

TP Guidé Réseau de capteur sans fil

Sommaire

Table des figures	iii
Introduction	1
1. Etudes préalables.....	2
2. Installation des outils.....	17
3. Cas pratique.....	42
Annexes	58
Webographies	iv
Bibliographies	vi

Table des figures

Figure 1: Anatomie d'un capteur	2
Figure 2: Architecture d'un RCSF	3
Figure 3: Modèle en couches pour la communication dans les RCSF	5
Figure 4: Fonctionnement de SPIN	8
Figure 5: Structuration de la mémoire sous TinyOS	10
Figure 6: Etapes de compilation d'une application sous TinyOS	13
Figure 7: Exemple d'une application TinyOS	14
Figure 8: Structuration d'un composant tinyos	14
Figure 9: Structuration d'une configuration tinyos	15
Figure 10: Structuration d'une configuration tinyos	15
Figure 11: Fonctionnement d'une interface tinyOS	16
Figure 12 : Renseignement de la clé de signature « TinyProd »	17
Figure 13 : Renseignement du site de dépôt de TinyOS	17
Figure 14 : Mise à jour de la liste des fichiers disponibles dans les dépôts APT	18
Figure 15 : Résolution de l'erreur : Aucune clé publique n'est disponible	18
Figure 16: Mise à jour de la liste des fichiers disponibles dans les dépôts APT	19
Figure 17 : Installation de nesc, tinyos-tools, msp430-46, avr-tinyos avec erreur	19
Figure 18 : Installation de nesc, tinyos-tools, msp430-46, avr-tinyos sans erreur	20
Figure 19: Etape avant fin de l'installation de nesc, tinyos-tools, msp430-46, avr-tinyos	20
Figure 20 : Commentaire d'une ligne dans le fichier « sources.list »	21
Figure 21 : Installation de nesc, tinyos-tools, msp430-46, avr-tinyos	21
Figure 22 : Vérification de l'installation de « tinyos-tools »	22
Figure 23 : Ajout d'une ligne au fichier « sources.list »	22
Figure 24 : Mise à jour de la liste des fichiers disponibles dans les dépôts APT	23
Figure 25 : Résolution de l'erreur relative à la clé publique non disponible	23
Figure 26 : Etape 1 du succès de l'installation de nesc, tinyos-tools, msp430-46, avr-tinyos	24
Figure 27 : Etape 2 du succès de l'installation de nesc, tinyos-tools, msp430-46, avr-tinyos	24
Figure 28 : Succès de l'installation de nesc, tinyos-tools, msp430-46, avr-tinyos	25
Figure 29 : Téléchargement de TinyOS	25
Figure 30 : Décompression de TinyOS	25
Figure 31 : Changement du nom de TinyOS	26
Figure 32 : Création d'un environnement pour TinyOS	26
Figure 33 : Ajout de l'environnement TinyOS au fichier « .bashrc »	27
Figure 34 : Vérification de l'installation de TinyOS	27
Figure 35 : Accord de privilège à l'utilisateur	27
Figure 36 : Echec 1 de compilation de l'application « Blink » pour la plateforme « telosb »	28
Figure 37 : Installation de « make »	28
Figure 38 : Echec 2 de la compilation de l'application « Blink » pour la plateforme « telosb »	28
Figure 39 : Succès de la compilation de l'application « Blink » pour la plateforme « telosb »	29
Figure 40 : Décompression d'eclipse	29
Figure 41 : Ajout d'eclipse dans les applications	30
Figure 42 : Test d'ajout d'eclipse dans les applications	30
Figure 43 : Etape 1 d'installation du plugin cadena	31
Figure 44 : Etape 2 d'installation du plugin cadena	31
Figure 45 : Etape 3 d'installation du plugin cadena	32
Figure 46 : Etape 4 d'installation du plugin cadena	32

Figure 47 : Etape 5 d'installation du plugin cadena	33
Figure 48 : Etape 6 d'installation du plugin cadena	33
Figure 49 : Etape 7 d'installation du plugin cadena	34
Figure 50 : Etape 8 d'installation du plugin cadena	34
Figure 51 : Etape 9 d'installation du plugin cadena	35
Figure 52 : Fin d'installation du plugin « cadena »	35
Figure 53 : Vérification de l'installation du plugin de TinyOS.....	35
Figure 54 : Succès de vérification de l'installation du plugin de TinyOS.....	36
Figure 55 : Ouverture du simulateur « JTosSim »	37
Figure 56 : Ouverture d'un projet déjà existant dans JTosSim	37
Figure 57 : Ouverture du projet Blink	38
Figure 58 : Projet « Blink » dans « JTosSim ».....	38
Figure 59 : Etape 1 de compilation de l'émulateur mspSim	39
Figure 60 : Etape 2 de compilation de l'émulateur mspSim	40
Figure 61 : Création d'un alias « mspSim »	40
Figure 62: Changement du nom de main.exe en main.elf.....	40
Figure 63 : Lancement de la simulation de l'application « Blink ».....	40
Figure 64 : Programme « Blink » dans mspSim.....	41
Figure 65 : Architecture du cas pratique	42
Figure 66 : Lancement d'Eclipse.....	42
Figure 67 : Choix du workspace.....	43
Figure 68 : Création d'un projet.....	43
Figure 69 : Choix du langage du projet.....	44
Figure 70 : Saisie du nom du projet.....	44
Figure 71 : Suppression du fichier « specification »	45
Figure 72 : Confirmation de suppression du fichier « specification ».....	45
Figure 73 : Création d'un nouveau dossier.....	45
Figure 74 : Saisie du nom du dossier.....	46
Figure 75 : Création d'un nouveau fichier	46
Figure 76 : Saisie du nom du fichier	47
Figure 77 : Chois de l'éditeur de texte	47
Figure 78 : Création d'un nouveau fichier	51
Figure 79 : Saisie du nom du fichier	51
Figure 80 : Choix de l'éditeur de texte.....	52
Figure 81 : Création d'un nouveau fichier	54
Figure 82 : Saisie du nom du fichier	54
Figure 83 : Compilation de l'application « Temperature »	55
Figure 84 : Changement du nom du fichier main.exe en temperature-demo.firmware.....	55
Figure 85 : Déplacement de temperature-demo.firmware dans /opt/mspsim/firmware/esb/.....	55
Figure 86 : Ouverture du fichier « Makefile »	56
Figure 87 : Modification du fichier « Makefile ».....	56
Figure 88 : Lancement de l'émulateur mspSim	56
Figure 89 : Application « Temperature » dans mspSim.....	57
Figure 90: Réseau mode infrastructure et mode ad-hoc	58

Introduction

Depuis leur création, les réseaux de communication sans fil ont connu un succès sans cesse croissant au sein des communautés scientifiques et industrielles. Grâce à ses divers avantages, cette technologie a pu s'instaurer comme acteur incontournable dans les architectures réseaux actuelles. Le média hertzien offre en effet des propriétés uniques, qui peuvent être résumées en trois points : la facilité du déploiement, l'ubiquité de l'information et le coût réduit d'installation. Au cours de son évolution, le paradigme sans fil a vu naître diverses architectures dérivées, telles que : les réseaux cellulaires, les réseaux locaux sans fils et autres. Durant cette dernière décennie, une architecture nouvelle a vu le jour : les réseaux de capteurs sans fil. Ce type de réseaux résulte d'une fusion de deux pôles de l'informatique moderne : les systèmes embarqués et les communications sans fil.

TinyOS est un système d'exploitation intégré, modulaire, destiné aux réseaux de capteurs miniatures. Cette plate-forme logicielle est open source et contient une série d'outils développés par l'Université de Berkeley. Il est enrichi par une multitude d'utilisateurs. En effet, TinyOS est le plus répandu des OS pour les réseaux de capteurs sans-fil. Cet OS est capable d'intégrer très rapidement les innovations en relation avec l'avancement des applications et des réseaux eux même tout en minimisant la taille du code source en raison des problèmes inhérents de mémoire dans les réseaux de capteurs. La librairie TinyOS comprend les protocoles réseaux, les services de distribution, les drivers pour capteurs et les outils d'acquisition de données. TinyOS est en grande partie écrit en C mais on peut très facilement créer des applications personnalisées en langages C, NesC, et Java.

NesC est un langage conçus pour incarner les concepts structurant et le modèle d'exécution de TinyOS. C'est une extension du langage C orientée composant ; il supporte alors la syntaxe du langage C et il est compilé vers le langage C avant sa compilation en binaire.

Dans ce document, il sera question dans un premier temps de réaliser un aperçu des réseaux de capteurs sans fil. Ensuite nous installerons tous les outils nécessaires à la programmation sur TinyOS. Enfin nous essayerons d'implémenter un cas pratique d'application sur TinyOS qui nous permettra de capter la température d'un environnement.

1. Etudes préalables

1.1. Réseau de Capteurs Sans Fil

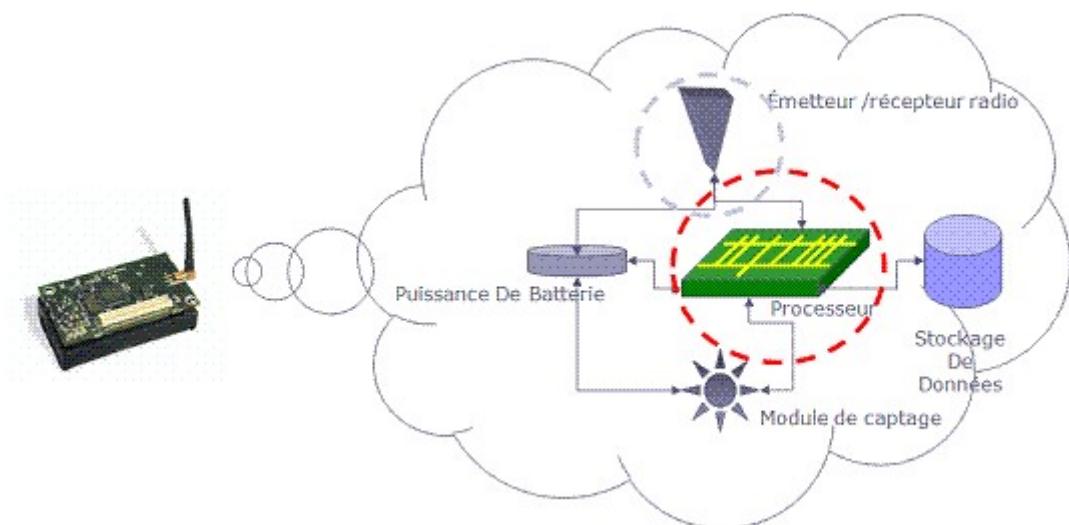
1.1.1. Définition

Un réseau de capteurs sans fil (RCSF) est un réseau ad hoc¹, constitué d'un ensemble de capteurs capables de recueillir et de transmettre des données d'une manière autonome. Ces capteurs sont soit posés à un endroit précis, soit dispersés aléatoirement (souvent déployés par voie aérienne à l'aide d'avions ou hélicoptères).

1.1.2. Constituant d'un capteur

Les progrès conjoints de la micro-électronique, microtechnique, des technologies de transmission sans fil et des applications logicielles ont permis de produire à coût raisonnable des micro-capteurs de quelques millimètres cubes de volume, susceptibles de fonctionner en réseaux. Ils intègrent :

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



Source 1: <https://moodle.utc.fr/file.php/498/SupportWeb/res/Anatomie.png>

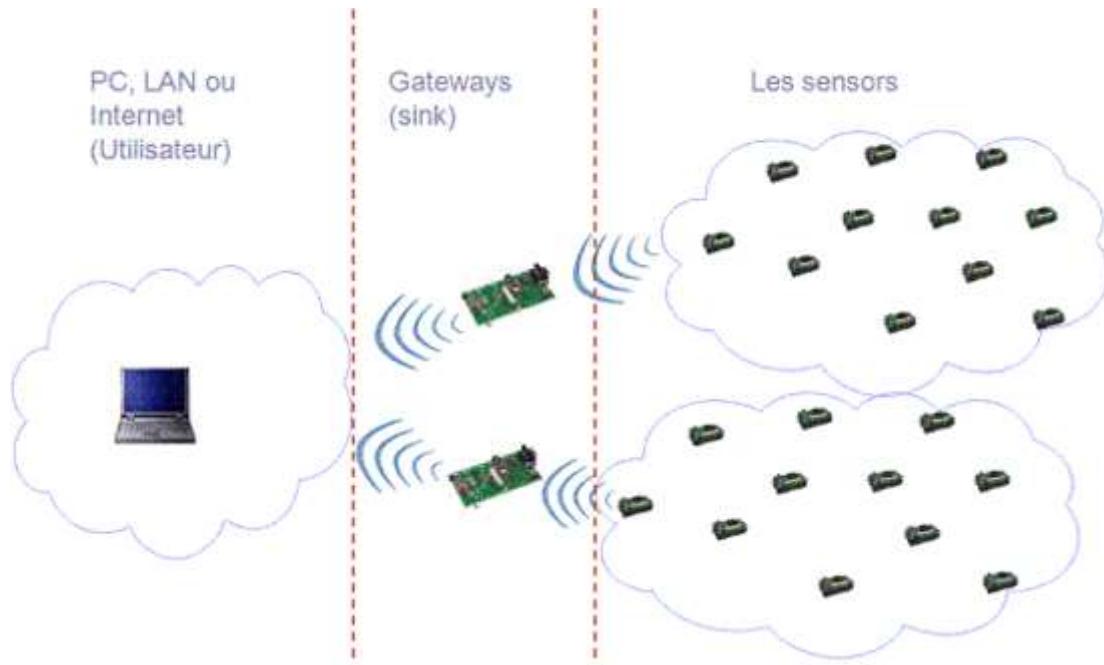
Figure 1: Anatomie d'un capteur

1.1.3. Principe de fonctionnement

Dans un RCSF, les capteurs sont considérés comme des nœuds. Les données captées par les nœuds sont acheminées grâce à un routage multi-saut vers un nœud de base (point de collecte,

¹ Annexe1

appelé aussi nœud-puits ou sink). Ce dernier peut être connecté à l'utilisateur du réseau (via Internet, un satellite ou un autre système). L'usager peut adresser des requêtes aux autres nœuds du réseau, précisant le type de données requises et récolter les données environnementales captées par le biais du nœud puits.



Source 2: <https://moodle.utc.fr/file.php/498/SupportWeb/res/Architecture%20RCSF.png>

Figure 2: Architecture d'un RCSF

1.1.4. Architecture d'un RCSF

1.1.4.1. La topologie en étoile

Dans cette topologie une station de base envoie ou reçoit un message via un certain nombre de nœuds. Ces nœuds peuvent seulement envoyer ou recevoir un message de l'unique station de base, il ne leur est pas permis de s'échanger des messages.

- Avantage : simplicité et faible consommation d'énergie des nœuds, moindre latence de communication entre les nœuds et la station de base.
- Inconvénient : la station de base est vulnérable, car tout le réseau est géré par un seul nœud.

1.1.4.2. La topologie point par point

Dans ce cas (dit « communication multi-sauts »), tout nœud peut échanger avec n'importe quel autre nœud du réseau (s'il est à portée de transmission). Un nœud voulant transmettre un

message à un autre nœud hors de sa portée de transmission, peut utiliser un nœud intermédiaire pour envoyer son message au nœud destinataire.

- Avantage : Possibilité de passer à l'échelle du réseau, avec redondance et tolérance aux fautes.
- Inconvénient : Une consommation d'énergie plus importante est induite par la communication multi-sauts. Une latence est créée par le passage des messages des nœuds par plusieurs autres avant d'arriver à la station de base.

1.1.4.3. La topologie maillé

Les réseaux maillés sont employé dans les réseaux de capteurs plus denses. Ils consistent à interconnecter tous les nœuds ensembles selon le degré d'interconnexion voulu. Ils sont très utilisés dans la domotique ou l'industrie pour la communication entre équipements (M2M) car ils sont très peu sujets aux pannes et extensibles. Ils sont plus complexes à mettre ne place, notamment dû au nombre de liaisons nécessaire : pour N terminaux il faut $N*(N-1)/2$ liaisons.

1.1.4.4. L'architecture de groupe

L'architecture de groupe consiste à hiérarchiser les équipements du réseau : à chaque groupe de nœuds (cluster) on associe un nœud chef (cluster-head). A l'intérieur d'un groupe, les nœuds membres n'ont que la capacité de communiquer avec un nœud chef. Seul le noeud chef peut communiquer avec la station de base (dans un routage d'étoile à étoile) ou avec d'autres nœuds chefs (routage inter-cluster). Dans cette configuration, la panne d'un objet de niveau supérieur entraîne celle des objets situés en dessous.

1.1.4.5. La topologie hybride

Une topologie hybride entre celle en étoile et en grille fournit des communications réseau robustes et diverses, en assurant la minimisation de la consommation d'énergie dans les réseaux de capteurs. Dans ce type de topologie, les nœuds capteurs autonomes en énergie ne routent pas les messages, mais il y a d'autres nœuds qui ont la possibilité de faire le routage des messages. En général, ces nœuds disposent d'une source d'énergie externe.

1.1.5. Médias de transmission

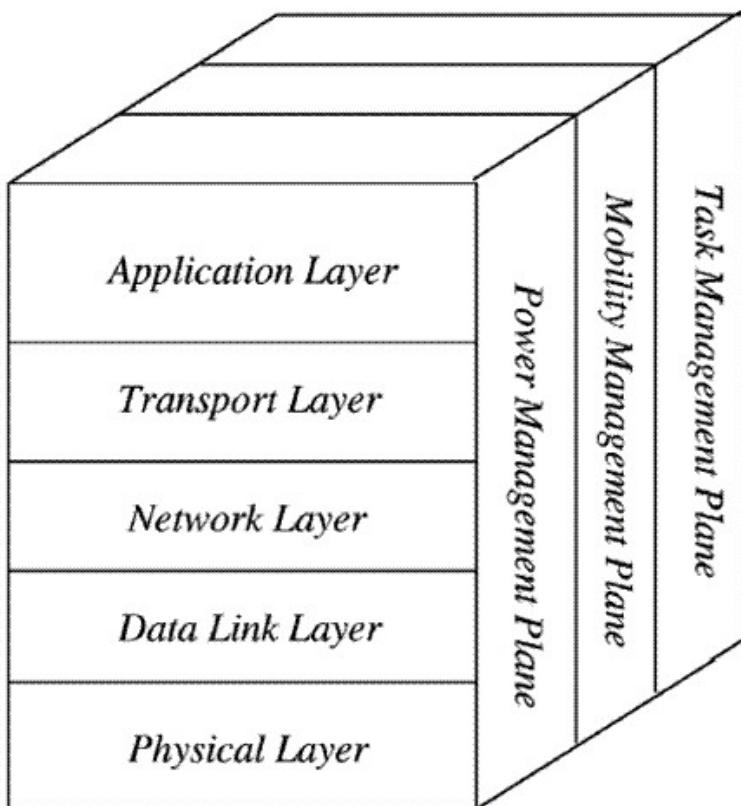
Dans un RCSF, les capteurs sont reliés par une architecture sans-fil. Pour permettre des opérations sur ces réseaux dans le monde entier, le média de transmission doit être normé. On utilise le plus souvent l'infrarouge (qui est license-free, robuste aux interférences, et peu

onéreux), le bluetooth et les communications radio ZigBee. ZigBee est un protocole de haut niveau permettant la communication d'équipements personnels ou domestiques équipés de petits émetteurs radios à faible consommation ; il est basé sur la norme IEEE 802.15.4 pour les réseaux à dimension personnelle.

1.1.6. Communication dans les RCSF

1.1.6.1. Modèle en couches

Le rôle de ce modèle consiste à standardiser la communication entre les composants du réseau afin que différents constructeurs puissent mettre au point des produits (logiciels ou matériels) compatibles. Ce modèle comprend 5 couches qui ont les mêmes fonctions que celles du modèle OSI ainsi que 3 couches pour la gestion de la puissance d'énergie, la gestion de la mobilité ainsi que la gestion des tâches (interrogation du réseau de capteurs). Le but d'un système en couches est de séparer le problème en différentes parties (les couches) selon leur niveau d'abstraction. Chaque couche du modèle communique avec une couche adjacente (celle du dessus ou celle du dessous). Chaque couche utilise ainsi les services des couches inférieures et en fournit à celle de niveau supérieur.



Source 3 : <https://moodle.utc.fr/file.php/498/SupportWeb/res/couches.png>

Figure 3: Modèle en couches pour la communication dans les RCSF

1.1.6.2. Rôle des 5 couches

- La couche physique : Spécifications des caractéristiques matérielles, des fréquences porteuses, etc...
- La couche liaison : Spécifie comment les données sont expédiées entre deux nœuds/routeurs dans une distance d'un saut. Elle est responsable du multiplexage des données, du contrôle d'erreurs, de l'accès au media,... Elle assure la liaison point à point et multipoint dans un réseau de communication.
- La couche réseau : Dans la couche réseau le but principal est de trouver une route et une transmission fiable des données captées des nœuds capteurs vers le puits "sink" en optimisant l'utilisation de l'énergie des capteurs. Ce routage diffère de celui des réseaux de transmission ad hoc sans fils par les caractéristiques suivantes:
 - ❖ il n'est pas possible d'établir un système d'adressage global pour le grand nombre de capteurs.
 - ❖ les applications des réseaux de capteurs exigent l'écoulement des données mesurées de sources multiples à un puits particulier.
 - ❖ les multiples capteurs peuvent produire de mêmes données à proximité d'un phénomène (redondance).
 - ❖ les capteurs exigent ainsi une gestion soigneuse des ressources.

En raison de ces différences, plusieurs nouveaux algorithmes ont été proposés pour le problème de routage dans les réseaux de capteurs.

- La couche transport : Cette couche est chargée du transport des données, de leur découpage en paquets, du contrôle de flux, de la conservation de l'ordre des paquets et de la gestion des éventuelles erreurs de transmission.
- La couche application : Cette couche assure l'interface avec les applications. Il s'agit donc du niveau le plus proche des utilisateurs, géré directement par les logiciels.

1.1.6.3. Plans de gestion

Les plans de gestion d'énergie, de mobilité et de tâche contrôlent l'énergie, le mouvement et la distribution de tâche au sein d'un capteur. Ces plans aident les capteurs à coordonner la tâche de captage et minimiser la consommation d'énergie. Ils sont donc nécessaires pour que les capteurs puissent collaborer ensemble, acheminer les données dans un réseau mobile et partager les ressources entre eux en utilisant efficacement l'énergie disponible. Ainsi, le réseau peut prolonger sa durée de vie.

- Plan de gestion d'énergie : contrôle l'utilisation de la batterie. Par exemple, après la réception d'un message, le capteur éteint son récepteur afin d'éviter la duplication des messages déjà reçus. En outre, si le niveau d'énergie devient bas, le capteur diffuse à ses voisins une alerte les informant qu'il ne peut pas participer au routage. L'énergie restante est réservée au captage ;
- Plan de gestion de mobilité : détecte et enregistre le mouvement du capteur. Ainsi, un retour arrière vers l'utilisateur est toujours maintenu et le capteur peut garder la trace de ses voisins. En déterminant leurs voisins, les capteurs peuvent balancer l'utilisation de leur énergie et la réalisation de tâche ;
- Plan de gestion de tâche : balance et ordonne les différentes tâches de captage de données dans une région spécifique. Il n'est pas nécessaire que tous les capteurs de cette région effectuent la tâche de captage au même temps ; certains capteurs exécutent cette tâche plus que d'autres selon leur niveau de batterie.

1.1.7. Protocoles de routage dans les RCSF

La propagation et la délivrance des données dans un RCSF représentent la fonctionnalité la plus importante du réseau. Elle doit prendre en considération toutes les caractéristiques des capteurs afin d'assurer les meilleures performances du système : durée de vie, fiabilité, temps de réponse, ... etc. Vu la spécificité des RCSF, un nombre important de recherches sont orientées vers une violation du découpage en couches protocolaires indépendantes, et introduisent la notion de « cross layer optimization ». Par exemple, en utilisant des mécanismes d'agrégation, les routeurs intermédiaires doivent accéder à la donnée afin d'établir des résumés des lectures de la région.

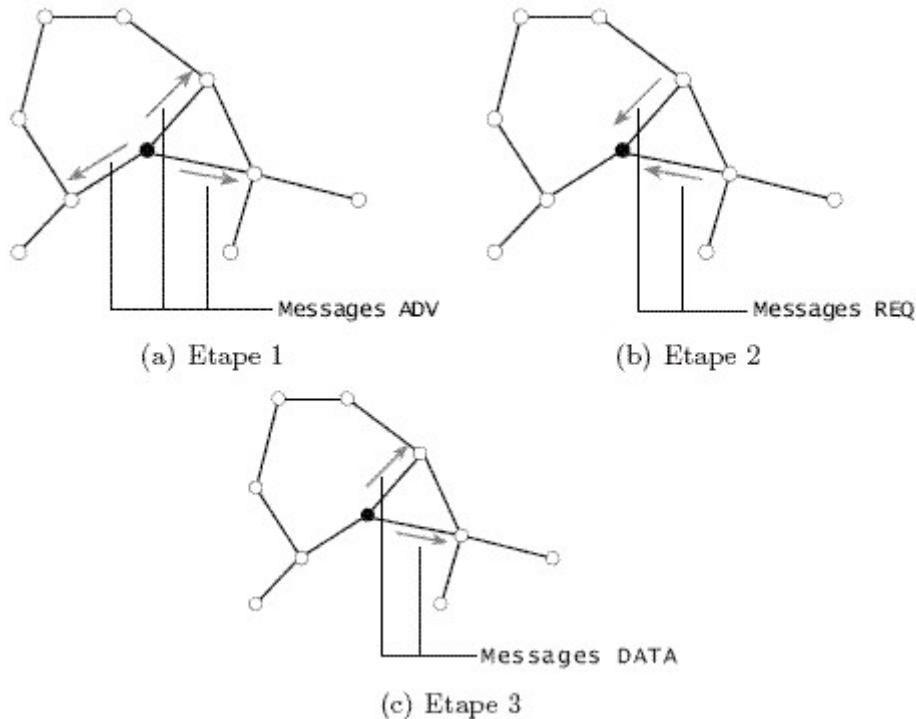
1.1.7.1. Exemple de protocole de routage : SPIN

Les communications dans SPIN se font en trois étapes :

- Lorsqu'un nœud veut émettre une donnée, il émet d'abord un message ADV contenant une description de la donnée en question.
- Un nœud recevant un message ADV, consulte sa base d'intérêt. S'il est intéressé par cette information, il émet un message REQ vers son voisin.
- En recevant un message REQ, l'émetteur transmet à l'intéressé la donnée sous forme d'un message DATA.

Lorsque le nœud s'aperçoit que son énergie est descendue sous un certain seuil, il change son mode de fonctionnement, et ne répond à aucun message ADV.

La figure suivante illustre ces trois étapes :



Source 4: <https://moodle.utc.fr/file.php/498/SupportWeb/res/spin.png>

Figure 4: Fonctionnement de SPIN

1.1.8. Système d'exploitation pour RCSF

Les systèmes d'exploitation pour RCSF sont des systèmes d'exploitation embarqués². Ils sont spécifiquement conçus pour optimiser l'usage des ressources matérielles limitées (peu de mémoire RAM, une vitesse de traitement processeur faible et peu d'énergie électrique) dont ils disposent. De nombreux systèmes d'exploitation spécialisés existent, parmi lesquels on note : Contiki, ERIKA Enterprise, Nano-RK, TinyOS, MantisOS, RETOS, Senses, Cormos, LiteOS, NanoQplus.

Le rôle du système d'exploitation pour RCSF est d'être l'interface entre des ressources matérielles limitées et des applications distribuées³. Le but est de faciliter la programmation des applications, mais aussi d'optimiser les utilisations de ressources.

Malgré les limites matérielles, les capteurs doivent intégrer les fonctionnalités de base d'un système d'exploitation telles que :

- l'allocation de mémoire (RAM, EEPROM);

² Annexe 2

³ Annexe 3

- le système de fichiers;
- le contrôle d'exécution de code;
- le chargement, le lancement et la gestion des applications;
- la transmission de données;
- la gestion de l'énergie.

