

Table des matières

Résumé	i
Abstract	i
ملخص	ii
Remerciements.....	iii
Table des matières.....	v
Liste des figures	x
Liste des tableaux.....	xii

Chapitre I : Introduction générale

I.1. Problématiques et motivations.....	2
I.2. Objectifs.....	3
I.3. Contributions	5
I.4. Structure de la thèse et description des chapitres	5

Chapitre II : Les réseaux de capteurs sans fil, présentation et applications

II.1. Introduction	8
II.2. Définition des réseaux de capteurs sans fil	8
II.3. Les nœuds capteurs sans fil	9
II.3.1. L'unité d'acquisition (ou de capture).....	10
II.3.2. L'unité de traitement.....	10
II.3.3. L'unité de communication	10
II.3.4. Le module d'énergie ou batterie	11
II.3.5. L'unité de géolocalisation	11
II.3.6. L'unité de Mobilité	11
II.4. Défis de recherche	13
II.4.1. Niveau matériel.....	13
II.4.2. Niveau physique.....	13
II.4.3. Niveau MAC	14
II.4.4. Niveau Routage.....	14
II.4.5. Niveau Transport	14
II.4.6. Niveau Application	14
II.5. Les RCMSF.....	15
II.5.1. Capteur vidéo	16
II.5.2. Nœud capteur multimédia.....	17
II.5.3. Caméras basées sur des composants commerciaux	18
II.5.3.1. MeshEye.....	18
II.5.3.2. Cyclops.....	19
II.5.3.3. CMUcam	19
II.5.3.4. IMote	20

II.5.3.5. Stargate.....	20
II.5.4. Caméras conçues spécifiquement pour RCSF	21
II.6. Caractéristiques et contraintes d'un RCMSF	22
II.6.1. Consommation et efficacité énergétique.....	22
II.6.2. Bande passante.....	23
II.6.3. Scalabilité.....	24
II.6.4. Tolérance aux pannes.....	24
II.6.5. Sécurité	24
II.6.5.1. Gestion efficace de la Qualité d'Expérience (QoE) et de la Qualité de Service (QoS)	24
II.6.5.2. La confidentialité	24
II.6.5.3. Authentification et localisation d'un nœud	25
II.6.6. Développement de la plateforme	25
II.6.7. Couverture.....	25
II.6.8. Qualité des images capturées	25
II.7. Systèmes d'exploitation	26
II.7.1. TinyOS	26
II.7.2. Contiki OS	27
II.7.3. SOS	27
II.7.4. FreeRTOS	27
II.7.5. Mantis OS	28
II.7.6. Nut/OS	28
II.7.7. Nano-RK.....	29
II.8. Méthodes d'évaluation des performances	29
II.8.1. La méthode analytique	29
II.8.2. La simulation	29
II.8.3. L'émulation.....	30
II.8.4. Banc d'essai	30
II.9. Applications des RCMSF	33
II.9.1. La sécurité militaire	34
II.9.2. La protection des sites sensibles	34
II.9.3. Le pilotage intelligent	35
II.9.4. La surveillance environnementale	35
II.9.5. Surveillance et suivie médicale.....	35
II.9.6. Applications publiques.....	36
II.10. Conclusion.....	36

CHAPITRE III : Les feux de forêt et les méthodes de détection

III.1. Introduction.....	38
III.2. Une forêt, un feu, un incendie de forêt	38
III.3. Évolution des systèmes forestiers pour la détection d'incendie.....	40
III.4. Les RCSF et la détection des incendies de forêt.....	43

III.4.1. Contributions théoriques.....	43
III.4.2. Contributions expérimentales.....	44
III.4.3. Contributions basées sur la simulation	46
III.4.4. Synthèse et choix de la méthode adéquate	47
III.5. Le système de détection canadien.....	51
III.5.1. Indice du combustible léger (ICL).....	52
III.5.2. Indice d'humidité de l'humus (IH).....	52
III.5.3. Indice de sécheresse (IS)	52
III.5.4. Indice de propagation initiale (IPI).....	52
III.5.5. Indice du combustible disponible (ICD)	53
III.5.6. Indice forêt météo (IFM)	53
III.6. Le système de détection canadien adapté au contexte local	54
III.6.1. Algorithme de détection canadien	54
III.6.2. Autres systèmes de détection des feux de forêt.....	55
III.7. Conclusion	58
 CHAPITRE IV - Contribution1 : Conception, implémentation et évaluation du système M2FS (Multimedia Forest Fire System)	
IV.1. Introduction.....	60
IV.2. Approche de détection des feux de forêt adoptée	60
IV.3. Conception du système multimodal M2FS.....	61
IV.4. Implémentation du système multimodal M2FS	64
IV.4.1. Module de capture	65
IV.4.2. Module de transmission.....	66
IV.4.2.1. Les relais.....	66
IV.4.2.2. Le collecteur	66
IV.4.2.3. La passerelle	67
IV.4.3. Module d'analyse et de sauvegarde.....	67
IV.4.4. Un nœud capteur scalaire	68
IV.4.4.1. Les capteurs Dht11/Dht22	68
IV.4.4.2. La carte Arduino.....	69
IV.4.4.3. Le module radio Xbee et le Shield Xbee	71
IV.4.5. Un nœud capteur multimédia	73
IV.4.5.1. Nœuds indépendants.....	73
IV.4.5.2. Nœud unique « Deux en un »	74
IV.4.5.3. La caméra OV2640.....	75
IV.4.5.4. Le Shield Arducam.....	76
IV.4.5.5. Xbee explorer	77
IV.4.6. La station de base	79
IV.5. Résultats et discussion	80
IV.5.1. Contrainte temporelle	81
IV.5.1.1. Temps de détection.....	81

IV.5.1.2. Temps d'envoi	84
IV.5.2. Consommation d'énergie	86
IV.5.3. Réception des données capturées	91
IV.6. Conclusion	95
CHAPITRE V - Contribution2 : Compression des données pour l'économie d'énergie dans le système M2FS	
V.1. Introduction	97
V.2. Compression de données.....	98
V.2.1. Notions générales.....	98
V.2.2. Types de compression.....	99
V.2.3. La compression dans les RCSF	99
V.2.4. Caractéristiques de la compression de données.....	101
V.2.4.1. Perte de données	101
V.2.4.2. Symétriques et Asymétrique	101
V.2.4.3. Adaptatifs et non adaptatif	102
V.3. L'agrégation de données	102
V.3.1. Notions générales.....	102
V.3.2. Schémas d'agrégation de données	102
V.3.3. État de l'art	103
V.4. La corrélation de données	104
V.5. Solutions adoptées pour la compression de données dans M2FS	105
V.5.1. TiNA	106
V.5.2. L'approximation polynomiale comme technique de compression dans les RCSF	109
V.5.2.1. Notions de base	109
V.5.2.2. Algorithmes de compression.....	111
V.5.2.3. Modélisation de la technique d'approximation.....	114
V.6. Résultats et discussion.....	116
V.6.1. Contrainte temporelle	117
V.6.2. Taux de compression	120
V.6.2.1. APFF	120
V.6.2.2. TiNA	122
V.6.2.3. Discussion	123
V.6.2.4. Restitution de données	123
V.6.2.5. Compression multimédia	124
V.6.3. Contrainte énergétique.....	125
V.7. Conclusion.....	128

Conclusion générale & perspectives

Annexe A : Code sources et programmes

1. Introduction	132
2. Programme de M2FS.....	132

3. Programme de compression pour M2FS	135
Bibliographie.....	137

Liste des figures

Figure 2.1 : Architecture globale d'un réseau de capteur sans fil	9
Figure 2.2: Architecture d'un nœud capteur	10
Figure 2.3 : Chronologie de développement des nœuds capteurs	12
Figure 2.4: Architecture type d'un RCMSF proposé dans [18]	16
Figure 2.5 : Plateforme MeshEye [23]	19
Figure 2.6 : Caméra cyclops attachée à un nœud capteur Mica2 [25]	19
Figure 2.7 : CMUCam3 attachée à TMote Sky [26]	20
Figure 2.8 : CMUCam5 Pixy [26]	20
Figure 2.9 : Plateforme basée iMote2 [27]	20
Figure 2.10 : Stargate attaché avec Webcam [28]	20
Figure 2.11 : Un capteur d'image ALOHA attachée à un mote Mica2 [20]	21
Figure 2.12 : Facteurs de conception des RCSF et RCMSF	22
Figure 2.13 : Typologie des réseaux de capteurs sans fil	33
Figure 2.14 : Classification des applications des RCSF	34
Figure 3.1 : Vue générale de l'espace forestier et de la région d'intérêt	39
Figure 3.2 : Historique sur le nombre d'incendies et de superficie durant les cinq dernières années	40
Figure 3.3 : Etapes de calcul de l'indice IFM	51
Figure 3.4 : Signification de l'indice IFM sur terrain : (a) IFM=9, (b) IFM=17 et (c) IFM=24	54
Figure 3.5 : Algorithme de détection de l'approche canadienne	55
Figure 3.6 : Déroulement des messages lors d'une alerte	56
Figure 3.7 : Schéma bloc du système FireWxNet [103]	58
Figure 4.1 : Etapes principales de la solution proposée [1]	61
Figure 4.2 : Modélisation par diagramme de cas d'utilisation de l'environnement d'évaluation	62
Figure 4.3 : Diagramme de séquence pour l'échange de messages entre les différents acteurs de l'application	63
Figure 4.4 : Schéma général de l'application selon le scénario1	64
Figure 4.5 : Schéma détaillé de l'application selon le scénario 2	65
Figure 4.6 : Les capteurs DHT11 et DHT22	68
Figure 4.7 : Brochage de capteur Dht22 à la carte Arduino	69
Figure 4.8 : Vue de face de la carte Arduino due	70
Figure 4.9 : module Xbee	73
Figure 4.10 : Shield Xbee	73
Figure 4.11 : XBee Explorer	73
Figure 4.12 : Illustration du nœud multimédia utilisé	75
Figure 4.13 : Vue de face de la caméra Ov2640	75
Figure 4.14 : Vues face et profile du Shield Arducam Rev.c	76
Figure 4.15 : Aperçu du logiciel XCTU	78
Figure 4.16 : Organigramme de capture d'une image	79
Figure 4.17 : Aperçu du banc d'essai déployé correspondant au système M2FS	80
Figure 4.18 : Aperçu des résultats obtenus	81
Figure 4.19 : Exemple de calcul d'un délai	83
Figure 4.20 : Aperçu sur les résultats obtenus (suite de la figure 4.19)	84
Figure 4.21 : Aperçu des signaux mesurés par un oscilloscope	86
Figure 4.22 : Brochage du nœud capteur utilisé avec l'oscilloscope	87
Figure 4.23 : Consommation énergétique de quelques plateformes selon l'application installée (chaque ligne du tableau 4.8)	88
Figure 4.24 : Énergie consommée par quelques nœuds programmés par l'application Blink	90

(led)	
Figure 4.25 : Énergie consommée par quelques nœuds programmés pour capturer la température et l'envoyer au Sink	90
Figure 4.26 : Énergie consommé par Arduino Uno programmé par les fonctions du programme principal	90
Figure 4.27 : Énergie consommé par Arduino Uno et Due programmés par les fonctions du programme principal	90
Figure 4.28 : Code source de l'application python qui stocke les données récupérées du port USB dans un fichier texte	92
Figure 4.29 : Code source (en python) transférant une image écrite en hexadécimale en image JPEG	93
Figure 4.30 : Image construite par le code python montré dans la figure 4.30	93
Figure 4.31 : Extrait du code source (en python) permettant de télé-verser une image stockée dans le disque dur vers le <i>cloud</i>	93
Figure 4.32 : Succession d'images prises lors d'une simulation du déclenchement du feu	94
Figure 4.33 : Contenu de la base de données au moment de la simulation d'incendie	94
Figure 5.1 : Classification des approches basées sur la donnée [29]	98
Figure 5.2 : Classification de quelques corrélations considérées dans les mécanismes de compression de données [113]	105
Figure 5.3 : Algorithme de détection et de compression en utilisant TiNA dans un nœud capteur	108
Figure 5.4 : Implémentation de TiNA au niveau du le Sink	108
Figure 5.5 : Aperçu sur le mode de compression par approximation	110
Figure 5.6 : Algorithme de détection et de compression en utilisant APFF d'un nœud	112
Figure 5.7 : Algorithme de réception de données par le Sink	113
Figure 5.8 : Algorithme de compression polynômiale avec fenêtre variable	114
Figure 5.9 : Diagramme de cas d'utilisation de M2FS en tenant compte de la compression	115
Figure 5.10 : Diagramme de séquence de M2FS en tenant compte de la compression	116
Figure 5.11 : Comparaison entre les délais de compression des méthodes TiNA et APFF	118
Figure 5.12 : Comparaison entre le temps de transmission des trois méthodes	119
Figure 5.13 : Impact de la taille d'une séquence sur l'efficacité de l'approximation de l'APFF	121
Figure 5.14 : Impacte de la valeur du Tct sur le taux de compression à l'aide de TiNA	122
Figure 5.15 : Comparaison entre la restitution de la température et de l'humidité compressées à la source	124
Figure 5.16 : Signal généré par l'oscilloscope qui mesure le courant en tenant compte de la compression APFF	126
Figure 5.17 : Signal généré par l'oscilloscope pour mesurer le courant en tenant compte de la compression TiNA	126
Figure 5.18 : Méthode de calcul du temps pris par une tâche	127
Figure 5.19 : Comparaison entre la consommation d'énergie selon trois modes : sans compression, avec les compressions TiNA et APFF	128

Liste des tableaux

Tableau 2.1 : Comparaison de quelques plates-formes de nœuds capteurs [11]	12
Tableau 2.2 : Caractéristiques des systèmes d'exploitation dédiés aux RCSF	28
Tableau 2.3 : Comparaison entre les méthodes d'évaluation des performances	32
Tableau 3.1 : Comparaison entre les méthodes de détection des feux de forêt	40
Tableau 3.2 : Classification des solutions dédiées aux incendies de forêt	50
Tableau 3.3 : Inflammation potentielle par rapport à la valeur de l'ICL [98]	53
Tableau 3.4 : Danger potentiel d'incendie d'après l'indice IFM [98]	53
Tableau 3.5 : Indice de danger d'incendie [100]	56
Tableau 3.6 : Indice de danger coréen pour la détection des feux de forêt [102]	57
Tableau 4.1 : caractéristiques des capteurs DHT11 et DHT22	68
Tableau 4.2 : Caractéristiques techniques de la carte Arduino due	71
Tableau 4.3 : Principales caractéristiques du Xbee	72
Tableau 4.4 : Caractéristiques de la camera Ov2640	76
Tableau 4.5 : Caméras supportées par le shield Arducam	77
Tableau 4.6 : Caractéristiques des PC utilisés	80
Tableau 4.7 : Temps pris par trois cartes Arduino de quelques tâches	82
Tableau 4.8 : Composantes de chaque application installée sur une plateforme	89
Tableau 5.1 : Temps de capture, de calcul, de compression et de transmission des données scalaires	117

Chapitre I

Introduction générale

I.1. Problématiques et motivations

Aujourd'hui, plusieurs domaines de recherche orientent leurs études scientifiques vers la recherche à des solutions liées essentiellement aux problématiques relatives à l'environnement. Les congrès et les conférences internationales organisées pour faire face à ces nouveaux axes de recherche, appellent la communauté scientifique à trouver des solutions urgentes, sérieuses et viables à ses nouveaux défis. Cette communauté bénéficie souvent de financements de projets coûteux pour lutter contre les changements climatiques, et plus spécifiquement aux réchauffements climatiques et le dégagement des gaz à effet de serre dans la planète [1].

Parmi ces problématiques qui ont attiré le plus d'attention ces dernières années, on trouve les catastrophes causées par des incendies de forêt. Les incendies de forêt contribuent à peu près à 30 % des émissions des gaz à effet de serre, plus particulièrement le CO₂ [1]. Ces dernières années, la gestion de ces incendies est devenue une tâche très difficile et ardue. Les phénomènes les plus fréquents dans la plupart des incendies de forêt sont la perte de vies humaines, de végétation et de la faune et de la flore. En plus, ils provoquent une grande perte de ressources naturelles précieuses, les incendies de forêt représentent un risque accru non seulement pour la vie des êtres humains, mais également pour les habitations et les animaux sauvages qui vivent dans la forêt. Selon le Système national d'évaluation du risque d'incendie (NFDRS) [2], si un incendie est détecté dans les 6 minutes suivant son apparition, il peut être facilement éliminé avant qu'il ne se transforme en un incendie de grand ampleur. Pour cela, la détection en temps réel des feux de forêt présente un défi majeur pour lequel des solutions efficaces doivent être trouvées et qui pourront avoir des impacts très positifs sur le plan sociétal et économique d'un pays. Ainsi, l'utilisation des technologies de l'information et de communications modernes permettent d'apporter des éléments de réponses à ce problème. Comme exemple, l'utilisation de la technologie des réseaux de capteurs sans fil, capables de détecter un incendie très tôt, est une des solutions prometteuses proposées récemment en pratique. Ainsi, la plupart des solutions proposées sont de nature uni-modale dans le sens où elles emploient des mesures de grandeurs physiques scalaires comme la température, l'humidité, la vitesse du vent ...etc. Elles présentent malheureusement des limites au niveau de la précision de l'opération de détection des feux de forêt en générant par exemple de fausses alarmes dont l'impact économique est souvent considérables pour les autorités compétentes. L'idée de surmonter cette limite en améliorant cette précision dans le mécanisme de détection nous semble une problématique de recherche intéressante. En effet, l'emploi de moyens d'acquisition vidéo plus riche en information couplé aux données scalaires collectées permet de concevoir une solution multimodale (scalaire-véo) de détection plus efficace qui minimise considérablement (pour ne pas dire élimine) ces fausses alarmes indésirables.

Ainsi, nos travaux de recherche dans cette thèse s'inscrivent dans la perspective de lutter contre les feux de forêt en proposant des solutions technologiques efficaces sur les méthodes et techniques pouvant aider à détecter d'une manière sûre (donc sans provoquer des fausses alarmes) et rapide la première flamme de la forêt en vue d'en informer les autorités compétences.

I.2. Objectifs

Les avancées technologiques qui ont touché le domaine électronique, et plus spécifiquement, la miniaturisation des dispositifs microélectroniques, ont permis d'ouvrir un portail devant un spectre très large d'applications qui visent à résoudre de grands problèmes où les premières solutions n'ont pas été prévues. Comme résultat de ces avancées, on trouve les Réseaux de Capteurs Sans Fil (RCSF) (ou *Wireless Sensor Network* – WSN – en anglais). Ces équipements miniaturisés ont vulgarisé plusieurs domaines de recherche ces dernières années, comme par exemple la médecine, le domaine militaire, l'environnement, le transport, les villes intelligentes...[3]

La famille des RCSF est divisée en plusieurs catégories, chacune d'elles s'intéresse à un type de capture particulier. Nous nous intéressons dans cette thèse au type de capture scalaire effectuée à l'aide de capteurs spécifiques embraqués dans chaque nœud capteur (température, humidité...), et au capture multimédia (image/vidéo) selon le type de la caméra attachée au nœud capteur. Le réseau résultant, qui emploie les deux modalités (scalaire-image) dans sa fonction de capture est dit dans cette thèse Réseaux de Capteurs Multimédias Sans Fil (RCMSF).

Dans cette thèse, nous nous intéressons à la problématique de détection des feux de forêt à l'aide d'un RCMSF, cela afin de limiter au maximum l'inflammation potentielle des incendies et de les détecter assez rapidement possible s'ils ont lieu.

Notre objectif principal est de trouver un moyen permettant de définir l'intensité de danger d'inflammation suivant toute situation que peut avoir l'espace surveillé. C.-à-d. en état normal (sans danger) une indication faible de l'intensité, et élevée dans toute situation critique (où la probabilité de l'inflammation est importante).

Deuxièmement, après détection d'un événement considéré dangereux (seuil élevé), nous intégrons un mécanisme de vérification et de confirmation (de l'alerte générée), afin de consolider l'alarme par la surveillance multimédia grâce aux modules caméras.

Troisièmement, acheminer l'information pertinente (l'alarme) via le réseau déployé, en assurant une longue durée de vie du réseau, en intégrant des techniques de conservation d'énergie telle que la compression de données afin de gagner dans le volume de données à transmettre.

Le banc d'essai visé assure jusqu'à présent les opérations de collecte, de détection, de vérification et de prévention des services spécialisés. La base de connaissance générée par notre prototype permet de construire de nouvelles instances qui peuvent être générées en utilisant plusieurs mécanismes de l'intelligence artificielle (comme le *data mining*) afin de prédire de futures situations, et de consolider le mécanisme de prédiction d'événements et d'assurer une décision robuste par conséquent !

Le sujet que nous traitons à travers cette thèse est un sujet de recherche qui nécessite impérativement des implémentations pratiques. Nous proposons une solution pour la détection des incendies de forêt le plus rapidement possible et informer les services compétents d'intervention via le réseau pour une intervention rapide. L'objectif essentiel visé par ce travail consiste à exploiter des informations multimodales (scalaire et image) pour diminuer au maximum les fausses alertes qui sont pénalisantes sur le plan économique. Nous considérons dans ce qui suit les principales hypothèses pour développer notre solution :

- Collecter des grandeurs physiques, qui entrent dans le calcul de l'indice de danger IFM, à l'aide d'un capteur de température/humidité (comme DHT22).
- Affecter des constantes recommandées par les services météorologiques aux paramètres que nous n'avons pas pu mesurer (par manque d'une station météo) et qui entrent dans le calcul de l'indice IFM. Il s'agit précisément de la vitesse du vent (13 km/h) et les précipitations (0mm) pour le mois du juillet (07).
- Considérer au départ un banc d'essai simple (composé d'un nœud capteur, un coordinateur pour représenter un nœud intermédiaire de routage et un Sink) comme prototype de validation préliminaire selon le matériel disponible. Le facteur d'échelle peut être étudié selon la disponibilité du matériel correspondant dont dispose notre laboratoire (technologie Arduino avec ses différentes variantes).
- Placer d'une manière déterministe les composants du banc d'essai dans un espace forestier (10 m² de la forêt de M'SILA – Oran –) et détecter la présence ou non du feu à quelques mètres dont la position géographique peut être obtenue par le placement d'un capteur de position GPS.
- Intégrer des solutions de détection des fausses alarmes et passer au multimodale par l'introduction des techniques de confirmation à l'aide d'images ou du vidéo selon la faisabilité technique.
- Mesurer le temps nécessaire en tant que paramètre de performance pour détecter un incendie en utilisant le banc d'essai.

- Mesurer la quantité d'énergie consommée par chaque tâche (collecte, calcul de l'IFM, transmission) effectuée par un nœud capteur dans le processus de surveillance, et la considérer comme paramètre de comparaison pour le choix d'une technologie qui favorise l'économie d'énergie.
- Mesurer la quantité d'énergie consommée par un nœud source pour évaluer le gain énergétique obtenu par des techniques de compression de données scalaires, et effectuer une étude comparative de la solution développée avec des solutions de l'état de l'art.

I.3. Contributions

Le but de ce travail de thèse consiste à la mise en place, à base de deux contributions - détaillés par la suite -, d'un système de monitoring multi tiers des feux de forêt M2FS (*Multimédia Forest Fire System*) en utilisant la technologie des RCSF.

La première contribution permet de concevoir et de développer un système multi tiers de détection des feux de forêt suite à une détection d'incendie par des mesures des grandeurs scalaires (température, humidité ...) obtenues par le nœud capteur (premier tiers). Ce dernier passe le contrôle à un dispositif d'acquisition d'image (deuxième tiers) pour collecter d'autres informations riches, et capables d'aider efficacement les décideurs pour la prise de décision adéquate.

Par contre, la deuxième contribution s'intéresse à la problématique de consommation d'énergie. La compression de données présente une solution clé pour la conservation des niveaux énergétiques des nœuds capteurs. Nous intégrons cette solution à cette contribution dans les nœuds capteurs afin d'étendre la durée de vie de notre système de détection M2FS. Nous présentons d'abord quelques techniques de compression conçues spécialement pour les dispositifs à ressources réduites, et nous implémentons deux techniques de compression de données scalaires (TiNA et APFF), utilisés dans les travaux de recherche de notre laboratoire [4, 5], dans les nœuds capteurs. Enfin, une comparaison des résultats obtenus par les deux méthodes de compression avec celles présentées dans la première contribution (sans compression) montrent une amélioration remarquable des performances en termes de consommation d'énergie.

I.4. Structure de la thèse et description des chapitres

Le manuscrit est structuré en quatre chapitres en plus de ce chapitre (Introduction).

Le deuxième chapitre présente la classe des réseaux de capteurs sans fil (RCSF), technologie de base, employée dans cette étude. Nous abordons ensuite une extension de cette classe, à savoir, les réseaux de capteurs multimédia sans fil (RCMSF), qui permettent de manipuler les données multimédia image/vidéo. Nous terminons ce chapitre par la présentation de quelques applications pratiques de ces réseaux.

Le troisième chapitre est consacré aux feux de forêt et les méthodes de leur détection. Nous mettons l'accent sur l'utilisation des RCSF dans les mécanismes de détection des feux de forêt. Nous présentons le système Canadien, adapté par le contexte local, et nous montrons comment l'indice forêt météo (IFM) est calculé.

Le quatrième chapitre détaille notre première contribution : conception, implémentation et évaluation d'un système de détection des feux de forêt multitiers (scalaire, image/vidéo), capable de minimiser les fausses alarmes, que nous avons appelé M2FS.

Le chapitre cinq, aborde notre deuxième contribution dans l'objectif consiste à optimiser la consommation énergétique des nœuds capteurs en faisant appel à des techniques de compression des grandeurs physiques scalaires basées sur l'approximation polynomiale.

Le manuscrit se termine par une conclusion et quelques travaux en perspective.

Chapitre II

Les réseaux de capteurs sans fil, présentation et applications

II.1. Introduction

Au cours de son évolution, le paradigme sans fil a vu naître diverses architectures dérivées, telles que : les réseaux cellulaires, les réseaux locaux sans fil et autres. Durant ces dernières années, une architecture nouvelle a vu le jour : les réseaux de capteurs sans fil (RCSF). Ce type de réseaux résulte d'une fusion de deux pôles de l'informatique moderne : les systèmes embarqués et les communications sans fil. Les RCSF sont souvent considérés comme étant les successeurs des réseaux ad hoc [3]. En effet, les RCSF partagent avec les MANET (*Mobile Ad hoc NETworks*) plusieurs propriétés en commun, tels que l'absence d'infrastructure et les communications sans fil, mais l'une des différences clés entre ces deux architectures est le domaine d'application.

Contrairement aux réseaux MANET, qui n'ont pas pu connaître un vrai succès, cette technologie (RCSF) a pu s'instaurer comme acteur incontournable dans les architectures réseaux actuelles, vu leur réalisme et leur apport concret grâce au support hertzien qui offre des avantages par rapport à d'autres technologies, tels que :

- La facilité du déploiement,
- L'ubiquité de l'information,
- Le coût réduit d'installation.

Dans ce chapitre, nous présentons les principes de conception des RCSF multimédia, leurs différentes caractéristiques et domaines d'application, nous montrons notre choix sur la méthode d'évaluation des performances utilisée, enfin nous présentons quelques exemples d'applications des RCSF.

II.2. Définition des réseaux de capteurs sans fil

Un réseau de capteur sans fil (en anglais WSN : *wireless sensor network*) est un ensemble (de centaines voire des milliers) de petits dispositifs dits nœuds capteurs (*mote* en anglais). Ils se déploient généralement d'une façon aléatoire par voie aérienne à l'aide d'un avion ou d'un hélicoptère dans un champ de déploiement, afin de collecter des données – scalaires ou multimédia – et les transmettre à l'aide d'autres nœuds capteurs appelés nœuds relais (*Relay node* en anglais) pour atteindre la station de base ou puits (*Sink* ou *base station* en anglais) [3, 6].

La durée de vie d'un RCSF est un grand challenge qui dépend du paramètre jouant un rôle très important dans le fonctionnement de ce réseau, à savoir l'économie d'énergie. Les nœuds capteurs sont alimentés par des batteries, ce qui nécessite une gestion sérieuse de cette ressource face aux protocoles qui exécutent des algorithmes d'auto organisation (qui engendrent beaucoup de trafic pour détecter le voisinage et le traçage des chemins) d'une part, et aussi à l'effort fourni par les

nœuds capteurs afin de communiquer l'information dans les conditions les plus extrêmes dictées par l'environnement (comme la perte des nœuds – feu ou eau ou épuisement de la réserve énergétique –).

Le schéma général montrant la structure d'un RSCF est donné par la figure 2.1. Dans ce schéma nous trouvons l'ensemble des nœuds capteurs qui forment une zone de couverture et de monitoring (présenté sous forme d'un nuage). Ces nœuds communiquent l'information capturée au nœud collecteur (ou station de base), qui à son tour transmet cette information via le réseau de transport (satellite, internet, GSM – 3 G ou 4 G –) vers un Cloud ou directement vers l'utilisateur final chargé de la supervision..

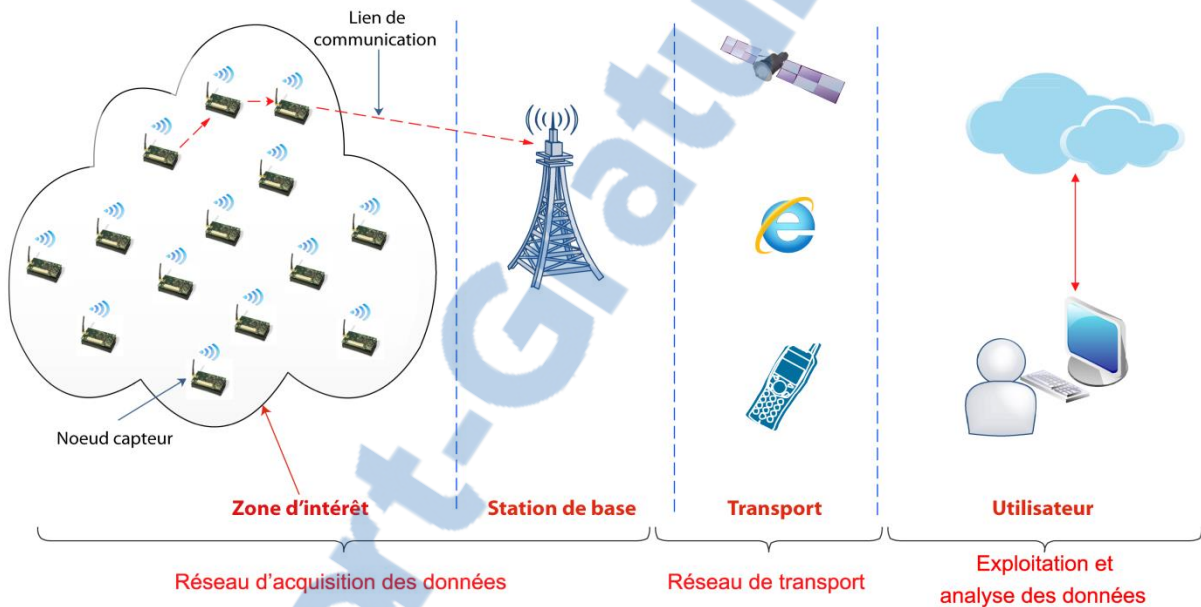


Figure 2.1 : Architecture globale d'un réseau de capteur sans fil

II.3. Les nœuds capteurs sans fil

Un nœud capteur est un petit équipement composé généralement de quatre modules (unités) de circuits électroniques élémentaires ; unité d'alimentation – en énergie –, unité d'acquisition, unité de traitement et unité de communication. D'autres modules sont optionnels peuvent être envisagés, à savoir, unité de géolocalisation, unité de mobilité et un module pour l'extraction de l'énergie de l'environnement. L'architecture matérielle de cet équipement est détaillée dans la figure 2.2.

Le fonctionnement du nœud capteur dépend de l'application installée, le processus de déroulement de l'application fait appel aux composants qui contribuent à son exécution. Dans ce qui suit nous découvrons la tâche affectée à chaque composant [3].

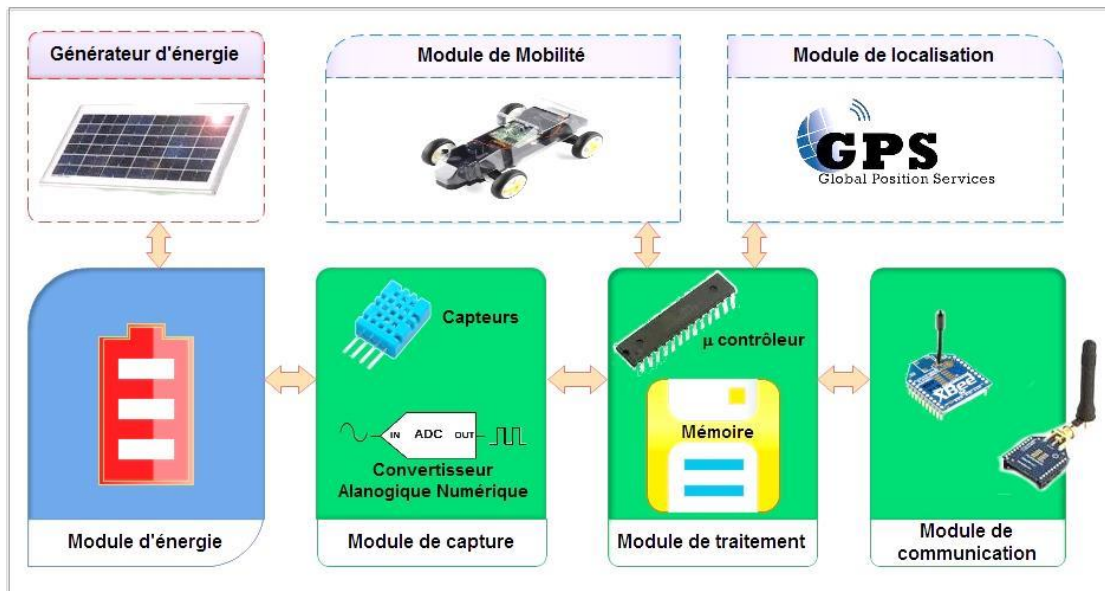


Figure 2.2 : Architecture d'un nœud capteur

II.3.1. L'unité d'acquisition (ou de capture)

Elle prend en charge la collecte des grandeurs physiques des phénomènes à observer par le(s) capteur(s) installé(s) (le type du capteur dépend de l'application et/ou la connectivité physique avec le nœud). Comme exemple de collecte on trouve la température, l'humidité, la lumière, etc. Les mesures brutes relevées par le(s) capteur(s) en mode analogique doivent être converties en valeurs numériques à l'aide du module ADC (*Analogic to Digital Converter*).

II.3.2. L'unité de traitement

Cette Unité est composé d'un microcontrôleur généralement de faible cadence de fonctionnement (quelques dizaines de MHz) qui commande tous les autres composants du nœud capteur et prend en charge le traitement et l'exploitation des données. La seconde sous unité c'est la « Mémoire » (RAM, ROM, Flash) dispose d'une quantité de stockage de l'ordre de quelques dizaines à quelques centaines de Kilo octets (ko). Elle permet de stocker les mesures capturées (données), l'application, et les données relatives aux protocoles de communication (gestion des tables de voisinage, de routage, encodage et décodage des paquets, ... etc.).

II.3.3. L'unité de communication

Elle gère la communication entre les différents nœuds du réseau par le biais d'un module radio. Ce module (appelé aussi *transciever*) fonctionne généralement sur la bande de fréquence libre ISM 2,4 GHz, et prend en charge le contrôle d'accès au médium, la mise en forme des trames, l'encodage et le décodage des signaux...etc. Ce module bascule entre les modes réception (Rx), transmission (Tx) et écoute de la porteuse (*idle*). Il se met souvent (s'il est programmé) en mode veille (*sleep*) en cas d'inactivité afin de minimiser considérablement la consommation d'énergie.

II.3.4. Le module d'énergie ou batterie

Ce module est responsable de fournir de l'énergie à tous les composants du nœud capteur. Il s'agit d'une batterie ou un jeu de piles de capacité limitée avec une autonomie qui peut atteindre quelques mois voir plusieurs années (selon l'application). Le remplacement de la batterie n'est pas toujours possible vu la difficulté d'accès aux nœuds capteurs. Cette contrainte a permis l'émergence d'une nouvelle approche d'extraction d'énergie de l'environnement comme par exemple l'énergie provenant des vibrations, de la chaleur, de la lumière, et même des ondes électromagnétiques, et sa conversion en courant électrique pour alimenter en permanence le nœud capteur [7-10].

II.3.5. L'unité de géolocalisation

Cette unité optionnelle permet de définir la position absolue dans l'espace des nœuds capteurs en employant des technologies comme le GPS afin de joindre convenablement les données fournies par les capteurs à leur emplacement. Entre autres, quand le déploiement des capteurs se fait manuellement, la carte du réseau peut être préétablie et la position des nœuds préenregistrée. D'autres techniques de localisation peuvent contribuer à la connaissance de l'emplacement des nœuds en se basant sur les positions relatives des nœuds voisins, comme la technique de triangulation qui basée sur l'estimation de distance à partir de l'atténuation des ondes radio, des délais de propagation des ondes sources, ...etc.

II.3.6. L'unité de Mobilité

Un Actionneur peut être ajouté au nœud capteur afin d'exécuter des commandes qui peuvent être programmées soit d'une façon périodique, soit selon le résultat d'un événement (calculé par le microcontrôleur ou mesuré par les capteurs). Cette action doit changer soit l'emplacement du nœud capteur dans le cas de mobilité, ou bien la position ou la forme d'un objet voisin dans le cas d'une installation industrielle par exemple.

Le tableau 2.1 présente une comparaison entre quelques principales plates-formes de nœuds capteurs commerciales, sachant que d'autres architectures (environ 140 autres modèles) sont disponibles dans la littérature et qui ne sont pas citées dans le tableau 2.1.

Nœud	Fabricant	Microcontrôleur	Horloge	Module radio	RAM	Mémoire Flash
Wec 99	Université Berkeley	Atmel at90ls8535	4 Mhz	Rfmonolithics tr1000	512 b	8 ko
Mica	Université Berkeley	Atmel atmega128l	8 Mhz	Rfmonolithics tr1000	4 ko	128 ko
Mica2	Memsic	Atmel atmega128l	8 Mhz	Chipcon cc1000	4 ko	128 ko
Micaz	Memsic	Atmel atmega128l	8 MHz	Chipcon cc2420	4 ko	128 ko
Imote2	Crossbow	Intel pxa271 xscale	13-416 MHz	Chipcon cc2420	256 ko	32 mb
Tmotesky/ Telosb	Memsic	Ti msp430f1611	8 MHz	Chipcon cc2420	10 ko	48 ko
Btnode rev3	Btnode	Atmel atmega128l	8 Mhz	Chipcon cc1000	4 ko	128 ko
Zn1	Hitachi	Renesas h8s/2218	4-24Mhz	Chipcon cc2420	12 ko	N/a
Sunspot	Oracle	Atmel arm920t	180 Mhz	Chipcon cc2420	512 ko	4 mb
Arduino bt	Arduino	Atmel atmega328	20 Mhz	X	2 ko	32 ko
Sensenode	Genetlab	Ti msp430f1611	8 Mhz	Chipcon cc2420	10 ko	48 ko
Z1	Zolertia	Ti msp430f2617	16 Mhz	Chipcon cc2420	8 ko	92 ko
Iris	Memsic	Atmel atmega1281	16 Mhz	Atmel at86rf230	8 ko	128 ko
Lotus	Memsic	Cortex m3	10-100Mhz	Atmel at86rf231	64 ko	512 ko
M12	Redwire	Freescale mc13224v	24-26Mhz	Freescale mc13224v	96 ko	128 ko
Bpart	Teco/kit	8051 on bluegiga ble112	32 Mhz	Cc2540	8ko	256ko
Panstamp Nrg	Panstamp	Ti msp430 (cc430f5137)	8-24Mhz	Chipcon cc11xx	4 ko	32 ko
Waspnote	Libelium	Atmel ATmega1281	147 456 MHz	X	8 KO	128 KO

Tableau 2.1 : Comparaison de quelques plates-formes de nœuds capteurs [11]

La figure 2.3 illustre une chronologie de fabrication des nœuds capteurs présentés dans le tableau 2.1.

