UNIVERSITÉ DE SHERBROOKE
Faculté de génie
Département de génie électrique et de génie informatique

MODÈLE FONCTIONNEL D'UN SOL INTELLIGENT

Mémoire de maîtrise
Spécialité : génie électrique

Benoit WEYL

Jury : Philippe MABILLEAU (directeur)
Bessam ABDULRAZAK (codirecteur)
Ahmed KHOUMSI (rapporteur)
Wael SULEIMAN

Sherbrooke (Québec) Canada Janvier 2013

Library and Archives Canada

Published Heritage Branch

395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliothèque et Archives Canada

Direction du Patrimoine de l'édition

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence
ISBN: 978-0-494-93309-1

Our file Notre référence
ISBN: 978-0-494-93309-1

RÉSUMÉ

L'assistance aux personnes ayant des troubles cognitifs, au laboratoire DOMUS, se fait selon deux approches. La première met en œuvre des outils d'assistance personnelle portés par la personne, comme le *panic button* ou encore une application déployée sur un téléphone intelligent. La seconde consiste à équiper l'environnement de la personne afin de lui proposer une aide discrète qui ne modifie que très peu ses habitudes de vie. Ce présent projet de recherche s'inscrit dans la seconde approche d'assistance.

Ce mémoire a pour but de proposer un prototype de sol intelligent permettant la localisation et même l'identification des personnes présentes dans un espace intelligent. Les sols intelligents jusqu'alors développés présentent tous la même particularité : ils nécessitent, pour les déployer et les exploiter, des connaissances avancées dans les domaines de l'informatique et de l'électronique ainsi que la mise en place d'un processus complexe pour leur déploiement et leur exploitation. L'architecture proposée pour ce prototype de sol intelligent vise à faciliter au maximum sa configuration et son utilisation afin de réduire au maximum les coûts qui y sont liés.

Pour faciliter le développement des algorithmes d'auto-configuration et d'exploitation des nœuds constituant le sol intelligent, un simulateur a été réalisé. Il permet de confirmer le comportement de ces algorithmes dans un réseau de grande dimension sans avoir à mettre en place une réalisation matérielle qui représente une étape complexe et coûteuse. La mise en œuvre, d'un point de vue matériel, au niveau matériel n'a été réalisée que sur un nombre limité de nœuds afin d'en démontrer la faisabilité.

Mots-clés : Informatique diffuse, Domotique, Sol intelligent, Réseau de capteurs, Transmission dans un réseau, Auto-configuration, cartographie d'un réseau

TABLE DES MATIÈRES

LISTE DES FIGURES

LISTE DES TABLEAUX

LISTE DES ACRONYMES

Acronyme	Définition
ADNOS	Algorithmic Device Network Organization System
ACK	Acknowledge
ARCHIPEL	ARCHItecture de gestion de contexte Pour améliorer l'autonomie fonctionnELle
ASN.1	Abstract Syntax Notation One
CEA	Consumer Electronics Association
CHSLD	Centre d'Hébergement de Soins de Longue Durée
CRC	Cyclic Redundancy Check
CSMA/CA	Carrier Sense Multiple Access with Collision Avoidance
CSMA/CD	Carrier Sense Multiple Access with Collision Detection
CS/$\overline{SS}$	Chips Select / Slave Select
DOMUS	laboratoire en DOmotique et informatique Mobile de l'Université de Sherbrooke
FSR	Force Sensing Resistor
GPS	Global Positioning System
GRF	Ground Reaction Force : Force de réaction du sol
GSM	Global system for Mobile
I2C	Inter Integrated Circuit
LLC	Logical Link Control
MAC	Media Access Control
MCMC	Markov Chain Monte Carlo
MEMS	Micro ElectroMechanical Systems
MISO	Master Input, Slave Output
MOSI	Master Output, Slave Input
MVC	Modèle Vue Contrôleur
NAK	Negative AcKnowledge
OK	Opportunity Knocks
OSI	Open Systems Interconnection
PWM	Pulse Width Modulation
RFID	Radio Frequency IDentification
RSSI	Received Signal Strength Indication
SCL, SCLK	Serial Clock
SDA	Serial DAta ligne
SPI	Serial Peripheral Interface
TLV	Tag Length Value
UART	Universal Asynchronous Receiver Transmitter
USCI	Universal Serial Communication Interface

CHAPITRE 1

INTRODUCTION

1.1 Contexte

Aujourd'hui, le mot « intelligent » se rattache à une multitude de systèmes, tels que les téléphones ou les voitures, etc. Le laboratoire DOMUS, laboratoire en **DO**motique et informatique Mobile de l'Université de Sherbrooke, travaille sur un espace intelligent : la maison, ou l'appartement. L'idée de maison intelligente nous renvoie souvent à la science fiction ou au luxe. Toutefois, nous pensons plus rarement à son utilisation en tant que support permettant d'aider les personnes présentant des troubles cognitifs. C'est dans ce contexte que s'inscrivent les projets du DOMUS.

1.1.1 Le laboratoire DOMUS

Le laboratoire DOMUS a été créé en 2002 et se situe au sous-sol du pavillon Marie - VICTORIN dans le bâtiment D7 de l'Université de Sherbrooke. Aujourd'hui, il comprend un appartement entièrement aménagé, Figure 1.1. Cet aménagement permet aux expériences menées au sein du laboratoire de prendre une dimension réaliste et aux chercheurs de penser aussi bien la faisabilité que l'intégration de technologies d'assistance sans imposer de contraintes à qui que ce soit. Il est aussi intéressant de souligner que le DOMUS est un laboratoire interdisciplinaire qui regroupe un ensemble de personnes, non pas d'après leurs compétences individuelles, mais plutôt autour d'un intérêt commun : l'assistance aux personnes souffrant de troubles cognitifs. Toutefois, le DOMUS est compétent dans divers domaines : l'électronique, l'informatique, la physiologie, l'ergonomie, etc. Par ailleurs, pour faire le lien entre le monde de la recherche et le monde actuel, le travail du laboratoire est soutenu par différents partenaires : des organismes de santé. Par exemple le DOMUS est impliqué dans un projet de résidence alternative. Ce faisant, il existe un lien réel entre les recherches développées dans le laboratoire et le monde extérieur, mais aussi entre les chercheurs, le personnel soignant et les proches du patient.

D'après Bussière *et al.* : « d'ici 2041, le Québec sera l'une des sociétés les plus vieilles en occident » et « les chercheurs estiment que près de 26 000 ménages à travers la province ont des besoins non comblés »[4]. Donner aux personnes présentant des troubles cognitifs

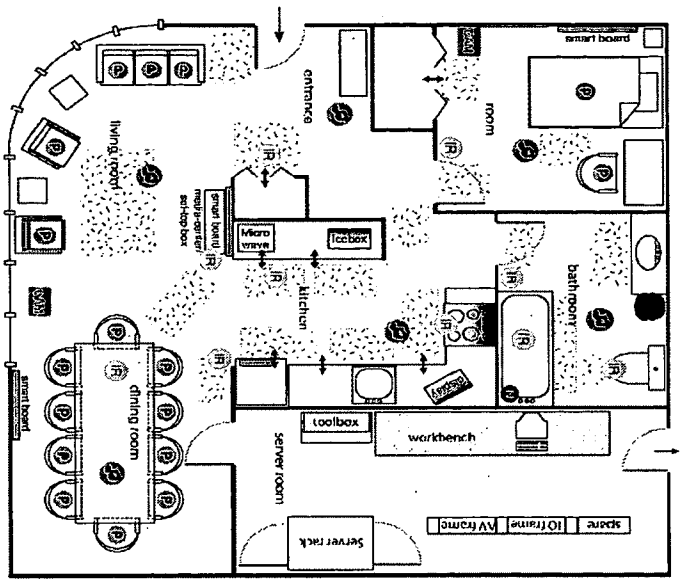


Figure 1.1 Appartement du laboratoire DOMUS

- comme la maladie d'Alzheimer, la démence ou encore les traumatismes crâniens - une plus grande autonomie tout en leur procurant les soins et l'attention nécessaires, est donc tout à fait d'actualité. Cet apport d'autonomie a pour objectif de soulager le personnel soignant, ainsi que les proches, et de diminuer la demande de placement en CHSLD (**C**entre d'**H**ébergement de **S**oin de **L**ongue **D**urée) ce qui est en accord avec un autre objectif de recherche du laboratoire : celui de réduire le sentiment de dépendance de ces personnes en leur assurant une vie décente en dehors des maisons de soin. Rappelons que cette demande d'autonomie croît chaque jour, en même temps que les *baby-boomers* vieillissent, et n'oublions pas que la population âgée de 65 ans et plus doublera quasiment d'ici 2041, Tableau 1.1 [14].

Tableau 1.1 Comparatif entre la population âgée de 65 ans et plus entre 2012 et 2041 au Québec.

	En 2012	En 2041
Hommes âgés de 65 ans et plus	574 105	1 127 011
Population masculine totale	3 973 464	4 504 669
% d'hommes âgés de 65 ans et plus	14,5%	25%
Femmes âgées de 65 ans et plus	726 797	1 283 045
Population féminine totale	4 035 563	4 544 522
% de femmes âgées de 65 ans et plus	18%	28,2%
Population de 65 ans et plus	1 300 902	2 410 056
Population totale	8 009 027	9 049 191
% de population de 65 ans et plus	16,2%	26,6%

1.1.2 La localisation

De nos jours, la localisation fait partie des grandes préoccupations de la vie quotidienne. L'étude faite en 2007 par la CEA (*Consumer Electronics Association*) montre que 55% des hommes et 47% des femmes possèdent un GPS. Ce système de localisation permet à une personne de connaître sa position sur la planète, mais aussi de faire de la planification de trajectoire ou encore de se procurer une liste de services proches d'une position, etc. Malheureusement, l'utilisation d'un GPS présente des limites. Du fait de la forte atténuation des signaux GPS à travers les matériaux de construction, un GPS ne fonctionne pas à l'intérieur des bâtiments. De plus, lorsque la personne sort de ces bâtiments, le GPS a besoin d'un délai d'environ une minute avant de se relocaliser [5]. Par ailleurs, la localisation par GPS dépend aussi de l'environnement extérieur : un GPS a plus de difficultés à se localiser dans une grande ville, du fait de l'obstruction faite par les grands bâtiments, que dans un village, [22].

Le désir de connaître la position ne se limite pas au seul cas d'une personne ne sachant pas le trajet pour se rendre dans un endroit inconnu : il s'étend aussi aux systèmes de surveillances. Dans certains contextes, il est nécessaire de savoir où se trouve une tierce personne. Par exemple, pour la surveillance d'un prisonnier assigné à résidence [5], afin de soulager le système carcéral, mais aussi pour la surveillance des personnes présentant une quelconque déficience, dans le but de soulager, cette fois-ci, les centres de santé. Cette localisation de surveillance agit directement sur l'autonomie de la personne. Dans le premier exemple, l'autonomie est diminuée tandis que dans le second, elle est accrue. Cette autonomie peut être augmentée encore davantage en proposant un élargissement de la localisation aux objets. En effet, le fait de ne plus savoir où se situent les objets importants, tels que les clefs ou encore la télécommande de la télélvision, est perturbant pour une personne souffrant de troubles cognitifs. Pour sa part, le laboratoire DOMUS cherche activement les meilleures solutions afin d'accroître l'autonomie des individus.

1.2 Problématique

Un des moyens qu'utilise le laboratoire DOMUS pour redonner de l'autonomie aux personnes souffrant de troubles cognitifs consiste à doter l'environnement de capteurs. L'appartement situé dans le laboratoire possède des capteurs de mouvements, d'ouverture de porte, de consommation électrique, mais aussi des tapis tactiles et des microphones. Ils permettent de détecter l'activité, la présence ou le déplacement d'une personne. Il est donc possible d'analyser le comportement d'une personne et d'identifier les situations anormales

et/ou dangereuses qu'elle pourrait rencontrer. Beaucoup de ces capteurs sont non intrusifs et diffus. Ils ne nécessitent pas d'attention particulière.

Cependant, l'utilisation de tous ces capteurs engendre trois grands problèmes :

P.1 : D'installation, même si, aujourd'hui, la tendance en informatique est au système décentralisé et diffus, la plupart des systèmes informatiques comprennent encore une topologie en étoile ou centralisée. Un exemple de sol intelligent centralisé est donné par Kaddoura *et al.* [19]. Son installation nécessite 72 heures de travail pour une surface de $32m^2$. Ce temps est en partie du au *mapping* des capteurs à l'unité de calcul. Il en serait de même au laboratoire DOMUS où il faudrait relier tous nouveaux capteurs à l'automate programmable.

P.2 : De maintenance, lorsqu'une défaillance intervient dans un réseau, il faut d'abord en trouver la source qui peut être humaine ou matérielle. Les sols intelligents ne sont pas à l'abri des dégâts des eaux et de l'usure normale des composants électroniques. La localisation d'une défaillance dans un sol intelligent et sa prise en compte pour assurer une continuité de fonctionnement consituent une problématique importante. L'algorithme de type ADNOS (***A**lgorithmic **D**evice **N**etwork **O**rganization **S**ystem*), conçu par Glaser *et al.* permet de réagir aussi bien à la défectuosité d'un composant électronique qu'à une nouvelle connexion, mais aussi de s'adapter à la forme du réseau [10]. De cette manière, la maintenance est facilitée et l'échéance pour changer un composant qui tombe en panne peut être repoussée.

P.3 : D'extensibilité, au laboratoire DOMUS, pour ajouter de nouveaux capteurs dans l'environnement, il faut en premier lieu que l'automate programmable possède encore des entrées libres. Ensuite, il faut installer et identifier un fil entre l'automate et chacun des capteurs. Une fois que les nouveaux capteurs sont installés, il faut encore programmer l'automate pour qu'il prenne en compte et interprète ces nouvelles sources de données. Idéalement, il serait mieux de connecter le nouveau capteur au système et que ce dernier s'adapte de lui-même aux modifications.

Parmi les capteurs non intrusifs, les tapis tactiles présentent une utilisation plus restrictive de l'environnement. Effectivement, il faut entrer la position des capteurs dans le système de localisation et fixer ces mêmes capteurs au sol pour ne pas devoir le refaire à chaque déplacement par inadvertance. Il faut également y connecter des câbles pour l'alimentation électrique et la liaison avec le système. De fait, un tapis tactile ne peut se trouver au milieu d'une pièce, ni dans la salle de bain ou la cuisine, sans induire des risques de courts-circuits

dus aux câblages et de détérioration plus rapide. Qui plus est, la figure 1.2 montre qu'un tapis tactile ne répond pas forcément aux critères esthétiques attendus dans un logement.

Les tapis tactiles sont répartis aux endroits stratégiques de l'appartement du laboratoire. Ces endroits peuvent être des lieux de passages fréquents (pas de porte) ou encore être des lieux à risques, devant une cuisinière par exemple. La disposition de ces tapis tactiles ne permet donc pas de retracer la totalité du trajet d'une personne. N'oublions pas non plus qu'un tapis tactile ne comporte qu'une seule zone d'activation qui est en général de grande taille, environ $1m^2$. L'interprétation des données est donc restreinte à la présence ou non de quelque chose, cela peut être une personne tout comme un objet.

La question de recherche qui se pose est donc :

Comment faciliter la localisation des personnes dans un espace intelligent ?

1.3 Objectifs

La possibilité de suivre les mouvements d'une personne, sans pour autant lui donner la sensation d'être épiée, peut être fournie par le sol sur lequel elle évolue. Effectivement, installer un système de capteurs de présence diffus dans le sol réduit considérablement le côté intrusif d'un système de surveillance. Par conséquent, les contraintes d'utilisation sont elles aussi réduites, comme oublier de porter le capteur ou de recharger ses batteries, et son fonctionnement s'en trouve facilité. Par ailleurs, lorsqu'une personne fait un mouvement, il y a obligatoirement un changement dans l'appui au sol. Grâce au sol intelligent qui peut détecter la présence d'une personne, suivre son évolution dans l'habitat et identifier les situations dangereuses ou nécessitant une intervention auprès d'autres systèmes intelligents, il n'y a plus aucune intrusion dans la vie des utilisateurs.

Pour que ce projet de sol intelligent puisse voir le jour dans un grand nombre de logements, il faut que les coûts de conception, d'installation, d'utilisation et d'entretien soient abordables. Pour cela, le sol doit être conçu de sorte que son déploiement ne comporte pas de manipulations spécifiques aux connaissances des électriciens et informaticiens. Il doit pouvoir résister à tous les produits qu'il est possible de trouver dans un logement afin d'être installé dans toutes les pièces et qu'aucun entretien spécifique ne lui soit apporté.

L'objectif général de ce projet est donc le suivant :

Concevoir un premier modèle fonctionnel de sol intelligent

Les points suivants le composent :

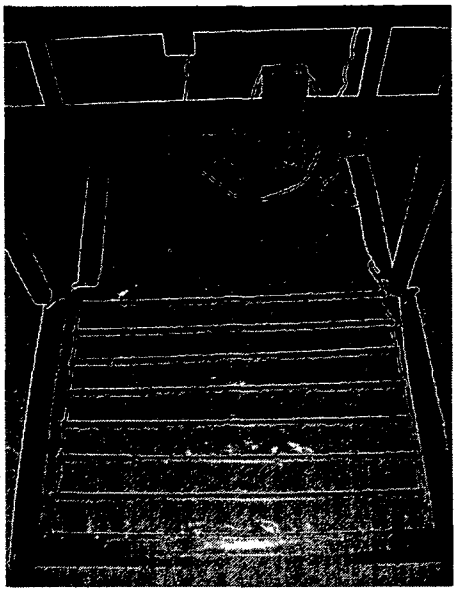


Figure 1.2 Tapis tactile présent dans l'appartement du laboratoire DOMUS.

Proposer des services/fonctionnalités et les étudier : le choix des services/fonctionnalités que pourra proposer le sol intelligent est exhaustif. Il est possible d'établir une première liste en se basant sur des études comparatives et des propositions d'implémentation.

Choisir l'architecture du réseau : le choix de cette architecture sera basé sur une étude des différentes topologies de réseaux maillés existant ainsi que de leur méthode de communication.

Choisir un protocole de communication : ce protocole de communication devra pouvoir acheminer l'information extraite des capteurs vers une centrale de calcul. Il devra pouvoir faire face aux pannes de certains nœuds.

Implémenter un algorithme d'auto-configuration : cet algorithme permettra au réseau de se construire et de se maintenir lui même.

1.4 Plan du document

Après ce chapitre introductif, c'est une revue de la littérature qui va être exposée dans le chapitre 2. Dans un premier temps, nous étudierons les domaines d'application des sols intelligents. Ensuite, nous nous concentrerons sur les différentes approches qui permettent la localisation dans un espace intelligent. Enfin, nous nous attarderons sur les étapes nécessaires à la conception d'un sol intelligent.

Le chapitre 3 contient les différentes étapes ayant servi à la conception d'un premier prototype fonctionnel de réseau, comprenant un ensemble de capteurs pouvant communiquer entre eux. La conception commence avec la présentation de différentes topologies de réseau à la section 3.1. Ces topologies de réseau sont basées soit sur des connexions point à point permettant aux nœuds, élements du réseau, de communiquer directement avec leurs voisins, soit sur l'utilisation de bus à travers desquels les nœuds peuvent joindre un nombre indéfini de nœuds. S'enchaîne ensuite la section 3.2 comprenant les définitions des trois couches matérielles du modèle OSI constituant un nœud. Dès lors, il est possible de détailler les phases de fonctionnement d'un nœud au cours des différents états du réseau, section 3.3. À ce moment, il est possible d'implémenter les couches du modèle OSI dans un micro-contrôleur et de concevoir un simulateur de réseau afin de valider le comportement des algorithmes, section 3.4.

Enfin, le chapitre 4 sera une synthèse dans laquelle nous rappelerons les contributions et nous proposerons des idées pour des travaux ultérieurs.

CHAPITRE 2

REVUE DE LA LITTÉRATURE

Pour bien comprendre l'enjeu de la géolocalisation, nous présenterons dans la section 2.1 de ce chapitre des applications concernant des activités de la vie quotidienne. Les données qui concernent la localisation sont issues de solutions technologiques variées et ce sont des équipes de chercheurs, qui se distinguent tout autant dans le milieu académique qu'industriel, qui les ont développées. Dans la section 2.2, nous présenterons un ensemble de ces solutions. Enfin, dans la section 2.3, nous présenterons les trois étapes qui permettent l'élaboration d'un système de localisation.

2.1 Domaines d'utilisation

Le fait de localiser une personne dans un environnement intelligent permet un meilleur support cognitif. Ici, sont exposées quelques applications mettant en rapport la volonté de rendre de l'autonomie aux personnes souffrant de déficiences cognitives avec la problématique de localisation. Dans un premier temps, nous aborderons la question des déplacements extérieurs : les limites du système GPS sont mis en évidence et une solution hybride GPS/GSM/802.11 ainsi qu'un GPS focalisé sur la personne plutôt que sur le véhicule sont présentés, §2.1.1. Dans un deuxième temps, nous examinerons la détection des situations à risques, telles que les chutes par exemple : des capteurs d'ambiance permettent la détection, §2.1.2. Enfin, dans un dernier temps, nous étudierons la question de la localisation au profit de l'analyse des activités de la vie quotidienne : une intelligence artificielle permet de faire le lien entre l'activation des capteurs et les activités réalisées, §2.1.3.

2.1.1 Aide aux déplacements

Pour pouvoir aider une personne souffrant de troubles cognitifs dans ses déplacements, il faut nécessairement connaître sa localisation. Bien que le GPS couvre la majorité de la surface terrestre avec une précision d'environ 10m, il ne peut suivre un homme une journée durant. En effet, Lamarca *et al.* montre que pour trois personnes différentes — un immunologiste, une femme au foyer et un caissier — la couverture temporelle du GPS, pendant une journée de travail, est en moyenne de 4,5% [23]. Nous en déduisons que le GPS n'est pas l'unique solution en matière de localisation. Le laboratoire *Place Lab* se

penche sur ce problème d'une autre façon. Il considère que la couverture temporelle est plus importante que la précision. En fait, ses chercheurs ont mis au point un système de localisation basé sur les balises radios présentes dans l'environnement et la connaissance de leur emplacement. Il fonctionne grâce aux points d'accès 802.11 et aux tours GSM qui envoient des messages périodiques qui contiennent leur identifiant unique (une adresse MAC). Les coordonnées de ces balises sont connues grâce aux bases de données constituées par les *War-Driver* qui ont recensé la position et l'identifiant de 2.2 millions de balises en 2005. C'est par la triangulation que le système en question calcule la position de l'individu. Les essais réalisés dans trois régions différentes : en zone urbaine, résidentielle et en banlieue, donnent une couverture temporelle de 100% et une précision qui oscille entre 13m et 30m. Ces résultats sont fonction du nombre de balises radios présentes.

Aujourd'hui, la tendance en urbanisme est de prendre en compte l'intégration des personnes souffrant d'un handicap physique ou cognitif. Toutefois, la réhabilitation des villes ne peut se faire dans leur intégralité. C'est pourquoi, contrairement au système du laboratoire *Place Lab*, plusieurs autres systèmes permettent aux personnes souffrant de troubles cognitifs d'utiliser plus facilement l'infrastructure déjà en place. D'aucuns pensent que les transports en commun ne sont pas toujours adaptés pour les handicapés physiques, mais c'est également le cas pour les handicapés cognitifs. En effet, l'utilisation des transports en commun induit la possibilité de se tromper de direction. Ils sont donc intimidants et stressants [32]. Par exemple " *Opportunity Knocks* " est un système dédié aux personnes présentant des troubles cognitifs qui leur permet de se diriger dans plusieurs endroits familiers [32]. Ce système fait de la localisation par GPS. Il utilise l'écran d'un téléphone portable, avec lequel il communique par *bluetooth*, pour afficher l'interface homme-machine. Pour qu'*Opportunity Knocks*, OK, soit opérationnel, l'utilisateur n'a pas besoin d'entrer une quelconque information. Il lui suffit de porter OK pendant ses déplacements. De cette manière, OK mémorise tout seul les trajets usuels et entraine sa base de données. En phase d'utilisation, lorsque la personne est immobile pendant un certain temps, OK se manifeste et propose les endroits où la personne a l'habitude d'aller depuis cette localisation. La personne choisit la destination et suit les indications d'OK. Ainsi, OK détecte lorsqu'un mauvais trajet est emprunté. Lorsqu'aucune destination n'est sélectionnée, OK retourne en phase d'entrainement. Comme il s'agit d'un système focalisé sur la personne et non sur le moyen de transport, OK peut guider l'utilisateur aussi bien à pied, que lorsqu'elle se déplace en bus.

2.1.2 Détection des chutes

La section précédente montre quelques utilisations de la localisation extérieure, aller d'un endroit à un autre en fait partie. Les problématiques associées à la localisation intérieure sont toutes autres. En effet, guider une personne dans une maison avec un appareil portatif n'est pas envisageable car cela impose de fortes contraintes d'utilisation pour un objectif discutable. C'est pourquoi il existe plusieurs axes de recherche concernant le problème de la localisation intérieure. Un de ces axes consiste à détecter les situations dangereuses.

