être des concepts abstraits ou bien des interfaces logicielles aux entrées-sorties matérielles de la cible étudiée (carte ou dispositif électronique). L'implémentation de composants s'effectue en déclarant des tâches, des commandes ou des événements qui permettent de faire appel aux fonctionnalités d'autres composants auxquels ils sont liées :

- Les tâches sont utilisées pour effectuer la plupart des blocs d'instruction d'une application.

- Une commande permet l'exécution d'une fonctionnalité dans un autre composant.
- Les événements permettent de faire le lien entre les interruptions matérielles (pression d'un bouton, changement d'état d'une entrée,...) et les couches logicielles que constituent les taches.

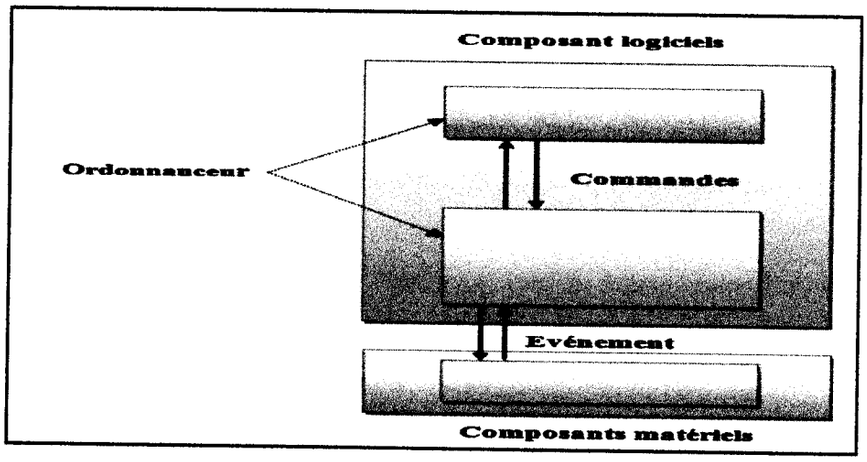


Figure 3. 3: Schéma des interactions internes au système TinyOS

2.6 Cibles possibles pour TinyOS

Il existe de nombreuses cibles possibles pour ce système d'exploitation embarqué. Malgré leurs différences, elles respectent toutes globalement la même architecture basée sur un noyau central autour duquel s'articule les différentes interfaces d'entrée-sortie, de communication et d'alimentation. La figure 3.4 représente cette architecture.

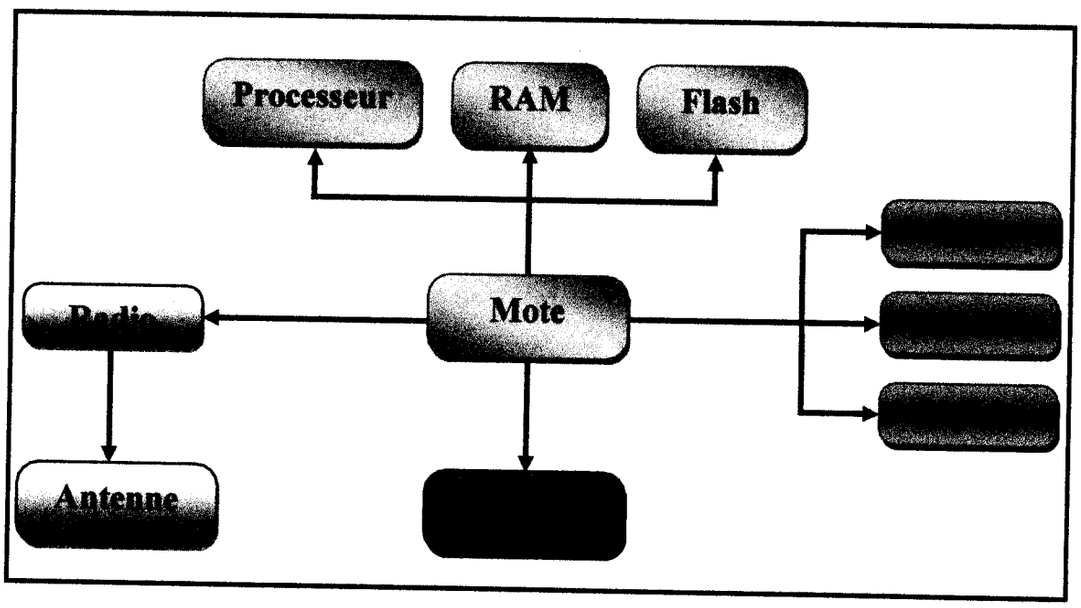


Figure 3. 4: Architecture générale des cibles utilisant TinyOS

3.2 Les Principales caractéristiques de NesC

NesC est constitué d'interfaces et de composants. Une interface peut être utilisée ou peut être fournie. Les composants sont des modules ou des configurations.

Une application est un ensemble de composants, regroupés et reliés entre eux (voir figure 3.5). Les interfaces sont utilisées pour les opérations qui décrivent l'interaction bidirectionnelle. Le fournisseur de l'interface doit mettre en application des commandes, alors que l'usager de l'interface doit mettre en application des événements.

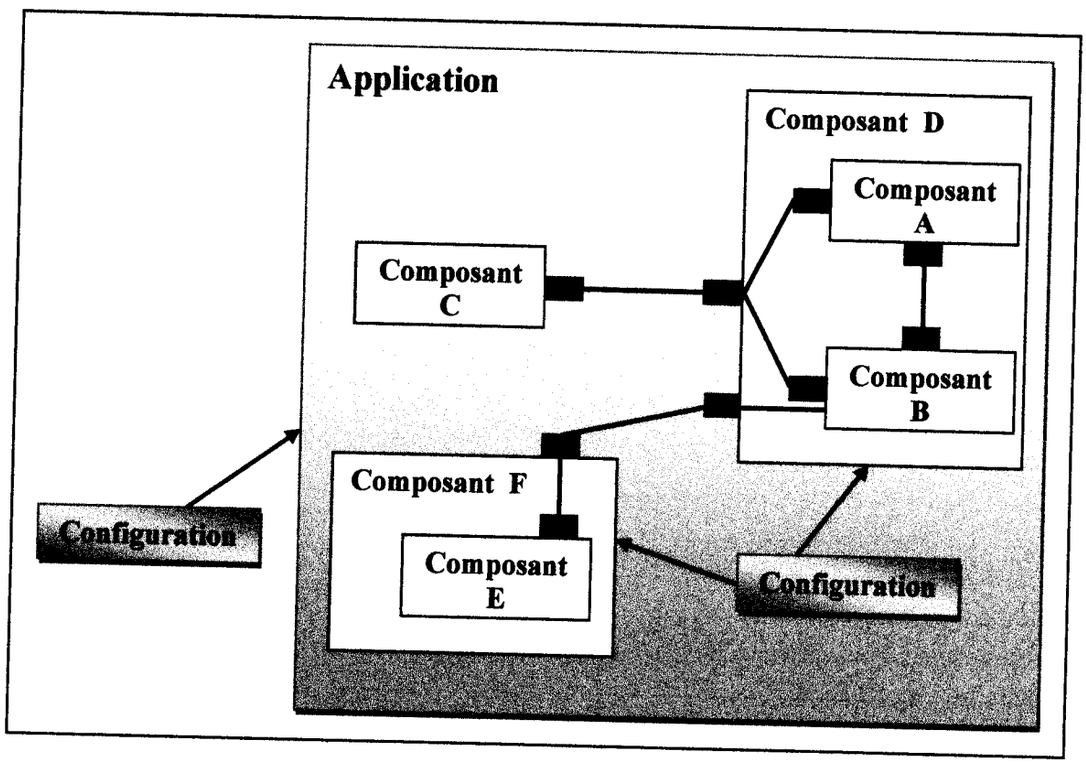


Figure 3. 5 : Architecture d'une application NesC.

Deux types de composants existent :

1. **Les modules**, qui mettent en application des spécifications d'un composant.
2. **Les configurations**, qui se chargeront d'unir différents composants en fonction des interfaces (commandes ou événements). La figure suivante (figure 3.6) montre un diagramme de blocs dans lequel est décrit le processus de compilation pour une application TinyOS écrite en NesC [HS08].

3.3 Les fichiers dans NesC

Les fichiers dans NesC sont classés en trois types : Interfaces, modules et configurations.

- **Interface** : Ce type de fichier déclare les services fournis et les services qui seront utilisés. Ils se trouvent dans le répertoire /tos/interface. (exemple : StdControl.nc).
- **Module** : Le type Module contient le code de l'application, en mettant en œuvre une ou plusieurs interfaces.
- **Configuration** : Dans ces fichiers est déclarée la manière d'unir les différents composants et comment effectuer le contrôle des flux.

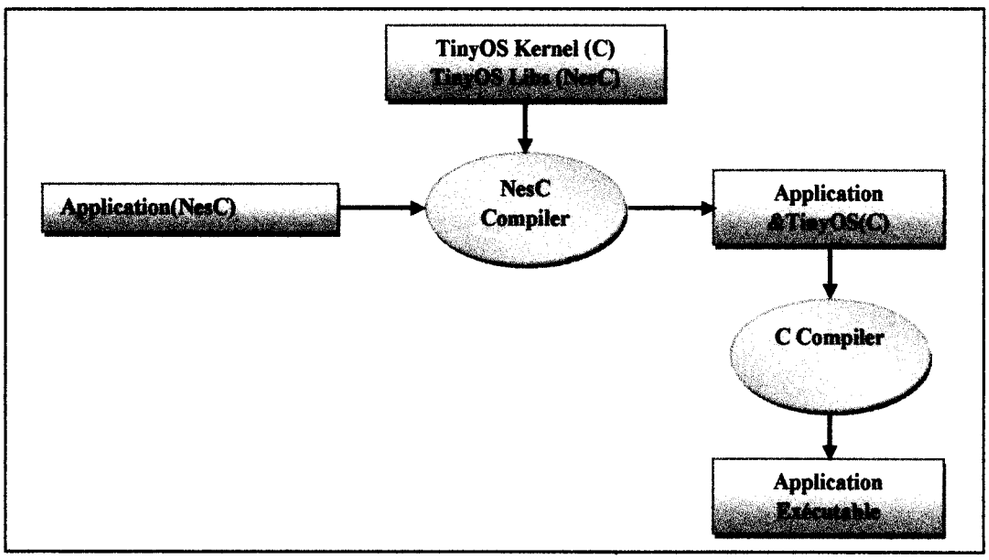


Figure 3. 6 : Etapes de compilation d'une application NesC.

3.4 Construction d'une application en NesC

Toutes les applications ont besoin d'un fichier de configuration de haut niveau qui est normalement nommé d'après l'application elle-même. Dans ce cas, **NomApplication.nc** est le fichier de configuration de l'application et le fichier source que le compilateur nesC utilise pour générer l'exécutable. **NomApplicationM.nc** quant à lui correspond à l'implémentation à proprement parler de l'application. Comme nous l'avons expliqué dans la partie précédente, les composants nesC peuvent se relier les uns aux autres, et ici c'est le fichier **NomApplication.nc** qui permet de faire cette connexion entre le module **NomApplicationM.nc** et les autres composants auxquels l'application fait appel.

Dans une application, nous avons donc une configuration et un module qui vont ensemble. La convention de TinyOS veut alors que **NomApplication.nc** représente la configuration et **NomApplicationM.nc** représente le module correspondant.

- **Le fichier de configuration « NomApplication.nc »**

NomApplication.nc

```
configuration NomApplication {
    }
    implementation {
        components .....................................................;
    }
```

La première chose à noter est le mot clé « configuration » qui indique qu'il s'agit d'un fichier de configuration dont le nom est « **NomApplication** ». A l'intérieur des accolades, il est possible de spécifier des interfaces requises ou offertes par la configuration, tout comme on le fait à l'intérieur d'un module.