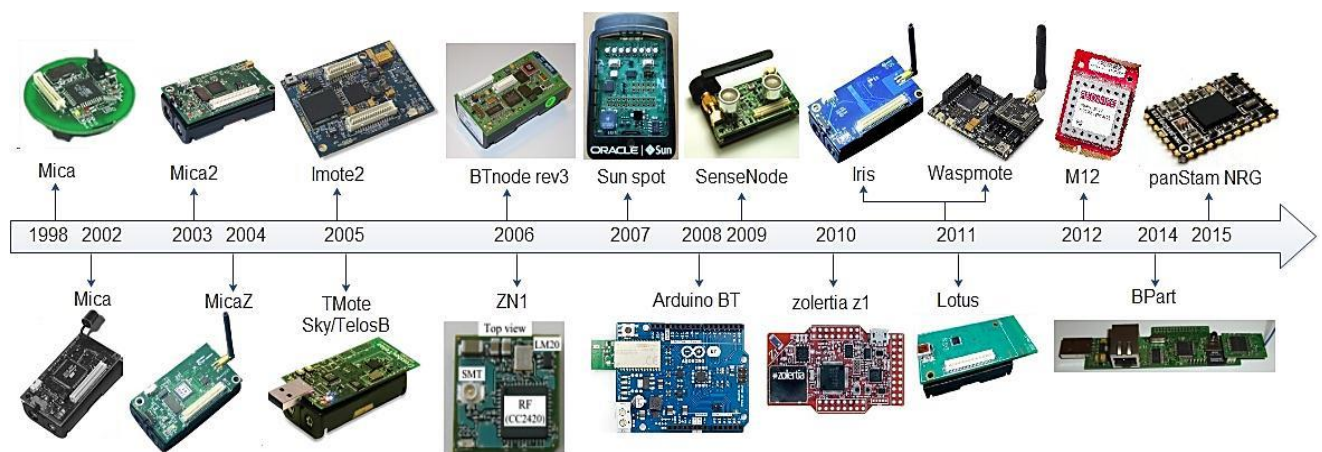


Figure 2.3 : Chronologie de développement des nœuds capteurs

Le dernier nœud capteur fabriqué – à nos connaissances – (qui n’est pas présenté ni dans le tableau 2.1 ni dans la présentation chronologique de la figure 2.3), est celui de l’université iranienne « *Kermanshah Zagros* » sorti en 2016. Ce nœud est composé d’un processeur de type Atmel Atmega128L avec 8-16 MHz comme fréquence d’horloge, 10 Ko de Ram, 48 ko de mémoire flash et une mémoire de donnée de 2 GB sur carte SD. Il communique les mesures capturées via deux modules, ZigBee de 802.15.4 (esp8266 chip & Nrf24l01) et Bluetooth (Mitsumi WML-C46N class 2). Il est doté de plusieurs capteurs, dont la température, la luminosité, l’accéléromètre, gyroscope, l’inclinaison et d’autres.

II.4. Défis de recherche

Dans chaque paradigme de recherche considéré nouveau, un ensemble de défis surgissent afin de répondre aux différents propriétés et paramètres de définition de ce nouveau domaine. Nous énumérons dans ce qui suit, quelques défis de recherches actuels dans le domaine des RCSF classés selon la pile protocolaire du modèle ISO proposés dans les travaux de recherches [12-14].

II.4.1. Niveau matériel

Les recherches dans ce domaine se concentrent principalement sur la miniaturisation des dispositifs de captures, ou bien les modules d’énergies qui les alimentent (les batteries), et à la discrétion des capteurs. Parmi les exemples sur le domaine de la santé, on trouve des études qui se concentrent sur l’implantation des nano-capteurs physiologiques miniatures qui devront être non invasifs, légers et fournissent une information précise (sensibilité et calibration des capteurs) sur la personne. Encore, de multitudes de mini-capteurs qui forment une nécessité presque absolue en vue de leur utilisation quotidienne qui s’intègrent dans la fabrication des nouveaux Smartphones, tablettes, montres et ordinateurs. Néanmoins, l’intégration de ces mini-capteurs pose encore des défis liés à la définition de protocoles et de mécanismes de communication adaptés qui prennent en compte les techniques de compression (limites de l’unité de traitement) et l’intégration des méthodes de récupération d’énergie [14].

II.4.2. Niveau physique

Cette couche doit assurer à la fois une modulation simple et robuste, elle doit fournir un schéma de codage qui améliore la fiabilité dans l’acquisition des données collectées. D’autres challenges sont liés aussi au support à ce niveau concernent les problèmes d’interférences et d’interopérabilité entre les nœuds (couplage entre les technologies sans fil hétérogène).

II.4.3. Niveau MAC

Les fonctionnalités demandées du protocole MAC pour résoudre les problèmes liés aux bruits et aux interférences sont responsables d'assurer et de gérer l'accès au médium et doivent réduire les collisions entre les nœuds capteurs qui se situent dans la même zone de couverture radio. Actuellement, et d'après [13, 14] plusieurs travaux de recherche comme [15] se focalisent sur le développement des protocoles MAC avec qualité de service qui permettent de garantir un délai acceptable de transfert des données critiques (alertes envoyées par un capteur de chute de tension par exemple) en prenant en compte la mobilité et la conservation d'énergie. Le moyen principal utilisé dans ces travaux consiste à réduire l'écoute inutile des capteurs et également à gérer l'accès au médium. Aussi sur des travaux qui se sont focalisés sur des protocoles MAC multicanaux avec interfaces multiples.

II.4.4. Niveau Routage

Un acheminement efficace en termes d'énergie et de délai pour le transfert des données critiques vers le Sink est assuré par la présente couche. Le défi dans un scénario de transfert de données entre les nœuds éloignés du Sink (dans un acheminement multi-saut), et alourdissent par conséquent les nœuds proches des nœuds émetteurs, consiste à rechercher des solutions de routage avec équilibrage de charges[16]. Ces solutions doivent intégrer des techniques d'agrégation et de contrôle du débit afin de garantir l'intégrité de données, et une plus grande durée de vie du réseau.

II.4.5. Niveau Transport

Compte tenu de taux d'erreur de paquets relativement importants pouvant être rencontrés lors de l'exploitation dans la bande ISM, cette couche doit assurer un transport fiable de bout en bout en assurant des protocoles de contrôle du transport et de la congestion adaptés aux exigences du système. La fiabilité de transfert de données devrait être assurée au niveau des couches supérieures par le biais d'accusés de réception et de retransmissions plutôt que par le biais d'un codage complexe de contrôle d'erreur au niveau de la couche physique. Les capteurs sans fil peuvent également tirer parti de la redondance du système pour fournir des performances robustes et réduire l'énergie de transmission. Du fait qu'un très grand nombre de nœuds capteurs sont déployés, ces derniers peuvent réduire ou combiner les informations provenant de plusieurs nœuds en fonction de leur emplacement afin de réduire et de compresser les exigences de transmission de données [17].

II.4.6. Niveau Application

À ce niveau, les défis sont principalement liés au traitement de l'information lors de développement d'applications afin de garantir l'organisation et la pertinence de l'information extraite des données

fusionnées (données utiles). En plus, le grand nombre de dispositifs coopérants envisagés et les exigences spécifiques en termes de coût et d'efficacité énergétique exigent des solutions révolutionnaires dans des domaines allant de l'intégration du micro fabrication à la mise en réseau auto-organisé et économe en énergie. Parmi ces solutions, on trouve l'intégration des systèmes multi-agents [17].

II.5. Les RCMSF

Un RCMSF (pour Réseau de Capteurs Multimédia Sans Fil) est un réseau ad hoc, formé d'un ensemble de nœuds capteurs autonomes dotés chacun d'une petite caméra, communiquant entre eux sans liaison filaire et sans l'utilisation d'une infrastructure établie ni d'une gestion de réseau centralisée. Le but d'un capteur est le plus souvent de surveiller une zone se trouvant dans son champ visuel, de prendre régulièrement des images, de les compresser (selon l'application), et de les relayer vers une station de base, à travers ses capteurs voisins qui se trouvent dans sa portée radio. Le même schéma de la figure 2.1 est considéré dans cette définition à la différence dans les capteurs qui sont des nœuds dotés d'une caméra. En général, les nœuds sont supposés être limités en ressources, à savoir, en énergie, en mémoire et en capacité de traitement, contrairement à la station de base qui possède toutes ses capacités pour manipuler et gérer les flux multimédias.

D'autre part, une autre architecture type des RCMSF proposée par I. F. Akyildiz et al. [18] est illustrée par la figure 2.4, où trois réseaux de capteurs (a, b, c) avec des différentes caractéristiques sont montrés, éventuellement déployés dans différents emplacements.

Le premier réseau (a) est un réseau à un seul niveau de capteurs vidéo homogènes. Quelques nœuds capteurs multimédias déployés possèdent des capacités de traitement très élevées et sont donc appelés des concentrateurs de traitement (*hub*). L'union de ces hubs constitue une architecture de traitement distribuée. Le flux multimédia collecté est relayé en mode multi-sauts vers une passerelle sans fil interconnectée à un concentrateur de stockage, qui est chargé de stocker le contenu multimédia localement pour une récupération future. Il est clair que des architectures plus complexes pour le stockage distribué peuvent être mises en œuvre lorsque l'environnement et les besoins de l'application le permettent, ce qui peut entraîner des économies d'énergie puisqu'en stockant localement le contenu multimédia qui n'a pas besoin d'être retransmis vers des emplacements distants en mode sans fil. La passerelle sans fil est connectée aussi à une station de base (*Sink*), qui implémente l'interface logicielle pour l'interrogation et l'attribution de tâches du réseau.

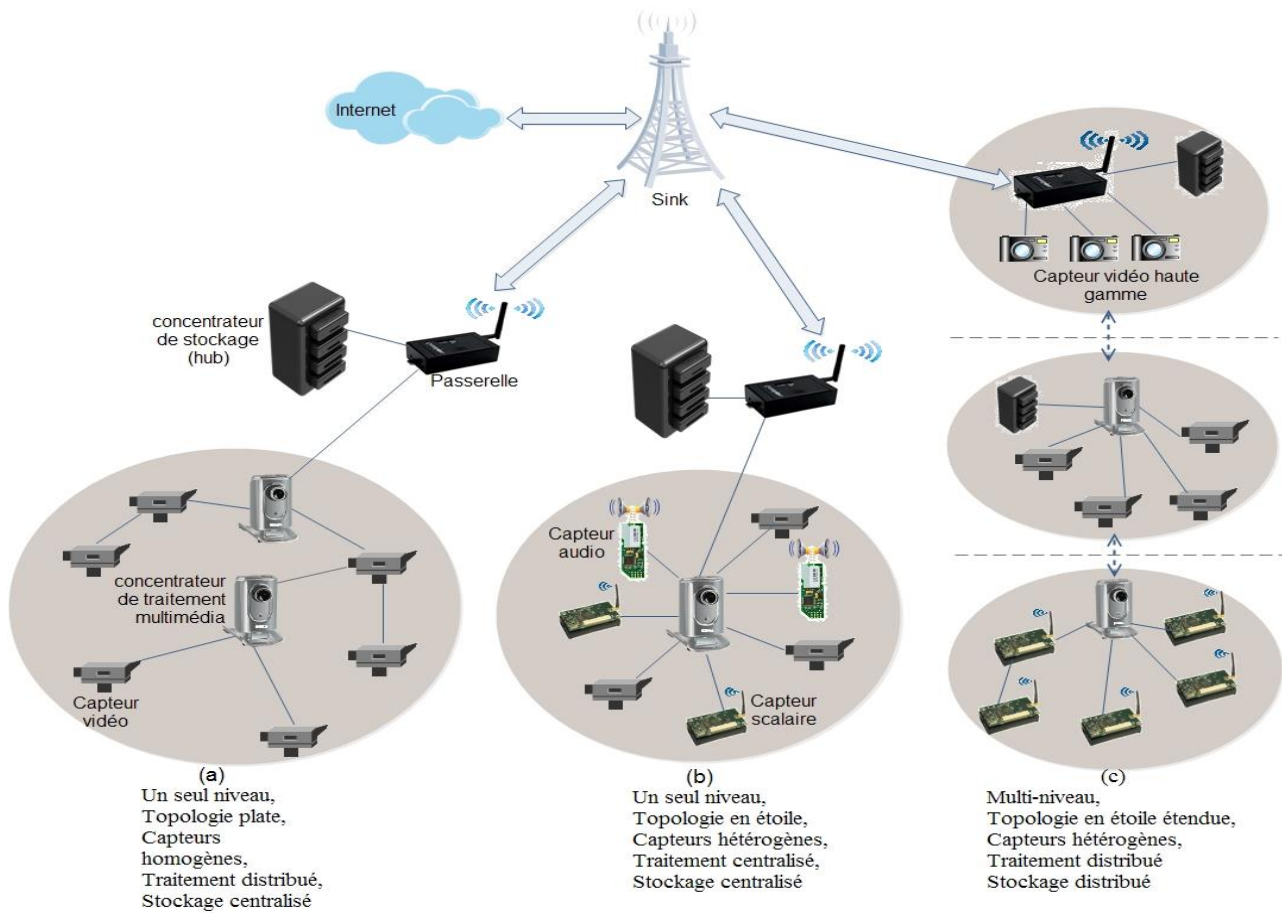


Figure 2.4 : Architecture type d'un RCMSF proposée dans [18]

Le deuxième réseau (b) représente une architecture en cluster à plusieurs niveaux de capteurs hétérogènes (dans la figure 2.4 un seul cluster est représenté). Les capteurs vidéo, audio et scalaires relaient les données vers un cluster-chef centrales, qui est également chargé d'effectuer un traitement multimédia intensif sur les données (concentrateur de traitement). Le cluster-chef relaie les données collectées vers la passerelle sans fil et vers le concentrateur de stockage.

Le dernier réseau (c) représente un réseau à plusieurs niveaux avec des capteurs hétérogènes. Chaque niveau est chargé d'un sous-ensemble des fonctionnalités. Les capteurs scalaires de faible puissance et à ressources limitées sont chargés d'effectuer des tâches plus simples, telles que la capture de mesures physiques scalaires, tandis que les dispositifs riches en ressources et à haute puissance sont responsables d'assurer des tâches plus complexes. Le traitement et le stockage des données peuvent être effectués de manière distribuée dans chaque niveau.

II.5.1. Capteur vidéo

Un capteur caméra [19] est une surface (plaque) sensible qui, grâce à ses millions de pixels, va recevoir de la lumière en provenance de l'objectif de l'appareil. Il va alors transformer cette lumière en signaux électriques (courants) qui vont être convertis en images numériques à l'aide d'un

processeur ; et stockés dans une mémoire (tampon) ou dans une file d'attente (comme FIFO) pour des futures manipulations. Le capteur caméra est donc l'équivalent de la pellicule (film) utilisée dans le photographique classique.

Il existe deux grandes familles de capteurs de caméras, des capteurs CCD et des capteurs CMOS.

- Inventé dans les laboratoires Bell en 1969 par Willard Boyle et George E. Smith, le capteur *Charge-Coupled Device* (CCD) [19] est connu en français sous le nom de « Dispositif à Transfert de Charge (DTC) ». Il existe trois types de capteurs CCD : le *CCD Full Frame* (plein cadre), le *CCD Full-Frame Transfer* (à transfert de trame), et le CCD « Interligne ».
- *Complementary Metal Oxide Semi-Conductor* communément appelé capteur CMOS [20], il a été inventé à peu près à la même époque, mais il n'a pas été tout de suite adopté. Car, à l'époque, la qualité d'image des capteurs CCD était meilleure ; ce qui n'est plus le cas aujourd'hui.

Les deux types de caméras sont équipés d'un capteur de type « tri » ou « mono ». Dans un capteur mono CCD/CMOS, les photosites sont recouverts d'un filtre bleu, rouge, ou vert, mais la sensibilité d'un photosite au vert est beaucoup moins importante, ce qui nécessite 2 photosites pour le vert. Par conséquent, pour obtenir une couleur, il faut 4 photosites : 2 pour le vert, un pour le rouge, et un pour le bleu. Dans ce mode, plus la quantité de photosites sur un capteur est importante, plus le nombre de pixels est grand. Or, pour des raisons de prix, dans les entrées de gamme, on réduit la taille des capteurs, par conséquent, quand il y a augmentation du nombre de photosites sur une surface plus petite de capteurs, il y a baisse de la qualité de l'image. Mais un capteur tri CCD/CMOS est composé de trois capteurs qui captent chacun de façon indépendante l'une des couleurs : rouge, bleue et verte, pour cet effet la définition sera plus précise, et l'image mieux restituée.

Aujourd'hui, en termes de comparaison entre les deux familles de capteurs, les capteurs CMOS n'ont presque plus d'inconvénients par rapport aux capteurs CCD de plusieurs points de vue. Les capteurs CMOS offrent une vitesse d'acquisition, consommation d'énergie faible, élimination de l'équilibrage des tapes, et moins de composants électroniques (techniquement, le capteur CMOS et les circuits électroniques peuvent être montés en un seul système).

II.5.2. Nœud capteur multimédia

Dans cette section, nous examinons et classifions les nœuds capteurs multimédias qui apparaissent dans une application de RCMSF. En particulier, nous discutons le matériel existant, avec un accent particulier sur les dispositifs de capture vidéo, passons en revue les implémentations existantes des

réseaux de capteurs multimédias, et discutons des possibilités actuelles pour la récupération d'énergie pour les nœuds capteurs multimédias.

Des caméras à zoom panoramique de haute gamme et des caméras numériques à haute résolution sont largement disponibles sur le marché. Cependant, alors que ces appareils sophistiqués peuvent être utilisés en tant que réseaux de capteurs multimédias de haute qualité, néanmoins les ressources limitées dont disposent les nœuds capteurs « sans fil » en tant que capacité mémoire, traitement et énergie n'autorisent en effet que les modules caméras dont les caractéristiques sont compatibles à ceux qu'on a présenté plus haut dans le tableau 2.1. Pour cela, nous nous concentrons sur des dispositifs d'imagerie et de traitement à faible coût et à faible consommation d'énergie qui fourniront des informations visuelles détaillées à partir de plusieurs points de vue, et assureront donc une meilleure interaction avec l'environnement.

Plusieurs études [21, 22] ont présenté quelques plateformes de carte de capture multimédia (nœuds multimédias ou capteurs d'image) existantes dans la littérature. Ces plateformes sont destinées principalement à être intégrées dans les téléphones portables, les ordinateurs portables, et les *PDA*. Deux catégories de ces plateformes peuvent exister, la première présente des caméras qui sont basées principalement sur des composants commerciaux, et la deuxième catégorie conçue spécialement pour les RCSF.

II.5.3. Caméras basées sur des composants commerciaux

Sous le sceau de la commercialisation, plusieurs captures d'images ont été développées pour répondre aux besoins clients afin de produire leurs plateformes multimédias. Dans ce qui suit, nous présentons quelques exemples de prototypes proposés pour ce genre de caméra en utilisant des composants électroniques commerciaux :

II.5.3.1. MeshEye

C'est une plateforme de vision (capteur d'image) à deux couches [23]. Elle est composée d'un système de vision stéréo à basse résolution pour déterminer la position, la portée et la taille des objets cibles qui sont en mouvement, et d'une caméra couleur haute résolution. MeshEye intègre un microcontrôleur Atmel AT91SAM7S, une mémoire flash MMC/SD, deux capteurs d'images ADNS-3060, une caméra VGA ADCM-2700 et un module radio CC2420 de Texas Instruments conformes à la norme 802.15.4 utilisés pour la connexion sans fil. Ce nœud est alimenté via une interface mini-USB ou par des piles standard de types AA.

II.5.3.2. Cyclops

C'est une caméra développée en collaboration entre les deux laboratoires Agilent et CENS (*Center for Embedded Network Sensing*) de l'UCLA [24]. Cyclops est composée d'un module de capture d'images de type CMOS (Agilent ADCM-1700), un processeur ATmega128L de Atmel avec 128 Ko de mémoire flash pour le stockage du code de l'application et 4 Ko de mémoire SRAM, une mémoire SRAM avec 64 Ko et une mémoire flash pour le stockage de données. Cette caméra permet de capturer et de traiter des images de faible résolution avec une consommation d'énergie relativement faible. Elle possède un connecteur de 51 pins lui permettant d'être connectée sur des nœuds de type Mica2 ou MicaZ. Cyclops1, Cyclops2A, Cyclops2B et Cyclops2C, sont les quatre versions développées.

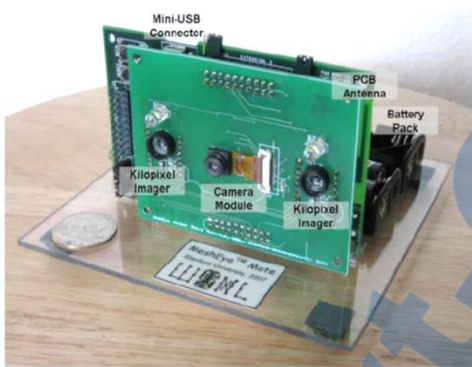


Figure 2.5 : Plateforme MeshEye [23]

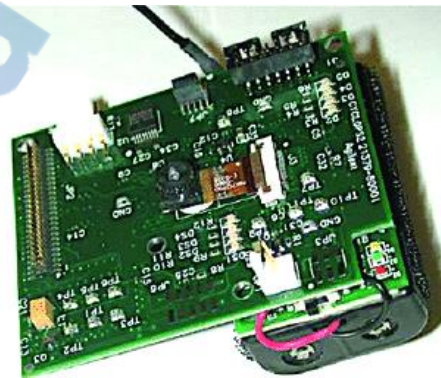


Figure 2.6 : Caméra cyclops attachée à un nœud capteur Mica2[25]

II.5.3.3. CMUcam

C'est une plateforme de vision embarquée programmable développée à l'institut de robotique de l'Université Carnegie Mellon pour certaines plateformes, et en collaboration avec le laboratoire *Charmed Labs* pour d'autres comme le cas de CMUcam5 Pixy [26]. CMUcam1, CMUcam2, CMUcam3, CMUcam4 et CMUcam5 Pixy sont les cinq versions de cette plateforme qui sont fabriquées entre 2008 et 2014. La variation entre la première version et la dernière réside dans l'amélioration des plateformes et enrichir la base de la connectivité avec différents nœuds capteurs. Sa dernière version (CMUcam5 Pixy) est basée sur un processeur NXP LPC4330 à une vitesse d'horloge de 204 MHz en double corps, elle comprend une caméra Omnivision OV9715 avec 1280x800 de résolution, des mémoires RAM de 264 Ko et flash de 1Mo. Elle communique via les ports UART serial, SPI, I2C, USB. Les figures 2.7 et 2.8 montrent la plateforme CMUcam3 et CMUcam5 Pixy.



Figure 2.7 : CMUCam3 attachée à TMoteSky[26]

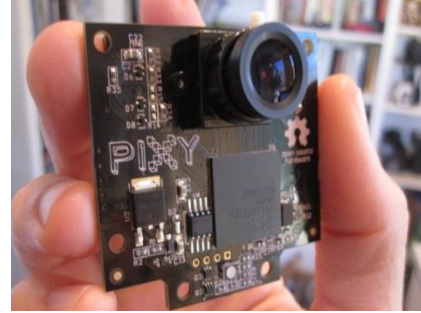


Figure 2.8 : CMUCam5 Pixy [26]

II.5.3.4. IMote

Cette plateforme de vision pour RCSF se compose d'une interconnexion directe entre le nœud Imote2, et un capteur d'image omnivision OV7649 via une interface série [17]. Ce dernier peut capturer des images couleur VGA (640x480) à 30fps et QVGA (320x240) à 60fps. IMote2 est basée sur le processeur PXA271 XScale sous fréquences 13/104/208/312/416 MHz, et des mémoires SRAM de 256 KO, SDRAM de 32 Mo et flash de 32 Mo. Elle communique en utilisant la radio CC2420 en mode Zigbee (IEEE 802.15.4). Le port Mini-USB permet à la fois d'interfacer la plateforme avec le programmeur, et de l'alimenter.



Figure 2.9 : Plateforme basée iMote2 [17]



Figure 2.10 : Stargate attaché avec Webcam [27]

II.5.3.5. Stargate

Stargate est une plateforme développée par Intel et vendue par *CrossbowTM*, elle fournit une résolution d'imagerie moyenne [27]. La diffusion directe de la vidéo n'est possible avec cette carte, car elle n'a pas d'interface sans fil, mais elle peut être connectée à un nœud capteur ou à une carte Wi-Fi. Le processeur utilisé dans Stargate est un Intel PXA255 d'une fréquence de 400 MHz. Stargate utilise 64 Mo de mémoire SDRAM et 32 Mb de mémoire flash. Elle dispose d'un port GPIO de 51 pin qui lui permet d'être connecté à un nœud de type micaz ou mica2, aussi, cette carte

permet de connecter via les ports I2C, carte Ethernet, RS-232, JTAG et le port USB. Une carte Stargate attachée à une Webcam est illustrée sur la figure 2.10.

II.5.4. Cameras conçues spécifiquement pour RCSF

Contrairement aux caméras basées sur des composants commerciaux, ce genre de capteurs d'image sont développés au sein des laboratoires de recherches spécifiquement pour les RCSF afin de répondre à un certain nombre de contraintes liées généralement sur-le-champ de visions, la résolution, la consommation d'énergie, l'allocation mémoire, etc. parmi ces dispositifs on trouve ALOHA :

ALOHA [20] est un capteur d'image de type CMOS (illustré dans la figure 2.11), conçu spécifiquement pour les RCSF. Il intègre une représentation de l'information numérisée en se basant sur des événements. Ce mécanisme décrit le mode d'accès multiples au bus de données (technique d'accès ALOHA) après exécution des événements si des pixels individuels atteignent un seuil de tension déterminé. Ce dernier (le pixel) envoie une requête à un circuit spécifique pour lui permettre de transmettre sa valeur en manipulant son adresse sur le bus, ceci se fait par activation d'une cellule ROM sur l'intersection des lignes et colonne.

Jusqu'à présent, deux versions du capteur d'images ALOHA ont été développées :

- ALOHAim1, un capteur d'image disposant d'une matrice de 32x32 pixels et d'un circuit détecteur de contention (ou collision) analogique.
- ALOHAim2 qui dispose d'une matrice de 64x64 pixels (4 quadrants de 32x32 pixels) et d'un détecteur de collision numérique.



Figure 2.11 : Un capteur d'image ALOHA attachée à un mote Mica2 [20]

La consommation électrique de ALOHAim1 est d'environ $115\mu\text{W}$ pour une alimentation numérique et d'environ $680\mu\text{W}$ pour alimentation en analogique. Pour ALOHAim2, la consommation d'énergie atteint environ $5.72\mu\text{W}$ par quadrant.

Dans la littérature, d'autres plateformes de vision ont été construites (qui ne sont pas tous décrites dans cette thèse), qui peuvent être utilisés pour les mêmes besoins que celui qui sont présentés

avant, nous citons parmi celles-ci : Citric, WiCa, SeedEyes, Eye-RIS, Panoptes, FireFly, ArduCam, etc.

II.6. Caractéristiques et contraintes d'un RCMSF

Les données multimédias (son, image, vidéo) nécessitent certaines contraintes de QoS (*Quality of Services*) qui devient une nécessité fondamentale pour la transmission dans un environnement pauvre en ressources. Plusieurs facteurs à prendre en considération influencent la conception de ces réseaux. Ils constituent la base de développement de nombreux protocoles de communication. Ci-après nous résumons les principaux facteurs en focalisant sur les critères de performances importants pour tout type d'applications. La figure 2.12 montre les exigences de conception des RCSF et RCMSF.

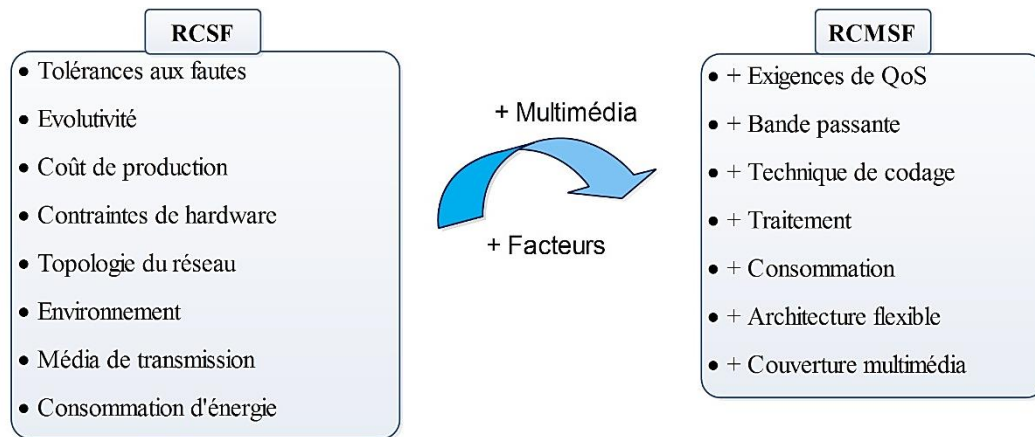


Figure 2.12 : Facteurs de conception des RCSF et RCMSF

Nous présentons ci-dessous, un aperçu sur quelques facteurs – les plus importants et fréquemment utilisé – pour évaluer les performances des RCSF en général et les RCMSF en particulier [23, 24, 28, 29].

II.6.1. Consommation et efficacité énergétique

Comme dans le cas des RCSF, la consommation d'énergie est une préoccupation majeure dans les réseaux RCMSF. Dans certaines applications (comme la surveillance de l'environnement par exemple), les réseaux RCMSF sont déployés dans des régions éloignées et inaccessibles – généralement – (montagnes, déserts, forêts et zones rurales) afin de collecter des informations multimédias pour une durée souhaitée d'être prolongée. Les capteurs des réseaux RCMSF (fonctionnant sur batteries irremplaçables) consomment généralement encore plus d'énergie par rapport aux RCSF scalaires. L'énergie consommée par un nœud capteur est la somme des petites énergies consommées par chaque composant qui le constitue (radio, traitement, capture, ... etc). L'énergie consommée par un composant est la somme des petites consommations dissipées dans

chaque cycle de fonctionnement de ce composant, ainsi que du nombre de cycles dont ce composant a besoin pour exécuter cette tâche. Les paramètres qui influent sur la consommation d'un nœud sont : la puissance du signal, le temps d'émission et de réception, le temps de traitement et la fréquence d'échantillonnage. Les principales sources de surconsommation recensées dans la littérature [18, 30] sont :

- ✓ Les collisions, les retransmissions.
- ✓ L'écoute abusive « *overhearing* » : un nœud reçoit des données qui ne lui sont pas destinées.
- ✓ « *L'overemitting* » : un nœud destinataire n'est pas prêt à recevoir des paquets qui lui sont destinés.
- ✓ L'envoi des paquets de contrôle « *overhead* ».
- ✓ La perte des liens du réseau (nécessité d'auto-organiser le réseau).
- ✓ Le phénomène d'autodécharge des piles.
- ✓ L'écoute à vide « *idle listening* » : les nœuds restent actifs en absence de données transférées.

En effet, les applications multimédias produisent/manipulent des volumes de trafic élevés, qui nécessitent non seulement des débits de transmission élevés, mais également des ressources de traitements et de stockages importantes. De ce fait, lors de la conception d'une application il est indispensable d'étudier la complexité de l'algorithme à implémenter dans les nœuds capteurs afin de gérer de manière intelligente les ressources énergétiques. Actuellement, plusieurs recherches ont focalisé leurs intérêts sur les méthodes d'extraction de l'énergie de l'environnement tels que : la lumière, la vibration mécanique, le bruit acoustique, énergie solaire, ondes radio [10, 21]. Cette tendance – rechargement par extraction d'énergie – a donné une extension à la durée de vie du réseau, toutefois, il demeure important de dresser ces techniques afin d'améliorer la quantité produite vu le volume limité généré par ces méthodes. Bref, une basse consommation d'énergie des nœuds reste une caractéristique de la plus haute importance.

II.6.2. Bande passante

La bande passante dans la transmission de données varie selon le contenu envoyé, le flux multimédia nécessite une bande passante très élevée. En particulier, un streaming vidéo peut nécessiter des débits supérieurs à l'ordre de 500 kbps. Le débit maximal d'un nœud capteur est de 250 kbps avec le standard 802.15.4. Des études [22] ont montré que le débit réel utile est de l'ordre de 70 kbps. Par conséquent des solutions matérielles et logicielles sont nécessaires pour une fourniture suffisante en bande passante afin de supporter la transmission de flux multimédia.

II.6.3. Scalabilité

Les applications de surveillance dans des environnements hostiles peuvent nécessiter un déploiement très dense des capteurs multimédia. Lorsque le nombre de capteurs augmente, cela permet d'assurer et d'améliorer la qualité et la fiabilité des mesures, et il ne doit en aucun cas influencer sur les performances du réseau. Les différents algorithmes et protocoles (en particulier les protocoles d'accès au médium MAC et les protocoles de routage) proposés pour les RCMSF doivent prendre en considération un contrôle sur l'évolutivité (surcharge) du réseau.

II.6.4. Tolérance aux pannes

La tolérance aux pannes est la capacité de soutenir les fonctionnalités d'un réseau sans causer d'interruption lorsqu'un capteur cesse de fonctionner (dû à une faible énergie, ou une panne). Le RCMSF doit pouvoir remédier aux défaillances d'une partie de ses capteurs et continuer à fonctionner même avec des performances dégradées. Le bon comportement d'un capteur est lié principalement à trois aspects fondamentaux : la fiabilité des mesures collectées de la personne/environnement, la fiabilité des liens de communication inter et intra RCMSF, et la fiabilité de l'analyse des données collectées au niveau du Sink (pertinence de l'information extraite des données fusionnées).

II.6.5. Sécurité

La sécurité des RCMSF est résumée et groupée sur les quatre catégories des travaux présentés dans[31] :

II.6.5.1. Gestion efficace de la Qualité d'Expérience (QoE) et de la Qualité de Service (QoS)

L'exigence des applications multimédias pose de nouveaux problèmes liés à la communication sans fil et aux infrastructures, ressources limitées en énergie et en calcul, l'agrégation du traitement distribué, de la surcharge doivent être résolus pour gérer la confidentialité/sécurité et la qualité de service.

II.6.5.2. La confidentialité

Le type d'informations (sur les individus par exemple) peut spécifier des informations explicites, par conséquent, dans certaines circonstances, les données peuvent être utilisées pour violer la vie privée des individus. À cet effet, la confidentialité est une exigence clé pour de nombreux scénarios d'application des RCMSF. Les solutions de confidentialité, telles que le masquage sécurisé des données, la sécurisation des canaux de communication et la définition de stratégies de confidentialité ne suffisent pas pour fournir une solution de confidentialité complète aux RCMSF. Car, chaque solution satisfaite uniquement des exigences spécifiques des problèmes *ad hoc*.

II.6.5.3. Authentification et localisation d'un nœud

Dans les communications des RCSF, une taxonomie comprend six attaques est décrite dans[32]. L'authentification des données empêche les parties non autorisées de participer aux réseaux, et les nœuds légitimes doivent pouvoir détecter les messages des nœuds autorisés et les rejeter. L'authenticité de ces données et commandes est une condition critique pour le comportement correct d'un RCMSF. Le problème de l'authentification est strictement lié à la question de la localisation des nœuds sécurisés [27]. En raison de la nature distribuée des réseaux RCMSF, la localisation des capteurs multimédias est nécessaire pour assurer la fourniture des services. Par conséquent, l'intégrité et la confidentialité des informations de localisation sont fondamentales et il est nécessaire de définir les contre-mesures par rapport aux attaques malveillantes possibles.

II.6.6. Développement de la plateforme

L'intégration de solutions telles que les algorithmes d'agrégation, la technique de compression, la localisation sécurisée, les mécanismes d'authentification devrait être autorisées par la plate-forme d'expérimentation qui prend en compte la QoE, la sécurité, la confidentialité et les contraintes technologiques. La plate-forme de référence doit être hiérarchique où chaque niveau de la hiérarchie pourrait utiliser différents protocoles/algorithmes et technologies.

II.6.7. Couverture

La couverture peut être considérée comme l'une des métriques de la QoS d'un réseau de capteurs. La répartition des nœuds dans le champ de déploiement joue un rôle fondamental dans la communication entre les nœuds capteurs eux-mêmes et/ou entre les nœuds capteurs avec la station de base. La solution appropriée aux problèmes de couverture permet de disperser plus de nœuds capteurs dans le champ de déploiement, et de procéder à leur mise en veille en alternance afin de prolonger la durée de vie du réseau et d'assurer également une tolérance aux pannes dans le RCMSF.

II.6.8. Qualité des images capturées

La capacité à exploiter les informations visuelles d'une image collectée par les nœuds capteurs multimédias est primordiale. Afin de manipuler les données multimédias dans le réseau, nous devons faire appel aux techniques de compression d'images. Ces dernières peuvent s'effectuer de plusieurs manières : avec ou sans perte de données, centralisée ou distribuée. La compression avec perte de données est généralement choisie dans ce type du réseau vu les contraintes des nœuds capteurs qu'ils constituent. Un taux de compression élevé participe à l'amélioration des performances du réseau (énergie des nœuds relai et le Sink, et durée de vie du réseau par

conséquent), mais peut dégrader fortement la qualité de l'image restituée d'une part, [33] et les ressources énergétiques des nœuds capteurs de l'autre part.

II.7. Systèmes d'exploitation

Les systèmes d'exploitation dédiés aux réseaux de capteurs sont aujourd'hui embarqués dans le nœud capteur, cette tendance a permis de développer davantage du caractère intelligent de ces nœuds capteurs. À cet effet, plusieurs systèmes d'exploitation ont été développés, les plus connus dans ce domaine sont TinyOs et Contiki,

II.7.1. TinyOS

Est un système d'exploitation intégré, modulaire, open source destiné aux RSCF. Sa conception a été entièrement réalisée en NesC « langage orienté composant syntaxiquement proche du C », cette plate-forme logicielle ouverte est une série d'outils développés par l'Université de Berkeley et enrichie par une multitude d'utilisateurs. Cet OS est capable d'intégrer très rapidement les innovations en relation avec l'avancement des applications et des réseaux eux-mêmes tout en minimisant la taille du code source NesC en raison des problèmes inhérents de mémoire dans les réseaux de capteurs. Afin de permettre l'extension des capteurs par d'autres fonctionnalités, TinyOS est structuré en plusieurs composants avec interfaces bidirectionnelles [34].