Ils doivent en plus gérer les spécificités propres aux systèmes d'exploitation pour RCSF:

- la gestion efficace de l'énergie;
- la fiabilité du système d'exploitation;
- une faible empreinte mémoire;
- la reconfiguration du système d'exploitation;
- le support de protocole de routage optimisé (μ IP⁴, Rime, μ IPv6);
- le support Temps-réel.

Les systèmes d'exploitation tels que TinyOS, Contiki, SOS, MantisOS, Nano-RK sont conçus pour répondre à ces contraintes.

1.2. TinyOS

1.2.1. 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.

Le fonctionnement d'un système basé sur TinyOS s'appuie sur la gestion des événements se produisant. Ainsi, l'activation de tâches, leur interruption ou encore la mise en veille du capteur s'effectue à l'apparition d'événements, ceux-ci ayant la plus forte priorité. Ce fonctionnement évènementiel (event-driven) s'oppose au fonctionnement dit temporel (time-driven) où les actions du système sont gérées par une horloge donnée. TinyOS a été conçu pour minimiser la consommation en énergie du capteur. Ainsi, lorsqu'aucune tâche n'est active, il se met automatiquement en veille.

⁴ Annexe 4

1.2.2. Allocation de la mémoire

TinyOS a une empreinte mémoire très faible puisqu'il ne prend que 300 à 400 octets dans le cadre d'une distribution minimale. En plus de cela, il est nécessaire d'avoir 4 Ko de mémoire libre qui se répartissent entre 3 composants :

- 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.

La gestion de la mémoire possède de plus quelques propriétés. Ainsi, il n'y a pas d'allocation dynamique de mémoire et pas de pointeur de fonctions. Bien sûr cela simplifie grandement l'implémentation. Par ailleurs, il n'existe pas de mécanisme de protection de la mémoire sous TinyOS ce qui rend le système particulièrement vulnérable aux crashes et corruptions de la mémoire.

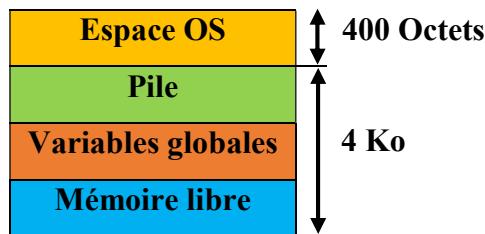


Figure 5: Structuration de la mémoire sous TinyOS

1.2.3. 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, ADC...) et peut être réutilisé dans différentes applications. Ces applications sont des ensembles de composants associés 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).

1.2.3.1. Composants

L'implémentation de composants s'effectue en déclarant des tâches, des commandes ou des événements.

- une tâche est un travail de « longue durée » ;
- une commande est l'exécution d'une fonctionnalité précise dans un autre composant ;
- un événement est l'équivalent logiciel à une interruption matérielle.

1.2.3.1.1. Tâche

Les tâches sont utilisées pour effectuer la plupart des blocs d'instruction d'une application. À l'appel d'une tâche, celle-ci va prendre place dans une file d'attente de type FIFO (First In First Out) pour y être exécutée. Il n'y a pas de mécanisme de préemption entre les tâches, et une tâche activée s'exécute en entier. Ce mode de fonctionnement permet de bannir les opérations pouvant bloquer le système (inter-blocage, famine, ...). Par ailleurs, lorsque la file d'attente des tâches est vide, le système d'exploitation met en veille le dispositif jusqu'au lancement de la prochaine interruption (on retrouve le fonctionnement event-driven).

1.2.3.1.2. Evènements

Les évènements sont prioritaires par rapport aux tâches et peuvent interrompre la tâche en cours d'exécution. Ils 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 tâches.

1.2.3.2. Type de composant

Dans la pratique, NesC permet de déclarer 2 types de composants : les modules et les configurations.

- Les modules constituent les briques élémentaires de code et implémentent une ou plusieurs interfaces. Les interfaces sont des fichiers décrivant les commandes et évènements proposés par le composant qui les implémente. L'utilisation des mots clefs « Use » et « Provide » au début d'un composant permet de savoir respectivement si celui-ci fait appel à une fonction de l'interface ou redéfinit son code.
- Une application peut faire appel à des fichiers de configuration pour regrouper les fonctionnalités des modules. Un fichier « top-level configuration » permet de faire le lien entre tous les composants.

1.2.4. L'ordonnanceur TinyOS

L'ordonnanceur TinyOS est au cœur de la gestion des tâches et des évènements du système. Il comprend:

- 2 niveaux de priorité (bas pour les tâches, haut pour les évènements) ;
- 1 file d'attente FIFO (disposant d'une capacité de 7).

Par ailleurs, entre les tâches, un niveau de priorité est défini permettant de classer les tâches, tout en respectant la priorité des interruptions (ou évènements). Lors de l'arrivée d'une nouvelle tâche, celle-ci sera placée dans la file d'attente en fonction de sa priorité (plus elle est grande, plus le placement est proche de la sortie). Dans le cas où la file d'attente est pleine, la tâche dont la priorité est la plus faible est enlevée de la FIFO.

1.2.5. Equipment supportant TinyOS

Voici une liste rapide des équipements supportés initialement par TinyOS dans sa version « tinyos-1.1.11-3is ».

ATMega8	AVRMote	Mica	Mica2
Micadot	Mica128	Micaz	MSP430
Rene2	Telos	Telos2	PC

Source 5: https://fr.wikipedia.org/wiki/TinyOS#Cibles_posibles_pour_TinyOS

Tableau 1: Liste des équipements supportant TinyOS

Au-delà de cette liste, il est possible d'implémenter tout type de plateforme embarquée physique en redéveloppant les bibliothèques nécessaires à la prise en compte des entrées sorties nécessaires.

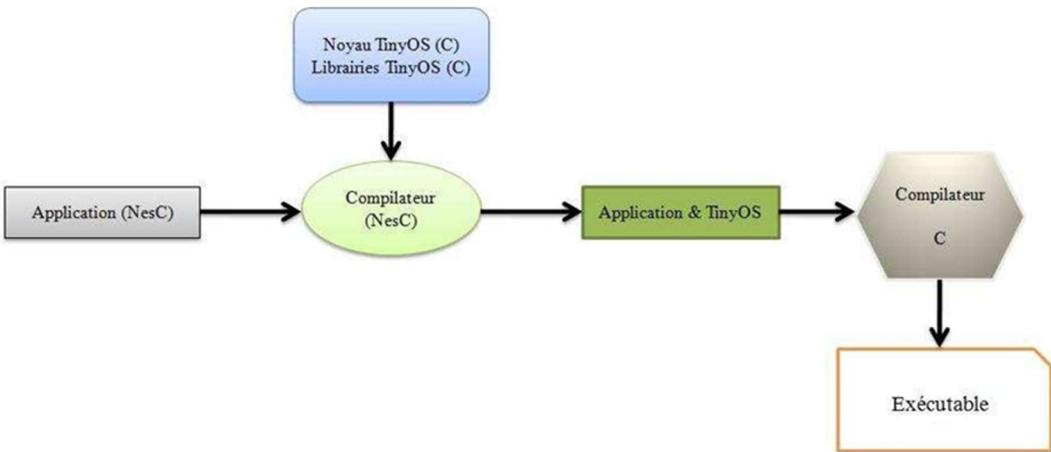
1.3. Langage de programmation nesC

NesC est un langage de programmation dérivé du langage C, conçu pour minimiser l'utilisation de mémoire et de puissance de calcul par les capteurs, qui très souvent disposent de ressources très limitées (batterie de faible puissance et non changeable, mémoire réduite...). Il est ainsi essentiellement dédié pour la réalisation de systèmes embarqués distribués. Il cible en particulier l'implémentation d'applications pour les réseaux de capteurs. Il propose de nombreuses optimisations pour diminuer l'occupation de l'espace mémoire.

1.3.1. Compilation

La première étape de ce processus consiste à compiler les fichiers nécessaires à l'application et au système d'exploitation. Celle-ci est réalisée via le compilateur NesC fourni par TinyOS. Son rôle est premièrement de transformer les fichiers NesC en fichier C et deuxièmement d'y intégrer les fichiers du noyau de TinyOS. Ce qui permet d'obtenir

un fichier source C unique. Une fois cette étape accomplie, il ne reste alors qu'à utiliser un compilateur C traditionnel qui va utiliser le fichier précédemment créé afin de générer une application exécutable. Celle-ci sera donc constituée par la « fusion » du système d'exploitation et du code applicatif. Ces différentes phases peuvent être synthétisées comme l'illustre la figure suivante.



Source 6: Les Système Embarqué TinyOS, MEDJHOUM Khaled, page 13

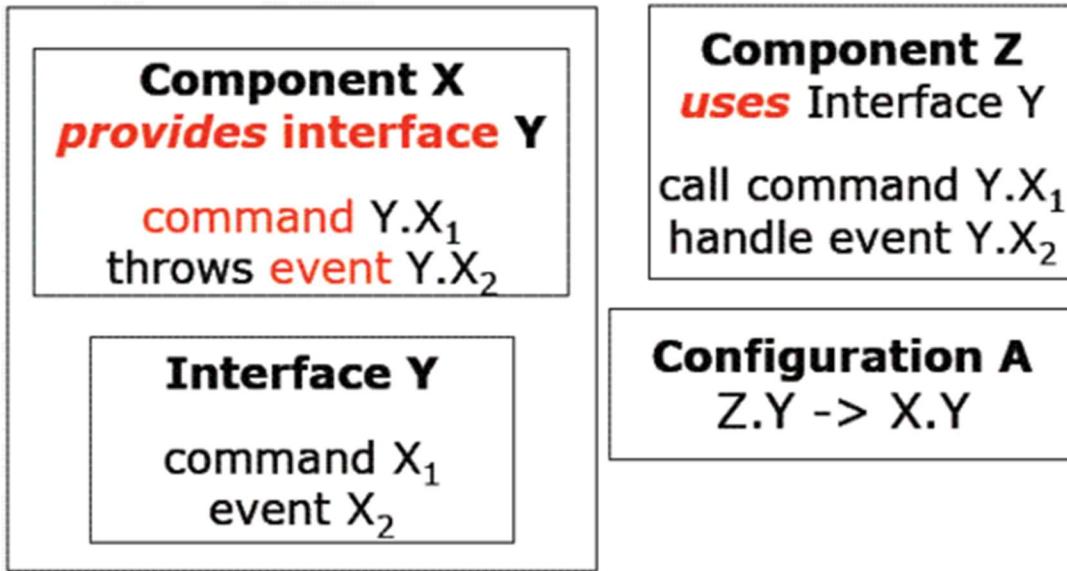
Figure 6: Etapes de compilation d'une application sous TinyOS

1.3.2. Concepts de nesC

Les concepts de base de nesC sont les suivants:

- ✚ Application: un ou plusieurs composants reliés ensemble pour former un exécutable ;
- ✚ Composant: un élément de base pour former une application nesC. Les composants fournissent et utilisent des interfaces. Il existe deux types de composants: modules et configurations ;
 - ❖ Module : composant qui implémente une ou plusieurs interfaces
 - ❖ Configuration : composant qui relie d'autres composants ensemble
- ✚ Interface: définie d'une manière abstraite les interactions entre deux composants ;

La programmation nesC comprend deux types de fichiers : Composants et Interfaces.



Source 7: <https://moodle.utc.fr/file.php/498/SupportWeb/res/tinyos-exemple-app.png>

Figure 7: Exemple d'une application TinyOS

1.3.3. Composant

Un composant est toujours un fichier (avec l'extension du fichier « .nc ») écrit en nesC, qui présente deux sections, contenant sa spécification et son implémentation. Cependant, la section d'implémentation diffère selon qu'il s'agit d'un module ou d'une configuration.

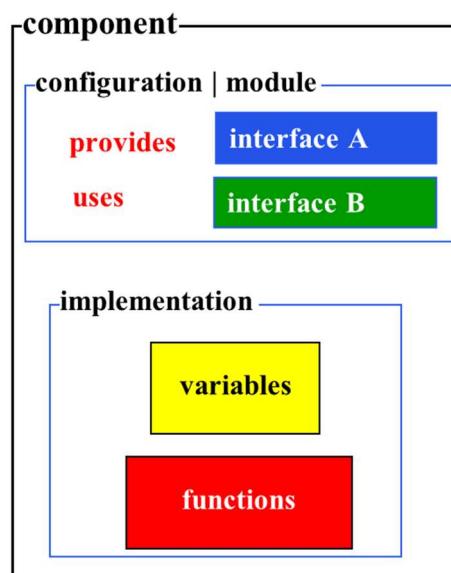


Figure 8: Structuration d'un composant tinyos

Les sections d'implémentation de module consistent en un code qui ressemble à C. Le code de module déclare les variables, les fonctions et appelle les fonctions.

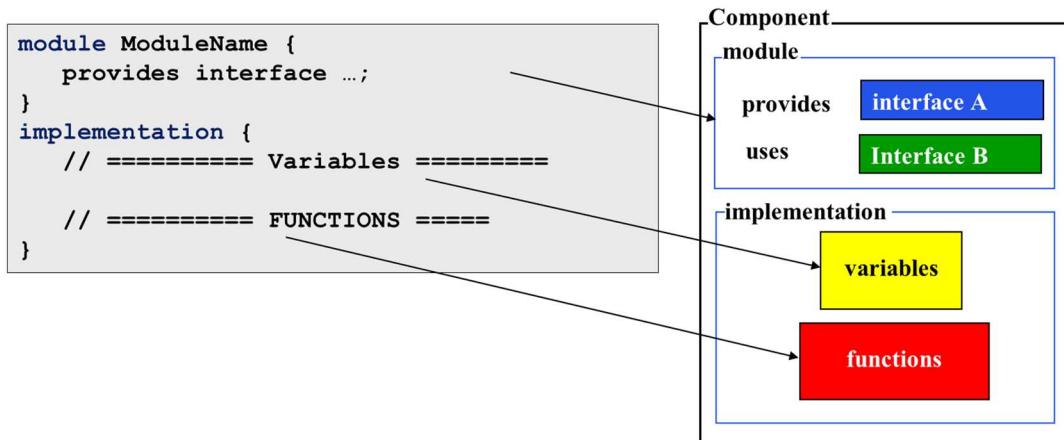


Figure 9: Structuration d'une configuration tinyos

Les sections d'implémentation de la configuration se composent du code qui relie les composants entre eux. Les configurations sont la principale différence entre nesC et C (et les autres dérivés de C).

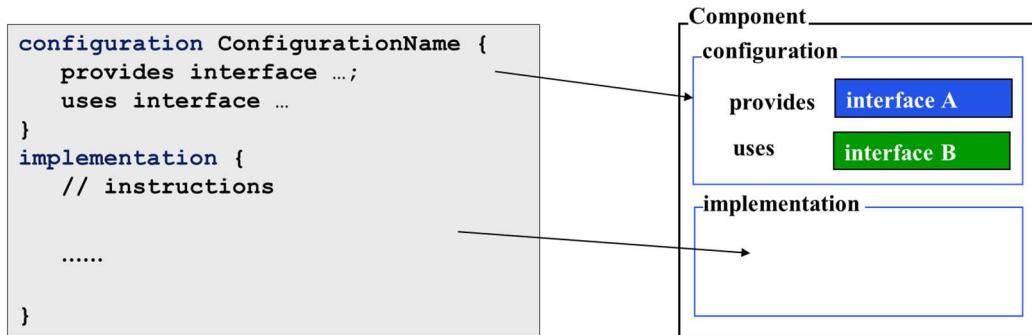


Figure 10: Structuration d'une configuration tinyos

Tous les composants ont donc deux blocs de code. Le premier bloc décrit sa signature et le deuxième bloc décrit sa mise en œuvre.

1.3.4. Interface

Une interface définit les interactions entre deux composants. Les interfaces sont bidirectionnelles. Elles spécifient un ensemble de fonctions à implémenter par les composants.

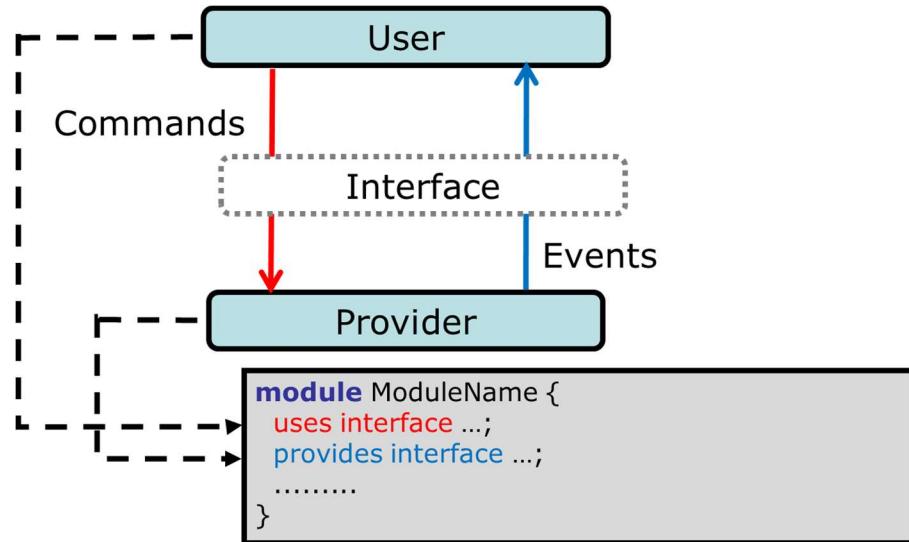
On peut distinguer deux cas d'utilisations :

- ✚ Un composant fourni (provides) une interface à un autre composant.
- ✚ Un composant utilise (uses) l'interface fournie par un autre composant.

Si un composant appelle une commande (command), il utilise l'interface et le composant qui implémente cette commande est le fournisseur (provider) de cette interface.

Si un composant appelle une fonction d'événement (event), il en est le fournisseur et le composant qui l'implémente est l'utilisateur (user) de l'événement.

Les commandes « command » font typiquement des appels du haut vers le bas (des composants applicatifs vers les composants plus proches du matériel), alors que les événements « event » remontent les signaux du bas vers le haut.



Source 8: tinyos-tutorial-wsn, page 11

Figure 11: Fonctionnement d'une interface tinyOS

2. Installation des outils

2.1. Prérequis

Installons tout d'abord le logiciel de virtualisation « VMware ». Ensuite, installons la distribution linux Debian 7 version 32 bits dans la machine virtuelle. Ici, nous avons opté pour la version 7.1.0. Le « code Name » de la distribution est « wheezy ». Pour le connaitre il suffit de tapez « lsb_release -a » dans la console.

2.2. Installation de tinyos

Informons « apt » sur la clé de signature « TinyProd » avec la commande :

```
« wget -O - http://tinyprod.net/repos/debian/tinyprod.key | sudo apt-key add - »
```



The screenshot shows a terminal window titled "Terminal (au nom du superutilisateur)". The window title bar includes the date "sam. 29 déc., 03:09" and the user "ibam2019". The terminal content shows the command being run:

```
Fichier Édition Affichage Rechercher Terminal Aide
root@debian:/home/ibam2019# wget -O - http://tinyprod.net/repos/debian/tinyprod.key | sudo apt-key add -
--2018-12-29 03:09:38-- http://tinyprod.net/repos/debian/tinyprod.key
Résolution de tinyprod.net (tinyprod.net)... 173.11.87.195
Connexion vers tinyprod.net (tinyprod.net)|173.11.87.195|:80...connecté.
requête HTTP transmise, en attente de la réponse...200 OK
Longueur: 1748 (1,7K) [application/pgp-keys]
Sauvegarde en : « STDOUT »

100%[=====] 1 748 --.- K/s   ds 0s
2018-12-29 03:09:41 (95,0 MB/s) - envoi vers sortie standard [1748/1748]

OK
root@debian:/home/ibam2019#
```

Figure 12 : Renseignement de la clé de signature « TinyProd »

Tapons les lignes suivantes :

- « cd /etc/apt/sources.list.d »
- « echo "deb http://tinyprod.net/repos/debian wheezy main" >> tinyprod-debian.list »
- « echo "deb http://tinyprod.net/repos/debian msp430-46 main" >> tinyprod-debian.list »

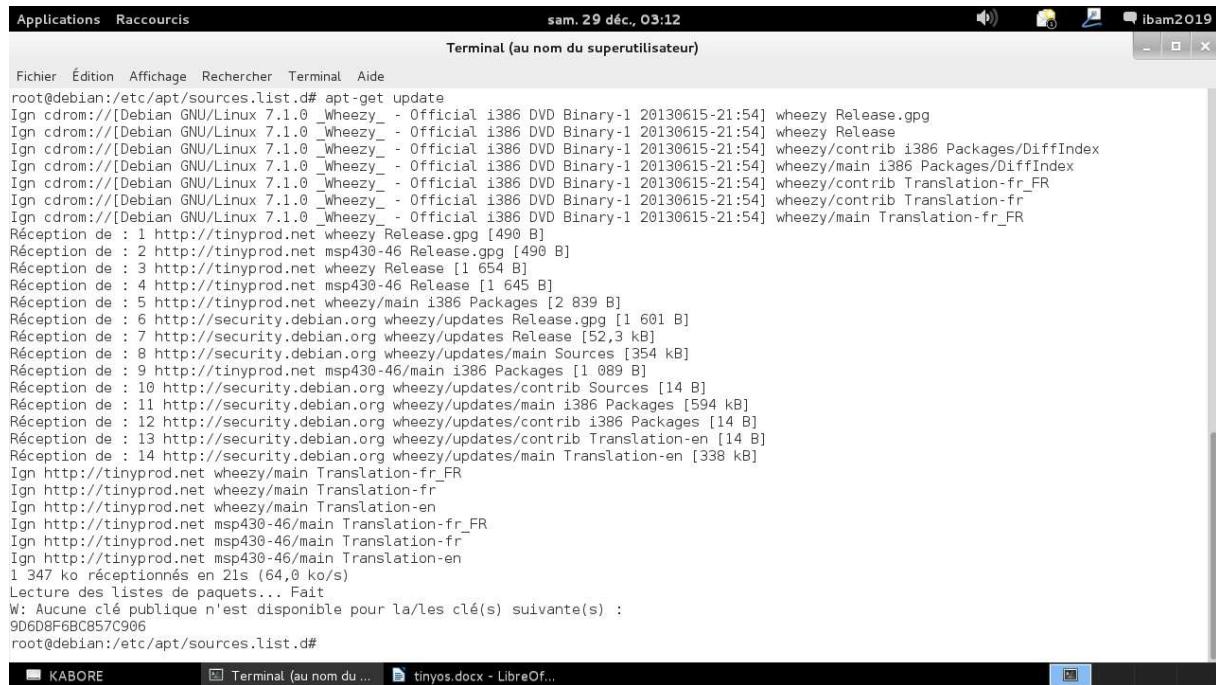


The screenshot shows a terminal window titled "Terminal (au nom du superutilisateur)". The window title bar includes the date "sam. 29 déc., 03:11" and the user "ibam2019". The terminal content shows the commands being run:

```
Fichier Édition Affichage Rechercher Terminal Aide
root@debian:/home/ibam2019# cd /etc/apt/sources.list.d
root@debian:/etc/apt/sources.list.d# echo "deb http://tinyprod.net/repos/debian wheezy main" >> tinyprod-debian.list
root@debian:/etc/apt/sources.list.d# echo "deb http://tinyprod.net/repos/debian msp430-46 main" >> tinyprod-debian.list
root@debian:/etc/apt/sources.list.d#
```

Figure 13 : Renseignement du site de dépôt de TinyOS

Faisons une mise à jour de la liste des packages : « apt-get update »



```
root@debian:/etc/apt/sources.list.d# apt-get update
Ign cdrom://[[Debian GNU/Linux 7.1.0 _Wheezy_ - Official i386 DVD Binary-1 20130615-21:54] wheezy Release.gpg
Ign cdrom://[[Debian GNU/Linux 7.1.0 _Wheezy_ - Official i386 DVD Binary-1 20130615-21:54] wheezy Release
Ign cdrom://[[Debian GNU/Linux 7.1.0 _Wheezy_ - Official i386 DVD Binary-1 20130615-21:54] wheezy/contrib i386 Packages/DiffIndex
Ign cdrom://[[Debian GNU/Linux 7.1.0 _Wheezy_ - Official i386 DVD Binary-1 20130615-21:54] wheezy/main i386 Packages/DiffIndex
Ign cdrom://[[Debian GNU/Linux 7.1.0 _Wheezy_ - Official i386 DVD Binary-1 20130615-21:54] wheezy/contrib Translation-fr_FR
Ign cdrom://[[Debian GNU/Linux 7.1.0 _Wheezy_ - Official i386 DVD Binary-1 20130615-21:54] wheezy/contrib Translation-fr
Ign cdrom://[[Debian GNU/Linux 7.1.0 _Wheezy_ - Official i386 DVD Binary-1 20130615-21:54] wheezy/main Translation-fr_FR
Récception de : 1 http://tinyprod.net wheezy/Release.gpg [490 B]
Récception de : 2 http://tinyprod.net msp430-46 Release.gpg [490 B]
Récception de : 3 http://tinyprod.net wheezy/Release [1 654 B]
Récception de : 4 http://tinyprod.net msp430-46 Release [1 645 B]
Récception de : 5 http://tinyprod.net wheezy/main i386 Packages [2 839 B]
Récception de : 6 http://security.debian.org wheezy/updates Release.gpg [1 601 B]
Récception de : 7 http://security.debian.org wheezy/updates Release [52,3 kB]
Récception de : 8 http://security.debian.org wheezy/updates/main Sources [354 kB]
Récception de : 9 http://tinyprod.net msp430-46/main i386 Packages [1 089 B]
Récception de : 10 http://security.debian.org wheezy/updates/contrib Sources [14 B]
Récception de : 11 http://security.debian.org wheezy/updates/main i386 Packages [594 kB]
Récception de : 12 http://security.debian.org wheezy/updates/contrib i386 Packages [14 B]
Récception de : 13 http://security.debian.org wheezy/updates/contrib Translation-en [14 B]
Récception de : 14 http://security.debian.org wheezy/updates/main Translation-en [338 kB]
Ign http://tinyprod.net wheezy/main Translation-fr_FR
Ign http://tinyprod.net wheezy/main Translation-fr
Ign http://tinyprod.net wheezy/main Translation-en
Ign http://tinyprod.net msp430-46/main Translation-fr_FR
Ign http://tinyprod.net msp430-46/main Translation-fr
Ign http://tinyprod.net msp430-46/main Translation-en
1 347 ko réceptionnés en 21s (64,0 ko/s)
Lecture des listes de paquets... Fait
W: Aucune clé publique n'est disponible pour la/les clé(s) suivante(s) :
9D6D8F6BC857C906
root@debian:/etc/apt/sources.list.d#
```

Figure 14 : Mise à jour de la liste des fichiers disponibles dans les dépôts APT

Si nous rencontrons une erreur du genre :