La véritable configuration est implémentée au sein des deux accolades qui suivent le mot clé « **implementation** ».La ligne « **components** » spécifie les différents composants auxquels la configuration fait référence. Le reste de l'implémentation consiste à connecter les interfaces utilisées par certains composants aux interfaces fournies par les autres.Le composant Main est le premier exécuté dans une application TinyOS. Pour être plus précis, la commande **Main.StdControl.init()** est la première à être exécuté suivi de **Main.StdControl.start()**. Donc toute application TinyOS doit avoir un composant **"Main"** dans sa configuration. StdControl est une interface de base utilisée afin d'initialiser et de démarrer des composants TinyOS. Regardons le fichier qui la définit (**StdControl.nc** qui se trouve dans le répertoire **tos/interfaces/** de TinyOS):

```
interface StdControl {

    command result_t init();
    command result_t start();
    command result_t stop();
}
```

Nous pouvons voir que « StdControl » définit trois commandes, « init() », « start() » et « stop() ». La première est appelée quand un composant est initialisé pour la première fois, « start() » quand il est démarré, c'est à dire exécuté pour la première fois et « stop() » quand le composant est arrêté, par exemple pour éteindre le dispositif physique qu'il contrôle.Un appel à une de ces commandes dans un composant doit être répercuté dans tous les composants auxquels il se rattache.

- **Le module « NomApplicationM.nc »**

```
includes .....;
    module NomApplicationM {
        provides        {
            interface StdControl;
        }
        uses {
            interface Timer ;
            .....    }
    }
    implementation {
        ..............................;    }
```

La première ligne nous est relativement familière, c'est l'inclusion d'un fichier d'en tête. La deuxième signale qu'il s'agit d'un module appelé « NomApplicationM » et déclare les interfaces fournies et utilisées. Nous voyons qu'il fournit l'interface « StdControl » par exemple. Cela veut dire que ce module implémente cette interface. Comme expliqué plus haut, cela est nécessaire pour initialiser et démarrer le composant. Il utilise ensuite plusieurs interfaces: par exemple : le Timer qui comme son nom l'indique permet de contrôler des timers.Ces déclarations donnent accès à « NomApplicationM » aux commandes de ces interfaces, et l'oblige à implémenter les événements (events) déclarés dans ces interfaces.

```
interface Timer {
command result_t start(char type, uint32_t interval);
command result_t stop();
event result_t fired();
                }
```

La commande start() est utilisée pour définir le type de timer ainsi que l'intervalle à laquelle le timer expire (exprimé en milliseconde). Les deux types valides de timer sont « TIMER_REPEAT » et « TIMER_ONE_SHOT ». Comme leur nom l'indique, le premier continue indéfiniment jusqu'à ce qu'on l'arrête grâce à la commande « stop() », et le second s'arrête dès que l'intervalle de temps expire.

A chaque période, le timer envoie un « event » qui est « attrapé » par l'application grâce à l'implémentation de la fonction « fired() ».

En d'autres termes, un event est une fonction que l'implémentation d'une interface va signaler quand un certain événement arrive. Dans ce cas, l'event « fired() » est signalé à chaque fois que l'intervalle est révolue.

3.5 TOSSIM : le simulateur de TinyOS

TOSSIM est un simulateur discret basé sur la programmation par événement et qui a été conçu et désigné pour simuler les réseaux de capteurs qui utilise la plateforme TinyOS.

Le principale but de TOSSIM est de créer une simulation très proche de ce qui ce passe dans ces réseaux dans le monde réel. TOSSIM simule le comportement des applications de TinyOS à un niveau très bas. Le réseau est simulé au niveau des bits et chaque interruption dans le système est capturée.

TOSSIM fourni deux modèles de radios pour la communication. Le modèle par défaut est celui « simple ». Les paquets sont transmis dans le réseau avec aucune erreur et ils sont reçus par chaque nœud. Avec ce modèle il est ainsi possible que deux nœuds différents peuvent envoyer un paquet en même temps avec la conséquence que ces deux paquets seront alors détruit à cause du chevauchement des signaux. Le deuxième modèle est le modèle « lossy ». Dans ce modèle les nœuds sont placés dans un graphe direct formé d'un couple (a, b) ce qui signifie qu'un paquet envoyé par le nœud a peut être reçu par le nœud b. TOSSIM est équipé aussi d'un simulateur graphique TinyViz. Cette application est équipée par plusieurs API plug-ins qui permet d'ajouter plusieurs fonctions à notre simulateur comme par exemple contrôler les entrées de notre radio ou bien suivre la dépense d'énergie en utilisant un autre simulateur qui s'appelle PowerTOSSIM [WZ06].

3.6 TinyViz

Pour une compréhension moins complexe de l'activité d'un réseau, TOSSIM peut être utilisé avec une interface graphique, TinyViz, permettant de visualiser de manière intuitive le comportement de chaque capteur au sein du réseau.

TinyViz est une application graphique qui donne un aperçu de notre réseau de capteurs à tout instant, ainsi que des divers messages qu'ils émettent. Il permet de déterminer un délai entre chaque itération des capteurs afin de permettre une analyse pas à pas du bon déroulement des actions [MB06].

4. Conclusion

Le système TinyOS est un système embarqué, développé par l'Université de Berkeley, dédié spécialement pour les réseaux de capteurs, qui présente un environnement de calcul particulier soumit a plusieurs contraintes tel que l'énergie, l'espace mémoire, etc. TinyOS permet le développement d'application simplifié par la mise en relation de composants cibles, tout en offrant des primitives ainsi que des bibliothèques réseaux qui le rendent complet. Il est classé parmi les systèmes temps réel mou, non préemptif, événementiel. TinyOS a un grand avenir dans les systèmes embarqué garce à ses qualités qui séduisent plusieurs développeurs qui participent dans son développement vu qu'il est open source.

Dans ce chapitre, nous avons étudié l'architecture du système TinyOS ; nous avons examiné son mode d'exécution et son mode d'ordonnancement des différents types d'action qui sont évènement, commande et tâche. Nous avons aussi présenté le langage NesC ainsi que le simulateur TOSSIM.

Dans le Chapitre suivant, nous allons modéliser et implémenter notre application pour la détection des feux de forêt sur l'environnement TinyOS et le simulateur TOSSIM en utilisant le langage NesC.

Chapitre IV
Implémentation et résultats

1. Introduction

Avant sa mise en place, le déploiement d'un réseau de capteurs nécessite une phase de test afin de s'assurer du bon fonctionnement de tous ses composants; soft ou hard. Pour se faire, il existe à ce jour trois grandes solutions: l'expérimentation réelle, la simulation et l'émulation. Les deux dernières solutions sont d'autant plus importantes que l'expérimentation en environnement réel qui est très coûteuse et lourde à mener: mettre en place un réseau multi sauts, connaître sa topologie dans le temps, lancer des expérimentations, récupérer et fusionner les traces demandent un investissement important en matériels, en temps et en main d'œuvre. La simulation est donc une étape incontournable, notamment pour faire des évaluations sur des réseaux à large échelle; comme peuvent l'être certains réseaux de capteurs par exemple et réduire le maximum d'erreurs de conception possibles.

Dans ce chapitre, nous commencerons par définir les outils nécessaires pour la réalisation de notre projet. Ensuite, nous décrirons d'une manière générale les étapes d'implémentation de l'application ainsi que toutes les structures de données, algorithmes et processus décrits lors de la conception. Par la suite, nous exposerons l'interface graphique de notre plateforme en expliquant toutes les fonctionnalités offertes à l'utilisateur. Finalement, nous évaluerons la performance de notre application en utilisant la métrique du taux de détection et nous présenterons les différents scénarios de simulations adoptés ainsi que l'interprétation des résultats obtenus à l'issue de ces simulations.

2. Différentes approches de test

Que ce soit pour un protocole ou une application, le développeur est confronté aux mêmes problèmes de test et d'évaluation que dans les réseaux filaires. En effet, lors du développement d'un protocole ou d'une application répartie, il est nécessaire d'évaluer le comportement des différents mécanismes proposés, de pouvoir les comparer avec ceux existants et mesurer les performances dans des conditions de réseaux particulières. Pour les réaliser, il est possible d'utiliser plusieurs méthodes: le test en environnement réel, la simulation et l'émulation [Con 06].

2.1. Test en environnement réel

La première solution consiste à utiliser un réseau de capteurs réel pour bien mener son évaluation. Bien qu'elle permette d'avoir des conditions parfaitement réalistes, cette solution n'est pas complètement satisfaisante car elle ne permet pas de contrôler l'ensemble des paramètres de l'environnement. De plus, cette solution ne permet pas de reproduire plusieurs fois de suite les mêmes conditions avec précision.

2.2. Simulation

La seconde solution, souvent privilégiée par les chercheurs, est la simulation. Celle-ci permet d'évaluer un modèle d'application ou de protocole dans un environnement entièrement contrôlable. Pour cela, la simulation s'appuie sur des modèles décrivant l'environnement, des modèles décrivant les couches de communication utilisées par les terminaux sans fil ainsi que d'autres équipements du réseau et un modèle du trafic circulant sur le réseau. Cependant, la simulation ne travaille pas en temps réel ce qui empêche par exemple l'évaluation d'applications interactives.

2.3. L'émulation

L'émulation peut être vue comme un compromis entre les deux solutions précédentes en permettant d'évaluer un protocole ou une application dans un environnement contrôlable et reproductible qui simule en temps réel les conditions telles que: les débits, les délais et les pertes que l'on observe dans le réseau cible. Pour cela, l'émulation va simuler les effets des couches basses de telle manière qu'un protocole ou une application s'exécute dans les mêmes conditions que celles de l'environnement réel.

3. Outil de programmation et environnement de simulation

Dans cette section, nous présentons les outils utilisés pour la mise en œuvre de notre application pour la détection des feux de forêt. Nous commençons tout d'abord par TinyOS, le système d'exploitation conçu pour les RCSF. Nous parlons ensuite du langage de programmation NesC avec lequel nous avons programmé le codes de l'application. Nous terminons cette partie par la présentation d'un simulateur des RCSF: TOSSIM qui offre l'interface graphique TinyViz pour visualiser le déroulement de la simulation.

3.1. Choix de système d'exploitation

Les systèmes d'exploitation pour les nœuds capteurs sont généralement moins complexes que les autres systèmes. Plusieurs systèmes d'exploitation ont été proposés pour les RCSF parmi lesquels on trouve SOS [CEM05], Contiki [ABT04], MANTIS [CCJ06]. TinyOS reste néanmoins le plus répandu pour les RCSFs car il répond aux exigences particulières des applications des RCSF (voir chapitre III).

3.2. L'environnement de développement

NesC est un langage de programmation orienté composants syntaxiquement proche du langage C. Il est conçu pour la réalisation des systèmes embarqués distribués, en particulier, les RCSF. [HJK08]

3.3. L'environnement de simulation

TOSSIM est le simulateur de TinyOs. Il permet de simuler le comportement d'un capteur (envoie/réception de messages via les ondes radios, traitement de l'information, etc...) au sein d'un réseau de capteurs. Pour une compréhension moins complexe de l'activité d'un réseau, TOSSIM peut être utilisé avec une interface graphique TinyViz, voir la Figure 4.1 qui permet de visualiser de manière intuitive le comportement de chaque capteur au sein du réseau.