Pour compiler une application sous TinyOs, trois fichiers sont nécessaires :

- Un fichier configuration qui détermine les interfaces proposées par le composant et les interfaces des autres composants nécessaires à ce dernier. Ce fichier doit aussi expliciter les connexions qui existent entre ces différents composants.
- Un fichier module qui contient l'implémentation des interfaces du composant, ainsi le code source de l'application.
- Un fichier *makefile* qui est compilé lors de la commande *make*, il fait appel à d'autres fichiers de compilation contenant les options de la carte de programmation, l'identifiant de groupe, la sélection de la bande radio et des canaux ainsi que la puissance d'émission.

Afin de prolonger la durée de vie du capteur [34], TinyOS opte plutôt pour une programmation événementielle, dans ce type de programmation, l'exécution des instructions s'effectue en fonction des événements enregistrés par l'unité de traitement. L'avantage de cette technique est de laisser les capteurs en mode veille dans le cas où il n'y a aucun événement.

Les capteurs qui utilisent le système d'exploitation TinyOS sont essentiellement développés par la société Xbow, comme les capteurs Mica2, MicaZ, TelosB, Imote2, etc.

II.7.2. Contiki OS

Est un système d'exploitation léger, flexible, conçu pour les RSCF, ces dernières années, le concurrent de TinyOS suédois (Contiki) est devenu populaire grâce aux principales caractéristiques et fonctionnalités d'un système d'exploitation tout en favorisant une consommation énergétique et une empreinte mémoire minimales. Écrit en langage C, Contiki [35, 36] est constitué d'un noyau, de bibliothèques, d'un ordonnanceur et d'un jeu de processus. Comme tout système d'exploitation, son rôle est de gérer les ressources physiques telles que le processeur, la mémoire, les périphériques d'entrées/sorties. Il fournit ensuite aux applications des interfaces permettant d'utiliser ces ressources. Contiki propose un simulateur de réseau appelé Cooja. Ce simulateur permet l'émulation de différents capteurs sur lesquels seront chargés un système d'exploitation et des applications. Cooja permet ensuite de simuler les connexions réseau et d'interagir avec les capteurs. Cet outil permet aux développeurs de tester les applications à moindre coût. Les capteurs supportés à ce jour par Cooja sont : Exp5438, Z1, Wismote, Micaz, Sky, Jcreate & Sentilla-USB, Esb. Pour simplifier le développement, Contiki propose un environnement de développement complet et fonctionnel nommé Instant Contiki, téléchargeable sous la forme d'une machine virtuelle VMware. Cette machine virtuelle contient un environnement de développement Eclipse et tout le code source de Contiki avec quelques exemples. Elle contient également le simulateur Cooja avec plusieurs exemples de simulations.

D'autres systèmes d'exploitation sont développés pour des mêmes fins, sont moins connus vu leurs utilisations réduites soit sur les nœuds capteurs, là où on trouve le système est conçu pour un ou deux types de capteurs comme le cas de Nano-PK, soit de point de vue amélioration de produit, là où on trouve des difficultés d'usage suite aux bugs non corrigés – généralement –. Voici une liste non exhaustive de ces systèmes :

II.7.3. SOS

Est un système événementiel (comme TinyOs), développé par l'université de Los Angeles en Californie (UCLA), écrit en langage C [37].

II.7.4. FreeRTOS

Est un système d'exploitation temps réel (RTOS) faible empreinte, portable, préemptif et Open source pour microcontrôleur. Il a été porté sur 33 architectures différentes. Créé en 2003 par *Richard Barry* et *Real Time Engineers Ltd*, il est aujourd'hui parmi les plus utilisés dans le marché des systèmes d'exploitation temps réel. Sa conception est volontairement minimaliste pour qu'il puisse être installé sur de petits systèmes. L'image binaire du noyau pèse entre 4 ko et 9 ko (4,3 ko compilés sur ARM7). Le kit minimal ne compte que quelques fonctions pour gérer les tâches et la

mémoire1. De plus, les files d'attente, les sémaphores et les mutex sont implémentés. Le noyau de la version 7.3.0 n'est composé que de cinq fichiers codés en langage C. Des sections en Assembleur permettent d'assurer la compatibilité de FreeRTOS avec les différentes architectures matérielles [37].

II.7.5. Mantis OS

C'est un système d'exploitation embarqué multiplateforme multithread dédié aux réseaux de capteurs, développé par l'université du Colorado (USA) et écrit en langage C. Mantis OS s'articule autour d'un modèle commandé par l'exécution de processus [37].

II.7.6. Nut/OS

Est un système d'exploitation modulaire, open source et temps réel pour les plateformes embarquées. Il est facilement configurable et optimisé pour fonctionner sur des microcontrôleurs 8 et 32 bits. Il contient la pile réseau libre Nut/Net, qui fournit une API pour de nombreux protocoles réseau. Ainsi, Nut/OS est une plateforme idéale pour développer des applications pour l'Internet des objets [37].

Le tableau 2.2 résume quelques caractéristiques liées aux systèmes d'exploitation dédiés aux RCSF.

	TinyOS	Contiki	Mantis	Nano-RK	LiteOS
Publication (année)	ASPLOS (2000)	EmNets (2004)	MONET (2005)	RTSS (2005)	IPSN (2008)
Type	Statique	Dynamique	Dynamique	Statique	Dynamique
Modèle de programmation	Event & Thread (TinyThread, TOSThreads)	Event & Threads (Protothreads)	Thread & Event (TinyMOS)	Thread	Event & Thread (through callback)
Monolithique/Modulaire	Monolithique	Modulaire	Modulaire	Modulaire	Modulaire
Réseau	Message Actif	uIP, uIPv6, Rime	« comm »	socket	File-Assisted
Langage de programmation	nesC	C	C	C	LiteC++
Système de fichier	Un seul niveau	Coffee	non	non	Hiérarchique de type Unix
Reconfiguration	oui	oui	non	non	oui
Débogage distant	oui	non	oui	non	oui

Tableau 2.2 : Caractéristiques des systèmes d'exploitation dédiés aux RCSF

II.7.7. Nano-RK

Est un système d'exploitation temps réel basé sur les réservations (RTOS) entièrement préemptif. Conçu par l'université de Carnegie Mellon avec une prise en charge de routage multi sauts pour une utilisation dans les réseaux de capteurs sans fil. Nano-RK fonctionne actuellement sur la plateforme FireFly ainsi que sur les nœuds MicaZ [37].

II.8. Méthodes d'évaluation des performances

Dans ce qui suit, nous allons citer et détailler les différentes méthodes d'évaluation des performances utilisées dans les RCSF et résumée dans [38], à savoir : les méthodes analytiques, les méthodes basées sur la simulation, les méthodes basées sur l'émulation, les méthodes basées sur un banc d'essai et celles basées sur une expérimentation réelle. Dans un deuxième temps, nous montrons l'importance de la méthode par expérimentation sur laquelle repose notre thèse, et nous donnons des arguments forts pourquoi nous nous sommes orientés vers ce genre de méthode dans notre thèse considérée pratique et qu'elle valide – la méthode – sur le terrain des résultats recherches qu'on puisse les obtenir par simulation ou par émulation.

II.8.1. La méthode analytique

Nécessite certaines simplifications pour modéliser et prédire les performances. Cette méthode repose sur des modèles trop simplistes qui peuvent conduire à des résultats inexacts, ce qui n'est pas souhaitable dans le domaine de détection où le facteur de risque d'inflammation est élevé. Par conséquent, cette méthode est classée inappropriée pour ce domaine en raison de la complexité inhérente et aux défis liés à la nature des réseaux RCSF (densité de nœud, mobilité de nœud, topologie dynamique, caractéristiques du support sans fil, condition de canal variant dans le temps, etc.). Tester et évaluer des RCSF par le biais de la modélisation analytique est complexe, coûteux et prend du temps.

II.8.2. La simulation

La simulation est l'approche la plus populaire, la plus efficace et la plus réalisable pour la concevoir, développer et tester des protocoles de réseau pour les réseaux RCSF. Les simulateurs s'efforcent de modéliser et de prédire avec précision – suivant les paramètres de simulations – le comportement d'un environnement réel dans différents scénarios. Les approches basées sur la simulation offrent certains avantages tels que la réduction des coûts, l'évolutivité, le temps et la facilité de mise en œuvre. En règle générale, les modèles adoptés par les simulateurs peuvent ne pas être précis ; par conséquent, les résultats de la simulation peuvent ne pas être aussi précis que les implémentations réelles. Les simulateurs peuvent être classés en simulateurs d'usage général et spécialisés pour les RCSF.

a) Simulateurs à usage général : ce sont des simulateurs qui ne sont pas conçus pour les RCSF, mais offrent quand même des extensions spécialisées pour ce type de réseau. Certains des simulateurs à usage général les plus en vue, pertinents et fréquemment mis à jour utilisés pour les RCSF sont : NS-2 [39] /NS-3, OMNET++ [40], J-SIM [41], GloMoSim/QualNet [42], OPNET [43]...

b) Simulateurs spécifiques aux RCSF : En plus des simulateurs généraux, peu d'efforts ont été dépensés pour développer des simulateurs spécialisés pour les RCSF. Cela revient dans la plupart du temps au développement et à l'intégration des fonctionnalités et des bibliothèques supplémentaires dans les simulateurs de réseau polyvalents pour l'évaluation des performances des RCSF. Par contre, SensorSim [35], Castalia [36], VisualSense, Viptos, Sidh, Prowler/JProwler et SENS (Sensor, Environment and Network Simulator) sont quelques exemples des simulateurs créés spécialement pour l'évaluation des performances des RCSF.

D'autre part, et en plus de ce qui est cité plus haut, SENSIM, Mobility Framework, MiXiM sont des extensions développées pour le simulateur à usage général OMNeT ++ pour le même effet.

II.8.3. L'émulation

L'émulation est une approche hybride combinant les deux compléments matériels et logiciels, dans laquelle certains composants sont implémentés réellement sur du matériel (par exemple, des nœuds) et d'autres sont simulés (par exemple, les liens, le trafic...). Les émulateurs les plus connus qui peuvent être donnés comme exemple sont : TOSSIM [44], ATEMU [45], Avrora [46], SENSE (Sensor Network Simulator and Emulator), EmStar [47], sachant qu'il existe d'autres émulateurs dans la littérature pour les RCSF qui sont moins populaires tels que VMNet. Les approches basées sur l'émulation utilisent toutefois les mêmes mécanismes de simulation logiciels, mais présentent plus d'avantages par rapport aux approches basées sur la simulation dans la partie matérielle qui résout pas mal de problèmes techniques. Cela fait de ces approches une alternative de ceux qui se sont basés sur l'expérimentation réelle, et dévoilent une première solution proche de la réalité.

II.8.4. Banc d'essai

Les trois premières méthodes font appel parfois à des hypothèses irréalistes et à des modèles trop simplistes conduisent à des résultats inexacts. Pour résoudre ce problème, le banc d'essai est un prototype physique constitué d'un ensemble de nœuds capteurs qui s'efforce de combler le fossé entre la simulation – et/ou l'émulation – et le déploiement réel. Il fournit un environnement de test et d'évaluation de protocole similaire au déploiement réel. Le principal but derrière leur utilisation est de configurer, d'exécuter et de surveiller à distance des expériences tout en évaluant des modèles, des algorithmes, des protocoles et des applications avant d'aller à un large déploiement. Voici quelques exemples des environnements de banc d'essai physiques développés et testés sur

terrain : *MoteLab* [48], *SensorScope* [49], *Emulab* [50], *SignetLab* [51], *SENSORNET* [25], *ORBIT* [52],

Après cette courte description de chaque méthode, on s'intéresse maintenant aux critères ou règles à suivre afin de choisir la méthode la plus appropriée pour chaque domaine d'étude.

La première décision à prendre pour choisir un simulateur, un émulateur et/ou un banc d'essai dépend fonctionnellement des besoins et des contraintes. Chacune d'entre elles – méthodes – peut être appropriée dans une étape différente de la mise en œuvre, dans différents scénarios et dans différentes situations.

Le simulateur est un bon choix lors des premières phases de conception et de développement, car il fournit un niveau d'abstraction plus élevé. Et comme exemple explicatif où le choix de cette méthode est adéquat, c'est la conception et l'élaboration de protocoles de routage et les décisions de contrôle de topologie qui peuvent être prises sur la base d'une simulation. Il implique le coût le plus bas, il peut simuler un réseau avec des milliers de nœuds. Aussi, les contraintes liées au temps, aux efforts et aux ressources nécessaires pour la simulation sont petites. Par contre, les résultats générés peuvent ne pas être précis en raison du manque de modèles réalistes pour la radio, la mobilité, etc.

Bien que les simulateurs ont des caractéristiques liées à l'architecture et aux modes d'exécutions, ils constituent les cinq classes suivantes :

- Les simulateurs orientés objet simplifient la mise en œuvre et l'extensibilité grâce à une approche modulaire, mais ils manquent d'évolutivité.
- Les simulateurs à base de composants sont plus efficaces et offrent une extensibilité, une réutilisabilité et une évolutivité supérieures.
- Les simulateurs basés sur Java sont indépendants de la plate-forme, mais leur exécution est relativement lente.
- Les simulateurs commerciaux tels que OPNET offrent une bonne interface graphique, une aide et un support pour la modélisation de matériel spécifique au capteur, mais impliquent des coûts importants.
- D'autres simulateurs, tels que *GloMoSim*, *QualNet*... prend en charge la simulation parallèle à évènement discret pour les systèmes complexes, néanmoins, sans résoudre certains problèmes.

Enfin, généralement cette méthode – approche basée sur la simulation – reste non capable de garantir l'exactitude des résultats en raison d'hypothèses simplifiées prises comme supposition.

Les émulateurs sont plus efficaces et utiles, en particulier lorsque des bancs d'essai sont impossibles à déployer pour certaines applications de réseau de capteurs. D'autre part, du coup que les émulateurs exécutent le même code téléversé sur des nœuds réels, ils réduisent donc les efforts de mise en œuvre, et ils fournissent des résultats plus précis que celles générées par les simulateurs.

Aussi, la plupart des émulateurs sont basés sur une architecture à base de composants afin d'offrir une plus grande évolutivité. Néanmoins, le problème des émulateurs est qu'ils peuvent simuler du code pour des plates-formes spécifiques, par exemple l'émulateur TOSSIM ne peut simuler que des codes générés pour le même type que le nœud capteur, *EmStar* ne peut simuler du code que sur des nœuds MICA2 et *iPAQ* (basés sur micro serveurs). Les émulateurs populaires tels que TOSSIM sont incapables d'émuler des RCSF hétérogènes.

Les bancs d'essai constituent le choix le plus approprié, en particulier lorsqu'il est impossible de tester les performances dans un environnement réel en raison des coûts, de la durée et du terrain inaccessible. Il fournit un environnement de test et d'évaluation de protocole similaire au déploiement réel. Le but est de combler le fossé entre la simulation et le déploiement réel. Les résultats obtenus par les bancs d'essai sont plus précis que les outils logiciels existants tels que les simulateurs et les émulateurs. Les bancs d'essai sont développés pour l'évaluation des performances dans un environnement contrôlé, et accueillent toutes les informations sur la région d'intérêt afin de prendre des décisions et d'améliorer le modèle initialement tracé afin de remédier à des éventuelles lacunes lors de la première conception. Cette technique n'est pas assurée par les autres méthodes, puisqu'elles ne permettent pas de projeter les résultats dans la réalité. Malgré cela, les bancs d'essai ont une portée limitée et peuvent ne pas fournir un environnement réaliste dans certains cas surtout quand il s'agit d'un déploiement aléatoire pour contrôler une application critique. Le tableau 2.3 clarifie l'utilité de notre approche par rapport aux autres méthodes.

Méthode	Avantages	Inconvénients
Analytique	Se base sur des modèles mathématiques, simples à maitre en œuvre et à programmer.	Ne donne pas de résultats précis à cause de simplifications apportées au modèle initialement conçu.
Simulation	Idéale pour tester préalablement plusieurs versions de l'application, permet de simuler des réseaux denses, un coût négligeable dépensé.	Se base sur un ensemble de suppositions et contraintes dont ils peuvent altérer les résultats de la simulation.
Emulation	Combine les avantages de simulation avec une implémentation matérielle du nœud émulé, ce qui présente des résultats plus crédibles.	Comme la partie logicielle est simulée, les inconvénients de simulation se reportent sur la partie simulée.
Banc d'essai	Une image exacte de l'application est fournie par les capteurs programmés, relève les limites des autres méthodes, stimule la création de nouvelle application et conduise à la résolution d'autres problématiques	Nécessite un coût supplémentaire et une connaissance sur les plateformes d'expérimentations, et de programmation.

Tableau 2.3 : Comparaison entre les méthodes d'évaluation des performances

II.9. Applications des RCMSF

Les RCSF sont conçus afin d'être déployés dans divers endroits. Ce dernier (le champ de déploiement) est soumis à un ensemble de défis selon le type du réseau (sur terre, sous terre, sous l'eau). Il est formé de six groupes de famille des RCSF, à savoir : les réseaux de capteurs terrestres sans fil (WTSN), les réseaux de capteurs souterrains sans fil (WUSN), les réseaux de capteurs sous-marins sans fil (WUwSN), les réseaux de capteurs corporels sans fil (WBSN), les réseaux de capteurs mobiles sans fil (WMSN), les réseaux de capteurs multimédias sans fil (WMuSN) [53].

La figure 2.13 représente une vue de l'ensemble de différents types de réseau de capteur.

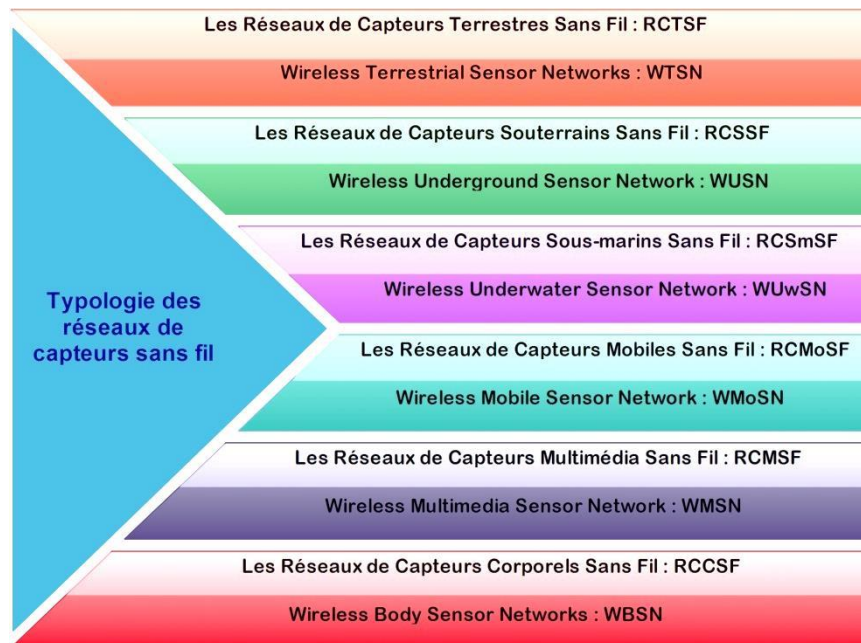


Figure 2.13 : Typologie des réseaux de capteurs sans fil

Aujourd'hui, les RCSF ont connu un intérêt très important dans plusieurs domaines d'application. Cette diversité peut être comprise et expliquée par les générosités d'opérations assurées par les nœuds capteurs (collecte/détection des informations, leurs traitement/agrégation – selon l'application – et communication des données via des liens sans fil pour atteindre l'objectif final – station de contrôle et de supervision –). Donc, ces trois modes d'opérations assurés par ces petits capteurs (détection, traitement et communication) donnent lieu à un grand nombre d'applications.

Deux catégories d'application peuvent exister, les applications de surveillance (monitoring) et celles de suivi (*tracking*) [27, 54]. La figure 2.14 illustre cette classification avec quelques exemples d'application sur chaque domaine. Le déploiement rapide, le coût réduit, l'auto-organisation et la tolérance aux pannes des réseaux de capteurs sont des caractéristiques qui rendent ces réseaux un outil appréciable dans les domaines d'application. Bien que le nombre d'applications par domaine

désormais s'agrandir chaque jour suivant l'évolution technologique des dispositifs de capture. Dans ce qui suit, nous détaillons quelques applications en donnant des exemples concrets :

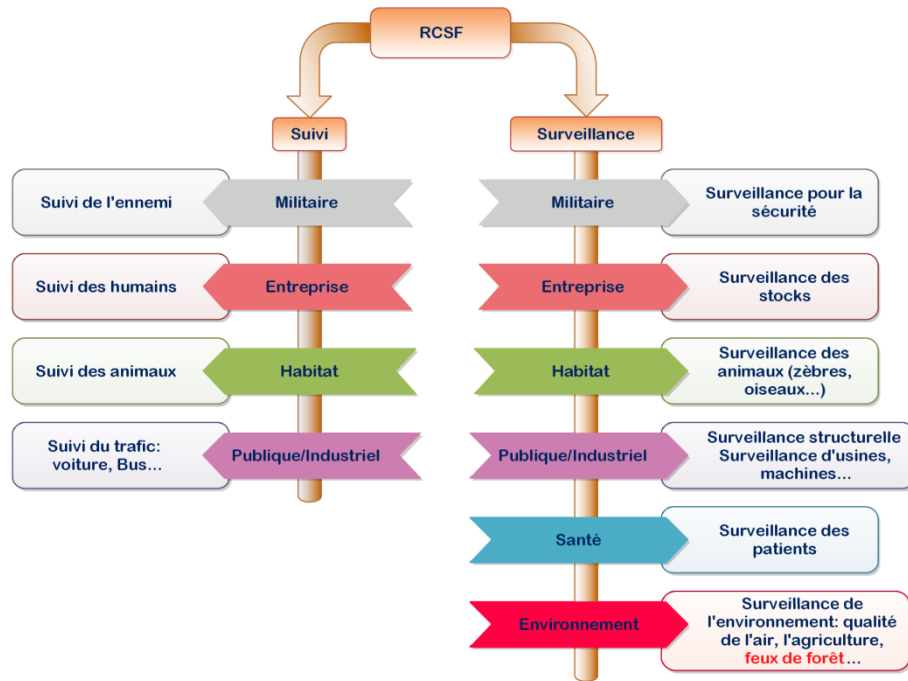


Figure 2.14 : Classification des applications des RCSF

II.9.1. La sécurité militaire

Les RCSF jouent un grand rôle dans les systèmes militaires en rendant les guerres « intelligentes », ils sont devenus une partie complémentaire du commandement militaire, du contrôle, de la communication et des systèmes de renseignement (C4ISRT [18] : *Command, Control, Communications, Computing, Intelligence, Surveillance, Reconnaissance and Targeting*) [55]. La surveillance des champs de bataille et des secteurs stratégiques avec un réseau de capteurs déployé permet par exemple de surveiller tous les mouvements (amis ou ennemis), ou d'analyser le terrain avant d'envoyer des troupes afin de détecter de potentiels dangers comme les agents chimiques, biologiques...

II.9.2. La protection des sites sensibles

La protection et la surveillance dans le domaine public et industriel permettent d'installer des capteurs appropriés à proximité d'objets et sites sensibles tels que : les centrales nucléaires, les ponts stratégiques, les pipelines de pétrole et de gaz. L'investissement dans ce domaine est beaucoup moins coûteux par rapport aux anciennes méthodes de surveillance et de protection, tout en gagnant aussi plus du temps pour la détection et la localisation d'un probable accident [56].

II.9.3. Le pilotage intelligent

Plusieurs domaines intègrent dans leurs solutions ce brillant point «pilotage intelligent». Le montage des capteurs sur des robots véhicules sans pilote, chars, avions de combat, drones, sous-marins, missiles permet de les guider pour éviter les obstacles et d'atteindre leurs destinations afin d'accomplir leur mission.

II.9.4. La surveillance environnementale

Dans les deux modes de son utilisation dans les RCFS – comme illustré dans la figure 2.14 – la surveillance de l'environnement présente une multitude d'applications dans lesquelles les capteurs sont utilisés pour mesurer, surveiller et suivre une variété de paramètres ou des conditions environnementales. L'agriculture de précision, la surveillance des volcans, la surveillance de la qualité de l'air, la météorologie, l'état du pergélisol dans les Alpes suisses [53] et les feux de forêt sont quelques exemples d'applications liées à l'environnement. Parmi les applications de surveillance de l'environnement citées, celle des feux de forêt qui fait l'objet de cette thèse dont nous allons la détailler plus dans les suivants chapitres.

II.9.5. Surveillance et suivie médicale

La technologie RCSF a révolutionné les applications de soins, de la santé et du suivi des patients et des personnes âgées. Un déploiement des capteurs autour d'un patient et plus particulièrement des personnes âgées peut contrôler leurs comportements et alerte le médecin traitant dès que le comportement nécessite une attention immédiate. D'autre part, les capteurs intégrés dans le corps permettent de récupérer à distance les signes vitaux et des paramètres physiologiques des patients, et offrent un suivi long terme, non invasif et ambulatoire des maladies chroniques en connaissant leurs positions géographiques. La réception des rapports instantanés de l'état actualisé des patients par les médecins, permet de mettre à jour le dossier médical des patients et de leur donner des conseils et recommandations médicales. Cette avancée technologique a présenté un développement radical dans la suivie médicale, détection et traitement à temps en cas d'urgences, réduction des coûts de soins et de déplacement des patients – surtout les personnes âgées – et enfin une réduction considérable des erreurs humaines [57]. Récemment, et afin de nettoyer le corps des restes d'appareils, des chercheurs américains comme Tekla S. Perry [58] travaillent sur des capteurs médicaux qui peuvent se dissoudre dans l'organisme humain une fois leur tâche accomplie. Parmi les réseaux RCSF on peut citer le système basé sur des capteurs biomédicaux sans fil [53, 57] qui utilise un microcontrôleur de type «MSP430» fonctionnant avec le système d'exploitation «TinyOS» et communiquent avec la station de base suivant la norme IEEE 802.15.4/Zigbee. Aussi, un prototype de RCCSF pour la surveillance de la santé des patients décrits dans [29, 53] qui utilise

des nœuds compatibles IEEE 802.15.4 et des capteurs de mouvement et de mesure de l'activité cardiaque.

II.9.6. Applications publiques

Les RCSF sont largement utilisés pour fournir des solutions efficaces dans le domaine public. Plusieurs applications ont été développées pour automatiser, surveiller, contrôler et commander à distance des applications de parking, conduite, circulation et évacuation en cas d'un danger...

Le parking intelligent est connu aujourd'hui un intérêt accru par les conducteurs automobiles. Pour bénéficier de ce service, l'intéressé doit lancer une requête de recherche de place à l'aide de son Smartphone connecté au réseau de capteurs dispersé dans le parc en question, rapidement les capteurs lui indiquent les emplacements creux. Un Système de gestion de parking intelligent (SPARK : Smart Parking) est proposé dans [53, 59] pour la réservation de parking, surveillance de stationnement à distance, orientation automatique.

L'automatisation des feux de circulation, l'indication des points noirs – l'embouteillage – dans des tableaux d'affichage ou par communication avec les Smartphones des conducteurs et l'équilibrage de l'émission de la fumée dégagée par les voitures dans les villes, sont des applications actuelles où les capteurs jouent le rôle principal dans tous les scénarios en communiquant périodiquement les différents paramètres pour le traitement de chaque problématique. D'autres recherches [53, 60] dans le domaine de RCSF sont celles de RCSF intra-véhicule, essaient de remplacer les capteurs câblés et leurs câbles par des capteurs sans fil afin de réduire le poids des véhicules et de diminuer leur consommation d'énergie.

II.10. Conclusion

Nous avons présenté dans ce chapitre notre domaine d'étude les RCSF. Nous avons défini en premier lieu la technologie des RCSF, et en particulier l'unité de base fondamentale le nœud capteur. Nous avons cité ensuite les défis de recherche liés à la technologie des RCSF qui entraînent ce type de réseau de s'implémenter. Encore, nous avons approfondi dans ce domaine en présentant la partie multimédia de cette technologie – RCMSF – ainsi ses caractéristiques, les systèmes d'exploitation dédiés à ces nœuds capteurs et quelques applications pratiques en utilisant les RCSF. Une étude sur le choix de ce domaine est faite selon une évaluation des performances des outils utilisés pour la validation de ce choix, en précisant la méthode la plus appropriée pour notre sujet pratique.

CHAPITRE III

Les feux de forêt et les méthodes de détection

III.1. Introduction

Aujourd'hui, les congrès et les conférences internationales sur la recherche environnementale concluent leurs forums avec des recommandations intensifs sur la préservation de l'écosystème et de lutter au maximum contre les émissions brutales des gaz à effet de serre [54]. Aussi, ils instiguent les participants à lancer des campagnes de boisement, de créer des espaces verts et de contrôler les biens forestiers à grande échelle. La communauté biologique constituée des plantes, des animaux, de l'homme et de l'environnement représente l'écosystème dont il fallait le préserver et le maintenir contre d'éventuelles perturbations afin de soutenir le bien-être de la palette de l'homme en conséquence.

Dans ce chapitre, nous allons définir l'acteur principal engendrant l'émission des gaz à effet de serre en représentant numériquement sa gravité, nous étudions les différents mécanismes proposés pour détecter et surveiller les feux de forêt, en se focalisant sur le moyen de détection. Nous présentons également les différents systèmes de détection des feux de forêts basées sur les réseaux de capteurs sans fil. Enfin, nous présentons une modélisation détaillée du système de détection adapté pour notre territoire, à savoir, le système canadien.

III.2. Une forêt, un feu, un incendie de forêt

Une forêt ou un massif forestier est une étendue boisée, relativement grande, constituée d'un ou plusieurs peuplements d'arbres, arbustes et arbrisseaux, et aussi d'autres plantes indigènes associées. Les définitions du terme « forêt » sont nombreuses en fonction des latitudes et des usages (selon l'Organisation des Nations unies pour l'alimentation et l'agriculture FAO – *Food and Agriculture Organization of the United Nations*—) ; et elle définit la forêt comme « des terres occupant une superficie de plus de 0,5 hectare avec des arbres atteignant une hauteur supérieure à cinq mètres et un couvert arboré de plus de 10 %, ou avec des arbres capables d'atteindre ces seuils in situ. La définition exclut les terres à vocation agricole ou urbaine prédominante » [61]. Mais, pour notre étude, une forêt est simplement un regroupement d'arbres, de plantes ou un espace vert, qui dégagent des déchets (les matières dures) dits combustibles.

Les forêts représentent mondialement au total près de 4 milliards d'hectares, soit 30 % de la superficie des terres émergées selon la FAO [61].

À l'échelle réduite, les forêts et les maquis couvrent 4,1 millions d'hectares, soit un taux de boisement de 16,4 % pour le nord de l'Algérie et de 1,7 % seulement sur les régions sahariennes arides. Localement, la surface forestière de la wilaya d'Oran, toutes formations confondues, est de 41 258 hectares. Couvrant un taux de boisement de 20 % [62]. La figure 3.1 illustre une vue générale de l'espace forestier (zones vertes) de la wilaya d'Oran.

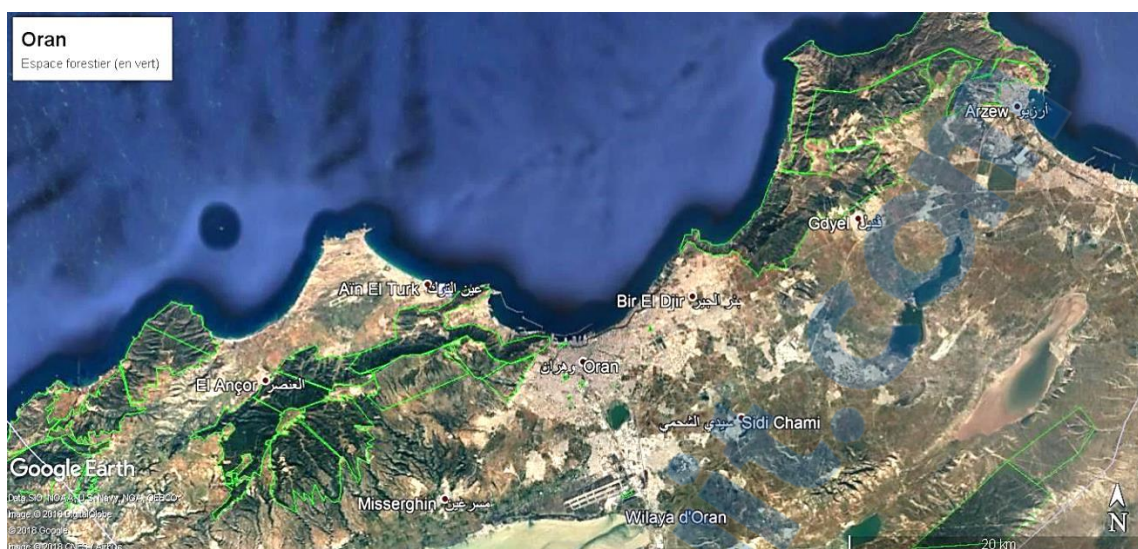


Figure 3.1 : Vue générale de l'espace forestier et de la région d'intérêt (image obtenue par Google map)

Le feu est défini comme étant un dégagement simultané de chaleur, de lumière et de la flamme produite par la combustion vive dans certains corps (bois, feuille, tapis herbacé... etc.). L'incendie est une combustion qui se développe sans contrôle dans le temps et l'espace, contrairement au feu qui est une combustion maîtrisée [63].

Un feu de forêt s'explique essentiellement par la présence dans le milieu, d'aire et abondance, de combustibles vulnérables et d'une source de chaleur produisant l'inflammation et la combustion. Ces trois éléments constituent la pyrologie forestière « le triangle de feu ». Si on retire l'un de ces éléments, le feu n'aura pas lieu ! [62, 64]

La conservation des forêts de la wilaya d'Oran a enregistré 38 foyers d'incendie durant la campagne de prévention et de lutte contre les feux de forêt en 2017. Ces foyers ont parcouru une superficie totale de $48 \text{ ha}^1 34 \text{ Ares}^2 10 \text{ Ca}^3$ (et $12 \text{ Ha} 65 \text{ Ares}$ en 2018). La moyenne calculée par incendie est de $01 \text{ Ha} 27 \text{ Ares} 21 \text{ Ca}$, et d'une fréquence d'un incendie tous les deux jours [62]. Un historique représentant la superficie incendiée et le nombre de foyers enregistrés durant les six dernières années, à savoir du 2013 à 2018 est donnée sur le graphe donné par la figure 3.2.

Selon la FAO [65], les feux de forêt détruisent chaque année jusqu'à 700 000 hectares en région méditerranéenne. Souvent, plus de 100 000 incendies éclatent durant la saison chaude. Dans certains pays, on enregistre plus de 20 000 feux de forêt par an. Mondialement, les surfaces

¹ Ha : Hectare, $1\text{Ha} = 10000 \text{ m}^2$

² 1 Acre = $4046,86 \text{ m}^2$

³ Ca : Centiare, $1\text{Ca} = 1 \text{ m}^2$

incendiées représentent environ 350 millions d’hectares chaque année (presque 6 fois la surface de la France), soit plus de 11 hectares par seconde !

Les incendies de forêt génèrent chaque année d’énormes catastrophes qui ont un impact négatif sur le plan économique, social et écologique, et qui affectent considérablement la vie humaine et les ressources forestières. Ces incendies dégagent selon WWF (*World Wide Fund for Nature*) quelle que soit la cause, « 20 % des émissions globales de gaz à effets de serre ».

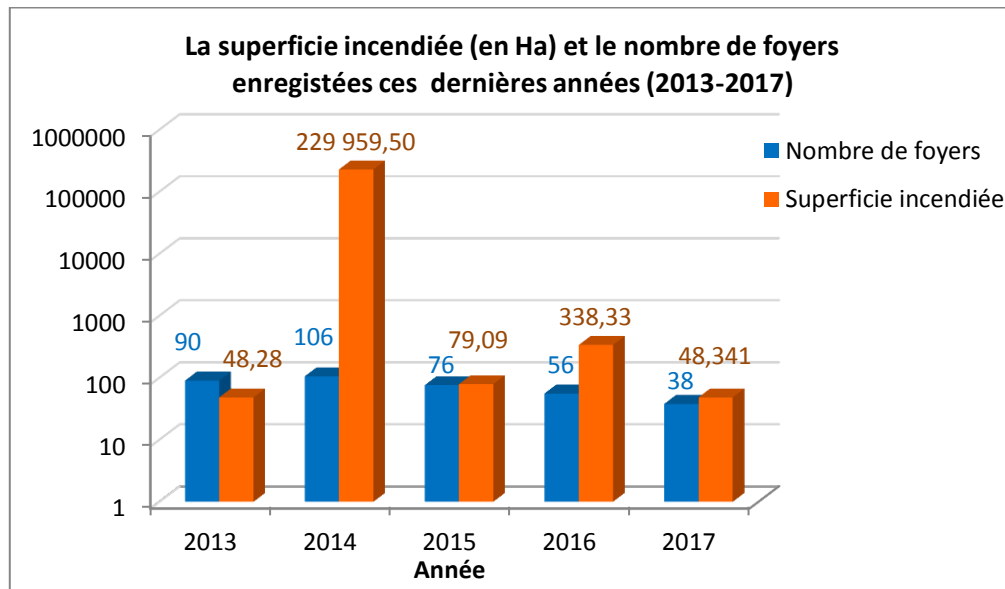


Figure 3.2 : Historique sur le nombre d’incendies et de superficie durant les cinq dernières années

III.3. Évolution des systèmes forestiers pour la détection d’incendie

Les mécanismes de détection des incendies de forêt sont classés en deux groupes, traditionnels et modernes. Les quatre grands axes qui forment les deux catégories sont présentés dans le tableau comparatif Tableau 3.1 :

Paramètres de comparaison	Systèmes de détection des incendies de forêt			
	Basés sur les observations humaines	Système satellite	Caméras optiques	Réseaux de capteurs sans fil
Coût	Faible	Très haut	Haut	Moyen
Efficacité et praticité	Faible	Faible	Moyen	Haute
Répétition des fausses alertes	Faible	Faible	Moyen	Moyen
Localisation précise du feu	Faible	Moyen	Moyen	Haute
Temps de détection	Longue	Très long	Longue	Court
Information sur le comportement du feu	—	Oui	—	Oui
Peut être utilisé à d’autres fins	Non	Oui	Non	Oui

Tableau 3.1 : Comparaison entre les méthodes de détection des feux de forêt

La première catégorie « traditionnelle » englobe les anciennes méthodes de détection, où la présence de l'être humain est indispensable dans toutes les opérations relatives à cette thématique « la détection », telles que la surveillance, la localisation et de l'extinction du feu en cas d'incendie. Ces méthodes sont utilisées actuellement dans tous les espaces forestiers à travers la planète, même dans les pays qui ont développé d'autres mécanismes plus avancées (classés dans la catégorie « actuelles ou modernes ») comme l'Australie, le Canada et les états unis américains, etc. Les méthodes de détection traditionnelles comptent sur le facteur humain pour composer des brigades mobiles, de surveiller dans des tours de garde et de piloter un avion de surveillance. Bien que ces méthodes fassent appel à des dispositifs d'aide à la surveillance comme les jumelles (pour rapprocher les zones éloignées), *Osborne Fire Finder* [66, 67] (permet de localiser l'incendie et d'indiquer sa direction par la carte topographique imprimée dans un disque gradué en bordure) et l'avion de surveillance. Les inconvénients de cette méthode se résument au niveau du manque de fiabilité des observations, et les conditions de vie difficiles pour la personne dans le milieu forestier, ou les conditions climatiques qui gênent la surveillance (nuage, les radiations solaires puissantes...).