La situation qui suscite le plus d'hospitalisations au Canada est la chute. En effet, pour la tranche d'âge des 65 ans et plus, 85% des hospitalisations s'effectuent suite à une chute [35]. Pour les détecter, plusieurs approches sont envisageables. Soit elles utilisent des capteurs embarqués qui interprètent les mouvements de la personne qui les porte, soit elles utilisent des capteurs d'ambiance qui analysent les interactions avec l'environnement ou soit elles utilisent des caméras d'ambiance qui détectent à la fois les formes et les mouvements [45]. Seuls les capteurs d'ambiance répondent aux contraintes fixées par le laboratoire DOMUS parce qu'ils ne sont pas intrusifs.

Lorsqu'une personne se déplace, elle provoque des vibrations qui se propagent à travers le sol. Ces vibrations peuvent être analysées et identifiées. Alwan *et al.* a choisi de monter un capteur piézoélectrique dans une boîte posée à même le sol [2]. Ce capteur peut détecter 100% des chutes avec un indice de confiance de 95%. De plus, il peut différencier la chute d'un objet de masse allant jusqu'à 6,5kg d'un corps dans un rayon de 4,5m au rez-de-chaussée et de 6m dans un étage. Cependant, il ne faut pas poser le capteur contre un mur, car les données de la pièce voisine pourraient interférer. Werner *et al.* détecte lui aussi les chutes grâce aux vibrations captées par des accéléromètres situés aux pieds des murs d'une pièce [42] . Le projet *eHome* qui a duré 507 jours a montré que le système détecte 87% des chutes. Ces deux études prouvent qu'il est possible de différencier l'activité et les chutes dans une pièce en se basant sur la propagation des vibrations.

Une autre méthode pour détecter les chutes consiste à analyser les pressions exercées sur le sol. Les entreprises *Futur Shape* et VIGI$^{\text{metric}}$ commercialisent respectivement les sols *Sensfloor* [8] et TAPIS$^{\text{metric}}$ [13], composés de capteurs de pression. Dès lors il est possible de suivre les déplacements d'une personne en suivant les empreintes de pression exercées par les pieds. Lorsque la personne chute, il y a une différence de forme dans la pression appliquée. Rappelons que la pression d'une empreinte de pas est ponctuelle tandis que lors d'une chute, cette même pression est répartie sur la surface de contact entre la personne et le sol.

2.1.3 Analyse des activités de la vie quotidienne

Détecter les situations dangereuses, une fois qu'elles sont apparues, est important pour leur apporter une réponse rapide. En ce qui concerne les chutes, il est difficile de les anticiper et de réagir pour minimiser leurs conséquences. Cependant, il est possible de prévenir d'autres situations à risques.

La prévention commence par l'analyse du comportement des personnes souffrant de troubles cognitifs : elle permet d'identifier et de distinguer un certain nombre de leurs actions. Jalal *et al.* identifie dix activités de la vie quotidienne — comme par exemple faire le ménage, la cuisine, marcher, prendre un objet, etc. — il filme les individus et analyse leurs silhouettes sur lesquelles il s'appuie pour reconnaître les activités en cours [17]. L'identification se fait par des modèles de Markov cachés, avec un taux de réussite de 96.55%. Cependant, la présence de caméras inclut un plus grand sentiment de surveillance que celui recherché et s'éloigne du projet initial d'assistance du laboratoire DOMUS. Aujourd'hui, l'utilisation de caméras dans un habitat intelligent est très controversée. En effet, pour certains, les caméras portent atteinte à la vie privée, tandis que pour d'autres elles sont un inconvénient nécessaire pour garder une certaine autonomie.

Par exemple, lors de la préparation des repas, le laboratoire DOMUS utilise ARCHIPEL qui possède deux fonctions [3]. La première est de guider la personne au cours de la confection des recettes pour le repas. La deuxième est de surveiller les actions de l'individu afin d'identifier celles dont la réalisation suscite des difficultés. Cette surveillance se fait lorsque la personne interagit avec l'habitat intelligent, soit lors d'ouvertures/fermetures des portes, d'enclenchement d'interrupteur, de passages sur un tapis tactile, soit encore en inter-agissant avec l'interface homme-machine [9]. Finalement, lorsque la personne n'interagit pas avec l'habitat, la localisation se fait de manière générale grâce aux capteurs infrarouges. Il n'est donc pas possible de connaître à tout moment la localisation de la personne. Par ailleurs, ARCHIPEL fonctionne uniquement avec une seule personne.

Un autre système d'assistance dans l'utilisation de la cuisine a été développé par Wai *et al.* [41]. Tout comme avec ARCHIPEL, la cuisine est munie de capteurs, d'interrupteurs *reed*, de tags RFID, etc. L'identification des différentes activités se fait par l'intermédiaire de la logique floue, à partir d'inférences recueillies depuis les données captées. De cette manière, les situations dangereuses peuvent être détectées.

Les travaux ultérieurs qui seront menés sur ces deux systèmes d'assistance [3, 41] auront pour objectif d'améliorer et/ou d'ajouter un système de localisation intérieure qui offrira

un meilleur suivi des actions entreprises, ce qui devrait également permettre de faire de ces systèmes des systèmes multi-utilisateurs.

2.2 Différentes approches de localisation dans un espace intelligent

Pour déterminer la position d'une personne dans un espace intelligent, plusieurs solutions peuvent être envisagées. Une première consiste à équiper la personne observée de capteurs ou de marqueurs, §2.2.1. Ainsi, l'espace intelligent localise les éléments portés par la personne. Une autre consiste à introduire des capteurs dans l'espace : ceci permet une analyse directe de l'évolution d'une personne., §2.2.2.

2.2.1 Localisation des capteurs/marqueurs

Pour faire une localisation de personne à partir de capteurs ou de marqueurs, il faut que la personne à localiser les emmène dans chacun de ses déplacements. Dès lors qu'une personne porte un capteur/marqueur sur elle, il est plus facile de l'identifier parmi d'autres dans le même espace. En effet, la plupart des technologies pensées sur ce principe utilisent des moyens de communication sans-fil basés sur l'emploi de tags RFID. Parmi celles-ci, il est possible de mettre à jour deux utilisations différentes. La première consiste à équiper la personne d'un tag RFID, dont les lecteurs sont répartis dans l'espace intelligent, tandis que pour la seconde, la disposition des tags et des lecteurs est inversée. Dans ce second cas, la personne est munie d'un lecteur qui lit les tags disséminés dans l'espace intelligent.

Pour illustrer la première utilisation, Yun *et al.* équipe les travailleurs d'une entreprise métallurgique de tags RFID afin de superviser les évacuations lors d'alertes incendie [46]. L'espace possède un réseau *Zigbee* de nœuds lecteurs sans-fil. Les tags RFID transmettent un message deux fois par seconde, de sorte que les nœuds lecteurs analysent la force du signal capté, soit le RSSI (***Received Signal Strength Indication***), avec un filtre à particule à double couche, et triangulent la position du tag dans l'espace. Lors des essais, ce système a montré qu'il était possible de suivre 100 personnes dans un espace de 100m x 50m en temps réel, avec une erreur maximale de 3m.

Le second cas peut être illustré par l'étude de Jung *et al.* qui inverse la répartition des tags et des lecteurs [18]. On introduit ces derniers dans des pantoufles et/ou dans des gants que la personne peut porter. Ils permettent de lire les tags dispersés dans l'espace intelligent. Le sol contient une grille de tags espacés de 30cm. Les lectures effectuées par

la grille de tags en corrélation avec le plan de l'espace intelligent permettent de localiser la personne avec une précision de 30cm. Ainsi, il est possible de retracer ses déplacements. Par ailleurs, la conception de gants reposant sur le même principe que les pantoufles permet de superviser les actions entreprises par l'intermédiaire d'objets tagués. Le traitement de l'information issue des lecteurs RFID est transmis à un serveur via une liaison *Zigbee*. De ce fait, le serveur peut fonctionner avec plusieurs personnes munies de pantoufles et de gants lecteurs dans le même espace. En ce qui concerne l'autonomie de la batterie, ce système de lecteur de tags et de transmetteur sans-fil consomme 40mA par tag scanné et information transmise et 10mA en mode veille. Ce système peut donc fonctionner pendant au moins une journée avec une batterie de 1 000mA/h.

2.2.2 Analyse de l'évolution de la personne

Bien que la localisation par capteurs/marqueurs montre de bons résultats quant à la précision et à l'utilisation par de multiples utilisateurs, elle apporte aussi des contraintes concernant l'utilisation de matériels embarqués. Effectivement, pour obtenir de bons résultats, il faut que tous les utilisateurs se prennent au jeu : tous doivent faire attention à l'état des capteurs/marqueurs, à la charge de la batterie et à la manière de bien porter l'équipement. Pour éviter ces contraintes, il est possible d'intégrer dans l'espace intelligent des capteurs mesurant l'état de cet espace.

Identification par le profil de pression

Les sols intelligents ont connu plusieurs évolutions. Les premiers concepts consistent en la réalisation d'un sol composé de dalles qui reposent sur des capteurs de type cellules de charge. La répartition de ces capteurs sous les dalles peut se faire de différentes manières. Le système de dalles, dans l'étude de Addlesee *et al.*, est constitué de 9 dalles de 50cm par 50cm et les mesures se font avec des cellules de charge de type jauge de contraintes [1]. Chaque cellule de charge supporte le coin de quatre dalles. Une autre possibilité de répartition des capteurs peut être illustrée par le système de l'étude d'Orr et Abowd constitué d'une simple dalle qui repose, en ses quatre coins, sur une cellule de charge [31].

Le fait que le nombre de dalles change dans les deux solutions permet tout de même de faire une comparaison. En effet, dans le premier système, les mesures et les traitements se réalisent en déconnectant les dalles extérieures. Ainsi, seule la dalle centrale est utilisée. Pour effectuer l'identification des empreintes de pas, les deux études se basent sur le profil de la force de réaction du sol GRF (***Ground Reaction Force***). Cependant, chacune met en oeuvre un algorithme différent. Dans le système de Addlesee *et al.*., l'identification se fait

par l'intermédiaire du modèle caché de Markov et permet d'obtenir un taux de succès de 92%. L'étude de Orr et Abowd, quant à elle, procède à une identification par la méthode des 1-ppv (1 **P**lus **P**roche **V**oisin) à partir des points particuliers suivants : la moyenne, l'intégrale, la longueur, les deux maximums et le minimum local du profil GRF. Cette méthode atteint un taux de succès de 93% et il est indépendant de la paire de chaussures portées.

Ces deux systèmes présentent de bons taux d'identification. Cependant, les tests ont été effectués dans des conditions particulières : empreintes de pas au centre des dalles. De plus, pour le système d'Addlesee *et al.*, les 8 dalles voisines ont été déconnectées. Lors des tests avec les 9 dalles connectées et une utilisation normale, le taux d'identification chute à 50%. Quant à l'étude d'Orr et Abowd, le cas d'identification d'une empreinte de pas non centrée n'a pas été testé.

La réalisation de ces deux systèmes montre que l'utilisation du profil GRF pour l'identification de l'empreinte de pas est judicieuse et que plusieurs algorithmes d'identification peuvent être envisagés, notamment par la méthode des frontières ou par la mise en place d'un réseau de neuronnes. Par ailleurs, le choix de la disposition des capteurs dans le sol est important : il faut envisager une autre méthode que celle d'Addlesee *et al.*, et confirmer la possible utilisation de celle d'Orr et Abowd dans un système multi-dalles. Il faut aussi envisager le cas de présences multiples sur le sol intelligent et son impact sur l'identification.

Localisation avec un système multi-dalles

Après avoir étudié les systèmes de sols intelligents à simple dalle, il nous parait intéressant de nous pencher sur une technologie permettant un déploiement total dans une pièce. Pour cela, il faut envisager le système comme un réseau de capteurs, tel que l'a fait Kaddoura *et al.* [19]. Le système qui a été mis en place se compose de dalles qui reposent en leur centre sur un capteur de type FSR (***F**orce **S**ensing **R**esistor*). Chaque capteur est connecté à un nœud *Phidgets* [11]. Un nœud accepte un maximum de 8 capteurs et est connecté directement à un ordinateur. Ce système de sol est donc un réseau de capteurs dont la topologie est en étoile. Il a été testé dans la maison intelligente : *Gator Tech Smart House* [12], et a obtenu un taux de détection allant jusqu'à 87%, avec une couverture de 100% de la surface d'une pièce. Pourquoi s'agit-il d'un taux de détection et non d'identification comme pour les autres systèmes ? Parce que Kaddoura *et al.* a eu pour objectif l'industrialisation du système et donc un besoin de réduire les coûts. Cela a eu pour effet de limiter le nombre de capteurs utilisés et de rendre impossible une quelconque identification.

Le fait d'avoir testé le système dans des conditions réelles, c'est-à-dire dans une vraie maison, a mis en évidence le problème de connexion des capteurs, bien que l'utilisation de la technologie Atlas [21] *plug and play* ait simplifié l'interface physique du système et n'ait pas fait chuter le réseau lors de la défaillance d'un nœud, problème P.2. En effet, dans la phase de test, le sol a été déployé sur une surface de 350 pieds carrés, à raison de 1 capteur par 1 pied carré, il a donc fallu les connecter tous aux bons nœuds et ces derniers aux bons endroits, problème P.1. Par ailleurs, ce système nécessite un sol surélevé pour pouvoir déployer tous les câbles de manière sécuritaire.

Comme le sol intelligent de la *Gator Tech Smart House* est installé dans une maison intelligente, d'autres systèmes de détection/identification peuvent se greffer au sol, problème P.3. Il est d'ailleurs projeté d'inclure des caméras pour analyser la posture, l'orientation et même faire l'identification des habitants de la maison, et ce de manière plus efficace. On considère, en effet, que la position extraite du sol intelligent peut réduire la zone d'images dans laquelle il est nécessaire de faire un traitement et donc de réduire le nombre de calculs nécessaires. Une autre étude montre que le couplage entre un sol intelligent et des caméras permet de rendre le système multi-utilisateurs [44]. Dans une pièce, quatre caméras sont réparties de manière à ne pas avoir d'angle mort et des capteurs de pression sont installés dans le sol. Le calibrage des positions entre le sol et les caméras respecte cependant la méthode basée sur la force brute, qui essaie toutes les configurations possibles jusqu'à trouver la bonne. Les deux sources de données agissent alors de manière complémentaire. Lorsqu'une caméra a des difficultés à analyser les données, notamment lors d'un problème d'ombre, les capteurs de pression rendent la détection plus facile. Mais lorsque deux personnes se situent dans la même zone, les capteurs de pression ne permettent pas de différencier la présence des deux personnes. Les caméras, elles, rendent possible cette distinction.

Les systèmes industrialisés

Plusieurs entreprises ont industrialisé des sols intelligents qui présentent des cibles et applications différentes. *Vstone* est une entreprise japonaise qui propose le sol *"Sensafuroashisutemu"* [40]. Ce sol a été conçu pour observer le déplacement de l'homme dans son environnement. Il se compose de capteurs binaires qui détectent la présence d'une charge dès 200 - 250 $g.cm^{-2}$. Il permet d'étudier le comportement humain mais aussi de faire un suivi de sécurité dans les logements ou l'industrie. Toutefois, ce système nécessite des conditions particulières d'utilisation : il ne doit pas rentrer en contact avec l'eau, il est sensible à la poussière, à l'exposition prolongée au soleil et aux ondes électromagnétiques, problème P.2.

Ces sensibilités réduisent ses capacités à rendre des services en tout temps. C'est pourquoi des systèmes comme le TAPIS^{metric}[13] ou le SensFloor [8] ont directement été conçus pour être robustes face aux conditions d'utilisation domestiques ou industrielles, problème P.2. TAPIS^{metric} est similaire à du Linoleum et est constitué de deux feuilles métalisées et d'une sous-couche de mousse trouée, située entre les deux feuilles métalisées, qui joue le rôle de capteur. Lorsque qu'une pression est exercée sur le tapis, la mousse est compressée et des contacts entre les deux feuilles métalisées s'établissent. Il peut être installé dans toutes les pièces de la maison sans imposer d'autres restrictions d'utilisation que le Linoléum. Le TAPIS^{metric} a pour but d'identifier tous les types de chutes et d'avertir les services de santé. L'installation de ce système peut être effectuée de la même manière que le Linoléum mais requiert l'intervention d'un électricien en courant faible pour faire la liaison des systèmes électroniques, problème P.1. Ce système a été conçu dans le but unique de prévenir les systèmes de santé et ne présente que peu d'intérêt pour des habitants autres que les personnes agées.

Le deuxième système mentionné, SensFloor, a été développé par les entreprises allemandes FuturShape et Infineon [15, 38]. Il se compose de puces électroniques directement intégrées dans les textiles, en une grille de 20cm. À chaque puce, huit capteurs de pression sont connectés. L'algorithme en charge du système est du type ADNOS. Cet algorithme permet de faire une cartographie automatique de la zone couverte en cas de panne d'un composant. De plus, la cartographie se remet à jour s'il y a un court-circuit et l'algorithme déconnecte automatiquement la partie du composant concernée, problème P.2. Ce tapis peut être utilisé aussi bien dans une maison intelligente et être couplé à d'autres systèmes, que pour faire du contrôle de sécurité dans un bâtiment [6]. Ce système montre bien l'intérêt d'avoir la capacité *plug and play* dans un réseau de capteurs puisqu'une maison reste un environnement à risques. Plusieurs solutions permettent de passer outre ces risques : avec un algorithme isolant des parties du sol, tel l'ADNOS, ou en isolant l'électronique à l'intérieur des dalles du sol.

Tous les sols intelligents présentés jusqu'à maintenant ont pour finalité une application dans la vie quotidienne, que ce soit domestique, bureautique ou industrielle. Les systèmes supportent une ou plusieurs applications comme l'identification ou la détection de personnes, la détection de chutes et l'observation des comportements. Cependant, aucun ne permet de décomposer et d'analyser la manière de marcher dans un but médical. Pour sa part, l'entreprise E.Q. Inc. [7] a conçu, pour la recherche sur la démarche, le tapis Gaitmat II. Il s'agit d'un tapis destiné uniquement aux domaines de la recherche et du milieu hospitalier puisqu'il maximise l'identification de neuf paramètres inhérents à la démarche,

comme la longueur du pas, la longueur de la foulée, les temps de chaque phase, etc. Du fait de ses caractéristiques : 3.84m de long, 0.6m de large et une matrice de 38x256 capteurs qui donnent une résolution de 15mm et de 5ms, le système est très couteux. Par ailleurs, il ne tient compte que des déplacements en ligne droite. Les paramètres analysés peuvent aussi servir à distinguer des personnes. Ce système démontre la possibilité d'optimiser l'identification mais ne résout pas le problème du coût. Il faut donc trouver un compromis entre fonctionnalité et coût d'installation.

2.3 Sol intelligent

L'exhaustivité des sols intelligents jusqu'alors conçus permet d'en décomposer la conception en trois étapes distinctes et indépendantes :

- Étape 1 : le choix et l'étude du ou des capteurs, §2.3.1
- Étape 2 : le choix et la mise en place de la topologie du réseau de capteurs, §2.3.2
- Étape 3 : le choix et la conception de la ou des techniques d'analyse des données, §2.3.3

Cependant, la création d'applications à base de sols intelligents ne nécessite pas obligatoirement l'enchainement de ces trois étapes. En effet, l'intégration des capteurs dans le sol intelligent est indépendant du réseau mis en place. Certains sols intelligents sont constitués d'un capteur directement relié à un ordinateur : ils mettent de côté la partie réseautique, pour afficher et interpréter les données issues du sol [1, 31]. D'autres mettent en place un réseau afin d'acheminer les données jusqu'à un ordinateur. Le fait qu'il existe des sols intelligents basés sur des réseaux en étoiles ou maillés montre que différentes topologies de réseaux mènent aux mêmes objectifs [19, 38]. Enfin, dès lors que l'on a choisi le type de données — analogique ou numérique — il est possible de mettre en place les techniques d'analyse des données avec des algorithmes d'intelligence artificielle [1, 28, 31, 37, 43].

2.3.1 Étape 1 : le choix et l'étude du ou des capteurs

La récolte d'informations se fait par l'intermédiaire de capteurs. Cette section présente les différentes technologies utilisées et envisagées dans les sols intelligents. Étant donné la volonté du laboratoire DOMUS d'être le moins intrusif possible, il n'est pas envisagé de faire porter un ou des capteurs à l'utilisateur ; l'aspect mobilité des capteurs n'est donc pas abordé. Avant d'aller plus loin dans la présentation des capteurs possibles, il faut savoir quelles sortes de mesures peuvent être faites à travers un plancher. Pour la détection d'une

ou plusieurs personnes, il apparaît que l'analyse du poids excercé sur le sol est suffisant d'après Kaddoura *et al.* [19] et Savio et Ludwig [38]. En ce qui concerne l'identification d'une personne, les études d'Addlesee *et al.* et d'Orr et Abowd se sont basées sur le profil de la force de réaction du sol lors d'un pas, en utilisant des capteurs de type cellule de charge, Figure 2.1 [1, 31]. Pour la détection d'activité, il est aussi possible d'utiliser des capteurs de force résistifs mesurant la pression exercée sur le sol ou encore des accéléromètres captant les vibrations qui se propagent dans le sol [2, 42].

Cellule de charge

Pour les besoins industriels, il existe de nombreux types de cellules de charge [29]. Leurs technologies peuvent reposer sur la variation d'une pression hydraulique ou pneumatique, mais aussi sur la déformation mécanique mesurée par des jauges de contraintes, etc. Dans le cas d'une utilisation dans un sol, les cellules de charge de types hydrauliques et pneumatiques ne sont pas envisageables du fait de la base même de leur technologie. Il est préférable d'utiliser des cellules de charge de type bouton ou de type S, Figure 2.2. Le principe d'utilisation de ces cellules de charge consiste à poser les dalles du sol intelligent directement sur celles-ci. La pression exercée sur une dalle est suffisante pour déformer la cellule de charge qui quantifie cette déformation et retourne une tension électrique. Ensuite, pour faire le lien entre la pression exercée et la tension électrique, il suffit d'appliquer l'équation suivante :

$$Force\ Mesurée = A \times Tension\ Mesurée + B \tag{2.1}$$

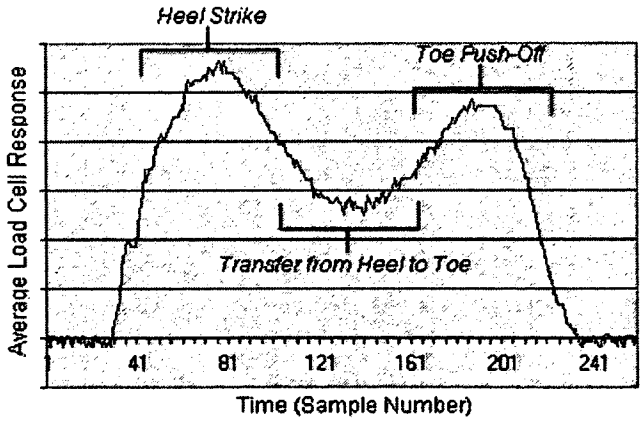


Figure 2.1 Profil de la force de réaction du sol GRF (source [31]).

avec

$$A = \frac{Charge\ Maximale}{Tension\ Nominale\ De\ Sortie}$$

$$B = 0 - A \times Tension\ \grave{A}\ Vide$$

Bien que les cellules de charge peuvent mesurer avec précision le profil GRF, elles sont difficiles à mettre en œuvre puisqu'il faut les étalonner une à une et fixer les paramètres A et B dans les nœuds correspondants.

Capteurs de force résistifs

Les ***Force Sensing Resistor*** (FSR), Figure 2.3, sont des capteurs imprimés flexibles et ultra-fins, d'épaisseur inférieure à 0.5mm. Ils sont constitués de deux films flexibles protecteurs sur lesquels est imprimé un motif avec de l'encre conductrice. Ainsi, lorsqu'une pression est exercée sur le capteur, la résistance varie de manière inversement proportionnelle [16]. Comme il s'agit d'un capteur imprimé, la forme des FSR n'est pas fixée, il en existe notamment en forme d'empreinte de pied. Pour une utilisation dans un sol intelligent, deux implémentations sont envisageables. La première dispose les capteurs qui sont protégés par un film sur le dessus de la dalle. Le capteur est ainsi directement activé par le pied de la personne. Cependant, cette installation expose le capteur à plus de risques d'un point de vue matériel. La seconde implémentation dispose les capteurs sous la dalle et limite les risques d'altération mécanique, problème P.2. De cette manière, toute force exercée sur la dalle est détectée par le FSR. Pour que cela soit possible, il faut que les dalles soient légèrement surélevées afin que les FSR soient disposés au niveau des pieds de la dalle surélevée [19].

Accéléromètre

Comme son nom l'indique, l'accéléromètre permet de calculer une accélération, suivant un ou plusieurs axes. Plusieurs technologies peuvent être à la base d'un accéléromètre, mais seuls les accéléromètres MEMS (***Micro ElectroMechanical Systems***) sont considérés dans cette étude. Ces capteurs sont formés à partir de deux grilles entrecroisées qui agissent comme un condensateur. Lors d'une accélération, les grilles se déplacent et modifient la valeur de la capacité du condensateur qui se trouve dans un circuit RC. De ce fait, la valeur de l'accélération appliquée est proportionnelle à la tension de sortie. L'ajout d'accéléromètres dans un sol intelligent est relativement simple. Effectivement, de par leurs petites tailles, $3mm \times 5mm \times 1mm$, ils ne sont pas encombrants, problème P.1, Figure 2.4. Selon son principe de fonctionnement, l'accéléromètre peut être fixé à une dalle sans subir de contraintes mécaniques de déformation ou d'écrasement. Aussi la durée

(a) (b)

Figure 2.2 Cellule de charge de type : (a) Bouton (Source [26]) (b) Déformation en S (Source [25]).