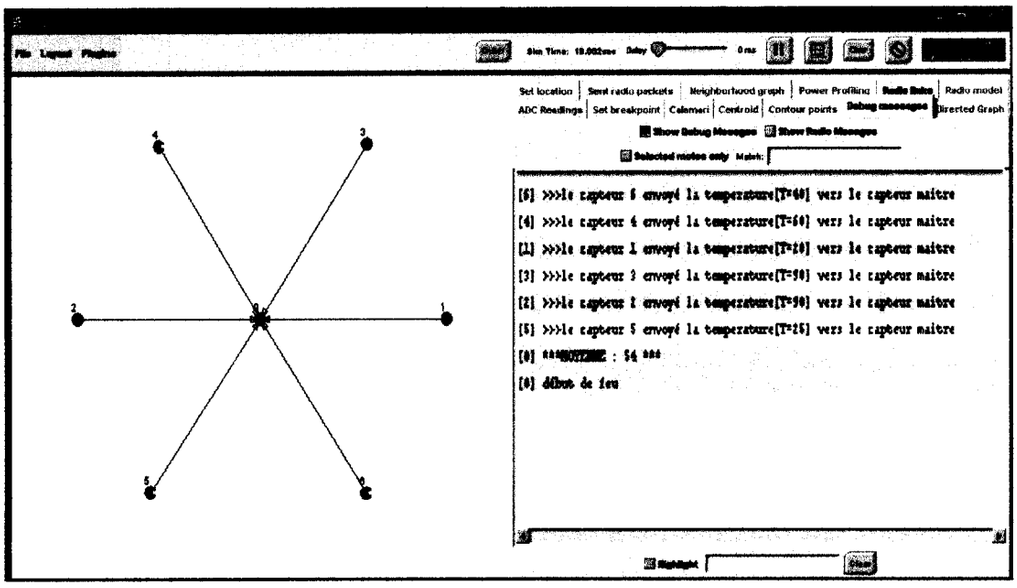


Figure 4.1 : Fenêtre graphique TinyViz

Cependant, malgré nos recherches, nous n'avons pas trouvé comment deux capteurs peuvent exécuter deux applications différentes. En effet, il serait plus intéressant de distinguer les capteurs maîtres des capteurs esclaves, en leur demandant d'exécuter des codes différents. Pour pallier ce problème, nous avons eu recours à une astuce. Chaque capteur est identifiable par *un numéro unique* (dans la plateforme) nous arrivons donc à différencier le rôle de chaque capteur par son numéro [FA08].

4. Modèle du système (réseau)

Le principe de base pour détecter les incendies en utilisant les RCSF consiste à déployer des capteurs pour couvrir une zone cible. Ces capteurs devraient être capables de recueillir des données de type température. Généralement, trois types de capteurs sont utilisés. Le premier type s'occupe de recueillir les données concernant le milieu où ces capteurs sont déployés et de les envoyer au deuxième type. Le deuxième type de capteur sont les sinks et ont pour rôle de transmettre les données traitées à la station de base. Le troisième type de capteur est connectée directement à un ordinateur via un port USB (Universal Serial Bus). Son rôle est de transmettre les données reçues à l'ordinateur.

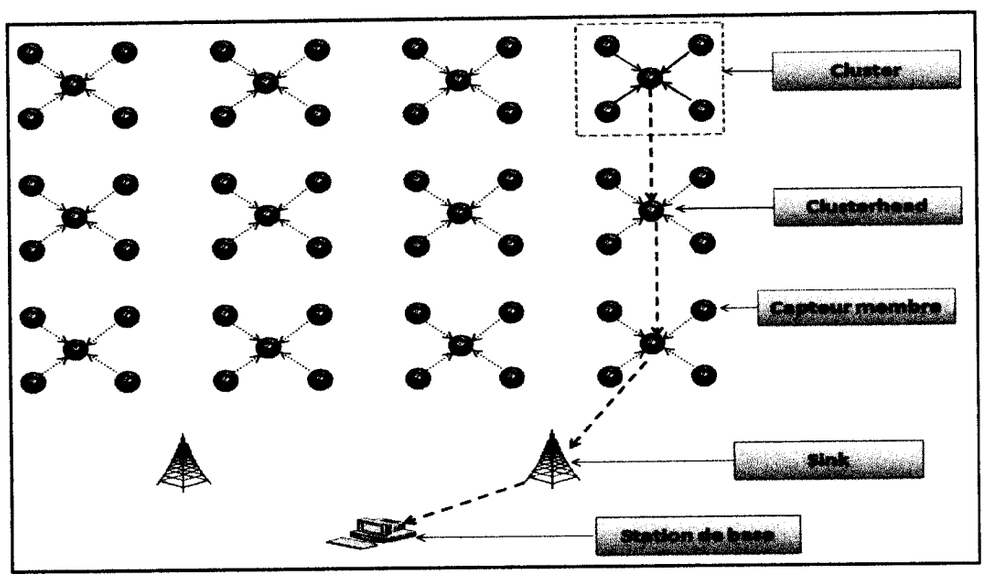


Figure 4.2 : Topologie du réseau

Nous avons effectué une simulation sur un ensemble de nœuds déployés sur une surface de 100m X 100m, le réseau est partitionné en 12 zones, chaque zone a un identifiant, composé de 4 capteurs membres qui ont pour rôle de recueillir la température

et un capteur maitre qui reçoit les données captées, les traite et les envoie à une station de base passant par une passerelle (sink).

L'information envoyée par le maitre de chaque zone, comporte la moyenne de la température, la médiane de la température et le numéro de la zone. Cette information doit être acheminée jusqu'à la station de base.

4.1. Modèle statistique considéré (fonction d'agrégation)

L'agrégation de données dans les réseaux de capteurs consiste à remplacer les lectures individuelles de chaque capteur par une vue globale, collaborative sur une zone donnée (clustering). On peut utiliser par exemple de simples fonctions d'agrégat telles que MIN, MAX ou MOYENNE, MEDIANE qui permettent à partir d'une série de n messages reçus par un « chef de zone » de ne renvoyer vers le puits qu'un seul message résumant l'information contenue dans ces n messages. Ceci réduit le nombre de messages envoyés et donc économise l'énergie.

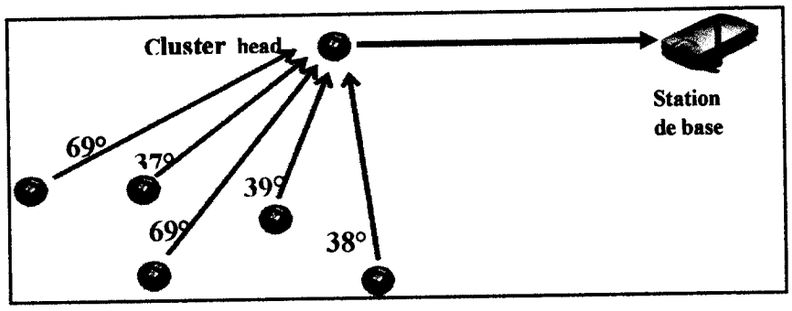


Figure 4.3 : Agrégation de données (température)

Dans notre application, nous avons utilisés deux fonctions statistiques pour agréger les données envoyées par les capteurs :

La moyenne : elle exprime la grandeur qu'auraient chacun des membres de l'ensemble s'ils étaient tous identiques sans changer la dimension globale de l'ensemble.

La médiane : la médiane d'un ensemble de valeurs est la valeur *m* telle que le nombre de valeurs de l'ensemble supérieures ou égales à *m* est égal au nombre de valeurs inférieures ou égales à *m*. Intuitivement, on peut dire que la médiane est le point milieu de l'ensemble, qu'elle divise en deux moitiés.

Dans l'exemple précédant illustré dans la Figure 4.3 :

La moyenne des valeurs collectées (37°,38°,39°,69°,69°)=50°

Alors que la médiane =39° (état normal)

5. Implémentations et déroulements

5.1. Implémentation

Dans cette section, nous décrivons les structures de données ainsi que les principaux commandes et événements nécessaires pour l'implémentation de notre application.

5.1.1. Structures de données

Le paquet dans TinyOS est envoyé dans une structure appelée TOS_Msg, qui est contenue dans un champ « int8_t data[TOSH_DATA_LENGTH] ». Les structures de données du paquet diffèrent selon le rang du nœud (puits, CH ou membre).

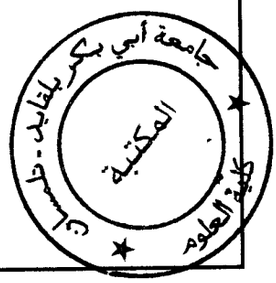
- **Le nœud cluster-head (CH)**

```
typedef struct MaitreMessage {
   uint16_t moyenne;
   uint16_t mediane ;
   uint16_t zone ;

} MaitreMessage;

enum {

  AM_SENSE_MESSAGE = 4
};
```



- **Le nœud membre**

```
typedef struct MembreMessage {

   uint16_t valeur;
   uint16_t id ;

} MembreMessage;

enum {
  AM_SENSE_MESSAGE1 = 4
};
```

5.1.2. Evénements et commandes

➢ **Les événements**

Nous citons ici les principaux événements utilisés pour l'implémentation de l'application.

Evénement	Sortie	Fonction TinyOS
Timer.fired()	result_t	lance le timer
ADC.dataReady(uint16_t x)	result_t	Captage de l'événement
SendMessageMaitre.sendDone	result_t	Affirmation de l'émission (maitre)
SendMessageMembre.sendDone	result_t	Affirmation de l'émission (membre)
ReceiveMessageMembre.receive	TOS_MsgPtr	reception message member
ReceiveMessageMaitre.receive	TOS_MsgPtr	reception message maitre

➢ **Les commandes**

Commande	Sortie	Fonction pour
StdControl.init()	result_t	Commande initialisation
StdControl.start()	result_t	Commande démarrer
StdControl.stop()	result_t	Commande arrêter

➢ **Les configurations**

- **Main**: permet de démarrer l'exécution de l'application.
- **GenericComm**: c'est l'interface responsable de la communication, elle s'occupe des envois et des réceptions des messages.
- **TimerC**: nous donne accès à un timer pour déclencher des évènements spécifiques.
- **LedsC**: nous permet de contrôler le fonctionnement des LEDs du capteur en adaptant leurs clignotements au fonctionnement de l'application .
- **DemoSensorC** : l'interface de captage .

➢ **Compilation**

Les capteurs possèdent en général la même architecture et sont programmables via le langage NesC. Cependant plusieurs firmes fabriquent de tels capteurs, ce qui implique quelques différences qui sont palliées par le compilateur de NesC appelé ncc. En effet celui-ci propose une compatibilité pour les plates formes de capteurs les plus répandues. Pour effectuer la compilation, les deux fichiers (fichier module et fichier configuration Annexe A), doivent se situer dans le même répertoire contenant aussi un makefile de la forme :

```
COMPONENT=DALTA //nom de l'application
include ../Makerules
```

Ce Makefile permet de compiler le composant température en spécifiant en paramètre la Plateforme sur laquelle doit fonctionner l'application. Par exemple, pour un capteur de type mica2, la commande permettant de compiler l'application sera : **make mica2**. Le compilateur ncc offre aussi la possibilité de pouvoir compiler l'application pour l'utiliser sur TOSSIM, le simulateur de TinyOS. Dans ce cas là, la commande sera : **make pc**. Cette commande génère un exécutable « main.exe » dans l'arborescence **/repertoire_courant/build/pc.**

5.2. Déroulement

Dans cette partie, nous expliquons et déroulons les phases de notre application en faisant appel à TinyViz. A propos des captures d'écran de ce dernier, nous nous limitons, dans cette étape, sur la partie où l'on peut visualiser les capteurs.