Actuellement, la deuxième catégorie représente les avancées dans ce domaine. Commençons par les systèmes de vidéosurveillance automatique, généralement, les caméras de type CCD (*Charge Coupled Device*) et IR (*Infra Rouge*) dans ce système sont installées au sommet des montagnes et des tours. Plusieurs perfectionnements ont été apportés à ce système afin d'améliorer la qualité de détection et sa rapidité, comme la fabrication des caméras sensibles aux spectres visibles qui sont basés sur la reconnaissance de la fumée pendant le jour et la reconnaissance de la flamme pendant la nuit. Aussi, des caméras infrarouges thermiques qui sont basées sur la détection du flux de chaleur provenant d'un incendie, des caméras détectent et mesurent par laser la lumière (LIDAR) rétrodiffusée par les particules de la fumée. Sur le marché, plusieurs alternatives de ces caméras sont disponibles, à savoir, *FireWatch* (Allemagne), *FireHawk* (Afrique du Sud), *ForestWatch* (Canada), *FireVu* (Angleterre), *UraFire* (France). Le principal inconvénient de ces systèmes est le taux élevé de fausses alarmes dues aux conditions atmosphériques (nuages, réflexion de la lumière) et la fumée provenant des activités industrielles.

Les systèmes de détection basés sur l'imagerie par satellite sont aussi un moyen utilisé pour cet effet. Le capteur MODIS (*Spectroradi Imaging Resolution Spectroradiometer*) à résolution modérée embarqué sur les satellites *Terra* et *Aqua* de la NASA est utilisé pour analyser la surface de la Terre chaque jour depuis 1999 [68, 69]. Depuis 2012, VIIRS (*Visible Infrared Imaging Radiometer Suite*) embarqué sur le satellite *Suomi NPP* a contribué à cet effet en produisant des images à plus haute résolution de la surface de la Terre. Ces satellites ont des orbites similaires,

mais la résolution spatiale des bandes thermiques de leurs capteurs est différente. La bande thermique de MODIS a une résolution de 1 000 mètres par pixel, tandis que VIIRS a une résolution de 375 mètres par pixel. La résolution élevée du VIIRS lui a permis de détecter des incendies sur lesquels MODIS les ignore. Malgré cela, ce dernier fournit des images d'arrière-plan plus nettes puisqu'il offre une résolution spatiale de 250 mètres pour les bandes non thermiques, et il peut produire des images plus détaillées de la surface terrestre. Néanmoins, VIIRS est mieux classé pour surveiller l'activité des feux, et il permet aux scientifiques et aux pompiers de modéliser et de prévoir avec plus de précision les changements de comportement d'un incendie. D'autres satellites lancés par la NASA pour diverses missions peuvent aussi détecter les feux de forêt pendant leurs missions. Parmi ces satellites on trouve [70] :

- AVHRR (*Advanced Very High Resolution Radiometer*) prévu comme satellite de météorologie, mais les bandes visibles de ses capteurs peuvent détecter la fumée et les contours d'incendies. La bande thermique infrarouge peut détecter les points chauds et les incendies actifs.
- GOES (*Geostationary Operational Environmental Satellites*) assure toutes les 15-30 minutes une couverture de l'hémisphère et capture des images à une résolution de 1 km par le capteur de luminosité visible afin de détecter de la fumée, et à 4 km par les quatre capteurs thermiques infrarouges pour détecter la chaleur d'un feu.
- Landsat est une série de satellites à haute résolution (jusqu'à 15 m par pixel) qui surveillent et observent les caractéristiques et les changements sur la Terre.
- TOMS (*Total Ozone Mapping Spectrometer*) produit un ensemble de données sur les différentes couches d'ozone, qui permettent d'observer une variété d'événements sur la Terre, notamment des incendies de forêt, des tempêtes de poussière et incendie de biomasse (*masse de matière vivante, animale ou végétale, présente sur la Terre*).
- TRMM (*Tropical Rainfall Measuring Mission*) son but principal est de mesurer les précipitations sur la terre et l'océan. Il peut aider à identifier les zones qui peuvent être exposées aux d'incendies.

Finalement, les réseaux de capteurs sans fil (RCSF) ont conduit ce domaine à une large évolution en proposant plusieurs méthodes et algorithmes sur le plan de détection d'incendie et en prévention. Cette méthode est considérée comme étant la meilleure solution dans ce domaine, pour les raisons suivantes :

- Les RSCF peuvent fournir toutes les informations requises qui influencent sur l'environnement à tout moment avec précision.
- Les RSCF peuvent couvrir une grande zone, même si on ajoute des extensions (nouveau réseaux).
- Les RSCF peuvent observer et influencer le monde physique autour d'eux (capturer les grandeurs physiques).
- Les RSCF peuvent être connectés à de nombreux dispositifs et peuvent ajouter de nombreux types de capteurs pour mesurer différents paramètres.
- Les RSCF peuvent être déployés n'importe où, même dans des endroits inaccessibles.

Nous montrons dans ce qui suit les différentes études qui s'intéressent à l'intégration de ces magnifiques dispositifs dans le processus de détection et de prévention.

III.4. Les RCSF et la détection des incendies de forêt

Nous détaillons dans cette partie les améliorations apportées aux systèmes de détection des feux de forêt qui utilisent les réseaux de capteurs sans fil soit comme moyen de détection, soit comme moyen de prévention, où les deux au même temps. Les recherches dans cette thématique s'articulent autour de trois axes essentiels, on trouve : des contributions théoriques, des contributions basées sur les résultats expérimentaux et des contributions qui emploient la simulation pour valider les résultats de recherche obtenus.

III.4.1. Contributions théoriques

Ces contributions cherchent des améliorations liées à la méthode de détection et aux algorithmes utilisés pour cet effet. Dans les travaux présentés dans [71-73], les auteurs ont utilisé la logique floue pour la détection des feux de forêt en mesurant les grandeurs physiques des conditions météorologiques comme la température, l'humidité, les précipitations ... etc. La Prise de Décision Intelligente (PDI) dans ces études apparaît comme caractéristique principale et l'objectif de son utilisation est double. Premièrement, il faut économiser de l'énergie, car le réseau fonctionne sous batteries, deuxièmement, il faut activer les actions nécessaires requises (envoyer des pompiers avec peu de moyens, s'engager à une grande alerte, ... etc.) en fonction des niveaux de sensibilité prédéfinis. L'étude présentée dans [72] se concentre plus sur le deuxième objectif en utilisant les niveaux de sensibilité prédéfinis par le développement d'un algorithme de la logique floue. Dans [74], les auteurs ont comparés plusieurs techniques d'apprentissage par ordinateur pour la prévision des feux de forêt telles que le SVM (*Support Vector Machine*), la régression, les arbres de décision, les réseaux de neurones, ... etc. Cette approche présente l'intérêt de la régression par la grande précision et rapidité de détection par rapport à d'autres techniques

d'apprentissage par machine. L'utilisation de l'Agent Mobile (MA) dans les RCSF a contribué pour augmenter la durée de vie du module d'énergie [75], le système utilisé qui intègre l'Agent Mobile aide aussi à la détection plus rapide des incendies de forêt et à la surveillance en assurant une consommation d'énergie minimale. Dans la littérature, plusieurs algorithmes de vision pour la détection de la fumée des incendies de forêt ont été développés, les auteurs dans [76] ont essayés par le biais de leur système de minimiser les fausses alertes par réduction de champs de vision, au lieu de lancer un traitement sur une image à grand porté, c.-à-d. l'image capturée illustre un grand espace (où la fumée est difficile à détecter), ils préfèrent donc, de capturer plusieurs images à échelle réduite (petites images bien ciblées) jusqu'à la couverture totale de la région d'intérêt. Après plusieurs tests, les auteurs ont trouvés que la détection de début d'allumage d'une surface forestier de 95 m×20 m capturée par leur système confirme un degré élevé de fiabilité en termes de détection précoce réussie et un taux de fausses alarmes très réduit. L'étude présentée dans [77] propose deux algorithmes pour la détection des incendies de forêt. Les algorithmes proposés sont basés sur des techniques de fusion d'informations. Le premier utilise une méthode de seuil, où les nœuds sont équipés avec des capteurs de température, d'humidité et de lumière. Le deuxième algorithme utilise la théorie *Dempster-Shafer* et suppose que les nœuds utilisent des capteurs de température et d'humidité. Les résultats de l'évaluation montrent que les deux méthodes sont capables de détecter efficacement les incendies dans leur phase initiale.

Les théories appliquées aux méthodes de détection des feux de forêt utilisent pour la prise de décision des bases de données stockées auparavant qui datent généralement de quelques mois à quelques années, ce qui rend cette décision en quelque sorte non contrôlée, c.-à-d. la décision ne reflète pas l'état de la forêt pendant le traitement des mesures, donc la décision qui sera prise (en retard) n'a pas d'effet sur terrain au moment de l'incendie. Par conséquent, il devient intéressant de joindre la théorie à la pratique, en d'autres termes programmer dans de vrais capteurs ces théories, et comparer l'état instantané (actuel) de la forêt avec les mesures capturées. Cela confirme la convergence de la décision prise par la théorie. En cas de divergence, il devient nécessaire de passer à d'éventuels ajustements des décisions prises de ces théories.

III.4.2. Contributions expérimentales

Ce type de contribution présente des résultats relevés par des capteurs en utilisant des bancs d'essai (*testbed*), ces recherches traduisent l'état physique de la région d'intérêt en valeur explicative de l'état. Ainsi, la plupart de ces travaux implémentent des algorithmes et des théories pour détecter le feu, parmi ces travaux on trouve le travail présenté dans [78], qui présente la conception d'un nœud capteur à faible débit et à faible puissance pour la détection précoce et le

contrôle du feu. Il implémente l'algorithme de comparaison et l'algorithme DST (*Dempster-Shaffer Theory*) pour la détection des incendies. L'information détectée est partagée en temps réel aux services concernés en utilisant le logiciel *teamviewer* via Internet. Les résultats ont montré que l'algorithme DST a plus de complexité de point de vue conception de logiciel que d'algorithme de comparaison, mais il donne de meilleures performances que l'algorithme de comparaison. En se basant sur les points faibles de MODIS (*Moderate Resolution Imaging Spectroradiometer*) et les réseaux de capteurs basiques, les auteurs dans [79] ont développé un nouveau système de détection des feux de forêt avancée pour conserver le fonctionnement de système le plus long possible grâce au module d'énergie qui fonctionne en mode *Maximum Power Point Tracking* (MPPT), aussi, par le choix de module radio (*xbee s2*). D'autre part, la méthode conserve les capteurs de l'incendie en étudiant son emplacement. Enfin ils ont développé une interface sur le serveur web pour la surveillance de l'état de réseau. La contribution [80] propose un nouveau système de détection des feux de forêt FFDI (*Forest Fire Danger Index*) en utilisant les drones équipés d'une caméra, l'idée est de calculer un indice de couleur sur la végétation illustrée sur l'image capturée, en se basant sur des méthodes de classification de la végétation. Cet indice est adapté pour détecter les nuances de flammes et de la fumée par le calcul de la rougeâtre (quantité de rouge) dedans. L'étape suivante permet de normaliser les couleurs et de lancer un traitement sur les couleurs RGB (*Red Green Blue* normalises). Enfin, un traitement numérique sur les valeurs obtenues donnera l'indice final. Les auteurs ont calculé plusieurs paramètres comme la précision de détection, *True Positives*, *TrueNegatives*, (False positif et False négatifs) qui entrent dans le calcul des indices : indice de *Jaccard*, indice de *Manhattan* et indice de *Dice*, afin d'étudier le rapport taille/précision de détection et taille/temps de calcul de l'indice. Cette étude montre que les drones qui capturent des images et les envoient au Sink pour le traitement sont des moyens de détection en temps réel et fiable pour les feux de forêt [79, 81]. Les auteurs dans [82] proposent une implémentation basée sur la logique floue pour gérer l'incertitude dans le problème de détection de feux de forêt et améliore sa précision. Les nœuds capteurs sont utilisés pour détecter en temps réel la probabilité d'incendie avec différentes variations d'une journée des conditions météorologiques et les envoient avec la variation temporelle à la station de base. Les auteurs dans [83] ont présenté une approche de déploiement de capteurs statiques et mobiles dans les forêts par une répartition sur des endroits aléatoires jusqu'à la couverture maximale possible. Ils ont proposé un algorithme (le plus court chemin) à économie d'énergie. Ils révèlent que les résultats produisent un taux de livraison de paquets stable et efficace en énergie par rapport aux modèles qui utilisent le clustering. Les auteurs dans [84] ont divisés le réseau distribué au hasard en trois sous réseaux,

chacun fonctionne 10 minutes toutes les 30 minutes, cela augmente la durée de vie du réseau de 2,7 % et la performance énergétique de 63 % par rapport aux réseaux de détection d'incendie normaux. D'une manière semblable à notre implémentation de la méthode canadienne présente dans [85] et en introduisant un nouvel indice (*Chandler Burning Index : gravité du danger d'incendie*) et un module d'énergie solaire, les auteurs dans [86] ont adapté la méthode à leur région (*Nanjing* en chine) et ils ont étudié différents scénarios pendant deux ans. Le travail présenté dans [87], implémente l'algorithme *FireSense* dans un véritable banc d'essai pour une analyse plus approfondie de son efficacité en fonction de divers paramètres tels que défaillance de liens, topologie, changement de configuration physique, direction du vent, position du point d'inflammation et variations de la période d'échantillonnage. Dans [77, 88] les auteurs ont capturé des mesures sur les conditions climatiques, la position et les différents types de gaz émis par le feu, et ils les transmettent en utilisant des communications sans-fil vers l'application graphique au poste de traitement pour visualiser et surveiller d'éventuel danger. Une proposition de modèle de gestion des catastrophes pour détecter le feu de forêt est étudiée dans [89] et ils ont mesuré de nombreux autres paramètres dans la forêt qui sont très utiles pour l'alerte d'un incendie.

Les études présentées dans cette partie montrent les avantages que nous trouverons utiles pour implémenter notre nouveau système dont la présentation est détaillée plus bas dans ce document.

III.4.3. Contributions basées sur la simulation

La simulation est une étape préliminaire pour tester une expérience suivant les contraintes d'entrées et les paramètres de simulation. En employant des simulateurs dédiés aux RCSF, les auteurs dans [90] ont présenté un système qui comprend des propositions pour l'architecture du réseau, le schéma de déploiement des capteurs et les protocoles de clustering et de communication, afin de détecter un d'incendie aussi tôt que possible. Le simulateur développé en langage C# a montré après plusieurs expériences que le système proposé fournit une réaction rapide et efficace aux incendies de forêt avec une consommation efficace de l'énergie. Les auteurs dans [91] ont présenté aussi leur système de détection et de localisation des incendies de forêt et l'ont mis en œuvre après sa conception en incluant un protocole «basé sur la portée radio» pour un déploiement à grande échelle. L'efficacité de la méthode a été démontrée par des simulations (sous J-Sim), et ils ont montré que le système fournit une réponse rapide avec une consommation d'énergie efficace. Une nouvelle approche pour la détection des incendies forestiers basée sur l'intégration des techniques de *Data Mining* dans les nœuds capteurs est présentée dans [92]. L'idée consiste à utiliser un RCSF en cluster où chaque nœud décide de détecter le feu individuellement en utilisant un classifieur technique de *Data Mining*. Lorsqu'un

incendie est détecté, le nœud correspondant transmet une alerte par le biais de son cluster-chef qui traversera les nœuds relais jusqu'à ce qu'il atteigne le Sink, qui à son tour informe les services consternés (les pompiers). Le simulateur *CupCarbon* [92] est utilisé dans cette étude pour valider et évaluer l'approche proposée et montrer qu'elle peut apporter une réaction rapide aux incendies de forêt tout en consommant de l'énergie efficacement. Les travaux présentés dans [93, 94] présentent des utilisations des réseaux de neurones pour la détection des feux de forêt. Dans [94], un système intelligent pour la prévision du feu basé sur les réseaux de capteurs sans fil est présenté, ce système obtient la probabilité d'incendie et de comportement du feu dans une zone particulière. Cette information permet aux pompiers d'obtenir des chemins d'évacuation et de déterminer des stratégies pour combattre le feu. Un pompier peut accéder à cette information avec un périphérique portable sur chaque nœud du réseau.

Malheureusement, la plupart de ces études choisissent de simuler leurs solutions proposées au lieu de faire des expériences dans de vrais bancs d'essai, car ce type d'installation pose des difficultés supplémentaires.

Le tableau 3.2 présente une comparaison de différentes solutions de lutte contre l'incendie en fonction de la description de la méthode, des objectifs attendus, du type de données détectées et de l'approche de validation. Nous constatons que la plupart des études basées sur l'expérimentation n'ont pas validé leurs résultats, et qu'elles ne considèrent qu'un seul objectif qui est la détection ou la prévention.

III.4.4. Synthèse et choix de la méthode adéquate

Les méthodes théoriques et celles basées sur la simulation présentent des lacunes par rapport à celles qui se sont basées sur des expérimentations réelles en utilisant des nœuds capteurs. Les études théoriques essayent de trouver des solutions aux problèmes liées à une problématique, ou bien améliorer une solution qui a présenté déjà des résultats, mais ne sont pas souhaitées. À chaque changement dans un des paramètres d'entrée, on se trouve face à des améliorations de la méthode afin de l'adapter aux nouveaux défis. Comme extension aux méthodes théoriques, les méthodes basées sur la simulation fournissent plus d'avantages de point de vue étude d'efficacité, de robustesse et passage à l'échelle. Mais, cela reste lié aux paramètres de simulation octroyés au simulateur.

Le simulateur déroule l'exécution des tâches planifiées pour chaque nœud capteur selon les paramètres (topologie, temps d'exécution, nombre de nœuds...) qui lui sont affectés et fournit des résultats, l'utilisateur par conséquent est sensé d'analyser les résultats obtenus par la simulation en les projetant sur la théorie et/ou paramètres appliquées au simulateur.

La projection des deux méthodes précédentes sur le terrain présente dans la plupart du temps des incohérences liées aux insuffisances relatives à chaque méthode. Par exemple, dans les méthodes théoriques, le problème majeur est celui de l'incertitude, rien ne confirme la validité de la méthode autre que son application réelle. Et pour les méthodes basées sur la simulation, les résultats obtenus négligent souvent des facteurs extérieurs comme (les obstacles, l'atténuation du signal, les changements climatiques, bruits...) qui affecte par conséquent les résultats obtenus.

Afin de présenter de meilleurs résultats et de confirmer la validité de la méthode et de mesurer exactement les différents paramètres d'évaluations, nous sommes orientés vers l'étude expérimentale en utilisant des capteurs réels sur le terrain pour nous concentrer sur les résultats obtenus sous l'effet des différents facteurs qui entraînent le bon fonctionnement du réseau distribué, et de juger le fonctionnement de la méthode selon l'existant perçu.

Réf	année	Brève description de l'étude	Objectifs		Type de donnée capturée		Validation de l'approche		
			Détection	Prévention	Scalaire	Multimédia	Simulation	Expérimentation	Théorique
[71]	2018	Utilisent la logique floue pour détecter des incendies de forêt en mesurant les grandeurs physiques météorologiques	✓	✓	✓			✓	✓
[72]	2013		✓		✓		✓		✓
[73]	2017		✓		✓				✓
[74]	2015	Présente un meilleur fonctionnement de la régression pour la détection des incendies de forêt avec une grande précision par rapport à d'autres techniques d'apprentissage automatique.	✓		✓				✓
[75]	2014	L'intégration de l'AM contribue à la détection rapide des feux de forêt et à la surveillance en garantissant une consommation d'énergie minimale.	✓		✓				✓
[76]	2012	Minimiser les fausses alertes en réduisant le champ de vision, en traitant plusieurs images capturées à petite échelle	✓	✓		✓		✓	
[78]	2016	Implémente les algorithmes DST et comparaisons pour la détection d'incendie et présente la conception d'un nœud de capteur à faible débit et à faible puissance.	✓		✓			✓	✓
[83]	2016	Propose un algorithme à économie d'énergie et une approche de déploiement et révèle que les résultats produisent un débit de transmission de paquets stable et économe en énergie par rapport aux modèles qui utilisent le clustering.	✓				✓	✓	

[80]	2016	Utilisent des drones équipés d'une caméra pour calculer un indice de couleur basé sur la classification de la végétation pour détecter les nuances de flammes et de fumée	✓			✓		✓	
[87]	2009	Dirige l'algorithme <i>FireSense</i> vers une véritable utilisation sur banc d'essai pour une analyse approfondie de son efficacité en fonction de divers paramètres.	✓		✓		✓	✓	
[89]	2016	Ils capturent les informations météo et communiquent l'alarme au Sink si un feu est détecté.	✓		✓			✓	
[88]	2017		✓		✓			par comparaison avec un autre banc d'essai	
[92]	2016	Présente une approche de détection de feux basée sur l'intégration des techniques de Data mining sur des nœuds capteurs.	✓		✓		✓ utilise leur simulateur CupCarbon	✓	✓
[91]	2013	Présente un modèle d'implémentation de détection et de localisation des feux de forêt et de l'environnement de contrôle.	✓				✓ démontrer l'efficacité du modèle par J-Sim		✓
[93]	2017	Utilisent les réseaux de neurones pour la détection des feux de forêt en se basant sur les réseaux de capteurs sans fil	✓		✓		✓	✓	✓
[94]	2011		✓	✓	✓		✓ (par le simulateur c++ développé)	✓	
notre étude	2019	Une approche multimodale proposée pour détecter les feux de forêt, validée par des capteurs multimédias pour résoudre le problème des fausses alarmes, et tire des résultats expérimentaux (consommation d'énergie, latence et choix de technologie) intéressantes.	✓	✓	✓	✓	✓ (Peut être ajouté en utilisant l'émulateur Tossim comme perspective)	✓ (validé par des capteurs scalaires et multimédia)	

Tableau 3.2 : Classification des solutions dédiées aux incendies de forêt

III.5. Le système de détection canadien

La méthode canadienne [95, 96] pour la détection des feux de forêt est inventée afin de traduire l'état instantané d'une zone forestière à une valeur numérique significative calculée dite IFM. La valeur de l'indice IFM « Indice Forêt Météo », ou en anglais « *FWI Forest Weather Index* » est communiquée au Sink périodiquement.

La méthode consiste essentiellement à résoudre une série d'équations (Van Wagner et Pickett, 1985) [95] qui peuvent être calculées rapidement par ordinateur. Elle s'oppose avec les anciennes méthodes qui utilisaient une série de courbes qui étaient converties directement sous forme de tables. IFM peut également être calculé à l'aide de la série publiée de neuf tables établies à partir de ces équations.

L'IFM est calculée en plusieurs étapes par l'introduction des valeurs météorologiques et des valeurs antérieures des anciens indices qui entrent dans le calcul de l'indice final IFM. Seule la valeur de l'indice IFM est envoyée au Sink et supprime la nécessité d'envoyer toutes les mesures calculées et/ou capturées afin de minimiser la consommation énergétique des capteurs. La figure 3.3 montre les étapes de calcul de l'indice IFM.

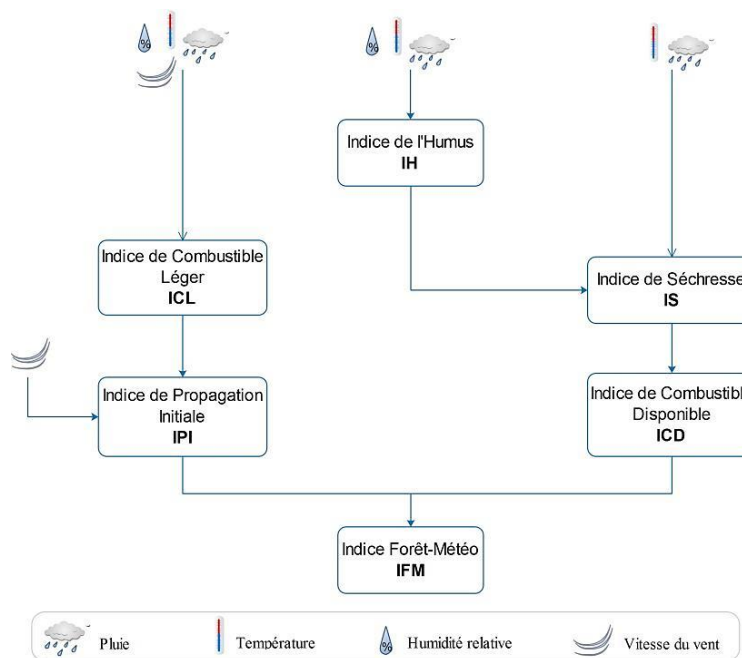


Figure 3.3 : Etapes de calcul de l'indice IFM

Il est à remarquer que les mesures météorologiques sont prises au milieu de l'après-midi lorsque le danger d'incendie est au maximum, l'instant généralement établi est à 16 h. En effet, au cours des premiers travaux, des corrélations ont été établies entre les variables pour les conditions météorologiques à midi et les données sur la teneur en eau du combustible léger et les feux

expérimentaux, données qui ont été obtenues plus tard dans l'après-midi. À cet égard, la méthode IFM est semblable à toutes les autres méthodes utilisées au Canada depuis 1938 [85].

La méthode comprend six indices normalisés. Les trois premiers (IH, IS, ICL) indiquent les variations journalières de la teneur en eau de trois types de combustibles forestiers ayant différentes vitesses de dessèchement. Les trois autres (IPI, ICD, IFM) se rapportent au comportement du feu, et représentent respectivement la vitesse de propagation du feu, de la quantité de combustibles brûlés et de l'intensité du feu. La méthode repose uniquement sur la détermination quotidienne, à midi (heure normale locale), des observations consécutives température, humidité relative, vitesse du vent et précipitations au cours des 24 dernières heures. Les six composantes standards fournissent des évaluations numériques du potentiel incendie de végétation.

III.5.1. Indice du combustible léger (ICL)

ICL ou en anglais FFMC (*Fine Fuel Moisture Code*) est une évaluation numérique de la teneur en eau de la litière – de profondeur 1 à 2 centimètres – et d'autres combustibles légers. Il donne une indication sur l'inflammabilité du combustible léger, ou la probabilité d'inflammation.

III.5.2. Indice d'humidité de l'humus (IH)

IH ou en anglais DMC (*Duff Moisture Code*), est une évaluation numérique de la teneur moyenne en eau des couches organiques peu tassées de moyenne épaisseur – pesant environ 5 kg (masse anhydre) par mètre carré –. IH détermine la probabilité de l'inflammabilité Due à la foudre et montre également le taux de consommation de combustible dans des couches de profondeur modérées qui varient entre 5 et 10 cm.

III.5.3. Indice de sécheresse (IS)

IS ou en anglais DC (*Drought Code*), est une évaluation numérique de la teneur moyenne en eau des épaisses couches organiques compactes pesant peut-être 25 kg (masse anhydre) par mètre carré. C'est un indicateur utile des effets saisonniers des sécheresses sur les combustibles forestiers et du degré de latence du feu dans les épaisses couches organiques et les grosses billes. Il indique aussi la consommation de combustible dans les couches profondes (10-20 cm).

III.5.4. Indice de propagation initiale (IPI)

IPI ou en anglais ISI (*Initial Spread Index*), est une évaluation numérique du taux prévu de propagation du feu. Il combine les effets du vent et de l'indice du combustible léger sur le taux de propagation sans l'influence de quantités variables de combustible. Autrement dit, IPI indique le taux de propagation du feu immédiatement après l'allumage.

III.5.5. Indice du combustible disponible (ICD)

ICD ou en anglais BUI (*Build Up Index*) est une évaluation numérique de la quantité totale de combustible disponible au feu en progression, il combine l'indice d'humidité de l'humus et l'indice de sécheresse.

III.5.6. Indice forêt météo (IFM)

IFM ou en anglais FWI (*FireWeather Index*), est une évaluation numérique d'intensité du feu qui combine l'indice de propagation initiale et l'indice du combustible disponible. Il constitue un indice général du danger d'incendie dans l'ensemble des régions forestières.

Le calcul des différents indices se fait à l'aide de fonctions mathématiques qui seront décrites plus loin. De plus, chaque indice a une échelle de valeurs qui lui est propre. Les indices qui nous intéressent plus et qui présentent une signification remarquable au simple utilisateur de la méthode canadienne sont l'ICL et l'IFM. Les deux tableaux (Tableau 3.3 et Tableau 3.4) présentent la signification de chaque valeur produite lors de processus de calcul des indices ICL et IFM.

Comme mentionné dans la définition, ICL peut être utilisé pour indiquer la facilité d'allumage, ou la probabilité d'inflammation.

Valeurs d'intervalle de l'ICL	Allumage potentiel
0–76	Bas
77–84	Moyen
85–88	Élevé
89–91	Très élevé
92+	Extrême

Tableau 3.3 : Inflammation potentielle par rapport à la valeur de l'ICL[97]

Aussi, l'indice IFM est utile pour déterminer les besoins de lutte contre l'incendie en plus d'être utilisé pour l'information sur les conditions de risque d'incendie.

IFM	Valeur	Description	Type de danger d'allumage potentiel
Bas	0-5	Feu commence de glisser	Une personne peut éteindre le feu
Moyen	5-10	Violence basse du feu	Simple équipement stoppe le feu
Élevé	10-20	Violence moyenne du feu	La présence des pompiers est obligatoire
Très élevé	20-30	Surface du feu très intense	Difficile à contrôler le feu
Extrême	30+	Développement actif du feu	L'incendie devient critique et intense

Tableau 3.4 : Danger potentiel d'incendie d'après l'indice IFM[97]

Une forte valeur de l'indice IFM indique qu'en cas de déclenchement d'un incendie, le feu serait difficile à contrôler. Cette intuition est soutenue par plusieurs études. Voici à titre d'exemple trois indices IFM avec leurs significations sur le terrain [98]:

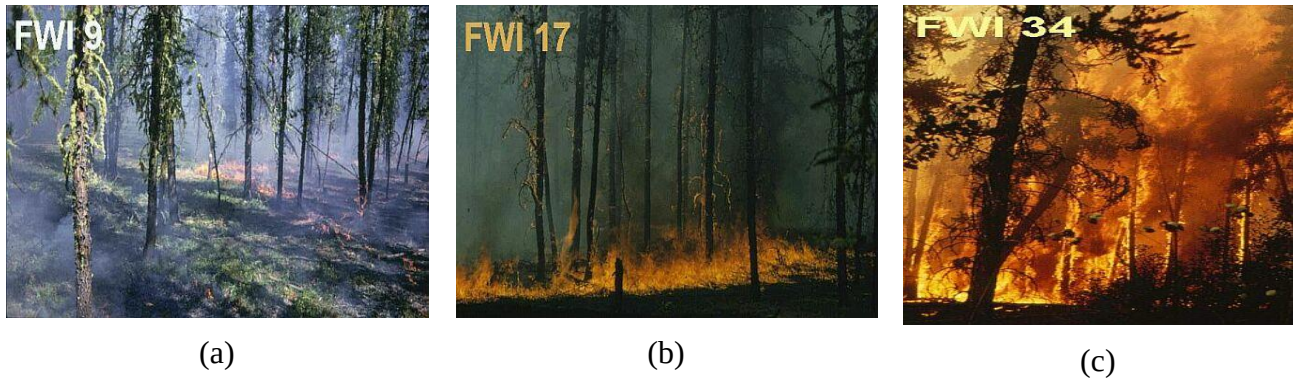


Figure 3.4 : Signification de l'indice IFM sur terrain : (a) IFM=9, (b) IFM=17 et (c) IFM=24

III.6. Le système de détection canadien adapté au contexte local

Le système canadien de détection des incendies de forêt est importé par plusieurs pays qui combattent contre les incendies de forêt, parmi celles-ci on trouve : France, Australie, Nouvelle-Calédonie, Croatie...

Suite à une visite pour demande d'informations sur les mécanismes utilisés localement pour la détection des feux de forêt, les experts des services météorologiques algériens nous ont déclaré que la méthode canadienne est peut être appliquée dans la région méditerranéenne. Ils justifient cette flexibilité du fait que les conditions d'usage de ce système sont proches des nôtres, en prenant en considération bien sûr les modifications relatives aux paramètres d'adaptation pour le nouvel endroit d'application de la méthode, comme la position géographique (latitude) et les changements qui se déclenchent dessous.

III.6.1. Algorithme de détection canadien

Le calcul de l'indice IFM revient à mesurer premièrement les paramètres météo température, humidité, vitesse du vent et la quantité de l'eau provenant de la pluie pendant les 24 heures précédentes, ensuite, il faut passer au calcul des indices intermédiaires ICL, IS, IH, IPI, ICD avec une suite d'équations définis dans [95]. Enfin, il faut assembler les résultats de calcul des indices précédent par une formule magique déclarée aussi dans le document précédemment cité.

La figure 3.5 montre l'algorithme de calcul de l'indice IFM, évoqué plus haut, sous forme d'organigramme d'une part, et la signification attribuée à chaque valeur de l'IFM trouvé conformément au tableau de classification des types de danger de l'allumage potentiel (Tableau 3.4).

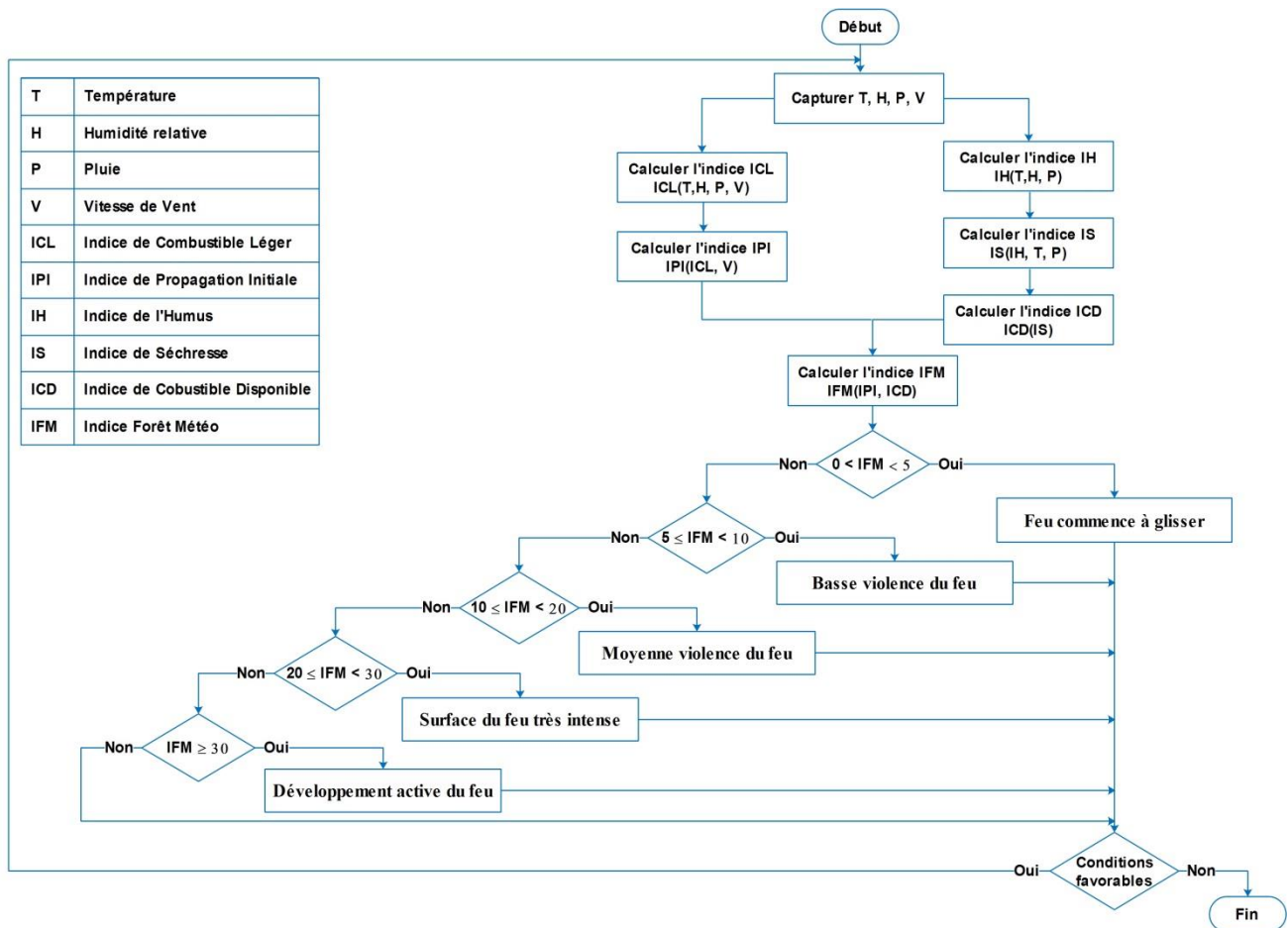


Figure 3.5 : Algorithme de détection de l'approche canadienne

III.6.2. Autres systèmes de détection des feux de forêt

La France, l'Australie, le Portugal, la Nouvelle-Zélande, l'Indonésie, la Malaisie et d'autres pays ayant des climats très différents du Canada ont appliqué la méthode canadienne de détection des feux de forêt afin d'examiner son efficacité. Les résultats obtenus par cette méthode ont prouvé sa large flexibilité après bien sûr une petite adaptation de certains paramètres selon la région d'application. Cela a empêché les chercheurs de trouver d'autres solutions concurrentes et plus fortes. Mais, d'autres pays ont préféré de créer leurs propres méthodes, parmi ceux-ci on trouve la Corée de sud, les États-Unis d'Amérique (USA), l'Australie, l'Espagne...

En Australie, l'indice de risque de feux de forêt de *McArthur* [99] est utilisé pour prévoir l'influence des conditions météorologiques sur le comportement des incendies. Les services météorologiques australiens mesurent régulièrement des prévisions concernant les indices d'herbe de forêt et de danger de feux de forêt (*Grassland* et *Forest Fire Danger Index* – GFDI et FFDI –). L'indice compte sur le facteur de la sécheresse, basé sur les précipitations et l'évaporation, ainsi que sur des variables météorologiques comme vitesse du vent, température et humidité pour calculer l'indice Australien FFDI. Similaire au système canadien, FFDI est calculé par une suite d'équations

développées par les services forestiers en se basant sur des ajustements de la première version inventée par *McArthur*. Le tableau suivant (Tableau 3.5) montre les différents niveaux que peut prendre l'indice de danger d'incendie FFDI et celui d'herbe GFDI.

Catégorie	Indice de danger d'incendie	
	FFDI	GFDI
Catastrophique	+ 100	+ 150
Extrême	75–99	100–149
Sévère	75–99	50–99
Très Haut	75–99	50–99
Haut	12–24	50–99
Bas-Moyen	0–11	50–99

Tableau 3.5 : Indice de danger d'incendie[99]

Les Espagnols [100], ont déployé un système de détection des feux de forêt composés de plusieurs nœuds capteurs, des caméras IP et d'un routeur de Cisco – Linksys WRT54GL – afin d'interfacer le réseau déployé avec les administrateurs. La méthode utilisée consiste à laisser les nœuds capteurs (multi-capteurs) détecter un incendie, ensuite, une alarme est générée par ce dernier et envoyée au serveur, qui à son tour envoie une requête de capture vidéo de la région où s'est déclenchée l'alarme qui sera communiquée (le streaming vidéo) aux pompiers. Le schéma illustré par la figure 3.6 montre les échanges de messages lors de la détection d'une alarme.