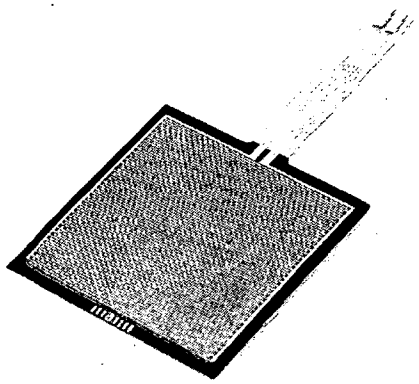


Figure 2.3 *Force Sensing Resistor* (Source [34]).

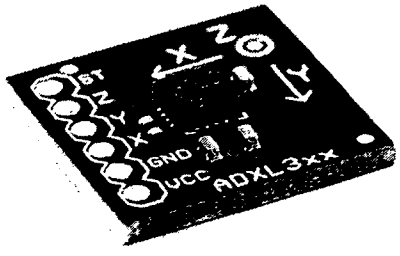


Figure 2.4 Accéléromètre 3 axes sur une carte d'interface [27].

Tableau 2.1 Comparatif entre les capteurs

	Cellule de charge	*Force Sensing Resistor*	Accéléromètre
Taille (Ordre de grandeur)	$\approx 25mm \times 25mm \times 10mm$	ultra-fin $\approx 1mm$	$\approx 4mm \times 4mm \times 1mm$
Charge maximale	1,000kg	450kg	5,000g
Étalonnage	Calcul du coefficient directeur et de l'ordonnée à l'origine	Ø	Ø
Contraintes mécaniques liées à l'utilisation	Déformation du capteur lors d'une surexposition à une charge	Ø	Bruits possibles venant des dalles voisines
Durée de vie	Perte de 1% de précision par an	-5% après 10 millions d'activations	plusieurs années
Prix	$50\$ \leq x \leq 100\$$	$5\$ \leq x \leq 10\$$	$x \leq 5\$$

de vie de l'accéléromètre est indépendante de l'utilisation du sol intelligent, problème P.2. Enfin, la plupart des accéléromètres ont une fonction de *Self Test* qui leur permet de vérifier l'intégrité de leurs mesures. Cependant, l'utilisation d'accéléromètres dans un sol intelligent est soumis à des limitations. Celles-ci viennent de la propagation à travers le sol des ondes vibratoires produites lors de la marche. Ainsi, avant d'envisager un sol intelligent constitué d'accéléromètres, il faut étudier la propagation des ondes à travers les dalles du sol. En effet, Alwan *et al.* et Werner *et al.* utilisent un nombre limité de capteurs : un capteur pour le premier et quatre pour le second, qui se situent à proximité des murs et qui détectent et localisent les chutes [2, 42].

Le tableau 2.1 propose une comparaison entre les trois types de capteurs présentés précédemment.

2.3.2 Étape 2 : le choix et la mise en place de la topologie du réseau de capteurs

Certaines études exposées précédemment montrent qu'il est possible de passer outre la deuxième étape. Pour Addlesee *et al.* et Orr et Abowd, le sol intelligent est constitué de capteurs directement reliés à un ordinateur [1, 31], tandis que Kaddoura *et al.* les répartit dans un réseau en étoile [19]. De cette manière, le routage de l'information à travers le réseau reste simple mais moins robuste vis-à-vis d'une défaillance d'un nœud, problème P.2.

Pour pallier au problème de la robustesse dans un réseau, Savio et Ludwig implémentent l'algorithme ADNOS dans un réseau maillé [24, 38].

Topologie du réseau

Une grande partie des sols intelligents analysés dans la section précédente mettent en évidence le problème de connexion entre les capteurs et le serveur qui effectuent les calculs de détection et/ou d'identification, problème P.1. Pour éviter toutes ces connexions, il est possible d'envisager un réseau de capteurs reposant sur une topologie de grille plutôt que sur une topologie en étoile. Dès lors, chaque connexion entre un capteur et le serveur se fait de manière indirecte en transitant par des nœuds intermédiaires.

Pour donner un exemple de réseau maillé, rappelons que Pépin *et al.* a conçu un sol intelligent composé de dalles qui communiquent avec chacune de leurs voisines mais aussi avec un agent situé sur la dalle [33]. Bien que cette étude considère le problème de planification de trajectoire multi-agents, l'architecture de ce réseau évite un câblage complexe. En effet, il n'y a pas besoin de relier chaque dalle à une unité de calcul centralisée puisque chacune communique avec ses voisines directes et ne connaît que celles-ci. Le principe de planification de trajectoire repose sur l'interrogation des dalles voisines, pour savoir si elles sont disponibles ou non, c'est-à-dire si un agent est positionné sur la dalle ou non, mais aussi sur la propagation locale de l'information lorsqu'une dalle change d'état. Les dalles disposent de deux moyens pour détecter la présence d'un agent. Le premier consiste en une liaison de communication directe avec l'agent, sans-fil ou à contact, et le second en l'analyse de la pression exercée sur la dalle par des capteurs situés sous la dalle. Ces deux méthodes diffèrent par la détection à coup sûr de l'agent, pour la première, et par la détection d'une présence qui n'est pas obligatoirement un agent pour la seconde. Pour pouvoir utiliser cette topologie avec un serveur, il suffirait d'ajouter un algorithme de propagation orienté vers un ou plusieurs nœuds connectés à un serveur.

Pour la couverture d'une grande surface, il est nécessaire d'installer un réseau de grande dimension. Du fait de son application, le réseau de Pépin *et al.* peut être agrandi facilement, problème P.3 [33]. La corrélation entre la taille du réseau et son utilisation est faible, puisque chaque dalle communique seulement localement. Cependant, dans le cas d'un réseau connecté à un serveur, le temps d'acheminement d'un message est dépendant du nombre de nœuds à traverser et donc de la taille du réseau. Ainsi, pour un réseau de grande dimension, il est envisageable d'introduire un ou plusieurs bus de communication entre nœuds. Dès lors, le temps de transmission d'un message dépendrait de la longueur du bus ainsi que du nombre de nœuds connectés à ce bus et non plus du nombre de nœuds

à traverser. Dans le cas d'un réseau multi-bus il faut ajouter le fait de passer d'un bus à l'autre. Malgré l'ajout du facteur de dépendance par rapport au temps d'acheminement d'un message, cette configuration devrait être plus efficace pour les grands réseaux. Il serait alors intéressant de comparer ces deux implémentations pour connaître les limites de chacune.

Routage de l'information

Le routage de l'information depuis les capteurs jusqu'au serveur est dépendant de la topologie du réseau. En ce qui concerne les sols intelligents dont les capteurs sont reliés directement à un ordinateur, comme pour ceux d'Addlesee *et al.* et d'Orr et Abowd, il n'y a pas de routage des données extraites des capteurs [1, 31]. Pour les réseaux de capteurs en étoile comme celui de Kaddoura *et al.*, un algorithme de routage est nécessaire [19]. Cependant, cet algorithme reste simple puisqu'il existe une notion de hiérarchie entre les nœuds. Aussi, le routage de l'information se fait de manière analogue au parcours d'un arbre planaire enraciné où les feuilles représentent les capteurs et la racine l'application serveur.

Concernant les sols intelligents mettant en œuvre un réseau maillé, il est nécessaire de concevoir un algorithme de routage permettant à chacun des nœuds de communiquer de manière indirecte avec le serveur. Il n'est pas obligatoire de mettre en place une liaison bidirectionnelle entre le serveur et les nœuds. En effet, les nœuds sont capteurs et non actuateurs. Contrairement à celui de Pépin *et al.*, le sol intelligent de Savio et Ludwig a besoin de communiquer avec un serveur [33, 38]. C'est pour cela qu'ils ont mis en place un algorithme de type ADNOS, problème P.2, [24]. Dans un premier temps, le réseau détecte les courts-circuits et isole les fils concernés. Puis, les nœuds communiquent avec leurs voisins directs pour avoir une première table de routage locale. Une fois que chaque nœud connaît ses voisins, le nœud connecté au serveur communique avec ses voisins directs pour les informer de sa présence et de sa direction. Ensuite, chaque nœud relaie cette information à tous ses voisins et ainsi de suite jusqu'à ce que tous les nœuds aient connaissance de la direction et de la distance vers le nœud connecté au serveur. C'est ainsi que tous les nœuds savent quelle est la route optimale pour joindre le portail. Enfin, le portail assigne une adresse à chacun des nœuds pour faciliter l'analyse des données. Dans le cas d'une défaillance d'une partie du réseau, ce même algorithme se répète et fait apparaître de nouveaux chemins.

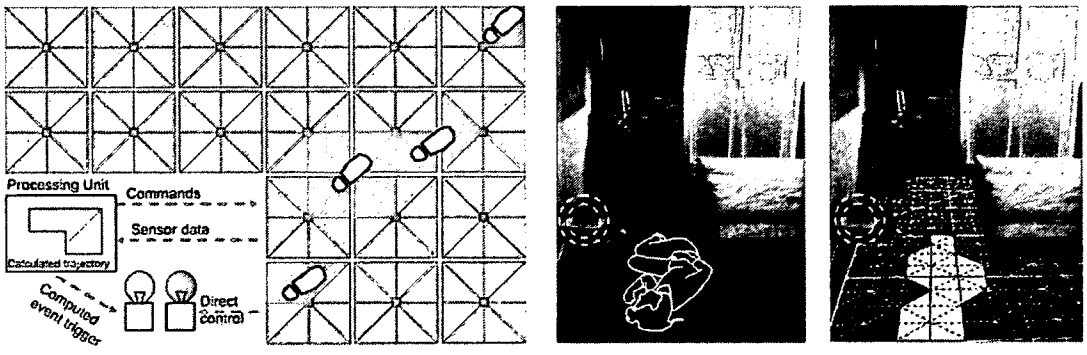
Autrement dit, lors de la conception d'un sol intelligent, il faut faire un compromis entre la complexité des connexions et celle du routage de l'information.

2.3.3 Etape 3 : le choix et la conception de la ou des techniques d'analyse des données

L'analyse des données est fonction de la nature des signaux captés. Dans le cas de signaux binaires, il est possible d'analyser l'activité sur le sol intelligent. En effet, avec une détection des zones du sol en cours d'utilisation, la surface active est accessible. Dès lors, il est possible de différencier deux contacts ponctuels qui font référence aux contacts des deux pieds d'une personne sur le sol ou au contact global pouvant signifier une chute, Figure 2.5 [8, 12, 38]. Une fois la chute détectée, il est possible de savoir si la personne est inconsciente ou si elle tente de se relever et même si elle réussit à se relever par la présence et le type d'activité sur le sol, en vue d'avertir les bonnes personnes relativement aux situations.

Pour des données provenant de capteurs analogiques, les méthodes d'interprétation du paragraphe précédent sont toujours envisageables et l'on suppose qu'il est possible d'identifier des personnes qui marchent sur le sol intelligent. En effet, dans le domaine analogique, le profil GRF de la démarche d'une personne peut être interprété et identifié [1, 31]. Cependant, plusieurs contraintes touchent l'identification des personnes. Il faut tout d'abord que la personne à identifier soit connue par l'application. La connaissance de cette personne par l'application est fonction de la méthode d'intelligence artificielle utilisée pour l'identification et elle nécessite une série de données dites d'entrainement. Il faut aussi que l'application prenne en compte le fait qu'une empreinte de pas peut être à cheval sur plusieurs capteurs à la fois.

En ce qui concerne l'analyse des activités domestiques, il peut être intéressant d'analyser et d'anticiper la trajectoire réalisée par la personne [9, 17]. De cette manière, le système



(a) (b)

Figure 2.5 (a) Détection des pas avec des contacts ponctuels. (b) Détection d'une chute avec un contact global. Source [8]

qui connaît le ou les chemins possibles pour effectuer une activité peut l'identifier plus rapidement. Il en va de même pour la détection de comportements dangereux.

Outre la possibilité d'identifier une personne marchant sur un sol intelligent par l'intermédiaire de son profil GRF, les sols intelligents peuvent aussi faire un suivi de trajectoire, pour une ou plusieurs personnes. Cette section présente quelques méthodes de calcul pour le suivi de trajectoire, sans entrer dans le détail des aspects matériels.

Un premier exemple de suivi de trajectoire se retrouve dans l'étude de Murakita *et al.* [28]. Cette étude utilise le sol intelligent de Vstone §2.2.2. Ici, le suivi de trajectoire est effectué par une méthode de ***Markov Chain Monte Carlo*** (MCMC). Cette méthode est particulièrement adaptée au suivi de trajectoire humaine puisque l'anticipation est faite en connaissance de l'état présent. De cette façon, quand un pied quitte le sol, la perte de signal n'affecte pas l'anticipation de l'état futur. Pour que la méthode MCMC fonctionne, il lui faut un modèle de prédiction. Pour cette raison, Murakita *et al.* fait la comparaison entre deux modèles de prédiction : (a) un modèle gaussien, donc linéaire, (b) un modèle bipède, hautement non-linéaire, Figure 2.6. L'étude comparative de ces deux modèles de prédiction montre que le modèle bipède nécessite moins de particules que le modèle gaussien pour un même résultat. De manière générale, il est possible de suivre une personne avec une erreur maximale de 59cm et moyenne de 20cm. Il est également possible de différencier deux personnes avec un taux de réussite de 90%, si elles se maintiennent à plus de 80cm l'une de l'autre.

Le filtre de Bayes peut aussi faire un suivi de trajectoire. Par exemple, Rajalingham *et al.* met en place un filtre de Bayes non paramétrique avec une approche de Monte Carlo [37]. Les données sont ainsi récoltées par un sol intelligent composé de 6×6 dalles carrées de 30cm de côté : chacune des dalles étant équipée dans les coins de quatre FSR. La comparaison faite par Rajalingham *et al.* entre les systèmes de surveillance par sol intelligent et par caméra met en évidence la perte de la résolution 3D concernant la position

(a) (b)

Figure 2.6 Modèle de prédiction pour la méthode MCMC : (a) gaussien, linéaire (b) bipède, non linéaire (source : [28]).

du corps, pour le premier. Elle met aussi au jour le fait qu'une caméra ne peut correctement analyser les interactions entre l'homme et les objets, notamment en ce qui concerne les relations de forces en jeu. Tandis qu'avec un sol intelligent, le suivi de trajectoire se fait par l'observation des forces appliquées sur ce sol ainsi que sur le modèle cinématique de la partie inférieure du corps, caractérisé par un vecteur à 19 dimensions. De cette manière, le système fait la différence entre le pied droit et le pied gauche, avec un succès de 100%. En ce qui concerne l'erreur de position, à mettre en relation avec la résolution du sol, elle est de 15cm. Avec un plus grand nombre de capteurs, il est possible d'extraire une meilleure analyse de la posture et des mouvements du corps, comme pour le contrôleur d'avatar de jeu vidéo de Yin et Pai [43].

CHAPITRE 3

TILEUS

La revue de littérature a mis en évidence la possibilité de décomposer en trois étapes la conception d'un sol intelligent. Dans le cadre de ce projet de maîtrise, seulement l'étape de conception du réseau est envisagée. Les deux autres étapes peuvent constituer de la même manière d'autres projets.

Pour aborder la question de la conception d'un réseau de capteurs, une étude portant sur les topologies de réseau est proposée, section 3.1. L'étude met en lumière les aspects problématiques de chacune des topologies envisagées. En considérant les problématiques générales du projet de sol intelligent — à savoir : une installation (P.1) et une maintenance (P.2) accessibles à tous, ainsi que la possibilité d'étendre facilement la surface du sol (P.3) — cette étude permet d'identifier les topologies adéquates à l'utilisation dans un sol intelligent et de choisir celle utilisée dans le premier prototype de réseau.

Pour que le développement du code du réseau soit simple et facilement ré-utilisable et modifiable, les fonctionnalités du nœud sont organisées suivant les trois couches matérielles du modèle OSI (*Open System Interconnection*), soit les couches Réseau, Liaison de données et Physique, section 3.2. Avec ce modèle, chaque couche inter-agit avec les autres via des interfaces. Ainsi, si l'on veut changer le type de micro-contrôleurs, il n'y a qu'à ré-implémenter la couche Physique, en respectant l'interface définie.

Lorsque toutes les couches nécessaires au fonctionnement du réseau TileUs ont été implémentées, le réseau peut être monté. Pendant leurs exécutions, les nœuds peuvent se trouver dans deux états : en cours d'initialisation ou opérationnel. Chacune de ces phases de fonctionnement est détaillée dans la section 3.3. La phase d'initialisation d'un nœud permet au réseau de s'initialiser automatiquement afin de simplifier l'installation, problème P.1. Quant à la phase opérationnelle du nœud, elle a pour fonction de capturer les données issues des capteurs tout en maintenant le nœud initialisé, cela rend le réseau dynamique face aux défaillances et aux ajouts de nœuds, problèmes P.2 et P.3.

Dans le but de confirmer le fonctionnement des algorithmes mis en place dans les trois couches matérielles, une étape de validation a été effectuée, section 3.4. Elle s'est faite sur deux supports différents. Le premier support est un micro-contrôleur MSP430F5529. Un réseau de trois nœuds de ces micro-contrôleurs a confirmé le fonctionnement local

des algorithmes. Pour le fonctionnement global du réseau, un simulateur a été réalisé, permettant de tester les algorithmes sur un réseau composé de plusieurs dizaines de nœuds.

3.1 Différentes topologies de réseau

La revue de la littérature a mis en évidence le fait que la conception de la partie réseautique d'un sol intelligent peut se faire de diverses manières. Les topologies de réseau existantes se répartissent en plusieurs catégories : réseau en étoile, en anneau ou encore en grille. Seule la catégorie en grille est étudiée ici. Dans un réseau en grille, il y a peu de relations hiérarchiques entre les nœuds. De ce fait, il est possible de faire le postulat suivant :

Tous les nœuds sont identiques.

Ceci permet de concevoir l'installation du réseau comme un simple puzzle à assembler, constitué de pièces toutes identiques. Les §3.1.1 et §3.1.2 proposent et examinent quatre types de topologies de réseau basés sur des communications directes ou via un bus. Quant à lui, le §3.1.3 examine les moyens de communication disponibles sur la plupart des micro-contrôleurs.

3.1.1 Topologie connexion point à point

À partir de la topologie de connexion point à point, il est possible d'extraire deux sous-topologies. La première est bidirectionnelle, les nœuds communiquent en *full duplex*. C'est à dire qu'il leur est possible d'envoyer un message tout en en recevant un, par l'intermédiaire de trois fils de communication entre les nœuds communicant. La seconde est unidirectionnelle, elle s'inscrit dans l'optique de réduire le nombre de fils à connecter, à deux, lors de l'installation, problème P.1.

Bidirectionnelle

La topologie de type connexion point à point bidirectionnelle se base sur la communication directe en un nœud et chacun de ses voisins. De ce fait, ce type de réseau présente une forte redondance, problème P.2 : dans un réseau décomposable en forme géométrique simple (rectangle), chaque nœud a au minimum deux voisins, donc deux connexions à la mise sous tension. Les nœuds situés dans les coins ont deux voisins, ceux sur les bords en ont trois et tous les autres en ont quatre, Figure 3.1.

Pour implémenter un tel réseau, il faut répondre aux points listés ci-dessous :

a) **Orientation du nœud,** comme un nœud est parfaitement symétrique, il n'y a pas besoin d'introduire une notion d'orientation, problème P.1.

b) **Placement des nœuds passerelles,** il faut introduire plusieurs nœuds passerelles, faisant la liaison entre le réseau et la centrale de calcul, afin de répondre au critère de redondance. À priori, il serait plus pertinent de les placer sur le bord du réseau pour réduire le problème de connexion avec la centrale de calcul, problème P.1. Le nombre de nœuds passerelles est dépendant du nombre de nœuds et de la forme du réseau. Dans le cas où un îlot se forme à la suite de défaillances, le fait d'avoir mis un nœud passerelle dans chaque forme géométrique simple suffit pour que le réseau puisse continuer à fonctionner normalement, problème P.2.

c) **Choix du nœud passerelle,** du fait de la présence de plusieurs nœuds passerelles dans le réseau, il faut que chaque nœud choisisse lequel de ses propres nœuds est la destination du message à envoyer. Il faut donc définir une heuristique pour ce choix : elle pourrait être basée sur la distance métrique, le temps de transmission ou encore la congestion du réseau.

d) **Défaillance d'un nœud,** l'algorithme de routage doit être en mesure de détecter et de prendre en compte la défaillance d'un nœud. Dans la plupart des cas, cette défaillance n'a que peu d'impact sur le réseau du fait de la forte redondance, problème P.2.

Unidirectionnelle

La topologie de type point à point unidirectionnelle repose sur une simplification matérielle de la topologie point à point bidirectionnelle. Cette topologie enlève un fil à chaque liaison inter-nœud et permet l'utilisation de deux ports de communication que la majorité des micro-contrôleurs possèdent. Elle implique une orientation dans le sens des communications entre les nœuds, Figure 3.2.

Cependant, pour implémenter une telle topologie, il faut résoudre les points suivants :

a) **Orientation du nœud,** elle doit être définie par rapport aux autres nœuds.

b) **Dialogue entre deux nœuds,** un dialogue entre deux nœuds du réseau implique un minimum de quatre nœuds et une plus grande connaissance du réseau pour chaque nœud. Il faut donc une plus grande table de routage, de taille dynamique.

c) **Agencement des colonnes et des lignes,** suivant le nombre de colonnes ou lignes, les nœuds des coins peuvent être isolés en émission ou réception.

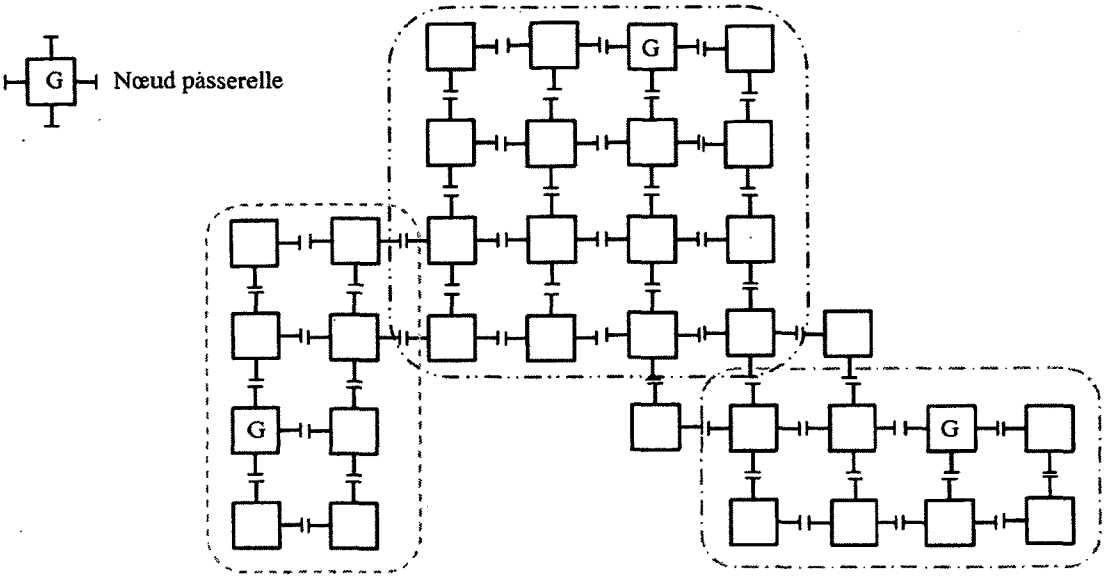


Figure 3.1 Exemple d'un réseau de type connexion point à point bidirectionnelle.

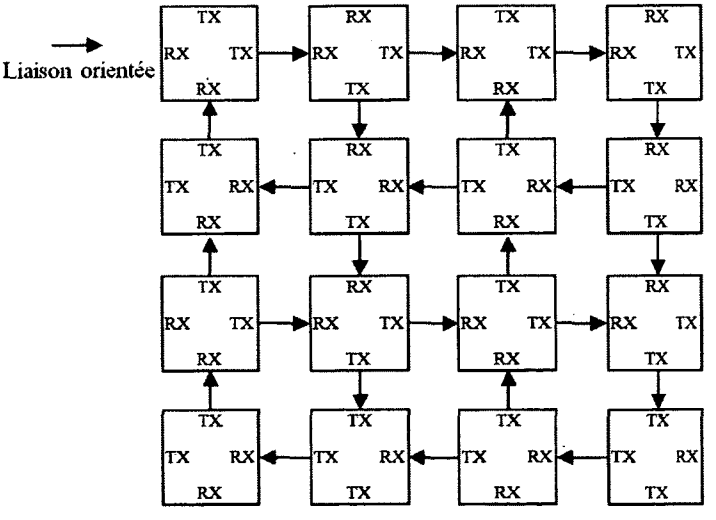


Figure 3.2 Exemple d'un réseau de type connexion point à point unidirectionnelle.

d) **Routage des messages inter-nœuds,** il est différent de la solution bidirectionnelle puisqu'il n'est alors plus possible d'émettre et de recevoir dans la même direction. Une solution est de faire un routage en connaissant la position relative du nœud destinataire par rapport à la position du nœud émetteur dans le réseau. Le routage des messages destinés au monde extérieur reste identique à la solution bidirectionnelle.

e) **Placement des nœuds passerelles,** un placement aléatoire peut enlever la redondance des communications. Il ne faut pas placer les nœuds passerelles dans les coins, ni sur les bords, pour que la redondance permette d'émettre les messages dans deux directions différentes. Même principe pour la réception, problème P.2.

f) **Défaillance d'un nœud,** la défaillance d'un nœud a des conséquences plus grandes dans le réseau. Dans un rayon minimum de deux voisins, les nœuds sont affectés par la chute d'un des leurs, problème P.2.