5.2.1. Formation de groupes(les clusters)

L'approche de clustering consiste à partitionner le réseau en un certain nombre de clusters, et former une topologie virtuelle. Les clusters sont généralement identifiés par un nœud particulier appelé cluster-head. Ce dernier permet de coordonner entre les membres de son cluster, d'agréger leurs données collectées et de les transmettre à la station de base.

Il existe plusieurs algorithmes de clustering mais dans notre cas, nous supposons que les groupes sont déjà formés. En effet, le problème de clustering ne fait pas partie de notre travail (out of scope).

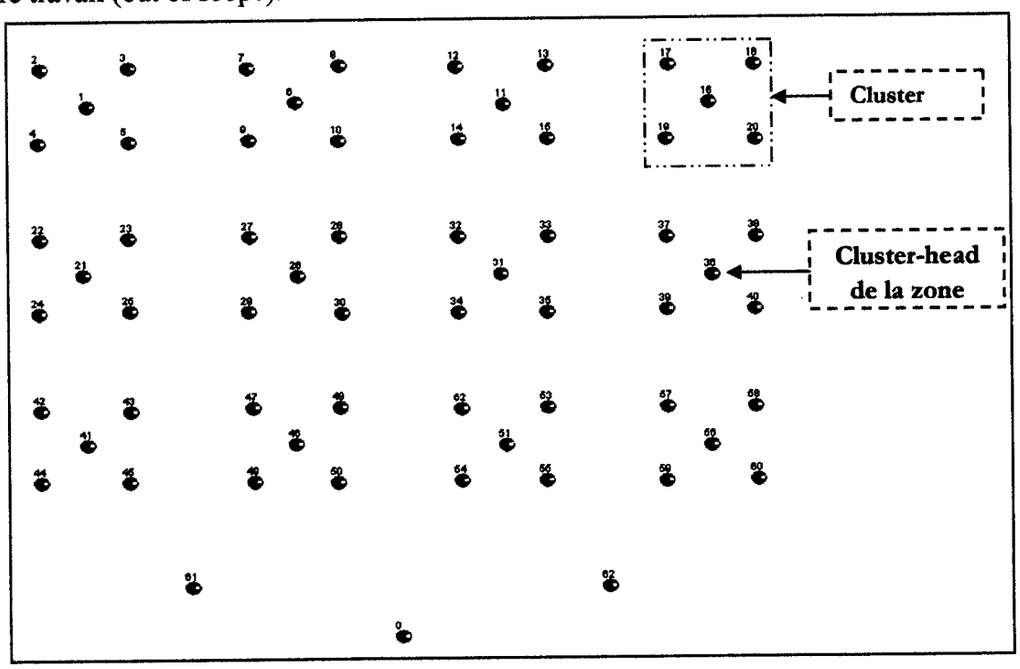


Figure 4.4 : Formation de groupes

La figure 4.4 présente le déploiement de notre réseau. Les capteurs sont organisés dans des clusters (12 Zones). Chaque capteur est rattaché a un chef de cluster (cluster-head).

5.2.2. Collecte des données et envoi de température

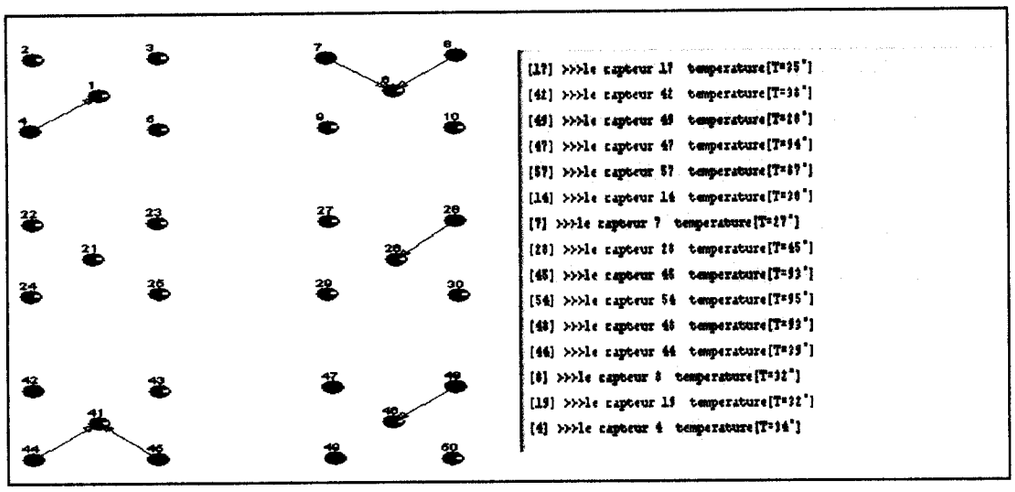


Figure 4.5 : Collecte des données

La figure 4.5 représente quelques transmissions unicast de capteur membre vers le responsable de la zone, Chaque membre capte la température et attend le début de son slot pour qu'il puisse l'envoyer à son CH. Dans cette phase les leds des capteurs qui captent une température supérieure à 50° s'allument en rouge et les autres (inferieur à 50°) s'allument en vert.

5.2.3. Traitement des données (l'agrégation)

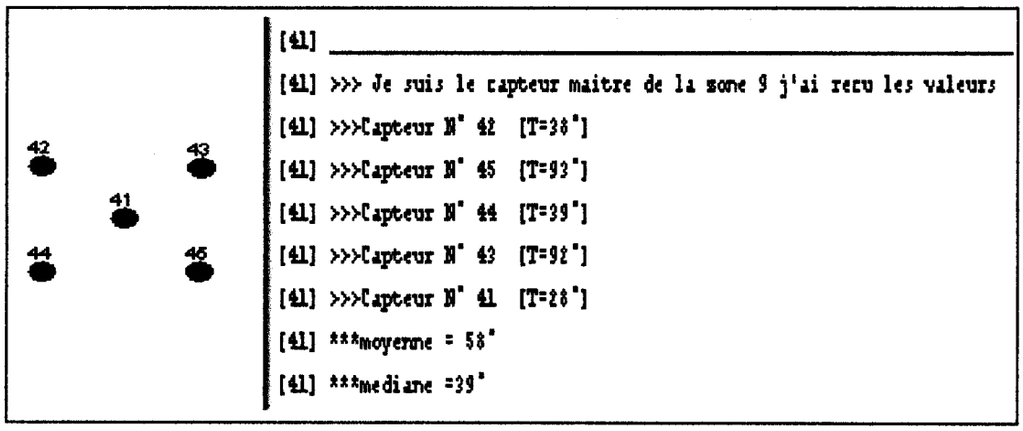


Figure 4.6 : Le traitement des températures reçues.

La figure 4.6 représente le traitement des températures (Zone N°9) reçu par le maitre N°41 , Le traitement des températures captées se fait au niveau du maitre de chaque zone, en utilisant les deux fonctions statistiques (moyenne et médiane).Les valeurs reçues sont : [28°,38°,39°,92°,93°] alors la moyenne égale à 58°(début de feux) et la médiane égale 39° (état normale).

Nous avons utilisé deux fonctions statistiques, à savoir la moyenne et la médiane ; ceci dans le but de choisir celle qui est plus robuste et qui génère le moins de fausses alarmes

4.2.4. Envoi des résultats d'agrégation des températures à la station de base et affichage de résultat finale

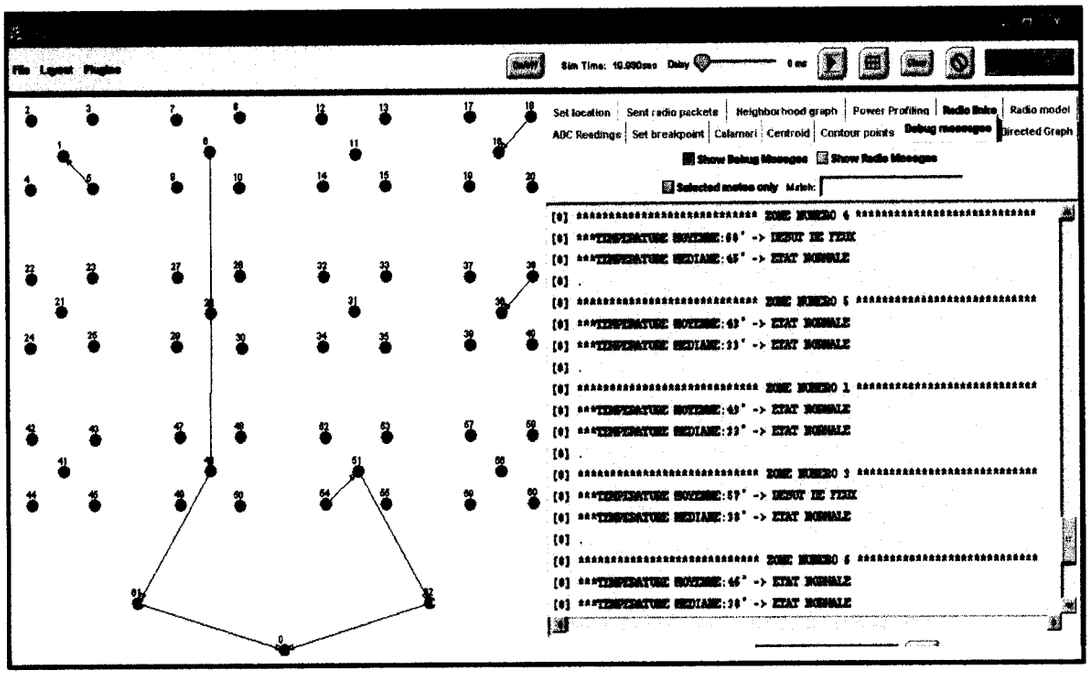


Figure 4.7 : Envoi des résultats d'agrégation à la station de base

La figure 4.7 représente l'envoi du résultat d'agrégation (zone 2 et 11) vers la station de base. Cette phase consiste à envoyé le résultat obtenu par le maitre et qui représente la température moyenne de cette zone à la station de base passant. Cette dernière teste chaque résultat reçu (température moyenne de chaque zone) avec le seuil (50°) et enfin afficher l'état des 12 zones.

6. Développement d'une interface utilisateur

Dans cette partie, nous allons montrer les différentes fonctionnalités de notre application java. Cette dernière est capable d'interagir avec un réseau de capteurs pour récupérer les différents évènements qui s'y produisent.

Pour concevoir une application qui communique avec un réseau de capteurs, on utilise généralement JAVA. En effet, plusieurs applications intéressantes incluses dans TinyOs sont écrites à l'aide de ce langage. On peut citer comme exemple : TinyViz, Oscilloscope, SerialForwarder,... etc.