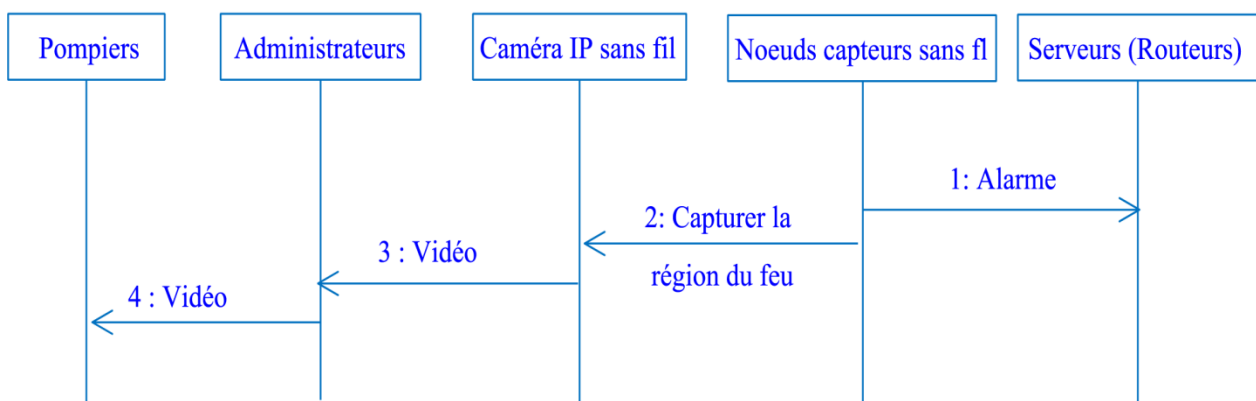


Figure 3.6 : Déroulement des messages lors d'une alerte

En Corée du sud, les auteurs de [101] ont développés un système de détection des feux de forêt en utilisant les RCSF nommées FFSS (*Forest Fire Surveillance System*), ils utilisent des capteurs de type TIP50CM (copie clonée de la firme Telos) afin de capturer la température, l'humidité et la fumée, ensuite ils calculent un indice de danger avec la formule suivante :

$$Y = 6.87 + (0.64 * P) + (0.15 * EF) + \frac{1774.94}{CS} \quad (3.1)$$

$$EF = (1 - r)[H0 + (r * H1) + (r^2 * H2) + (r^3 * H3) + (r^4 * H4)] \quad (3.2)$$

Où : P = Précipitation (mm), EF = l'humidité effective (%), CS = les radiations solaires de la journée (MJ/m²), r = 0,7, H_i = l'humidité relative au jour i.

La signification de la variation de l'indice Y calculée est détaillé dans le tableau 3.6.

Y	Indice de danger	Intervalle de danger d'incendie	État et couleur
10	100	81-100	Extrême (rouge)
11	90		
12	80		
13	70	61-80	Élevée (Jaune)
14	60		
15	50	Moins de 60	Bas (Blue)

Tableau 3.6 : Indice de danger coréen pour la détection des feux de forêt[101]

FireFireWxNet est un système de monitoring constitué dans son premier prototype par trois nœuds capteurs et de deux webcams, déployé la première fois en 2005 dans *Selway-Salmon* à l'état IDAHO (USA) [102]. La récupération des données environnementales par les capteurs dans un large éventail d'emplacement permet aux analystes forestiers de mieux prédire le comportement du feu pour mieux gérer l'ampleur de l'incendie et de renforcer aussi les mesures de sécurité nécessaires. Ce prototype est amélioré en intégrant des radios directionnelles pour étendre les capacités de déploiement dans des espaces au-delà des infrastructures prévues dans le premier prototype. *FireWxNet* est composée de deux sous réseaux, le premier sous réseau est composé de deux parties, la première est celle qui capte les changements environnementaux par les nœuds capteurs de type Mica2 de CROSSBOW, et les communique au Sink (Soekris net4801) via le port USB qui est responsable de l'interfaçage de ces nœuds avec le deuxième sous réseau via un switch Ethernet. La deuxième partie de premier sous réseau est les deux caméras (Sony SNC-RZ30N et Panasonic KX-HCM280) connectées aussi au Switch Ethernet. Le deuxième sous réseau (*backhaul*) permet de connecter les deux parties du premier sous réseau avec les administrateurs via plusieurs radios (*Trango Broadband Wireless : The Trango Access5830 et Trango M900S Access Point / Subscriber Module Radios*) afin d'atteindre l'antenne parabolique finale qui offre une connectivité Internet pour la liaison de données. Ce processus est résumé dans le diagramme montré dans la figure 3.7.

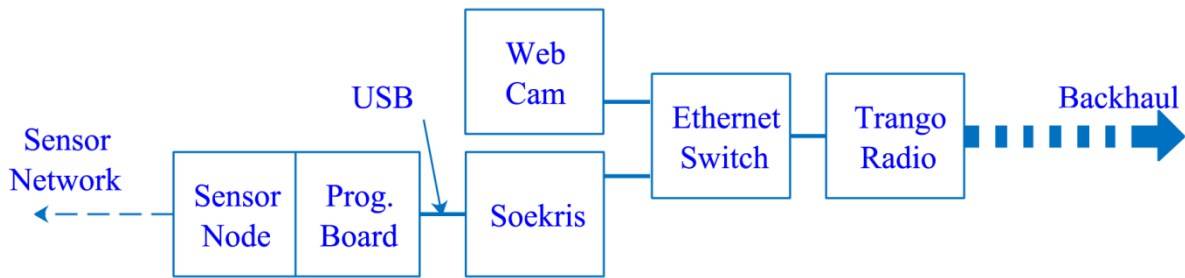


Figure 3.7 : Schéma bloc du système FireWxNet [102]

III.7. Conclusion

Dans ce chapitre, nous avons présenté le périmètre forestier qui définit la région d'intérêt de notre étude en surface et en superficie incendie chaque année, ensuite, un petit historique sur les méthodes de détection des incendies des forêts a été évoqué. Par la suite nous avons détaillé la méthode de détection la plus adéquate, moins chère, efficace et rapide en termes de détection. Nous avons présenté aussi une classification de trois types d'études traitant la problématique abordée comme sujet, et nous avons montré notre sélection de la méthode adoptée parmi d'autres. Une adaptation de la méthode canadienne selon le contexte local est donnée en détaillant son principe de fonctionnement ainsi que l'algorithme qui l'implémente. Enfin, nous avons cité quelques exemples d'autres méthodes existantes mais qui ne sont pas assez connus dans la littérature.

CHAPITRE IV

Contribution1 : Conception, implémentation et évaluation du système M2FS (*Multimedia Forest Fire System*)

IV.1. Introduction

Dans ce chapitre, nous allons présenter notre approche de détection en détaillant la méthodologie adoptée pour concevoir notre système de détection des feux de forêt que nous avons appelé M2FS (*Multimedia Forest Fire System*). Nous détaillons ensuite, les différentes parties qui constituent ce système de point de vue équipements – matériel – et de point de vue logiciel. Nous présentons les résultats de tests effectués accompagnés de discussions en examinant certains paramètres comme le temps de détection, le temps de transmission et l'énergie consommée.

IV.2. Approche de détection des feux de forêt adoptée

Le prototype que nous avons conçu dans sa première version [34, 85] permet de capturer, de détecter et d'acheminer les grandeurs physiques et les alarmes vers la station de contrôle et de prise de décision via le réseau de capteurs sans fil. Notre principal objectif dans la présente thèse consiste à passer d'une architecture monomodale où nous avons utilisé des capteurs scalaires pour collecter les grandeurs physiques liées à notre application des feux de forêt (température, humidité, ...etc), calculer les indices de danger, générer les alarmes et les communiquer via le réseau, vers une architecture multimodale où un nouveau sous-système de capture – multimédia– est mis en place et qui s'ajoute au système monomodale pour améliorer le processus de détection. L'intérêt principal de cette partie multimédia, qui constitue une couche supplémentaire à couche scalaire, est double. Premièrement, elle permet d'introduire une nouvelle source de capture riche en information – la capture multimédia – à l'aide d'une caméra qui permet de fournir plus de données sur la région d'intérêt. Deuxièmement, elle permet de consolider la couche inférieure – scalaire – afin de lutter contre les fausses alarmes générées par le système initial de base qui est incapable de les détecter.

Les étapes principales de l'organigramme de la figure 4.1 montrent le passage du modèle de base (monomodale) vers la nouvelle solution complète proposée (multimodale).

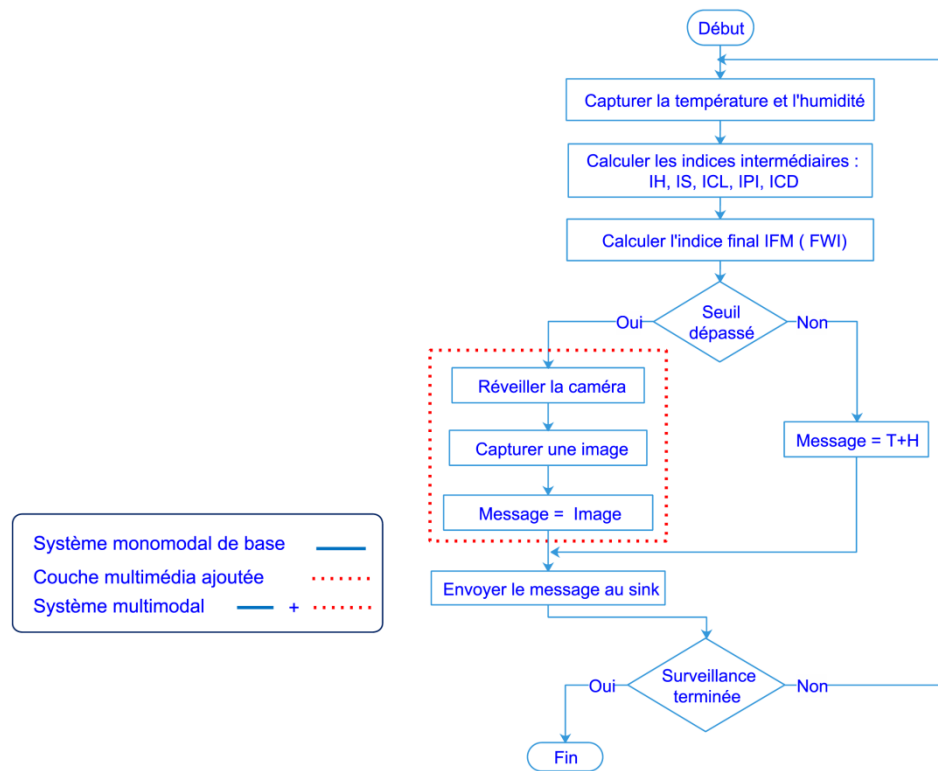


Figure 4.1 : Etapes principales de la solution proposée [103]

IV.3. Conception du système multimodal M2FS

Le système multimodal repose sur trois modules : module de capture, module de transmission et module de stockage (voir figure 4.5) où les deux tiers (scalaire et multimédia) sont élaborés. Ce système montre les différentes entités connectées pour capturer les grandeurs physiques (température et humidité), et la donnée multimédia (l'image) en cas d'un incendie. Ensuite, cette image sera transmise à travers des nœuds relais de différentes technologies (Micaz, Iris, TelosB, Waspnote, Arduino, Raspberry) vers le nœud collecteur. Ce dernier la diffuse par la suite avec le centre de traitement pour la prise de décision et la stocke dans un Cloud pour une interrogation à distance en cas de besoin.

L'intérêt de l'utilisation de différentes technologies comme nœuds relais revient premièrement à la réduction du coût du système conçu, et aussi à l'exploitation des avantages fournis par quelques technologies afin de gagner plus du temps dans la transmission et au traitement local de l'information. Pour le deuxième intérêt, les travaux de recherche de C. Pham [22] ont montré l'utilisation des plateformes de type Micaz comme relais réduit considérablement le temps écoulé dans la procédure d'envoi, la retransmission, la lecture des données dans l'espace mémoire de l'application et dans la réception par rapport aux plateformes TelosB, Arduino et Waspnote. D'autre part, l'utilisation de nouvelles plateformes conduit à un gain remarquable dans la miniaturisation des prototypes de test et d'évaluation, et aussi aux coûts de conception des

systèmes. Par exemple, l'utilisation des nœuds raspberry dans les bancs d'essai diminue considérablement le coût du système, car cette nouvelle carte fournit plusieurs interfaces réseau, ce qui facilite le transfert de données entre les nœuds de différentes technologies (ZigBee, wifi, Ethernet). Enfin, et encore mieux, si nous remplaçons le fameux module radio Xbee par son successeur Lora [104], cela présente un gain double. Le premier est de diminuer le nombre de nœuds relais vu la grande portée offerte par ces modules qui peuvent aller jusqu'à 15 km dans les zones suburbaines [104], cela diminue par conséquent le coût. Le deuxième gain, c'est le temps de communication réduit par l'omission des nœuds relais intermédiaires (les retransmissions, les ACK...).

Pour modéliser notre application, la figure 4.2 illustre le diagramme UML de cas d'utilisation (use case), pour présenter les différentes interactions entre les acteurs et le système.

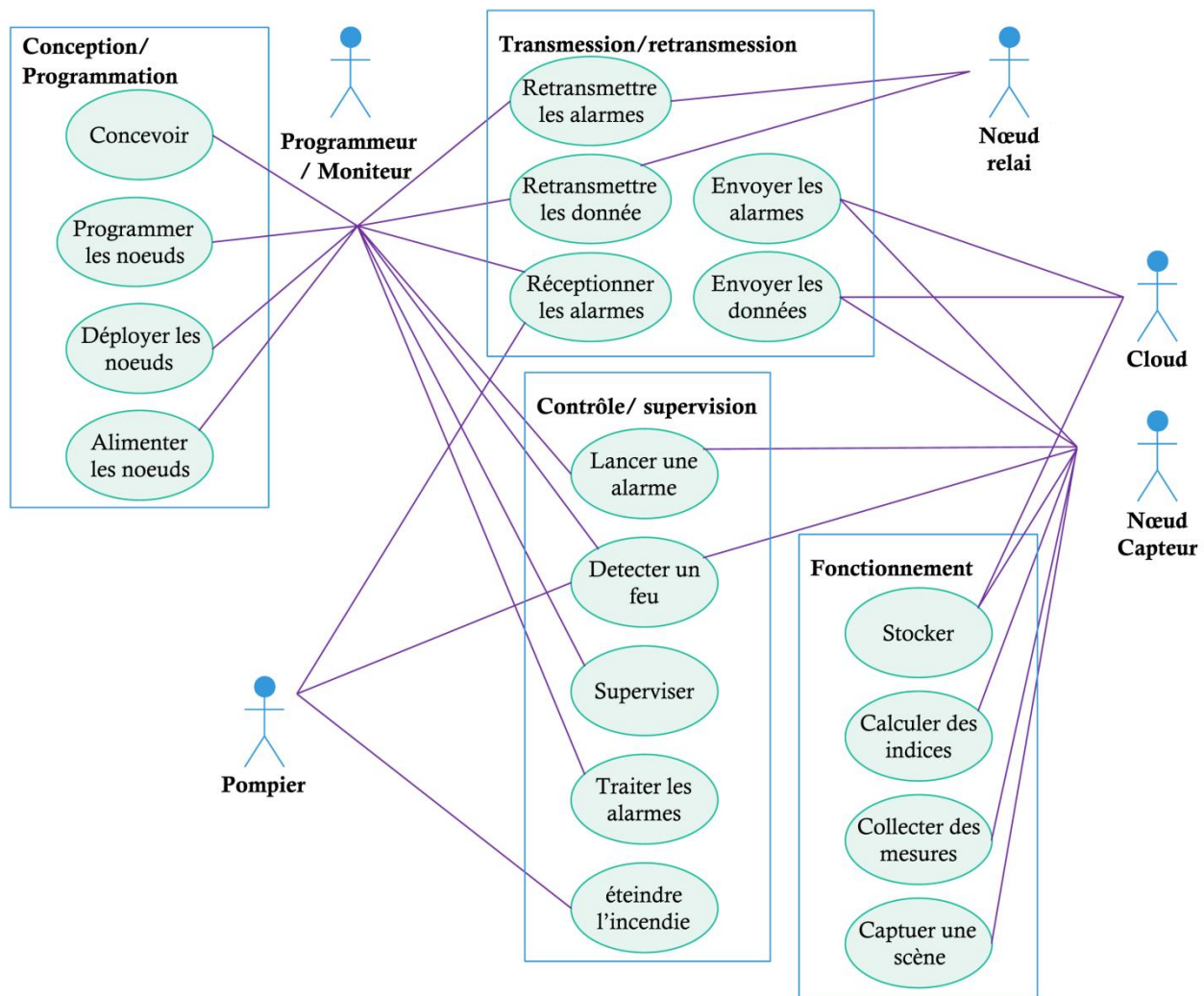


Figure 4.2 : Modélisation par diagramme de cas d'utilisation de l'environnement d'évaluation

Ce diagramme est souvent établi lors de la phase de conception d'un projet afin de transmettre les besoins des utilisateurs et leur fonctionnalité attendus à l'administrateur du système (maître

d'œuvre) pour réajuster et/ou redéfinir les tâches de chaque utilisateur. Dans ce diagramme, les différentes interactions représentées par les acteurs (programmeur/moniteur, pompier, nœud capteur, nœud relai et le nuage de stockage) avec l'environnement d'évaluation et de test (M2FS) montrent les différentes utilisations de notre environnement selon plusieurs aspects : conception/programmation, transmission/retransmission, contrôle/supervision et fonctionnement possible.

La figure 4.3 montre la modélisation UML par un diagramme de séquence dont l'objectif est de d'illustrer les séquences par lesquelles passe toute opération de collecte de données depuis la source jusqu'au centre d'analyse (les pompiers). Ceci permettra de comprendre le déroulement de notre approche, qui commence par la programmation des nœuds (capteurs, relais, Sink), ensuite le lancement de l'application en calculant l'indice de danger à partir des mesures scalaires (température et humidité) jusqu'à la détection d'un événement (le feu). Ce dernier déclenchera une capture de l'image qui devra être transmise à l'administrateur (centre d'analyse) par le biais des nœuds relais pour d'éventuelles décisions.

Ce diagramme englobe plusieurs petites parties, chacune définit une modélisation d'un aspect pertinent de l'environnement d'évaluation et de test comme le déploiement et la programmation des nœuds, le transfert des messages dans le(s) réseau(x) et l'interaction entre les acteurs physiques (personnes) et matériels (nœuds, feu).

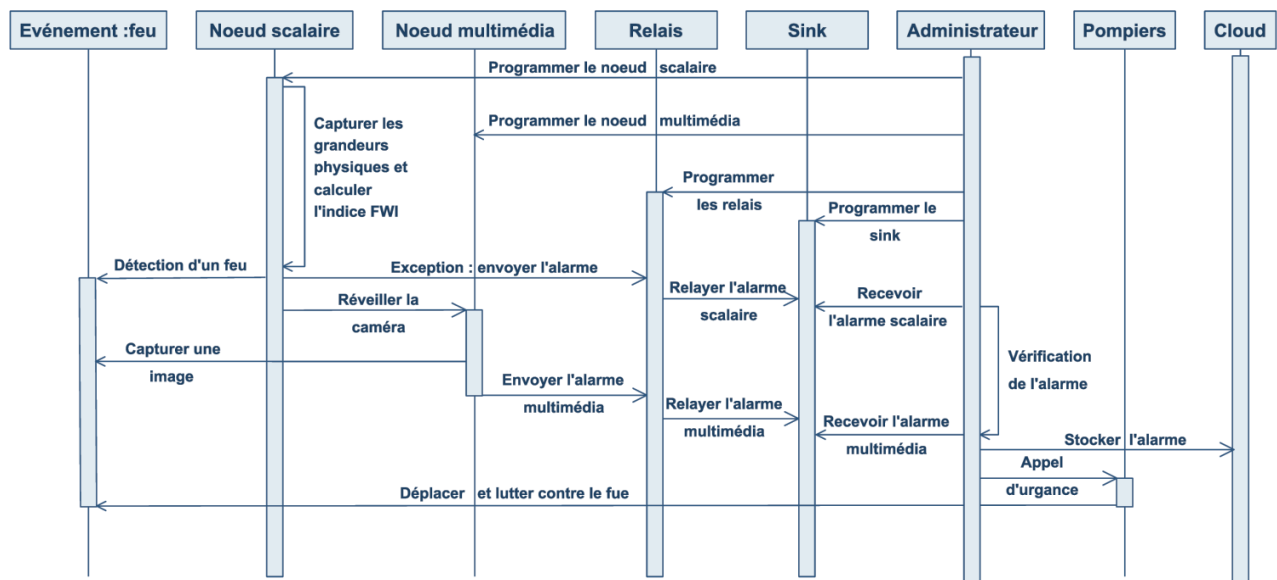


Figure 4.3 : Diagramme de séquence pour l'échange de messages entre les différents acteurs de l'application

IV.4. Implémentation du système multimodal M2FS

L'application de l'extension proposée (la super couche multimédia) sur le système existant peut se faire de deux manières.

- Un scénario qui préserve le système de base initialement conçu à lequel on ajoute un nouveau sous-système multimédia. Dans ce cas, deux types de nœuds capteurs sont déployés, un ensemble de nœuds capteurs scalaires qui se chargent de capturer les grandeurs physiques, calculer les différents indices pour la génération de l'indice final IFM (FWI, voir le chapitre 3), et au moment de déclenchement d'une alarme, la couche –scalaire – envoie un message de demande d'informations supplémentaires à la couche supérieure – multimédia – qui à son tour doit capturer une scène de la région cible où l'alarme est déclenchée, et transmettre l'image capturée via le réseau vers la station de traitement et de prise de décision. Ce scénario est schématisé dans la figure 4.4 en montrant les différents modules utilisés pour son élaboration.

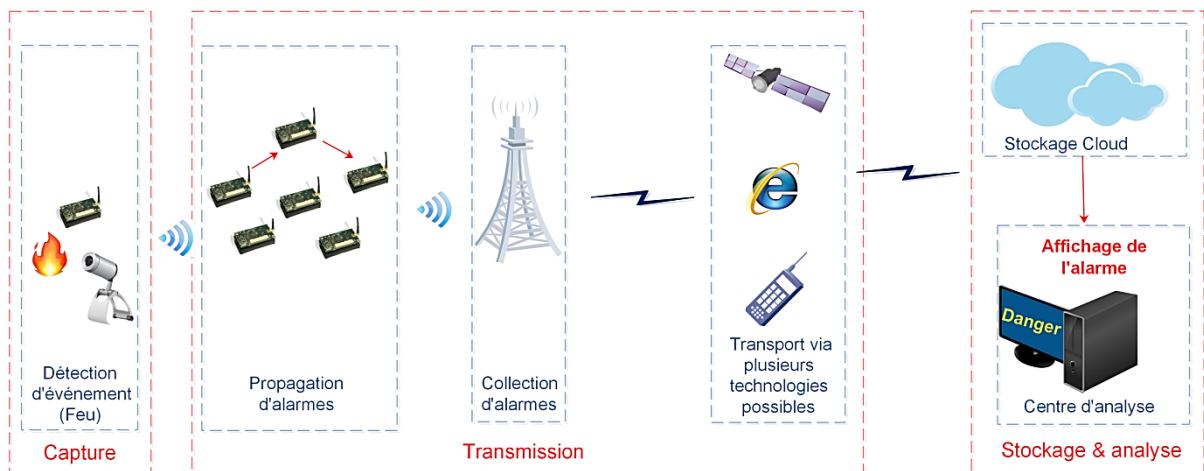


Figure 4.4 : Schéma général de l'application selon le scénario1

- Le deuxième scénario, permet de remplacer le module de capture composé dans le premier scénario (deux capteurs scalaire et multimédia séparés) par un seul module intégré plus simple (voir figure 4.5). Le nouveau module intégré englobe les deux fonctionnalités – capture scalaire et capture multimédia – dans un seul module de capture intégral. Autrement dit, un nouveau nœud capteur est déployé, qui remplace les deux initialement déployés, est capable d'assurer à la fois la capture des grandeurs physiques, le calcul d'indices, la capture multimédia dans le cas où une alarme est déclenchée et enfin la transmission de celle-ci via le réseau vers le centre décisionnel.

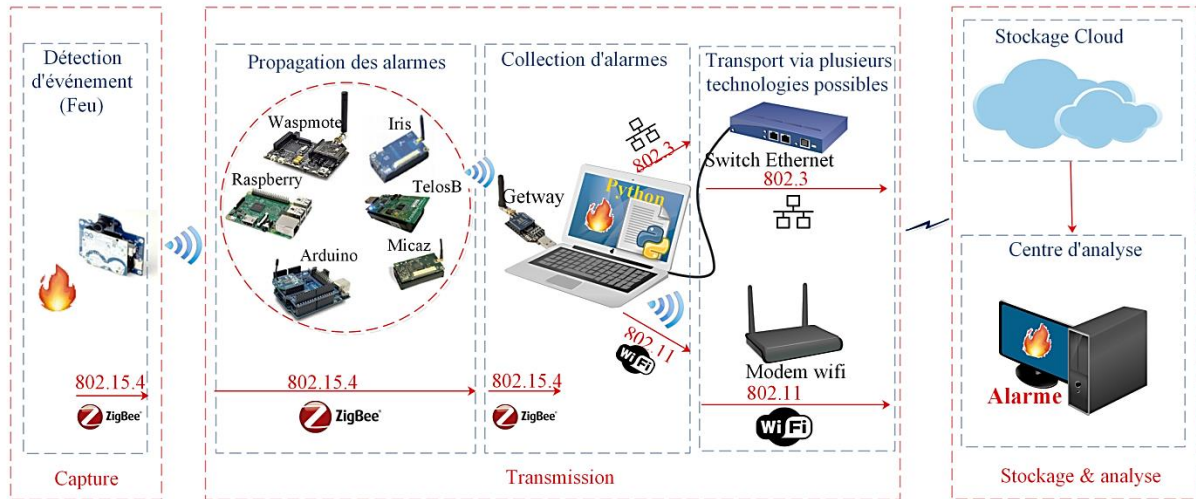


Figure 4.5 : Schéma détaillé de l'application selon le scénario 2

Dans ce qui suit, nous détaillons les trois modules constituant notre application (capture, transmission, stockage et analyse), ensuite nous comparons les deux scénarios proposés auparavant pour étudier lequel est meilleur et adéquat pour les besoins d'implémentation de notre application.

IV.4.1. Module de capture

Le module de capture est l'élément principal dans les deux scénarios. Il est responsable de la collecte périodique des mesures météorologiques et de calcul des indices intermédiaires qui entrent dans le calcul de l'indice final exprimant le danger IFM. Ensuite, il permet de capturer une/des image(s) et de générer l'alarme en cas d'un éventuel incendie. Ce module communique au module transmission des données périodiques sur l'état actualisé de l'espace surveillé dans des situations où le risque d'incendie est faible, et dans le cas contraire où le risque est élevé, il communique des alarmes contenant l'état de la région surveillée c.-à-d. une alarme multimédia qui est composé de la valeur de l'indice IFM et de l'image capturée. Dans le premier scénario présenté dans la figure 4.4, ce module est constitué d'un nœud capteur scalaire utilisé pour la détection d'un incendie, et d'un nœud capteur multimédia séparé pour la capture d'une image. Ce dernier reste inactif tant que le nœud scalaire ne l'envoie pas une requête de vérification suite à une alarme scalaire déclenchée après dépassement de l'indice IFM une valeur seuil qui exprime le danger. Dans le deuxième scénario présenté dans la figure 4.5, un nœud capteur s'occupe de toutes les tâches assurées par les deux nœuds capteurs qui composent le module de capture dans la première architecture, en assurant les mêmes fonctions de capture, de calcul de l'IFM, du réveil de la caméra attachée à ce nœud et de la génération d'alarme. Plusieurs points peuvent différencier ces deux configurations, nous citons dans ce qui suit quelques-uns :

- Le coût : la première configuration nécessite deux nœuds (scalaire et multimédia) pour surveiller une région en utilisant le système M2FS. Cela nécessite deux cartes de capture, deux cartes mère (microcontrôleur, Ram, mémoire de stockage), deux modules radio et deux batteries/jeu de piles. Par contre, la deuxième configuration fait intervenir un seul nœud avec une seule carte de capture, une carte mère, un module radio et une batterie/jeu de piles pour exécuter la même tâche. D'où, le coût nécessaire pour acquérir deux nœuds capteurs est sans doute plus élevé par rapport à l'acquisition d'un seul nœud qui assure les mêmes fonctions.
- Le trafic : dans la deuxième configuration, les communications inter module n'auront pas lieu, ainsi aucune communication ne se produit entre les nœuds capteurs puisqu'un nœud est par nature autonome. Par contre, dans la première configuration, les communications dans le module de captures sont de deux types : les messages de contrôle entre le nœud scalaire et le nœud multimédia, et les requêtes déclenchées par le nœud scalaire suite à l'existence d'un incendie et leur transmission au nœud multimédia.
- L'effort : le programmeur dans le premier scénario dépense plus d'effort pour programmer les nœuds capteurs, et cela à cause de plusieurs problèmes qui peuvent avoir lieu comme les interférences, la congestion, la sécurité et le nombre dupliqué de nœuds à programmer.

IV.4.2. Module de transmission

Ce module est responsable de la liaison de données générées (valeurs périodiques de l'état de la région surveillé ou les alarmes) par le module de capture et le module sauvegarde et analyse. Il assure un envoi de bout en bout en garantissant les paramètres de qualité de service tel que la fiabilité (l'alarme doit arriver au Sink d'une manière sûre), contrainte temporelle (l'alarme doit arriver dans un temps raisonnable) et la sécurité (le cheminement de l'alarme doit être sécurisé contre toute attaque ou acte malveillants). Ce module est composé de trois sous modules interconnectés,

IV.4.2.1. Les relais

Leur rôle consiste à propager les données et les alarmes reçues vers un collecteur de données. Plusieurs types de nœuds relais peuvent être utilisés selon les travaux de recherche de C. Pham [22]. Ces relais doivent impérativement être de la même technologie, c.-à-d. ils doivent communiquer avec le même protocole réseau. ZigBee est le protocole utilisé dans le réseau déployé au sein de nos expérimentations schématisées dans la figure 4.5.

IV.4.2.2. Le collecteur

C'est un terminal placé à proximité des nœuds relais. Son rôle est de récupérer l'alarme reçue, la partie image qui est acheminée en valeurs hexadécimales, et de la convertir en image JPEG à l'aide

d'un scripte python (présenté dans la partie résultat). Ensuite, il réexpédie à nouveau l'alarme vers les routeurs et les passerelles. Ce terminal est doté de plusieurs interfaces réseau, une de type ZigBee pour la réception des données, et les interfaces Wifi et Ethernet configurés comme émettrices. Techniquement, une carte Raspberry peut assurer ces tâches et se comporter comme un terminal.

IV.4.2.3. La passerelle

Son rôle est d'assurer la liaison entre le réseau de capteur sans fil déployé avec le réseau internet afin de permettre une communication entre le site de surveillance et la station de contrôle.

IV.4.3. Module d'analyse et de sauvegarde

Ce module est composé de deux services, un centre d'analyse et un centre de stockage au niveau du Cloud. Ce sont deux services pour traiter les données reçus et les stocker. Ce qui fait la différence entre le centre d'analyse et le traitement dans le Cloud est l'automatisation des tâches. Dans un centre d'analyse, la présence de l'homme est indispensable pour analyser les informations qu'il reçoit et prendre la bonne décision (informer les pompiers et suivre l'activité et la propagation du feu). Dans sa version automatique, le Cloud computing permet d'exploiter la puissance de calcul offerte par des serveurs distants en exploitant ou en intégrant des services dans cet espace. Aussi, il permet de stocker les décisions et les alarmes, enfin, d'informer les services concernés (les pompiers) par des SMS ou des notifications sur Smartphones ou toute autre application technologique qui fait partie des avancées de l'internet des objets en cas d'incendie.

Le système de détection monomodal conçu dans nos premiers travaux [34, 85] a été établi en utilisant un kit de base micaz formé de deux nœuds capteurs et de deux nœuds relais avec une station de base de la firme Crossbow. Dans la présente thèse nous avons adopté la technologie Arduino comme solution d'expérimentation, ce choix est motivé par l'absence de détecteurs multimédia dans le premier kit. Mais, l'implémentation de la solution multimodale exige l'utilisation des ressources multimédias afin de maitre en œuvre cette solution. Le passage à la solution Arduino n'empêche pas de maintenir les nœuds de type Micaz utilisés dans le système monomodal ou toute autre technologie équivalente (TelosB, Iris...) comme nœud relais pour la remonter l'alarme vers le Sink comme ça été expliqué plus haut.

Dans ce qui suit, nous détaillons les différentes entités composant le banc d'essai (banc d'essai) utilisé pour l'implémentation de système M2FS. Nous avons besoin donc d'un nœud capteur scalaire, un nœud capteur multimédia, une station de base (Sink) attachée à un poste de traitement, et un Cloud sur *DropBox*. Dans ce qui suit, nous présentons chaque module à part, nous montrons

ensuite la façon de les faire fonctionner ensemble et quelles sont les outils dont nous avons besoin pour cette réalisation.

IV.4.4. Un nœud capteur scalaire

Ce nœud a pour but de *capturer* les grandeurs physiques température et humidité, *calculer* par la suite l'indice de forêt météo (IFM, *FWI* en anglais) indice de danger, qui donne l'information sur l'inflammabilité de la zone surveillée. Ensuite, *communiquer* les valeurs capturées et les paquets d'alarmes avec le module communication. Ce nœud est constitué d'un capteur Dht11/Dht22, une carte de type Arduino, un module radio de type Xbee relié à la carte Arduino.

IV.4.4.1. Les capteurs Dht11/Dht22

Nous avons utilisé deux types de capteurs Dht11 et Dht22 (AM2302) présentés dans la figure 4.6. Chaque capteur collecte la température et l'humidité à la fois « deux en un ». Ils sont tous les deux compatibles aux tensions d'alimentation 3,3 volts et 5 volts (le fabricant recommande cependant de toujours alimenter le capteur en 5 volts pour avoir des mesures précises). Aussi, ces capteurs ont la même norme de câblage et de protocole de communication. Le tableau 4.1 résume et compare les caractéristiques des deux capteurs.

	DHT22/AM2302	DHT11
Humidité (relatif %)	0 ~ 100 %	20 ~ 80 %
Précision (humidité)	+/- 2 % (+/- 5 % aux extrêmes)	+/- 5 %
Température	-40 ~ +150 °C	0 ~ +50 °C
Précision (température)	+/- 0,5 °C	+/- 2 °C
Fréquence de mesure max	2 Hz (2 mesures par seconde)	1 Hz (1 mesure par seconde)
Tension d'alimentation	3 ~ 5 volts	3 ~ 5 volts
Stabilité à long terme	+/- 0,5 % par an	+/- 1 % par an

Tableau 4.1 : Caractéristiques des capteurs DHT11 et DHT22

D'après le tableau 4.1, le capteur DHT22 est beaucoup plus précis et stable que le DHT11, néanmoins, il est deux fois plus cher. Le choix se résume donc à une question de balance entre les critères coût/précision/rapidité de mesure.

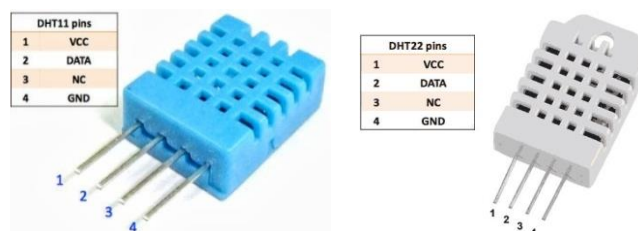


Figure 4.6 : Les capteurs DHT11 et DHT22

Pour commencer à utiliser ces capteurs, nous commençons par le montage, nous nous suivons le câblage illustré dans la figure 4.7 qui montre la connexion des trois pins nécessaires à la carte Arduino. Le rôle de la résistance (4,7 KOhm) intégrée dans le circuit est de calibrer et d'ajuster la tension d'alimentation de la carte Arduino avec celle du capteur.

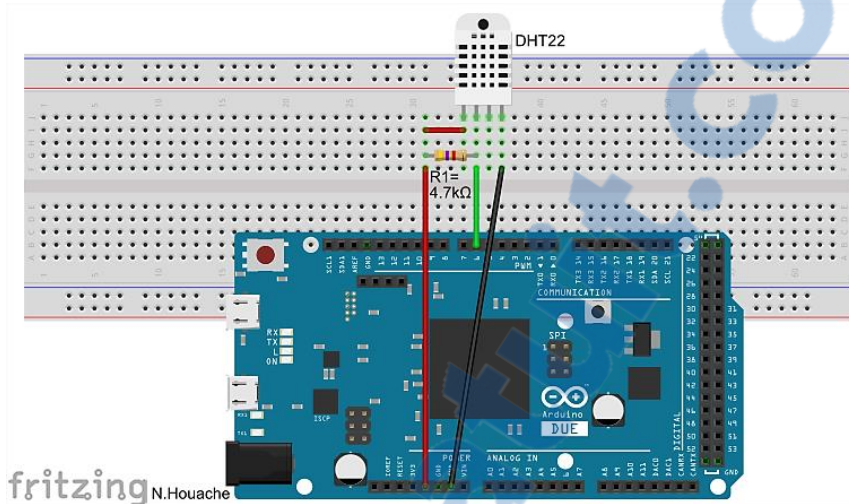


Figure 4.7 : Brochage de capteur Dht22 à la carte Arduino

Après le montage, il ne reste qu'à téléverser le code de l'application qui permet de mesurer la température, l'humidité et calculer l'indice IFM (le calcul de l'IFM est bien détaillé dans[95]) dans le nœud capteur.

IV.4.4.2. La carte Arduino

Arduino, c'est l'histoire d'un groupe d'enseignants et d'étudiants en école d'art qui avait besoin d'une plateforme de développement simple et peu coûteuse pour permettre la réalisation de projets d'arts interactifs. L'idée de base du projet Arduino est simple, mais terriblement efficace : proposer une carte à microcontrôleur (un microcontrôleur n'est rien de plus qu'un tout petit ordinateur), certes simpliste, mais disposant de tout le nécessaire pour commencer un projet. En parallèle de cette carte, l'idée du projet Arduino était de fournir un logiciel de développement informatique multiplateforme (Windows, Mac et Linux) permettant de mettre en œuvre la carte rapidement sans prise de tête. Pour faire simple : ils donnent la base physique et la base logicielle, maintenant il ne reste plus qu'à construire par-dessus. C'est simple et ça marche. Et comme parfois l'histoire réserve de drôle surprise, ce qui était à sa sortie en 2005 un simple outil pour quelques étudiants est aujourd'hui une véritable révolution dans le monde du «*Do It Yourself*», de l'électronique amateur et même de l'informatique. Il reste qu'à dire pourquoi nous avons choisi la plateforme Arduino ?

- Arduino est peu cher,
- Arduino est multiplateforme,
- Arduino propose un environnement de programmation simple et clair,

- Le logiciel de programmation d'Arduino est gratuit et open source,
- Le matériel Arduino est open source.

Aujourd'hui, le marché d'Arduino propose plusieurs variantes de carte, chacune offre des améliorations et des mises à jour du firmware. Dans notre projet, nous avons choisi la carte adéquate à nos besoins, la carte Arduino Due.

La carte Arduino Due illustrée dans la figure 4.8 est la première carte Arduino basée sur une architecture ARM 32bits. Usuellement, les cartes Arduino sont plutôt basées sur une architecture ATmega 328.