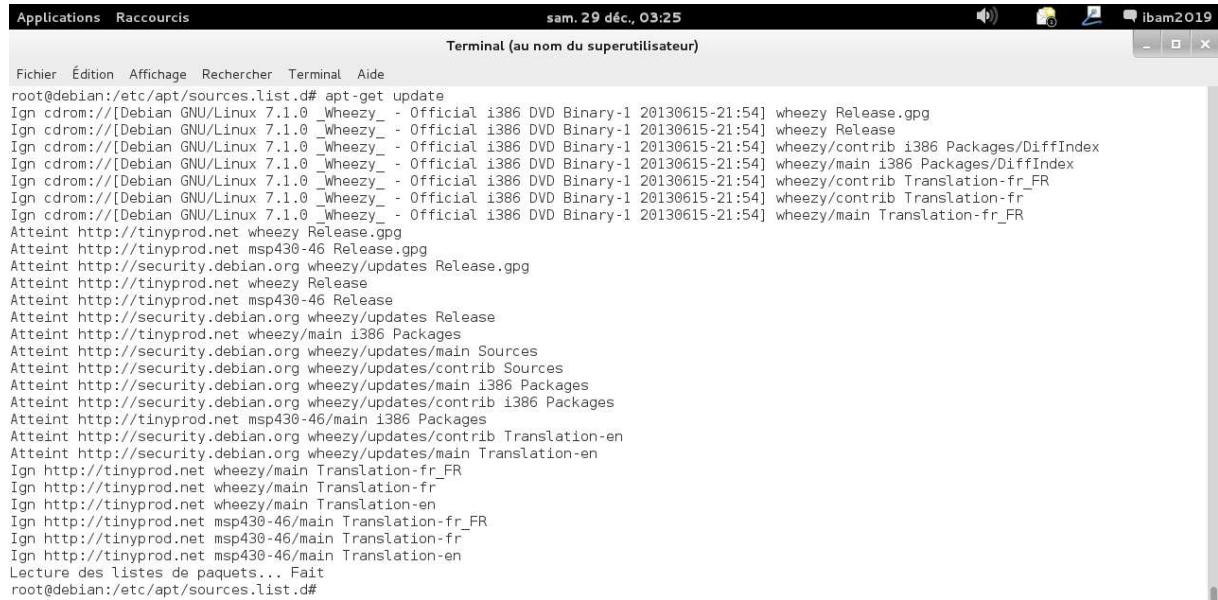
« w : Aucune clé publique n'est disponible pour la/les clé(s) suivante(s) :
9D6D8F6BC857C906 », tapons la commande suivante pour la résoudre:
« apt-key adv --recv-keys --keyserver keyserver.ubuntu.com 9D6D8F6BC857C906 »



```
root@debian:/etc/apt/sources.list.d# apt-key adv --recv-keys --keyserver keyserver.ubuntu.com 9D6D8F6BC857C906
Executing: gpg --ignore-time-conflict --no-options --no-default-keyring --secret-keyring /tmp/tmp.ypmYbI9MhR --trustdb-name /etc/apt/trustdb.gpg --keyring /etc/apt/trusted.gpg --primary-keyring /etc/apt/trusted.gpg --keyring /etc/apt/trusted.gpg.d/debian-archive-squeeze-automatic.gpg --keyring /etc/apt/trusted.gpg.d/debian-archive-squeeze-stable.gpg --keyring /etc/apt/trusted.gpg.d/debian-archive-wheezy-automatic.gpg --keyring /etc/apt/trusted.gpg.d/debian-archive-wheezy-stable.gpg --recv-keys --keyserver keyserver.ubuntu.com 9D6D8F6BC857C906
gpg: demande de la clef C857C906 sur le serveur hkp keyserver.ubuntu.com
gpg: clef publique « Debian Security Archive Automatic Signing Key (8/jessie) <ftpmaster@debian.org> » importée
gpg: aucune clef de confiance ultime n'a été trouvée
gpg: Quantité totale traitée : 1
gpg: importées : 1 (RSA: 1)
root@debian:/etc/apt/sources.list.d#
```

Figure 15 : Résolution de l'erreur : Aucune clé publique n'est disponible

Refaisons la mise à jour de la liste des packages. Nous verrons que cela se passera sans problème : « apt-get update »

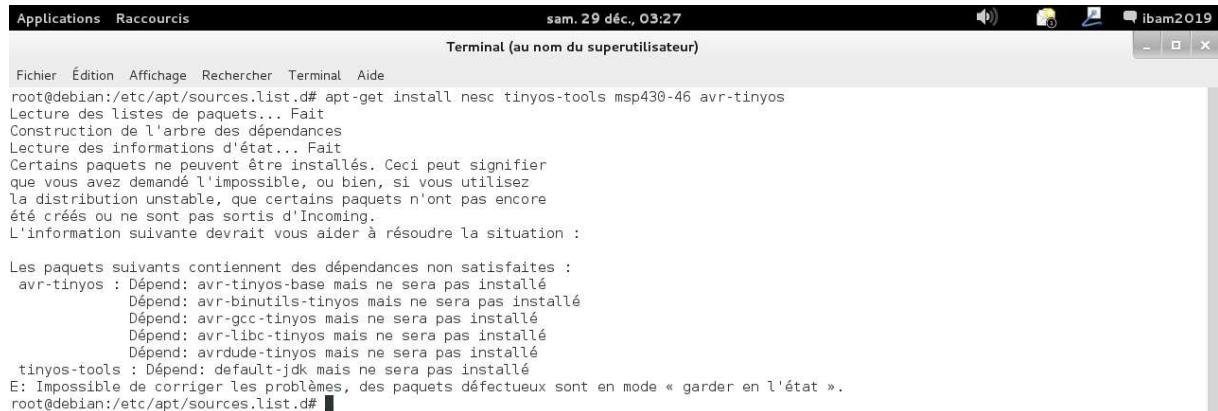


```
Fichier Édition Affichage Rechercher Terminal Aide
root@debian:/etc/apt/sources.list.d# apt-get update
Ign cdrom://[[Debian GNU/Linux 7.1.0 _Wheezy_ - Official i386 DVD Binary-1 20130615-21:54] wheezy Release.gpg
Ign cdrom://[[Debian GNU/Linux 7.1.0 _Wheezy_ - Official i386 DVD Binary-1 20130615-21:54] wheezy Release
Ign cdrom://[[Debian GNU/Linux 7.1.0 _Wheezy_ - Official i386 DVD Binary-1 20130615-21:54] wheezy/contrib i386 Packages/DiffIndex
Ign cdrom://[[Debian GNU/Linux 7.1.0 _Wheezy_ - Official i386 DVD Binary-1 20130615-21:54] wheezy/main i386 Packages/DiffIndex
Ign cdrom://[[Debian GNU/Linux 7.1.0 _Wheezy_ - Official i386 DVD Binary-1 20130615-21:54] wheezy/contrib Translation-fr_FR
Ign cdrom://[[Debian GNU/Linux 7.1.0 _Wheezy_ - Official i386 DVD Binary-1 20130615-21:54] wheezy/contrib Translation-fr
Ign cdrom://[[Debian GNU/Linux 7.1.0 _Wheezy_ - Official i386 DVD Binary-1 20130615-21:54] wheezy/main Translation-fr_FR
Atteint http://tinyprod.net wheezy Release.gpg
Atteint http://tinyprod.net msp430-46 Release.gpg
Atteint http://security.debian.org wheezy/updates Release.gpg
Atteint http://tinyprod.net wheezy Release
Atteint http://tinyprod.net msp430-46 Release
Atteint http://security.debian.org wheezy/updates Release
Atteint http://tinyprod.net wheezy/main i386 Packages
Atteint http://security.debian.org wheezy/updates/main Sources
Atteint http://security.debian.org wheezy/updates/contrib Sources
Atteint http://security.debian.org wheezy/updates/main i386 Packages
Atteint http://security.debian.org wheezy/updates/contrib i386 Packages
Atteint http://tinyprod.net msp430-46/main i386 Packages
Atteint http://security.debian.org wheezy/updates/contrib Translation-en
Atteint http://security.debian.org wheezy/updates/main Translation-en
Ign http://tinyprod.net wheezy/main Translation-fr_FR
Ign http://tinyprod.net wheezy/main Translation-fr
Ign http://tinyprod.net wheezy/main Translation-en
Ign http://tinyprod.net msp430-46/main Translation-fr_FR
Ign http://tinyprod.net msp430-46/main Translation-fr
Ign http://tinyprod.net msp430-46/main Translation-en
Lecture des listes de paquets... Fait
root@debian:/etc/apt/sources.list.d#
```

Figure 16: Mise à jour de la liste des fichiers disponibles dans les dépôts APT

Installons à présent ces packages avec la commande :

« apt-get install nesc tinyos-tools msp430-46 avr-tinyos »



```
Fichier Édition Affichage Rechercher Terminal Aide
root@debian:/etc/apt/sources.list.d# apt-get install nesc tinyos-tools msp430-46 avr-tinyos
Lecture des listes de paquets... Fait
Construction de l'arbre des dépendances
Lecture des informations d'état... Fait
Certains paquets ne peuvent être installés. Ceci peut signifier
que vous avez demandé l'impossible, ou bien, si vous utilisez
la distribution unstable, que certains paquets n'ont pas encore
été créés ou ne sont pas sortis d'Incoming.
L'information suivante devrait vous aider à résoudre la situation :

Les paquets suivants contiennent des dépendances non satisfaites :
avr-tinyos : Dépend: avr-tinyos-base mais ne sera pas installé
          Dépend: avr-binutils-tinyos mais ne sera pas installé
          Dépend: avr-gcc-tinyos mais ne sera pas installé
          Dépend: avr-libc-tinyos mais ne sera pas installé
          Dépend: avrdude-tinyos mais ne sera pas installé
tinyos-tools : Dépend: default-jdk mais ne sera pas installé
E: Impossible de corriger les problèmes, des paquets défectueux sont en mode « garder en l'état ».
root@debian:/etc/apt/sources.list.d#
```

Figure 17 : Installation de nesc, tinyos-tools, msp430-46, avr-tinyos avec erreur

Si nous rencontrons une erreur du genre :

« E : Impossible de corriger les problèmes, des paquets défectueux sont en mode « garder en l'état », retapons la commande et répondons à « Y » aux questions:

« aptitude install nesc tinyos-tools msp430-46 avr-tinyos »

```

Applications Raccourcis sam. 29 déc., 03:29 ibam2019
Terminal (au nom du superutilisateur)

Fichier Édition Affichage Rechercher Terminal Aide
root@debian:/etc/apt/sources.list.d# aptitude install nesc tinyos-tools msp430-46 avr-tinyos
Les NOUVEAUX paquets suivants vont être installés :
  avr-binutils-tinyos{a} avr-libc-tinyos{a} avr-optimal-tinyos{a} avr-tinyos avr-tinyos-base{ab} avrdude-tinyos{a}
  binutils{a} default-jdk{a} gcc{a} gcc-4.7{a} graphviz{a} libc-dev-bin{a} libc6-dev{a} libcdt4{a} libcgraph5{a} libgraph4{a}
  libgvc5{a} libgvpr{a} libice-dev{a} libitm1{a} libpthread-stubs0{a} libpthread-stubs0-dev{a} libsm-dev{a}
  libx11-dev{a} libx11-doc{a} libxau-dev{a} libxcb1-dev{a} libxdmcp-dev{a} libxdot4{a} libxt-dev{a} linux-libc-dev{a} manpages-dev{a}
  msp430-46 msp430-binutils-46{a} msp430-gcc-46{a} msp430-gdb-46{a} msp430-libc-46{a} msp430mcu-46{a} nesc openjdk-6-jdk{a}
  openjdk-7-jdk{a} openjdk-7-jre{a} openjdk-7-jre-headless{ab} python-serial{a} tinyos-tools x11proto-core-dev{a} x11proto-input-dev{a}
  x11proto-kb-dev{a} xorg-sgml-doctools{a} xtrans-dev{a}

Les paquets suivants seront mis à jour :
  default-jre default-jre-headless icedtea-6-jre-cacao icedtea-6-jre-jamvm libc-bin libc6 libx11-6 openjdk-6-jre openjdk-6-jre-headless
  openjdk-6-jre-lib
10 paquets mis à jour, 52 nouvellement installés, 0 à enlever et 348 non mis à jour.
Il est nécessaire de télécharger 258 Mo/270 Mo d'archives. Après dépaquetage, 149 Mo seront utilisés.
Les paquets suivants ont des dépendances non satisfaites :
  openjdk-7-jre-headless : Dépend: libscpi (>= 1.0.10+dfsg) qui est un paquet virtuel
    Casse: icedtea-netx (< 1.4.2) mais 1.3.2-1 est installé et a été conservé
  avr-tinyos-base : Dépend: tinyos-base qui est un paquet virtuel
  libc6-i686 : Pré-Dépend: libc6 (= 2.13-38) mais 2.13-38+deb7u12 doit être installé.

Les actions suivantes permettront de résoudre ces dépendances :

Supprimer les paquets suivants :
1) default-jre
2) default-jre-headless
3) libc6-i686

Conserver les paquets suivants dans leur version actuelle :
4) avr-binutils-tinyos [Non installé]
5) avr-gcc-tinyos [Non installé]
6) avr-libc-tinyos [Non installé]
7) avr-optimal-tinyos [Non installé]
8) avr-tinyos [Non installé]
9) avr-tinyos-base [Non installé]

KABORE Terminal (au nom du ... tinyos.docx - LibreOf...

```

Figure 18 : Installation de nesc, tinyos-tools, msp430-46, avr-tinyos sans erreur

Lorsque nous répondons à la dernière question, si nous voyons apparaître quelque chose du genre « Changement de support : veuillez insérer le disque dont le nom est « Dedian GNU/Linux 7.1.0 _Wheezy_ - Official i386 DVD Binary-1 20130615-21:54 » dans le lecteur « /media/cdrom/ » et presser Entrée. »

```

Applications Raccourcis sam. 29 déc., 03:29 ibam2019
Terminal (au nom du superutilisateur)

Fichier Édition Affichage Rechercher Terminal Aide
7) avr-optimal-tinyos [Non installé]
8) avr-tinyos [Non installé]
9) avr-tinyos-base [Non installé]
10) avrdude-tinyos [Non installé]
11) default-jdk [Non installé]
12) msp430-46 [Non installé]
13) openjdk-7-jdk [Non installé]
14) openjdk-7-jre [Non installé]
15) openjdk-7-jre-headless [Non installé]
16) tinyos-tools [Non installé]

Laisser les dépendances suivantes non satisfaites :
17) linux-image-3.2.0-4-686-pae recommande libc6-i686
18) libc6 recommande libc6-i686
19) avrdude-tinyos recommande avr-tinyos
20) avrdude-tinyos recommande avr-optimal-tinyos

Accepter cette solution ? [Y/n/q/?] Y
Les NOUVEAUX paquets suivants vont être installés :
  binutils{a} gcc{a} gcc-4.7{a} graphviz{a} libc-dev-bin{a} libc6-dev{a} libcdt4{a} libcgraph5{a} libgraph4{a} libgvc5{a} libgvpr{a}
  libice-dev{a} libitm1{a} libpthread-stubs0{a} libpthread-stubs0-dev{a} libsm-dev{a} libx11-dev{a} libx11-doc{a}
  libxau-dev{a} libxcb1-dev{a} libxdmcp-dev{a} libxdot4{a} libxt-dev{a} linux-libc-dev{a} manpages-dev{a} nesc openjdk-6-jdk{a}
  x11proto-core-dev{a} x11proto-input-dev{a} x11proto-kb-dev{a} xorg-sgml-doctools{a} xtrans-dev{a}

Les paquets suivants seront ENLEVÉS :
  default-jre{a} default-jre-headless{a} libc6-i686{a}

Les paquets suivants seront mis à jour :
  icedtea-6-jre-cacao icedtea-6-jre-jamvm libc-bin libc6 libx11-6 openjdk-6-jre openjdk-6-jre-headless openjdk-6-jre-lib
8 paquets mis à jour, 33 nouvellement installés, 3 à enlever et 347 non mis à jour.
Il est nécessaire de télécharger 81,6 Mo/93,7 Mo d'archives. Après dépaquetage, 63,8 Mo seront utilisés.

Voulez-vous continuer ? [Y/n/?] Y
Changement de support : veuillez insérer le disque dont le nom est « Debian GNU/Linux 7.1.0 _Wheezy_ - Official i386 DVD Binary-1 20130615-21:54 » dans le lecteur « /media/cdrom/ » et presser Entrée.

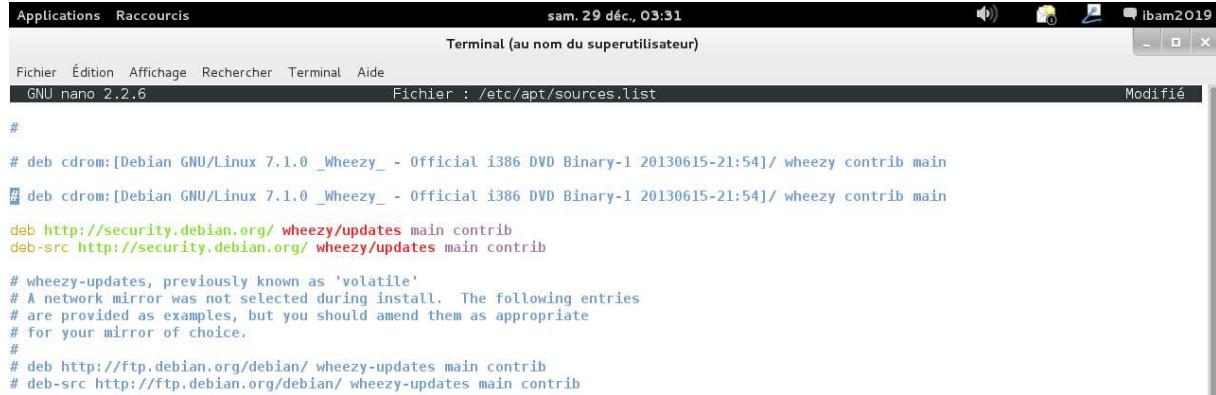
KABORE Terminal (au nom du ... tinyos.docx - LibreOf...

```

Figure 19: Etape avant fin de l'installation de nesc, tinyos-tools, msp430-46, avr-tinyos

Commentons la ligne suivante :

« deb cdrom:[Debian GNU/Linux 7.1.0 _Wheezy_ - Official i386 DVD Binary-1 20130615-21:54]/ wheezy contrib main » dans le fichier « /etc/apt/sources.list »

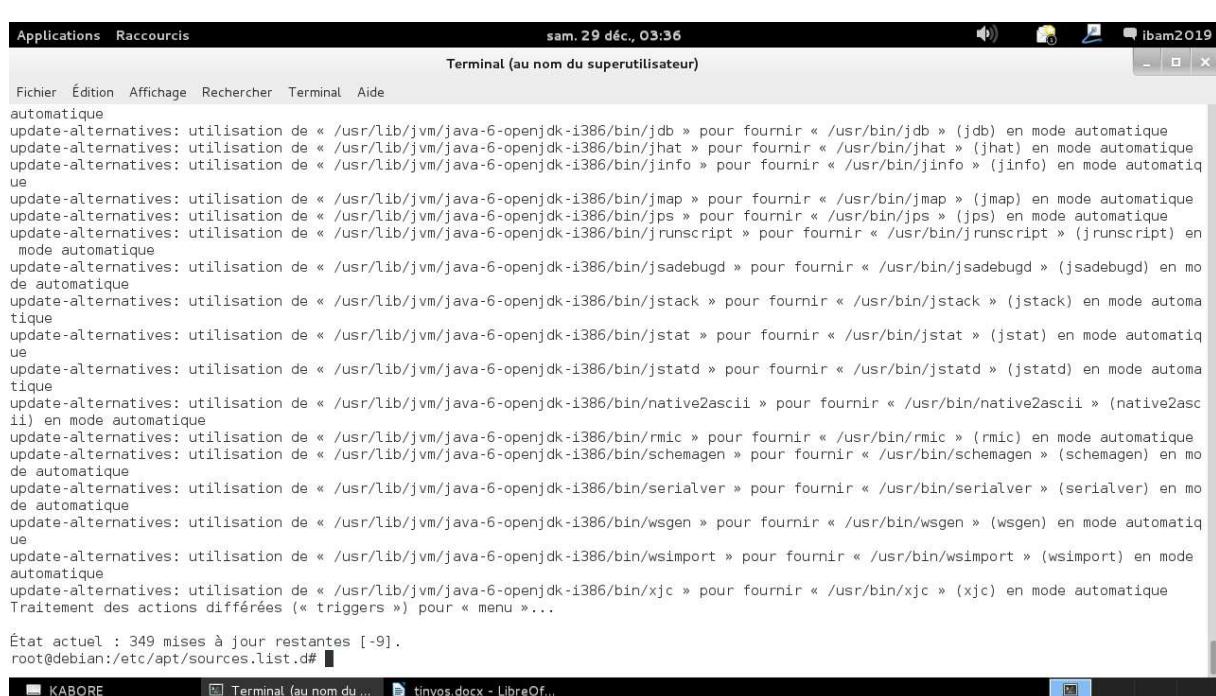


The screenshot shows a terminal window titled "Terminal (au nom du superutilisateur)". The status bar indicates it's "sam. 29 déc., 03:31". The window title is "ibam2019". The terminal content is the /etc/apt/sources.list file, which contains the following text:

```
#  
# deb cdrom:[Debian GNU/Linux 7.1.0 _Wheezy_ - Official i386 DVD Binary-1 20130615-21:54]/ wheezy contrib main  
# deb cdrom:[Debian GNU/Linux 7.1.0 _Wheezy_ - Official i386 DVD Binary-1 20130615-21:54]/ wheezy contrib main  
deb http://security.debian.org/ wheezy/updates main contrib  
deb-src http://security.debian.org/ wheezy/updates main contrib  
# wheezy-updates, previously known as 'volatile'  
# A network mirror was not selected during install. The following entries  
# are provided as examples, but you should amend them as appropriate  
# for your mirror of choice.  
#  
# deb http://ftp.debian.org/debian/ wheezy-updates main contrib  
# deb-src http://ftp.debian.org/debian/ wheezy-updates main contrib
```

Figure 20 : Commentaire d'une ligne dans le fichier « sources.list »

Puis retapons la commande : « aptitude install nesc tinyos-tools msp430-46 avr-tinyos ». Normalement, tout devrait bien se passer et nous devrions avoir quelque chose comme suit :



The screenshot shows a terminal window titled "Terminal (au nom du superutilisateur)". The status bar indicates it's "sam. 29 déc., 03:36". The terminal content shows the output of the aptitude command:

```
automatique  
update-alternatives: utilisation de « /usr/lib/jvm/java-6-openjdk-i386/bin/jdb » pour fournir « /usr/bin/jdb » (jdb) en mode automatique  
update-alternatives: utilisation de « /usr/lib/jvm/java-6-openjdk-i386/bin/jhat » pour fournir « /usr/bin/jhat » (jhat) en mode automatique  
update-alternatives: utilisation de « /usr/lib/jvm/java-6-openjdk-i386/bin/jinfo » pour fournir « /usr/bin/jinfo » (jinfo) en mode automatique  
update-alternatives: utilisation de « /usr/lib/jvm/java-6-openjdk-i386/bin/jmap » pour fournir « /usr/bin/jmap » (jmap) en mode automatique  
update-alternatives: utilisation de « /usr/lib/jvm/java-6-openjdk-i386/bin/jps » pour fournir « /usr/bin/jps » (jps) en mode automatique  
update-alternatives: utilisation de « /usr/lib/jvm/java-6-openjdk-i386/bin/jrunscript » pour fournir « /usr/bin/jrunscript » (jrunscript) en mode automatique  
update-alternatives: utilisation de « /usr/lib/jvm/java-6-openjdk-i386/bin/jsadebugd » pour fournir « /usr/bin/jsadebugd » (jsadebugd) en mode automatique  
update-alternatives: utilisation de « /usr/lib/jvm/java-6-openjdk-i386/bin/jstack » pour fournir « /usr/bin/jstack » (jstack) en mode automatique  
update-alternatives: utilisation de « /usr/lib/jvm/java-6-openjdk-i386/bin/jstat » pour fournir « /usr/bin/jstat » (jstat) en mode automatique  
update-alternatives: utilisation de « /usr/lib/jvm/java-6-openjdk-i386/bin/jstated » pour fournir « /usr/bin/jstated » (jstated) en mode automatique  
update-alternatives: utilisation de « /usr/lib/jvm/java-6-openjdk-i386/bin/native2ascii » pour fournir « /usr/bin/native2ascii » (native2ascii) en mode automatique  
update-alternatives: utilisation de « /usr/lib/jvm/java-6-openjdk-i386/bin/rmic » pour fournir « /usr/bin/rmic » (rmic) en mode automatique  
update-alternatives: utilisation de « /usr/lib/jvm/java-6-openjdk-i386/bin/schemagen » pour fournir « /usr/bin/schemagen » (schemagen) en mode automatique  
update-alternatives: utilisation de « /usr/lib/jvm/java-6-openjdk-i386/bin/serialver » pour fournir « /usr/bin/serialver » (serialver) en mode automatique  
update-alternatives: utilisation de « /usr/lib/jvm/java-6-openjdk-i386/bin/wsgen » pour fournir « /usr/bin/wsgen » (wsgen) en mode automatique  
update-alternatives: utilisation de « /usr/lib/jvm/java-6-openjdk-i386/bin/wsimport » pour fournir « /usr/bin/wsimport » (wsimport) en mode automatique  
update-alternatives: utilisation de « /usr/lib/jvm/java-6-openjdk-i386/bin/xjc » pour fournir « /usr/bin/xjc » (xjc) en mode automatique  
Traitement des actions différées (« triggers ») pour « menu »...  
État actuel : 349 mises à jour restantes [-9].  
root@debian:/etc/apt/sources.list.d#
```

Figure 21 : Installation de nesc, tinyos-tools, msp430-46, avr-tinyos

Tapons par exemple « aptitude show tinyos-tools » pour voir le package « tinyos-tools » est installé.



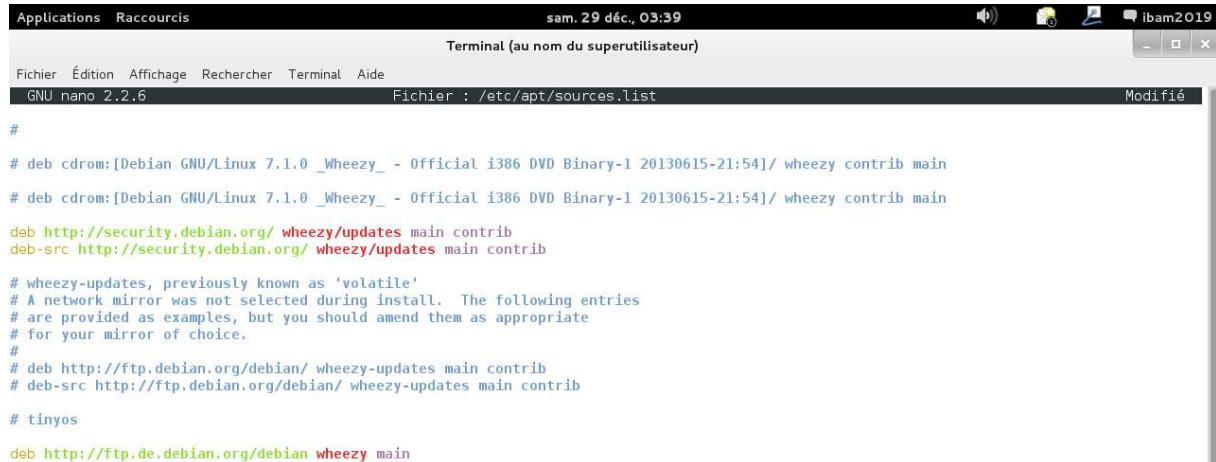
```
Fichier Édition Affichage Rechercher Terminal Aide
root@debian:/etc/apt/sources.list.d# aptitude show tinyos-tools
Paquet : tinyos-tools
Nouveau: oui
État: non installé
Version : 1.4.2-tinyprod1
Priorité : optionnel
Section : devel
Responsable : Eric B. Decker <ccire831@gmail.com>, Andras Biro <andras.biro@ucmote.com>
Architecture : i386
Taille décompressée : 0
Dépend: python, python-serial, perl, nesc, default-jdk, default-jre
Description : Development-tools for TinyOS

root@debian:/etc/apt/sources.list.d#
```

Figure 22 : Vérification de l'installation de « tinyos-tools »

Nous voyons bien que le package n'est pas installé malgré la commande tapé plus haut.