3.1.2 Topologie à base de bus de communication

Avec une communication reposant sur un ou plusieurs bus de communication, il devient possible pour un nœud de communiquer directement avec d'autres nœuds que ses voisins. Aussi, l'échange de messages entre deux nœuds distants ne fait intervenir qu'un nombre limité de nœuds, ce qui réduit le temps de transmission des messages.

Topologie comportant un bus commun

La topologie de réseau comportant un bus commun reprend les topologies de connexion point à point (uni ou bidirectionnelle) et ajoute un bus qui relie tous les nœuds du réseau. C'est une solution hybride entre les connexions point à point et l'utilisation de bus de communication. Cet ajout de bus a pour but de réduire le nombre de nœuds qui interviennent dans la transmission d'un message. Ainsi, pour communiquer avec ses voisins, le nœud fait intervenir les connexions point à point et pour transmettre un message au nœud passerelle relié à l'ordinateur, le nœud utilise le bus commun.

Cette topologie présente cependant quelques inconvénients :

a) **Taille du réseau,** avec l'ajout d'un bus commun reliant tous les nœuds du réseau, la taille du réseau est limité par la longueur du bus et le nombre de nœuds présents, problème P.3. Pour un bus de grande longueur, il faut ajouter à intervalle régulier des amplificateurs pour pallier au phénomène d'atténuation des signaux électriques. En ce qui concerne le nombre de nœuds, la limitation provient du protocole de communication choisi et du nombre d'adresses disponibles sur le réseau.

b) **Accès au bus,** plus il y a de nœuds présents sur le bus, plus l'accès au médium devient compliqué. Les techniques d'accès synchronisé sont conçues pour des envois périodiques de messages et posent problèmes lorsque l'on veut ajouter de nouveaux nœuds au réseau puisqu'il faut modifier l'ordonnancement des périodes d'émission, problème P.3. Avec les techniques d'accès asynchrone, le temps de transmission d'un message dépend fortement de la congestion du bus.

c) **Routage des messages,** avec un bus commun à tous les nœuds, les nœuds passerelles auraient la possibilité de lire tous les messages. Ainsi, le routage des messages serait superflu.

d) **Défaillance du bus,** l'utilisation d'un seul bus commun ne répond pas au critère de redondance. Une défaillance du bus peut avoir un impact considérable sur le réseau, problème P.2.

e) **Communication avec les voisins,** pour échanger des messages avec ses voisins, un nœud utilise les moyens de communication de la topologie point à point.

f) **Communication sans-fil,** un bus commun peut, par la suite, être remplacé par une transmission sans-fil.

Topologie comportant des bus colonnes et des bus lignes

Après la solution hybride avec l'utilisation de connexion point à point et un bus commun, voici une topologie composée uniquement de bus. Pour commencer, une description de la topologie sera faite, puis, ensuite, nous proposerons une méthode d'initialisation du réseau.

Description Un réseau composé d'une grille de bus permet à un nœud d'envoyer un message vers le monde extérieur sans avoir à traverser un grand nombre de nœuds. Avec cette topologie, un message traverse un nœud uniquement pour changer de bus : d'un bus colonne vers un bus ligne ou inversement, Figure 3.3. Avec une bonne répartition géographique des nœuds passerelles, un message est, dans la plupart des cas, à deux sauts des nœuds passerelles.

Cependant, pour qu'une telle topologie puisse être implémentée, il faut passer outre tous les problèmes listés ci-après. Ces problèmes font abstraction du type de bus utilisé.

a) **Choix du bus,** le choix du bus de communication peut se faire en fonction des contraintes liées à l'utilisation du réseau.

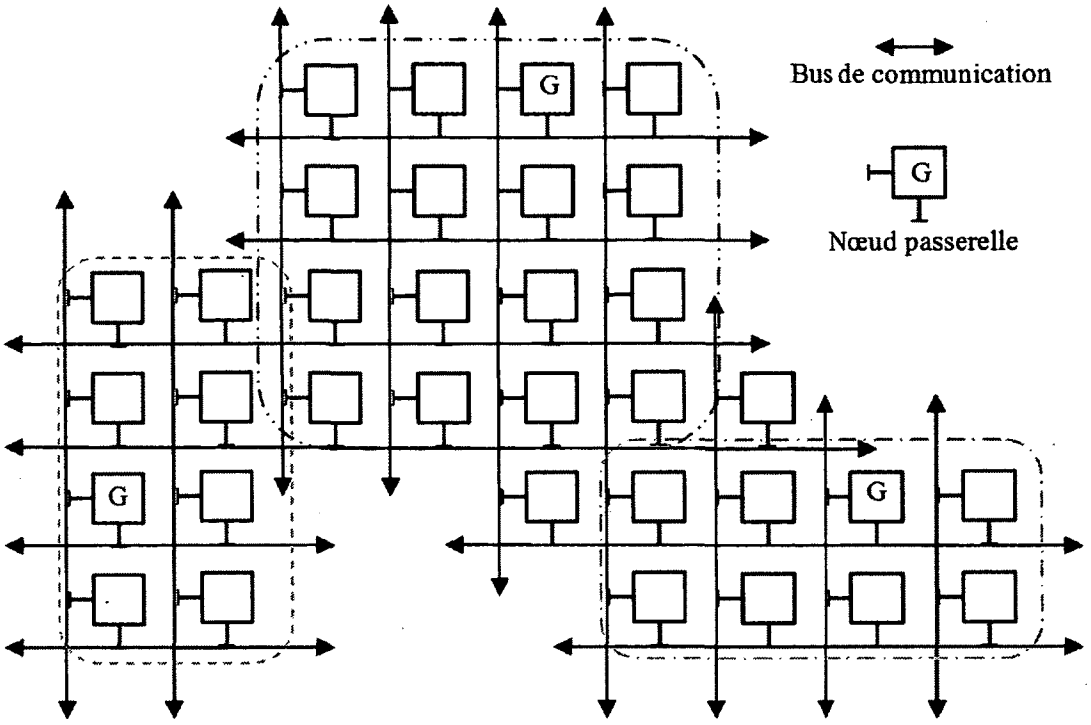


Figure 3.3 Exemple d'un réseau de type bus lignes/colonnes.

b) Accès au médium, l'émission de messages sur un bus peut introduire des collisions. Plusieurs protocoles peuvent être utilisés parmi ceux dits probabilistes. CSMA/CD peut être utilisé pour les bus SPI et UART mais pas I2C. CSMA/CA peut lui, être utilisé par ces trois bus de communication. Il existe aussi des méthodes déterministes de type bus à jeton ou TDM, mais elles sont plus complexes à mettre en œuvre.

c) Réalisation de la cartographie, pour réaliser la cartographie d'un réseau reposant sur des bus, il faut que les nœuds connaissent leurs positions relatives, c'est-à-dire qu'ils doivent connaître leurs voisins directs. Une solution est d'élire un premier nœud qui coupe le bus pour qu'il en résulte deux sous-réseaux. Sur chacun des sous-réseaux, les nœuds font des découvertes de voisins (en faisant un *broadcast* de leur ID). Après un temps fixe et prédéfini, le bus est rétabli et un autre nœud est élu pour couper le bus à un autre endroit et ainsi de suite. À la fin, les nœuds connaissent tous leurs voisins.

d) Changement de bus pour un message, pour qu'un message puisse être envoyé à travers le réseau, il doit être capable de passer d'un bus ligne à un bus colonne, et inversement. Pour cela, il est possible de déléguer le choix du nœud de changement de bus au nœud émetteur, avec un adressage du message. Le nœud émetteur doit

avoir dans sa table de routage les nœuds de changement. Sinon, le nœud émetteur peut faire un *broadcast* du message. Le message est analysé par chacun des nœuds sur le bus et c'est à eux que revient le choix du changement de bus. La première solution implique une connaissance plus complète du réseau dans chacun des nœuds et donc une couche réseau plus complexe. Tandis que la deuxième peut dupliquer les messages ou arrêter leur transfert prématurément, il faut alors reporter la complexité vers la couche transport.

e) **Choix du nœud passerelle,** comme plusieurs nœuds passerelles sont disséminés dans le réseau, pour la redondance, et qu'il existe plusieurs méthodes de changement de bus, le choix du nœud passerelle n'est pas immédiat. Il peut être fait de différentes manières. La première consiste à envoyer le message à tous les nœuds passerelles. La deuxième définit une heuristique qui peut être la distance en terme de nœuds à traverser ou en terme de changement de bus.

f) **Détection des défaillances,** pour détecter les nœuds défaillants, il est possible d'envoyer des messages périodiques entre nœuds demandant une réponse, problème P.2. Pour cela, il faut stocker tous les nœuds présents sur le bus dans la table de routage. Une défaillance détectée sera ainsi transmise aux nœuds concernés.

g) **Apparition de nouveaux nœuds,** lorsqu'un nœud est ajouté au réseau, il y a une grande probabilité qu'il soit ajouté à l'extrémité d'un bus (sauf cas de remplacement de nœud). Ainsi, pour savoir où le nœud se trouve, le principe de réalisation de la cartographie est appliqué par les nœuds situés aux extrémités, problème P.3. Si le nouveau nœud n'est pas localisé, il y a eu un remplacement de nœud et la cartographie reprend du début.

Méthode d'initialisation, pour pouvoir dire que le réseau est initialisé, il faut que chacun des nœuds connaisse ses voisins directs ainsi que les nœuds se trouvant sur les bus colonnes et lignes auxquels il est connecté. Il doit également pouvoir recenser tous les nœuds passerelles avec leur heuristique. Le choix de l'heuristique n'est pas fixé. Une méthode pour que les nœuds d'un bus connaissent leur position relative consiste à couper le bus à différents endroits puis à recenser les nœuds présents sur chacune des portions de bus restantes. Pour cela, il suffit d'ajouter à chacun des nœuds un *bus switch* sur le bus colonne et le bus ligne, Figure 3.4

Les étapes de cette initialisation vont être détaillées dans un exemple avec quatre nœuds connectés à travers un bus, Figure 3.5.

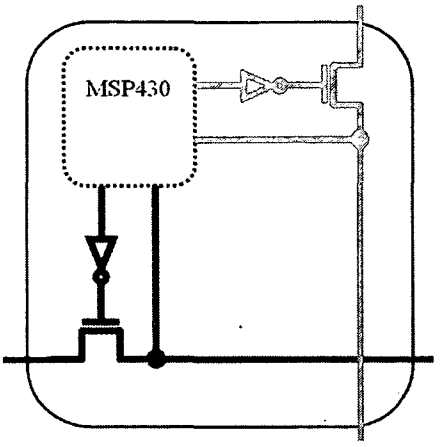


Figure 3.4 Schéma interne d'un nœud de topologie bus lignes/colonnes avec *bus switches*

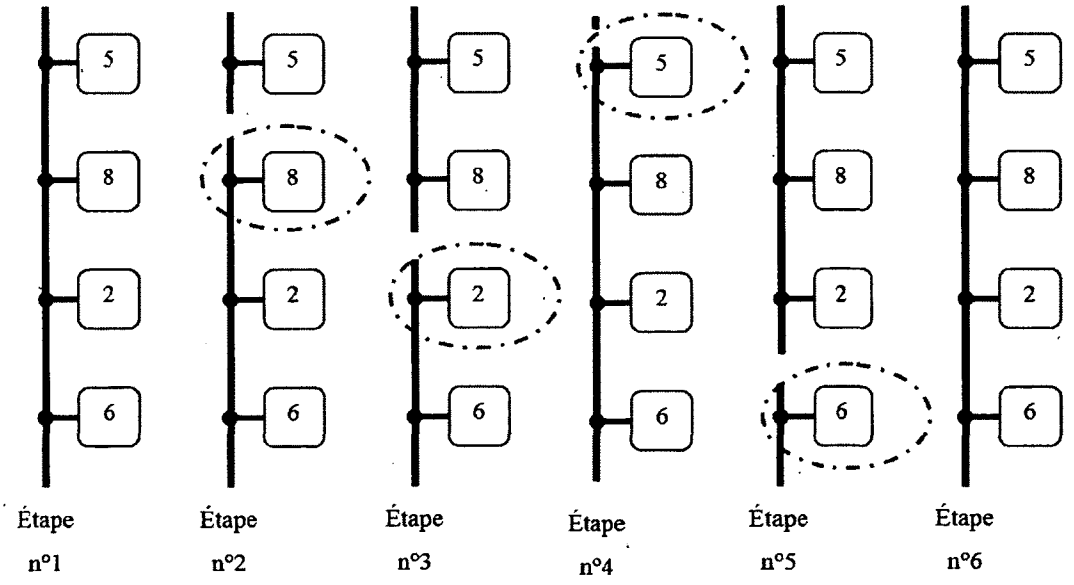


Figure 3.5 Exemple d'initialisation d'un bus ligne ou colonne

Étape n° 1 : à la mise sous tension du réseau, tous les nœuds du bus émettent un message tour à tour pour signaler leur présence. Une fois tous les nœuds signalés, soit après un temps d'inactivité sur le bus, ceux-ci peuvent calculer le temps alloué pour chaque coupure du bus. Par extension, il est possible de connaître le temps d'initialisation d'un bus.

Étape n° 2 : elle commence avec la prise de parole d'un nœud sur le bus. Dans l'exemple, le premier nœud à prendre la parole est le nœud n° 8. Celui-ci coupe le bus et il en résulte deux sous-réseaux. Dans chacun des sous-réseaux, les nœuds se signalent aux autres pour obtenir un recensement, Tableau 3.1. Cette étape permet de dire que le nœud n° 5 se situe à une extrémité du bus. Quand le temps alloué par coupure est dépassé, le nœud n° 8 reconnecte les deux sous-réseaux et le signale. Ensuite, la parole peut être prise par un autre nœud. Durant cette phase d'initialisation, le nœud n° 8 ne peut plus reprendre la parole pour couper le bus. Les étapes suivantes sont des répétitions de cette étape. À la fin, tous les nœuds pourront se positionner les uns par rapport aux autres sur le bus.

Étape n° 3 : ici, c'est le nœud n° 2 qui prend la parole et qui coupe le bus. Le recensement dans les sous-réseaux permet de dire que les nœuds n° 5 et n° 8 sont voisins ainsi que les nœuds n° 2 et n° 6. Comme l'étape n° 2 a permis de dire que le nœud n° 5 est à une extrémité, les différentes possibilités pour le réseau sont : {5, 8, 6, 2}, {5, 8, 2, 6}, {2, 6, 8, 5} ou {6, 2, 8, 5}, en considérant un classement en position absolue. Dans le cas d'un réseau TileUs, il importe peu de connaître les positions absolues, les positions relatives sont suffisantes. Dans cette optique les possibilités sont réduites par deux.

Étape n° 4 : elle permet de confirmer que le nœud n° 5 est à une extrémité du bus.

Étape n° 5 : elle indique que le nœud n° 6 se trouve à la deuxième extrémité du réseau.

Étape n° 6 : maintenant, la phase de coupure du bus de l'initialisation est terminée. Avec les informations récoltées aux étapes précédentes, il est possible de déduire l'agencement des nœuds sur le bus : {5, 8, 6, 2}.

Tableau 3.1 Recensement des nœuds en fonction des étapes durant l'initialisation d'un bus colonne ou ligne

Étapes	Nœuds considérés	Nœuds recensés	Nombre d'ordres possibles des nœuds
1	5	8, 2, 6	$4! = 24$
	8	5, 2, 6	
	2	5, 8, 6	
	6	5, 8, 2	
2	5	Ø	$2 \times 3! = 12$
	8	2, 6	
	2	8, 6	
	6	8, 2	
3	5	8	$2 \times 2! = 24$
	8	5	
	2	6	
	6	2	
4	5	8, 2, 6	Pas de changement, $2 \times 2! = 24$
	8	5, 2, 6	
	2	5, 8, 6	
	6	5, 8, 2	
5	5	8, 2	$2 \times 1! = 2$
	8	5, 2	
	2	5, 8	
	6	Ø	

3.1.3 Moyens de communication possibles

Pour illustrer ces différentes topologies de réseau, voici une description des trois moyens de communication répandus dans les micro-contrôleurs.

Le bus UART

Parmi les trois moyens de communication proposés par la majorité des micro-contrôleurs, seul l'UART propose un fonctionnement symétrique n'utilisant pas de nœud maître ou esclave. Ainsi, les nœuds n'ont pas besoin d'attendre la permission d'un autre nœud pour émettre. De plus, l'UART fonctionne avec deux signaux et trois fils : RX et TX, ce qui limite le nombre de connexions à faire entre les nœuds, problème P.1. Enfin, il est envisageable de connecter RX et TX en prenant soin d'ajouter des résistances de *pull up*. Ceci permet d'écouter le bus pendant l'émission d'un message et de savoir directement s'il y a collision entre plusieurs messages afin d'implémenter le protocole CSMA/CD, mais réduisant la communication de *full duplex* à *half duplex*. Le débit des modules UART est standardisé et peut aller de 9 600, 19 200, 38 400, 57 600 à 115 200 bauds.

Le bus SPI

Contrairement à l'UART, le SPI est un bus fonctionnant avec la notion de maître et esclaves. Pour que des nœuds puissent communiquer via SPI, à un instant donné, il faut qu'un seul nœud soit le maître de la communication et que tous les autres soient des esclaves. Il est envisageable d'avoir plusieurs nœuds maîtres sur le même bus. Cependant, il faut qu'un nœud passe de maître à esclave et inversement. Cela implique d'effectuer un changement dans les attributions des *pins* au niveau des entrées/sorties, par l'ajout de résistances de *pull up*. La mise en place matérielle d'un bus SPI requiert le plus de connexions entre nœuds parmi les trois bus exposés. En effet, un bus SPI est constitué de quatre fils : MISO (***Master Input, Slave Output***), MOSI (***Master Output, Slave Input***), SCLK (***Serial Clock***) et CS/$\overline{SS}$ (***Chip Select/ Slave Select***), problème P.1. Les deux fils MISO et MOSI permettent au SPI de fonctionner en *full duplex*. Finalement, deux choix sont offerts :

- implémenter le protocole CSMA/CA et pouvoir communiquer en *full duplex* ou,
- implémenter le protocole CSMA/CD en connectant les fils MOSI et MISO, et communiquer en *half duplex*.

La vitesse de transfert des données est contrôlée par le signal d'horloge transmis par le fils SCLK. Cette horloge est dérivée des différentes horloges disponibles sur le micro-contrôleur.

Le bus I2C

Comme le bus SPI, l'I2C fonctionne avec la notion de maître et esclaves. Dans ce cas, il ne peut y avoir qu'un seul maître sur le bus à un instant donné. Cependant, l'I2C a été conçu pour être un bus multi-maîtres. L'implémentation d'un bus I2C multi-maîtres est donc plus aisée que pour un bus SPI. Un bus I2C comporte deux signaux et trois fils : SDA (*Serial Data Line*) et SCL (*Serial Clock*) directement reliés à une résistance de *pull up*. Ainsi, l'I2C est comparable à l'UART en terme de connexions. Du fait de la présence d'un seul fil pour la transmission de données, l'I2C ne peut communiquer qu'en *half duplex*. Par conséquent, il n'est pas possible de détecter les collisions sur le bus et donc d'utiliser le protocole CSMA/CD, problème P.1. Comme pour le SPI, la vitesse de transfert des données est contrôlée par les différentes horloges disponibles dans le micro-contrôleur.

Pour que les nœuds d'un bus I2C puissent communiquer ensemble, il faut que chacun possède une adresse unique sur sept ou dix bits. Par ailleurs, dans le cas d'une implémentation de nœuds TileUs avec différents micro-contrôleurs, il se peut que le contrôleur I2C soit lui aussi différent et nécessite d'autres types de trames. De fait, il est difficilement envisageable de monter un réseau TileUs sans se soucier de ce qu'il y a à l'intérieur du nœud, problème P.3.

3.2 Modèle OSI de TileUs

Le développement du réseau TileUs s'est fait selon le modèle OSI. Ce modèle propose une organisation hiérarchique des propotocoles de communication suivant une pile de sept couches, dont trois dites matérielles et quatre dites hôtes. Pour le réseau TileUs, seulement les couches matérielles sont nécessaires et ont donc été implémentées. Leurs implémentations sont exposées dans les §3.2.1, §3.2.2 et §3.2.3.

3.2.1 Couche réseau

La couche réseau est la couche de plus haut niveau implémentée dans TileUs. Il s'agit de la couche matérielle qui s'occupe des communications de proches en proches ainsi que du routage des messages qui transitent dans le réseau. Elle est constituée d'une table de routage comprenant des informations sur ses quatre voisins. Cette couche garde aussi en mémoire des informations permettant d'identifier les dix derniers messages envoyés afin de détecter un éventuel cas de routage défaillant.

Table de routage

Un nœud TileUs peut recourir à deux méthodes distinctes pour envoyer des messages. Ces méthodes sont fonction du type de message à envoyer. La première méthode consiste à envoyer le message en *broadcast*. Cela ne veut pas pour autant dire que tous les nœuds reçoivent le message : il se répand à travers tous les nœuds concernés. Lorsqu'un nœud reçoit un message qui ne lui présente pas d'intérêt, il ne propage pas ce message. Ainsi, le *broadcast* circule uniquement dans les nœuds interprétant le message. La deuxième méthode consiste à envoyer un message vers le nœud passerelle. Selon cette méthode, seuls les nœuds se trouvant sur le chemin entre le nœud passerelle et le nœud d'émission reçoivent et acheminent le message, sans pour autant interpréter le message.

Pour un nœud, la connaissance du réseau se situe dans sa table de routage, Tableau 3.2. La table de routage de chaque nœud comporte quatre lignes qui renvoient aux quatre voisins potentiels, ainsi qu'un pointeur qui renvoie vers la ligne dite meilleure. Elle correspond à la ligne dont le nombre de sauts jusqu'au nœud passerelle est minimal. Le Tableau 3.2 est un exemple de table de routage. Dans celui-ci, on peut lire à la ligne *NORTH* que l'identifiant du nœud est 0 et que son nombre de sauts est -1. Cela indique que dans la direction nord, le nœud ne possède pas de voisin ou qu'il est incapable de communiquer avec.

Pour envoyer un message au nœud passerelle, il suffit d'envoyer le message dans la direction de la meilleure ligne de la table de routage. Dans le cas où deux lignes ont le même nombre de sauts pour atteindre la destination, la direction d'envoi correspond à la direction qui apparaît en premier dans la table de routage. Le nœud voisin qui reçoit ce message fait de même jusqu'à atteindre un nœud passerelle. Le nœud passerelle transmet ensuite le message à l'application en charge.

Routage dans un cas de défaillance

Un cas de défaillance dans le réseau survient lorsqu'une liaison entre deux nœuds est rompue ou lorsqu'un nœud est hors service. Dans ce cas, les nœuds voisins détectent ces défaillances lors de l'échange d'un message de type quelconque. En effet, comme il y a

Tableau 3.2 Exemple de table de routage.

Direction (1 octet)	Identifiant (8 octets)	Nombre de sauts (1 octet)
EAST	22 D4 CC 47 07 00 09 00	5
SOUTH	60 D9 B4 46 73 0C BD 06	4
WEST	62 14 90 46 23 00 12 00	4
NORTH	0	-1

Tableau 3.3 Ligne de la base de données de stockage des messages.

Identifiant du nœud d'origine du message	Numéro d'émission du message	Direction de réception de la première itération du message	Port d'émission/réception utilisé (1 bit chacun)							
			Émission				Réception			
			N	W	S	E	N	W	S	E

défaillance, le message n'est soit pas correctement transmis, soit non transmis. La sous-couche LLC du nœud émetteur détecte donc la défaillance après les trois essais d'émission.

Juste après la détection de la défaillance, le nœud modifie sa table de routage en mettant -1 dans la case du nombre de sauts. Ultérieurement, le nœud recommencera à émettre des requêtes d'initialisation vers ce nœud pour vérifier si la défaillance persiste ou si le nœud a bien été remplacé.

Avant chaque émission de message, le nœud stocke les identifiants du message dans une base de données, Tableau 3.3. De cette manière, à chaque réception, le nœud est capable de dire si le message a déjà transité par lui ou non et d'éviter ainsi tout bouclage. La base de données peut actuellement stocker 10 messages. Il faut préciser que l'information stockée ne permet pas de connaître l'information transmise mais de différencier le message de tous les autres : il s'agit d'un identifiant unique extrait de l'identifiant du nœud d'origine et du numéro d'émission du message. La base de données ne stocke pas tous les messages, seulement ceux de données et de cartographies, car il est inutile de considérer les messages locaux et périodiques. De plus, à chaque fois que le même message est reçu ou envoyé, le nœud garde en mémoire les ports utilisés. Si tous les ports ont été utilisés, le nœud renvoie le message dans la direction de réception de la première itération du message. Ainsi, lors d'une défaillance, un message retrouve le chemin du nœud passerelle, après une exploration de type profondeur d'abord.