Nous avons donc tenté de créer notre propre application pour communiquer avec un réseau de capteurs. Pour expliquer ce mécanisme, on doit d'abord évoquer un certain principe : Quand on veut communiquer avec un capteur réel on doit passer par une application intégrée dans TinyOs qui est le « *serialforwarder* ». Ce dernier a pour but de connecter le port série du capteur à une ou plusieurs applications comme illustré dans le schéma suivant :

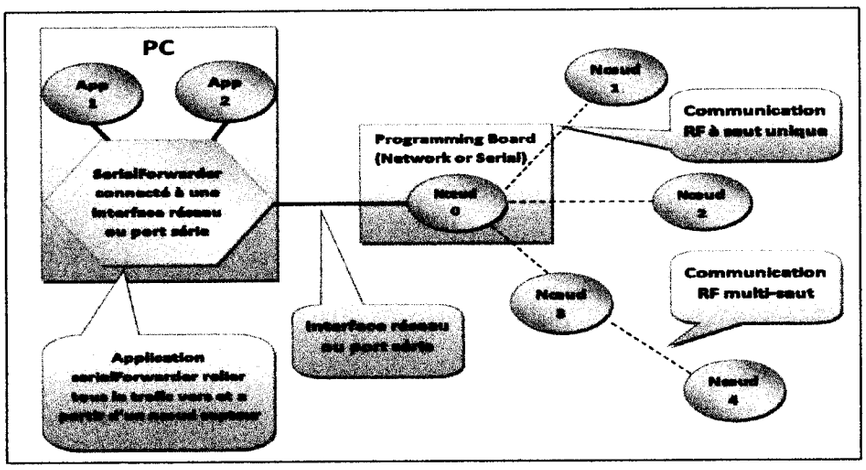


Figure 4.8 : Schéma de communication entre une application java et un réseau de capteurs

On désire suivre le même résonnement pour une simulation. Dans ce cas, l'application est connecté à l'aide d'un port série virtuel au nœud 0 (le puits) du réseau. On peut relier autant d'application que l'on souhaite. Cependant, dans notre cas, on n'en utilise qu'une seule **[CS07]**. Le schéma suivant explique le fonctionnement de cette solution :

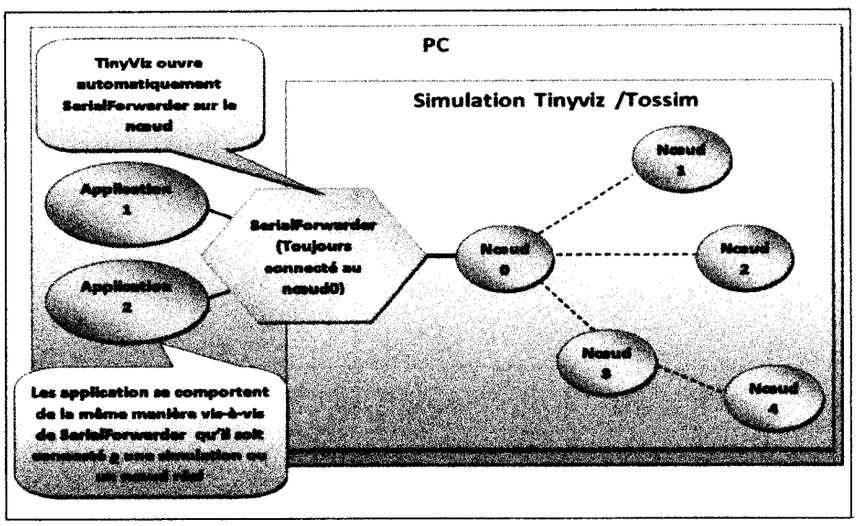


Figure 4.9 : Schéma de communication entre une application java et Tossim ou TinyViz

En appliquant ce dernier schéma à notre cas, on se retrouve avec l'application suivante qui fonctionne en interaction avec TinyViz et SerialForwarder.

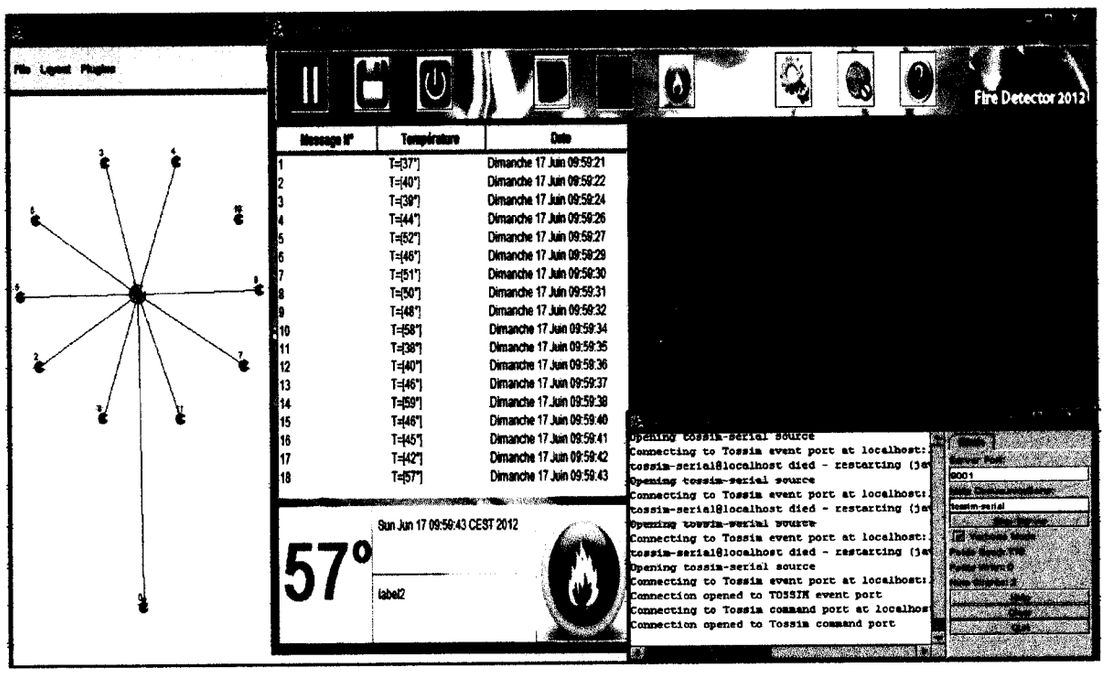


Figure 4.10 : Interaction TinyViz, SerialForwarder et notre application

Ainsi, nous avons la possibilité d'avoir une véritable communication avec le réseau (réel ou simulé) en lui envoyant des messages et en récupérant ceux qui transite par le nœud puits. Cependant, pour réaliser cet échange, on doit traduire les messages (les structures de données) écrits en NesC vers le java. Pour faire face à ce problème un outil de conversion a été créé: le MIG (**java** message interface génerator for nesC). Ce dernier transforme une structure NesC en code java.

Figure 4.11 : java message interface générator

5.1. Fonctionnement de l'application

Comme expliqué auparavant, notre application permet d'interagir avec le réseau de capteurs. Dans cette section, on va lister toute les fonctionnalités que nous avons implémentées :

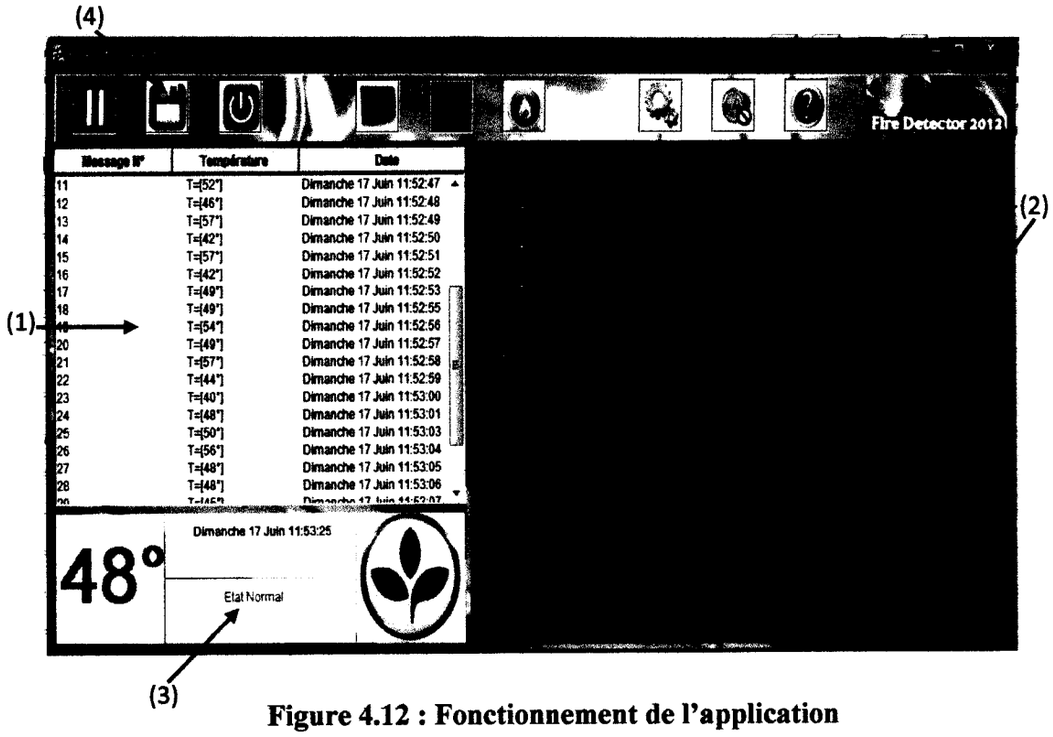


Figure 4.12 : Fonctionnement de l'application

1 : Afficher les différents messages (numéro du message, températures captées, date).

2 : Ce graphe affiche le changement de la température dans le réseau.

3 : permet d'indiquer l'état de la zone (un incendie ou état normale).

4 : Enregistrer les résultats dans un fichier.

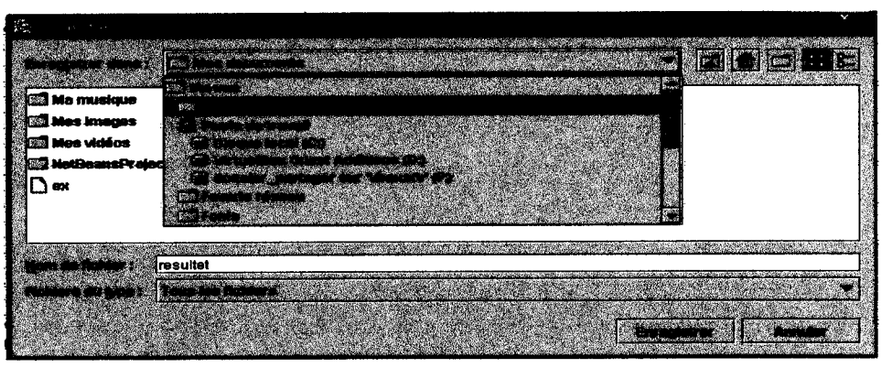


Figure 4.13 : Enregistrer les résultats dans un fichier

7. Evaluation des performances

7.1. Métrique considérée

Dans notre simulation, nous nous intéresserons essentiellement au *taux de détection* de notre application puisqu'elle constitue le paramètre le plus critique dans l'évaluation de performance d'un système de détection d'un incendie.

- **Taux de détection**

Le but principal de cette application revient finalement à minimiser les dégâts des feux de forêt, en détectant le plus d'incendies possible avant qu'elle propage complètement. Pour cela, il est nécessaire d'évaluer cette métrique qui représente le pourcentage des feux que le réseau peut détecter avec succès.

7.2. Résultats de simulation

7.2.1. Taux de détection dans le cas idéale (pas de valeurs aberrants)

Nous supposons que tous les capteurs du réseau fonctionnent correctement. Les résultats de post-simulation sont récapitulés dans le tableau 4.1 ci-dessous :

Taille du cluster :	10	20	30	40	50	60
médiane						
moyenne	100%	100%	100%	100%	100%	100%

Tableau 4. 1: Résultat taux de détection (cas idéale).