Ajoutons la ligne suivante : « deb http://ftp.de.debian.org/debian wheezy main » au fichier « /etc/apt/sources.list »



```
Fichier Édition Affichage Rechercher Terminal Aide
GNU nano 2.2.6 Fichier : /etc/apt/sources.list Modifié
#
# deb cdrom:[Debian GNU/Linux 7.1.0 _Wheezy_ - Official i386 DVD Binary-1 20130615-21:54]/ wheezy contrib main
# deb cdrom:[Debian GNU/Linux 7.1.0 _Wheezy_ - Official i386 DVD Binary-1 20130615-21:54]/ wheezy contrib main
deb http://security.debian.org/ wheezy/updates main contrib
deb-src http://security.debian.org/ wheezy/updates main contrib

# wheezy-updates, previously known as 'volatile'
# A network mirror was not selected during install. The following entries
# are provided as examples, but you should amend them as appropriate
# for your mirror of choice.
#
# deb http://ftp.debian.org/debian/ wheezy-updates main contrib
# deb-src http://ftp.debian.org/debian/ wheezy-updates main contrib

# tinyos

deb http://ftp.de.debian.org/debian wheezy main
```

Figure 23 : Ajout d'une ligne au fichier « sources.list »

Puis faisons une mise à jour des packages avec la commande : « apt-get update ».



```
Fichier Édition Affichage Rechercher Terminal Aide
root@debian:/opt/tinyos-main/apps/Blink# nano /etc/apt/sources.list
root@debian:/opt/tinyos-main/apps/Blink# apt-get update
Atteint http://tinyprod.net wheezy Release.gpg
Réception de : 1 http://ftp.de.debian.org wheezy Release.gpg [2 373 B]
Réception de : 2 http://ftp.de.debian.org wheezy Release [191 kB]
Atteint http://tinyprod.net msp430-46 Release.gpg
Atteint http://tinyprod.net wheezy Release
Atteint http://security.debian.org wheezy/updates Release.gpg
Réception de : 3 http://ftp.de.debian.org wheezy/main i386 Packages [5 857 kB]
Atteint http://tinyprod.net msp430-46 Release
Atteint http://security.debian.org wheezy/updates Release
Atteint http://tinyprod.net wheezy/main i386 Packages
Atteint http://security.debian.org wheezy/updates/main Sources
Atteint http://security.debian.org wheezy/updates/contrib Sources
Atteint http://tinyprod.net msp430-46/main i386 Packages
Atteint http://security.debian.org wheezy/updates/main i386 Packages
Atteint http://security.debian.org wheezy/updates/contrib i386 Packages
Atteint http://security.debian.org wheezy/updates/contrib Translation-en
Atteint http://security.debian.org wheezy/updates/main Translation-en
Réception de : 4 http://ftp.de.debian.org wheezy/main Translation-fr [751 kB]
Réception de : 5 http://ftp.de.debian.org wheezy/main Translation-en [3 846 kB]
Ign http://tinyprod.net wheezy/main Translation-fr_FR
Ign http://tinyprod.net wheezy/main Translation-fr_
Ign http://tinyprod.net wheezy/main Translation-en
Ign http://tinyprod.net msp430-46/main Translation-fr_FR
Ign http://tinyprod.net msp430-46/main Translation-fr
Ign http://tinyprod.net msp430-46/main Translation-en
10,6 Mo réceptionnés en 24s (431 ko/s)
Lecture des listes de paquets... Fait
W: Aucune clé publique n'est disponible pour la/les clé(s) suivante(s) :
7638D0442B90D010
```

Figure 24 : Mise à jour de la liste des fichiers disponibles dans les dépôts APT

Si nous rencontrons une erreur du genre :

« w : Aucune clé publique n'est disponible pour la/les clé(s) suivante(s) :
7638D0442B90D010 », tapons la commande suivante pour la résoudre:
« apt-key adv --recv-keys --keyserver keyserver.ubuntu.com 7638D0442B90D010 »



```
Fichier Édition Affichage Rechercher Terminal Aide
root@debian:/opt/tinyos-main/apps/Blink# apt-key adv --recv-keys --keyserver keyserver.ubuntu.com 7638D0442B90D010
Executing: gpg --ignore-time-conflict --no-options --no-default-keyring --secret-keyring /tmp/tmp.Wj5MSszJFB --trustdb-name /etc/apt/trustdb.gpg --keyring /etc/apt/trusted.gpg --primary-keyring /etc/apt/trusted.gpg --keyring /etc/apt/trusted.gpg.d/debian-archive-squeeze-automatic.gpg --keyring /etc/apt/trusted.gpg.d/debian-archive-wheezy-stable.gpg --keyring /etc/apt/trusted.gpg.d/debian-archive-wheezy-automatic.gpg --keyring /etc/apt/trusted.gpg.d/debian-archive-wheezy-stable.gpg --recv-keys --keyserver keyserver.ubuntu.com 7638D0442B90D010
gpg: demande de la clef 2B90D010 sur le serveur hkp keyserver.ubuntu.com
gpg: clef 2B90D010 : clef publique « Debian Archive Automatic Signing Key (8/jessie) <ftpmaster@debian.org> » importée
gpg: aucune clef de confiance ultime n'a été trouvée
gpg: Quantité totale traitée : 1
gpg: importées : 1 (RSA: 1)
```

Figure 25 : Résolution de l'erreur relative à la clé publique non disponible

Retapons alors la commande : « aptitude install nesc tinyos-tools msp430-46 avr-tinyos »



```
Fichier Édition Affichage Rechercher Terminal Aide
root@debian:/opt/tinyos-main/apps/Blink# aptitude install nesc tinyos-tools msp430-46 avr-tinyos
Terminal (au nom du superutilisateur)
sam. 29 déc., 04:11
ibam2019

Les NOUVEAUX paquets suivants vont être installés :
  avr-binutils-tinyos{a} avr-gcc-tinyos{a} avr-libc-tinyos{a} avr-tinyos avr-tinyos-base{ab} avrdude-tinyos{a}
  default-jdk{a} libice-dev{a} libpthread-stubs0{a} libpthread-stubs0-dev{a} libscapl{a} libsm-dev{a} libx11-dev{a} libx11-doc{a}
  libxau-dev{a} libxcb1-dev{a} libxdmcp-dev{a} libxt-dev{a} lksctp-tools{a} msp430-46 msp430-binutils-46{a} msp430-gcc-46{a}
  msp430-gdb-46{a} msp430-libc-46{a} msp430mcu-46{a} openjdk-7-jdk{a} openjdk-7-jre{a} openjdk-7-jre-headless{a} python-serial{a}
  tinyos-tools xlproto-core-dev{a} xlproto-input-dev{a} xlproto-kb-dev{a} xorg-sgml-doctools{a} xtrans-dev{a}

Les paquets suivants seront mis à jour :
  default-jre default-jre-headless icedtea-netx icedtea-netx-common
4 paquets mis à jour, 36 nouvellement installés, 0 à enlever et 411 non mis à jour.
Il est nécessaire de télécharger 183 Mo d'archives. Après dépaquetage, 101 Mo seront utilisés.

Les paquets suivants ont des dépendances non satisfaites :
  avr-tinyos-base : Dépend: tinyos-base qui est un paquet virtuel
Les actions suivantes permettront de résoudre ces dépendances :

  Conserver les paquets suivants dans leur version actuelle :
1)   avr-binutils-tinyos [Non installé]
2)   avr-gcc-tinyos [Non installé]
3)   avr-libc-tinyos [Non installé]
4)   avr-optional-tinyos [Non installé]
5)   avr-tinyos [Non installé]
6)   avr-tinyos-base [Non installé]
7)   avrdude-tinyos [Non installé]

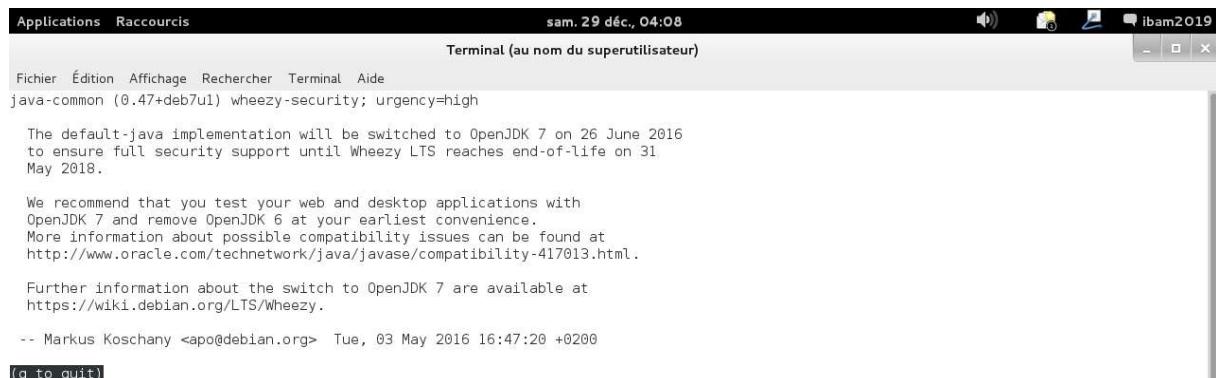
  Laisser les dépendances suivantes non satisfaites :
8)   avrdude-tinyos recommande avr-tinyos
9)   avrdude-tinyos recommande avr-optional-tinyos

Accepter cette solution ? [Y/n/q/?] Y
Les NOUVEAUX paquets suivants vont être installés :
  default-jdk{a} libice-dev{a} libpthread-stubs0{a} libpthread-stubs0-dev{a} libscapl{a} libsm-dev{a} libx11-dev{a} libx11-doc{a}
  libxau-dev{a} libxcb1-dev{a} libxdmcp-dev{a} libxt-dev{a} lksctp-tools{a} msp430-46 msp430-binutils-46{a} msp430-gcc-46{a}

KABORE Terminal (au nom du ... tinyos.docx - LibreOf...
```

Figure 26 : Etape 1 du succès de l'installation de nesc, tinyos-tools, msp430-46, avr-tinyos

Lorsqu'il nous sera demandé de taper sur la touche « q », faisons-le.



```
Fichier Édition Affichage Rechercher Terminal Aide
java-common (0.47+deb7u1) wheezy-security; urgency=high
Terminal (au nom du superutilisateur)
sam. 29 déc., 04:08
ibam2019

The default-java implementation will be switched to OpenJDK 7 on 26 June 2016
to ensure full security support until Wheezy LTS reaches end-of-life on 31
May 2018.

We recommend that you test your web and desktop applications with
OpenJDK 7 and remove OpenJDK 6 at your earliest convenience.
More information about possible compatibility issues can be found at
http://www.oracle.com/technetwork/java/javase/compatibility-417013.html.

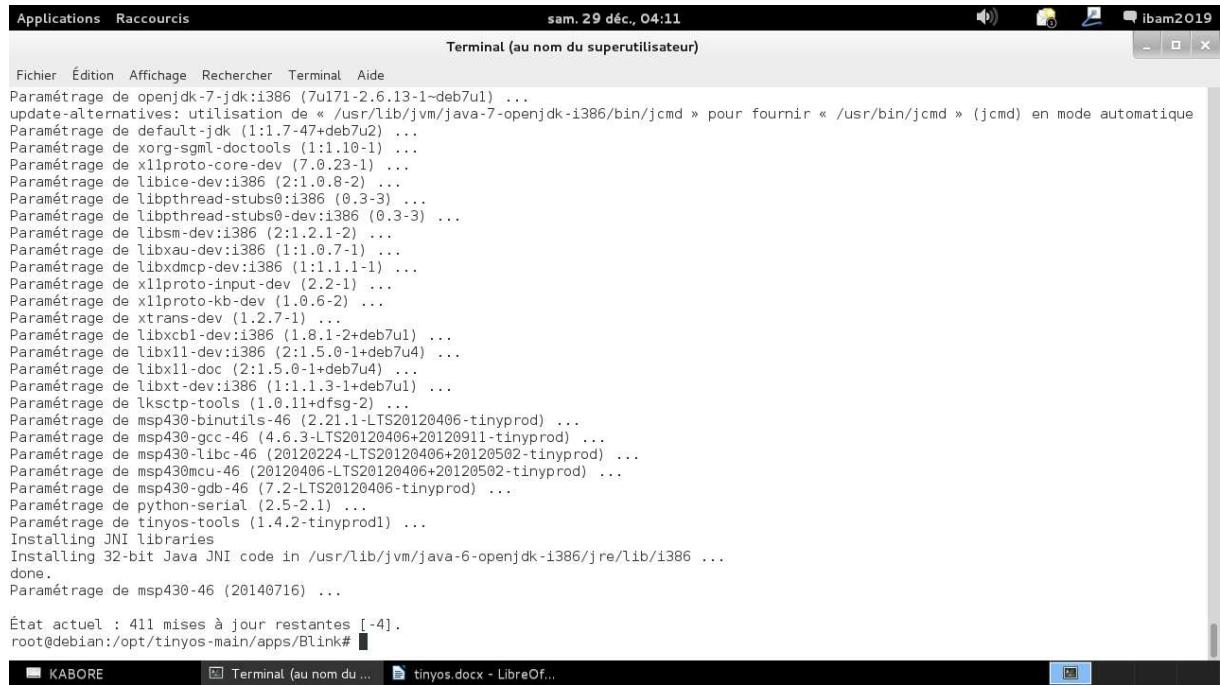
Further information about the switch to OpenJDK 7 are available at
https://wiki.debian.org/LTS/Wheezy.

-- Markus Koschany <apo@debian.org> Tue, 03 May 2016 16:47:20 +0200

(q to quit)
```

Figure 27 : Etape 2 du succès de l'installation de nesc, tinyos-tools, msp430-46, avr-tinyos

Si tout se passe bien, nous devons avoir ceci :



```
Fichier Édition Affichage Rechercher Terminal Aide
Paramétrage de openjdk-7-jdk:i386 (7u171-2.6.13-1~deb7u1) ...
update-alternatives: utilisation de « /usr/lib/jvm/java-7-openjdk-i386/bin/jcmd » pour fournir « /usr/bin/jcmd » (jcmd) en mode automatique
Paramétrage de default-jdk (1:1.7-47+deb7u2) ...
Paramétrage de xorg-sgml-doctools (1:1.10-1) ...
Paramétrage de x11proto-core-dev (7.0.23-1) ...
Paramétrage de libICE-dev:i386 (2:1.0.8-2) ...
Paramétrage de libpthread-stubs0:i386 (0.3-3) ...
Paramétrage de libpthread-stubs0-dev:i386 (0.3-3) ...
Paramétrage de libSM-dev:i386 (2:1.2.1-2) ...
Paramétrage de libXAU-dev:i386 (1:1.0.7-1) ...
Paramétrage de libXdmcp-dev:i386 (1:1.1.1-1) ...
Paramétrage de x11proto-input-dev (2.2-1) ...
Paramétrage de x11proto-kb-dev (1.0.6-2) ...
Paramétrage de xtrans-dev (1.2.7-1) ...
Paramétrage de libxcb1-dev:i386 (1.8.1-2+deb7u1) ...
Paramétrage de libx11-dev:i386 (2:1.5.0-1+deb7u4) ...
Paramétrage de libx11-drv (2:1.5.0-1+deb7u4) ...
Paramétrage de libxt-dev:i386 (1:1.1.3-1+deb7u1) ...
Paramétrage de lksctp-tools (1:0.11+dfsg-2) ...
Paramétrage de msp430-binutils-46 (2.21.1-LTS20120406-tinyprod) ...
Paramétrage de msp430-gcc-46 (4.6.3-LTS20120406+20120911-tinyprod) ...
Paramétrage de msp430-libc-46 (20120224-LTS20120406+20120502-tinyprod) ...
Paramétrage de msp430mcu-46 (20120406-LTS20120406+20120502-tinyprod) ...
Paramétrage de msp430-gdb-46 (7.2-LTS20120406-tinyprod) ...
Paramétrage de python-serial (2.5-2.1) ...
Paramétrage de tinyos-tools (1.4.2-tinyprod1) ...
Installing JNI libraries
Installing 32-bit Java JNI code in /usr/lib/jvm/java-6-openjdk-i386/jre/lib/i386 ...
done.
Paramétrage de msp430-46 (20140716) ...

État actuel : 411 mises à jour restantes [-4].
root@debian:/opt/tinyos-main/apps/Blink#
```

Figure 28 : Succès de l'installation de nesc, tinyos-tools, msp430-46, avr-tinyos

Plaçons nous dans le dossier « opt » en tapant la commande « cd /opt » et tapons la commande : « wget http://github.com/tinyos/tinyos-release/archive/tinyos-2_1_2.tar.gz »



```
Fichier Édition Affichage Rechercher Terminal Aide
root@debian:/etc/apt/sources.list.d# cd /opt
root@debian:/opt# wget http://github.com/tinyos/tinyos-release/archive/tinyos-2_1_2.tar.gz
- 2018-12-29 03:49:22- http://github.com/tinyos/tinyos-release/archive/tinyos-2_1_2.tar.gz
Résolution de github.com (github.com)... 140.82.118.4, 140.82.118.3
Connexion vers github.com (github.com)[140.82.118.4]:80...connecté.
requête HTTP transmise, en attente de la réponse...301 Moved Permanently
Emplacement: https://github.com/tinyos/tinyos-release/archive/tinyos-2_1_2.tar.gz [suivant]
- 2018-12-29 03:49:24- https://github.com/tinyos/tinyos-release/archive/tinyos-2_1_2.tar.gz
Connexion vers github.com (github.com)[140.82.118.4]:443...connecté.
requête HTTP transmise, en attente de la réponse...302 Found
Emplacement: https://codeload.github.com/tinyos/tinyos-release/tar.gz/tinyos-2_1_2 [suivant]
- 2018-12-29 03:49:27- https://codeload.github.com/tinyos/tinyos-release/tar.gz/tinyos-2_1_2
Résolution de codeload.github.com (codeload.github.com)... 192.30.253.120, 192.30.253.121
Connexion vers codeload.github.com (codeload.github.com)[192.30.253.120]:443...connecté.
requête HTTP transmise, en attente de la réponse...200 OK
Longueur: non spécifié [application/x-gzip]
Sauvegarde en : «/tinyos-2_1_2.tar.gz»

[          <=>                               ] 21 952 048 367K/s  ds 1m 56s

2018-12-29 03:51:24 (185 KB/s) - «/tinyos-2_1_2.tar.gz» sauvé [21952048]
```

Figure 29 : Téléchargement de TinyOS

Tapons la commande : « tar xf/tinyos-2_1_2.tar.gz »

```
root@debian:/opt# tar xf/tinyos-2_1_2.tar.gz
root@debian:/opt# ls
tinyos-2_1_2.tar.gz /tinyos-release/tinyos-2_1_2
```

Figure 30 : Décompression de TinyOS

Ceci aura pour effet de décompresser le fichier « tar xf tinyos-2_1_2.tar.gz » au nom de « tinyos-release-tinyos-2_1_2 ». Renommons-le au nom de « tinyos-main ».

```
root@debian:/opt# mv tinyos-release-tinyos-2_1_2 tinyos-main
root@debian:/opt# ls
tinyos-2_1_2.tar.gz  tinyos-main
root@debian:/opt#
```

Figure 31 : Changement du nom de TinyOS

Créons un fichier du nom de « tinyos.env » à l'aide de la commande « nano tinyos.env » dans le répertoire « /opt/tinyos-main » et plaçons les instructions suivantes :

- # Here we setup the environment
- # variables needed by the/tinyos
- # make system
- export TOSROOT="/opt/tinyos-main"
- export TOSDIR="\$TOSROOT/tos"
- export CLASSPATH=\$CLASSPATH:\$TOSROOT/support/sdk/java
- export MAKERULES="\$TOSROOT/support/make/Makerules"
- export PYTHONPATH=\$PYTHONPATH:\$TOSROOT/support/sdk/python
- echo "setting up TinyOS on source path \$TOSROOT"



The screenshot shows a terminal window titled 'Terminal (au nom du superutilisateur)' running on a Linux desktop. The window title bar includes the date 'sam. 29 déc., 03:55' and the user 'ibam2019'. The menu bar has 'Applications' and 'Raccourcis'. The toolbar includes icons for volume, network, and window control. The status bar shows 'Fichier : tinyos.env' and 'Modifié'. The terminal content is a script named 'tinyos.env' containing the following code:

```
# Here we setup the environment
# variables needed by the/tinyos
# make system

export TOSROOT="/opt/tinyos-main"
export TOSDIR="$TOSROOT/tos"
export CLASSPATH=$CLASSPATH:$TOSROOT/support/sdk/java
export MAKERULES="$TOSROOT/support/make/Makerules"
export PYTHONPATH=$PYTHONPATH:$TOSROOT/support/sdk/python

echo "setting up TinyOS on source path $TOSROOT"
```

Figure 32 : Crédit d'un environnement pour TinyOS

Enregistrons avec les touches « ctrl + o » puis faisons « entrée » puis « ctrl + x » pour quitter l'éditeur de texte.

Ouvrons le fichier « .bashrc » avec la commande « nano ~/.bashrc » et ajoutons la ligne suivante « source /opt/tinyos-main/tinyos.env » à la fin du fichier.



```

Applications Raccourcis sam. 29 déc., 03:57 ibam2019
Terminal (au nom du superutilisateur)
Fichier Édition Affichage Rechercher Terminal Aide
GNU nano 2.2.6 Fichier : /root/.bashrc Modifié
# ~/bashrc: executed by bash(1) for non-login shells.

# Note: PS1 and umask are already set in /etc/profile. You should not
# need this unless you want different defaults for root.
# PS1='${debian_chroot:+($debian_chroot)}\h:\w\$ '
# umask 022

# You may uncomment the following lines if you want 'ls' to be colorized:
# export LS_OPTIONS='--color=auto'
# eval "$dircolors"
# alias ls='ls $LS_OPTIONS'
# alias ll='ls $LS_OPTIONS -l'
# alias l='ls $LS_OPTIONS -la'
#
# Some more alias to avoid making mistakes:
# alias rm='rm -i'
# alias cp='cp -i'
# alias mv='mv -i'

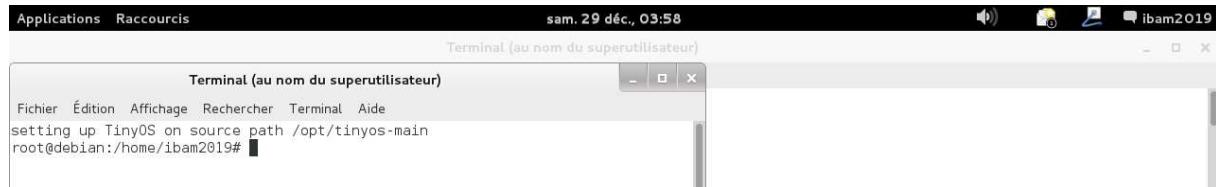
# tinyos

source /opt/tinyos-main/tinyos.env

```

Figure 33 : Ajout de l'environnement TinyOS au fichier « .bashrc »

Ouvrons un terminal et connectons en tant que super administrateur. Nous devons voir apparaître ceci « setting up TinyOS on source path /opt/tinyos-main » en haut du terminal.



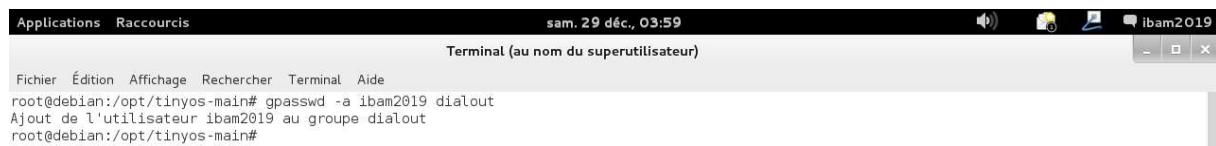
```

Applications Raccourcis sam. 29 déc., 03:58 ibam2019
Terminal (au nom du superutilisateur)
Fichier Édition Affichage Rechercher Terminal Aide
setting up TinyOS on source path /opt/tinyos-main
root@debian:/home/ibam2019# 

```

Figure 34 : Vérification de l'installation de TinyOS

Dans l'optique de pouvoir programmer des capteurs, nous devons accéder aux ports série. Cela peut être fait en rejoignant le groupe qui accorde ce privilège. Tapons la commande : « gpasswd -a ibam2019 dialout »



```

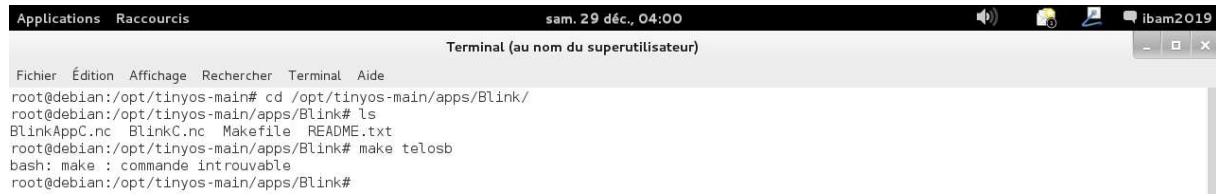
Applications Raccourcis sam. 29 déc., 03:59 ibam2019
Terminal (au nom du superutilisateur)
Fichier Édition Affichage Rechercher Terminal Aide
root@debian:/opt/tinyos-main# gpasswd -a ibam2019 dialout
Ajout de l'utilisateur ibam2019 au groupe dialout
root@debian:/opt/tinyos-main#

```

Figure 35 : Accord de privilège à l'utilisateur

Ou « ibam2019 » correspond au nom de l'utilisateur sur notre système d'exploitation Debian.

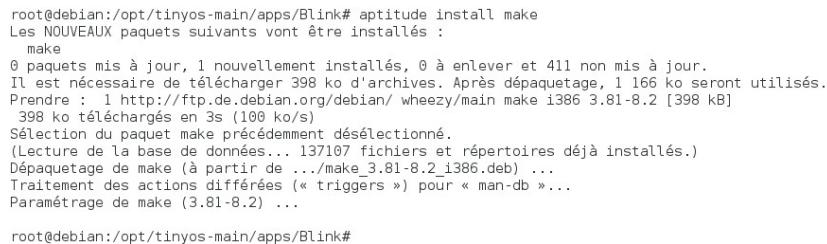
Effectuons un test en tapant « make telosb » dans le répertoire « /opt/tinyos-main/apps/Blink/ ».



```
Fichier Édition Affichage Rechercher Terminal Aide
root@debian:/opt/tinyos-main# cd /opt/tinyos-main/apps/Blink/
root@debian:/opt/tinyos-main/apps/Blink# ls
BlinkAppC.nc  BlinkC.nc  Makefile  README.txt
root@debian:/opt/tinyos-main/apps/Blink# make telosb
bash: make : commande introuvable
root@debian:/opt/tinyos-main/apps/Blink#
```

Figure 36 : Echec 1 de compilation de l'application « Blink » pour la plateforme « telosb »

Si tout se passe bien, nous devons avoir le message suivant : « writing TOS image ». Dans le cas contraire, si nous obtenons une erreur du « bash : make : commande introuvable », installons le package « make » avec la commande : « aptitude install make ».

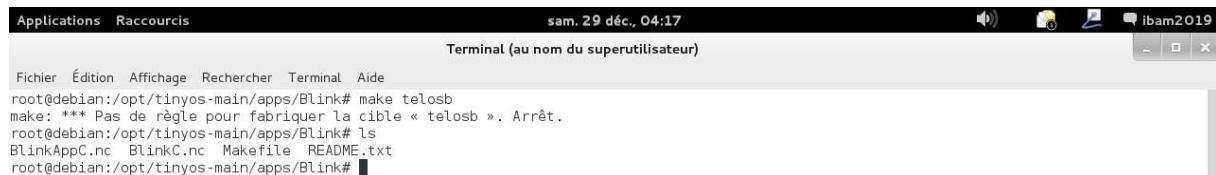


```
root@debian:/opt/tinyos-main/apps/Blink# aptitude install make
Les NOUVEAUX paquets suivants vont être installés :
  make
0 paquets mis à jour, 1 nouvellement installés, 0 à enlever et 411 non mis à jour.
Il est nécessaire de télécharger 398 ko d'archives. Après dépaquetage, 1 166 ko seront utilisés.
Prendre : 1 http://ftp.de.debian.org/debian/ wheezy/main make i386 3.81-8.2 [398 kB]
  398 ko téléchargés en 3s (100 ko/s)
Sélection du paquet make précédemment désélectionné.
(Lecture de la base de données... 137107 fichiers et répertoires déjà installés.)
Dépaquetage de make (à partir de .../make_3.81-8.2_i386.deb) ...
Traitement des actions différencées (« triggers ») pour « man-db »...
Paramétrage de make (3.81-8.2) ...

root@debian:/opt/tinyos-main/apps/Blink#
```

Figure 37 : Installation de « make »

Retapons la commande « make telosb ».



```
Fichier Édition Affichage Rechercher Terminal Aide
root@debian:/opt/tinyos-main/apps/Blink# make telosb
make: *** Pas de règle pour fabriquer la cible « telosb ». Arrêt.
root@debian:/opt/tinyos-main/apps/Blink# ls
BlinkAppC.nc  BlinkC.nc  Makefile  README.txt
root@debian:/opt/tinyos-main/apps/Blink#
```

Figure 38 : Echec 2 de la compilation de l'application « Blink » pour la plateforme « telosb »

Nous voyons bien que la commande marche bien maintenant, mais néanmoins nous obtenons une erreur. Nous allons devoir redémarrer Debian pour que toutes les configurations précédemment initiées prennent effet.