Le message envoyé après une défaillance ne parcourt pas obligatoirement le chemin optimal mais atteint tout de même sa destination. Il est donc possible que le message ne respecte pas les contraintes temps réel du réseau et/ou de l'application. Par ailleurs, les nœuds, dont des lignes de la table de routage ont été désinitialisées recommencent à envoyer des requêtes d'initialisation dans les directions des lignes de routage non initialisées, selon les principes de maintien du réseau abordés au §3.3.2.

3.2.2 Couche liaison de données

La couche liaison de données se situe en dessous de la couche réseau. Elle a pour rôle de s'assurer de la fiabilité de la transmission entre deux nœuds du réseau. Pour cela, elle intègre la sous-couche LLC qui s'occupe du contrôle logique des données et la sous-couche MAC qui contrôle l'accès au médium.

Sous-couche MAC

L'implémentation de la sous-couche MAC dans le réseau TileUs est allégée. En effet, puisque chaque communication est point à point, il est superflu d'inclure un contrôle d'accès au médium. Aussi, la sous-couche MAC a pour unique fonctionnalité de détecter la fin et le début des trames à l'aide d'un entête introduit dans celles-ci, Tableau 3.4.

Cet entête comprend en premier lieu un octet qui définit le type du message : il peut être un message d'initialisation, une demande d'initialisation, un message comportant une ligne de la table de routage ou encore un message de données. Dans le cas du message de données, un deuxième octet est nécessaire pour connaître la taille des données envoyées. Ainsi, la taille d'un message de données n'est pas restreint à la conception du réseau. Pour les autres types de messages, l'octet de type de message permet de l'identifier et donc d'en connaître la taille. À la fin de la trame vient un octet de fin de trame : Couplé avec le premier octet, ils resynchronisent le traitement des trames s'il y a perte de synchronisation. Un tel cas peut survenir lorsque le *buffer* d'entrée dans le processeur est rempli. Le choix est fait de supprimer les trames les plus anciennes, qui peuvent ne plus répondre aux contraintes temps réel, pour laisser la place aux trames plus récentes. Ceci arrive notamment dans la phase d'initialisation lorsque le *buffer* est rempli de demandes d'initialisation. Il est aussi envisageable qu'un tel bourrage *buffer* puisse survenir lors d'une surcharge de messages dans le réseau.

Sous-couche LLC

L'implémentation de la sous-couche LLC se rapproche du type 2 dans les services de transmission. C'est-à-dire que la sous-couche aiguille bien les données vers la couche suivante, en incluant tout de même un contrôle de séquence, un contrôle de flux et des acquittements.

Tableau 3.4 Entête de la sous-couche MAC.

Type de message	Taille des données (Si message de type données)	Message avec entête LLC	Octet de fin de trame
1 octet	1 octet	n octets	1 octet

Tableau 3.5 Entête de la sous-couche LLC.

Message utile	CRC16
n octets	2 octets

Le contrôle de séquence se fait par l'intermédiaire du calcul d'un CRC16 sur la trame, Tableau 3.5. Lorsqu'un nœud reçoit une trame, il vérifie son contrôle de séquence. Si le contrôle de séquence est positif, le nœud récepteur envoie au nœud émetteur un acquittement positif (ACK). Dans le cas contraire, il renvoie un acquittement négatif (NAK). Le contrôle de flux s'effectue de sorte qu'à chaque réception d'un acquittement négatif ou de non-réception d'acquittement la période de répétition d'émission de messages redondants, comme les requêtes d'initialisation ou les messages d'initialisation, est doublée jusqu'à un maximum. Dans le cas où le nœud reçoit un acquittement positif, la période d'émission est réinitialisée.

Lors de l'émission d'une trame, si le nœud ne reçoit pas d'acquittement positif, il essaie de renvoyer la trame deux fois supplémentaires. Dans ce cas, un délai entrecoupe les envois de trame. Si au bout des trois essais d'émission, le nœud n'a toujours pas reçu d'acquittement positif, le nœud en conclut qu'il y a un problème dans la communication et relaie cette information à la couche réseau.

3.2.3 Couche Physique

La couche physique est la couche de plus bas niveau du modèle OSI. Elle s'occupe de transmettre octet par octet les messages entre deux nœuds. Pour cela, elle a accès aux fonctionnalités propres du micro-contrôleur utilisé, §3.4.1. Ainsi, il faut développer une couche physique pour chaque famille de micro-contrôleur différente utilisée dans le réseau TileUs. Étant donné la dépendance entre l'implémentation de la couche physique et le choix du micro-contrôleur utilisé, la description de cette couche reste volontairement large, mais il faut que le micro-contrôleur puisse au moins établir une connexion distincte pour chacun de ses quatre voisins. Les détails de son implémentation sont donnés au §3.4.1. La couche réseau doit proposer les deux fonctionnalités suivantes :

- Écriture d'un octet sur le médium.

- Lecture d'un octet sur le médium.

3.3 Phases de fonctionnement

Lorsque l'implémentation du réseau TileUs est faite, il est possible de distinguer deux étapes de fonctionnement différentes. Dans la première, chaque nœud du réseau cherche à s'initialiser, afin de connaître ses voisins et la direction que prendront les messages à destination du nœud passerelle qui transmet les messages à un ordinateur, §3.3.1. Une fois qu'un nœud est initialisé, il rentre en phase opérationnelle où il peut envoyer les données issues du ou des capteur(s) et maintenir sa table de routage §3.3.2. Le maintien de la table de routage permet de détecter les quelconques défaillances des nœuds voisins, ainsi que l'ajout de nouveaux nœuds au réseau.

3.3.1 Phase d'initialisation

Avant de commencer l'explication de la phase d'initialisation, il faut d'abord définir ce qu'est un nœud initialisé. Une fois cette définition posée, les mécanismes de découvertes des voisins peuvent être étudiés. Une fois qu'un ensemble de nœuds est initialisé, il est possible pour l'application analysant les données d'établir une cartographie du réseau, dans le but de traiter de manière cohérente les informations transmises par les nœuds.

Paramètres à initialiser

Pour qu'un nœud TileUs puisse être initialisé, il doit au moins connaître tous les paramètres d'un nœud voisin, c'est-à-dire :

- l'identifiant du nœud voisin : il doit être unique pour pouvoir distinguer chaque nœud.
- le nombre de nœuds à traverser pour atteindre un nœud dit passerelle (*Gateway*) qui sert de liaison avec le monde extérieur.

La table de routage regroupe les informations de chacun des quatre voisins potentiels. Pour un nœud voisin, elles sont stockées dans une ligne de la table de routage, Tableau 3.2. Pour un nœud autre que passerelle, l'état d'une ligne de la table de routage est fonction de la connaissance du nœud voisin et de la possibilité de communiquer avec lui, Figure 3.6. En fait, il suffit qu'une des quatre lignes de la table de routage soit initialisée pour que le nœud puisse l'être aussi, Figure 3.7.

La phase d'initialisation du réseau peut être décomposée en deux étapes qui séparent la connaissance des voisins et la diffusion de la carte topologique du réseau.

Étape 1 : connaissance des voisins

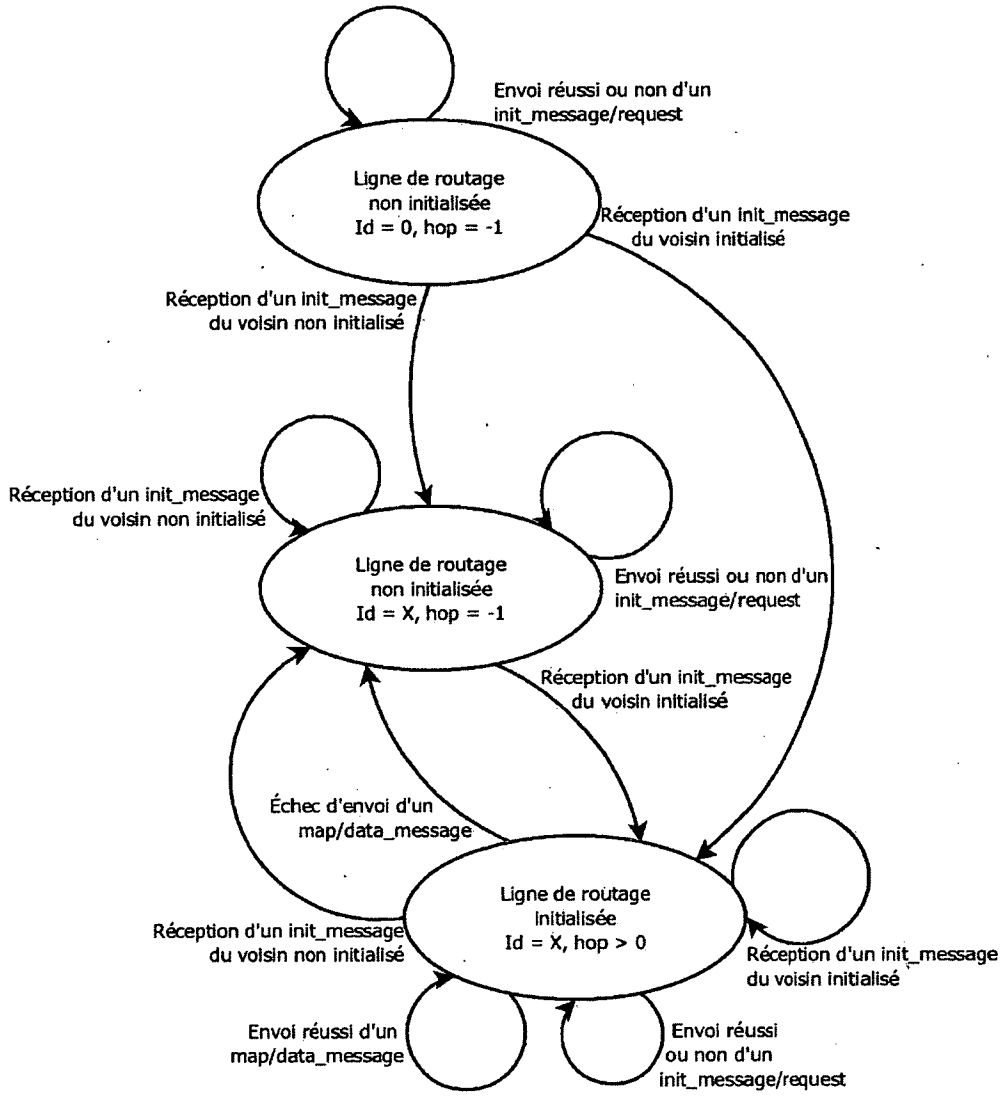


Figure 3.6 Machine à états finis d'une ligne de routage, pour un nœud non passerelle.

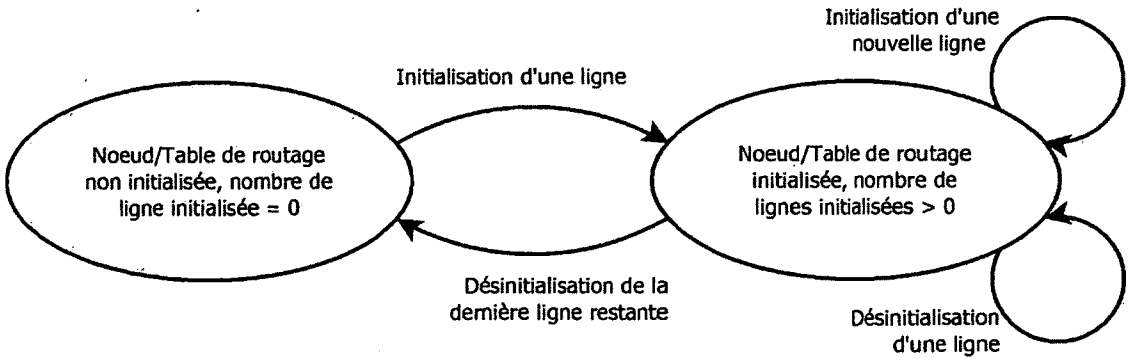


Figure 3.7 Machine à états finis de la table de routage, pour un nœud non passerelle.

Avant leur initialisation, les paramètres de chaque nœud ont une valeur par défaut : ils ont tous un 0 dans la case de l'identifiant des voisins et -1 dans la case du nombre de nœuds à traverser pour atteindre un nœud passerelle. Dans le cas d'un nœud passerelle la deuxième case de chaque voisin est mise à 0, et elle le restera tout au long de la vie du réseau. Les nœuds passerelles sont initialisés par défaut dès la mise sous tension du réseau.

Au cours de la phase 1, les nœuds envoient une requête d'initialisation à chacun de leurs voisins. La fréquence d'envoi de ces requêtes diminue avec le temps si le nœud ne reçoit aucune réponse de ses voisins, ce qui évite de congestionner le réseau à l'initialisation.

Lorsqu'un nœud n'est pas initialisé et qu'il recoit une requête d'initialisation, il ne prend pas en compte ce message. En revanche, s'il est initialisé, il renvoie un message d'initialisation, Figure 3.8, Tableau 3.6, Annexe A, à l'émetteur de la requête. Ce sont donc les nœuds passerelles qui permettent l'initialisation.

À la réception d'un message d'initialisation, le nœud met à jour sa table de routage. Si un changement intervient dans le nombre optimal de sauts, le nœud renvoie à tous ses voisins un message d'initialisation. Ainsi, chaque modification apportée au réseau se propage localement.

Autrement dit, tous les nœuds du réseau peuvent s'initialiser. Cependant, à l'instant où un nœud s'initialise, il ne connaît pas toutes les informations de chacun de ses voisins, uniquement d'un seul. Pour connaître celles qui manquent, il envoie des requêtes d'initialisation aux voisins, de la même manière que pour la demande d'initialisation : avec une diminution de la fréquence d'envoi.

Étape 2 : Récupération de la topologie du réseau

À chaque fois qu'une ligne de la table de routage est complétée ou modifiée, le nœud envoie l'information au nœud passerelle, Figure 3.9, Tableau 3.7, Annexe A, ce qui permet à l'application de faire la cartographie du réseau, Annexe B.

L'envoi d'un message à travers le réseau est possible dès lors qu'un nœud est initialisé et qu'il n'y a pas eu de modifications morphologiques (liaison coupée entre deux nœuds, défaillance d'un nœud, etc.) dans le réseau. Dans le cas d'une modification, les mécanismes permettant de la détecter et de la prendre en compte sont implémentés dans la couche réseau du protocole de communication, §3.2.1.

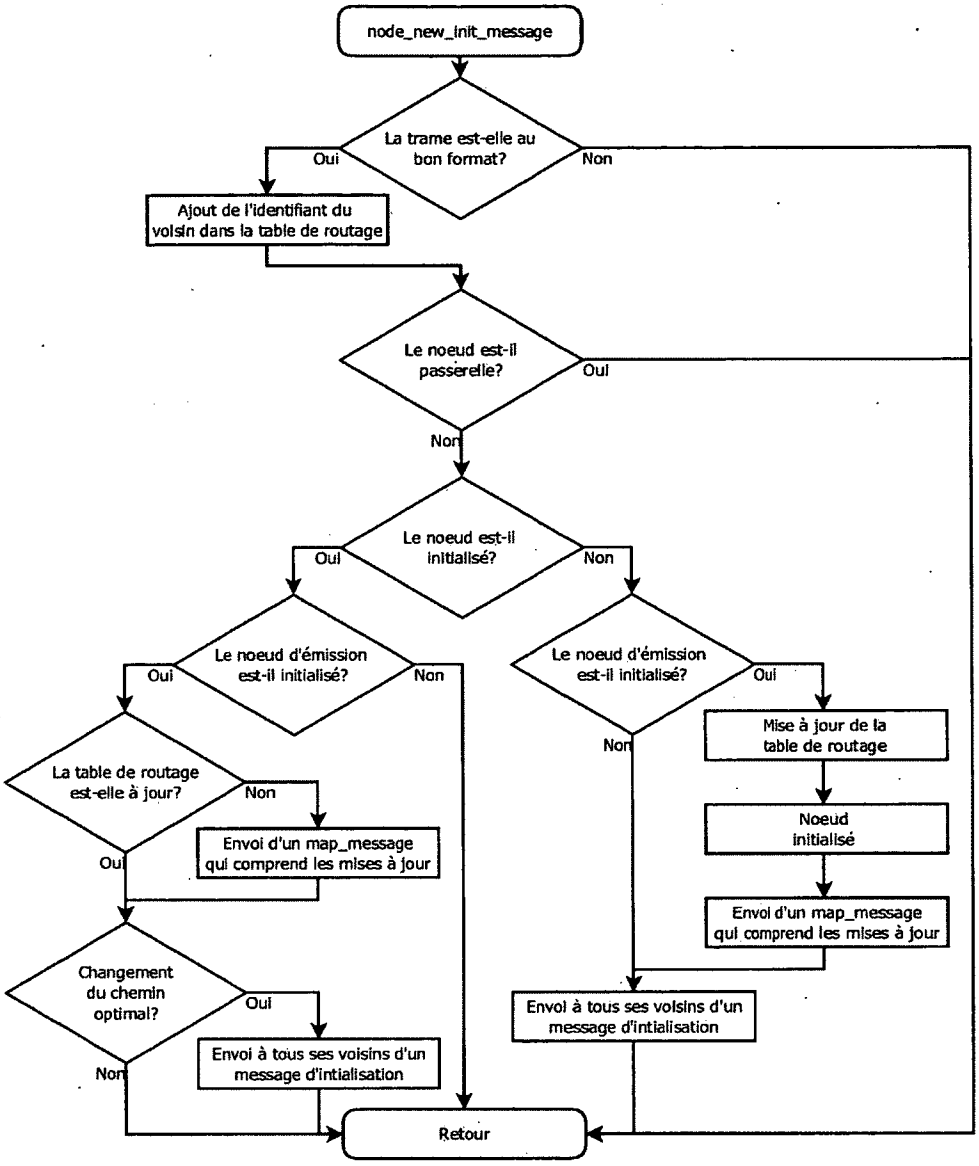


Figure 3.8 Organigramme de la fonction d'envoi d'un message d'initialisation

Tableau 3.6 Format d'un message d'initialisation

Type du message	Identifiant du nœud d'émission	Meilleur nombre de sauts à partir du nœud	CRC	Balise de fin
1 octet	8 octets	1 octet	2 octets	1 octet

Tableau 3.7 Format d'un message de cartographie

Type du message	Identifiant du nœud d'origine du message	Direction de la liaison avec le nœud voisin	Identifiant du nœud voisin
1 octet	8 octets	1 octet	8 octets
Nombre de sauts en passant par le nœud voisin	Numéro d'émission du message	CRC	Balise de fin de message
1 octet	1 octet	2 octets	1 octet

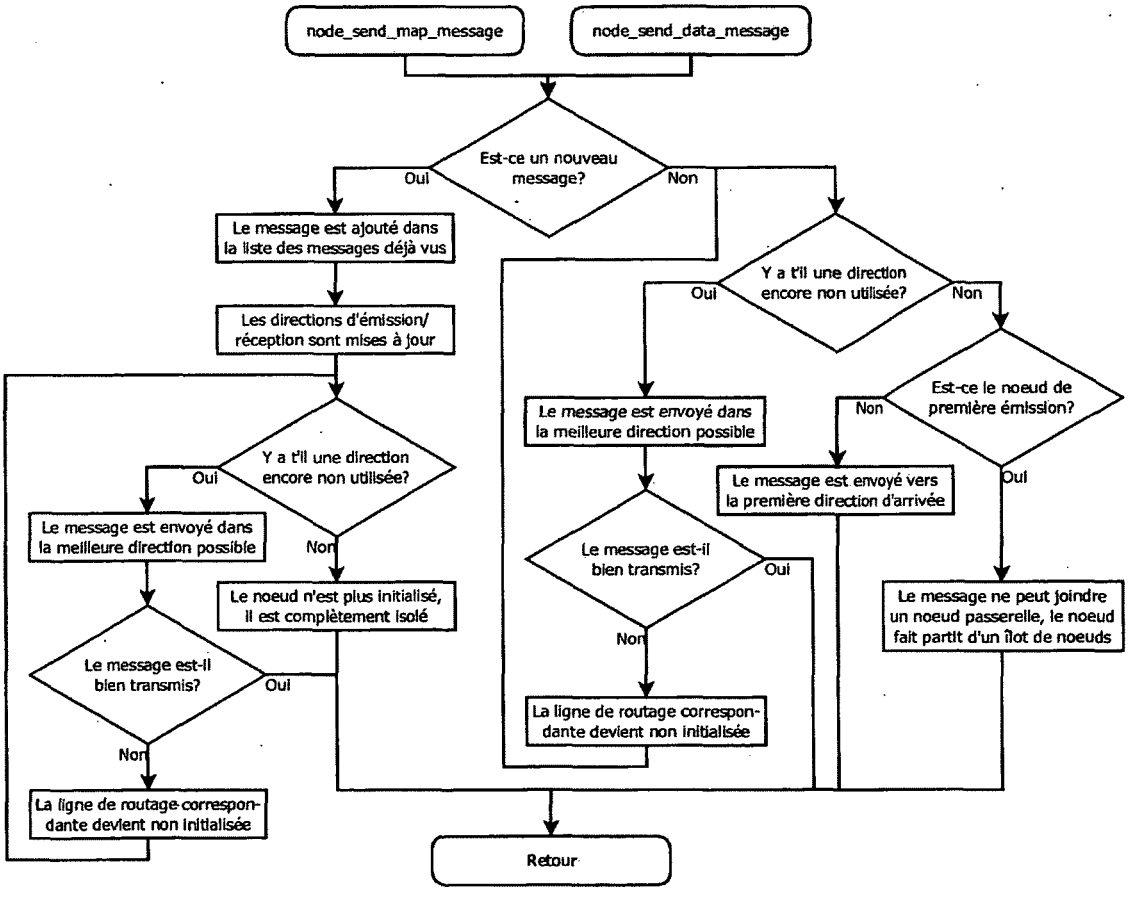


Figure 3.9 Organigramme de la fonction d'envoi d'un message de cartographie ou de données

3.3.2 Phase opérationnelle

Après l'initialisation d'un nœud, la phase opérationnelle commence. Dans cette phase s'effectue la capture de données provenant du ou des capteur(s) présent(s) sur le nœud. Le nœud maintient également sa table de routage pour détecter toutes défaillances ou nouveaux nœuds. Au niveau de l'application s'exécutant sur l'ordinateur relié au réseau TileUs, l'affichage et l'analyse des données commencent aussi.

Collecte d'informations

Lorsqu'un nœud est initialisé, il peut commencer à collecter des données. Celles-ci proviennent des capteurs interfacés avec le micro-contrôleurs. Comme les micro-contrôleurs intègrent des convertisseurs analogiques/numériques et des *timers*, les nœuds TileUs permettent aussi bien l'utilisation de capteurs à sorties binaires, analogiques ou en forme de *PWM* (***Pulse Width Modulation***). Le moyen de collecter des données est libre de choix. Il est tout aussi possible d'échantillonner les signaux venant des capteurs que de capter leurs transitions d'états, selon des seuils fixés.

Lorsque des données sont disponibles, le nœud les insère dans un message de données, Figure 3.9, Tableau 3.8, Annexe A. Une fois le message assemblé, il est envoyé vers le nœud passerelle le plus proche, selon le principe de routage de la couche réseau §3.2.1. Le message est alors envoyé, via une connexion série entre le nœud passerelle et un serveur sur lequel s'exécute une application pouvant interpréter ces données.

Maintien du réseau et *Plug and Play*

Une fois la phase d'initialisation complétée, tous les nœuds du réseau connaissent leurs voisins ainsi que les directions pour atteindre les nœuds passerelles les plus proches. Cependant, comme pour tout matériel électronique les nœuds TileUs peuvent tomber en panne. Deux types de panne peuvent survenir. Le premier type concerne un nœud avec lequel il n'est plus possible de communiquer, le nœud est dit hors-service. Tandis que le second

Tableau 3.8 Format d'un message de données

Type du message	Taille des données	Identifiant du nœud d'origine du message	Numéro d'émission du message
1 octet	1 octet	8 octets	1 octet
Données		CRC	Balise de fin de message
n octets, $n \in [1; 255]$		2 octets	1 octet

définit un nœud pour lequel une direction de communication n'est plus opérationnelle, il est toutefois possible de communiquer avec lui via une autre direction de communication.

Lorsqu'un nœud ne collecte pas l'information ou ne relaie pas de message vers un nœud passerelle, il effectue un certain nombre d'opérations visant à maintenir le réseau fonctionnel. Comme le montre la partie droite de l'organigramme, Figure 3.10, ces opérations sont effectuées après le déclenchement d'un `TimeOut`. La première opération consiste à envoyer des messages d'initialisation à tous les voisins du nœud considéré. De cette manière, chaque voisin peut maintenir sa table de routage et connaître la direction optimale vers le nœud passerelle le plus proche. Comme un nœud TileUs est directement intégré dans le sol, il possède une protection matérielle. C'est pour cette raison que cette opération n'a pas lieu à chaque déclenchement du `TimeOut`, mais après `INIT_RESEND` déclenchements.

La deuxième opération permet de ne pas surcharger les nœuds voisins de messages lorsqu'ils n'y répondent pas. Dans ce cas, le temps avant le déclenchement du `TimeOut` est doublé jusqu'à une valeur maximale. Ce temps se réinitialise à chaque fois que le nœud reçoit un message.