Les résultats du Tableau 4.1 sont représentés dans le graphe suivant de la figure 4.14 :

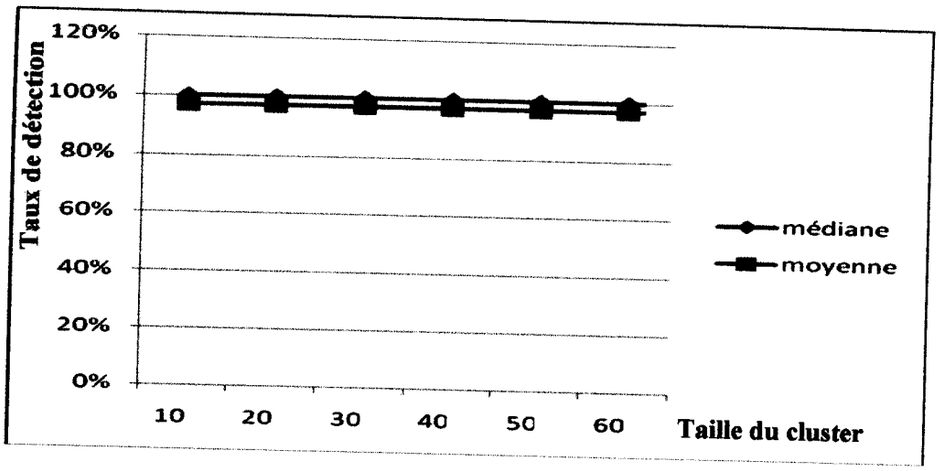


Figure 4.14 taux de détection (cas idéale)

En observant les courbes de simulation dans la figure 4.14, on remarque que le taux de détection reste stable avec l'augmentation de la densité des nœuds. En effet les deux fonctions statistiques donnent de bons résultats avec un taux de détection maximum (100%). Cependant, en réalité, il est quasiment impossible de tomber sur ce cas idéal à cause de la vulnérabilité des capteurs, et de l'ostilité de l'environnement dans lequel ils sont déployés.

7.2.2. Taux de détection dans le cas de valeurs aberrantes

- **Valeurs aberrantes**

Définit comme étant une valeur qui paraît suspecte parce qu'elle s'écarte d'une façon importante des autres valeurs de la variable étudiée (très grande) ou ne semble pas respecter une norme ou une relation bien définie.

Dans le cas de notre application, les valeurs aberrantes représentent les valeurs envoyé par des capteurs en panne ou attaqué (exposés à une source de chaleur de manière volontaire).

En considérant plusieurs taux de valeurs aberrantes, nous avons exécuté les scripts de simulation pour un réseau typique de 50 nœuds pour les deux fonctions statistiques moyenne et médiane. Les résultats de post-simulation sont récapitulés dans le tableau 4.2 ci-dessous :

Taux de valeurs aberrantes	10%	20%	30%	40%	50%	60%	70%
médiane							
moyenne	100%	100%	80%	20%	0%	0%	0%

Tableau 4. 2: Résultat taux de détection (cas valeurs aberrantes).

Les résultats du Tableau 4.2 sont représentés dans le graphe suivant de la figure4.15 :

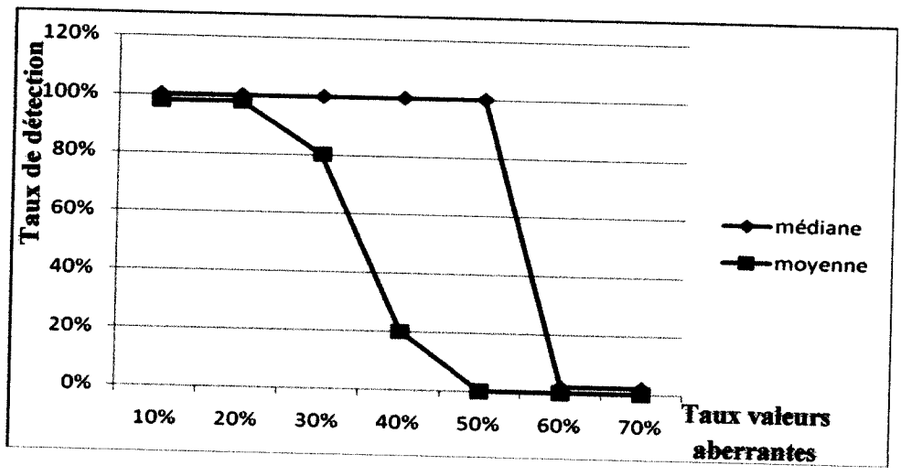


Figure 4.15 taux de détection (cas valeurs aberrantes)

La figure 4.15 illustre les taux de détection des deux fonctions statistiques (moyenne et médiane) et en considérant plusieurs taux de valeurs aberrantes, nous remarquons que la moyenne commence à détecter des fausses alarmes après seulement 30% de valeurs aberrantes, contrairement à la médiane qui reste stable et non-influençable par un taux de valeurs aberrantes qui atteint 50%, mais après ce taux, la médiane commence à détecter des fausses alarmes.

D'après Wagner [WAG04], la médiane est l'une des fonctions statistiques les plus robustes contre les valeurs aberrantes. En effet, elle résiste jusqu'à 50% de valeurs aberrantes. Cette limite représente un résultat satisfaisant par rapport à la moyenne qui elle résiste uniquement à 20% de valeurs aberrantes.

Généralement dans tous les travaux de recherches dans les réseaux de capteurs, des suppositions sont toujours faites sur le modèle du réseau. Si nous supposons qu'au moins la moitié des capteurs dans le cluster fonctionnent correctement, alors le modèle de détection considéré avec la fonction statistique médiane, est largement suffisant pour détecter les feux de forêts avec un taux de 100%.

Conclusion

Dans ce chapitre, nous avons présenté l'implémentation et la simulation de notre application. Pour se faire, nous avons utilisé le système d'exploitation TinyOS, une programmation entière en NesC et une simulation avec TOSSIM.
Par la suite, nous avons tenté de concevoir notre propre application en java pour communiquer avec le réseau de capteurs et interpréter les résultats obtenus par le réseau. De plus, nous avons montré les différentes phases de l'implémentation et du déroulement de notre application.

Enfin, nous avons évalué les performances de l'application sous deux métriques à savoir le taux de détection dans le cas de défaillance ainsi que le taux de détection dans le cas idéale. Cependant, des cas non prévisibles peuvent se présenter dans la réalité. D'où, une expérimentation réelle semble indispensable pour mieux évaluer ces différentes performances.

Conclusion générale

L'événement récent de la technologie des réseaux de capteurs sans-fil, conjugué au progrès de miniaturisation des composants va sans doute dans les années à venir constituer un développement technologique majeur apportant des solutions aux différents problèmes dans plusieurs domaines d'applications liés à l'environnement, la sécurité, la santé, l'agronomie, la domotique, etc.

Le feu représente le premier péril naturel pour les forêts et les zones boisées en Algérie, Il détruit plus d'arbres que toutes les autres calamités naturelles. Dans l'ensemble du bassin méditerranéen, les feux de forêt atteignent aujourd'hui le chiffre d'environ 50 000 par an et les superficies brûlées totales peuvent être estimées à environ 600 000 ha par an.

Dans ce travail, nous avons présenté d'abord un état de l'art sur les réseaux de capteurs et leurs domaines d'applications. Nous avons aussi réalisé une étude sur les feux de forêt en Algérie, Dans cette étude, nous avons présenté et analysé l'historique, les conséquences des feux de forêt en Algérie et les méthodes de lutte appliquées contre les incendies en Algérie, Ensuite, nous avons proposé notre contribution pour améliorer ces méthodes.

L'objectif principal de ce mémoire est de faire la conception d'un nouveau modèle de détection de feu de forêt qui se base sur l'utilisation des réseaux de capteurs et à implémenter une partie de l'application et à sa simulation. Ceci a été fait à l'aide de la plateforme TinyOs dédié aux réseaux de capteurs, le langage NesC et le simulateur Tossim.

Toutefois, tout au long de notre projet, nous avons constaté que l'implémentation et la simulation d'un réseau de capteurs sans-fil pose de grands défis auxquels il faut répondre. Parmi ces défis, la métrise du système d'exploitation TinyOs et du langage NesC à cause de la documentation très succincte, et du simulateur Tossim.

D'un point de vu personnel ce projet nous a permis de découvrir la programmation sur des systèmes embarqués. Un domaine tout à fait nouveau pour nous. Bien que la prise en main ne soit pas très évidente, nous jugeons tout de même que nous sommes arrivés à remplir notre cahier de charge avec succès.

Comme perspective, nous comptons améliorer le modèle de détection proposé en ajoutant d'autres paramètres à la température, par exemple l'humidité et la direction du vent, afin d'améliorer le taux de détection et minimiser les fausses alarmes, même en présence d'un taux de valeurs aberrantes dépassant les 40%. Il serait aussi plus intéressant de faire un testbed (expérimentation réelle) afin de valider notre modèle de détection. Il est clair qu'une simulation n'est pas une solution clé-en-main, et ne peut en aucun cas remplacer une expérimentation réelle.

D'autres voies peuvent être exploitées comme par exemple, utiliser des caméras afin de vérifier en cas d'alarme, l'existence du feu ; ceci a pour effet de minimiser les fausses alertes et de n'intervenir qu'en cas de vraies alertes.

REFERENCES

[AC11] Adel Chouha, Traitement et transfert d'images par réseaux de capteur sans fil, mémoire de magistère, Hadj lakhder-Batna,2010-2011

[GC10] Gerard Chalhoub, Les réseaux de capteurs sans fil, Support de cour, Clermont Université-France, 2009-2010.

[LM09] Lehsaini Mohamed, Diffusion et couverture basées sur les clustering dans les réseaux de capteurs : application à la domotique, thèse de doctorat, Abou Bekr Blkaid-Tlemcen, Franche Comté-France, 2009

[KB09] Kamel Beydoun, Conception d'un protocole de routage hierarchique pour les réseaux de capteurs, thèse de doctorat, Franche Comté-France, 2009

[YC08] Yacine Challal, Réseaux de capteurs, Support de Cours, 2008

[MM08] Messai Mohamed Lamine, Sécurité dans les réseaux de capteur sans fil, mémoire de magistère, Abderrahmane Mira-Bejaia, 2007-2008

[DE07] Dhib Eya, Routage avec QoS temps réel dans les réseaux de capteurs, ingenieur en télécommunication, école superieure des communication de Tunis,2006-2007

[YR07] Yasser Romdhane, «évaluation des performances des protocols S-MAC et directed diffusion dans les réseaux de capteurs », rapport de projet de fin d'étude, école supérieure des communications de Tunis, 2007

[KS07] Kazem Sohraby, Daniel Minoli, Taieb Znati, «wireless sensor networks: Technology, protocols and applications», John Wiley & Sons, Inc, 2007

[BK07]Bouabdellah Kechar, « problématique de la consommation de l'énergie dans les réseaux de capteurs sans fil », LIUPA, université d'Oran, Octobre 2007

[AH07] Abdelhakim HAMZI, «Plateforme basée agents pour l'aide à la conception et la simulation des réseaux de capteurs sans fil», Mémoire de fin d'études Pour l'obtention du diplôme de magistère en informatique, Institut National de formation en Informatique (I.N.I) Oued-smar Alger,2007.

[SK06] Sophia Kaplantzis, « security models for Wireless sensor networks », rapport de recherche , Mars 2006

[MI05] Mohammad Ilyas, Imad Mahgoub, « Handbook of Sensor Networks: Compact Wireless and Wired Sensing Systems», chapitre 15, CRC Press LLC, 2005

[BK05] Bhaskar Krishnamachari, «Networking wireless sensors», Cambridge university press, 2005

[MI04] Mohammad Ilyas and Imad Mahgoub. Handbook of sensor networks: compact wireless and wired sensing systems. CRC Press 2004.

[DD03] Dominique Dhoutaut. Etude du standard IEEE 802.11 dans le cadre des réseaux Ad hoc: de la simulation à l'expérimentation. Thèse pour obtenir le grade de Doctorat en Informatique. 2003.