Retapons la commande « make telosb » dans le répertoire « /opt/tinyos-main/apps/Blink/ ».

```

Applications Raccourcis sam. 29 déc., 04:21 ibam2019
Terminal (au nom du superutilisateur)

Fichier Édition Affichage Rechercher Terminal Aide
setting up TinyOS on source path /opt/tinyos-main
root@debian:/home/ibam2019# cd /opt/tinyos-main/apps/Blink/
root@debian:/opt/tinyos-main/apps/Blink# ls
BlinkAppC.nc BlinkC.nc Makefile README.txt
root@debian:/opt/tinyos-main/apps/Blink# make telosb
mkdir -p build/telosb
compiling BlinkAppC to a telosb binary
ncc -o build/telosb/main.exe -Os -fnesc-separator=_ -Wall -Wshadow -Wnesc-all -target=telosb -fnesc-cfile=build/telosb/app.c -board= -DDEFINED_TOS_AM_GROUP=0x22 -DIDENT_APPNAME=\"BlinkAppC\" -DIDENT_USERNAME=\"root\" -DIDENT_HOSTNAME=\"debian\" -DIDENT_USERHASH=0xdbc5c03L -DIDENT_TIMESTAMP=0x5c26e812L -DIDENT_UIDHASH=0xa96870c0L BlinkAppC.nc -lm
      compiled BlinkAppC to build/telosb/main.exe
      2538 bytes in ROM
      56 bytes in RAM
msp430-objcopy --output-target=ihex build/telosb/main.exe build/telosb/main.ihex
writing TOS image
root@debian:/opt/tinyos-main/apps/Blink# 

```

Figure 39 : Succès de la compilation de l’application « Blink » pour la plateforme « telosb »

Comme tout s’est bien passé, nous obtenons le message suivant : « writing TOS image ». Ceci montre bien que tinyos s’est bien installé.

2.3. Configuration de l’environnement de développement

Afin de pouvoir programmer au langage de programmation nesc, nous allons installer l’IDE eclipse. Nous allons utiliser la version 4.2.1 en 32bits (**eclipse-SDK-4.2.1-linux-gtk.tar.gz**) puisque notre OS Debian est en 32bits. Nous pouvons également le télécharger, à l’adresse suivante :

« <https://archive.eclipse.org/eclipse/downloads/drops4/R-4.2.1-201209141800/download.php?dropFile=eclipse-SDK-4.2.1-linux-gtk.tar.gz> »

Une fois téléchargé, nous allons le décompresser dans le dossier « /opt »



Figure 40 : Décompression d’eclipse

Nous allons ensuite ajouter eclipse aux applications installées de Debian pour permettre un accès facile. Avant tout, nous allons changer le propriétaire du dossier « /opt » pour l’attribuer à l’utilisateur « ibam2019 ». Pour se faire, tapons la commande suivante : « chown ibam2019 /opt »

Effectuons un lien symbolique de l’exécutable d’eclipse vers le répertoire « /usr/bin » afin que les utilisateurs de la machine puissent utiliser Eclipse. La commande est le suivant : « ln -s /opt/eclipse/eclipse /usr/bin/eclipse ». Cela suppose que nous avons décompressé l’archive d’eclipse dans le répertoire « /opt » comme nous nous l’avons précisé plus haut.

Si nous voulons avoir l'icône d'eclipse comme dans le menu Démarrer de Windows, créons un fichier nommé « *eclipse.desktop* » dans le répertoire « « /usr/share/applications » en utilisant la commande suivante : « *nano /usr/share/applications/eclipse.desktop* », puis plaçons-y le code suivant :

```
[Desktop Entry]
Encoding=UTF-8
Name=Eclipse 4.2.1
Comment=Eclipse Juno
Exec=/usr/bin/eclipse
Icon=/opt/eclipse/icon.xpm
Categories=Application;Development;Java;IDE
Version=1.0
Type=Application
Terminal=0
```



The screenshot shows a terminal window titled "Terminal (au nom du superutilisateur)". The date and time at the top right are "dim. 30 déc., 19:00". The window title bar includes icons for volume, network, and user "ibam2019". The terminal window has a dark background. At the top, there's a menu bar with "Applications" and "Raccourcis". Below it is a toolbar with "Fichier", "Édition", "Affichage", "Rechercher", "Terminal", and "Aide". The status bar at the bottom shows "GNU nano 2.2.6" and "Fichier : eclipse.desktop". The main area of the terminal contains the text of the desktop entry file:

```
[Desktop Entry]
Encoding=UTF-8
Name=Eclipse 4.2.1
Comment=Eclipse Juno
Exec=/usr/bin/eclipse
Icon=/opt/eclipse/icon.xpm
Categories=Application;Development;Java;IDE
Version=1.0
Type=Application
Terminal=0
```

Figure 41 : Ajout d'eclipse dans les applications

Sauvegardons le fichier puis fermons-le.

Nous pouvons maintenant lancer eclipse comme suit :



Figure 42 : Test d'ajout d'eclipse dans les applications

Pour pouvoir programmer au langage de programmation nesc sur eclipse, nous allons devoir installer des plugins de « cadena » sur celui-ci. Cadena est un cadre de développement et de modélisation intégré extensible basé sur Eclipse. Cadena dépend de quelques fonctionnalités, notamment d’Eclipse Modeling Framework (EMF) et de Graphical Editing Framework (GEF, que nous devons installer avant de pouvoir installer les plugins. Si ces fonctionnalités ne sont pas installées, le gestionnaire de mise à jour affichera un avertissement. Nous allons installer ces fonctionnalités manuellement. Nous pouvons les télécharger respectivement, aux adresses suivantes :

- https://www.eclipse.org/downloads/download.php?file=/modeling/emf/emf/drops/2.9.2/R201402031126/emf-runtime-2.9.2.zip&mirror_id=1068
- <https://www.eclipse.org/modeling/download.php?file=/tools/gef/downloads/drops/3.8.0/R201206112118/GEF-runtime-3.8.0.zip>

Une fois téléchargé, décompressons les deux fichiers sur le répertoire suivant : « /opt/eclipse ». S'il nous est demandé de fusionner ou de remplacer certains fichiers, acceptons-le. Cela suppose qu'eclipse n'est pas lancé.

A présent, nous pouvons lancer eclipse, puis faisons « Help → Install New → Software »

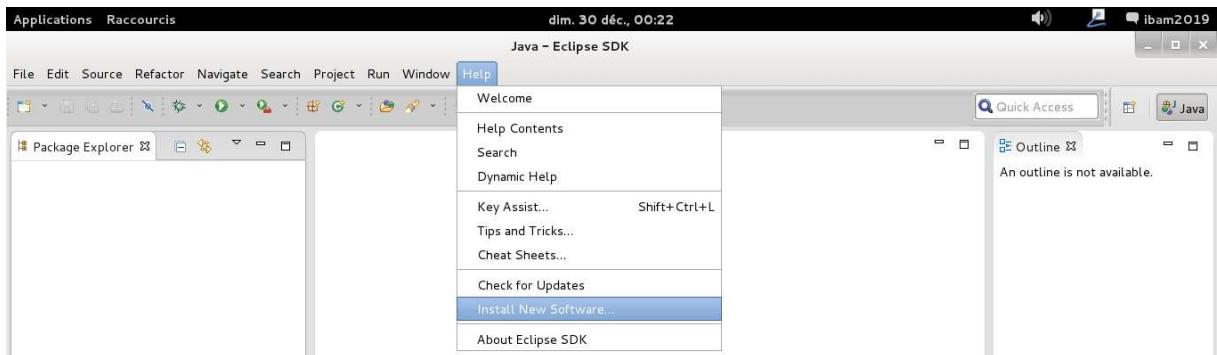


Figure 43 : Etape 1 d'installation du plugin cadena

Ensuite, cliquons sur « Add ... »

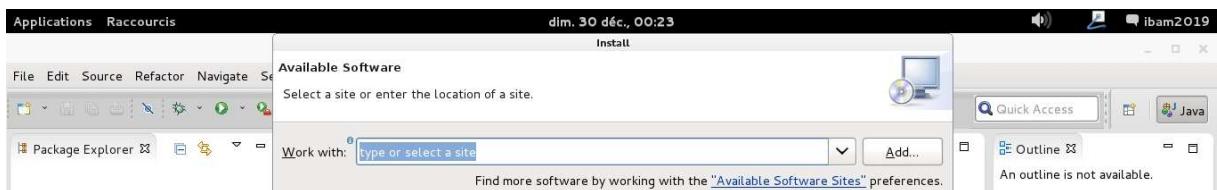


Figure 44 : Etape 2 d'installation du plugin cadena

Dans la boite de dialogue, au niveau de :

- Name, entrons ceci : Cadena update site
- Location, entrons ceci : <http://cadena.projects.cis.ksu.edu/update>

Cliquons ensuite sur « OK »

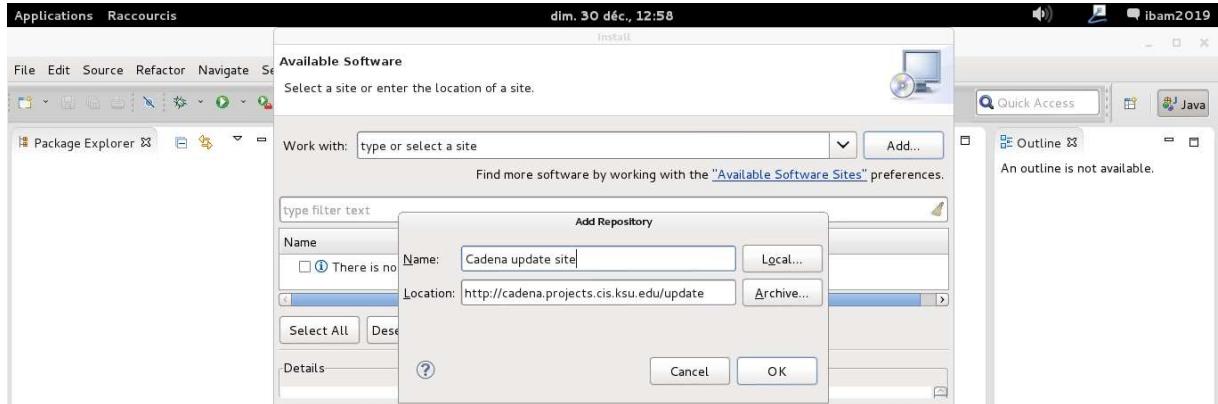


Figure 45 : Etape 3 d'installation du plugin cadena

Cochons « Core », déroulons « Platform », puis cochons seulement « Cadena TinyOS Platform ». Cliquons ensuite sur « Next ».

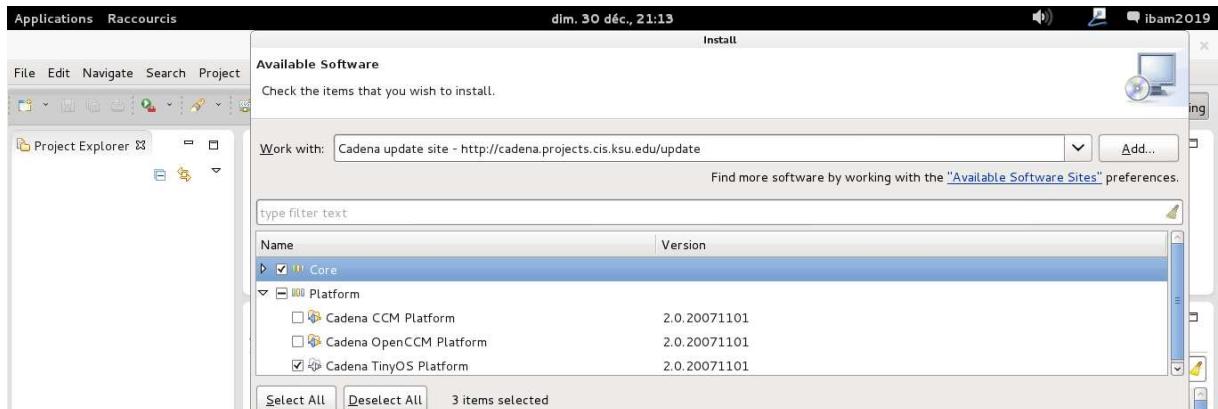


Figure 46 : Etape 4 d'installation du plugin cadena

Patientons un instant.

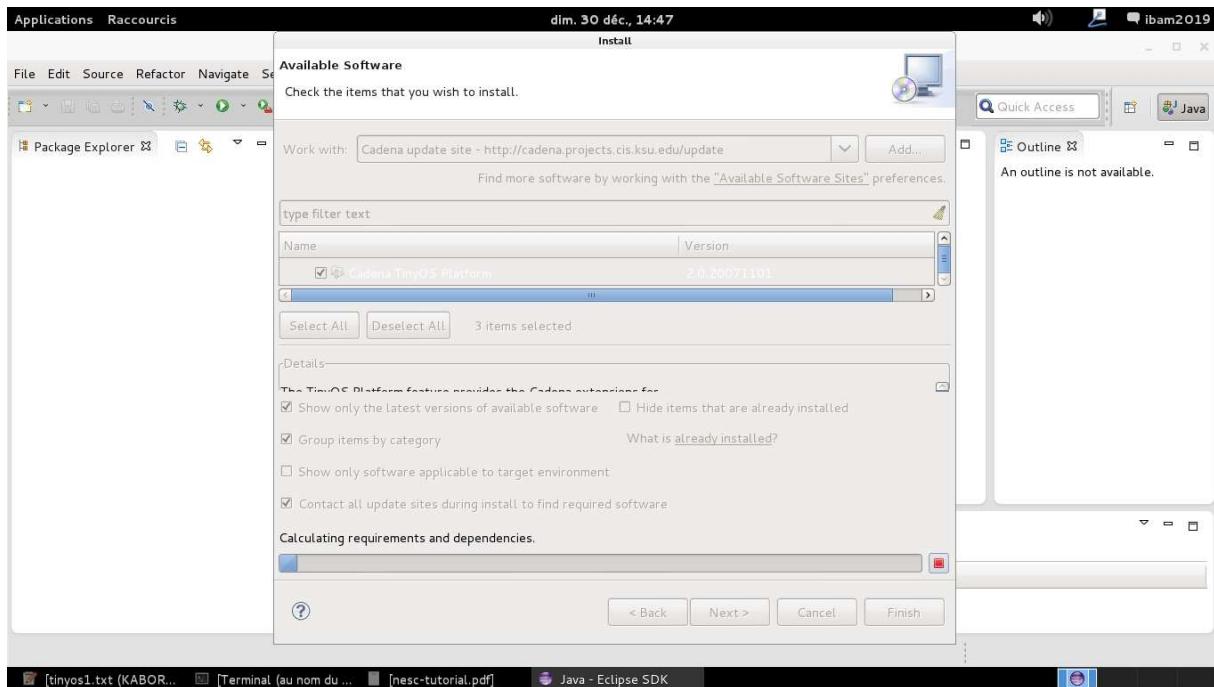


Figure 47 : Etape 5 d'installation du plugin cadena

Cliquons sur « Next ».

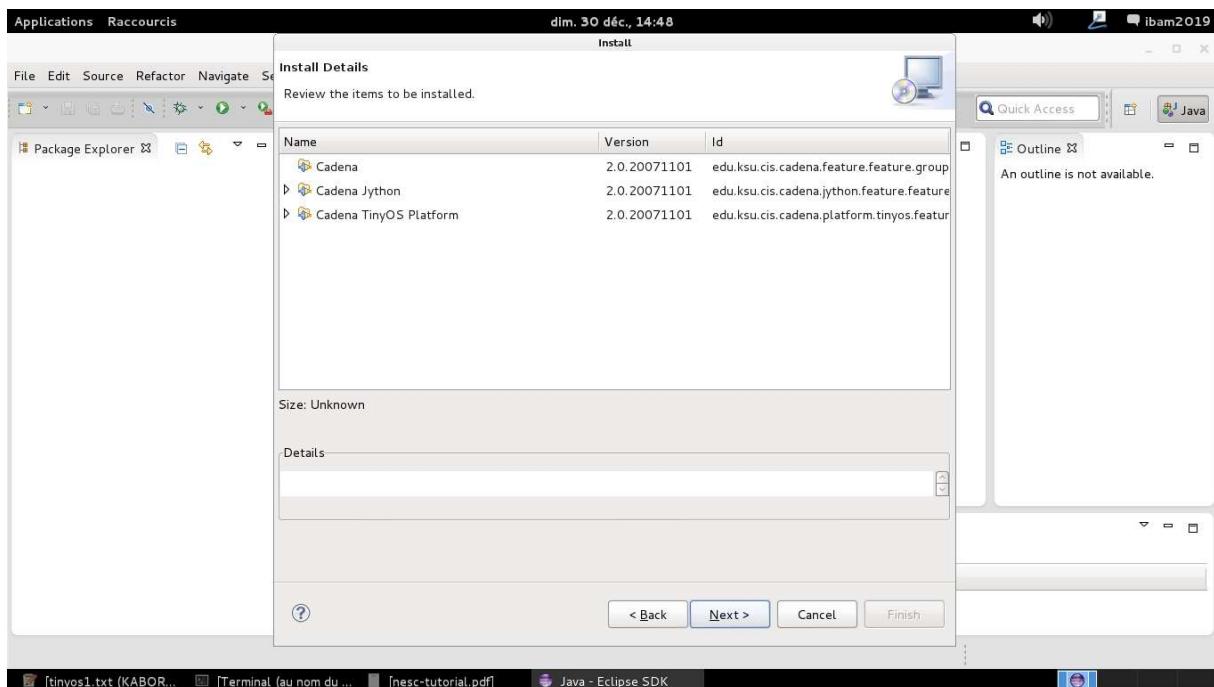


Figure 48 : Etape 6 d'installation du plugin cadena

Acceptons les termes de licence, cliquons sur « Finish »

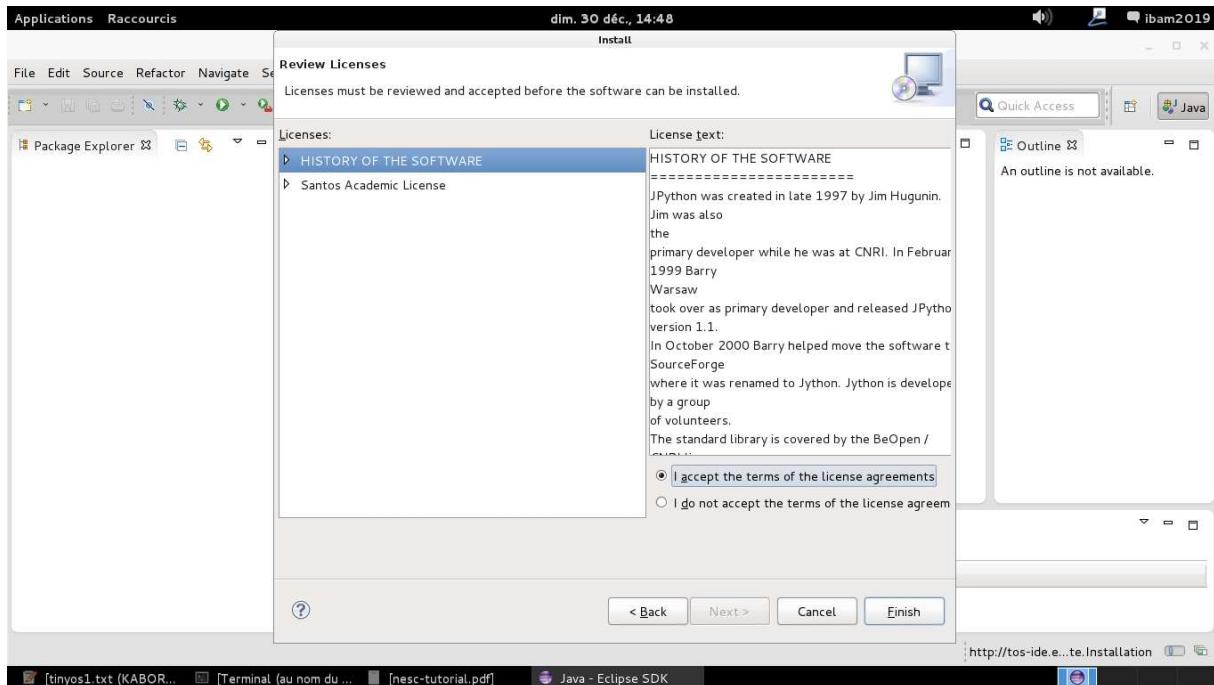


Figure 49 : Etape 7 d'installation du plugin cadena

Patientons un instant.

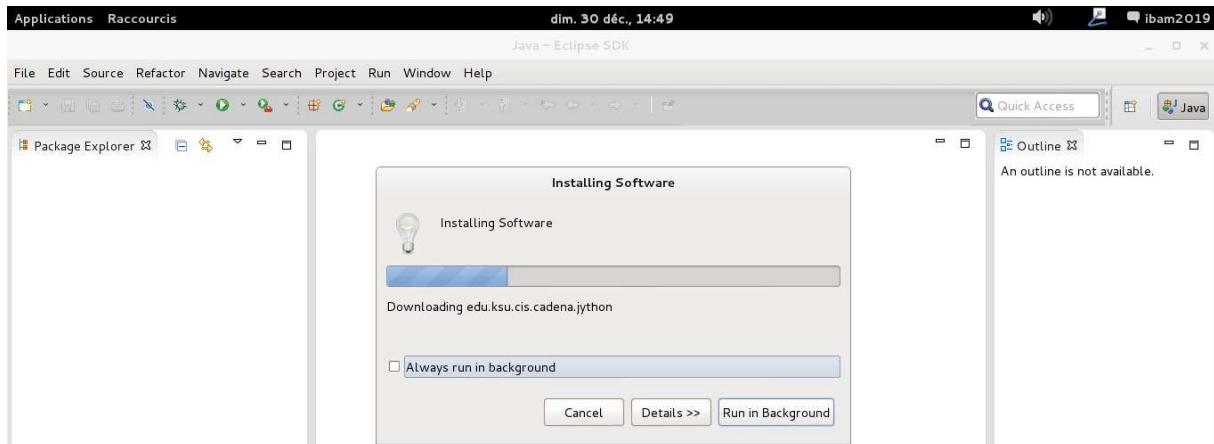


Figure 50 : Etape 8 d'installation du plugin cadena

Cliquons sur « OK ».

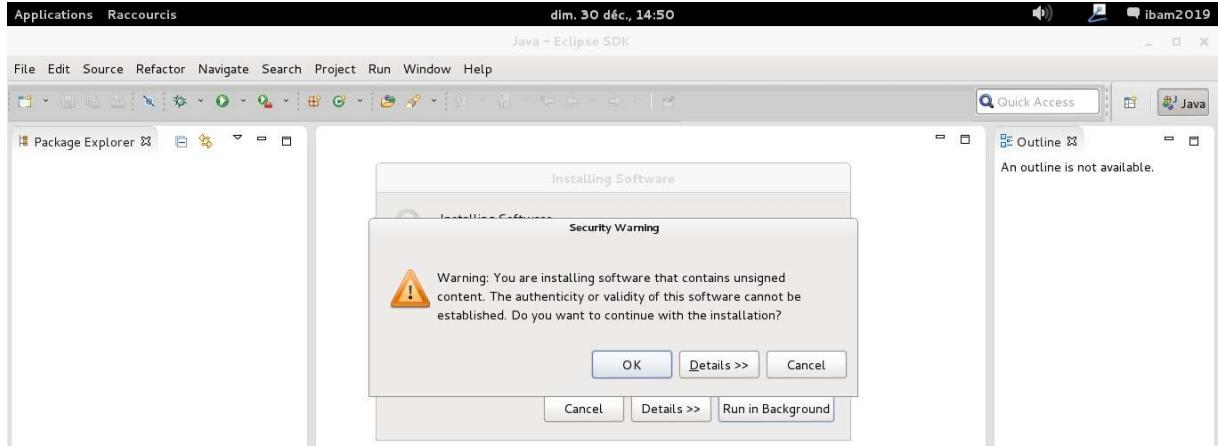


Figure 51 : Etape 9 d'installation du plugin cadena

Cliquons sur « Yes » pour redémarrer eclipse.

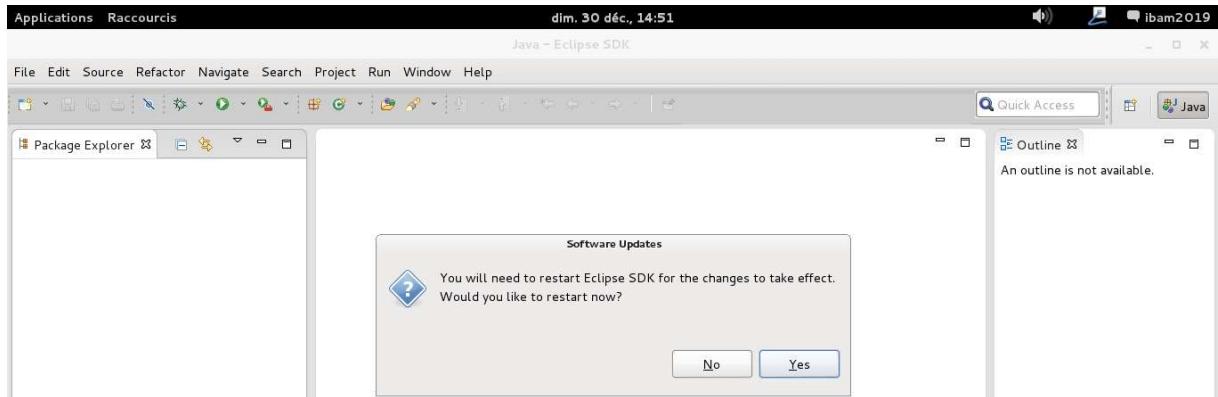


Figure 52 : Fin d'installation du plugin « cadena »

Pour voir si le plugin de TinyOS s'est bien installé, faisons « File → New → Other ».

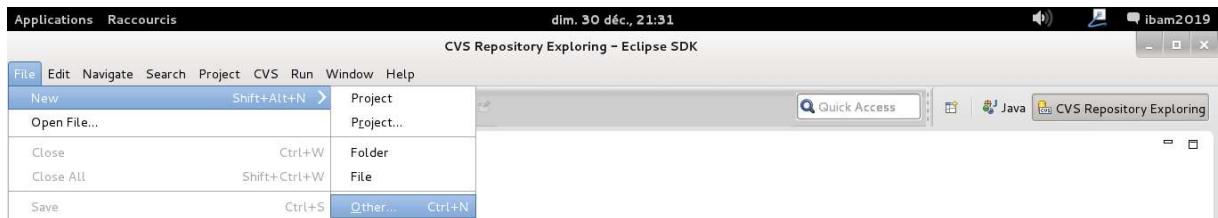


Figure 53 : Vérification de l'installation du plugin de TinyOS

Nous voyons bien « TinyOS Project » dans la boîte de dialogue.