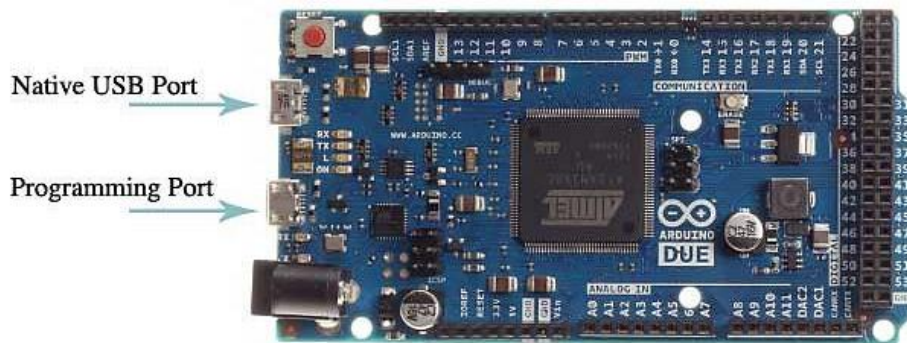


Figure 4.8 : Vue de face de la carte Arduino due

Due accueille un microcontrôleur Atmel SAM3X8E ARM Cortex-M3. Elle possède 54 entrées/sorties numériques (dont 12 peuvent être utilisées pour une sortie PWM), 12 entrées analogiques, 4 ports série UART, une horloge à 84 MHz, une connexion compatible USB OTG, 2 DAC (digital/analogique convertisseur), 2 TWI, une prise jack, un header SPI, un header JTAG et enfin un bouton de reset et un bouton d'effacement. Ces caractéristiques (présentés dans le tableau 3.2) rendent cette carte la plus puissante en termes de rapidité des calculs et espace de stockage, ce qui permet de gérer des algorithmes lourds, et d'apporter plus de réactivité aux équipements commandés par cette carte.

Remarque: Contrairement aux autres cartes Arduino (Uno, Mega, Duemilanove ou Leonardo), les ports d'entrées/sorties de cette carte Arduino Due sont destinés à recevoir uniquement des niveaux logiques 3,3 V ! Toutes tensions supérieures appliquées comme 5 V du Vcc risquent d'endommager la carte !

Caractéristique	Valeur
Microcontrôleur	AT91SAM3X8E
Tension de fonctionnement	3,3 V
Tension d'alimentation (recommandée)	7— 12 V
Tension d'alimentation (limites)	6 — 20 V
Nombre d'entrées/sorties	54 (dont 12 génèrent des signaux PWM)
Nombre de ports « Analogique/Numérique »	12
Nombre de ports « Analogique »	2 (DAC)
Courant max. par E/S	130 mA
Courant pour broches 3,3 V	800 mA
Courant pour broches 5 V	800 mA
Mémoire Flash	512 KB
SRAM	96 KB (2 banques 64 KB et 32 KB)
Vitesse d'horloge	84 MHz
Longueur	101,52 mm
Largeur	53,3 mm
Poids	36 g

Tableau 4.2 : Caractéristiques techniques de la carte Arduino Due

Le microcontrôleur ARM Cortex-M3 livré avec cette carte propose plus de performance que les habituels microcontrôleurs à 8 bits que l'on trouve sur les autres cartes Arduino. Ce microcontrôleur permet :

- Un noyau 32 bits qui autorise des opérations sur 4 octets dans un seul cycle d'horloge.
- Une vitesse d'horloge de la CPU de 84Mhz
- 96 Ko de SRAM
- 512 Ko de mémoire Flash pour le code
- Un contrôleur DMA qui peut soulager la CPU lors des tâches intensives en mémoire.

L'environnement de programmation (IDE) open source pour Arduino peut être téléchargé gratuitement du site du fabricant (www.Arduino.cc) pour les trois plateformes Mac OS X, Windows, et Linux.

La carte Arduino Due possède deux connecteurs USB (voir figure 4.8), le *programming port* et la native USB Port. Les deux peuvent être utilisées pour injecter le programme dans la carte, cependant, il est recommandé d'utiliser le *programming port* qui fonctionne même en cas de plantage de la MCU.

IV.4.4.3. Le module radio Xbee et le Shield Xbee

Les modules Xbee (figure 4.9) sont des modules de communication sans fil très populaires fabriqués par l'entreprise Digi International [105]. Ils ont été certifiés par la communauté

industrielle ZigBee Alliance en 2006. La certification Zigbee se base sur le standard IEEE 802.15.4 qui définit les fonctionnalités et spécifications des réseaux sans fil à usage personnelle (*Wireless Personal Area Networks* : WPANs). Le tableau 4.3 présente les principales caractéristiques de module Xbee :

Caractéristique	Valeur
Fréquence porteuse	2.4ghz
Portées variées	Assez faible pour les Xbee 1 et 2 (10 - 100 m), grande pour le Xbee pro (1000 m)
Faible débit	250 kb/s
Faible consommation	3.3v @ 50ma
Entrées/sorties	6 10-bit ADC input pins, 8 digital IO pins
Sécurité	Communication fiable avec une clé de chiffrement de 128 — bits
Faible coût	~ 25 €
Simplicité d'utilisation	Communication via le port série
Ensemble de commandes	At et Api
Flexibilité du réseau	Sa capacité à faire face à un nœud hors service ou à intégrer de nouveaux nœuds rapidement
Grand nombre de nœuds dans le réseau	65 000
Topologies de réseaux variées	Maillé, point à point, point à multipoint

Tableau 4.3 : Principales caractéristiques du Xbee

Plusieurs produits Xbee existent, ce qui peut créer quelques confusions. Il est à noter qu'il existe deux catégories de Xbee : la série 1 et la série 2. Les modules de la série 1 ont souvent un « 802.15.4 » qui s'ajoute à leurs noms. Les modules de la série 2 sont disponibles en plusieurs versions : XBeeZNet 2.5 (obsolète), le ZB (l'actuel) et le 2B (le plus récent). Il y a aussi des XBee Pro, qui font la même chose, mais avec de plus grandes capacités, notamment la portée qui semble pouvoir aller jusqu'à 1000 mètres !

Pour notre application, nous avons besoin de deux modules Xbee, le premier doit être branché sur le nœud capteur par le biais d'une carte d'extension qui s'appelle Shield (voir Figure 4.10), et le deuxième est attaché au PC de traitement en utilisant une autre carte d'interfaçage qui s'appelle Xbee explorer (voir Figure 4.11).



Figure 4.9 : Module Xbee



Figure 4.10 : Shield Xbee



Figure 4.11 : XBee Explorer

L'Xbee avec son Shield rendent la carte Arduino capable de communiquer en mode sans fil via le module radio avec les autres cartes, donc l'ensemble Shield XBee avec le module Xbee constitue une carte réseau Zigbee. L'avantage d'utiliser le Shield Xbee réside en réalité dans la sécurité de montage électronique de module Xbee avec la carte Arduino, car la communication standard sans utilisation des paramètres de QdS (ex : mode *sleep/wakeup*) diminuent le nombre de pin à connecter l'XBee avec l'Arduino, dans ce cas on utilise que quatre pins pour le brochage (3.3v, Gnd, Tx, Rx) du module Xbee avec la carte Arduino. Pour cela on pourra brancher le module Xbee avec la carte Arduino directement par câble, par conséquent, l'achat du Shield Xbee ne sera plus nécessaire.

IV.4.5. Un nœud capteur multimédia

Ce nœud peut se comporter en deux manières, comme nœud indépendant du nœud scalaire, ou bien comme nœud capteur scalaire et multimédia à la fois. Nous détaillons chaque mode, et nous choisissons le mode le plus approprié à nos besoins.

IV.4.5.1. Nœuds indépendants

Dans ce premier mode (deux nœuds indépendants), pour implémenter notre solution, nous avons besoin d'un Shield Xbee supplémentaire et d'un module Xbee en plus. Pour cela, le scénario doit suivre les étapes suivantes (scénario présenté dans le diagramme de séquence de la figure 4.2) :

- Programmer chaque nœud capteur,
- Capturer cycliquement les grandeurs physiques et calculer l'indice IFM correspondant,
- Communiquer une information sur l'état de la région surveillée avec la station de base si tout passe normalement,
- Communiquer l'alarme avec le nœud multimédia,
- Ce dernier (capteur multimédia) capture une scène de cette région et l'envoie avec l'alarme scalaire vers la station de base,
- Le superviseur du système examine les données reçues et déclenche la procédure adéquate.

IV.4.5.2. Nœud unique « Deux en un »

Le mode « deux en un » considère que les deux nœuds capteurs scalaires et multimédia composent un seul nœud qui déroule un scénario similaire à celui présenté dans le diagramme de séquence illustré par la figure 4.2. Premièrement, la carte Arduino récupère les mesures des captures des grandeurs physiques issues de capteurs Dht11 et/ou Dht22, calcul l'indice IFM et communique ensuite l'état de la région avec la station de base. Une fois l'indice IFM dépasse le seuil de danger, une requête de réveil de la caméra qui est initialement mise en veille pour capturer une scène. Enfin, l'alarme composée de l'image et de la valeur de l'IFM sera transmise vers la station de base pour la prise de décision. La différence entre les deux modes se résume sur les métriques, coût, interférences et durée de vie du réseau, voici les critiques :

- **Coût** : la différence de prix d'équipements entre les deux cas coûte au minimum 70 euros pour ajouter un module Xbee avec son Shield dans la première configuration, cela rend la deuxième configuration plus intéressante.
- **Les interférences** : le regroupement de nœuds permet de supprimer la communication entre les nœuds capteurs, seules les communications avec le module transmission sont maintenues, ce qui préserve un réseau non chargé et fiable. Par contre, dans le cas où les nœuds sont séparés, le réseau va contenir plus de chevauchement de messages ce qui génère des interférences et nécessite à implémenter dans ce cas des algorithmes de gestion de collisions et d'interférences.
- **Durée de vie de réseau** : cette contrainte se base sur la consommation énergétique du réseau. Dans le premier cas de figure, les réserves énergétiques vont s'épuiser plus rapidement que dans le deuxième cas, car, le système consomme plus d'énergie pour les communications entre le nœud scalaire et le nœud multimédia (c'est le module le plus gourmand en énergie), ces communications (entre les nœuds capteurs) n'existent pas dans le deuxième cas. Ainsi, le réseau va tenir plus longtemps pour ce deuxième cas.

Le capteur multimédia utilisé au sien de nos expérimentations qui est illustré dans la figure 4.12 est composé d'une carte Arducam Rev. C dédié d'un écran (Shield + écran) qui permet d'interfacer la caméra utilisée OV2640 de la firme Omnivision avec la carte Arduino.

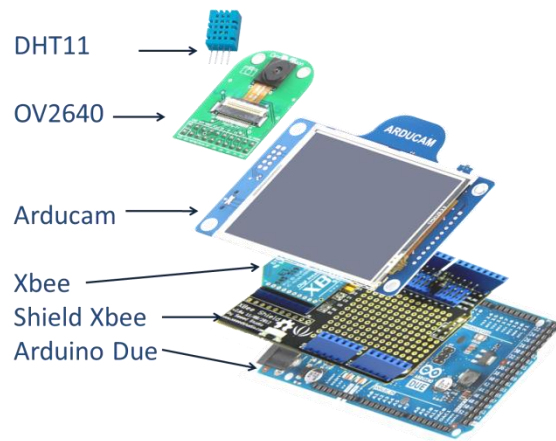


Figure 4.12 : Illustration du nœud multimédia utilisé

IV.4.5.3. La caméra OV2640

Ov2640 est un capteur d'image à basse consommation d'énergie qui intègre la technologie des semi-conducteurs CMOS (*Complementary metal-oxide-semiconductor*) de 2 mégapixels permettant d'obtenir des images de 1600 x 1200 pixels et des vidéos à 15 images/sec. Cette caméra (illustrée dans la figure 4.13) est utilisée pour la fabrication de plusieurs équipements comme :

- Téléphones portables.
- Les PDA (*Personal Digital Assistant*).
- Jouets.
- Robot des plates-formes Arduino, Maple, ChipKit, STM32, ARM, DSP, FPGA...



Figure 4.13 : Vue de face de la caméra Ov2640

Puisque le nombre de pins de la camera ov2640 est important (20 pins) et cela nécessite plus de prudence lors de son brochage avec la carte Arduino, nous avons décidé de la connecter à un Shield qui s'occupe de l'interfaçage de cette caméra avec la carte Arduino. Le Shield Arducam est très approprié pour s'occupe de cet interfaçage.

Le tableau 4.4 représente les principales caractéristiques de cette caméra.

Caractéristique	Valeur
Alimentation	5 Vcc
Consommation	70 mA en utilisation 20 mA au repos
Résolution	2 Méga pixels
Capteur	Omnivision OV2640
Formats supportés en vidéo	1600 x 1200 à 15 fps 1280 x 1024 à 15 fps 800 x 600 à 30 fps 352 x 288 à 60 fps
Formats supportés en photo	RAW, YUV, RGB et JPEG
Interface	I2C pour la configuration du capteur SPI pour les commandes et le flux de données
Dimensions	34 x 25 x 23 mm
Poids	20 g
Prix	~ 33 €

Tableau 4.4 : Caractéristiques de la camera Ov2640

Remarque : une fausse connexion entre la caméra et la carte Arduino peut abimer l'un des deux équipements ! Il faut toujours s'assurer de la liaison correcte avant d'alimenter l'ensemble carte et caméra !

IV.4.5.4. Le Shield Arducam

Arducam illustré figure 4.14 est un Shield (carte) de contrôle des caméras universels pour Arduino. Ce Shield masque la nature complexe de la caméra et simplifie son interfaçage pour son utilisation.

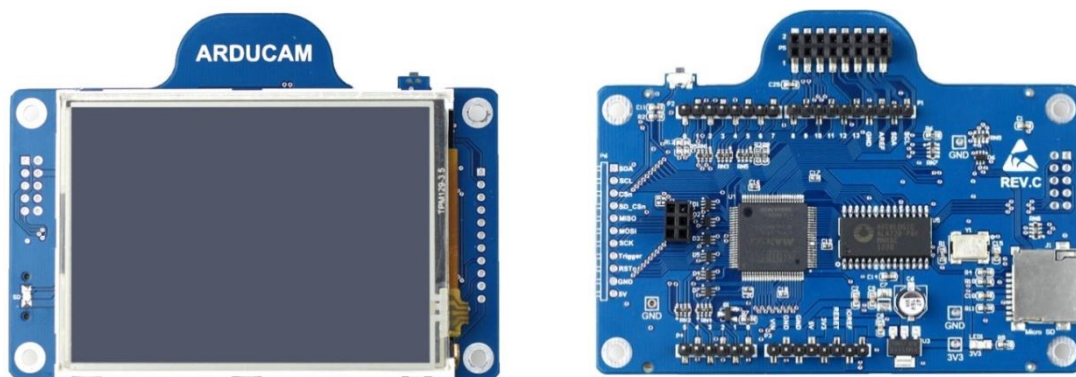


Figure 4.14 : Vues face et profile du Shield Arducam Rev.c

Il fournit une bibliothèque de code source pour d'éventuelles démonstrations. L'Arducam prend en charge des modules caméra variés de 0,3 MP (méga pixel) à 5 MP avec différents formats comme RAW, RGB, YUV, JPEG et peut être bien couplé avec des cartes Arduino standard. Son couplage avec un capteur de 5 MP en JPEG rend ce Shield très intéressant, ce qui en fait la solution idéale pour les applications IoT (Internet of Thing ou internet des objets en français). Arducam est conçu

pour Arduino, mais ne se limite pas à l'Arduino, il peut être utilisé dans toutes les plates-formes qui fournissent l'interface SPI et I2C comme Arduino, Raspberry Pi, Maple, Chipkit, Beaglebone black.

Voici les caractéristiques du Shield Arducam sous forme de points et d'autres sont décrites dans le tableau 4.5 :

- Supporte des modules de caméra de 0,3 MP à 5 MP,
- Livré avec écran TFT/LCD de 3,2 pouces (option),
- Inclut un support de carte SD/TF,
- Supporte le mode de compression JPEG,
- Prise de vue unique et multiple,
- Possibilité de capturer plusieurs frames à la fois,
- Faible consommation d'énergie,
- Supporte presque toute la plate-forme de microcontrôleur,
- Fournis une bibliothèque de code source ouvert, disponible sur : [github.org/Arducam](https://github.com/ArduCAM)
- Tous les ports d'E/S Arducam résistent aux tensions d'alimentation 5V et 3.3V.

Une fois la constitution du nœud multimédia est achevée, il est indispensable de s'assurer de la connectivité de ce mote avec le module transmission, pour se faire, une configuration de module de communication Xbee attaché avec tous les nœuds de réseau est requise. Le module radio Xbee mit à jour sa configuration en le connectant avec le Shield Xbee Explorer illustré dans la figure 4.11.

Résolution	Fournisseur de capteurs	Module Camera	Fabricant
0.3 MP	Omnivision	OV7660/OV7670/OV7675/OV7725	ArduCAM
0.3 MP	Aptina	MT9V111	ArduCAM
1.3 MP	Aptina	MT9M112/MT9M001	ArduCAM
2MP	Omnivision	OV2640	ArduCAM
2MP	Aptina	MT9D111/MT9D112	ArduCAM
3MP	Omnivision	OV3640	ArduCAM
3 MP	Aptina	MT9T112	ArduCAM
5 MP	Omnivision	OV5640/OV5642	ArduCAM

Tableau 4.5 : Caméras supportées par le Shield Arducam

IV.4.4.5. Xbee explorer

Il permet à la fois de convertir les signaux du réseau Zigbee générés par le module Xbee attaché au capteur en un signal série RS232 (le port USB est considéré comme port parallèle). D'autre part

permet de programmer les modules Xbee et mettre à jour le firmware et configurer le mode de fonctionnement.

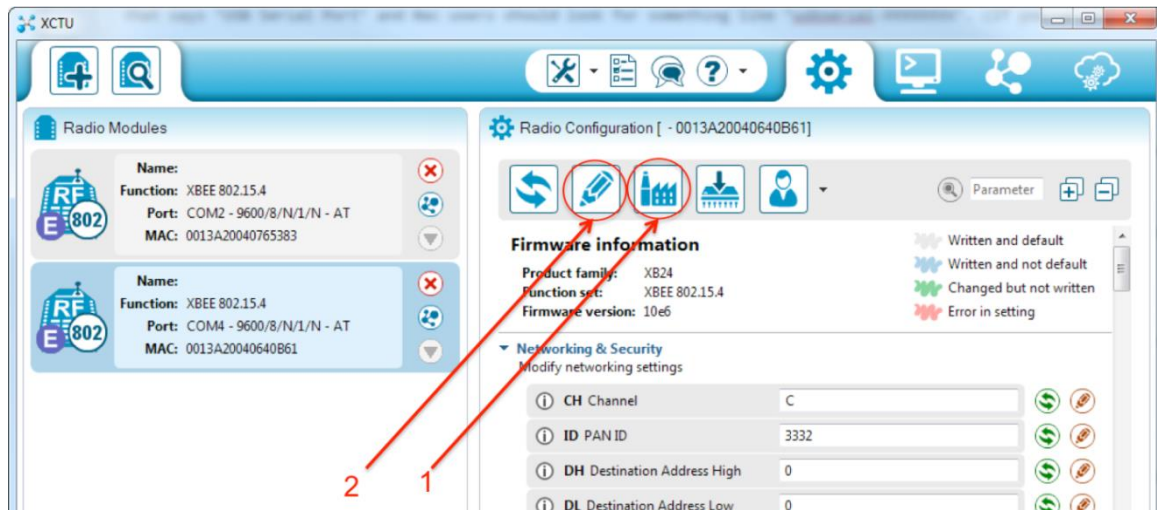


Figure 4.15 : Aperçu du logiciel XCTU

Les développeurs du module Xbee ont développé le logiciel de test et configuration XCTU. Ce logiciel (figure 4.15) permet de configurer tous les modules fabriqués par la firme Digi (Xbee, Bluetooth Bee...), chaque variante possède son propre firmware, et qu'il est aussi possible, voire indispensable, de reconfigurer les différents paramètres dont les plus importants, pour établir une communication :

- Identifiant du réseau (PANID),
- Channel de communication (CH),
- Vitesse de communication (BD),
- Le mode de fonctionnement,
- Et plein d'autres paramètres (pour une utilisation plus poussée).

Le moyen le plus simple pour connecter deux modules Xbee en mode point à point est de configurer les deux modules Xbee communiquant par les valeurs d'usine représentées dans la figure 4.15, en activant les deux icônes indiqués par les flèches 1 et 2.

Après la définition des différents éléments nécessaires pour l'implémentation matérielle de notre système multimodal, il ne reste qu'à définir le programme qui devra être téléversé sur la carte Arduino et lancer le processus de surveillance. Puisque notre solution est multimodale, nous avons développé un code Arduino qui va contenir deux parties, une qui s'occupe de la partie scalaire dont une partie est déjà décrite dans la section IV.4.4, et une qui traite la partie multimédia qui est responsable de la capture l'image.

La figure 4.16 montre l'organigramme qui montre le comportement du nœud capteur pour capturer une image. Le code source d'implémentation correspondant à ce diagramme est donnée dans l'annexe A.

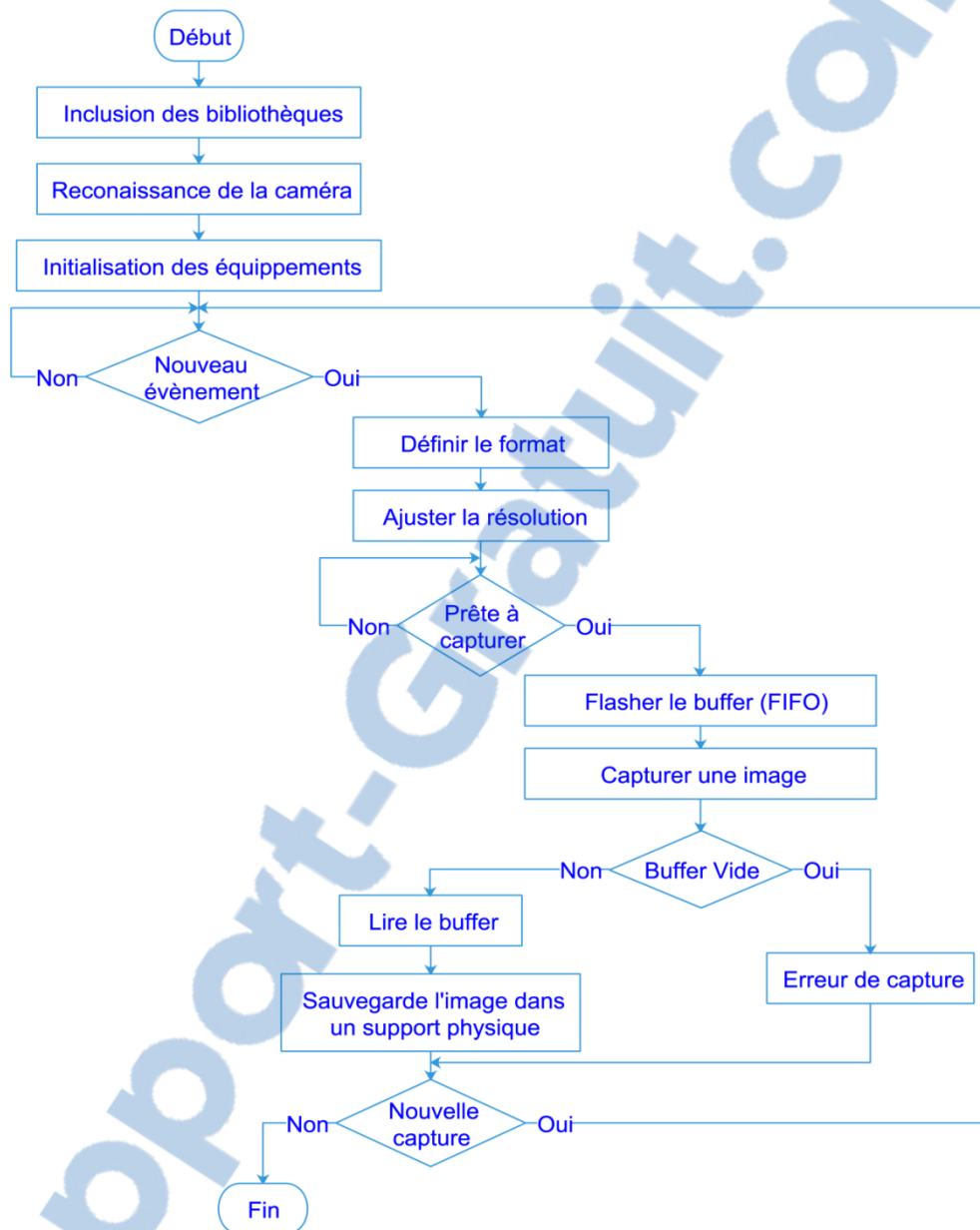


Figure 4.16 : Organigramme de capture d'une image

IV.4.6. La station de base

Une fois le banc d'essai préparé est prêt pour surveiller une région, nous nous intéressons maintenant à son investigation localement. Pour cela, nous avons besoin d'un nœud capteur multimédia doté d'un module de communication, d'autre part, un autre module de communication Xbee sera attaché avec un USB explorer et l'ensemble devrait être connecté à une station. Nous avons choisi un ordinateur de bureau et un autre portable dont les caractéristiques sont données par le tableau 4.6,

Caractéristiques	valeur	
Processeur	Intel i3 3,30 GHz	Intel Core i7 2,2 GHz
Ram	4 Go	8 Go
Disque dur	1 To	500 Go
System d'exploitation	Windows seven 64 bit Linux Ubuntu 14.04 - 64 bit	macOS High Sierra 10.13.6 Linux Ubuntu 14.04 - 64 bit
Logiciels	IDE Arduino v 1.8.4 XCTU v 6.3.2 VMware Workstation Player v 12.1.0 Terminal Python v 2,7 DropBox v 19.4.13 CoolTerm v 1.4.7	

Tableau 4.6 : Caractéristiques des PC utilisés

IV.5. Résultats et discussion

Dans cette section, nous présentons les résultats obtenus dans cette contribution par notre système de détection M2FS. Nous avons évalué des paramètres sensibles, qui ne peuvent pas être obtenus à l'aide des outils de simulation tels que le temps de détection d'un événement (de température par exemple), le temps de transmission de paquets d'alarmes entre deux nœuds communicants et les mesures d'énergie à l'aide d'un oscilloscope et enfin, la détection de fausses alarmes. La figure 4.17 montre un aperçu de déploiement du système M2FS dans un milieu forestier. La petite fenêtre de zoom montrée dans le coin supérieur gauche de la figure est un agrandissement du nœud capteur multimédia utilisé. On trouve aussi un module Xbee attaché à la carte Arduino Uno qui fait l'objet d'un nœud relais, enfin une station de base équipée d'une antenne Xbee (sous forme d'un ordinateur portable).

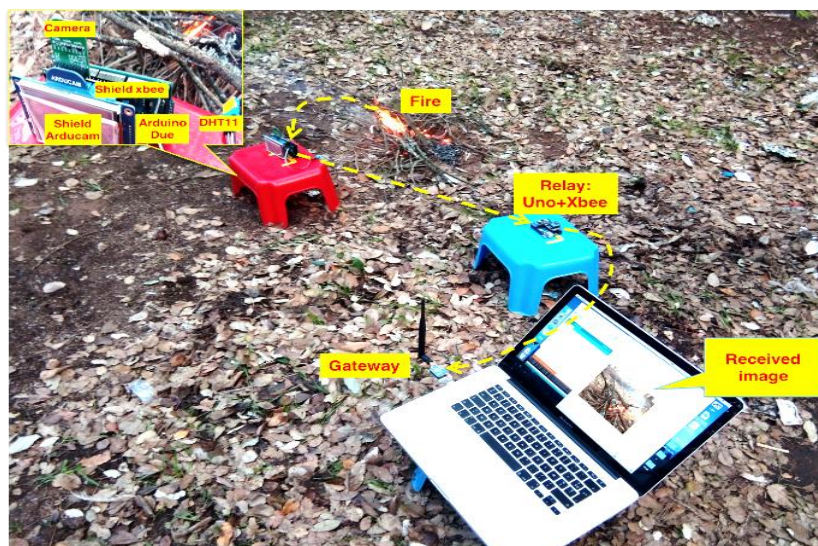


Figure 4.17: Aperçu du banc d'essai déployé correspondant au système M2FS

Une fois le nœud capteur est programmé, déployé, allumé et les communications sont établies, un hyper terminal est lancé dans la station de base afin de recevoir les messages qui s'adressent au module radio attaché à la station de base. La figure 4.18 montre un aperçu d'écran de l'hyper terminal CoolTerm qui affiche les données reçues.

```

Module ArduCAM démarrage...
Camera OV2640 connectée correctement
Carte SD détectée
----- Fin d'initialisation -----
Failed to read from DHT
Humidite: 60.00 %; Temperature: 25.00 Â°C; IFM: 4.75
Humidite: 60.00 %; Temperature: 25.00 Â°C; IFM: 4.75
Humidite: 60.00 %; Temperature: 25.00 Â°C; IFM: 4.75
Humidite: 60.00 %; Temperature: 25.00 Â°C; IFM: 4.75
Humidite: 60.00 %; Temperature: 25.00 Â°C; IFM: 4.75
Humidite: 60.00 %; Temperature: 26.00 Â°C; IFM: 4.88
Humidite: 60.00 %; Temperature: 26.00 Â°C; IFM: 4.88
Humidite: 61.00 %; Temperature: 26.00 Â°C; IFM: 4.77
Humidite: 60.00 %; Temperature: 27.00 Â°C; IFM: 5.01
----- Alerte scalaire ==> Activer la Camera -----
==> Mode : JPEG
==> Initialisation de la camera : Ok
==> Resolution = 160*120
==> Debut de capture
Humidite: 61.00 %; Temperature: 27.00 Â°C; IFM: 4.89
==> Capture terminée
==> Affichage de l'image ...
FF D8 FF E0 0 10 4A 46 49 46 0 1 1 1 0 0 0 0 0 0 FF DB 0 43 0 C 8 9 B 9 8 C B A B E D C E 12 1E
14 12 11 11 12 25 1A 1C 16 1E 2C 26 2E 2D 2B 26 2A 29 30 36 45 3B 30 33 41 34 29 2A 3C 52 3D 41
47 4A 4D 4E 4D 2F 3A 55 5B 54 4B 5A 45 4C 4D 4A FF DB 0 43 1 D E E 12 10 12 23 14 14 23 4A 32
2A 32 4A 4A 4A 4A 4A 4A 4A 4A 4A 4A 4A 4A 4A 4A 4A 4A 4A 4A 4A 4A 4A 4A 4A 4A 4A 4A 4A 4A 4A
4A 4A 4A 4A 4A 4A 4A 4A 4A 4A 4A 4A 4A 4A 4A 4A 4A 4A 4A 4A 4A 4A 4A 4A 4A 4A 4A 4A 4A 4A
0 0 0 0 0 0 1 2 3 4 5 6 7 8 9 A B FF C4 0 B5 10 0 2 1 3 3 2 4 3 5 5 4 4 0 0 1 7D 1 2 3 0 4 11 5
12 21 31 41 6 13 51 61 7 22 71 14 32 81 91 A1 8 23 42 B1 C1 15 52 D1 F0 24 33 62 72 82 9 A 16
17 18 19 1A 25 26 27 28 29 2A 34 35 36 37 38 39 3A 43 44 45 46 47 48 49 4A 53 54 55 56 57 58 59
5A 63 64 65 66 67 68 69 6A 73 74 75 76 77 78 79 7A 83 84 85 86 87 88 89 8A 92 93 94 95 96 97 98
99 9A A2 A3 A4 A5 A6 A7 A8 A9 AA B2 B3 B4 B5 B6 B7 B8 B9 BA C2 C3 C4 C5 C6 C7 C8 C9 CA D2 D3 D4
D5 D6 D7 D8 D9 DA E1 E2 E3 E4 E5 E6 E7 E8 E9 EA F1 F2 F3 F4 F5 F6 F7 F8 F9 FA FF C4 0 1F 1 0 3
1 1 1 1 1 1 1 1 0 0 0 0 0 0 1 2 3 4 5 6 7 8 9 A B FF C4 0 B5 11 0 2 1 2 4 4 3 4 7 5 4 4 0 1 2
77 0 1 2 3 11 4 5 21 31 6 12 41 51 7 61 71 13 22 32 81 8 14 42 91 A1 B1 C1 9 23 33 52 F0 15 62
72 D1 A 16 24 34 E1 25 F1 17 18 19 1A 26 27 28 29 2A 35 36 37 38 39 3A 43 44 45 46 47 48 49 4A

```

Figure 4.18 : Aperçu des résultats obtenus

Cette fenêtre contient les messages reçus sous forme de mesures météorologiques (indiquées par la flèche n° 1) et l'indice forêt météo IFM (le rectangle n° 2), le déclenchement de l'alarme suite au dépassement d'une valeur seuil (rectangle entre n° 2 et n°3) et l'envoi de l'image sous forme hexadécimale (la flèche n° 4) par la suite. Nous détaillons dans la partie réception des données reçus cette figure.

Remarque :

La section n° 4 dans la Figure 4.18 (le cadre) montre une partie de l'ensemble des valeurs hexadécimales de l'image capturée par la caméra.

IV.5.1. Contrainte temporelle

IV.5.1.1. Temps de détection

Le tableau 4.7 montre le temps de détection nécessaire pour la capture de la température et de l'humidité (T_{TH}), temps de calcul nécessaire de l'indice IFM en utilisant les capteurs DHT11

(T_{FWI11}) et DHT22 (T_{FWI22}), et le temps nécessaire pour manipuler une image (détection + compression + chargement dans la mémoire tampon de la carte Arduino) en mode JPEG avec plusieurs résolutions 160x120 (T_{IJ1})... 1600x1200 (T_{IJ9}), aussi en mode BMP avec la seule résolution offerte par le Shield Arducam 320x240 (T_{IBMP}).

Application		Caractéristiques		Temps (ms)		
				DUE	UNO	YUN
Température & Humidité T_{TH}		DHT11/DHT22		271	271	271
Calcul de l'IFM		DHT11 T_{FWI11}		2	4	3
		DHT22 T_{FWI22}		1	3	3
Manipulation de l'image	JPEG	1	160x120 T_{IJ1}	12	22,4	18
		2	176x144 T_{IJ2}	13,4	26	20,6
		3	320x240 T_{IJ3}	28	53,2	42,6
		4	352x288 T_{IJ4}	33,4	62,25	52
		5	640x480 T_{IJ5}	66	138,6	94,6
		6	800x600 T_{IJ6}	96,8	200	142,2
		7	1024x768 T_{IJ7}	149	290,75	218,8
		8	1280x1024 T_{IJ8}	222	421,6	332
		9	1600x1200 T_{IJ9}	320,2	630	527,2
	BMP	320x240 T_{IBMP}		1130	1337	1434

Tableau 4.7: Temps de détection pris par les capteurs de température/humidité et image

Selon nos mesures, le temps requis pour détecter un incendie (noté T_{DF}) est la somme du temps nécessaire pour détecter les grandeurs physiques (température et humidité) notées T_{TH} et le temps nécessaire pour calculer l'indice IFM (FWI) noté T_{FWI11} ou T_{FWI22} lorsque DHT11 et DHT22 sont utilisés respectivement, comme décrits par l'équation 4.1 :

$$T_{DF} = T_{TH} + T_{FWI11} \quad (4.1)$$

En utilisant cette formule et selon le tableau 4.7, le temps nécessaire pour détecter un incendie est de 273 ms avec le capteur DHT11 et de 272 ms avec le capteur DHT22.

Pour confirmer la présence d'un incendie, nous capturons une image. Le temps de détection globale nécessaire (avec capture multimédia) sera calculé ainsi :

$$T_{DF} = T_{TH} + T_{FWI11} + T_{IJ1} \quad (4.2)$$

Bien que l'équation 2 puisse prendre plusieurs formes, chaque forme dépend de la résolution choisie et du mode de capture (JPEG ou BMP).

Notons que chaque délai indiqué dans le tableau 4.7 est mesuré par la différence temporelle entre l'interruption générée pour le déclenchement de l'événement (température, humidité, image et IFM)

et l'instruction qui suit le lancement de cette interruption. Un exemple de la manière dont ce délai est calculé est donné par la figure 4.19.

```

70 gap=millis(); //Lancement du temporisateur
71 float h = dht1.readHumidity(); //Interruption : lecture de l'humidité
72 Thum=millis()-gap; //Fin de temporisateur et calcul de délai
73 Serial.print("Le temps nécessaire pour capturer l'humidité est = ");
74 Serial.println("Thum");

```

Figure 4.19 : Exemple de calcul d'un délai

Le temps de détection d'un événement dans les applications critiques telles que la surveillance environnementale est un paramètre très important pour lequel nous devons accorder une importance particulière.

Cependant, pour confirmer la présence d'un incendie, il suffit de capturer une image de petite résolution et qui illustre bien l'incendie. Donc, dans notre cas, nous pouvons dire que 284 ms (T_{DF}) est le temps nécessaire pour détecter avec confirmation la présence d'un incendie selon le tableau 4.7 et l'équation 4.2.

Bien que le choix de la carte Arduino Due par rapport aux deux cartes Arduino Uno et Arduino Yun a plusieurs intérêts, nous citons dans ce qui suit trois avantages :

- Premièrement, la rapidité de traitement, conformément au tableau 4.7 toutes les tâches (opération de captures) exécutées par la carte Arduino Due prennent moins de temps par rapport à celles exécutées par les deux autres cartes Arduino (Uno et Yun), et cela revient aux capacités de traitement que dispose cette carte.
- Deuxièmement, la richesse de cette carte en pin d'E/S. Le brochage de notre banc d'essai montré dans la figure 4.12 nécessite l'utilisation de 41 pins pour connecter les différents modules (sans compter les pins du Shield Xbee). Si nous utilisons les cartes Arduino Uno et Yun pour réaliser ce brochage, la totalité des pins sera occupée par le Shield Arducam, or les autres composants n'auront pas la chance d'être connectés.
- Troisièmement, la capacité mémoire de la carte Arduino Due, notre application de surveillance programmée sous l'IDE Arduino occupe 328 lignes dans son code source, ce code est compilé pour les trois cartes du tableau 4.7 citées plus haut et prennent l'espace mémoire suivant : 29 084 octets pour l'Arduino Uno, ce qui représente 90 % de l'espace mémoire (32 ko) offert par cette carte ; 58 404 octets est l'espace mémoire nécessaire par notre code compilé pour la carte Arduino Due, et cela représente 11 % de l'espace total disponible (512 ko) sur cette carte ; 31 346 octets est la taille du code compilé pour l'Arduino Yun qui représente 109 % avec une

insuffisance de stockage sur cette carte. Les ressources mémoire (28 ko mémoire flash et 4 ko pour le boot loader) de cette carte ne nous permettent pas de l'utiliser pour notre projet.

IV.5.1.2. Temps d'envoi

Théoriquement, le temps nécessaire pour envoyer une image de $160 \times 120 = 19\,200$ pixels avec une vitesse de transfert de 115 200 b/s est calculé ainsi.

La plupart des modèles de simulation ou des études analytiques considèrent que le temps de transmission de la trame est le délai de transmission uniquement.