[TL00] Tayeb Lemlouma. Le Routage dans les Réseaux Mobiles Ad Hoc. Mini projet proposé par Dr. Nadjib Badache septembre 2000.

[DGF02] Mameri D. (Avril 2002) Direction générale des forêts (D.G.F) : Bilan décennal des incendies de forêts en Algérie (1992-2001). 11 p.

[MATE03] Ministère de l'Aménagement du Territoire et de l'Environnement (2003). Rapport sur l'état et l'avenir de l'environnement. 465 p.

[AA08] ARFA Azzedine Mohamed Touffik, «Les incendies de forêt en Algérie : Stratégies de prévention et plans de gestion», Mémoire en vue de l'obtention du diplôme de Magistère en Ecologie et Environnement, université Mentouri Constantine, 2008

[YK12] Yacine KRAIMIA, «Mise en œuvre d'un réseau de senseurs sans fil pour la détection d'un incendie», Mémoire en vue de l'obtention du grade de maitre en science, Université Laval Québec, 2012.

[CP,FG01] Claude PICARD & Frédérique GIROUD, «Calibrage et validation des modèles de propagation des feux de forêt par un capteur de température et de flux de chaleur», Article ,2001.

[YE10] Yunus Emre Aslan August, «A framework for the use of wireless sensor network in forest fire detection and monitoring», Mémoire en vue de l'obtention d'un diplome de master en science, bilkent university, 2010

[JL,MG] Jaime Lloret, Miguel Garcia, D. B. et S. Sendra, « A wireless sensor network deployment for rural and forest _re detection and veri_cation », Integrated Management Coastal Research Institute, Polytechnic University of Valencia, Camino Vera s/n, 46022, Valencia, Spain.

[LY,NW] L. Yu, N. W. et X. Meng, « Real-time forest _re detection with wireless sensor networks », in Proceedings of International Conference On Wireless Communi-cations, Networking and Mobile Computing(WIMOB).

[KC,OG] Khaled Chethouna, Olivier SERO-GUILLAUME & Alain DEGIOVANNI «Calibrage et validation modèles de propagation de s feux de forêt par un capteur de température et de feux de chaleur »

[BA11] Benkhdda Amel ,AMRI Leyla, « installation et configuration d'un réseau de capteurs sans fil », mémoire pour l'obtention du diplôme de master en informatique, Université A.B Tlemcen, Juillet 2011.

[SB11] SAHRAOUI Belkheyr, «La Géo-localisation dans les Réseaux de Capteurs sans Fil. », mémoire pour l'obtention du diplôme de master en informatique, Université A.B Tlemcen, Juillet 2011.

[KM10] MEDJHOUM Khaled, «Les Système Embarqué TinyOS », Université de Bretagne Occidentale (UBO) France, 2010.

[RH10]M. Rabab HAFDAOUI, « Le routage dans les reseaux de capteurs sans fils », Projet De Fin d'Etudes, Université de khenchela,2010.

[HS08]H. Alatrista, S. Aliaga, K. Gouaich, J. Mathieu, « Implémentation de protocole sur une plateforme de réseaux de capteurs sans-fils, mémoire master , Université de Montpellier 2, 2008.

[CB08]Cedric BASTIEN ,« developpement d'un outil permettant le suivi d'un objet mobile », projet de fin d'etude,2008.

[WZ06]Wassim ZNAIDI , « Modélisation formelle de réseaux de capteurs à partir de TinyOS »,projet fin d'etude, école polytechnique de tunisie,2006.

[MB06] Mathieu BADET, Willy BONNEAU, « Réseaux de capteurs : 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,2006

[BC04] Borrong Chen, Geoff Werner Allen, Mark Hempstead, Matt Welsh, Victor Shnayder, « Simulating the Power Consumption of LargeScale Sensor Network Applications», Département de science informatique, Université de Harvard, 2004.

[NesC00] A programming language for deeply networked systems : http://nescc.sourceforge.net/

[CS07] Cruise Summerschool, «Sensor Network Application Development: ZIGBEE CONCEPTS 3», Johannes Kepler University, Linz / Austria, November 5 -7, 2007

[FA08] Mr. fares Abdelfatah, « Développement d'une bibliothèque de capteur sans fil », diplôme de master en informatique, université Montpellier 2, avril 2008

[Car03] Gianni A. Di Caro. Analysis of simulation environments for mobile ad hoc networks. Technical Report No. IDSIA-24-03. Dalle Molle Institute for Arti_cial Intelligence, Galleria 2, 6928 Manno, Switzerland. 2003.

[Con 06] Emmanuel Conchon. Définition et mise en oeuvre d'une solution d'émulation de réseaux sans fil. Institut National Polytechnique De Toulouse. Octobre 2006

[MWI06] Mathieu Badnet, Nicolas Belloir «Réseaux de capteurs : Mise en place d'une plateforme de test et d'expérimentation », Master Technologie de l'Internet 1ère année, France, 2005/2006.

[SYL07] Sylvie Tixier, « TinyOS », Mini rapport, LIF12, Université Lyon 1, 6 Décembre 2007.

[CEM05] C. Han, E. Kohler, M. Srivastava, R. Kumar, R. Shea, « A Dynamic Operating System for Sensor Nodes», Proceedings of the 3rd International Conference on Mobile Systems, Applications and Services (Mobisys), Page(s): 163-176, University of California, Los Angeles, June 2005.

[ABT04] Adam Dunkels, Björn Grönvall, Thiemo Voigt, « Contiki - a Lightweight and Flexible Operating System for Tiny Networked Sensors », 29th Annual IEEE International Conference on Local Computer Networks, Pages: 455–462, Swedish Institute of Computer Science, 2004.

[CCJ06] Cormac Duffy, Cormac J. Sreenan, John Herbert, Utz Roedig, « A Performance Analysis of MANTIS and TinyOS », Technical Report CS-2006-27-11, University College Cork, Ireland, November_2006.

[WAG,04] David Wagner, "Resilient aggregation in sensor networks", In Proceedings of the 2nd ACM workshop on Security of ad hoc and sensor networks, Washington DC, USA, 2004.

ANNEXES

Annexe A: Code source de l'application

Annexe B: Guide d'installation de TinyOS

Code source de l'application

DeltaM.nc

```
//ce projet est réalisé par
      // ARSAOUI  OMAR  &  SENOUCI BEREKSI DJAZIA
          // master en informatique
               // Université A.B Tlemcen, Juillet 2012
includes MaitreMessage;
includes MembreMessage;
module DALTAM
{
   provides     {
        interface StdControl;
        }
  uses  {
        interface Timer;
        interface Leds;
        interface ADC;
        interface SendMsg as SendMessageMembre;
        interface ReceiveMsg as ReceiveMessageMembre;
        interface SendMsg as SendMessageMaitre;
        interface ReceiveMsg as ReceiveMessageMaitre;
         }
}
implementation    {
     TOS_Msg data;
     int min=25;
     int max=55;
            uint16_t value;
     uint16_t val2,val3,val4,val5;
     uint16_t somme = 0;
```

```
            uint16_t num_zone ;
     uint16_t moy[12];
     uint16_t med[12];
     uint16_t zonn[12];
            uint16_t tableau_med[5];
            uint16_t tableau_id[5];
     int cmp = 0;
     int cmp1 = 0;
     int cmp2=0;
     int cmp3=0;
     uint16_t d;

//.....................................................................
  command result_t StdControl.init()    {
     return call Leds.init();
  }

//.....................................................................
command result_t StdControl.start() {
call Leds.yellowOn();
call Timer.start( TIMER_REPEAT,10000);
       return SUCCESS;          }
//.....................................................................
  command result_t StdControl.stop() {
    return call Timer.stop();           }
//.....................................................................
int  tri_tab(uint16_t val[],int taille)
 {
  int i,j;
  uint16_t x ;
for (i = 1; i < taille; i++)
    {
```

```
    x=val[i];
    for (j = i; j >0 && val[j-1]>x; j--)
          {
        val[j]=val[j-1];
        }
    val[j]=x;
    }
   return val[2];        }
//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
  result_t display()
  {
        call Leds.redOn();
         call Leds.greenOn();
         call Leds.yellowOn();
     return SUCCESS;  }
//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
  task void charger_MembreMessage()
  {
    MembreMessage *message = (MembreMessage *)data.data;
           atomic message->valeur = value;
           atomic message->id = TOS_LOCAL_ADDRESS;       }
//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
  void test_temperature(int  temperature)
  {
       if (temperature>50)
              {
            call Leds.redOn();
             call Leds.yellowOff();
                call Leds.greenOff();
              }
            else
```

```
                {
                call Leds.greenOn();
                 call Leds.yellowOff();
                    call Leds.redOff();
                  }          }
//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 int Random (int _iMin, int _iMax)
{
return (_iMin + (rand () % (_iMax-_iMin+1)));
}
//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 int destination  ()
{
  int des ;
             if ((TOS_LOCAL_ADDRESS ==41)||(TOS_LOCAL_ADDRESS
==42)|| (TOS_LOCAL_ADDRESS ==43)||(TOS_LOCAL_ADDRESS ==44))
             des=61;
             else  if ((TOS_LOCAL_ADDRESS ==46)||(TOS_LOCAL_ADDRESS
==47)||       (TOS_LOCAL_ADDRESS ==48)||(TOS_LOCAL_ADDRESS ==49))
             des=61;
             else if ((TOS_LOCAL_ADDRESS ==51)||(TOS_LOCAL_ADDRESS
==52)|| (TOS_LOCAL_ADDRESS ==53)||(TOS_LOCAL_ADDRESS ==54))
             des=62;
             else   if ((TOS_LOCAL_ADDRESS
==56)||(TOS_LOCAL_ADDRESS ==57)|| (TOS_LOCAL_ADDRESS
==58)||(TOS_LOCAL_ADDRESS ==59))
             des=62;
     else if ((TOS_LOCAL_ADDRESS ==61)||(TOS_LOCAL_ADDRESS ==62))
     des=0;
     else
     des=TOS_LOCAL_ADDRESS+20;
return des;
}
```

```
//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 event result_t Timer.fired()
  {
     if ((TOS_LOCAL_ADDRESS !=0)&&(TOS_LOCAL_ADDRESS !=61)&&
(TOS_LOCAL_ADDRESS !=62))
      {
         MembreMessage *message = (MembreMessage *)data.data;

         atomic{
            uint16_t id_maitre ;
            value = Random (min,max);
            if (value>45)
            value=value+40;
                    id_maitre= (((((TOS_LOCAL_ADDRESS-
1)/5)*5)+1)+(cmp2%4));
            if (TOS_LOCAL_ADDRESS ==1)
         num_zone =((TOS_LOCAL_ADDRESS-1)/5)+1;
         dbg(DBG_USR3, ">>>id maitre %d    \n", id_maitre);
          if (TOS_LOCAL_ADDRESS != id_maitre)
                {
                        test_temperature(value);

                 atomic message->valeur = value;
                 atomic message->id = TOS_LOCAL_ADDRESS;

               dbg(DBG_USR3, ">>>le capteur %d
temperature[T=%d°]  \n", TOS_LOCAL_ADDRESS,value);
        call SendMessageMembre.send(id_maitre,
sizeof(MembreMessage), &data);                                    }
                     cmp2++;
                    }           }
    return SUCCESS;
   }  //++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
async event result_t ADC.dataReady(uint16_t valx)

{

 return SUCCESS;

 }

//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
event result_t SendMessageMembre.sendDone(TOS_MsgPtr msg, result_t
success)

  {

   // dbg(DBG_USR1, ">>> envoi effectué pour le capteur %d\n",
TOS_LOCAL_ADDRESS);

    return SUCCESS;

  }

//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

  event result_t SendMessageMaitre.sendDone(TOS_MsgPtr msg,
result_t success)

  {

   // dbg(DBG_USR1, ">>> envoi effectué pour le capteur %d\n",
TOS_LOCAL_ADDRESS);

    return SUCCESS;

  }

//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

event TOS_MsgPtr ReceiveMessageMembre.receive(TOS_MsgPtr m) {

 MembreMessage *message_membre = (MembreMessage*)m->data;

 int dest,val_med,i;

        somme =somme +message_membre->valeur;

        tableau_med[cmp]=message_membre->valeur;

        tableau_id[cmp]=message_membre->id;

        cmp=cmp+1;

            if (cmp ==4)     {

                        MaitreMessage *message = (MaitreMessage
*)data.data;

                        atomic val5 = Random (min,max);

                        if (val5>45)
```



```
                    val5=val5+40;

                      tableau_med[4]=val5;

                      tableau_id[4]=TOS_LOCAL_ADDRESS;


                     test_temperature(val5);
```

```
{
 int dest2;
 MaitreMessage *message1 = (MaitreMessage*)m->data;
 uint16_t moye,medi,nzon;
           moye=message1->moyenne;
           medi=message1->mediane;
           nzon=message1->zone;
 if ((TOS_LOCAL_ADDRESS !=0))
       {
          MaitreMessage *message = (MaitreMessage *)data.data;
          atomic message->mediane = medi;
          atomic message->moyenne = moye;
                          atomic message->zone = nzon;
          dest2=destination();
          call SendMessageMaitre.send(dest2,
sizeof(MaitreMessage), &data);
          }
 else
 if ((TOS_LOCAL_ADDRESS ==0))
       {
          int k;
          moy[cmp1]=moye;
          med[cmp1]=medi;
          zonn[cmp1]=nzon;
           cmp1=cmp1+1;
           dbg(DBG_USR1, "****cmp1= %d\n",cmp1);
          if (cmp1==12)
                 {
                  display();
                  for (k=0;k<12;k++)
                      {
```

```
dbg(DBG_USR1, "*************** ZONE NUMERO %d
**********************  \n",zonn[k]);
if((moy[k])>=50)

          {

dbg(DBG_USR1, " ***TEMPERATURE MOYENNE:%d° -> DEBUT DE FEUX
\n",moy[k]);

                                  }

      else

      {

dbg(DBG_USR1, " ***TEMPERATURE MOYENNE:%d° -> ETAT NORMALE
\n",moy[k]);

                                  }

                     if((med[k])>=50)

                                   {


dbg(DBG_USR1, " ***TEMPERATURE MEDIANE:%d° -> DEBUT DE FEUX
\n",med[k]);

                                  }

                       else      {

dbg(DBG_USR1, " ***TEMPERATURE MEDIANE:%d° -> ETAT NORMALE
\n",med[k]);

}

dbg(DBG_USR1, ".  \n");

      cmp1=0;

                       }           }         }

  return m;

  }

}
```