Vient ensuite une opération qui se réalise uniquement lorsque le nœud n'est pas initialisé. Cela peut se produire à la mise sous tension du nœud ou après une défaillance qui l'isole des nœuds passerelles présents dans le réseau. Comme le nœud n'est pas initialisé, il émet des demandes d'initialisation à chacun de ses voisins et espère recevoir au moins une réponse qui lui permettrait de s'initialiser.

Enfin, une dernière opération prend en compte le cas où toutes les lignes de la table de routage ne se trouvent pas dans l'état initialisé de la figure 3.6. Ici, le nœud envoie une demande d'initialisation aux voisins concernés.

Toutes ces opérations permettent de maintenir le réseau, de garantir la connaissance d'un chemin optimal vers un des nœuds passerelles, de prendre en compte les défaillances possibles et d'agrandir le réseau pendant son utilisation.

Interprétation des données

Au fur et à mesure de l'utilisation du réseau TileUs, l'application s'exécutant sur le serveur connecté aux nœuds passerelles reçoit des messages de cartographie et des messages de données. Comme précédemment, dans l'étape 2 de la phase d'initialisation §3.3.1, un message de cartographie comprend les informations contenues dans une ligne de la table de routage d'un nœud. Ainsi, l'application possède, au fur et à mesure de l'évolution du réseau, une connaissance de chacune des tables de routage des nœuds. Elle peut alors reconstituer

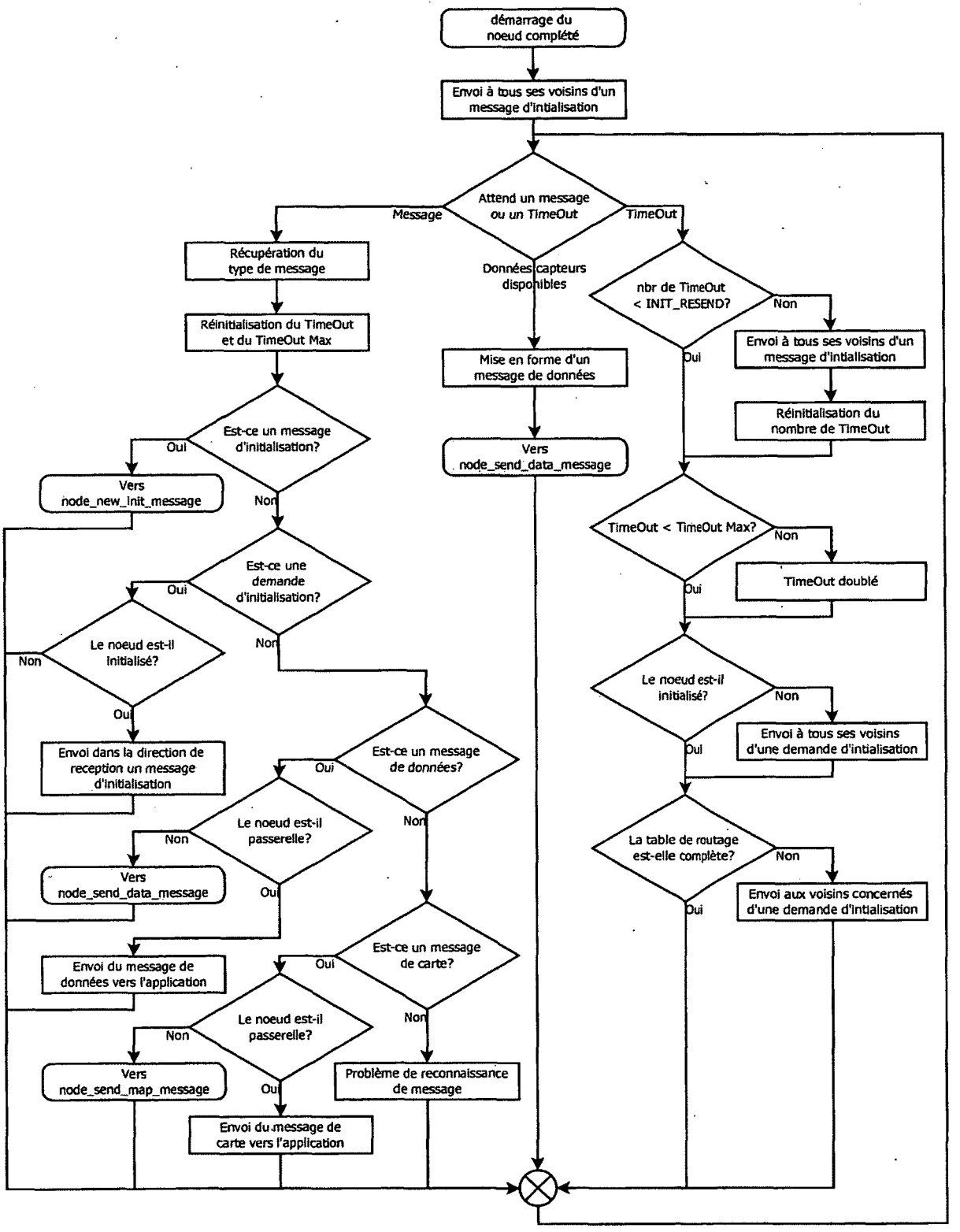


Figure 3.10 Organigramme d'un nœud TileUs.

et maintenir la topologie du réseau, avec les messages de cartographie, Annexe B. Pour l'utilisation des messages de données, suivant la nature des capteurs et leurs résolutions, il est possible d'implémenter certaines méthodes vues au §2.3.3.

3.4 Validation

Une fois que les fonctionnalités et le modèle d'un nœud ont été bien définis, la phase de validation peut commencer. Elle se décompose en trois parties. La première concerne l'implémentation matérielle §3.4.1. La seconde fait intervenir la conception d'un simulateur afin de confirmer le développement des algorithmes à grande échelle, §3.4.2. Enfin, la troisième et dernière partie fait le lien entre le réseau et une future application d'analyse de données, §3.4.3.

3.4.1 Matérielle

La validation matérielle commence avec les choix du micro-contrôleur et du moyen de communication. Ensuite, il est possible de mettre en place l'architecture de communication qui sert de base au réseau de topologie connexion point à point. Avant de charger le code caractérisé par le modèle OSI, dans la section §3.2, il faut analyser les modules internes du micro-contrôleur pour utiliser un maximum de ses fonctionnalités.

Choix de technologies

Pour choisir le micro-contrôleur à utiliser pour le réseau TileUs, il faut d'abord réflechir aux fonctionnalités nécessaires aux différentes phases de fonctionnement :

- Des modules d'acquisition de données : convertisseur Analogique/Digital (ADC), *timer*, et donc un nombre d'entrées/sorties suffisantes.
- Quatre liens de communication distincts et bidirectionnels.

En ce qui concerne le premier critère de sélection, la plupart des micro-contrôleurs possèdent des ADC et des *timers*. Aussi, le nombre d'entrées/sorties est un facteur important. Il faut avoir assez d'entrées/sorties pour permettre d'interfacer un nombre acceptable de capteurs. Cependant, relativement aux quatre liens de communication (USCI), il n'existe pas beaucoup de micro-contrôleurs les proposant. De plus, ceux qui les proposent ont des capacités en mémoire vive, mémoire non volatile et fréquence de calcul plus importantes que nécessaires.

Faire cohexister un micro-contrôleur à faible capacité (mémoire volatile et non-volatile, fréquence de calcul, etc.) avec les besoins en port de communication du réseau TileUs consiste à utiliser des puces externes proposant des interfaces de communication. Ainsi, avec les puces MAX3100, qui proposent une interface SPI/UART, il est possible de transformer un port SPI de l'USCI en quatre ports UART indépendants.

Pour plusieurs raisons, le choix du micro-contrôleur s'est porté sur le MSP430F5529, Figure 3.11. Comme il s'agit d'un micro-contrôleur à deux USCI, l'une est utilisée pour communiquer avec les voisins via les MAX3100, tandis que l'autre permet de réaliser une communication avec un ordinateur, si le nœud est passerelle. De plus, il contient assez de mémoire pour développer un programme conséquent, en laissant une marge pour les évolutions futures. Enfin, il possède un ADC, deux *timers* et des entrées/sorties en nombre suffisant, 80, pour le développement d'un prototype.

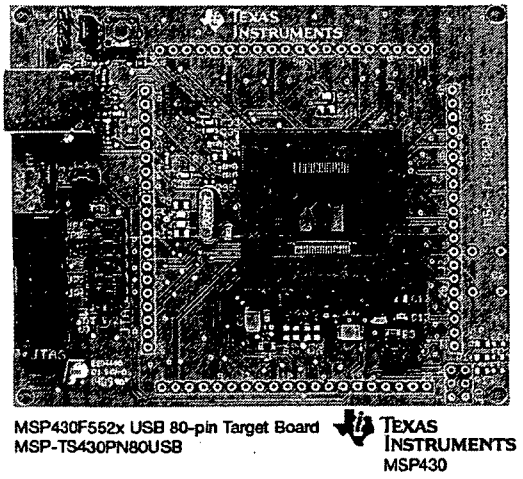


Figure 3.11 Carte de développement pour micro-contrôleur MSP430F5529 (Source [39])

Description de l'architecture de communication

La communication entre deux nœuds se fait par l'intermédiaire d'une communication série asynchrone (UART). Pour cela on utilise des puces MAX3100 qui convertissent un port SPI du micro-contrôleur en un port UART, Figure 3.12. Les convertisseurs SPI/UART sont des MAX3100 reliés au microprocesseur à travers l'USCI A en mode SPI. Le microprocesseur tient le rôle du maître tandis que le rôle de l'esclave est tenu par le convertisseur. L'utilisation de puces telles que les MAX3100 présente plusieurs avantages. Tout d'abord, il est possible de relier quatre MAX3100 à un seul port, ce qui réduit le nombre de ports

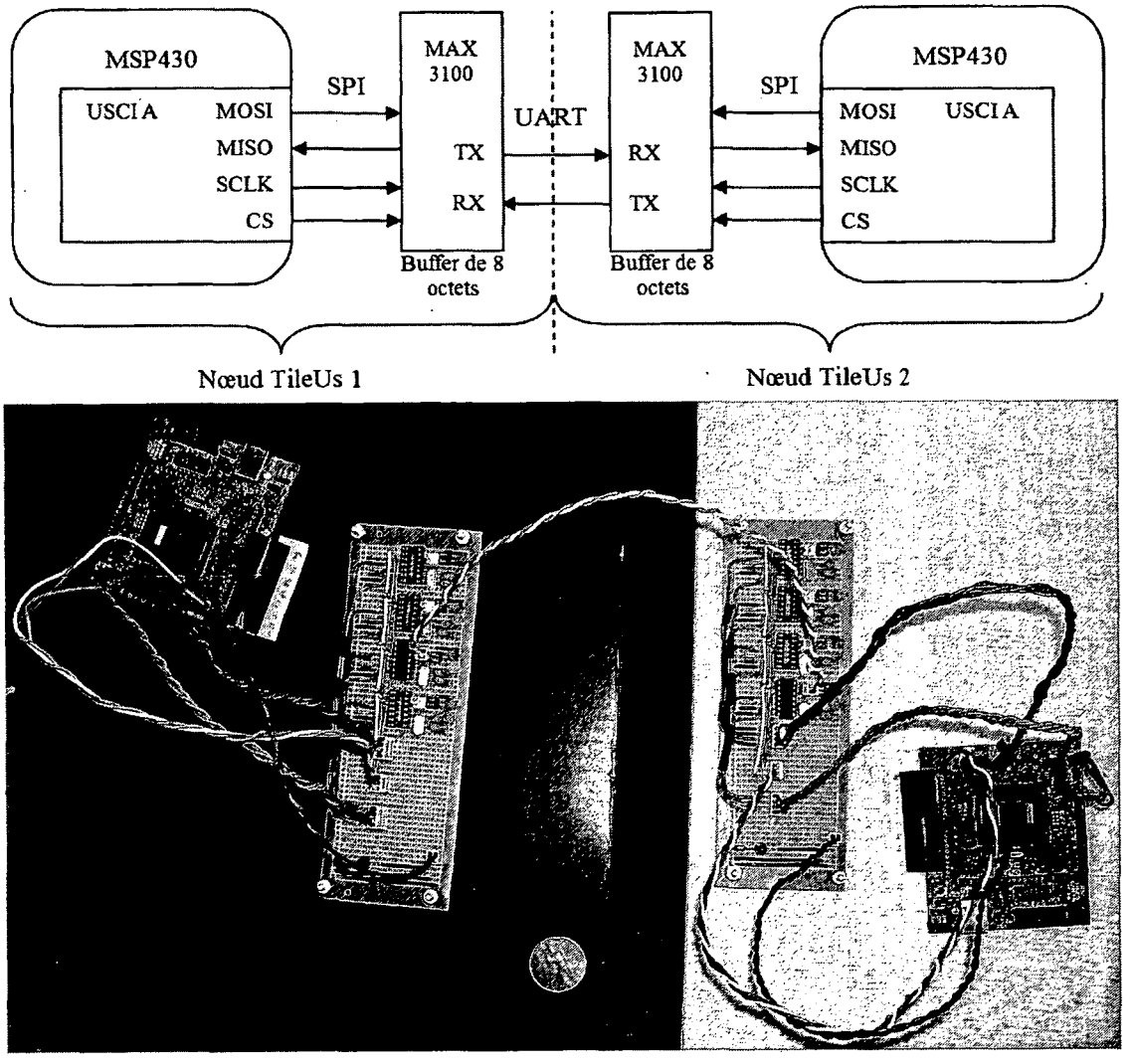


Figure 3.12 Architecture de communication entre deux nœuds.

nécessaires de quatre à un. De plus, un convertisseur MAX3100 possède un *buffer* de 8 octets, ce qui permet d'éviter les débordements dans la communication entre les micro-contrôleurs. Enfin, le fait de convertir le port SPI en UART réduit le nombre de connexions nécessaires pour la communication de quatre fils (MOSI, MISO, SCLK et CS/$\overline{SS}$) à deux fils (TX et RX). L'annexe D reprend le schéma de câblage d'un nœud TileUs.

Pour lire ou écrire un octet utile par l'intermédiaire de ce système il faut suivre le protocole nécessaire au MAX3100[1], c'est-à-dire ajouter un entête d'un octet. Cet entête contient toutes les informations nécessaires pour savoir si l'octet a été correctement transmis et si d'autres octets sont disponibles dans le *buffer*. Le protocole de communication du MAX3100 permet de détecter les erreurs de trames.

Modules internes utilisés

La programmation d'un micro-contrôleur fait aussi intervenir des modules internes de logique câblée. Parmi ceux présents dans le MSP430F5529, le module USCI est bien évidemment utilisé pour faciliter les communications. Ce micro-contrôleur possède également un module de calcul de CRC16. Bien que le calcul du CRC16 puisse se faire logiciellement, le faire matériellement présente plus d'avantages : développement évité et temps de calcul économisé.

Même s'il ne s'agit pas exactement d'un module de fonction interne, la table TLV est utilisée, Tableau 3.9. Elle possède toutes les informations concernant sa fabrication. Cette table contient notamment le numéro de lot du *wafer* ainsi que les coordonnées de la puce. Avec ces informations, il est possible d'extraire un identifiant unique de huit octets pour chaque micro-contrôleur. Cet identifiant est utilisé à la base de l'initialisation du réseau en fournissant un nom unique à chacun des nœuds.

3.4.2 Simulateur de réseau TileUs

Après avoir validé un premier fonctionnement des nœuds du réseau TileUs, le problème de test des algorithmes à grande échelle est vite apparu. Pour y remédier, l'idée est venue de se servir d'un simulateur. Comme aucun simulateur existant peut s'appliquer au réseau TileUs, il a été décidé d'en concevoir un qui répondrait aux besoins fixés. Pour être le plus transparent possible avec le réseau, ce simulateur encapsule le code procédural du nœud dans un objet. Il a donc fallu faire coexister et interagir les programmations procédurales et orientées objets. Le développement du simulateur lui permet de proposer des fonctionnalités et une utilisation similaires à un réseau.

[1]Section *Detailed Description* de la *datasheet* de la puce MAX3100 de MAXIM

Tableau 3.9 Extrait de la table TLV du MSP430F5529.

	Description	Address	Size bytes	'F5529
				Value
Info Block	Info lenght	01A00h	1	06h
	CRC length	01A01h	1	06h
	CRC value	01A02h	2	per unit
	Device ID	01A04h	1	55h
	Device ID	01A05h	1	29h
	Hardware revision	01A06h	1	per unit
	Firmware revision	01A07h	1	per unit
Die Record	Die Record Tag	01A08h	1	0A8h
	Die Record length	01A09h	1	0Ah
	Lot/Wafer ID	01A0Ah	4	per unit
	Die X position	01A0Eh	2	per unit
	Die Y position	01A10h	2	per unit
	Test results	01A12h	2	per unit

Buts du simulateur

Le besoin de réaliser un simulateur pour le projet TileUs nous est apparu très rapidement. Le développement d'un réseau présente plusieurs contraintes. La première est financière, il faut acheter assez de matériel pour s'assurer du bon fonctionnement du réseau. De plus, en début de projet, tous les besoins en matériel ne sont pas forcément bien identifiés et il est alors possible d'acheter des composants qui ne seront plus utilisés dans une version opérationnelle du projet. La seconde contrainte fait intervenir le temps de développement. Le fait de devoir reprogrammer un à un tous les nœuds à la suite d'un changement logiciel prend du temps. La dernière contrainte regroupe les deux précédentes. En effet, il n'est pas envisageable d'acheter autant de débogueurs que de nœuds, ni de les connecter tous à un ordinateur pour connaître l'état de chacun, à chaque instant.

La réalisation d'un simulateur doit pouvoir éviter d'acheter du matériel superflu, aider à développer les algorithmes et faire gagner du temps, mais avant tout, le simulateur doit permettre d'effectuer des tests à grande échelle.

Implémentation

Pour implémenter le simulateur de réseau TileUs, il faut avant tout choisir le langage de programmation, plusieurs sont possibles. Une fois le langage choisi, il faut organiser le projet de simulateur pour qu'il coexiste avec le code présent sur un nœud. De cette manière, les modifications apportées sur un nœud simulé sont aussi apportées au nœud, et vice-versa.

Choix du langage de programmation, l'outil de simulation de réseau TileUs doit être utilisable sur un ordinateur et présenter une interface graphique qui facilite les simulations. Plusieurs choix s'offrent alors dans le développement du simulateur. Chacun des choix possède des avantages. Par exemple, Java permet de lancer le simulateur sur une machine virtuelle et donc de le rendre portable sur différents systèmes d'exploitation. Pourtant, le langage de programmation d'interface graphique qui montre le plus d'avantages repose sur les bibliothèques de Qt : elles sont écrites dans un langage qui englobe le C/C++ ce qui permet de réutiliser le code déployé sur les nœuds dans le simulateur. Cependant, le passage entre la programmation non orientée objet C et celle orientée objet C++ n'est pas immédiat. L'annexe C montre la méthodologie utilisée par le projet TileUs pour pouvoir passer d'un langage à un autre.

Structure du projet, pour que le simulateur et les nœuds du projet TileUs soient développés en parallèle, il faut que la majeure partie du code déployé dans un nœud soit utilisable par le simulateur. De cette manière, le terme «de simulation» peut être employé à juste titre. Pour cela, les points suivants ont été appliqués :

Multithreading, l'utilisation d'une application *multithread* permet de faciliter la simulation d'un réseau. Exécuter un nœud dans un *thread* permet d'instancier autant d'objets nœuds que de nœuds dans le réseau et de lancer l'exécution de chacun des nœuds dans des *threads* indépendants. Dans ce cas, chaque nœud simulé possède sa propre pile d'appel. Toutefois, les *threads* et les nœuds partagent tous la mémoire virtuelle de l'application, il faut donc faire attention lors des accès mémoires. Par analogie avec un vrai réseau, le simulateur est développé de sorte qu'un *thread* ne possède pas de variable partagée avec un autre.

Ajouts d'interfaces, le partage de code entre le simulateur et les nœuds commence par son analyse puisqu'il faut identifier les parties qui peuvent être implémentées sur les deux cibles. C'est ainsi que les parties de code sont identifiées : elles ont toutes un rapport avec les fonctionnalités matérielles d'un nœud :

- la présence d'une table TLV (***Tag-Length-Value***) qui recense des informations uniques sur chaque puce.
- la transmission de données entre nœuds et le module physique de calcul de CRC.

De cette manière, les *drivers* sont isolés par rapport au code propre du nœud TileUs. De plus, les accès aux fonctionnalités liées aux *drivers* sont regroupés dans une in-

terface de type logicielle-matérielle qui se trouve dans le fichier `communication.h` et qui permet au simulateur de ré-implémenter les fonctions de l'interface. Par ailleurs, pour éviter les problèmes, lors des différentes compilations C ou C++, le fichier `communication.c` comporte des directives de compilation conditionnelle basées sur la présence du symbole `__SIMULATOR` défini dans le `make` du simulateur.

Vers la programmation orientée objet, pour faciliter la création d'un simulateur de réseau TileUs, un nœud est considéré comme un objet. Cependant, le code d'un nœud n'est pas orienté objet puisqu'il est écrit en C. Pour cela, il faut qu'un nœud devienne un objet. Avant tout, il faut préciser qu'un nœud est constitué par la structure suivante :

```
1  // node.c
2  struct node
3  {
4      struct routing_table routing_table;
5      uint64_t id_number;
6      enum IDENTITY_init id_init;
7      enum GATEWAY gateway;
8      enum DIRECTION last_buffer_checked;
9      void * max[NEIGHBOR_NBR];
10     int waitTime;
11     int initResend;
12     int timeStamp;
13     struct linked_attr message_list;
14 };
```

Avec le langage C++, il est possible de transformer une structure C en un objet. Pour cela, il faut créer une classe qui hérite de la structure à transformer. La classe `Tile` en est un exemple puisque l'objet `Tile` possède tous les attributs déclarés dans la structure `node`. Il est toutefois possible d'y ajouter de nouveaux attributs. Le §C.3 explique comment garder la notion d'objet en utilisant des fonctions C.

```
1  // tile.h
2  #include "../Hardware/node.h"
3
4  class Tile: public node
5  {
6  public:
7      Tile(GATEWAY is_gate);
8      void init(void * myTileObject);
9      ...
```

```
10 };
11
12 // tile.cpp
13 Tile::Tile(GATEWAY is_gate)
14 {
15     this->gateway = is_gate;
16 }
17
18 void Tile::init(void *myTileObject)
19 {
20     node_init(this, myTileObject);
21 }
```

Ré-implémentation des fonctionnalités matérielles, dans le simulateur, il est possible de ré-utiliser le code d'un nœud, en dehors des fonctions en lien avec les fonctionnalités matérielles. Pour cette raison, il faut les ré-implémenter.

Table TLV, pour pallier le manque de table TLV dans le simulateur, Tableau 3.9, l'application distribue un identifiant à tous les nœuds simulés et en garantit l'unicité.

Transmission de données, les bibliothèques de Qt présentent un autre avantage que le fait d'être écrites en C++. En effet, elles implémentent des fonctionnalités que le C++ n'offre pas. De plus, Qt propose une simplification du patron de conception MVC (Modèle Vue Contrôleur) en MV. Cette simplification est permise grâce aux signaux/*slots* utilisés pour la communication entre objets [36]. La communication entre objets nœuds s'effectue par l'entremise de ces mécanismes.

Pour simplifier, les signaux peuvent être comparés à des niveaux d'interruption qui déclenchent l'exécution de méthodes dites *slots*. À partir d'un signal, il est possible de déclencher plusieurs *slots* et même d'autres signaux. De plus, les types de connexion entre les signaux et les *slots* sont paramétrables. Pour la communication entre les nœuds, le simulateur met en œuvre des connexions de type directes qui permettent d'exécuter les *slots* lorsque le signal est émis. D'autres types de connexions sont fournis par Qt mais ne sont pas pertinents dans le cadre de la communication.

Pour transmettre des données entre les nœuds, donc entre les *threads*, deux possibilités sont envisageables. La première consiste à partager une variable

entre deux *threads* et un signal en déclenche la lecture. Cette solution n'est pas utilisée car elle implique une gestion d'accès à une mémoire partagée. La deuxième solution permet d'émettre des paramètres avec le déclenchement du signal. De cette manière, le partage de données peut se faire par copie et non par référence.

Utilisation du simulateur

Maintenant que le simulateur est opérationnel, son utilisation et ses fonctionnalités peuvent être présentées. Mais avant cela, il faut tout d'abord rappeler que la création de ce simulateur a pour but d'aider au développement des algorithmes présents dans le réseau, afin de réduire les coûts de conception et d'arriver à une solution fonctionnelle dès la première itération du prototype.

Création d'un réseau, la création d'un réseau commence avec la sélection de trois paramètres, Figure 3.13. Le premier paramètre «Projet vide» définit, si la case est cochée, la création d'une feuille vide dont les dimensions correspondent aux paramètres «longueur du reseau» et «largeur du reseau». Il faut donc ajouter manuellement les nœuds un à un. Dans le cas contraire, le projet est formé d'un réseau rectangulaire dont les dimensions correspondent aux mêmes paramètres.

Une fois la feuille de projet créée, il est possible de modifier la morphologie du réseau, soit en ajoutant/supprimant des nœuds, soit en créant/détruisant des connexions entre nœuds, ou encore en définissant le ou les nœud(s) passerelle(s), Figure 3.14. De cette manière, le réseau peut avoir une forme tout à fait quelconque et peut correspondre à n'importe quel profil.