Cependant, avant de pouvoir émettre une trame, le module radio doit recevoir la trame dans son tampon de transmission. Par conséquent, il faut tenir compte de ce temps qui est loin d'être négligeable, car la plupart des communications série ajoutent également 2 bits de surcharge – *overhead* – 1 bit de début et 1 bit d'arrêt pour chaque donnée de 8 bits [106]. Et comme chaque pixel est codé sur 8 bits, dorénavant, notre image ($160 \times 120 = 19\,200$ pixels) doit consommer dans la mémoire tampon du module radio $19200 * 3 * (8 + 2) = 576\,000$ bits.

Par conséquent, avec un débit de transfert de 115 200 b/s, le transfert de notre image (576000 bits) doit nécessiter $576\,000 / 115\,200 = 5$ secondes.

Pratiquement, et en utilisant notre prototype, l'image capturée et compressée est de taille 1728 octet, En appliquant les calculs de la partie théorique [taille = $1728 * 3 * (8+2) = 51840$ bits, temps de mise en buffer = $51840 / 115200 = 0.45$ seconde). C'est approximativement le temps (423 ms) affiché à la fin de la fenêtre illustrée par la figure 4.20 (suite cachée de la figure 4.18 déjà exposée).

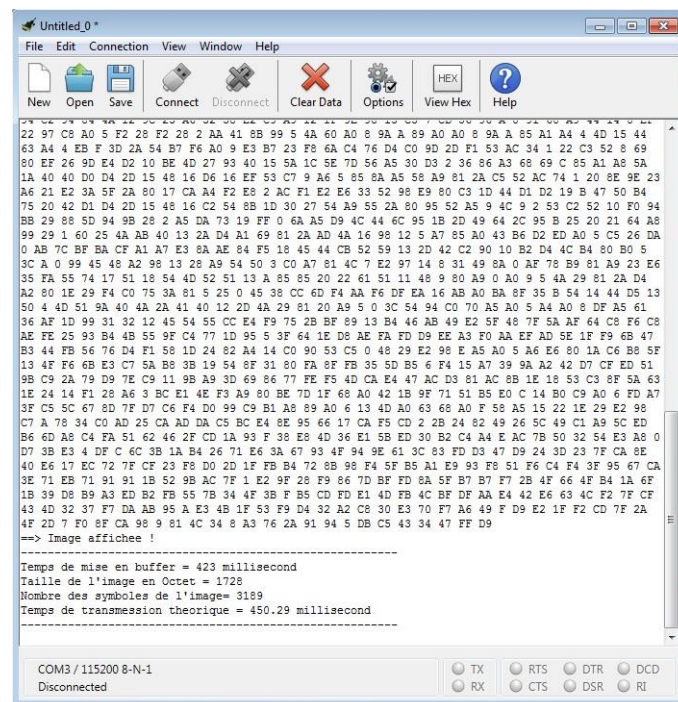


Figure 4.20 : Aperçu sur le reste des résultats obtenus (suite de la figure 4.18)

En pratique, le temps de transmission dépend de ce que le nœud envoie réellement par le module radio (image + messages supplémentaires ou surcharge). Ce temps est calculé par différence entre le temps d'envoi du premier bit émis par l'émetteur et le temps de réception du dernier bit reçu par le récepteur.

Prenant comme étude de cas notre image de dimension 160x120 (19 200 pixels) capturée avec notre banc d'essai (voir plus de détails dans la figure 4.20). Conformément à la norme 802.15.4, si on considère une transmission en unicast avec un exposant *backoff initial* (BE) mis à 0 (par défaut 3), nous avons besoin en général d'un minimum de $(5,44 \text{ ms} + t)$ pour envoyer un seul paquet de 100 octets s'il n'y a pas d'erreur [22]. Le délai t est calculé selon le débit de transfert utilisé (115 200 bps pour notre cas), appliquant aussi la règle d'ajout de deux bits pour le mode de communication série (2 bits de surcharge, 1 bit de début et 1 bit d'arrêt pour chaque donnée de 8 bits) donc notre paquet de 100 octets devrait contenir $100 \times (8 + 2) = 1000$ bits, par conséquent le délai $t = 1000 / 115\,200 = 8,68 \text{ ms}$. Donc, pour envoyer un octet en utilisant ce débit de transfert (115 200 b/s), nous avons besoin de $5,44 + 8,68 / 100 = 0,141\,2 \text{ ms}$.

Revenant à notre cas d'étude, notre charge de 3189 caractères aura besoin de $3189 \times 0,141\,2 = 450,28 \text{ ms}$ pour la transférer (T_t).

Pour confirmer ces calculs, nous avons utilisé le signal généré par un oscilloscope numérique (figure 4.22) utilisé pour la consommation du courant dans le temps. Le détail du brochage du nœud capteur est expliqué dans la section 5.2. La figure 4.21 montre deux signaux de consommation d'énergie, un (Blue) pour le nœud capteur multimédia (l'émetteur), et le deuxième (jaune) pour le Sink. En récupérant les mesures numériques dans un outil de calcul (Matlab) nous avons récupéré les transitions (oscillations significatives) du signal par les tanches T1, T2, T3, T4 et T5 qui représentent les grandeurs suivantes:

- T1 : le temps nécessaire pour l'initialisation de l'appareil, suite à la détection d'un événement,
- T2 : le temps de capture de l'image (12 ms selon le tableau 4.7) avec une latence de 1 seconde pour passer aux tâches suivantes,
- T3 : le temps de chargement de l'image dans le buffer, ce temps intègre un deuxième temps processus qui se lance après un laps de temps, qui est le temps de transmission de l'image montrée en jaune par T_t calculé théoriquement plus haut (450.28 ms),
- T4 : une autre latence pour terminer la transmission,
- T5 : le temps d'acquiescement et de fermeture de connexion.

L'ensemble de ces petites tranches (4102 ms) représente le temps du processus de capture et de transmission de l'image.

Le deuxième signal présenté dans la figure 4.21 (jaune) illustre la consommation d'énergie du nœud récepteur. Notre but au départ est de mesurer le temps nécessaire pour transférer le flux multimédia de bout en bout. Malheureusement, les niveaux du signal n'apparaissent pas à cause d'un très grand bruit qui enveloppe ce signal. Dans de tels cas, il est nécessaire d'appliquer un filtre (filtre passe-bas) pour éliminer ces bruits, ce qui nécessite des connaissances avancées en traitement du signal.

Aussi, dans le premier signal, une application d'un filtre permet de bien représenter le signal propre et de donner plus de précision de calcul dans les délais T1... T5 et Tt.

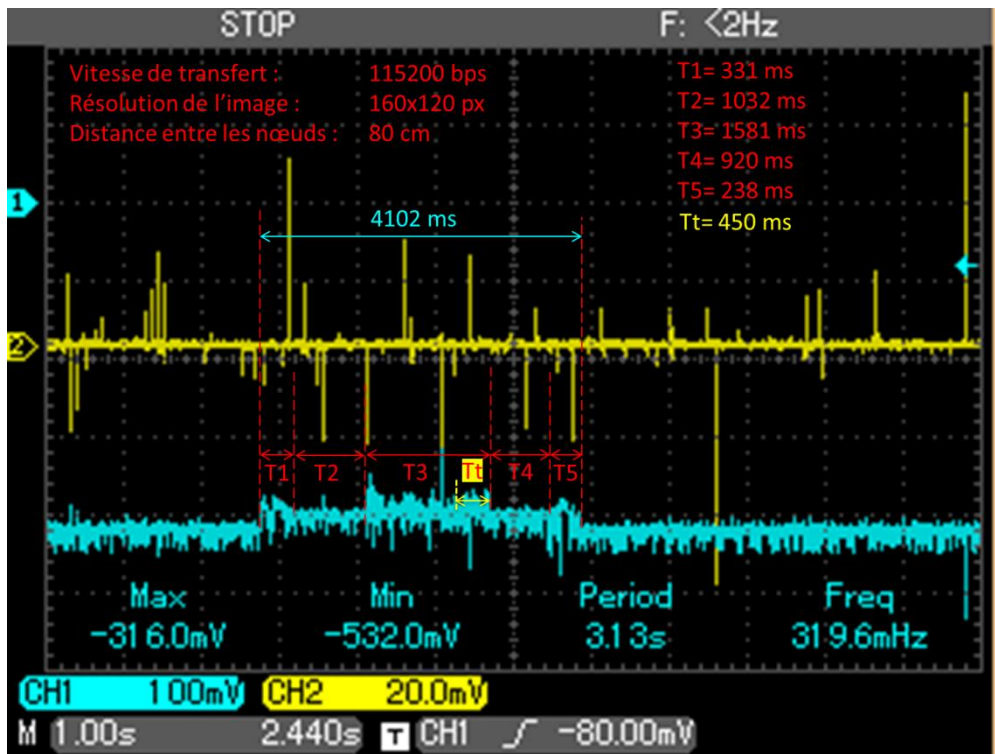


Figure 4.21 : Aperçu des signaux mesurés par un oscilloscope

IV.5.2. Consommation d'énergie

La consommation d'énergie est un enjeu critique dans le choix de la technologie à envisager pour l'implémentation d'une application. En règle générale, plus l'énergie consommée par un appareil est réduite, plus ce dernier sera souhaitable pour être utilisé dans l'expérimentation réelle des applications.

Notre but dans cette section est de mesurer réellement la quantité d'énergie consommée par notre système de détection M2FS. Cela va présenter une référence et une manière qui sera utilisée pour connaître la vraie consommation de ce banc d'essai, et d'en prévenir de la durée de vie de réseau.

Pour ces mesures, nous avons utilisé deux oscilloscopes numériques qui offrent les mêmes résultats (*Siglent SDS1042D* et *MCP lab electronics DQ8104* – Figure 4.22). Pour faire communiquer l'œil de l'électronicien (l'oscilloscope) avec le nœud capteur, nous allons connecter les bornes de la sonde de l'oscilloscope avec une résistance (1 Ω) branchée en série sur le fil (+) de l'alimentation de l'équipement en question, la méthode de brochage est montrée dans la figure 4.22.

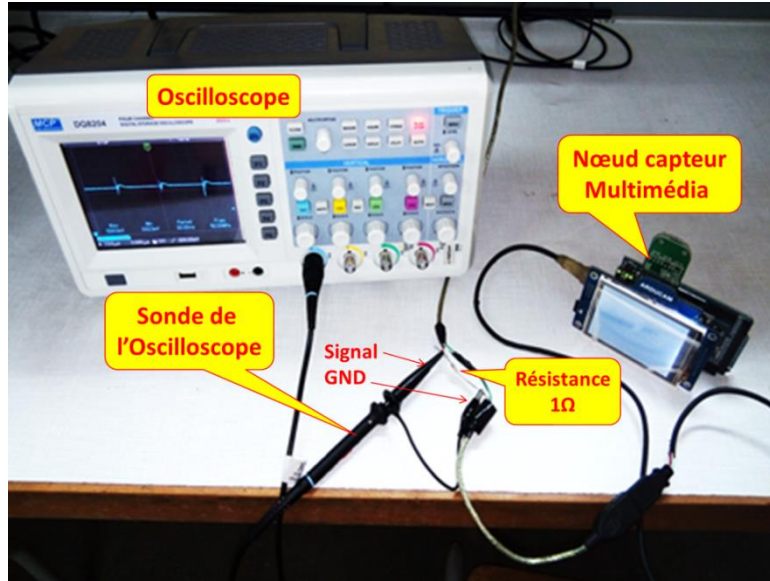


Figure 4.22 : Brochage du nœud capteur utilisé avec l'oscilloscope

La Figure 4.23 montre la consommation d'énergie de quelques applications détaillées dans le tableau 4.8 en utilisant quelques technologies (Micaz, TelosB XM1000, Arduino Uno, Arduino Due).

L'oscilloscope mesure instantanément la tension consommée par un équipement. Dans notre cas, l'oscilloscope permet de mesurer plutôt le courant aux bornes de la résistance de 1 Ω au lieu de la tension, car :

$$U(volts) = R(\Omega) * I(ampère) \quad (4.3)$$

Et puisque $R = 1 \Omega$ l'équation devient :

$$U = I \quad (4.4)$$

Ensuite, la puissance du signal en watt est donnée par l'équation suivante :

$$P(watt) = U(volts) * I(ampère) \quad (4.5)$$

La consommation d'énergie d'un signal représente la surface entre ce signal et l'axe des abscisses (axe du temps) dans l'intervalle de mesure Δt . Pour estimer cette consommation, on applique la formule suivante :

$$P(joule) = P(watt) * \Delta t(secondes) \quad (4.6)$$

La consommation d'énergie d'un signal se calcule par l'intégration de la fonction du courant dans un intervalle. Dans le cas où la fonction de courant est non disponible (comme dans notre cas), il devient nécessaire d'interpoler (opération mathématique permettant de construire une courbe à partir des données d'un nombre fini de points [106]) les données discrètes issues par l'oscilloscope afin de construire la forme du signal montré dans l'écran de l'oscilloscope. Il reste maintenant qu'à calculer l'intégrale sur la forme interpolée dans l'intervalle Δt . Une deuxième façon – plus simple – peut être appliquée pour intégrer sur l'intervalle Δt qui s'applique aussi sur les données discrètes est la méthode de Trapèze ou Simpson.

La figure 4.23 montre la consommation d'énergie de quelques plateformes (Micaz, TelosB, Arduino Uno, Arduino Due). Ces plateformes sont programmées à chaque fois par une fonction qui fait partie du programme principal de notre système de surveillance. Le tableau 4.9 montre les éléments nécessaires qui interviennent dans l'application installée selon le type de plateforme utilisée et la figure 4.23 résume les consommations énergétiques de certaines plates-formes selon l'application (chaque ligne du tableau 4.9).

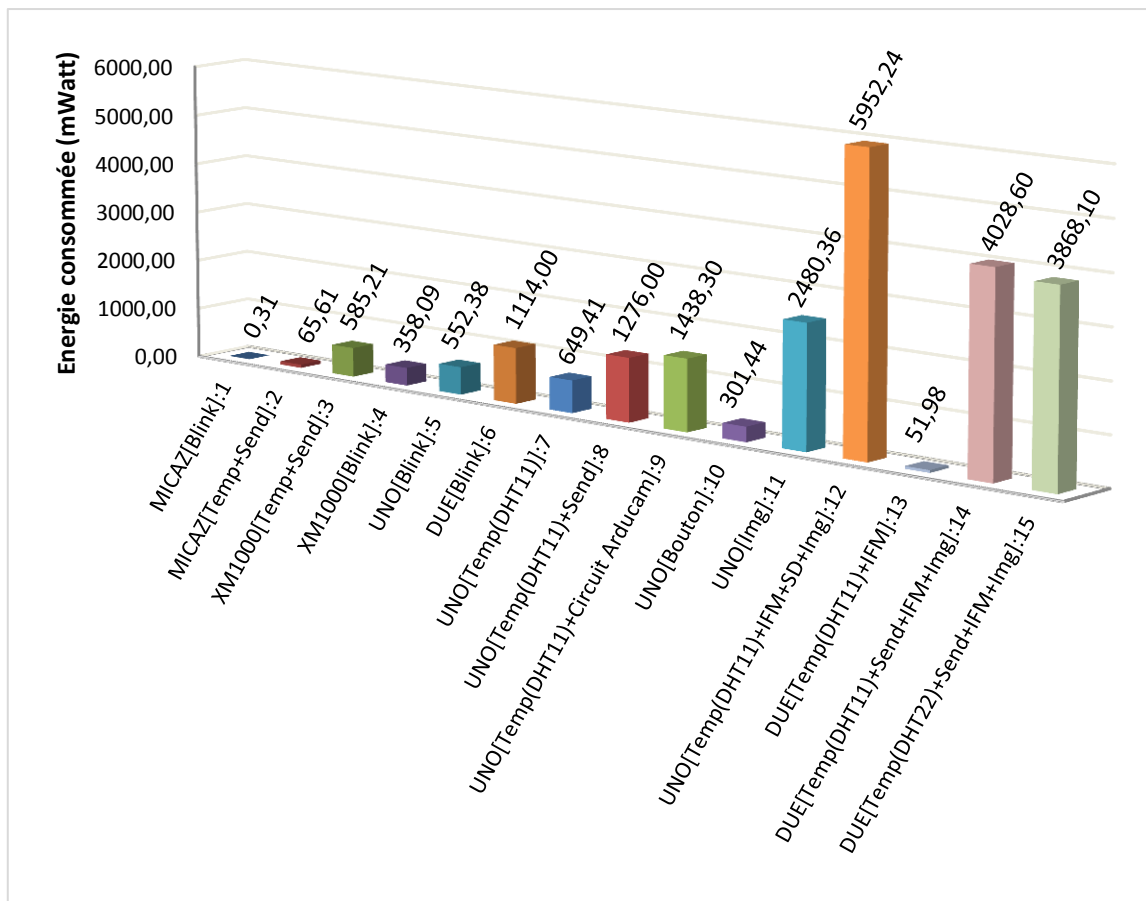


Figure 4.23 : Consommation énergétique de quelques plateformes selon l'application installée (chaque ligne du tableau 4.8)

Nœud	N	Application						
		Allumer la led (Blink)	Capturer la Température (Temp)	Envoi de données (Send)	Pression du bouton de prise de photo (Bouton)	Prise de photo (Img)	Calcul de l'IFM (IFM)	Stockage dans la carte SD (SD)
Micaz	1	✓						
	2		✓	✓				
XM1000	3		✓	✓				
	4	✓						
Uno	5	✓						
	7		avec DHT11					
	8		avec DHT11	✓				
	9 ⁴		avec DHT11					
	10				✓			
	11					✓		
	12		avec DHT11			✓	✓	✓
Due	6	✓						
	13		avec DHT11				✓	
	14		avec DHT11	✓		✓	✓	
	15		avec DHT22	✓		✓	✓	

Tableau 4.8 : Composantes de chaque application installée sur une plateforme

Le but derrière la mesure de la consommation d'énergie sur plusieurs plateformes est triple :

- Premièrement, connaître la quantité consommée réellement par le nœud avec le circuit activé selon la fonction programmée (blink, température, envoi...), car dans le document technique (datasheet) seule la consommation d'un élément est donnée (capteur Dht11 par exemple), et cela néglige bien sûr les quantités consommées par les autres circuits et les autres fonctions comme le calcul, le traitement, la transmission...
- Deuxièmement, comparer les quantités consommées par une fonction sur plusieurs plateformes afin de choisir celle la plus optimale (moins de consommation).
- Troisièmement, prévenir de la durée de vie du réseau à déployer, et de préparer les ressources énergétiques suffisantes pour assurer le bon fonctionnement sur la durée souhaitée.

⁴ Le Shield ArduCAM est attaché

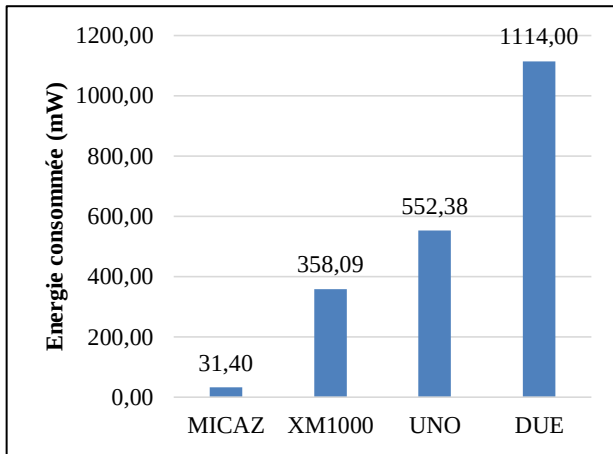


Figure 4.24 : Énergie consommée par quelques nœuds programmés par l'application Blink (led)

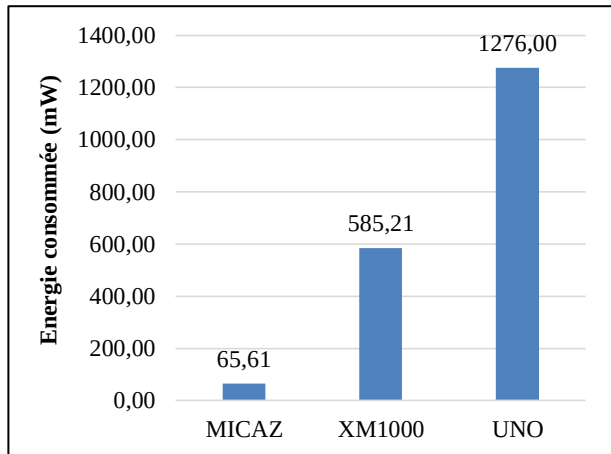


Figure 4.25 : Énergie consommée par quelques nœuds programmés pour capturer la température et l'envoyer au Sink

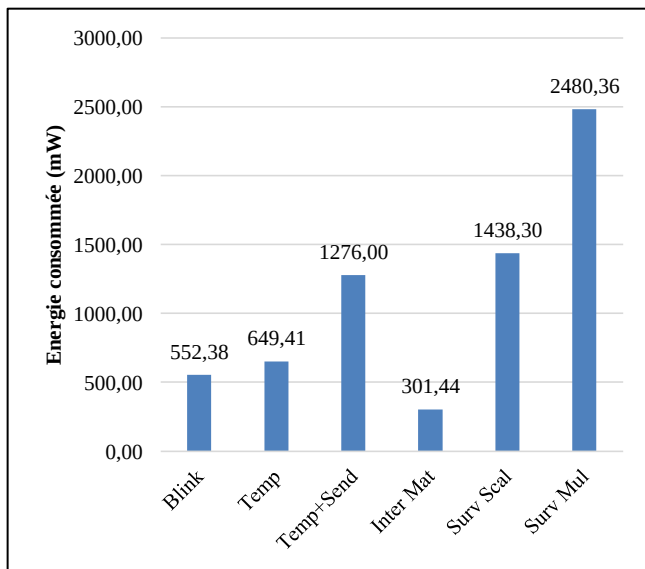


Figure 4.26 : Énergie consommée par Arduino Uno programmé par les fonctions du programme principal

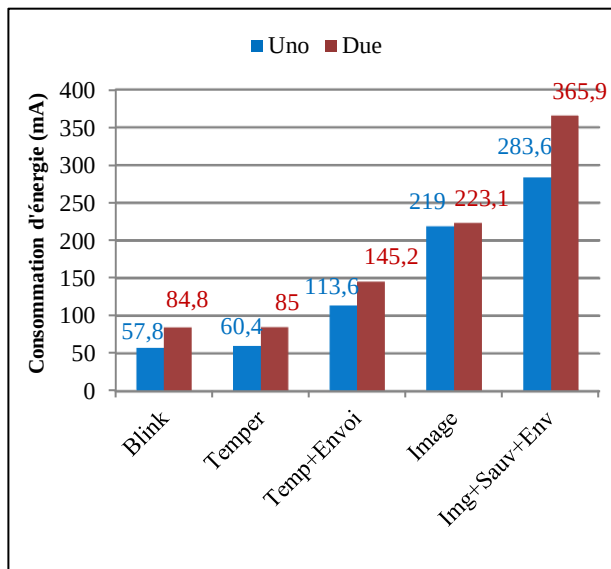


Figure 4.27 : Courant consommé par Arduino Uno et Due programmés par les fonctions du programme principal

Les figures 4.25, 4.26, 4.27 et 4.28 illustrent la consommation d'énergie de quelques applications en variant la technologie utilisée (la plateforme) à chaque fois. La figure 4.24 montre la quantité d'énergie minimale consommée par un nœud capteur (les circuits : processeur + ram + Led) des capteurs Arduino (Due, Uno), XM1000 de la famille TelosB et Micaz. Le rôle de la Led est d'indiquer à l'utilisateur de la bonne exécution d'une instruction, ce circuit ne consomme qu'une petite quantité – négligeable – en comparaison avec les autres circuits. Dans un deuxième temps nous avons gardé les mêmes plateformes dans la figure 4.25, et nous avons installé une application de capture de température et d'envoi de la mesure en mode *Broadcast*. Le but de cette expérience est d'illustrer principalement les quantités consommées par le circuit radio, puisque le circuit de température ne consomme pas beaucoup d'énergie. Les résultats affichés d'après ces deux figures

montrent toujours que le nœud capteur Micaz est le nœud le moins gourmand en électricité dans les deux applications installées par rapport aux autres nœuds concurrents. Cet avantage favorise l'utilisation de la technologie Micaz dans les applications de capture des grandeurs physiques et de la transmission (ou les retransmissions). TelosB (avec sa gamme XM1000) montre aussi un écart important dans la consommation d'énergie par rapport à la technologie Arduino.

La figure 4.26 montre la consommation d'énergie consommée par un nœud Arduino Uno utilisant les applications : Blink (Led), Temp (température), Temp+Env (capture de température et sa transmission via le module radio), SurvScal (est la surveillance scalaire, c.-à-d. capturer les paramètres météorologiques et calculer l'indice IFM), Inter Mat (est l'intervention matérielle pour prendre une image par le module Arducam – pression du bouton de prise de photo déclenchée manuellement–) et enfin, SurvMul (est la surveillance multimédia qui inclut à la fois la surveillance scalaire et la capture d'image). Enfin, la figure 4.27 montre le courant instantané pris par les nœuds Arduino Uno et Due pour les principales fonctions du programme de surveillance. Ces quantités sont mesurées cette fois-ci à l'aide d'un multimètre connecté en série avec les deux cartes (juste pour comparer la consommation des deux plateformes Uno et Due, en l'absence de l'oscilloscope). Les trois premières fonctions permettant d'allumer une Led, capturer la température, capturer la température et l'envoyer en *Broadcast*. La quatrième fonction permet de capturer une image, alors que la dernière permet de prendre une scène, la sauvegarder dans la carte SD et l'envoyer aussi en mode *Broadcast*. Nous remarquons que dans toutes ces applications le nœud Uno est moins gourmand en énergie par rapport au nœud Due, il peut donc être utilisé dans la majorité des applications. Mais pour notre application de surveillance, Uno présente des ressources limitées en stockage et traitement et connectivité des différents équipements (caméra, Xbee, carte sd et capteur DHT22 ensemble).

Il est intéressant de noter que les mesures d'énergie d'une application avec un nœud capteur varient selon plusieurs critères (tension d'alimentation, durée d'expérience, fréquence de répétition de l'événement mesuré, la résistance utilisée et l'équipement de mesure...) mais la variation de mesure (l'erreur) est négligeable par rapport à la valeur de mesure.

IV.5.3. Réception des données capturées

Les figures 4.18 et 4.20 illustrent des captures d'écran de l'Hyper terminal qui montent le contenu des paquets reçus par le Sink. Dans la figure 4.18, on constate bien les données scalaires capturées et affichées dans la zone 1, accompagné d'une vérification de la valeur IFM à chaque capture. Ces deux tâches constituent le traitement du premier tiers – partie scalaire –, où une alerte – scalaire – est déclenchée une fois l'indice IFM dépasse la valeur seuil ($IFM \geq 5$) affichée dans la zone n° 2 et définie dans le tableau de danger potentiel d'incendie (tableau 3.4 du chapitre 3). Ensuite, le second

tiers est sollicité par l'activation du module caméra afin de prendre une photo de la région où le capteur est placé.

Remarque : Nous supposons bien sûr que le déploiement des capteurs est choisi d'une façon déterministe, où la couverture de l'espace surveillé est assurée et la connectivité entre les nœuds déployés et le centre de traitement et de prise de décision est garantie ! Nous proposons en perspective d'optimiser le nombre de nœuds nécessaires qui permettent d'assurer une totale couverture et aussi une communication de données fiable (sans perte, minimiser les retransmissions...).

La troisième zone montre les préparations nécessaires (mode de capture, résolution...) pour la prise de photo par la caméra, cette dernière applique ces paramètres et produit une photo en hexadécimale (zone 4) compressée et sauvegardée dans la carte mémoire du module Arducam et envoyée via le module radio vers la station de base.

Le superviseur du système M2FS ne pourra pas interroger l'image reçue directement suite à la réception des caractères hexadécimaux, une étape de conversion et d'illustration est indispensable pour l'illustrer. Pour cela, nous avons installé un script en langage python qui examine les ports série et détecte la présence d'une image à partir de l'ensemble des données reçues. La figure 4.28 montre un extrait de ce code qui récupère les données reçues par un port et les enregistre dans un fichier texte :

```

1  import sys
2  import dropbox
3  locations=['/dev/ttyACM0','/dev/ttyACM1','/dev/ttyACM2','/dev/ttyACM3']
4  for device in locations:
5      try:
6          ser = serial.Serial(device, 115200)
7          break
8      except:
9          print "Échec de connecter au port :",device
10         while not connected:
11             serin = ser.read()
12             connected = True
13         text_file = open("image.txt", 'w')
14         text_file.write('0')
15         while 1:
16             if ser.inWaiting():
17                 x=ser.read()
18                 print(x)
19                 text_file.write(x)
20                 if x=="\n":
21                     text_file.seek(0)
22                     text_file.truncate()
23                     text_file.flush()
24
25                 text_file.close()
26                 ser.close()

```

Figure 4.28 : Code source de l'application python qui stocke les données récupérées du port USB dans un fichier texte

Ensuite, une deuxième partie qui transforme les caractères hexadécimaux de l'image délimitée par (FF D8 ... FF D9) en image JPEG dont un extrait est donné par la figure 4.29 :

```
1 output = open("output.jpeg", 'wb')
2 for line in sys.stdin:
3     output.write(bytearray(int(i, 16) for i in line.strip().split(' ')))
4 output.close();
```

Figure 4.29 : Extrait du code source (en python) transférant une image écrite en hexadécimale en image JPEG

Enfin, une image nommée output.jpeg est produite et enregistrée dans le disque dur. La figure 4.30 montre un exemple d'une image construite par ce code :

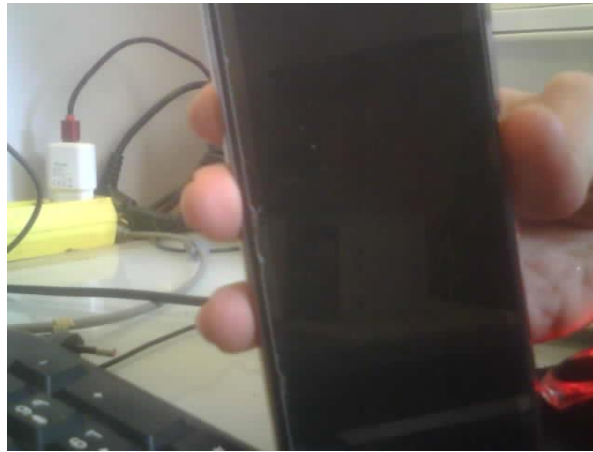


Figure 4.30 : Image construite par le code python montré dans la figure 4.30

Une partie supplémentaire de ce code est ajoutée afin de stocker l'image générée dans un Cloud (dans notre cas DropBox) afin d'étendre les opérations de stockage et de visualisation multiple de cette l'image. La figure 4.31 présente un extrait de code qui permet de télé-verser l'image vers le Cloud.

```
1 client = dropbox.client.DropboxClient("xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx")
2 print ('linked account: ', client.account_info())
3 f = open('/home/seize/Bureau/send/output.jpeg', 'rb')
4 response = client.put_file('/output.jpeg', f)
5 print ('uploaded: ', response)
```

Figure 4.31 : Extrait du code source (en python) permettant de télé-verser une image stockée dans le disque dur vers le *Cloud*

D'autres fonctionnalités sont assurées par le calcul Cloud (*Cloud computing*) autre que le stockage. Les services du Cloud sont divisés en trois catégories [107] : l'infrastructure en tant que service (IaaS), la plateforme en tant que service (PaaS) et le logiciel en tant que service (SaaS).

Notre besoin pour le moment sur l'utilisation du Cloud est de stocker les données et de les interroger à partir de n'importe quel équipement (PC, Tablette, Smartphone...) à n'importe quel endroit dans le monde via le réseau Internet. Pour cela, nous avons créé un compte sur le Cloud DropBox afin d'uploader les images capturées suite à une alarme et de mettre à jour la base de données qui permet de stocker les mesures périodiques.

La figure 4.32 montre une succession d'images capturées dans un intervalle du temps suite à une simulation du feu dans un espace forestier. Ces images montrent l'ampleur du feu dans le temps, et donnent sa direction une fois le nœud qui détecte le feu est repéré.



Figure 4.32 : Succession d'images prises lors d'une simulation du déclenchement du feu

La figure 4.33 montre le contenu de la base de données au moment de la simulation du feu effectuée dans un espace forestier. La première colonne représente la date et l'heure exacte de surveillance, la deuxième colonne représente la température, la troisième colonne correspond à l'humidité et les deux dernières colonnes représentent respectivement la latitude et la longitude du nœud capteur.

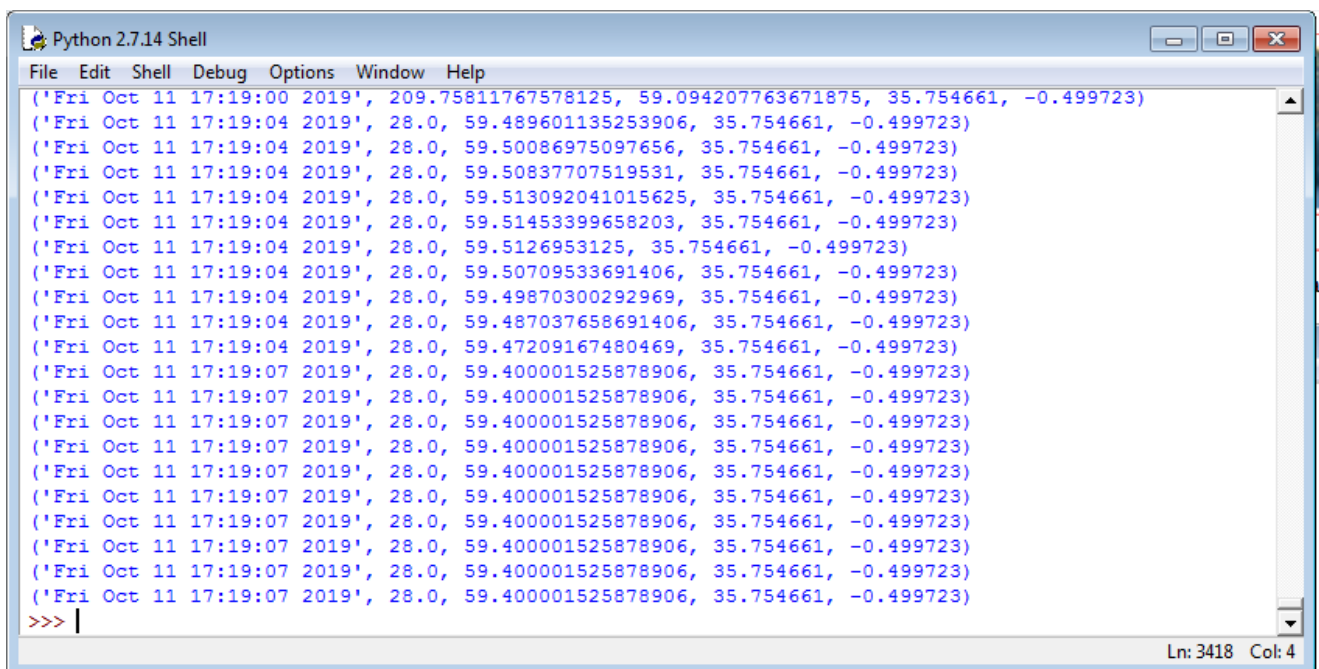


Figure 4.33 : Contenu de la base de données au moment de la simulation d'incendie

IV.6. Conclusion

Dans ce chapitre nous avons présenté notre première contribution. Dans ce chapitre nous avons présenté notre première contribution. Nous avons commencé par présenter l'approche méthodologique adoptée pour implémenter notre solution. Nous avons ensuite modélisé conceptuellement notre système M2FS par des diagrammes UML pour mieux présenter les différentes interactions entre les acteurs communicants, et tracer un acheminement temporel des différentes communications entre ces objets. Nous avons ensuite défini en détail les composants matériels et logiciels nécessaires pour l'implémentation réelle de notre système M2FS, en donnant plusieurs alternatives qui peuvent être utilisées pour la réalisation d'un prototype capable à mesurer, calculer, détecter un incendie et envoyer une alarme. Nous avons aussi discuté les résultats expérimentaux obtenus sur le terrain par le prototype conçu et réalisé en termes de temps de détection et de transmission de la donnée par les différentes alternatives proposées. Nous avons choisi à chaque fois la meilleure solution qui satisfait tous les besoins matériels et qui consomme moins de temps pour son exécution et d'énergie.

Quelques améliorations ont été soulignées à la fin de ce chapitre sous formes de travaux de recherche en perspective qui peuvent apporter robustesse et plus de souplesse à notre système M2FS.

CHAPITRE V

Contribution2 : Compression des données pour l'économie d'énergie dans le système M2FS

V.1. Introduction

Après avoir conçu, implémenté et expérimenté notre système M2FS, nous avons jugé utile de considérer l'aspect optimisation de la consommation énergétique des nœuds qui le composent dans la perspective d'étendre la durée de vie du réseau. Cette optimisation est motivée par le fait que le réseau RSCF sous-jacent de notre système est appelé à fonctionner pour une durée indéterminée en assurant une collecte périodique des données scalaires de l'état de la forêt (mesures météorologiques et indice de danger IFM) par chaque nœud source et leur transmission via le réseau vers le *Sink*. Ces communications (transmissions et réceptions) constituent la source principale de consommation d'énergie, il est ainsi primordial de minimiser au maximum cette consommation énergétique.

Vu que le phénomène étudié (feux de forêt) se développe particulièrement dans la période estivale, et que le nombre de communications périodiques de l'état normal de la forêt (sans présence de feux) constitue les $\frac{3}{4}$ de l'année, la partie la plus importante de données scalaires est générée et communiquée durant cette période. Il devient ainsi important de chercher des solutions efficaces pour diminuer cette quantité de données et permettre aux nœuds capteurs d'économiser l'énergie de leurs batteries. Car, Plus la quantité de données à transmettre d'un nœud capteur augmente, plus la consommation augmente aussi. Les techniques de compressions de données est l'une des solutions envisagées dans le contexte de cette thèse.

La compression de données joue un rôle très important dans ce domaine, et présente des solutions assez prometteuses une fois mises en œuvre. Deux modes de compression peuvent être appliqués à notre système M2FS. Dans ce chapitre nous nous intéressons au premier mode de compression des données scalaires vu son importance quantitative et périodique à travers le temps. Pour le deuxième mode, qui est moins important du moment qu'il est événementiel (se produit rarement), nous avons choisi la méthode de compression proposée par défaut en pratique, qui peut toujours être améliorée dans un travail de recherche future. Nous présentons un état de l'art sur les différentes méthodes de compression de données scalaires dédiées au domaine des réseaux de capteurs sans fil. Nous mettons l'accent sur quelques-unes de ces méthodes jugées proches de la thématique de cette thèse. Nous présentons ensuite notre propre méthode de compression scalaire basée sur l'approximation polynomiale inspirée de [4]. Deux algorithmes sont proposés : Approximation polynômiale avec fenêtre fixe (APFF) et Approximation polynomiale avec fenêtre variable (APFV), selon que la fenêtre de compression est fixe ou variable. Les résultats d'évaluations, par comparaison aux méthodes de compression concurrentes, sont présentés à la fin de ce chapitre.

V.2. Compression de données

V.2.1. Notions générales

Le principal objectif d'un algorithme de compression de données consiste à diminuer la quantité de données pour faciliter son stockage ou son transport, et peut conduire à une réduction généralement considérable des ressources énergétiques. Les algorithmes de compression de données à usage général utilisées dans les ordinateurs ne peuvent pas être implémentés directement dans les nœuds capteurs, vu les ressources limitées dont ils disposent en termes de mémoire de stockage, traitement et bande passante. Néanmoins, plusieurs algorithmes de ce genre ont été adaptés et améliorés de telle sorte qu'ils deviennent fonctionnels dans ces petits dispositifs.