Guide d'installation de TinyOS

Deux principales versions de TinyOS sont disponibles : la version stable (1.1.0) et la version en développement (2.0.2) qui nécessite l'installation de l'ancienne version pour fonctionner. TinyOS peut être installé sur Windows (2000 et XP), GNU/Linux, Mac OS ou sur un capteur. Nous avons procédé à l'installation de la première version de TinyOS sur Windows XP.

Procédure d'installation sous Windows XP

Ce guide propose l'installation du principal outil nécessaire au bon fonctionnement du système, notamment Cygwin (couche d'émulation de l'API Linux) qui permet d'avoir une interface Unix sous Windows. Cygwin est un environnement d'émulation Linux qui permet d'avoir un shell et de compiler et exécuter les programmes Linux (On dispose ainsi de gcc, apache, bash, etc.).

Figure . B-1 : Cygwin

1- Télécharger le fichier tinyos-1.1.0-1is.exe de la source http://www.tinyos.net/dist-1.1.0/tinyos/windows/ .

2- Exécuter ce fichier pour installer la version 1.1.0 sous windows XP. L'installation se fait automatiquement. Un raccourci de Cygwin est sauvegardé sur le bureau.

3- Accéder à **C:\tinyos\cygwin\opt\tinyos-1.x\doc\tutorial\verifyhw.html** et suivre les étapes que contient cette page afin de vérifier si l'installation est bien réussie.

Procédure de désinstallation

Cygwin ne possède pas de désinstalleur intégré, mais, ce logiciel étant propre, il n'éparpille pas ses fichiers sur le disque. Il est facile à désinstaller à la main. Si des services (tels que Apache ou sshd) ont été installées, il et très important de les arrêter et les désinstaller avant de désinstaller Cygwin.

1- Pour arrêter un service taper : **cygrunsrv -E nomDuService** Ou bien passer par le panneau de configuration. Puis supprimer le service : **cygrunsrv -R nomDuService**

2- Supprimer le répertoire **c:\cygwin** et tout ce qu'il contient.

3- Supprimer le sous-répertoire qui se trouve juste en dessous de setup.exe: il contient tout ce que l'installeur Cygwin a téléchargé. Ce répertoire porte un nom long qui correspond au miroir qui a été utilisé pour télécharger Cygwin. Par exemple : **http%3a%2f%2fcygwin.cict.fr**

4- Prendre Regedit et supprimer les 2 entrées suivantes en base de registre :

HKEY_LOCAL_MACHINE\Software\Cygnus Solutions

Et

HKEY_CURRENT_USER\Software\Cygnus Solutions

5- Retirer les raccourcis que Cygwin a créés sur le bureau et dans le menu Démarrer.

6- Eventuellement, retirer le chemin **c:\cygwin** ou **c:\cygwin\bin** qui a été ajouté à la variable d'environnement PATH.

(Clic-droit sur le **poste de travail > Propriétés > onglet > « Avancé » > Variables d'environnement**)

7- Si des services (tels que ssh, NFS ...) ont été installés, les scripts d'installation Cygwin ont probablement créé des utilisateurs spéciaux dans Windows pour faire tourner ces services (par exemple, l'utilisateur « sshd_server »pour le serveur ssh). Il faut également supprimer ces utilisateurs en passant par le panneau de configuration (ou bien en tapant **control userpasswords2**).

Installation de TinyViz

Les concepteurs développent au fur et à mesure l'outil TinyViz sans mettre à jour les fichiers sources déjà existants dans les anciennes versions. Cela ne permet pas de lancer TinyViz dans des conditions normales. Pour pouvoir le lancer, il est nécessaire de passe par les étapes suivantes:

1- Installer TinyOS-1.0

2- Accéder à: **cd /opt/tinyos-1.x/tools/java**

ET taper : **make**

3- Installer les mises à jour de NesC1.1.1 and TinyOS1.1.15.

Pour se faire, rechercher sur le net http://www.tinyos.net/dist-1.1.0/tinyos/windows/ ces mises à jour en téléchargeant le **rpm** et le mettant dans **C:\tinyos\cygwin\home\PLANETE PC**

Et taper dans le **shell**:

rpm -ivh --ignoreos nesc-1.1.2b-1.cygwin.i386.rpm

rpm -ivh --ignoreos --force tinyos-1.1.15Dec2005cvs-1.cygwin.noarch.rpm

4- Aller à **opt/tinyos-1.x/tools/java/net/tinyos/sim** et verifier si ces fichiers sont presents: **SimObjectGenerator.java** et **MoteSimObjectGenerator.java**

S'ils existent, alors les supprimer de ce répertoire.

5- Editer le **makefile** qui est dans **C:\tinyos\cygwin\opt\tinyos 1.x\tools\java\net\tinyos\sim**

et écrire cette instruction : **net/tinyos/message/avrmote/*.class**

6- Aller à **shell** et taper:

cd /opt/tinyos-1.x/tools/java/net/tinyos/sim

make clean

make

7- Accéder à l'application qui va être simulée. On prend par exemple, l'application Blink.

Accéder au **shell** et faire:

cd opt/tinyos-1.x/apps/blink

make pc

 # tinyviz

export PATH="$TOSROOT/tools/java/net/tinyos/sim:$PATH"

TinyViz -run build/pc/main.exe 20

///Insérer le nombre de noeuds. Par exemple 20

RESUME

En Algérie, les incendies de forêts constituent une menace permanente à l'égard de l'écosystème forestier, en particulier, au cours des saisons estivales. Grâce aux récents progrès des technologies sans fil, une nouvelle branche s'est créée pour offrir des solutions économiquement intéressantes pour la surveillance à distance et le traitement des données dans les environnements complexes et distribués : les réseaux de capteurs sans fil. Dans ce cadre, notre projet de fin d'étude vise à mettre en place un réseau de capteurs capable de recueillir et de traiter des informations environnementales (température, humidité, etc.) provenant d'une zone cible couverte. Ce réseau, est déployé afin de détecter d'éventuels feux à temps et permettre ainsi une intervention rapide pour sauvegarder l'écosystème forestier. Toutefois, pour prédire ou signaler la présence des feux de forêts de manière efficace, il faut s'assurer de mesurer les bons indices, i.e., établir des modèles statistiques de prédiction fiables, tout en minimisant les fausses alertes. Nos résultats de simulations ont démontré des résultats encourageants avec un bon taux de détection.

Mots clé : RCSF, TinyOS, NesC, TOSSIM.

ABSTRACT

In Algeria, forest fires are a constant threat to the ecosystem within them, especially during the summer season. Thanks to the recent progress in wireless technologies, a new branch of network was created to offer economically interesting solutions. These in order to be used for the remote monitoring and the data processing in complex and distributed environments: wireless sensor networks. In this contest, our project aims to set up a wireless sensor network capable of collecting and processing several environmental data like temperature, moisture…etc. coming from a target zone. Furthermore, this network is deployed to detect eventual fires in time in order to allow a fast intervention to safeguard the forest ecosystem. However, to predict and announce forest fires efficiently, it is necessary to measure different environmental data properly i.e. establishing reliable statistical models and in meantime minimizing false fire alarms. Our simulation results demonstrated encouraging results with a good detection rate.

Keywords: WSN, TinyOS , NesC , TOSSIM.

ملخص

في الجزائر، تشكل حرائق الغابات تهديدا مستمرا فيما يتعلق بالنظام الإيكولوجي للغابات، وخصوصا خلال فصل الصيف. مع التطور الكبير في مجال التقنيات اللاسلكية، تم إنشاء فرع جديد لتقديم حلول فعالة و اقتصادية لمراقبة ومعالجة البيانات عن بعد في بيئات مختلفة,و ذلك باستعمال شبكات الاستشعار اللاسلكية . في هذا السياق، يهدف هذا المشروع إلى إنشاء شبكة من أجهزة الاستشعار لجمع ومعالجة المعلومات البيئية (درجة الحرارة والرطوبة وما إلى ذلك). من منطقة معينة. يكون الهدف من هذه الشبكة الكشف عن حرائق محتملة في وقت مبكر ، والتدخل السريع لحماية النظام الإيكولوجي للغابات. من أجل تنبؤ عن وجود حرائق الغابات على نحو فعال، يجب مراعاة المقاييس المناخية المناسبة للكشف عن هذه الأخيرة و ذلك بإنشاء نماذج إحصائية للتقليل من الإنذارات الكاذبة .وقد أظهرت نتائج المحاكاة لدينا نتائج مشجعة مع معدل اكتشاف جيد للحرائق.

كلمات مفتاحية : شبكات الاستشعار اللاسلكية ، TinyOS، NesC، TOSSIM