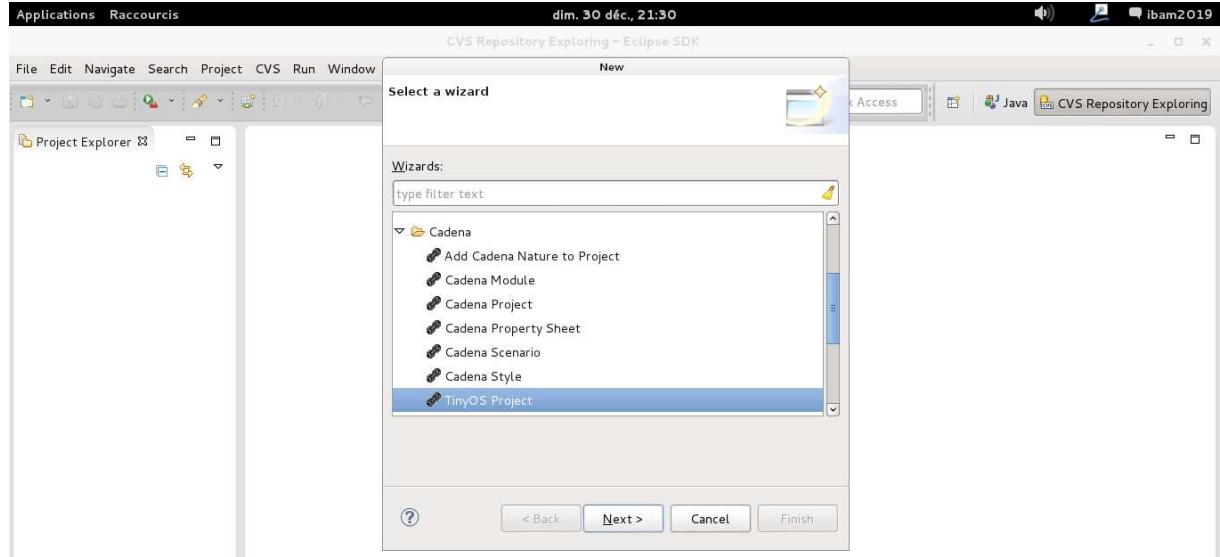


Figure 54 : Succès de vérification de l'installation du plugin de TinyOS

2.4. Simulateur TOSSIM

2.4.1. Présentation de TOSSIM

TOSSIM est un simulateur à événements discrets pour les réseaux de capteurs TinyOS. Il permet aux utilisateurs de déboguer, tester et analyser des algorithmes dans un environnement contrôlé et reproductible. Sachant que TOSSIM fonctionne sur un PC, les utilisateurs peuvent examiner leur code de TinyOS en utilisant les débogueurs et autres outils de développement. Il se concentre sur la simulation TinyOS et son exécution, au lieu de la simulation du monde réel. Alors que TOSSIM peut être utilisée pour comprendre les causes de certains comportements observé dans le monde réel. Il ne doit pas être utilisé pour les évaluations absolues.

2.4.2. Installation de JTOSSIM

TOSSIM dispose d'une GUI JTOSSIM. Il permet de définir les paramètres de simulation (comme les paramètres de la radio et la topologie du réseau) et offre une visualisation différente des résultats des simulations. On peut télécharger JTOSSIM à l'adresse suivante :

<https://sourceforge.net/projects/jtossim/files/latest/download>

Une fois téléchargé, décompressons le dans le dossier « /opt ». Ouvrons le dossier décompressé, puis faisons : « clic droit sur JTosSim.jar → Ouvrir avec OpenJDK Java 6 Runtime »



Figure 55 : Ouverture du simulateur « JTossim »

Nous devons obtenir une fenêtre de ce genre.

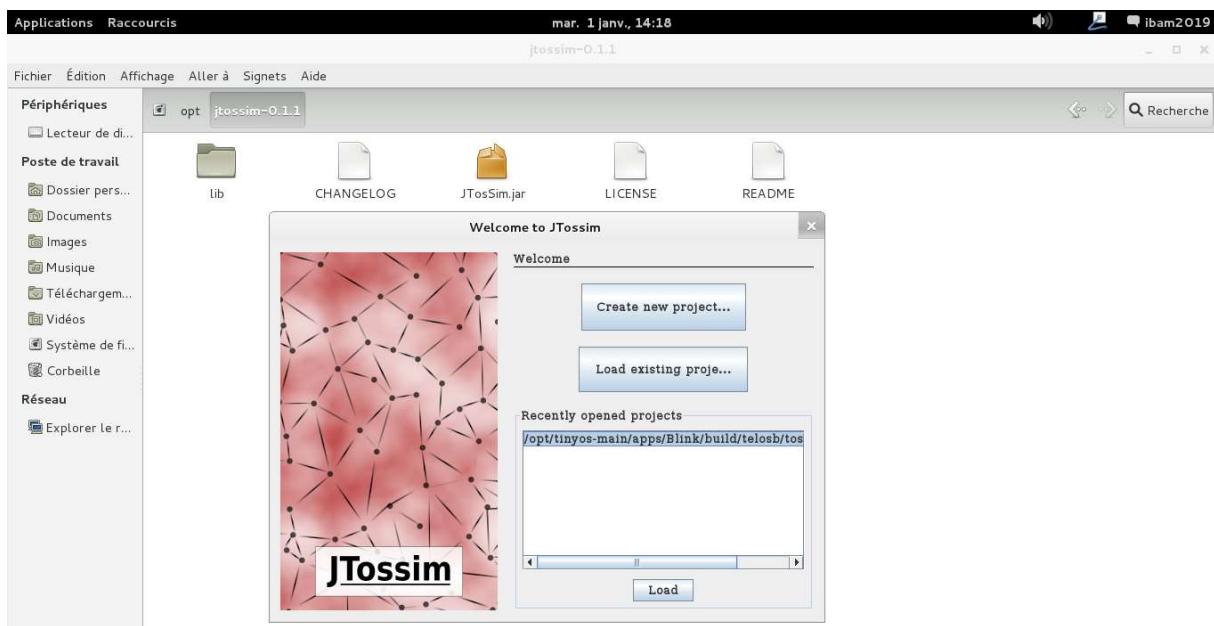


Figure 56 : Ouverture d'un projet déjà existant dans JTossim

Cliquons sur « Load existing project ... ». En guise de test, nous allons utiliser le programme « Blink ». Pour cela chargeons le fichier « tos_image.xml » qui se trouve dans le répertoire « /opt/tinyos-main/apps/Blink/build/telosb »

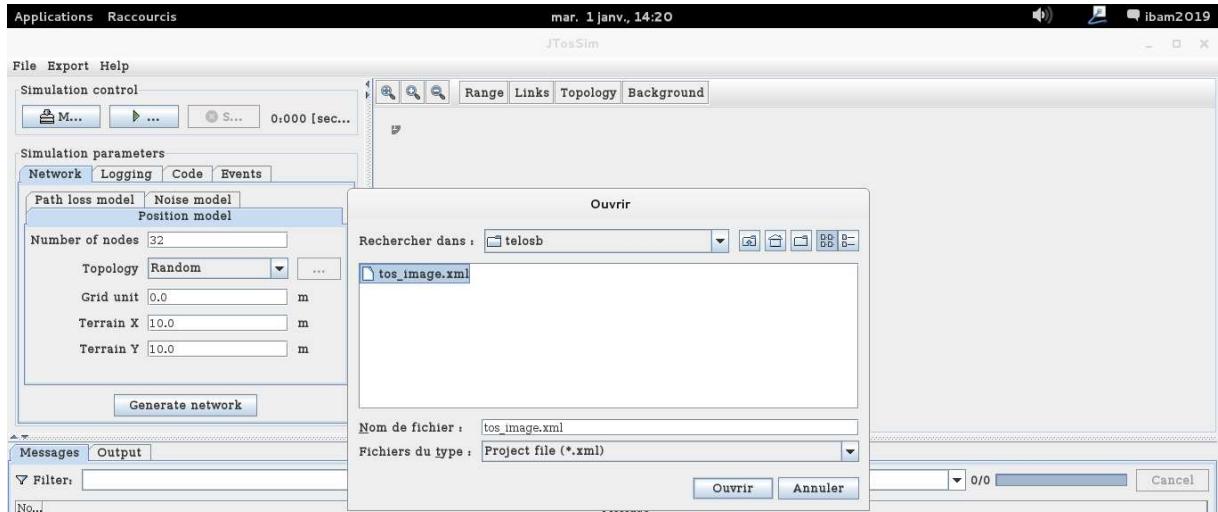


Figure 57 : Ouverture du projet Blink

Lorsqu'on clic sur « ouvrir », on doit avoir quelque chose comme suit :

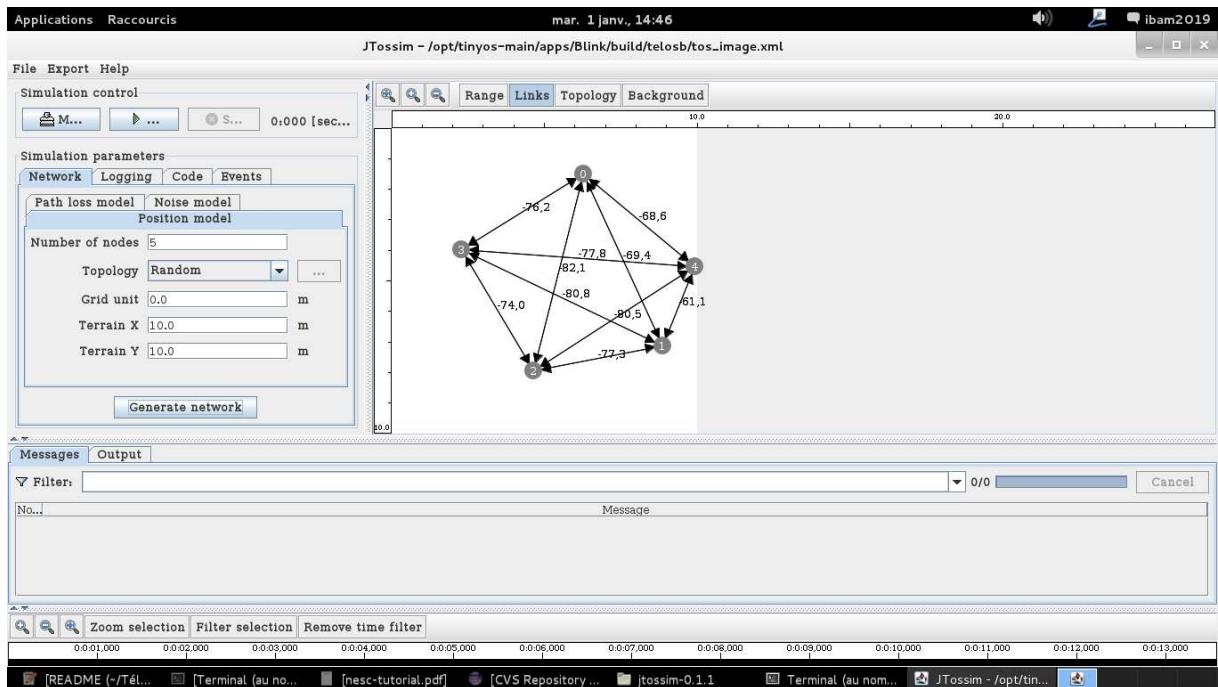


Figure 58 : Projet « Blink » dans « JTossim »

Ici, nous avons changé le nombre de capteur à 5.

2.5. Emulateur MSPSim

2.5.1. Présentation de MSPSim

MSPSim est un émulateur du microcontrôleur MSP430. Il prend en charge le chargement des fichiers de microprogrammes IHEX et ELF et dispose d'outils de surveillance de la pile, de définition de points d'arrêt et de profilage.

La famille MSP430 de Texas Instruments (TI) comprend des microcontrôleurs 16 bits à faible consommation d'énergie. Les périphériques comprennent un convertisseur analogique-numérique 12 bits, une horloge temps réel, des temporiseurs, une interface série asynchrone / synchrone (avec modes SPI / I2C) et un contrôleur DMA.

Le microcontrôleur MSP430 est utilisé dans l'appareil TelosB. TelosB est issu de la technologie Memscic (old Crossbow). Il est composé du microcontrôleur MSP430 (le MSP430F1611) et de la puce radio CC2420. Le microcontrôleur de ce type fonctionne à 4,15 MHz et possède une mémoire vive interne de 10 Ko et une mémoire flash de programme de 48 Ko.

2.5.2. Installation de MSPSim

Après avoir téléchargé la dernière version de MSPSIM à partir de l'adresse suivante :

« <https://sourceforge.net/directory/system-administration/emulators/os:windows/> », nous allons le décompresser dans le répertoire « /opt », puis nous allons le renommer en « mspsim ». Pour l'installer, positionnons nous dans le répertoire « /opt/mspsim », puis lançons la commande « make » comme suit :



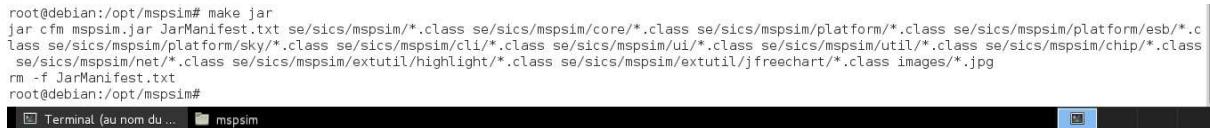
```

root@debian:/opt/mspsim# make
javac -deprecation -classpath ".:lib/jfreechart-1.0.11.jar:lib/jcommon-1.0.14.ja
r" se/sics/mspsim/Main.java
./se/sics/mspsim/ui/WindowUtils.java:36: warning: unmappable character for encoding UTF8
* Author : Joakim Eriksson, Niclas Finne, Fredrik Österlind
^
./se/sics/mspsim/net/HC01PacketHandler.java:49: warning: sun.java2d.SunGraphicsE
nvironment is internal proprietary API and may be removed in a future release
import sun.java2d.SunGraphicsEnvironment.TTFFilter;
^
./se/sics/mspsim/net/HC01PacketHandler.java:49: warning: sun.java2d.SunGraphicsE
nvironment is internal proprietary API and may be removed in a future release
import sun.java2d.SunGraphicsEnvironment.TTFFilter;
^
3 warnings
javac -deprecation -classpath ".:lib/jfreechart-1.0.11.jar:lib/jcommon-1.0.14.ja
r" se/sics/mspsim/platform/esb/ESBGui.java
javac -deprecation -classpath ".:lib/jfreechart-1.0.11.jar:lib/jcommon-1.0.14.ja
r" se/sics/mspsim/platform/sky/MotelIVNode.java
javac -deprecation -classpath ".:lib/jfreechart-1.0.11.jar:lib/jcommon-1.0.14.ja
r" se/sics/mspsim/platform/sky/SkyNode.java
javac -deprecation -classpath ".:lib/jfreechart-1.0.11.jar:lib/jcommon-1.0.14.ja
r" se/sics/mspsim/platform/sky/TelosNode.java
javac -deprecation -classpath ".:lib/jfreechart-1.0.11.jar:lib/jcommon-1.0.14.ja
r" se/sics/mspsim/ui/DoubleBarChart.java
javac -deprecation -classpath ".:lib/jfreechart-1.0.11.jar:lib/jcommon-1.0.14.jar" se/sics/mspsim/util/CCITT_CRC.java
javac -deprecation -classpath ".:lib/jfreechart-1.0.11.jar:lib/jcommon-1.0.14.jar" se/sics/mspsim/util/DotDiagram.java
se/sics/mspsim/util/DotDiagram.java:36: warning: unmappable character for encoding UTF8
* Author : Joakim Eriksson, Niclas Finne, Fredrik Österlind
^
1 warning
javac -deprecation -classpath ".:lib/jfreechart-1.0.11.jar:lib/jcommon-1.0.14.jar" se/sics/mspsim/util/NetworkPacket.java
javac -deprecation -classpath ".:lib/jfreechart-1.0.11.jar:lib/jcommon-1.0.14.jar" se/sics/mspsim/util/PluginBundle.java
javac -deprecation -classpath ".:lib/jfreechart-1.0.11.jar:lib/jcommon-1.0.14.jar" se/sics/mspsim/util/PluginRepository.java
javac -deprecation -classpath ".:lib/jfreechart-1.0.11.jar:lib/jcommon-1.0.14.jar" se/sics/mspsim/util/PrefixConfigManager.java

```

Figure 59 : Etape 1 de compilation de l'émulateur mspsim

Ensuite lançons la commande « make jar ». Cela aura pour effet de créer un fichier « mspsim.jar » dans le répertoire dont nous nous trouvons.



```
root@debian:/opt/mspsim# make jar
jar cfm mspsim.jar JarManifest.txt se/sics/mspsim/*.class se/sics/mspsim/core/*.class se/sics/mspsim/platform/*.class se/sics/mspsim/platform/esb/*.class se/sics/mspsim/platform/sky/*.class se/sics/mspsim/cli/*.class se/sics/mspsim/ui/*.class se/sics/mspsim/util/*.class se/sics/mspsim/chip/*.class se/sics/mspsim/net/*.class se/sics/mspsim/extutil/highlight/*.class se/sics/mspsim/extutil/jfreechart/*.class images/*.jpg
rm -f JarManifest.txt
root@debian:/opt/mspsim#
```

Figure 60 : Etape 2 de compilation de l'émulateur mspSim

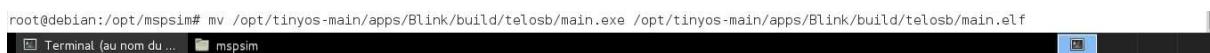
Pour pouvoir lancer mspsim facilement, nous allons créer un alias. Pour cela, tapons la commande suivante : « alias mspsim="java -jar /opt/mspsim/mspsim.jar" ». Désormais, nous pouvons simplement lancer mspsim avec le mot clé « mspsim ».



```
root@debian:/opt/mspsim# alias mspsim="java -jar /opt/mspsim/mspsim.jar"
root@debian:/opt/mspsim#
```

Figure 61 : Création d'un alias « mspSim »

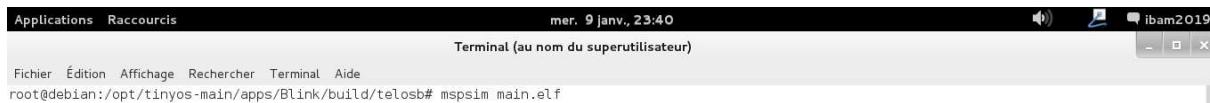
En guise de test, nous allons utiliser le programme Blink. Avant tout, nous allons renommer le fichier « /opt/tinyos-main/apps/Blink/build/telosb/main.exe » en « /opt/tinyos-main/apps/Blink/build/telosb/main.elf ».



```
root@debian:/opt/mspsim# mv /opt/tinyos-main/apps/Blink/build/telosb/main.exe /opt/tinyos-main/apps/Blink/build/telosb/main.elf
root@debian:/opt/mspsim#
```

Figure 62: Changement du nom de main.exe en main.elf

Plaçons nous dans le répertoire « /opt/tinyos-main/apps/Blink/build/telosb » puis lançons la commande suivante : « mspsim main.elf ».



```
Applications Raccourcis mer. 9 janv., 23:40
Terminal (au nom du superutilisateur)
Fichier Édition Affichage Rechercher Terminal Aide
root@debian:/opt/tinyos-main/apps/Blink/build/telosb# mspsim main.elf
```

Figure 63 : Lancement de la simulation de l'application « Blink »

Si des erreurs persistent, veillons redémarrer notre système DEBIAN, puis relançons la commande. Dans le cas, ou tout se passe bien, nous verrons cinq fenêtres s'ouvrir. Dans l'une des fenêtres, nous verrons l'image d'un capteur et trois leds qui clignotent comme le montre la figure ci-dessous :

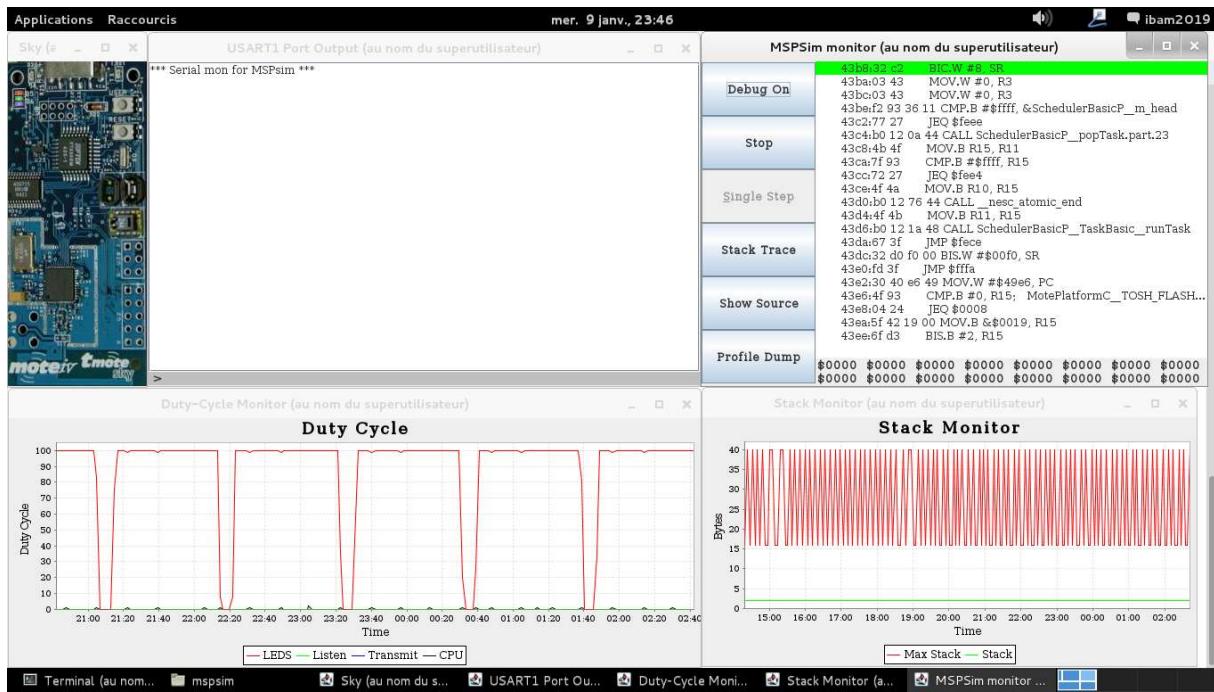


Figure 64 : Programme « Blink » dans mspSim

Cela montre bien que mspsim est bien installé.

3. Cas pratique

3.1. Aperçu

Le but de cette section est de programmer une application TinyOs qui nous permette de recueillir la température d'un environnement à travers un capteur. Ce capteur sera connecté à un port série de notre ordinateur. Pour visualiser la température collectée par notre capteur, nous utiliserons un programme « terminale de port série » ou « serial port terminal » en anglais.

Nous rappelons que le langage de programmation que nous allons utiliser sera « nesC » et la plateforme du capteur sera « telosb ».

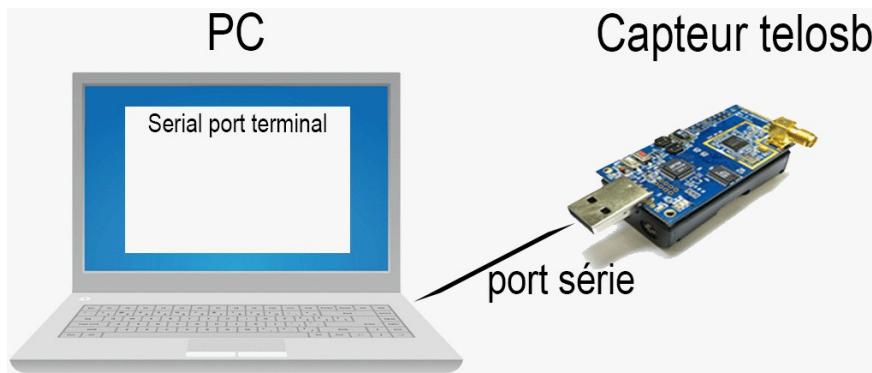


Figure 65 : Architecture du cas pratique

3.2. Application « Temperature »

Launch Eclipse as follows : Application → Programming → Eclipse 4.2.1



Figure 66 : Lancement d'Eclipse

Select the location of the « Workspace », that is to say the folder where we will save our project.

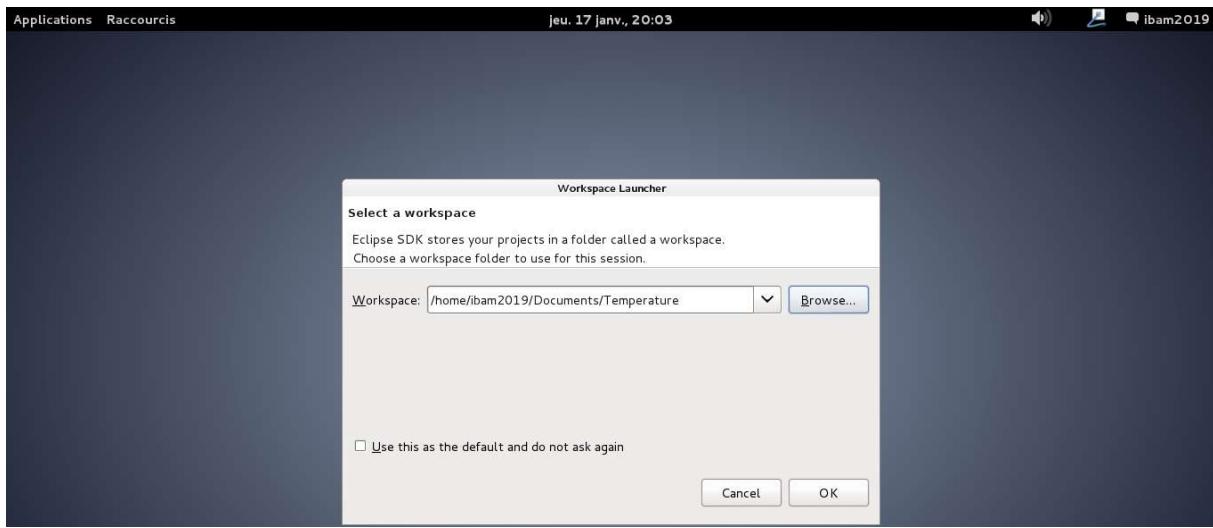


Figure 67 : Choix du workspace

Créons notre projet en faisant : File → New → Other...