Ouverture/enregistrement d'un réseau existant, pour simuler un réseau TileUs, il n'est pas obligatoire de créer un projet selon la procédure expliquée auparavant. Effectivement, il est possible d'importer un projet déjà existant dans le simulateur, tout comme il

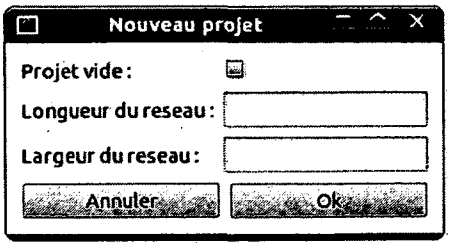


Figure 3.13 Fenêtre de création d'un nouveau projet TileUs.

Figure 3.14 Fenêtre de modification d'un nœud après avoir fait un clic droit sur la zone où il se trouve.

est possible de le sauvegarder. Pour cela, la morphologie du réseau TileUs est enregistrée dans un fichier au format *.xml*. Cette sauvegarde du réseau prend uniquement en compte sa morphologie. Ainsi, lorsqu'une sauvegarde a lieu après l'initialisation du réseau, son ouverture donne ce même réseau, mais dans l'état d'avant initialisation.

Voici un exemple de réseau TileUs représenté au format *xml*, il correspond au réseau de la Figure 3.15 :

```
1  <TileUs>
2      <properties width="3" mTime="25" height="4"/>
3      <Tile x="0" disconnected="0" y="0" shutdowned="0" gateway="0" present="0"/>
4      <Tile x="0" disconnected="0" y="1" shutdowned="0" gateway="0" present="1"/>
5      <Tile x="0" disconnected="1" y="2" shutdowned="0" gateway="0" present="1"/>
6      <Tile x="0" disconnected="0" y="3" shutdowned="0" gateway="0" present="0"/>
7      <Tile x="1" disconnected="0" y="0" shutdowned="0" gateway="1" present="1"/>
8      <Tile x="1" disconnected="0" y="1" shutdowned="0" gateway="0" present="1"/>
9      <Tile x="1" disconnected="12" y="2" shutdowned="0" gateway="0" present="1"/>
10     <Tile x="1" disconnected="7" y="3" shutdowned="0" gateway="0" present="1"/>
11     <Tile x="2" disconnected="2" y="0" shutdowned="0" gateway="0" present="1"/>
12     <Tile x="2" disconnected="0" y="1" shutdowned="0" gateway="0" present="1"/>
13     <Tile x="2" disconnected="0" y="2" shutdowned="0" gateway="0" present="1"/>
14     <Tile x="2" disconnected="0" y="3" shutdowned="0" gateway="0" present="0"/>
15 </TileUs>
```

La représentation *xml* d'un réseau TileUs commence toujours par la balise `<TileUs>` qui est la balise principale. Par la suite, la description possède deux types d'éléments : un élément qui contient les propriétés générales du sol intelligent `<properties>`, et les éléments `<Tile>` contenant la description de chaque dalle du sol.

L'élément *<properties>*, comme expliqué ci-dessus, cet élément contient les propriétés générales du projet, à savoir la largeur `<width>` et la longueur `<height>` de la feuille du projet. Cet élément contient aussi la valeur qui régle le délai de transmission d'un message entre deux nœuds `<mTime>`.

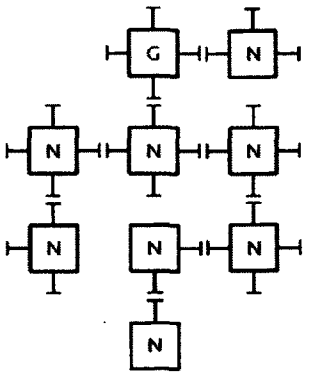


Figure 3.15 Réseau TileUs correspondant à la description *xml*.

L'élément *<Tile>*, cet élément constitue la base même du sol intelligent. En effet, il contient les attributs <x> et <y> qui définissent la position relative du nœud dans la feuille du projet. Il possède aussi les attributs suivants qui permettent de caractériser un nœud du sol intelligent :

- `<present>` : de valeur 0 si le nœud est présent dans le sol, 1 sinon.
- `<shutdowned>` : de valeur 0 si le nœud est actif, 1 dans le cas où il simule un nœud défaillant.
- `<gateway>` : de valeur 1 si le nœud est considéré comme *gateway*, à savoir pont de communication avec l'application serveur, 0 sinon.
- `<disconnected>` : de valeur appartenant à l'intervalle $[0;15]$, indiquant si une communication est rompue ou non :

$$\begin{aligned} disconnected = {} & NORTH\ déconnecté \times 2^3 + WEST\ déconnecté \times 2^2 \\ & + SOUTH\ déconnecté \times 2^1 + EAST\ déconnecté \times 2^0 \\ = {} & NORTH\ déconnecté \times 0b1000 + WEST\ déconnecté \times 0b0100 \\ & + SOUTH\ déconnecté \times 0b0010 + EAST\ déconnecté \times 0b0001 \end{aligned} \tag{3.1}$$

Démarrage du réseau, une fois que le réseau possède la forme désirée et au moins un nœud passerelle, le démarrage du réseau peut se faire. Pendant la phase de fonctionnement, un nœud affiche des informations nécessaires pour connaître l'état dans lequel il se trouve :

- un nœud dont l'écriture est bleue veut dire qu'il s'agit d'un nœud passerelle
- un «N» rouge, signifie que le nœud n'est pas encore initialisé
- un «X» gris représente un nœud défaillant

- un «init» vert montre que le nœud est initialisé et qu'il n'a relayé aucun message
- lorsqu'un nœud affiche un nombre, il fait référence au numéro d'émission du message en transit. Cela permet de retracer simplement le chemin effectué par un message.

Avant le développement d'applications client/serveur, pour symboliser la transmission de l'information contenue dans les messages à l'application serveur, le simulateur stocke tous les messages parvenant aux nœuds passerelles dans le fichier *log.txt*. Comme expliqué auparavant, le réseau TileUs peut transmettre deux types de messages, l'un dit de cartographie et l'autre dit de données. Voici le formalisme selon lequel ils sont stockés :

```
1 MAP    Node ID : 16,   Direction : 2,   Neighbor ID : 12,   Hop : 5,   TimeStamp : 1
2 DATA   Node ID : 12,   data : 65,       TimeStamp : 3
```

Les mots clefs *MAP* et *DATA* définissent la nature du message. Quant à *Node ID* et *TimeStamp*, ils mentionnent respectivement l'identifiant unique du nœud à l'origine du message et le numéro d'émission du message, ils sont tous communs aux deux types de messages. Les messages de données contiennent en plus le champ *data* indiquant la valeur lue sur le capteur. Pour ce qui est des paramètres spécifiques à un message de cartographie, *Direction* fait référence à la direction dans laquelle se situe le nœud voisin dont l'identifiant est *Neighbor ID* distant de *Hop* nœuds du nœud passerelle. Par la suite, au §3.4.3, ces messages sont mis sous format *xml* pour faciliter leur manipulation par d'autres applications.

Optimisations/Développements futurs

Désormais, un premier prototype de réseau et de simulateur de réseau, intégrable dans un sol intelligent, est fonctionnel. Comme il s'agit de la première itération du prototype, des améliorations peuvent être apportées.

Statistiques, évidemment, le simulateur de réseau TileUs peut être optimisé ou se voir rajouter de nouvelles fonctionnalités. Par exemple, il pourrait être intéressant de pouvoir extraire des statistiques quant à l'utilisation du réseau. En effet, dans le fichier transmis à l'application serveur, la seule donnée qui n'est pas issue des capteurs provient du *TimeStamp*, c'est-à-dire du numéro d'émission des messages qui permet uniquement d'ordonner ces émissions dans le temps. Des données supplémentaires, comme le nombre de messages émis comprenant les messages formés et relayés, le nombre de messages reçus, perdus, etc. peuvent servir de base à l'analyse des performances du réseau. En somme, ces nouvelles données peuvent permettre de comparer différentes morphologies, pour une même topolo-

gie. Par conséquent, il est possible de trouver un compromis entre la taille et la forme du réseau et le nombre de nœuds passerelles. Par ailleurs, avoir accès au chemin qu'a suivi le message peut être intéressant pour analyser la congestion du réseau dans un cas de forte utilisation, afin d'améliorer l'algorithme de routage de l'information. Pour le moment, les contraintes de temps du réseau TileUs et du simulateur de réseau ne sont pas liées. Il serait donc avantageux pour la simulation d'introduire la notion de temps, en faisant notamment un lien entre le temps réel et le temps simulé de transmission d'un message entre deux nœuds.

Implémentation et topologie du réseau, actuellement, le simulateur fonctionne sur la topologie d'un réseau maillé comportant des connexions point à point. La section 3.1 présente d'autres topologies possibles pour le réseau de capteur TileUs. Le fait de pouvoir choisir entre plusieurs topologies permettrait une plus grande appréciation des avantages de chacune. Cependant, en ce qui concerne un réseau comportant un ou plusieurs bus, il faut concevoir un modèle de bus adéquat. En effet, lorsque la simulation est effectuée sur un ordinateur, qu'il soit uni-cœur ou multi-cœurs, il faut être sûr que l'accès au médium peut être multiple, c'est-à-dire qu'au moins deux nœuds puissent y accéder en même temps, pour que l'algorithme gérant la contention soit assurément fonctionnel.

D'un autre point de vue, l'amélioration du simulateur permet l'amélioration du réseau TileUs. Pendant le développement du simulateur quelques points importants concernant l'implémentation physique sont apparus. Un premier touche la mise en veille d'un nœud. On se rappelle que dans la première implémentation du simulateur, le programme utilisé par tous les nœuds comportait une boucle infinie et que son utilisation avait provoqué une surcharge du processeur de l'ordinateur. Dans le cas de l'implémentation physique, cela provoquerait principalement une plus grande consommation d'énergie. Ainsi, dans un souci de réduction de la consommation électrique, il faudrait soit remplacer l'utilisation de la boucle infinie par l'utilisation d'un plus grand nombre de vecteurs d'interruption, soit introduire un mécanisme de mise en veille par zone inactive du plancher, tout en gardant un chemin de transmission vers un nœud passerelle.

Données capteurs, avec cette itération du simulateur de réseau TileUs, l'acquisition d'une donnée à partir d'un nœud capteur se fait par un simple clic sur le nœud. Dans cette configuration, il n'est pas possible d'actionner simultanément deux nœuds capteurs. De ce fait, il pourrait être pertinent de laisser à l'utilisateur le choix de la forme qui active les capteurs. En revanche, il est tout à fait possible de choisir des capteurs analogiques pour

le sol intelligent. Il faudrait alors implémenter un moyen de choisir l'amplitude de la force de pression et même son profil dans le temps.

3.4.3 Liaison Réseau/Ordinateur

Avec la première itération d'un prototype de réseau TileUs, il est possible de lire et d'analyser les messages issus des capteurs. Après avoir transité dans le réseau et avoir atteint un nœud passerelle, le message est transmis à une application dans laquelle s'effectue l'analyse des données. Cette application correspond à l'étape 3 du modèle de conception d'un sol intelligent, §2.3.3.

La liaison entre le réseau et l'application se fait sur le schéma de communication client/-serveur par l'intermédiaire d'une *socket* TCP/IP. Chaque nœud passerelle du réseau est connecté à un ordinateur sur lequel tourne un serveur. Ce serveur lit les messages provenant du réseau sur un port COM et l'envoie via une *socket* TCP au client qui analyse les données. De cette manière, il existe une application client par réseau TileUs connecté à autant d'applications serveurs reliées aux nœuds passerelles. On applique le même principe avec le simulateur à la différence que les serveurs sont directement intégrés dans le code. Il n'y a pas besoin d'exécuter une application recevant les données venant du simulateur.

Au niveau du format des données transmises, le serveur récupère les messages sous la forme d'un texte rappelé ci-dessous.

```
1 MAP    Node ID : 16,   Direction : 2,   Neighbor ID : 12,   Hop : 5,   TimeStamp : 1
2 DATA   Node ID : 12,   data : 65,       TimeStamp : 3
```

Pour classer ces données temporellement, le serveur ajoute aux messages la date d'arrivée du message. Puis, dans un souci de compatibilité avec d'autres types d'applications, le serveur transmet les messages en format *xml* au client.

```
1 <MAP ID="16" dir="2" NID="12" Hop="5" TimeStamp="1" Hour="Sat May 19 15:21:51 2012"/>
2 <DATA ID="16" data="65" TimeStamp="1" Hour="Sat May 19 15:24:51 2012"/>
```

CHAPITRE 4
CONCLUSION

4.1 Résumé

Ce projet de maîtrise avait pour objectif d'étudier la faisabilité d'un système permettant la localisation par le sol d'une personne dans un espace intelligent, domestique ou public, dans le but de l'intégrer dans l'appartement du laboratoire DOMUS.

Les recherches documentaires ont mis en lumière différentes possibilités quant à la constitution d'un sol intelligent. Suivant un ordre chronologique, les sols intelligents ont d'abord été pensés afin d'identifier une personne par la décomposition et l'analyse de la force de pression exercée sur le sol pendant la marche.

Ces premiers systèmes étaient efficaces pour une simple dalle du sol. En revanche, lorsqu'ils ont été appliqués à des pièces entières, leurs coûts d'installation — en lien avec les capteurs — et leurs maintenances ont atteint des coûts exorbitants. Dès lors, la phase suivante consiste à la création d'un sol faisant uniquement de la détection de présence. À l'heure actuelle, le coût d'installation reste encore notable du fait des connaissances nécessaires dans le système. Ce projet s'est donc principalement orienté vers la réduction possible des coûts et principalement celle de l'installation.

L'identification des trois étapes de réalisation du sol intelligent :

1. le choix et l'étude du ou des capteurs, §2.3.1
2. le choix et la mise en place de la topologie du réseau de capteurs, §2.3.2
3. le choix et la conception de la ou des techniques d'analyse des données, §2.3.3

a permis de déterminer les plus facilitantes. Il s'agit de l'étape n° 2 qui met en place la topologie de réseau du sol intelligent. Pour la réalisation du réseau de capteurs TileUs, comportant des connexions point à point, les contraintes suivantes ont été fixées et rencontrées :

- un nœud du réseau consiste en une boîte noire de forme carrée qu'il faut connecter sur chacun de ses côtés, de la même manière que l'on confectionne un puzzle et dont le but final est d'être intégré directement dans chaque dalle du sol intelligent

- le réseau de capteurs TileUs doit posséder un unique «bouton», celui-ci permettant d'activer l'alimentation en électricité du réseau

La seule différence d'installation avec un sol typique est de relier toutes les dalles entre elles avec un connecteur spécifique et de raccorder l'alimentation électrique.

Le fait de simplifier l'installation physique du réseau a ajouté une complexité au réseau. En effet, il a été nécessaire de concevoir plusieurs algorithmes pour le bon fonctionnement du réseau :

- un algorithme de configuration initialisant tous les nœuds
- un algorithme de routage, acheminant l'information d'un nœud vers un ordinateur
- un algorithme de maintien du réseau pour prendre en compte les diverses défaillances possibles ainsi que l'ajout de nœuds

Pour diminuer les coûts de développement du projet, la conception de ces algorithmes a nécessité la réalisation d'un simulateur de réseau. Il a été possible de développer les algorithmes avec un nombre restreint de nœuds pour valider leurs fonctionnements physiques, tout en validant un fonctionnement avec un nombre de nœuds plus important grâce au simulateur.

Par ailleurs, une étude théorique concernant différentes topologies de réseau a été présentée dans la section 3.1. Ceci dans le but de faire une comparaison des topologies applicables au projet de sol intelligent. Cette étude aborde les points pouvant poser des problèmes et propose des amorces de solutions qui servent de base au développement du prototype de réseau.

4.2 Contributions

La réalisation de ce projet a donné lieu á, plusieurs contributions. Les principales sont :

- l'extraction des trois étapes, relativement indépendantes, nécessaires à la réalisation d'un sol intelligent, mises au jour dans la revue de littérature. Cela permet à la conception d'un sol intelligent de se faire en parallèle pour que chaque branche de développement se rejoignent à la fin de la conception.
- la réalisation générique de la deuxième étape, consistant en la mise en place du réseau, permet une modularité avec les autres étapes. Ceci rend possible la réutilisation

et la modification de chaque étape de réalisation du réseau TileUs, afin de répondre au mieux aux besoins et aux évolutions futurs.

- la conception d'un prototype et d'un simulateur de réseau de capteurs TileUs, validant les algorithmes de configuration, de routage et de maintenance.

4.3 Travaux futurs

L'identification des trois étapes nécessaires à la réalisation d'un sol intelligent issue de la revue de littérature du chapitre 2 peut servir aussi bien pour les futurs projets axés sur le choix et l'étude des capteurs à utiliser que sur les techniques d'analyse des données. Pour que le prototype de ce projet de recherche sur le sol intelligent soit fonctionnel, l'étape n° 1 a été implémentée avec un simple interrupteur dont le changement de niveau déclenche l'envoi d'un message de données. En ce qui concerne l'étape n° 3, l'application sur l'ordinateur écrit les données dans un fichier, dans un but de stockage sans autre interprétation que celle faite par le développeur. Ainsi, pour que le prototype soit entièrement fonctionnel, il faudrait compléter les étapes n° 1 et n° 3. C'est-à-dire développer une chaîne d'acquisition de données et relier une interface graphique au réseau, comme au simulateur, à travers une *socket*.

Par ailleurs, le code présent dans un nœud du réseau peut être optimisé, afin de réduire la consommation électrique d'un nœud et donc son coût d'utilisation. Pour le moment, les nœuds sont en attente active de messages provenant d'un voisin ou de données issues d'un capteur. Un moyen de réduire la consommation d'énergie liée à l'attente active serait d'intégrer une mise en veille partielle du réseau, dépendamment de son utilisation. Pour y parvenir, il faudra être sûr qu'en désactivant une partie du réseau, une autre ne se retrouvera pas isolée. On pourrait également utiliser les fonctionnalités de réveil du microcontrôleur associé à la réception de données ou bien à l'activation du capteur.

En ce qui concerne le simulateur de réseau TileUs, les optimisations et travaux futurs sont abordés au §3.4.2.

La section 3.1 fait mention d'autres implémentations possibles pour le réseau de capteurs. Par la suite, il pourrait être intéressant de les réaliser afin qu'une comparaison approfondie en résulte. Celle-ci permettrait d'améliorer les performances du réseau en fonction de l'espace intelligent dans lequel il devra être intégré.

ANNEXE A

Description ASN.1 des messages TileUs

```
1  -- Variable type
2  UINT8 ::= OCTET STRING(SIZE(1));
3  UINT16 ::= OCTET STRING(SIZE(2));
4  UINT64 ::= OCTET STRING(SIZE(8));
5
6  ----------------------------------------------------------------------------
7  -- ID of all type of message supported by a TileUs node
8  MESSAGE_TYPE ::= ENUMERATED{
9      INIT_MESSAGE(0),
10     INIT_REQUEST(1),
11     DATA_MESSAGE(2),
12     MAP_MESSAGE(3)
13 }
14
15 -- Direction of each neighbor potential of a TileUs node
16 ORIGIN_DIRECTION ::= ENUMERATED{
17     EAST(0),
18     SOUTH(1),
19     WEST(2),
20     NORTH(3),
21     ALL(4)
22 }
23
24 -- ID makes a unique message
25 MESSAGE_ID ::= SEQUENCE{
26     ORIGIN_NODE_ID UINT64,
27     TIMESTAMP  UINT8
28 }
29
30 ----------------------------------------------------------------------------
31 -- Body of an initialization message
32 BODY_INIT_MESSAGE ::= SEQUENCE{
33     ORIGIN_NODE_ID UINT64,
34     HOP    UINT8  -- Number of nodes to traverse to reach the Gateway node
35 }
36
37 -- Body of an initialization request which is empty
38 BODY_INIT_REQUEST ::={
39 }
40
41 -- Body of a map message that is sent to the Gateway node after filling or
42 --  modified a row in the routing table
43 BODY_MAP_MESSAGE ::= SEQUENCE{
44     NODE_ID        MESSAGE_ID,
45     ORIGIN         ORIGIN_DIRECTION,
46     NEIGHBOR_NODE_ID  UINT64,
47     NEIGHBOR_BEST_HOP UINT8
```

```
48 }
49
50 -- Body of a data message which can have data which size between 0 to 255
51 BODY_DATA_MESSAGE ::= SEQUENCE{
52     DATA_SIZE  UINT8,
53     NODE_ID    MESSAGE_ID,
54     DATA       OCTET STRING(SIZE(DATA_SIZE))
55 }
56
57 -- Generic name for any type of message body
58 BODY_MESSAGE ::= CHOICE{
59     INIT_MESSAGE   BODY_INIT_MESSAGE,
60     INIT_REQUEST   BODY_INIT_REQUEST,
61     MAP     BODY_MAP_MESSAGE,
62     DATA       BODY_DATA_MESSAGE
63 }
64
65 ------------------------------------------------------------------------------
66 -- Structur of a TileUs message with headers of the three first OSI layers
67 MESSAGE ::= SEQUENCE{
68     TYPE    MESSAGE_TYPE,
69     BODY    BODY_MESSAGE,
70     CRC UINT16,
71     ETX     UINT8  -- End of text byte in ASCII Table
72 }
```

ANNEXE B

Conception du plan du réseau

Lorsqu'un nœud passerelle reçoit un message de cartographie, il l'interprète et le transmet à l'application via une liaison série. Dans le cas d'un réseau de 3x3 nœuds TileUs simulé, Figure B.1, l'application reçoit les données suivantes :

```
1  MAP    Node ID : 4,  Direction : 0, Neighbor ID : 7, Hop : 0,  TimeStamp : 0
2  MAP    Node ID : 4,  Direction : 1, Neighbor ID : 5, Hop : 0,  TimeStamp : 1
3  MAP    Node ID : 4,  Direction : 2, Neighbor ID : 1, Hop : 0,  TimeStamp : 2
4  MAP    Node ID : 7,  Direction : 2, Neighbor ID : 4, Hop : 1,  TimeStamp : 0
5  MAP    Node ID : 5,  Direction : 3, Neighbor ID : 4, Hop : 1,  TimeStamp : 0
6  MAP    Node ID : 1,  Direction : 0, Neighbor ID : 4, Hop : 1,  TimeStamp : 0
7  MAP    Node ID : 8,  Direction : 3, Neighbor ID : 7, Hop : 2,  TimeStamp : 0
8  MAP    Node ID : 2,  Direction : 3, Neighbor ID : 1, Hop : 2,  TimeStamp : 0
9  MAP    Node ID : 7,  Direction : 1, Neighbor ID : 8, Hop : 3,  TimeStamp : 1
10 MAP    Node ID : 1,  Direction : 1, Neighbor ID : 2, Hop : 3,  TimeStamp : 1
11 MAP    Node ID : 8,  Direction : 2, Neighbor ID : 5, Hop : 2,  TimeStamp : 1
12 MAP    Node ID : 2,  Direction : 0, Neighbor ID : 5, Hop : 2,  TimeStamp : 1
13 MAP    Node ID : 6,  Direction : 3, Neighbor ID : 5, Hop : 2,  TimeStamp : 0
14 MAP    Node ID : 5,  Direction : 1, Neighbor ID : 6, Hop : 3,  TimeStamp : 1
15 MAP    Node ID : 5,  Direction : 0, Neighbor ID : 8, Hop : 3,  TimeStamp : 2
16 MAP    Node ID : 2,  Direction : 1, Neighbor ID : 3, Hop : 4,  TimeStamp : 2
17 MAP    Node ID : 8,  Direction : 1, Neighbor ID : 9, Hop : 4,  TimeStamp : 2
18 MAP    Node ID : 5,  Direction : 2, Neighbor ID : 2, Hop : 3,  TimeStamp : 3
19 MAP    Node ID : 9,  Direction : 2, Neighbor ID : 6, Hop : 3,  TimeStamp : 0
20 MAP    Node ID : 3,  Direction : 0, Neighbor ID : 6, Hop : 3,  TimeStamp : 0
21 MAP    Node ID : 6,  Direction : 0, Neighbor ID : 9, Hop : 4,  TimeStamp : 1
22 MAP    Node ID : 6,  Direction : 2, Neighbor ID : 3, Hop : 4,  TimeStamp : 2
23 MAP    Node ID : 9,  Direction : 3, Neighbor ID : 8, Hop : 3,  TimeStamp : 1
24 MAP    Node ID : 3,  Direction : 3, Neighbor ID : 2, Hop : 3,  TimeStamp : 1
```

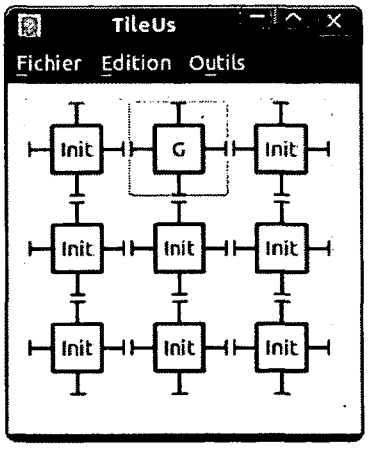
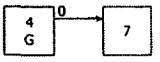
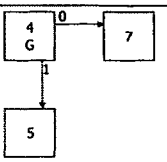
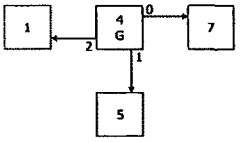
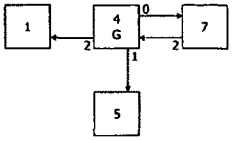
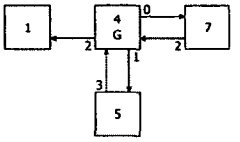
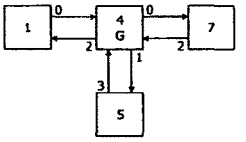
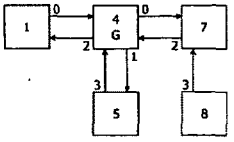
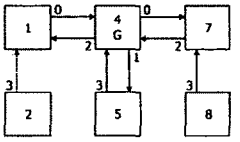
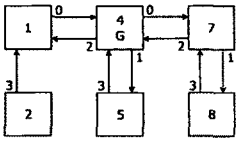
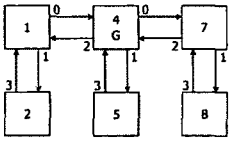
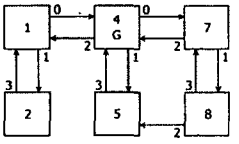
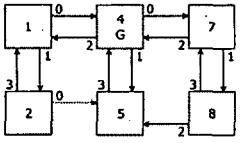
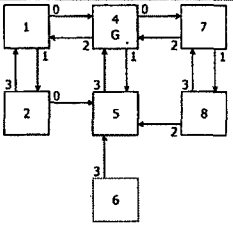
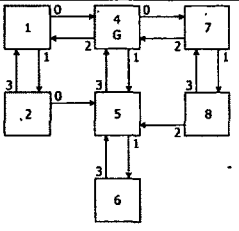
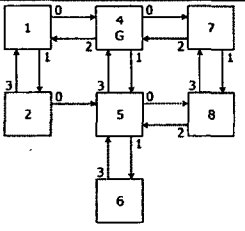
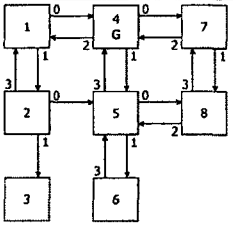
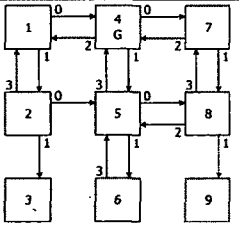
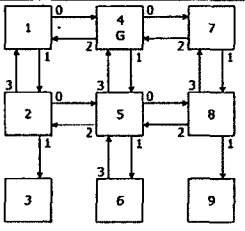
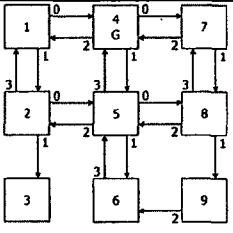
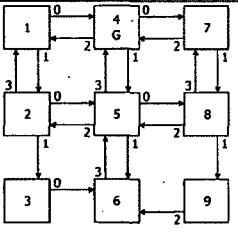
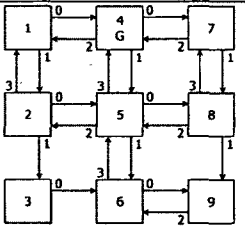
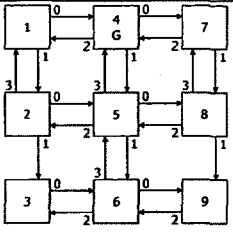
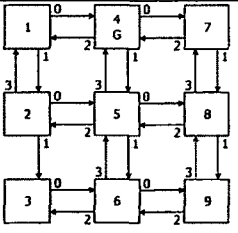
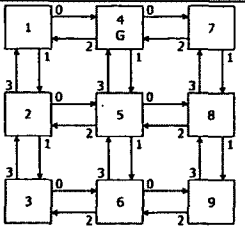


Figure B.1 Réseau TileUs 3x3.