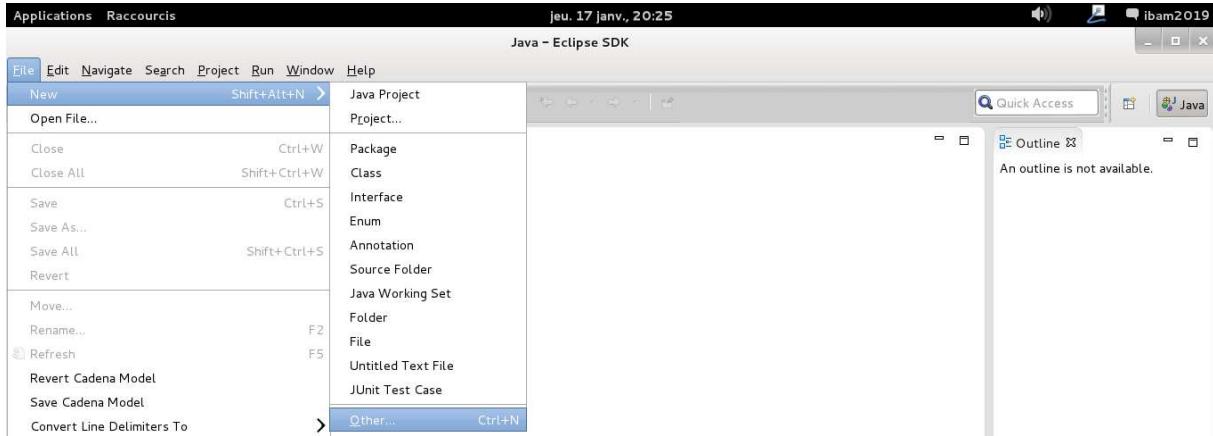


Figure 68 : Création d'un projet

Choisissons « TinyOS Project » et cliquons sur « Next »

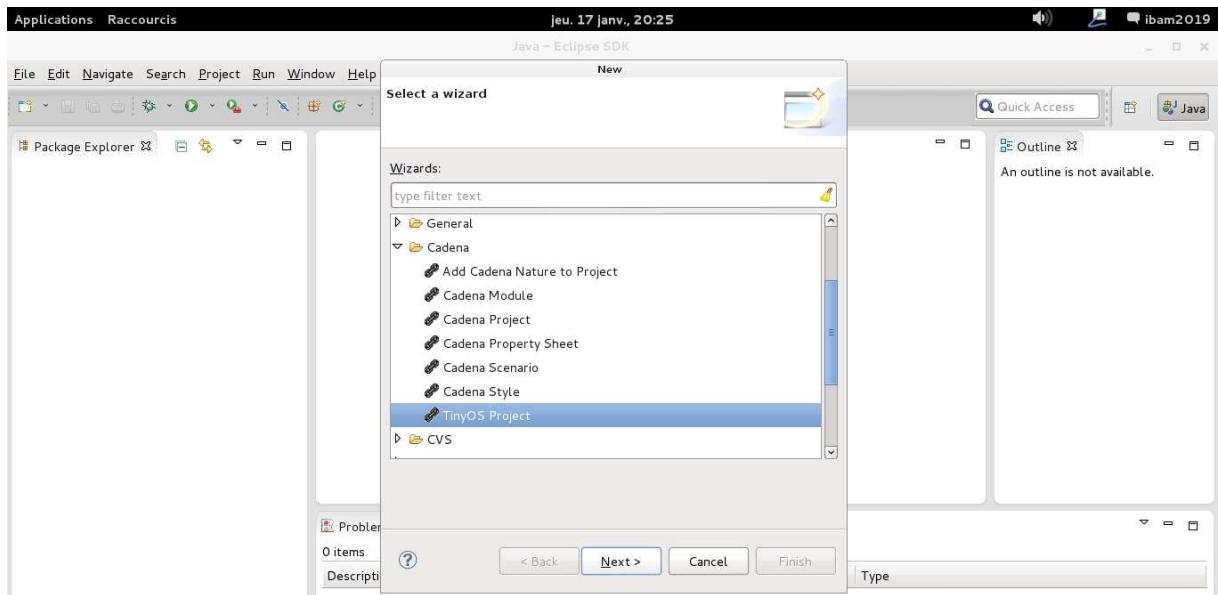


Figure 69 : Choix du langage du projet

Saisissons le nom du projet (« Temperature » dans notre cas) et cliquons sur « Finish »

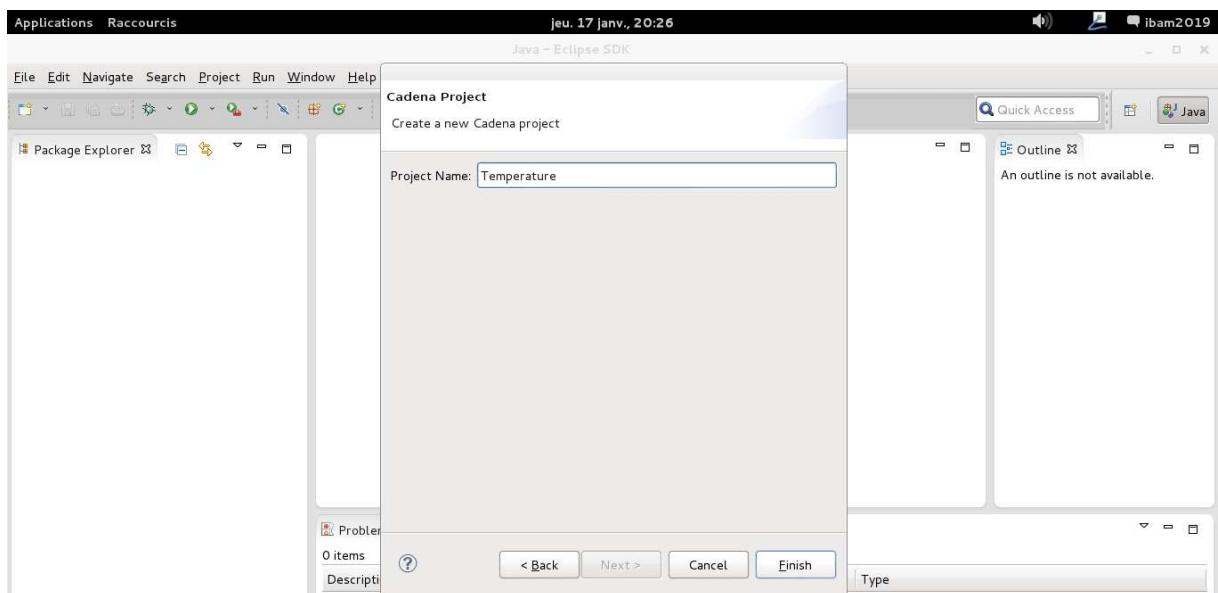


Figure 70 : Saisie du nom du projet

Supprimons le dossier « specification » qui est sous le dossier « Temperature » en faisant : clic droit sur « specification » → Delete



Figure 71 : Suppression du fichier « specification »

Cliquons sur « OK »

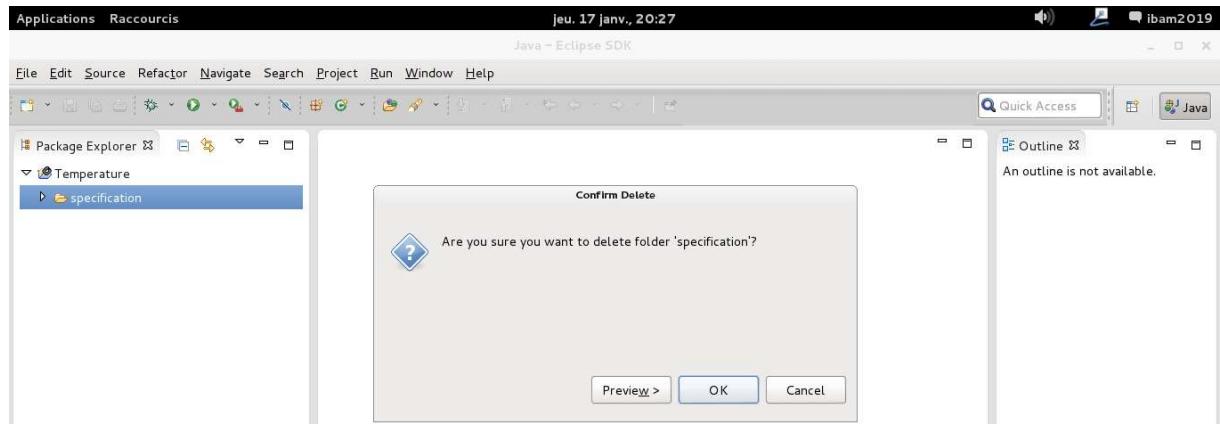


Figure 72 : Confirmation de suppression du fichier « specification »

Créons ensuite un nouveau dossier nommé « src » sous le dossier « Temperature » en faisant : clic droit sur « Temperature » → New → Folder

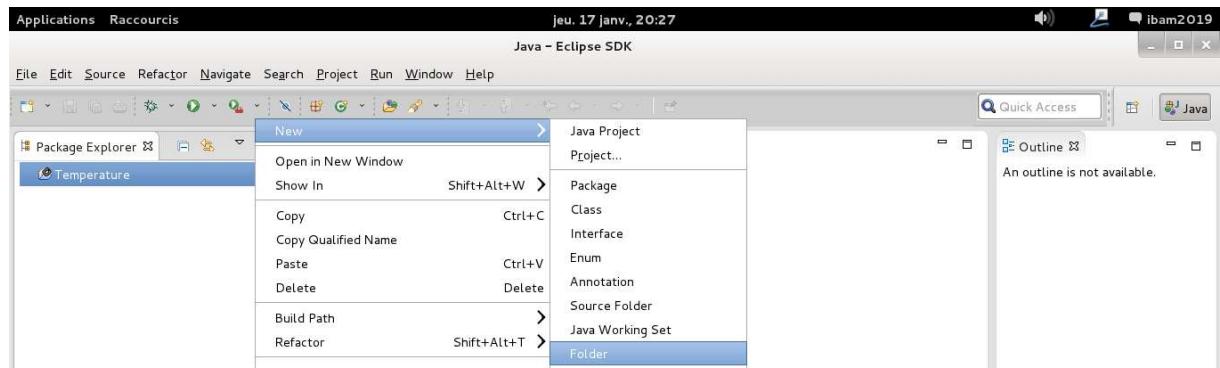


Figure 73 : Création d'un nouveau dossier

Saisissons « src » dans « Folder name » puis cliquons sur « Finish »

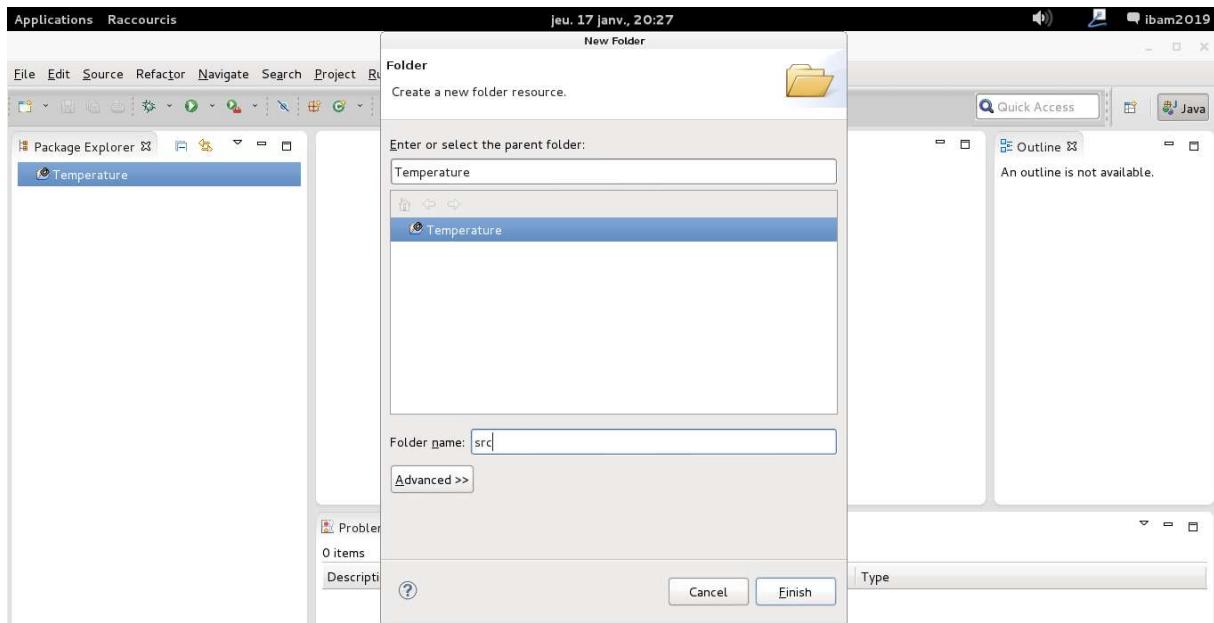


Figure 74 : Saisie du nom du dossier

3.2.1. Composant TemperatureC

Créons un nouveau fichier en faisant : clic droit sur « src » → New → File

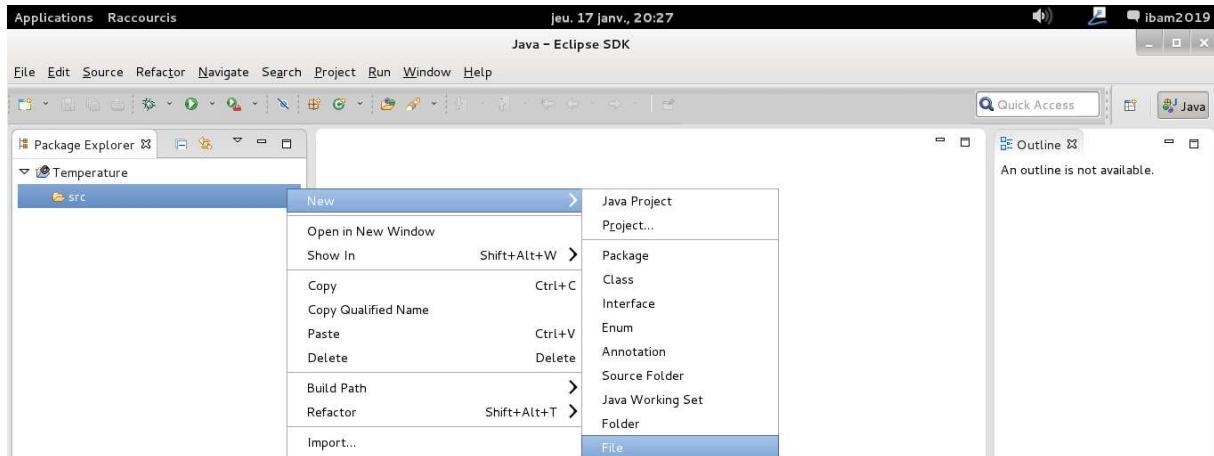


Figure 75 : Crédit d'un nouveau fichier

Saisissons « TemperatureC.nc » dans « File name »

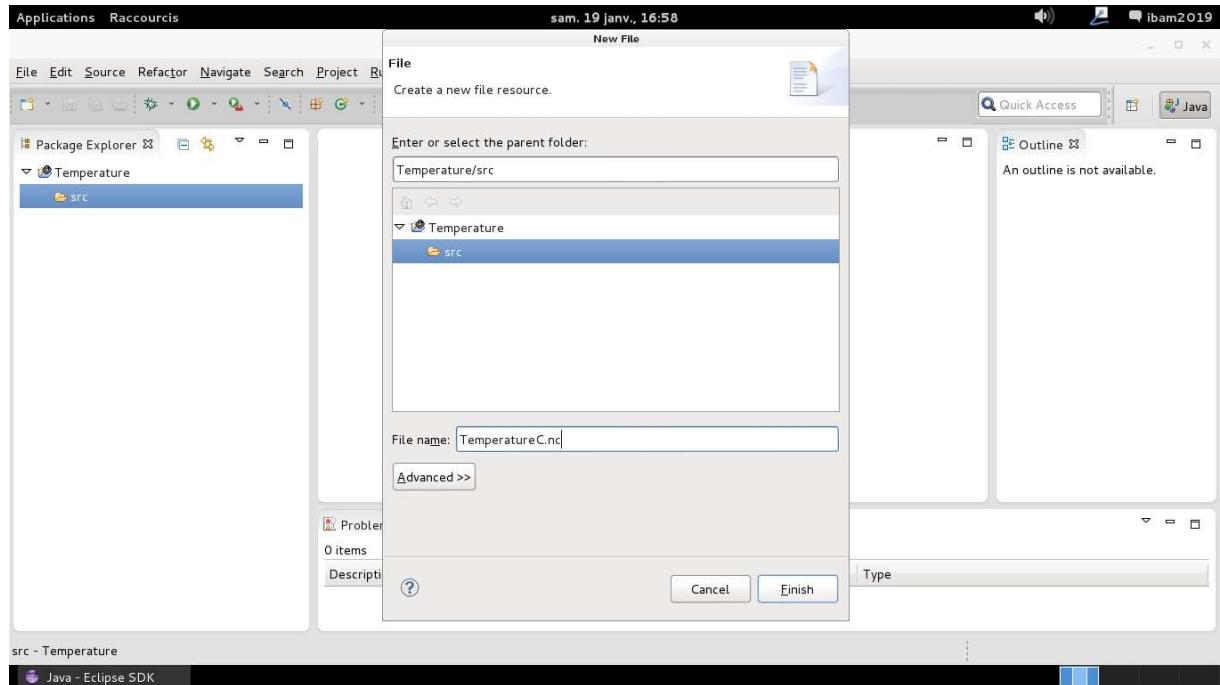


Figure 76 : Saisie du nom du fichier

Faisons clic droit sur « TemperatureC.nc » → Open With → nesC Editor

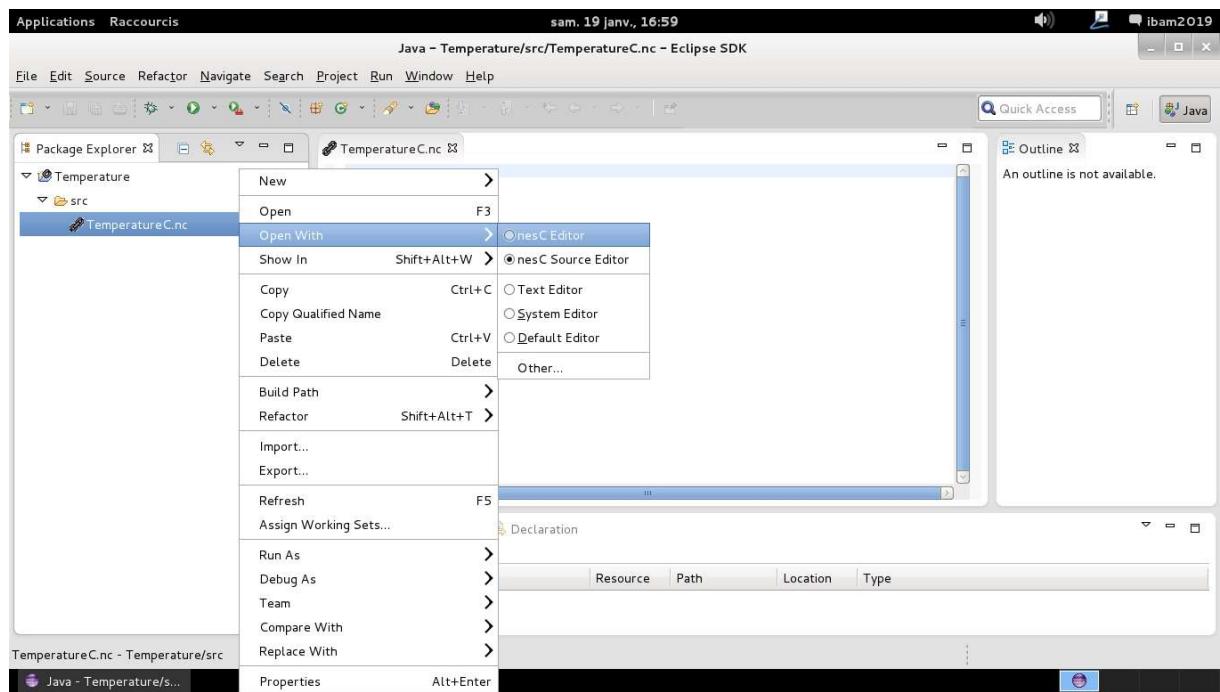


Figure 77 : Chois de l'éditeur de texte

3.2.1.1. Fichier header

Inclusion des fichiers « header » des bibliothèques standards.

#include <Timer.h>

Pour compter le temps ou des événements, les micro-contrôleurs disposent de blocs spécialisés, des timers. Un timer est constitué d'un compteur dont on peut ajuster l'horloge, lire la valeur et écrire la valeur.

#include <stdio.h>

Pour « Standard Input/Output Header » ou « En-tête Standard d'Entrée/Sortie », est l'en-tête de la bibliothèque standard du C déclarant les macros, les constantes et les définitions de fonctions utilisées dans les opérations d'entrée/sortie.

#include <string.h>

Il s'agit de l'en-tête de la bibliothèque standard du C, du langage C, qui contient les définitions des macros, des constantes et les déclarations de fonctions et de types utilisées non seulement pour la manipulation de chaînes de caractères, mais aussi pour diverses fonctions de manipulations de la mémoire.

3.2.1.2. Le module TemperatureC

❖ Spécification du module du module TemperatureC :

interface Boot ;

La séquence de démarrage de TinyOS comprend quatre étapes:

- Initialisation de l'ordonnanceur ;
- Initialisation du composant ;
- Signaler que le processus de démarrage est terminé ;
- *Lancement de l'ordonnanceur.*

interface Timer<TMilli>;

Permet de définir la fréquence de l'horloge en milliseconde.

interface Leds;

Permet d'accéder aux différentes leds qui se trouvent sur le capteur

interface Read<uint16_t> as TempRead;

Permet de lire des données de capteur d'entier non signé de 16 bits. L'interface de lecture a un seul argument, qui définit le type de la valeur de données qu'il produit.

❖ Implémentation du module TempertureC :

```
uint16_t centiGrade ;
```

Création d'une variable de type uint16_t

```
event void Boot.booted()
{
    call Timer.startPeriodic(1000);
    call Leds.led1On();
}
```

TempertureC utilise l'interface de démarrage pour démarrer une horloge périodique après l'initialisation du système, Et affiche la led 1.

```
event void Timer.fired()
{
    if(call TempRead.read() == SUCCESS)
    {
        call Leds.led2Toggle();
    }
    else
    {
        call Leds.led0Toggle();
    }
}
```

A chaque fréquence d'horloge TemperatureC appelle l'interface Read pour lire une valeur. Si on a pu lire la valeur, on affiche la led 2, dans le cas contraire, on affiche la led 0.

```
event void TempRead.readDone(error_t result, uint16_t val)
{
    if(result == SUCCESS)
    {
        centiGrade = (-39.6 + 0.01 * val);

        printf("La temperature est: %d degré celsius \r\n", centiGrade);
    }
    else
    {
        printf("Impossible de lire la température sur le capteur! \r\n");
    }
}
```

Dans le cas où on a pu lire la valeur, on la converti en degré Celsius puis on l'affiche. Dans le cas contraire, on affiche un message d'erreur.

Code complet :

```
// Libairies

#include <Timer.h>
#include <stdio.h>
#include <string.h>

// Modules

module TemperatureC
{
    uses
    {
        // Interfaces générales

        interface Boot;
        interface Timer<TMilli>;
        interface Leds;

        // Lecture

        interface Read<uint16_t> as TempRead;
    }
}

implementation
{
    uint16_t centiGrade;

    event void Boot.booted()
    {
        call Timer.startPeriodic(1000);
        call Leds.led1On();
    }

    event void Timer.fired()
    {
        if(call TempRead.read() == SUCCESS)
        {
            call Leds.led2Toggle();
        }
        else
        {
            call Leds.led0Toggle();
        }
    }

    event void TempRead.readDone(error_t result, uint16_t val)
    {
        if(result == SUCCESS)
```

```

    {
        centiGrade = (-39.6 + 0.01 * val);

        printf("La temperature est: %d degré celsius \r\n",
centiGrade);
    }
else
{
    printf("Impossible de lire la temperature sur le capteur! \r\n");
}
}
}

```

3.2.2. Composant TemperatureAppC

Créons un nouveau fichier en faisant : clic droit sur « src » → New → File

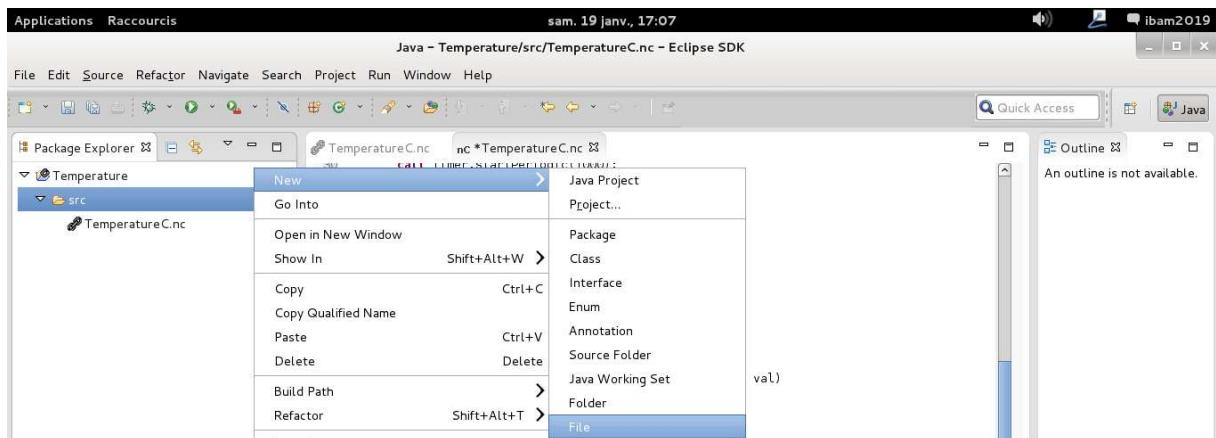


Figure 78 : Création d'un nouveau fichier

Saisissons « TemperatureAppC.nc » dans « File name »

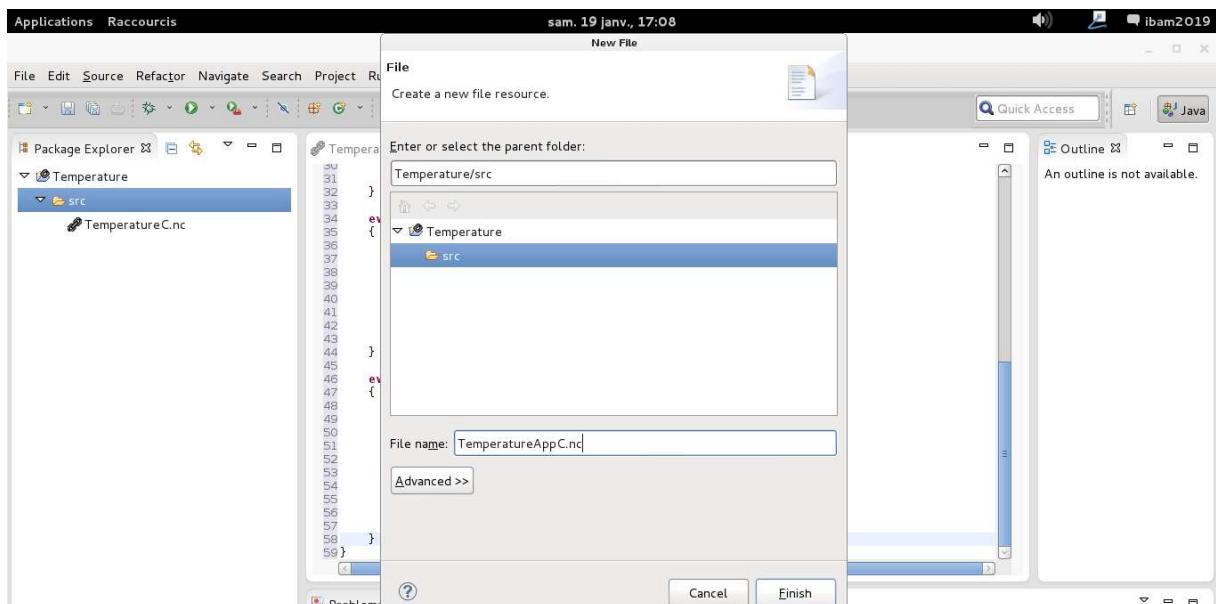


Figure 79 : Saisie du nom du fichier

Faisons clic droit sur « TemperatureAppC.nc » → Open With → nesC Editor

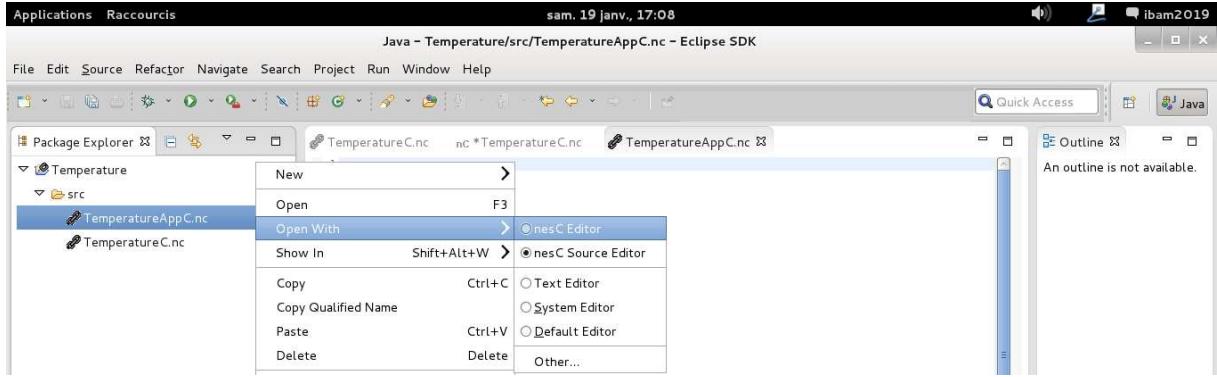


Figure 80 : Choix de l'éditeur de texte

❖ Implémentation de la configuration TemperatureAppC :

components TemperatureC as App;

components MainC, LedsC;

components new TimerMilliC();

Spécification de l'ensemble de composants référencés par la configuration TemperatureAppC.

Le composant MainC fournit le signal Boot.booted qui est essentiellement le point d'entrée de l'application.

Lorsqu'un composant instancie un TimerMilliC, il crée une copie du TimerMilliC.

App.Boot -> MainC;

App.Leds -> LedsC;

App.Timer -> TimerMilliC;

nesC utilise des flèches pour déterminer la liaison entre interfaces. Il lie la flèche vers la droite (->) "est liée à". Le côté gauche de la flèche relie une interface à son implémentation fournie par le côté droit de la flèche. En d'autres mots, le composant qui utilise une interface est du côté gauche de la flèche, et le composant qui fournit une implémentation de l'interface se trouve à droite de la flèche.

components SerialPrintfC;

Ce composant permet d'envoyer la valeur de la température lue vers le port série du pc.

components new SensirionSht11C() as TempSensor;

SensirionSht11C est un composant d'accès de haut niveau pour les capteurs d'humidité et de température « Sensirion » du modèle SHT11, disponible sur la plate-forme telosb. Les appels

simultanés pour lire la température et l'humidité sont arbitrés et exécuté dans un ordre séquentiel.

App.TempRead -> TempSensor.Temperature;

Code complet :

```
configuration TemperatureAppC
{
}

implementation

{
    // Composants générales

    components TemperatureC as App;
    components MainC, LedsC;
    components new TimerMilliC();

    App.Boot -> MainC;
    App.Leds -> LedsC;
    App.Timer -> TimerMilliC;

    // Permet de lire sur le port serie

    components SerialPrintfC;

    // Composant de la température

    components new SensirionSht11C() as TempSensor;

    App.TempRead -> TempSensor.Temperature;
}
```

3.2.3. Makefile

Créons un nouveau fichier en faisant : clic droit sur « src » → New → File

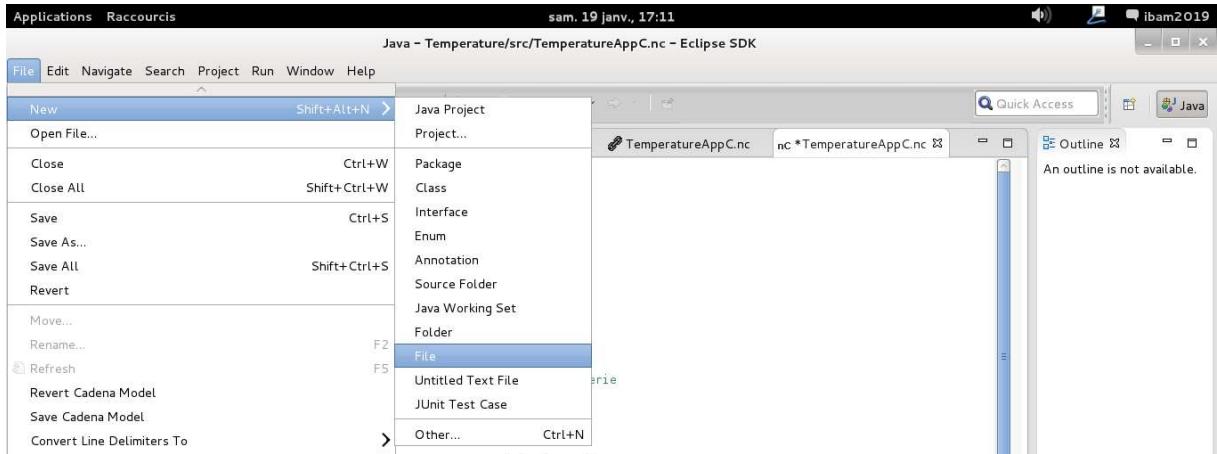


Figure 81 : Crédit d'un nouveau fichier

Saisissons « Makefile » dans « File name »

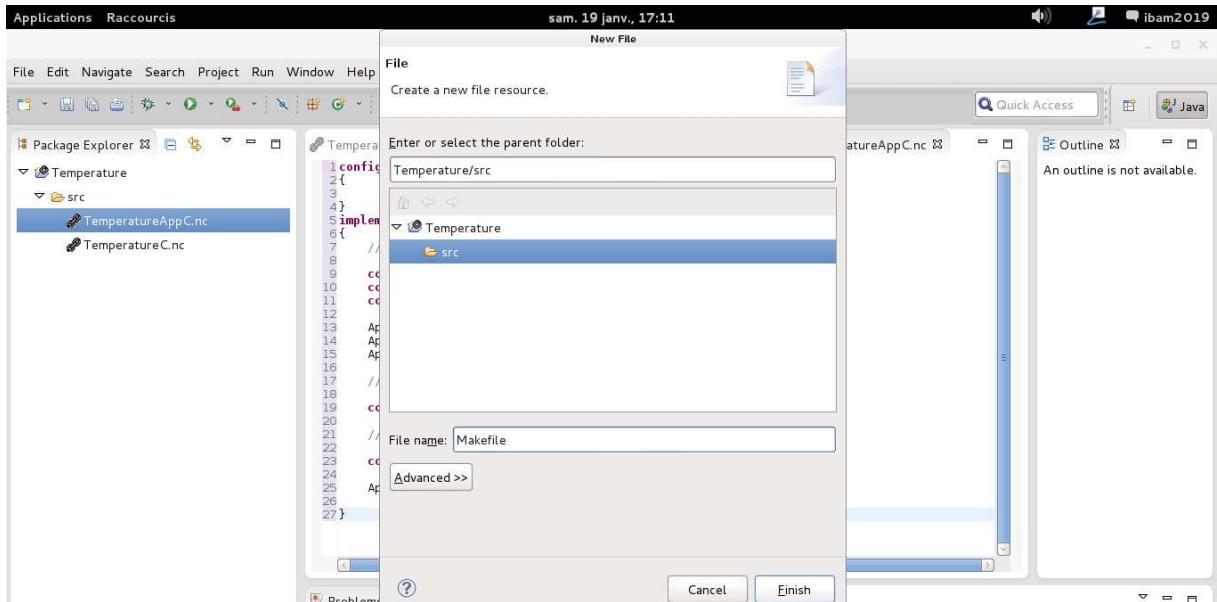


Figure 82 : Saisie du nom du fichier

Nous devons maintenant créer un Makefile afin que le compilateur puisse le compiler le programme. Le fichier makefile est un fichier (nommé par défaut "Makefile") contenant un ensemble de directives utilisées par un outil d'automatisation de compilation pour générer une cible (telobz dans notre cas).

Code complet :

```
COMPONENT = TemperatureAppC
PFLAGS += -I$(TOSDIR)/lib/printf
include $(MAKERULES)
```

Nous pouvons maintenant compiler notre application pour la cible « telosb », en tapant la commande : make telosb



```
Fichier Édition Affichage Rechercher Terminal Aide
setting up TinyOS on source path /opt/tinyosos-main
root@debian:/home/ibam2019# cd Documents/Temperature/Temperature/src/
root@debian:/home/ibam2019/Documents/Temperature/Temperature/src# ls
Makefile TemperatureAppC.nc TemperatureC.nc
root@debian:/home/ibam2019/Documents/Temperature/Temperature/src# make telosb
mkdir -p build/telosb
  compiling TemperatureAppC to a telosb binary
ncc -o build/telosb/main.exe -Os -I/opt/tinyosos-main/tos/lib/printf -fnesc-separator=_ -Wall -Wshadow -Wnesc-all -target=telosb -fnesc-cfile=build/telosb/app.c -boards= -DIDENT_TOS_AM_GROUP=0x22 -DIDENT_APPNAME=\"TemperatureAppC\" -DIDENT_USERNAME=\"root\" -DIDENT_HOSTNAME=\"debian\" -DIDENT_USE_RHASH=0x0dbc5c03L -DIDENT_TIMESTAMP=0x5c4365ael -DIDENT_UIDHASH=0x37938416L TemperatureAppC.nc -lm
./opt/tinyosos-main/tos/chips/sh11/SensirionSh11LogicP.nc: In function 'SensirionSh11LogicP__0__readSensor__runTask':
./opt/tinyosos-main/tos/chips/sh11/SensirionSh11LogicP.nc:322:11: warning: variable 'crc' set but not used [-Wunused-but-set-variable]
./opt/tinyosos-main/tos/chips/sh11/SensirionSh11LogicP.nc: In function 'SensirionSh11LogicP__0__performCommand':
./opt/tinyosos-main/tos/chips/sh11/SensirionSh11LogicP.nc:193:19: warning: variable 'crc' set but not used [-Wunused-but-set-variable]
  compiled TemperatureAppC to build/telosb/main.exe
    10534 bytes in ROM
      124 bytes in RAM
msp430-objcopy --output-target=ihex build/telosb/main.exe build/telosb/main.ihex
  writing TOS image
```

Figure 83 : Compilation de l'application « Temperature »

Changeons le nom du fichier main.exe qui se trouve dans « /home/ibam2019/Documents/Tempereture/Temperature/src/build/telosb/ » en temperature-demo.firmware



```
root@debian:/home/ibam2019/Documents/Temperature/Temperature/src# cd build/telosb/
root@debian:/home/ibam2019/Documents/Temperature/Temperature/src/build/telosb# ls
app.c ident_flags.txt main.exe main.ihex tos_image.xml
root@debian:/home/ibam2019/Documents/Temperature/Temperature/src/build/telosb# mv main.exe temperature-demo.firmware
root@debian:/home/ibam2019/Documents/Temperature/Temperature/src/build/telosb# ls
app.c ident_flags.txt main.ihex temperature-demo.firmware tos_image.xml
root@debian:/home/ibam2019/Documents/Temperature/Temperature/src/build/telosb# cp temperature-demo.firmware /opt/mspsim/firmware/esb/
```

Figure 84 : Changement du nom du fichier main.exe en temperature-demo.firmware

Déplaçons le fichier temperature-demo.firmware qui se trouve dans « /home/ibam2019/Documents/Tempereture/Temperature/src/build/telosb/ » vers « /opt/mspsim/firmware/esb/ »



```
root@debian:/home/ibam2019/Documents/Temperature/Temperature/src/build/telosb# cp temperature-demo.firmware /opt/mspsim/firmware/esb/
```

Figure 85 : Déplacement de temperature-demo.firmware dans /opt/mspsim/firmware/esb/

Ouvrons le fichier « Makefile » qui se trouve dans « /opt/msp-sim »



Figure 86 : Ouverture du fichier « Makefile »

Au niveau de « ESBFIRMWARE » nous devons avoir ceci : « ESBFIRMWARE = firmware/esb/temperature-demo.firmware »

```

#####
# SERVER OBJECTS
#####

ifndef FIRMWAREFILE
ESBFIRMWARE = firmware/esb/temperature-demo.firmware
SKYFIRMWARE = firmware/sky/blink.firmware
else
ESBFIRMWARE = ${FIRMWAREFILE}
SKYFIRMWARE = ${FIRMWAREFILE}
endif

CPUTEST := tests/cputest.firmware

SCRIPTS := ${addprefix scripts/,autorun.sc duty.sc}
BINARIES := README.txt license.txt CHANGE_LOG.txt images/*.jpg firmware/*/*.firmware ${SCRIPTS}

PACKAGES := se/sics/mspsim ${addprefix se/sics/mspsim/,core platform platform/esb platform/sky cli ui util chip net extutil/highlight extutil/jfrechart}

SOURCES := ${wildcard *.java ${addsuffix /*.java,$(PACKAGES)}}

OBJECTS := $(SOURCES:.java=.class)

JARFILE := mspsim.jar

#####
# MAKE
#####

```

Figure 87 : Modification du fichier « Makefile »

Pour lancer notre simulation, tapons : « make runesb ».

```
root@debian:/opt/mspsim# make runesb
```

Figure 88 : Lancement de l'émulateur mspSim

Nous devrions avoir ce qui suit :

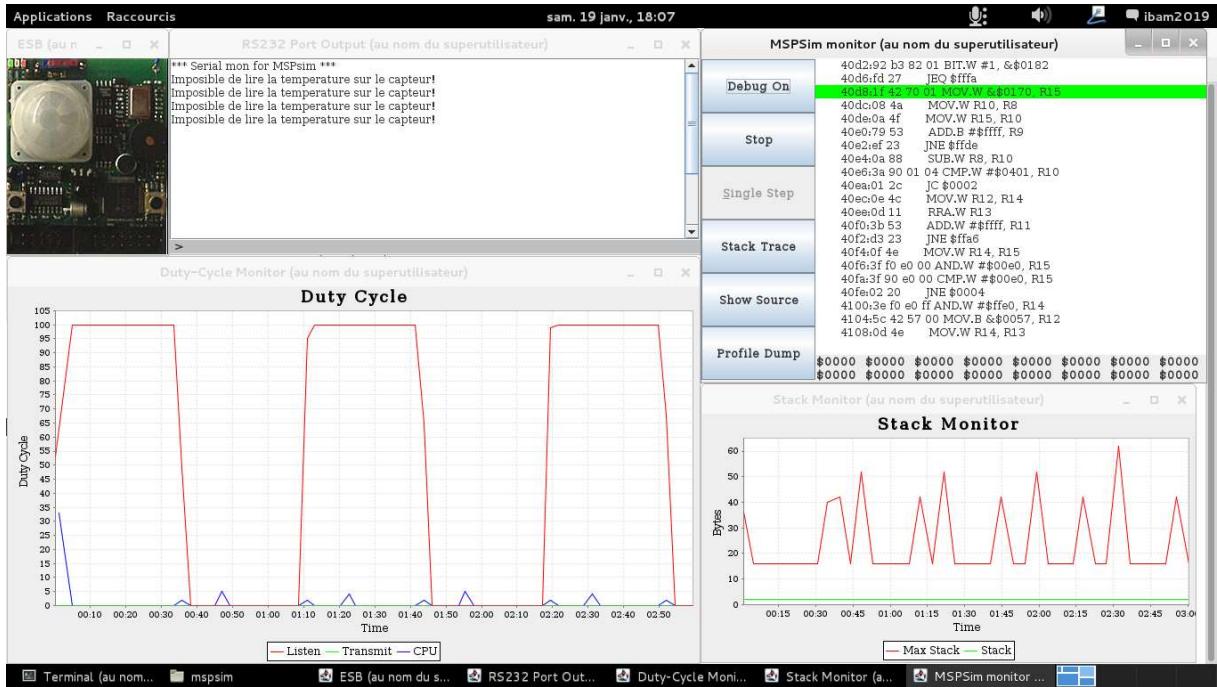


Figure 89 : Application « Temperature » dans mspSim

Nous pouvons voir dans le terminal « Impossible de lire la température sur le capteur ! ». Cela est normal d'autant plus que le simulateur ne nous permet pas de simuler un capteur de température. Néanmoins cela nous a permis de constater que notre code n'a pas générer d'erreur lors de la compilation. Aussi nous constatons que notre code est compatible à la cible telosb.

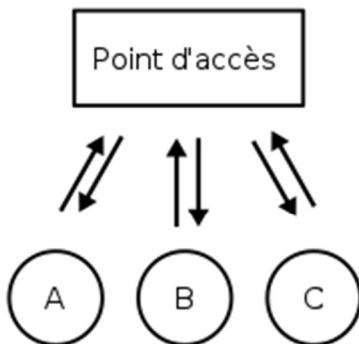
Annexes

Annexe 1

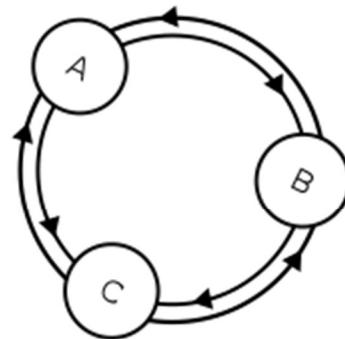
Les réseaux ad hoc sont des réseaux sans fil capables de s'organiser sans infrastructure définie préalablement. Par exemple d'un équipement à un autre sans infrastructure (point d'accès).

Chaque entité (node ou nœud) communique directement avec sa voisine. Pour communiquer avec d'autres entités, il lui est nécessaire de faire passer ses données par d'autres qui se chargeront de les acheminer. Pour cela, il est d'abord primordial que les entités se situent les unes par rapport aux autres, et soient capables de construire des routes entre elles : c'est le rôle du protocole de routage.

Ainsi, le fonctionnement d'un réseau ad-hoc le différencie notamment d'un réseau comme le réseau GSM ou des réseaux Wi-Fi avec des points d'accès : là où une ou plusieurs stations de base sont nécessaires à la plupart des communications entre les différents nœuds du réseau (mode Infrastructure), les réseaux ad-hoc s'organisent d'eux-mêmes et chaque entité peut jouer différents rôles.



Echanges en mode infrastructure



Echanges en mode ad-hoc

Source 9: <https://upload.wikimedia.org/wikipedia/commons/thumb/3/34/Ad-Hoc.svg/542px-Ad-Hoc.svg.png>

Figure 90: Réseau mode infrastructure et mode ad-hoc

Annexe 2

Un système embarqué est défini comme un système électronique et informatique autonome, souvent temps réel, spécialisé dans une tâche bien précise. Le terme désigne aussi bien le matériel informatique que le logiciel utilisé. Ses ressources sont généralement limitées. Cette limitation est généralement d'ordre spatial (encombrement réduit) et énergétique (consommation restreinte). Les systèmes embarqués utilisent généralement des

microprocesseurs à basse consommation d'énergie ou des microcontrôleurs, dont la partie logicielle est en partie ou entièrement programmée dans le matériel, généralement en mémoire dans une mémoire morte (ROM), EPROM, EEPROM, FLASH, etc. (on parle alors de firmware). Plutôt que des systèmes universels effectuant plusieurs tâches, les systèmes embarqués sont étudiés pour effectuer des tâches précises. Certains doivent répondre à des contraintes de temps réel pour des raisons de fiabilité et de rentabilité. D'autres ayant peu de contraintes au niveau performances permettent de simplifier le système et de réduire les coûts de fabrication.

- Les systèmes embarqués ne sont pas toujours des modules indépendants. Le plus souvent ils sont intégrés dans le dispositif qu'ils contrôlent.
- Le logiciel créé pour les systèmes embarqués est appelé firmware. Il est stocké dans de la mémoire en lecture seule ou de la mémoire flash plutôt que dans un disque dur. Il fonctionne le plus souvent avec des ressources matérielles limitées : écran et clavier de tailles réduites, voire absent, peu de mémoire, capacités de calcul relativement faibles.

Annexe 3

L'architecture distribuée ou l'informatique distribuée désigne un système d'information ou un réseau pour lequel l'ensemble des ressources disponibles ne se trouvent pas au même endroit ou sur la même machine. Ce concept, dont une version peut être une combinaison de transmissions du type client-serveur, s'oppose à celui d'informatique centralisée .

Internet est un exemple de réseau distribué puisqu'il ne possède aucun nœud central. Les architectures distribuées reposent sur la possibilité d'utiliser des objets qui s'exécutent sur des machines réparties sur le réseau et communiquent par messages au travers du réseau.

Annexe 4

uIP (micro IP) est une implémentation open source de la pile TCP/IP pour les microcontrôleurs 8bit et 16bit utilisés par exemple dans les réseaux de capteurs sans fil ou dans d'autres systèmes communicants embarqués miniatures. Initialement créé par Adam Dunkels du groupe "Networked Embedded Systems" du "Swedish Institute of Computer Science", uIP est disponible sous une licence type BSD, et maintenu par un groupe large de développeurs.

uIP est utile dans les systèmes embarqués car son code et son utilisation de la RAM sont minimes. Il a été porté sur plusieurs plateformes incluant des DSP.

En octobre 2008, Cisco, Atmel et SICS ont annoncé une extension de uIP conforme à IPv6, appelée uIPV6.

Webographies

- https://moodle.utc.fr/file.php/498/SupportWeb/co/Module_RCSF_11.html
- https://fr.wikipedia.org/wiki/R%C3%A9seau_de_capteurs_sans_fil
- https://fr.wikipedia.org/wiki/R%C3%A9seau_ad_hoc
- https://fr.wikipedia.org/wiki/R%C3%A9seau_ad_hoc#Description_d%C3%A9taill%C3%A9e
- https://fr.wikipedia.org/wiki/R%C3%A9seau_de_capteurs_sans_fil#Principe_de_fonctionnement
- https://fr.wikipedia.org/wiki/R%C3%A9seau_de_capteurs_sans_fil#Architectures_R%C3%A9seau_de_capteurs
- https://moodle.utc.fr/file.php/498/SupportWeb/co/Module_RCSF_14.html
- https://moodle.utc.fr/file.php/498/SupportWeb/co/Module_RCSF_15.html
- https://moodle.utc.fr/file.php/498/SupportWeb/co/Module_RCSF_48.html
- https://moodle.utc.fr/file.php/498/SupportWeb/co/Module_RCSF_59.html
- https://fr.wikipedia.org/wiki/Système_d'exploitation_pour_capteur_en_réseau
- https://fr.wikipedia.org/wiki/Système_embarqué%C3%A9
- [e](https://fr.wikipedia.org/wiki/Système_embarqué%C3%A9#Architectur)
- https://fr.wikipedia.org/wiki/Système_embarqué%C3%A9#Caract%C3%A9ristiques
- https://fr.wikipedia.org/wiki/Architecture_distribu%C3%A9e
- [https://fr.wikipedia.org/wiki/UIP_\(micro_IP\)](https://fr.wikipedia.org/wiki/UIP_(micro_IP))
- https://fr.wikipedia.org/wiki/Système_d'exploitation_pour_capteur_en_réseau#Réseau_B4le_du_système
- <https://fr.wikipedia.org/wiki/TinyOS>
- [https://fr.wikipedia.org/wiki/TinyOS#Propriét%C3%A9s_principales](https://fr.wikipedia.org/wiki/TinyOS#Propriét%C3%A9t%C3%A9s_principales)
- https://fr.wikipedia.org/wiki/TinyOS#Allocation_de_la_m%C3%A9moire
- https://fr.wikipedia.org/wiki/TinyOS#Structure_logicielle
- https://fr.wikipedia.org/wiki/TinyOS#L%E2%80%99ordonnanceur_TinyOS
- https://fr.wikipedia.org/wiki/TinyOS#Cibles_posibles_pour_TinyOS
- <https://air.imag.fr/index.php/TinyOS>
- https://moodle.utc.fr/file.php/498/SupportWeb/co/Module_RCSF_38.html

- https://www.advanticsys.com/wiki/index.php?title=Getting_started_with_nesC#Introduction_to_Components
- <http://tinyprod.net/repos/debian/>
- http://tinyos.stanford.edu/tinyos-wiki/index.php/Automatic_installation
- <https://www.itzgeek.com/how-tos/linux/debian/how-to-install-eclipse-ide-on-debian-9-ubuntu-16-04-linuxmint.html>
- <http://cadena.projects.cs.ksu.edu/update/>
- <https://sourceforge.net/projects/jtossim/files/jtossim-0.1.1.zip/download>
- <https://air.imag.fr/index.php/TinyOS#JTOSSIM>
- <http://tinyos.stanford.edu/tinyos-wiki/index.php/MSPSim>
- <http://tinyos.stanford.edu/tinyos-wiki/index.php/MSP430>
- <http://tinyos.stanford.edu/tinyos-wiki/index.php/TelosB>
- <http://tinyos.stanford.edu/tinyos-wiki/index.php/MSPSim#Installation>
- http://tinyos.stanford.edu/tinyos-wiki/index.php/MSPSim#Blink_application
- http://public.iutenligne.net/etudes-et-realisations/guinand/uControleur/utilisation_dun_timer.html
- <https://fr.wikipedia.org/wiki/Stdio.h>
- <https://fr.wikipedia.org/wiki/String.h>
- http://tinyos.stanford.edu/tinyos-wiki/index.php/Sensing#The_Sense_application
- <https://en.wikipedia.org/wiki/Makefile>
- http://tinyos.stanford.edu/tinyos-wiki/index.php/The_simplest_TinyOS_program

Bibliographies

- Cadena 2.0: Install Guide: A Guide to Installing Cadena and Platform Plugins, Todd Wallentine, 28 pages
- Algorithmes de diffusion dans les réseaux dynamiques de capteurs sans fil, Tarek Moulahi, 142 pages
- Cadena 2.0: nesC Tutorial: A guide to using Cadena to develop nesC/TinyOS applications, Todd Wallentine, 76 pages
- Les Système Embarqué TinyOS, MEDJHOUM Khaled, 18 pages
- lab2_TinyOS_nesC, 46 pages
- tinyos-tutorial-wsn, 28 pages
- TinyOS Programming, Philip Levis, June 28, 2006, 139 pages

Table des matières

Sommaire	ii
Table des figures.....	iii
Introduction	1
1. Etudes préalables.....	2
1.1. Réseau de Capteurs Sans Fil	2
1.1.1. Définition.....	2
1.1.2. Constituant d'un capteur.....	2
1.1.3. Principe de fonctionnement	2
1.1.4. Architecture d'un RCSF	3
1.1.4.1. La topologie en étoile.....	3
1.1.4.2. La topologie point par point	3
1.1.4.3. La topologie maillé	4
1.1.4.4. L'architecture de groupe	4
1.1.4.5. La topologie hybride.....	4
1.1.5. Médias de transmission.....	4
1.1.6. Communication dans les RCSF.....	5
1.1.6.1. Modèle en couches	5
1.1.6.2. Rôle des 5 couches	6
1.1.6.3. Plans de gestion.....	6
1.1.7. Protocoles de routage dans les RCSF	7
1.1.7.1. Exemple de protocole de routage : SPIN.....	7
1.1.8. Système d'exploitation pour RCSF	8
1.2. TinyOS.....	9
1.2.1. Présentation	9
1.2.2. Allocation de la mémoire	10
1.2.3. Structure logicielle.....	10
1.2.3.1. Composants	10
1.2.3.1.1. Tâche.....	11
1.2.3.1.2. Evènements	11
1.2.3.2. Type de composant	11
1.2.4. L'ordonnanceur TinyOS.....	11
1.2.5. Equipement supportant TinyOS.....	12
1.3. Langage de programmation nesC.....	12
1.3.1. Compilation.....	12

1.3.2.	Concepts de nesC	13
1.3.3.	Composant	14
1.3.4.	Interface	15
2.	Installation des outils	17
2.1.	Prérequis	17
2.2.	Installation de tinyos	17
2.3.	Configuration de l'environnement de développement	29
2.4.	Simulateur TOSSIM	36
2.4.1.	Présentation de TOSSIM	36
2.4.2.	Installation de JTOSSIM	36
2.5.	Emulateur MSPSim	38
2.5.1.	Présentation de MSPSim	38
2.5.2.	Installation de MSPSim	39
3.	Cas pratique	42
3.1.	Aperçu	42
3.2.	Application « Temperature »	42
3.2.1.	Composant TemperatureC	46
3.2.1.1.	Fichier header	48
3.2.1.2.	Le module TemperatureC	48
3.2.2.	Composant TemperatureAppC	51
3.2.3.	Makefile	54
Annexes		58
Annexe 1		58
Annexe 2		58
Annexe 3		59
Annexe 4		59
Webographies		iv
Bibliographies		vi
Table des matières		vii