À partir de ces données, il est possible de reconstituer la cartographie du réseau en place. Comme les messages arrivent par ordre chronologique, il est normal de les interpréter dans leur ordre d'arrivée. Les étapes de conception du plan pour ce réseau sont détaillées dans le Tableau B.1. Le nœud n° 4 étant le nœud passerelle qui transmet les messages à l'application apparaît en premier dans la liste des messages de cartographie. Il sert donc de point de départ à la conception. Dans cet exemple, les dalles sont considérées comme orientées. Ceci peut être obtenu en fabriquant les dalles de la même manière que l'on construit un puzzle. Tous les nœuds ont la même référence dans les directions. Ici, la direction 0 pointe vers la droite du nœud.

Tableau B.1 Détail des étapes de conception du plan du réseau.

 I.1	 I.2	 I.3
 I.4	 I.5	 I.6
 I.7	 I.8	 I.9
 I.10	 I.11	 I.12
 I.13	 I.14	 I.15
 I.16	 I.17	 I.18
 I.19	 I.20	 I.21
 I.22	 I.23	 I.24

ANNEXE C

Mixer les langages C et C++

Cette annexe explique la méthode utilisée pour mixer les codes C et C++ dans le projet TileUs [30]. L'explication se fait en suivant l'exécution d'une partie du programme du simulateur.

C.1 Inclure un fichier d'entête C dans du code C++

Dans un premier temps, il faut que le compilateur puisse faire la différence entre le code C et C++. Cela se fait par l'ajout au début de chaque entête d'un fichier C du code suivant :

```
1 // node.h
2 #ifndef NODE_H_
3 #define NODE_H_
4
5 #include "communication.h"
6
7 #ifdef __cplusplus
8 extern "C" {
9 #endif
```

La ligne 8 indique au compilateur que les fonctions déclarées entre les accolades sont en langage C. Dans le cas où l'on veut inclure des fichiers d'entêtes, il faut faire attention à la place de l'inclusion. Pour un fichier d'entête qui peut aussi bien servir pour du code C que du code C++, il faut le placer avant le « `extern "C" {` ». C'est le cas pour le fichier `communication.h` qui est utilisé par du code C++ dans le simulateur, mais par du code C dans les nœuds. Le symbole `__cplusplus` est un symbole défini par tous les compilateurs C++. Ainsi, les compilateurs C ne considèrent pas l'indication `extern "C"`.

Il faut ensuite fermer l'accolade ouverte à la ligne 8 en ajoutant le code suivant à la fin du fichier d'entête :

```
1 // node.h
2 #ifdef __cplusplus
3 }
4 #endif /* extern "C"*/
5
6 #endif /* NODE_H_ */
```

C.2 Appel d'une fonction C à partir de code C++

Dans l'exécution du programme, le premier changement de langage se fait en passant du C++ au C, au moment de l'initialisation d'un nœud.

```
// tileobject.cpp
void TileObject::init(void)
{
...
    this->myTile->init(this);
...
}
```

Il se fait par l'intermédiaire des méthodes de la classe `Tile`.

```
tile.h
#include "../Hardware/node.h"

class Tile: public node
{
    void init(void * myTileObject);
};

//tile.cpp
void Tile::init(void *myTileObject)
{
    node_init(this, myTileObject);
}
```

Les méthodes de la classe `Tile` possèdent toutes le paramètre « `void * myTileObject` » qui permet de transmettre un pointeur d'objet à une fonction C. L'utilisation de ce pointeur d'objet est expliquée au §C.3. Mis à part le fait de transmettre un pointeur d'objet, l'appel d'une fonction C à partir de C++ ne présente pas de difficultés particulières.

C.3 Appel d'une méthode orientée objet C++ à partir d'une fonction C

Pour pouvoir revenir au langage C++, tout en conservant la notion d'objet, il faut faire quelques ajouts dans les deux langages. Pour commencer, la fonction C doit avoir une connaissance de l'objet utilisé. C'est ici qu'intervient le pointeur d'objet « `void * myTileObject` ». Le projet TileUs est développé pour ne pas avoir à modifier un objet dans une fonction C. Aussi la connaissance en l'objet consiste juste en un pointeur. Un pointeur nul est utilisé dans chaque fonction faisant appel à une méthode C++.

```
// node.h
void node_new_init_message(struct node * handle, enum DIRECTION direction, void *
    myTileObject);
```

Les fonctions qui permettent de passer du langage C au C++ sont appelées « fonction *wrapper* ». Il faut les déclarer au début du fichier source C. Pour que le code puisse être utilisable par le simulateur et le nœud, il faut ajouter des directives conditionnelles de compilation.

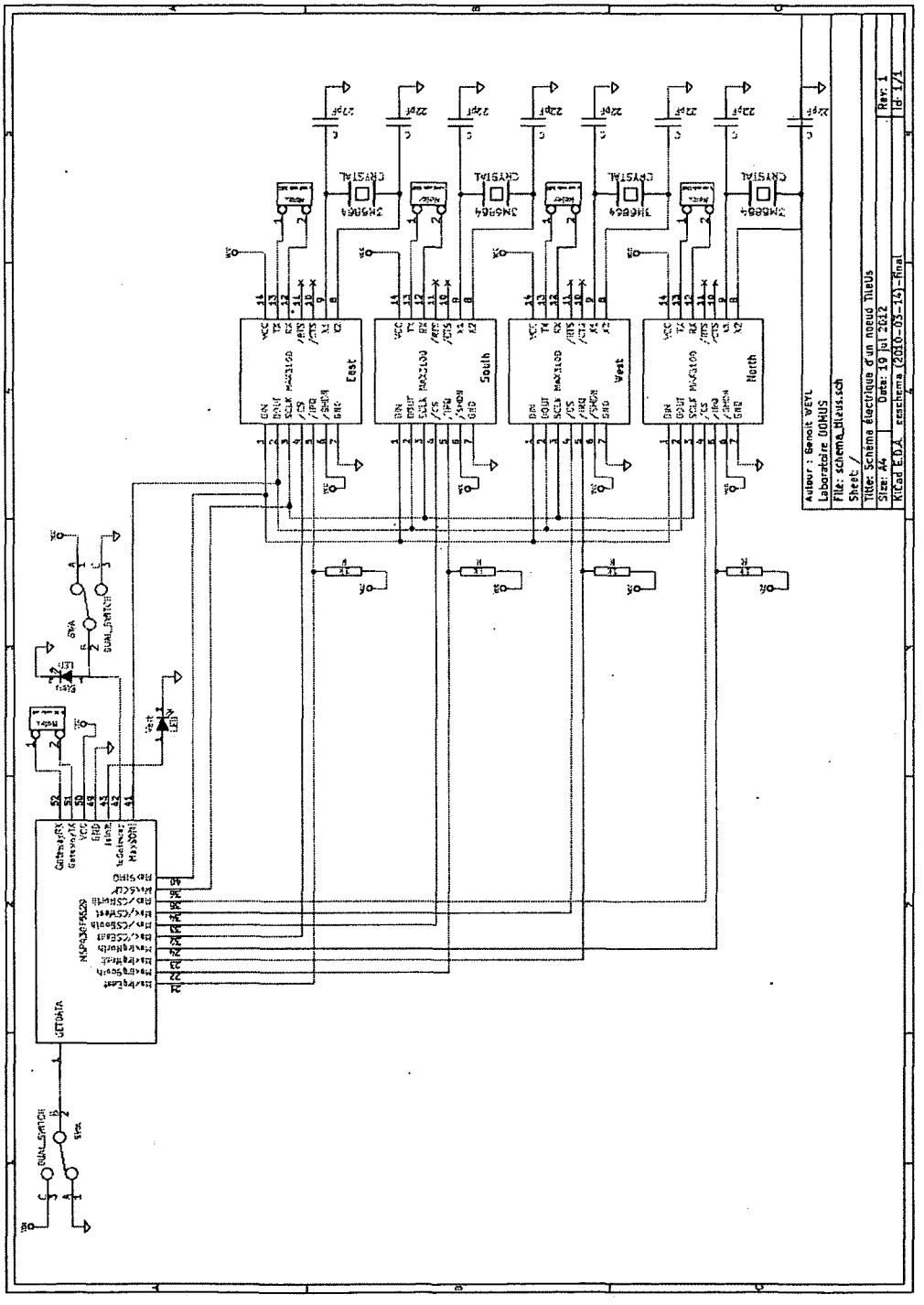
```
1  // node.c
2  #ifdef __SIMULATOR
3  /* declare the wrapper function */
4  void tile_init(void *, uint64_t );
5  void tile_get_data(void *, int);
6  void tile_send_map(void *, uint64_t, enum DIRECTION, uint64_t, int8_t, int);
7  void tile_send_data(void *, uint64_t, uint8_t, int);
8  #endif
9  ...
10 void node_new_init_message(struct node * handle, enum DIRECTION direction, void *
       myTileObject)
11 {
12 ...
13 #ifdef __SIMULATOR
14     tile_init(myTileObject, handle->id_number);
15 #endif
16 ...
17 }
```

Ces fonctions *wrapper* sont appelées en C mais elles sont exécutées dans un fichier C++. Elles permettent de revenir dans un langage orienté objet. En effet, le pointeur `myTileObject` n'est plus considéré comme un pointeur nul, mais comme un pointeur vers un objet `TileObject`. De cette manière, il est possible de faire appel aux méthodes de l'objet `TileObject`.

```
1  // tileobject.cpp
2  extern "C" void tile_init(TileObject * myTileObject, uint64_t id)
3  {
4      myTileObject->printInit(id);
5  }
6  extern "C" void tile_get_data(TileObject * myTileObject, int timeStamp)
7  {
8      myTileObject->printData(timeStamp);
9  }
10 extern "C" void tile_send_map(TileObject * myTileObject, uint64_t myId, DIRECTION
       direction, uint64_t neighborId, int8_t hop, int timeStamp)
11 {
12     myTileObject->sendMap(myId, direction, neighborId, hop, timeStamp);
13 }
14 extern "C" void tile_send_data(TileObject * myTileObject, uint64_t myId, uint8_t data,
       int timeStamp)
15 {
16     myTileObject->sendData(myId, data, timeStamp);
17 }
```

ANNEXE D

Schéma de câblage d'un nœud TileUs



LISTE DES RÉFÉRENCES

[1] Addlesee, M., Jones, A., Livesey, F. et Samaria, F. (1997). The ORL active floor [sensor system]. *Personal Communications, IEEE*, volume 4, numéro 5, p. 35–41.

[2] Alwan, M., Rajendran, P., Kell, S., Mack, D., Dalal, S., Wolfe, M. et Felder, R. (2006). A smart and passive floor-vibration based fall detector for elderly. Dans *Information and Communication Technologies, 2006. ICTTA '06. 2nd.* volume 1. p. 1003 –1007.

[3] Bauchet, J., Giroux, S., Pigot, H., Lussier-Desrochers, D., Lachapelle, Y. et Mokhtari, M. (2009). For some useful and usable acts of assistance when guiding people with cognitive impairments to complete activities. Dans *IJCAI '09 : Workshop on Intelligent Systems for Assisted Cognition (Twenty-first International Joint Conference on Artificial Intelligence).*

[4] Bussière, Y., Thouez, J.-P., Carrière, J. et Autres (2006). Le vieillissement de la population : Une nouvelle spécificité québécoise. *Fond de recherche société et culture Québec.* `http://www.fqrsc.gouv.qc.ca/fr/recherche-expertise/actualite/resultat-recherche.php` (page consultée le 9 mars 2012).

[5] Christ, R. (1996). Application and performance of personnel tracking systems. Dans *Security Technology, 1996. 30th Annual 1996 International Carnahan Conference.* IEEE, p. 120–128.

[6] David CRANE (s. d.). *Defense Review.* `http://www.defensereview.com/new-high-tech-sensor-laiden-smart-carpet-may-revolutionize-building-security/` (page consultée le 1 avril 2011).

[7] E.Q. Inc. (s. d.). *GaitMat II.* `http://www.gaitmat.com/` (page consultée le 7 mai 2012).

[8] Futur Shape (s. d.). *Futur Shape, SensFloor.* `http://www.future-shape.com/fr/technologies/41/sensfloor` (page consultée le 12 mars 2012).

[9] Giroux, S., Bauchet, J., Pigot, H., Lussier-Desrochers, D. et Lachappelle, Y. (2008). Pervasive behavior tracking for cognitive assistance. Dans *Proceedings of the 1st international conference on PErvasive Technologies Related to Assistive Environments.* PETRA '08. ACM, New York, NY, USA, p. 86 :1–86 :7.

[10] Glaser, R., Lauterbach, C., Savio, D., Schnell, M., Karadal, S., Weber, W., Kornely, S. et Stöhr, A. (2008). Smart carpet : A textile-based large-area sensor network.

[11] Greenberg, S. et Fitchett, C. (2001). Phidgets : easy development of physical interfaces through physical widgets. Dans *Proceedings of the 14th annual ACM symposium on User interface software and technology.* UIST '01. ACM, New York, NY, USA, p. 209–218.

[12] Helal, S., Mann, W., El-Zabadani, H., King, J., Kaddoura, Y. et Jansen, E. (2005). The gator tech smart house : a programmable pervasive space. *Computer*, volume 38, numéro 3, p. 50–60.

[13] Info Senior (s. d.). *Info Senior, TapisMetric : un sol détecteur de chutes.* `http://www.info-senior.com/tapismetric-chute-senior.html` (page consultée le 30 mars 2012).

[14] Institut de la statistique du Québec (s. d.). *Pyramide évolutive des âges.* `http://www.stat.gouv.qc.ca/donstat/societe/demographie/pyramide_age.htm` (page consultée le 9 mars 2012).

[15] Institution of Electrical Engineers (2003). Research news - walk this way for the smart floor. *Electronics Systems and Software*, volume 1, numéro 3, p. 5–7.

[16] Interlink Electronics (s. d.). *User Guide Force Sensing Resistor.* `http://www.phidgets.com/documentation/Phidgets/interlink-fsr-user-guide.pdf` (page consultée le 1 avril 2011).

[17] Jalal, A., Uddin, M. Z., Kim, J. T. et Kim, T.-S. (2011). Daily human activity recognition using depth silhouettes and $\mathcal{R}$ transformation for smart home. Dans Abdulrazak, B., Giroux, S., Bouchard, B., Pigot, H. et Mokhtari, M., *Toward Useful Services for Elderly and People with Disabilities*. Lecture Notes in Computer Science, volume 6719. Springer, p. 25–32.

[18] Jung, K. K., Son, D. S. et Eom, K. H. (2009). Rfid footwear and floor system. Dans *Computer Science and Information Engineering, 2009 WRI World Congress on.* volume 3. p. 72 –75.

[19] Kaddoura, Y., King, J. et Helal, A. (2005). Cost-Precision tradeoffs in unencumbered Floor-Based indoor location tracking. Dans *Proceedings of the third International Conference On Smart homes and health Telematic (ICOST)*.

[20] Kadouche, R., Chikhaoui, B. et Abdulrazak, B. (2010). User's behavior study for smart houses occupant prediction. *Annals of Telecommunications*, volume 65, p. 539–543.

[21] King, J., Bose, R., Yang, H., Pickles, S. et Helal, A. (2006). Atlas : A Service-Oriented sensor platform : Hardware and middleware to enable programmable pervasive spaces. Dans *Local Computer Networks, Proceedings 2006 31st IEEE Conference on*. p. 630–638.

[22] Kleusberg, A. et Langley, R. B. (1990). The limitations of GPS. *GPS World*, volume 1, numéro 2, p. 50–52.

[23] Lamarca, A., Chawathe, Y., Consolvo, S., Hightower, J., Smith, I., Scott, J., Sohn, T., Howard, J., Hughes, J., Potter, F., Tabert, J., Powledge, P., Borriello, G. et Schilit, B. (2005). Place lab : Device positioning using radio beacons in the wild. *Lecture notes in computer science*, p. 116–133.

[24] Lauterbach, C., Glaser, R., Savio, D., Schnell, M., Weber, W., Kornely, S. et Stöhr, A. (2005). A self-organizing and fault-tolerant wired peer-to-peer sensor network for textile applications. Dans Brueckner, S., Di Marzo Serugendo, G., Karageorgos, A. et Nagpal, R., *Engineering Self-Organising Systems*, Lecture Notes in Computer Science, *volume 3464*. Springer Berlin / Heidelberg, p. 256–266.

[25] Les Balances PAPP scales Inc. (s. d.). *Cellules de charge de type 'S' BSS de Transcell.* `http://www.balancepapp.ca/47-cellule-de-charge-bss.html` (page consultée le 7 mai 2012).

[26] Measurement Specialities (s. d.). *Measurement Specialities, cellule de charge FC22.* `http://www.meas-spec.com/product/t_product.aspx?id=2438` (page consultée le 7 mai 2012).

[27] Mon-club-elec (s. d.). *Mon Club elec.* `http://www.mon-club-elec.fr/pmwiki_mon_club_elec/pmwiki.php?n=MAIN.MaterielCapteurAnalogAccelerometreADXL3212axes18g` . (page consultée le 7 mai 2012).

[28] Murakita, T., Ikeda, T. et Ishiguro, H. (2004). Human tracking using floor sensors based on the markov chain monte carlo method. Dans *Proceedings of the Pattern Recognition, 17th International Conference on (ICPR'04) Volume 4 - Volume 04*. ICPR '04. IEEE Computer Society, Washington, DC, USA, p. 917–920.

[29] OMEGA (s. d.). *OMEGA, descriptif des types de cellules de charge.* `http://www.omega.com/prodinfo/loadcells.html` (page consultée le 1 avril 2011).

[30] Oracle Sun Developer Network (s. d.). *Mixing C and C++ Code in the Same Program.* `http://developers.sun.com/solaris/articles/mixing.html` (page consultée le 9 mars 2012).

[31] Orr, R. J. et Abowd, G. D. (2000). The smart floor : a mechanism for natural user identification and tracking. Dans *CHI '00 extended abstracts on Human factors in computing systems*. CHI EA '00. ACM, New York, NY, USA, p. 275–276.

[32] Patterson, D. J., Liao, L., Gajos, K., Collier, M., Livic, N., Olson, K., Wang, S., Fox, D. et Kautz, H. (2004). Opportunity Knocks : A System to Provide Cognitive Assistance with Transportation Services. p. 433–450.

[33] Pépin, N., Simonin, O. et Charpillet, F. (2009). Intelligent Tiles : Putting Situated Multi-Agents Models in Real World. Dans AAAI, A., *International Conference on Agents and Artificial Intelligence - ICAART'09*.

[34] Phidgets Inc. (s. d.). *Phidgets, Products for USB Sensing and Control.* `http://www.phidgets.com/products.php?category=3&product_id=3105` (page consultée le 7 mai 2012).

[35] Public Health Agency of Canada (2005). Report on senior's falls in canada. *Division of Aging and Seniors, Public Health Agency of Canada.* `http://www.phac-aspc.gc.ca/seniors-aines/publications/pro/`

injury-blessure/falls-chutes/index-eng.php (page consultée le 5 avril 2012).

[36] Qt Developer Network (s. d.). *Signlas & slots.* http://qt-project.org/doc/qt-4.8/signalsandslots.html (page consultée le 9 mars 2012).

[37] Rajalingham, R., Visell, Y. et Cooperstock, J. (2010). Probabilistic tracking of pedestrian movements via in-floor force sensing. Dans *Computer and Robot Vision (CRV), 2010 Canadian Conference on.* p. 143 –150.

[38] Savio, D. et Ludwig, T. (2007). Smart carpet : A footstep tracking interface. Dans *Advanced Information Networking and Applications Workshops, 2007, AINAW '07. 21st International Conference on.* volume 2. p. 754–760.

[39] Texas Instruments (s. d.). *Texas Instruments TS430PN80USB.* http://www.ti.com/graphics/tool/TS430PN80USB.jpg (page consultée le 7 mai 2012).

[40] Vstone (s. d.). *Vstone (traduction japonais/français).* http://www.vstone.co.jp/top/products/system/vs-ss-sf55.html (page consultée le 7 mai 2012).

[41] Wai, A. A. P., Devi, S. S., Biswas, J. et Panda, S. K. (2011). Pervasive intelligence system to enable safety and assistance in kitchen for home-alone elderly. Dans Abdulrazak, B., Giroux, S., Bouchard, B., Pigot, H. et Mokhtari, M., *Toward Useful Services for Elderly and People with Disabilities.* Lecture Notes in Computer Science, volume 6719. Springer, p. 276–280.

[42] Werner, F., Diermaier, J., Schmid, S. et Panek, P. (2011). Fall detection with distributed floor-mounted accelerometers : An overview of the development and evaluation of a fall detection system within the project ehome. Dans *Pervasive Computing Technologies for Healthcare (PervasiveHealth), 2011 5th International Conference on.* p. 354 –361.

[43] Yin, K. et Pai, D. K. (2003). Footsee : an interactive animation system. Dans *Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation.* SCA '03. Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, p. 329–338.

[44] Yu, C.-R., Wu, C.-L., Lu, C.-H. et Fu, L.-C. (2006). Human localization via multi-cameras and floor sensors in smart home. Dans *Systems, Man and Cybernetics, 2006. SMC '06. IEEE International Conference on.* volume 5. p. 3822 –3827.

[45] Yu, X. (2008). Approaches and principles of fall detection for elderly and patient. Dans *e-health Networking, Applications and Services, 2008. HealthCom 2008. 10th International Conference on.* p. 42 –47.

[46] Yun, K., Choi, S. et Kim, D. (2006). A robust location tracking using ubiquitous rfid wireless network. Dans Ma, J., Jin, H., Yang, L. et Tsai, J., *Ubiquitous Intelligence and Computing,* Lecture Notes in Computer Science, *volume 4159.* Springer Berlin / Heidelberg, p. 113–124.