Il est donc important d'adopter plusieurs approches pour réduire la consommation d'énergie des RCSF en fonction de son architecture. Elles incluent principalement trois techniques importantes, à savoir le cycle d'activité (ou *Duty Cycle* en anglais), les approches basées sur les données et la mobilité [28]. Nous nous intéressons ici aux approches basées sur les données.

L'approche basée sur la donnée consiste à réduire le nombre de données échantillonnées tout en maintenant la précision de la détection dans une plage acceptable. Cette approche est subdivisée, comme le montre la figure 5.1, en schémas de réduction de données et schémas d'acquisition de données à haute efficacité énergétique en fonction du problème [28].

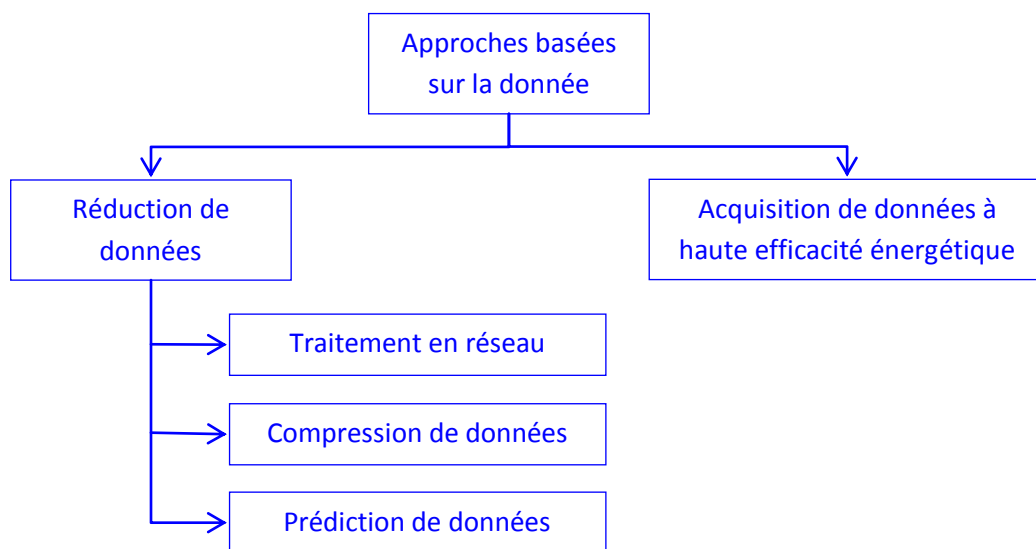


Figure 5.1 : Classification des approches basées sur la donnée [28]

La réduction des données a pour objectif de réduire la quantité de données à transmettre au nœud destinataire (Sink), celle-ci peut être divisée en trois techniques différentes, à savoir :

- a) Traitement en réseau : ce traitement comprend l'agrégation de données en traitement au niveau de nœuds intermédiaires entre la source et le Sink. Ainsi, la quantité de données est réduite lors de la propagation vers le récepteur.
- b) Compression de données : est utilisée pour réduire la quantité de données transmises par les nœuds sources. Elle inclut les informations de codage au niveau des nœuds générateurs de données et leur décodage au niveau du nœud récepteur final. Il existe de nombreux algorithmes de compression de données applicables aux RCSF, nous nous citerons quelques-unes plus loin dans ce chapitre.
- c) Prédiction des données : elle comprend l'abstraction de données détectées, considérée comme un modèle reliant l'évolution des données. Le modèle a la capacité de prédire les valeurs détectées par les nœuds capteurs dans certaines limites d'erreur, et existe à la fois au niveau des capteurs et du récepteur. La prédiction de données aide à réduire le nombre d'informations transmises par les nœuds sources et par conséquent, l'énergie nécessaire à la communication.

V.2.2. Types de compression

Trois types de compression dans le domaine des RCSF peuvent exister, à savoir [28] :

- Compression à la capture (*SC : Sampling compression*) : ce processus permet de limiter le nombre de captures/échantillons dans un intervalle de temps. Cette technique est déconseillée dans les applications de surveillance où la présence d'un événement peut coïncider à un temps de repos des capteurs.
- Compression de donnée (*DC : Data Compression*) : ce processus permet de convertir une séquence de données capturées en une séquence de donnée compressée avec un nombre de bites inférieur. Une étape de vérification de la structure est introduite pour soit identifier et éliminer les redondances de données collectées, soit pour appliquer des règles de choix et de sélection de données importantes décidées à communiquer.
- Compression de communication (*CC*) : ici c'est le nombre de paquets en émission/réception qui est réduit. En d'autres termes, le temps de fonctionnement de circuit de transmission est réduit, ce qui réduit au minimum le nombre des transmissions vers la station de base.

V.2.3. La compression dans les RCSF

La compression de données dans le domaine des RCSF présente un rapport important dans la préservation de l'énergie. Généralement, la communication est l'opération la plus dissipative en énergie, cela aussi est bien obtenu dans nos résultats expérimentaux à l'aide de l'oscilloscope présenté dans le chapitre4, et qui le sera aussi prouvé dans ce chapitre. Par contre, les opérations de traitement (les calculs et la compression) et de capture sont beaucoup moins gourmandes en temps d'exécution et en énergie par rapport à la communication, ce qui favorise les concepteurs

d'applications de bien intégrer lors de la conception de leurs solutions les mécanismes de compression de données.

Cependant, certaines contraintes doivent être respectées pour élaborer une solution de compression efficace telles que la simplicité et la corrélation entre les données [108]. Il est indispensable d'utiliser des mécanismes de compressions légers pour les implémenter dans des dispositifs à ressources limités, simples et non compliqués et qui prennent un temps d'exécution raisonnable. La corrélation ici, permet de définir des méthodes de compression très adéquates et réunit les notions fondamentales pour l'implémentation des algorithmes de compression à utiliser dans les RCSF.

Nous citons dans ce qui suit quelques algorithmes de compression développés spécialement pour les RCSF.

Une technique de compression temporelle légère (LTC : *Lightweight Temporal Compression*) est introduite dans [109]. Elle évalue un schéma de compression temporelle simple en observant sur une courte période, des échantillons de données linéaires, et génère une série de segments qui représentent avec précision les données. Cette précision est déterminée selon la marge d'erreur du nœud capteur. Plus le seuil d'erreur est élevé, plus la solution permet de réaliser des économies de stockage.

K-RLE est une version améliorée de l'algorithme RLE (*Run length coding*), proposée dans [110]. Il consiste à remplacer les n fois la variation de la donnée d dans la plage $[d-k, d+k]$, par la simple paire (n, d) . Où k indique le niveau de précision et d représente un élément de données.

S-LZW (*Sensor LZW*), est un modèle de compression basé sur le dictionnaire LZW (*Lempel Ziv Welch*), spécialement développé par Sadler et Martonosi [111] pour les RCSF. Cet algorithme [28, 111, 112] construit un dictionnaire pour encoder une nouvelle chaîne sur la base de la chaîne d'entrée d'une manière dynamique. Cet algorithme est affecté par un problème croissant de dictionnaire et ses performances de compression doivent en conséquence être améliorées.

La compression d'entropie sans perte (LEC : *Lossless entropy compression*) [108, 113] est une technique de codage prédictif comprenant un prédicteur et un codeur. LEC calcule la différence entre les mesures successives et les répartit en groupes dont la taille augmente de manière exponentielle. Chaque groupe représente un nombre de bits nécessaire pour indiquer les différences de mesure, puis les groupes sont codés en entropie par une table de compression fixe. LEC séquentielle (S-LEC) [108]- version améliorée de LEC-utilise les informations de contexte séquentielles parmi les résidus à proximité pour compresser les données dans les RCSF.

Une autre technique appelée compression adaptative sans perte de données (ALDC : *Adaptive Lossless Data Compression*) est née pour remédier aux inconvénients de la technique LEC [113].

ALDC est capable de s'adapter de manière dynamique aux modifications apportées aux données sources pour améliorer les performances de compression. Il fonctionne en une passe avec plusieurs options de code de manière adaptative et peut être appliqué à différents types de données.

Les auteurs de [114] ont proposé une technique de compression avec perte qui utilise le principe de la modulation adaptative dans un environnement de clustering ADPCM (*adaptive differential pulse-code modulation*). Dans ce schéma, lorsqu'un nœud source se rapproche de la valeur de l'échantillon, il ne transmet que son estimation quantifiée à la place de l'échantillon de données réel.

NIS (*Neighborhood Indexing Sequence*), développé par [108], est un algorithme qui vise à fournir un schéma de compression sans perte simple et efficace pour les RCSF. Il exploite les informations de voisinage sur la base d'une nouvelle procédure appelée « données de parcours basées sur les 0 et les 1 ». En utilisant cette procédure, NIS produit des mots de code de longueur plus courte pour chaque caractère de la séquence en entrée.

V.2.4. Caractéristiques de la compression de données

Nous présentons ici quelques caractéristiques liées aux mécanismes de compressions. Entre autres, elles permettent à l'utilisateur de classer ses besoins afin de choisir la méthode adéquate pour satisfaire les exigences de son application, et en réalisant des économies de stockage [28].

V.2.4.1. Perte de données

Deux familles de compression existent, une avec perte de données, et une deuxième sans perte de données. Dans l'algorithme de compression sans perte, la récupération des données d'origine est reconstruite en appliquant une décompression, alors qu'en mode avec perte, les données reconstruites ne sont pas exactement comme les données d'origine. Le mode avec perte entraîne une perte d'informations, mais garantit des taux de compression élevés.

V.2.4.2. Symétriques et Asymétrique

Dans les algorithmes de compression symétriques, la complexité de calcul liée aux tâches de compression et de décompression est la même. Par contre, en cas d'asymétrie, la complexité de la compression et de la décompression est différente. Le deuxième mode est préférable lors de la conception d'une application destinée pour les RCSF, car la faible complexité est privilégiée pour la compression au niveau des nœuds capteurs et la complexité élevée est laissée pour la décompression effectuée au niveau du Sink, suivant la distribution des niveaux d'énergie.

V.2.4.3. Adaptatifs et non adaptatif

Dans le mode de compressions non adaptatives, l'algorithme de compression (les opérations) et ses paramètres sont fixes et s'appliquent à un ensemble de données stationnaires dans lesquelles les statistiques ne changent pas avec le temps. Par contre, les méthodes de compression adaptatives ou dynamiques comprenant une instabilité des statistiques de données brutes, et par conséquent conduisent à une modification des opérations de compression et/ou des paramètres de l'algorithme pour améliorer ses performances. Les méthodes adaptatives conviennent aux données non stationnaires, mais classées plus complexe.

V.3. L'agrégation de données

V.3.1. Notions générales

L'agrégation est un paradigme souvent lié à la compression, en intégrant des traitements sur les données (calcul de Max, Min, Avg, etc.) à l'intérieur du réseau « *In-network proceeding* » avant qu'elles ne soient extraites pour une analyse externe. Cette opération d'agrégation, parallèlement, supprime les anciens paquets circulés dans le réseau afin d'éliminer les redondances, réunit deux paquets ou plus afin de réduire la charge des paquets, ou utilise des techniques de compression. Les tâches d'agrégation et de compression peuvent être appliquées séparément ou ensemble dans un nœud capteur ou souvent dans des nœuds coordinateurs [4].

Généralement, l'agrégation permet de sélectionner un sous-ensemble de nœuds capteurs pour fusionner (des fois collecter aussi) les données de détection envoyées et collectées par leurs voisins, puis les données agrégées sont transmises au destinataire (Sink). La technique d'agrégation de données est irrécupérable, car les données d'origine ne peuvent pas être dérivées de données agrégées [28].

Lors de l'agrégation de données, les transmissions de données dans le réseau sont réduites suite à la fusion (l'agrégation) des données collectées. Les techniques d'agrégation de données sont généralement simples, et visent à sélectionner les nœuds d'agrégation qui réduisent efficacement le nombre de transmissions de données dans le réseau [8, 9].

V.3.2. Schémas d'agrégation de données

Selon les structures de réseau, les schémas d'agrégation de données peuvent être classés comme suit selon [10, 28] :

- a) Schémas d'agrégation de données en arborescence : Il organise le réseau en une arborescence pour la collecte et l'agrégation de données.

- b) Schémas d'agrégation de données basés sur des grappes : Les grappes sont formées en regroupant des nœuds capteurs, puis chaque tête de grappe agrège les données détectées dans sa grappe.
- c) Schémas d'agrégation de données basés sur une chaîne : les données détectées sont transmises par chaque nœud capteur à son nœud voisin le plus proche et la structure de chaîne est formée par un réseau afin d'agréger les données détectées dans le réseau.
- d) Schémas d'agrégation de données sectoriels : Il adopte un concept de division en secteurs sous forme d'anneau pour regrouper les nœuds capteurs de manière à ce que les nœuds capteurs du même secteur soient assemblés en un seul groupe.

V.3.3. État de l'art

De nombreuses approches d'agrégations ont été développées pour éliminer les redondances de données capturées, nous présentons quelques-unes dans les paragraphes qui suivent.

Dans [115], une approche par échantillonnage adaptatif (ASAP : *adaptive sampling approach*) a été développée. L'idée principale derrière ASAP est d'utiliser un sous-ensemble de nœuds comme nœuds collecteurs qui changent dynamiquement. Les autres nœuds sont excusés de cette tâche momentanément, mais ses valeurs sont prédites à l'aide des modèles probabilistes localement et périodiquement construits.

Dans [116], une agrégation de données de réseaux de propagation arrière (BPNDA : *Back-Propagation Network Data Aggregation*) a été proposée pour agréger les données à l'aide des réseaux de neurones. Il adopte une stratégie hiérarchique de groupement adaptatif à basse consommation d'énergie (LEACH) [117] pour construire le groupe de nœuds capteurs. Un réseau de neurones à trois couches à rétrodiffusion (BP) a été utilisé pour éliminer la redondance des données et améliorer leur précision. Les neurones de la couche d'entrée sont situés au niveau des membres de la grappe, tandis que les neurones de la couche cachée et les neurones de la couche de sortie sont situés au niveau du cluster-chef (CH).

Liu et al. [118] ont proposé une approche de collecte de données efficace en énergie (EEDC : *Energy-Efficient Data Collection*) qui observe la corrélation spatiale et temporelle entre les données détectées à l'aide d'une technique d'approximation linéaire par morceaux. En explorant la corrélation spatiale entre les données détectées, il partitionne de manière dynamique les nœuds capteurs en grappes, de sorte que les nœuds de la même grappe possèdent des séries chronologiques de surveillance similaires. EEDC essaie de minimiser la fréquence des transmissions de données en respectant une limite donnée sur la précision estimée, et minimise la consommation d'énergie en introduisant une technique de planification aléatoire.

Une collecte de données efficace consciente de la corrélation spatio-temporelle (EAST : *Efficient Data Collection Aware Of Spatiotemporal Correlation*) [119]. EAST regroupe les nœuds membres en utilisant une approche de corrélation spatiale, et les nœuds représentatifs utilisent la technique de suppression temporelle afin d'estimer les futures données à partir du journal de données actuel. Cette prédiction diminue le trafic réseau acheminé par la technique de plus court chemin.

Dans [120], ARIMA (*Auto Regressive Integrated Moving Average*) est proposée pour prédire les données à partir du journal de données détectées précédemment, tout en maintenant ces valeurs (prévues) dans les limites du seuil d'erreur défini par l'application.

DPF (*Dual Prediction Framework*) [121] est un schéma qui utilise le système de moindres carrés (LMS) pour filtrer les données. Il ne nécessite aucun modèle préalable et permet aux nœuds capteurs de fonctionner de manière indépendante. La prédiction basée sur LMS a été étendue de manière hiérarchique pour effectuer une prédiction conjointe sur les données provenant de nœuds voisins.

L'élimination de la redondance pour une agrégation de données précises (READA : *Redundancy Elimination for Accurate Data*) [122] a été proposée pour exploiter la corrélation spatiale entre les données dans les RCSF. READA élimine les données redondantes en appliquant une approche de regroupement et de compression, et détermine la valeur aberrante à l'aide de la technique d'extrapolation des moindres carrés.

Dans [123], une technique d'agrégation de données sécurisée basée sur la régression polynomiale (PRDA : *Polynomial Regression Based Secure Data Aggregation*) a été présentée, dans lequel les nœuds capteurs représentent leurs données détectées sous forme de fonctions polynomiales. Au lieu de transmettre les données d'origine, les nœuds capteurs envoient secrètement des coefficients de ces fonctions polynomiales à des agrégateurs de données. L'agrégation de données est effectuée sur la base de ces coefficients et la station de base est capable d'extraire une bonne approximation des données de réseau à partir du résultat de l'agrégation.

V.4. La corrélation de données

Les données collectées par un RCSF forment à un instant donné une corrélation spatiale entre ces données. De plus, la nature du phénomène physique elle-même constitue une corrélation temporelle entre deux mesures consécutives [124]. Les corrélations prises en compte dans les mécanismes de compression dans les RCSF peuvent être divisées en trois catégories, à savoir, corrélation spatiale, corrélation de données et corrélation temporelle. La figure 5.2 illustre cette classification en citant quelques méthodes de compression de données.

La corrélation spatio-temporelle influe d'une manière rentable sur les méthodes de compression et de l'agrégation de données qui s'intéressent surtout aux applications de surveillances [125].

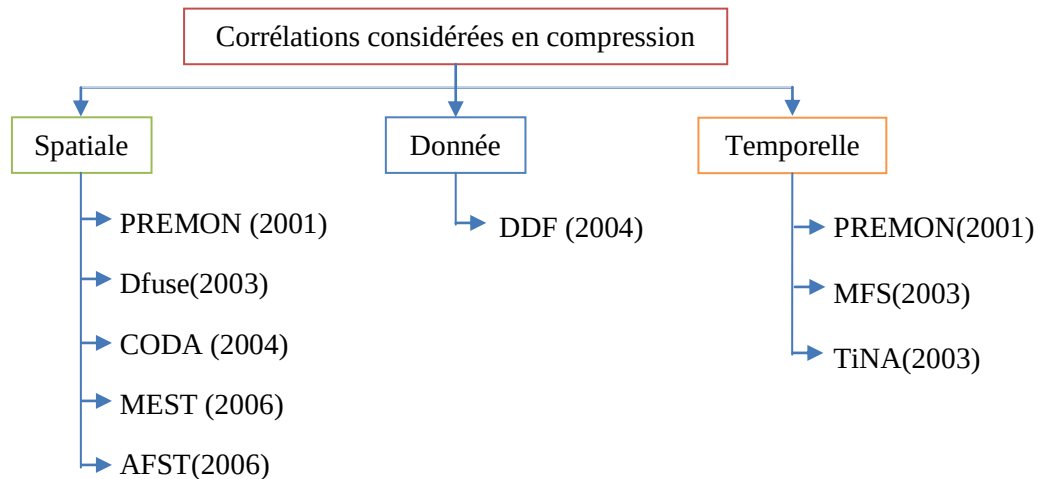


Figure 5.2 : Classification de quelques corrélations considérées dans les mécanismes de compression de données [112]

Les méthodes de corrélation classées comme spatiales suggèrent que les nœuds capteurs voisins (les plus proches de l'objectif) collectent généralement des données similaires. À cet effet il est possible d'appliquer la corrélation spatiale pour compresser les données collectées analogues.

Contrairement de la corrélation spatiale, la corrélation de données utilise la théorie de l'information pour coder des données à forte probabilité (disposent une haute similarité) et génère un court flux de données en sortie.

En plus de la similarité des données collectées, la contrainte de temps qui entre au moment de l'échantillonnage (la capture), transforme la dernière corrélation (corrélation de données) en corrélation temporelle en collectant les données détectées similaires sur une courte période.

Plusieurs recherches ont étudié l'impact des méthodes vues (corrélation, agrégation et compression) sur le routage avec compression et sur d'autres performances des RCSF [9, 10].

Nous présentons dans ce qui suit les méthodes que nous avons adoptées dans nos travaux afin d'intégrer des solutions de compression qui entrent dans le cadre de l'amélioration de l'efficacité énergétique, en exploitant différentes notions de corrélation spatio-temporelle, la tolérance et l'approximation lors de la conception de la solution finale.

V.5. Solutions adoptées pour la compression de données dans M2FS

Dans ce qui suit, nous utilisons quelques techniques simples de compression (TiNA et Approximation polynômiale) souvent appelées pour réduire le volume de données tout en maintenant la signification des données d'origine. Les enjeux sur l'utilisation de la technologie des

RCSF pour l'implémentation de ces techniques (compression) présentent un défi primordial et clé dans la préservation des ressources énergétiques des nœuds capteurs. Or, plus la technique de compression choisie assurera un taux de compression important tout en maintenant bien sûr une cohérence entre les données capturées - d'origine - et les données restituées, plus elle sera favorisée pour être implémentée sur le banc d'essai afin d'étendre la durée de vie du réseau et d'économiser les niveaux énergétiques des petits dispositifs (les nœuds capteurs).

V.5.1. TiNA

(**TiNA** : *Temporal Coherency Aware In-Network Aggregation*) [126], est un système permettant de réduire la consommation d'énergie dans les réseaux de capteurs en exploitant la tolérance temporelle. TiNA permet de réduire la quantité d'informations transmise par les nœuds individuels, et améliore la qualité des données lorsque toutes les mesures ne peuvent pas être propagées sur le réseau dans un délai donné.

L'idée de base de TiNA consiste à envoyer une lecture via l'arbre de routage vers la station de base seulement si elle est différente de la précédente lecture par un pourcentage de cohérence temporelle TCT (*Tolerance Coherency Temporal*). Si un nœud collecte une nouvelle lecture V_{new} , elle sera comparée avec l'ancienne lecture V_{old} . La nouvelle lecture est transmise si la relation suivante est vérifiée :

$$\frac{|V_{new}-V_{old}|}{V_{old}} > TCT \quad (5.2)$$

La valeur de TCT spécifie le degré de tolérance donné par l'utilisateur lié aux modifications de la valeur des lectures collectées par le nœud capteur. Par exemple : si l'utilisateur affecte $TCT = 10\%$, le réseau de capteurs ne signalera que les lectures qui diffèrent des lectures précédemment signalées de plus de 10% . La valeur de TCT est comprise entre 0 - ce qui indique de signaler les lectures en cas de changement-, à un nombre positif.

Un nœud capteur TiNA doit conserver des informations supplémentaires pour pouvoir utiliser les tolérances à la cohérence temporelle. Les informations conservées dépendent de la position du capteur dans l'arbre de routage (feuille ou nœud interne). Les nœuds feuille ne conservent que la dernière lecture rapportée (définie comme dernière lecture envoyée avec succès par un nœud à son parent). Les nœuds internes, en plus de la dernière lecture rapportée pour ce nœud, conservent les dernières données rapportées reçues de chaque enfant. Ces données peuvent être soit une simple lecture rapportée par un nœud feuille, soit un résultat partiel rapporté par un nœud interne [126].

Un problème particulier qui se pose dans le contexte de TiNA consiste à savoir comment gérer le cas de requêtes avec une clause WHERE. Dans le mécanisme d'agrégation en réseau, si les données détectées ne satisfont pas le prédicat WHERE, le capteur interrompt la transmission. Le problème

est de savoir comment le parent pourrait déterminer la différence entre une valeur qui ne satisfait pas le prédicat et une valeur comprise dans la plage spécifiée par le Tct. Pour cela, le nœud enfant doit envoyer un court message d'invalidation à son parent lorsque la valeur détectée ne satisfait pas le prédicat WHERE. Le parent n'utilisera donc pas la lecture stockée pour cet enfant [126].

Un problème relativement similaire est de savoir comment gérer les nœuds capteurs défaillants, dans lesquels un parent ne sera pas en mesure de faire la distinction entre une défaillance de nœud et la suppression Due à l'application de TiNA. Pour résoudre ce problème, un nœud enfant qui supprime la transmission envoie périodiquement un message de pulsation à son parent, même si la valeur détectée est toujours dans la limite Tct. Si le nœud parent n'a pas reçu le message de pulsation pendant un certain temps, il invalide l'entrée de cet enfant spécifique jusqu'à ce qu'il l'entende à nouveau. Cette technique fonctionnera également pour les nœuds capteurs mobiles, où un nœud capteur peut changer d'emplacement dans le réseau et ainsi changer de parent [126].

L'implémentation de TiNA sur un RCSF a été réalisée à l'origine sur une topologie hiérarchique du réseau (on y trouve la racine, les nœuds intermédiaires ou parents et les feuilles).

Dans cette thèse, nous nous limitons à l'implémentation pratique de la méthode TiNA dans un nœud capteur feuille vu que la topologie considérée dans notre cas est plate. La figure 5.3 illustre notre implémentation de TiNA sur un nœud feuille.

Le Sink dans cette méthode, ne dépense aucun effort autre que la régénération de l'ancienne valeur reçue dans le cas où une nouvelle valeur n'est pas transmise au Sink. La figure 5.4 montre le comportement de TiNA au niveau du Sink. En effet, le Sink reçoit les données, il les stocke dans la base de données. En plus, à l'arrivée d'une alarme, il stock celle-ci dans la base de données et informe les services concernées pour intervenir afin de lutter contre l'incendie.

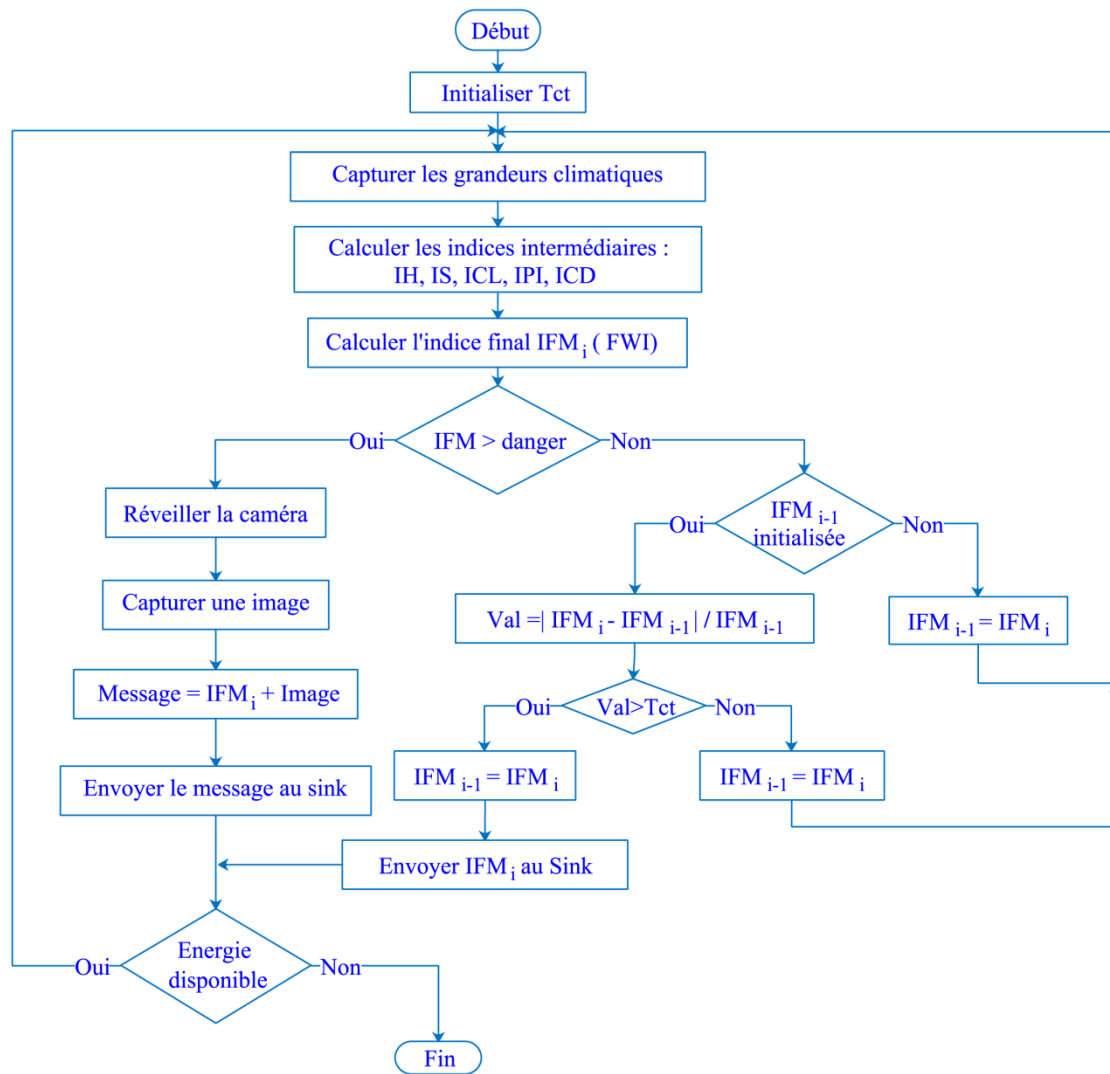


Figure 5.3 : Algorithme de détection et de compression en utilisant TiNA dans un nœud capteur

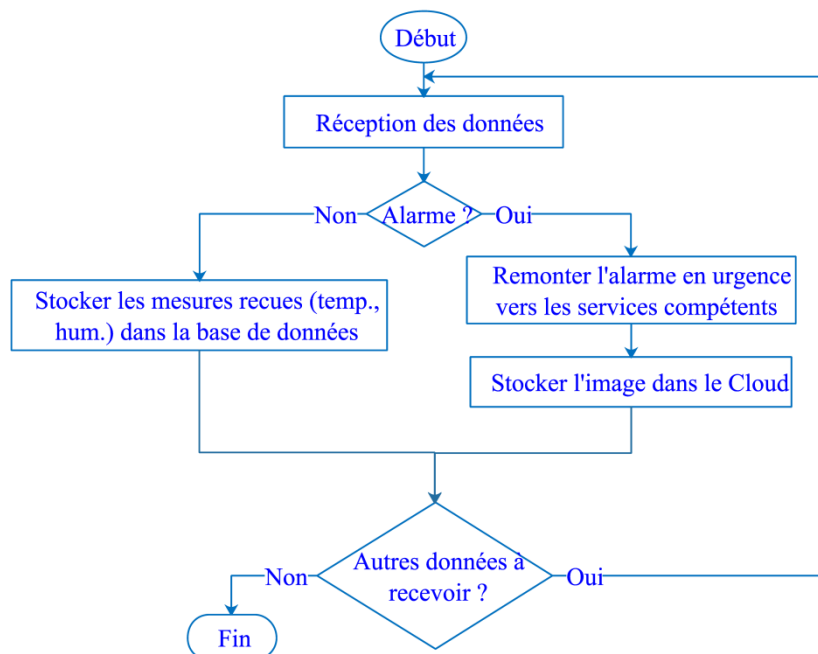


Figure 5.4 : Implémentation de TiNA au niveau du le Sink

V.5.2. L'approximation polynomiale comme technique de compression dans les RCSF

V.5.2.1. Notions de base

Dans les applications de surveillance, l'aspect temporel de l'information est important, il définit implicitement le moment de déclenchement de l'évènement. Ce paramètre fait partie de l'ensemble des informations envoyées à la station de base sous la forme : [temps de la lecture, valeur lue]. Les relevés climatologiques sont un exemple type de ces applications. Dans ce type d'application la redondance de données est acceptée, contrairement aux plusieurs approches de compression et d'agrégation [5].

La cohérence temporelle entre les données calculées issus des différentes observations capturées, et l'approximation (théorème de *Weierstrass*) de ces données forment les principaux axes de la présente étude.

Weierstrass [127, 128] a montré que toute fonction continue définie sur un intervalle fini peut être approchée uniformément par des fonctions polynomiales.

Le théorème de *Weierstrass* suppose pour une fonction f continue dans l'intervalle $[a, b]$ à valeurs réels. Pour tout $\varepsilon > 0$, il existe une fonction polynomiale p à coefficients réels telle que pour tout x dans $[a, b]$, $|f(x) - p(x)| \leq \varepsilon$.

La démonstration de ce théorème repose sur une généralisation de théorème de Bernstein. L'approximation de Bernstein est une méthode d'approximation polynomiale permettant d'approcher uniformément une fonction f continue sur l'intervalle $[0, 1]$ par une famille de polynômes convergents, appelés polynômes de Bernstein [129].

En appliquant le théorème de *Weierstrass*, nous essayons de trouver une représentation polynomiale simple aux données collectées temporellement cohérentes. Cela permet donc d'approximer les lectures collectées par un nœud pendant un intervalle de temps (fixe ou variable) et de construire un polynôme porteur de données, de degré inférieur au nombre d'échantillons capturés. Et au lieu de transférer les n paires (temps, mesure) de données collectées dans chaque paquet de données d'une manière individuelle, le nœud capteur génère le polynôme suivant:

$$P(x) = a_n x^n + a_{n-1} x^{n-1} + \dots + a_1 x + a_0 \quad (5.3)$$

Et seulement les coefficients a_i (obtenus par la méthode des moindres carrés [130] comme application numérique du théorème de *Weierstrass*) du polynôme $P(x)$ seront envoyés à la station de base.

Contrairement à l'interpolation qui n'a aucun impact sur la compression, l'approximation est plus flexible au sens où l'utilisateur peut choisir le degré de polynôme qu'il voulait obtenir, comme ça, si l'erreur d'approximation est tolérable, il gagne en termes de volume de données à transmettre.

La figure 5.5 montre un aperçu de ce mode de compression.

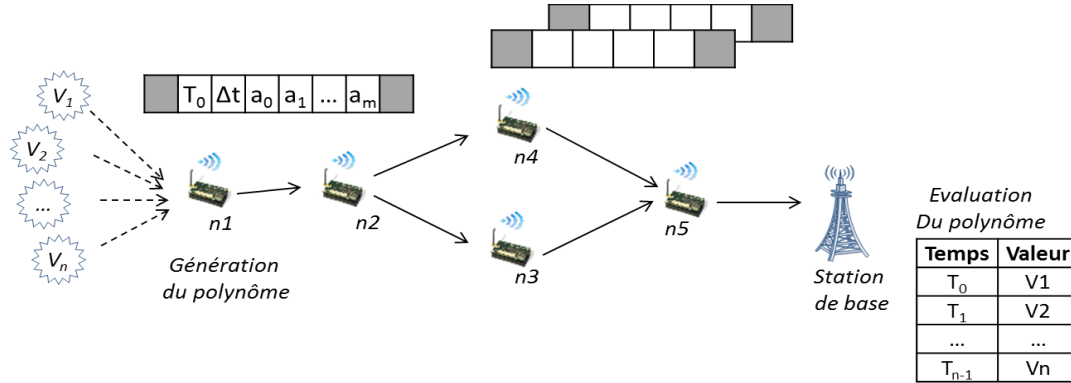


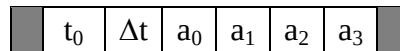
Figure 5.5 : Aperçu sur le mode de compression par approximation

En se basant sur la figure 5.5, et comme exemple explicatif, nous considérons le nœud $n1$ a collecté 20 valeurs $v_1 \dots v_{20}$ ($n=20$). En utilisant ce mode de compression, nous produisons un polynôme de degré $m < n-1$, et seulement les m coefficients seront réunis dans un paquet et envoyés à la station de base, et bien sûr on l'informe (la station de base) du temps de premier échantillon, qui est collecté à t_0 , et de délai pris entre chaque couple de collectes successives Δt .

Pour le degré $m=4$, notre polynôme aura la forme :

$$P(x) = a_3 \cdot x^3 + a_2 \cdot x^2 + a_1 \cdot x^1 + a_0.$$

Et le paquet constitué aura la forme suivante :



De son côté, la station de base reconstitue les paires de données (t_i, v_i) en recevant le paquet. Pour que cette information soit correcte et précise, l'application définit l'erreur d'approximation tolérée, qui garantit la qualité de l'information à la suite de l'évaluation.

En général, les applications de surveillance peuvent tolérer des observations approximatives sur le phénomène observé. Face à cette incertitude, il devient nécessaire de tolérer à divers niveaux de précision ces observations pour différents scénarios. Dans notre cas, le niveau de précision est contrôlé par l'observation d'erreur générée par la méthode utilisée lors de l'approximation (moindres carrés). L'erreur d'approximation E_i est définie comme la différence entre la valeur collectée par le nœud capteur et la valeur restituée par la station de base selon l'équation suivante :

$$E_i = X_i - P(t_i) \quad (5.4)$$

Où X_i est la valeur collectée par le nœud, $X'_i = P(t_i)$ est la même valeur restituée par la station de base.

Cette erreur définit une spécialisation de l'idée de la méthode des moindres carrés qui est basée sur la minimisation de l'erreur suivante:

$$\min \sum_{i=1}^m (x_i - p(t_i))^2 \quad (5.4)$$

Il est impossible de réduire à zéro cette erreur, mais il est toutefois possible de qualifier l'ampleur de l'erreur par rapport à l'erreur moyenne $\bar{E}$ d'une séquence afin de la contrôler. Il s'agit donc de calculer une variation de l'erreur.

La variation d'erreur [4] mesure le caractère de dispersion des erreurs E_i autour de la moyenne $\bar{E}$. Nous calculons la variation de l'erreur d'approximation (variat (E)) de chaque séquence de données par :

$$\text{variat}(E) = \frac{n \cdot \sum_{i=1}^n (E_i)^2 - (\sum_{i=1}^n E_i)^2}{n(n-1)} \quad (5.5)$$

Où n est la taille de la séquence de données.

Pour une application à caractère qualitative –comme la nôtre–, la tolérance de la variation de l'erreur est définie comme étant le seuil pour qu'une information sur un phénomène soit utile et interprétable. Autrement dit, elle suit la condition suivante :

$$\text{variat}(E_i) \leq \varepsilon \quad (5.6)$$

La valeur de ε est définie soit par les experts du domaine d'étude, soit suite à des analyses statistiques sur les phénomènes à observer.

Nous nous intéressons dans l'étude de l'approximation à deux méthodes. La première est une heuristique qui effectue l'approximation sur des fenêtres à intervalle de temps fixe. La deuxième effectue l'opération d'approximation sur des fenêtres à temps variables.

V.5.2.2. Algorithmes de compression

Nous rappelons ici que la technique de compression nous servira à améliorer notre système M2FS au niveau de la consommation énergétique et la latence. Ainsi, dans cette partie, nous présentons les algorithmes de compression adoptés lors de nos expériences. Dans un deuxième temps, nous modélisons par quelques diagrammes UML ces approches de compression basées sur l'approximation polynomiales. Les deux algorithmes adoptés sont :

- L'approximation polynomiale avec fenêtre fixe (APFF).
- L'approximation polynomiale avec fenêtre variable (APFV).

A. L'algorithme APFF :

Dans ce cas, le nombre d'échantillons et la grandeur de données à approximer sont connus au préalable. Après avoir collecté les n échantillons, l'algorithme calcule par la méthode des moindres carrée le polynôme à moindre degré représentatif de la séquence collectée (les n échantillons). Les coefficients du polynôme généré doivent être calculés suivant une variation d'erreur (ϵ), fixée par l'utilisateur. Le choix de cette valeur (la variation de l'erreur) peut fournir un support de décision pour le nombre de cycles de collection pour un phénomène donné. L'organigramme présenté dans la figure 5.6 montre le fonctionnement de notre algorithme de détection (M2FS) en tenant compte de la compression de données scalaires en utilisant l'algorithme APFF.

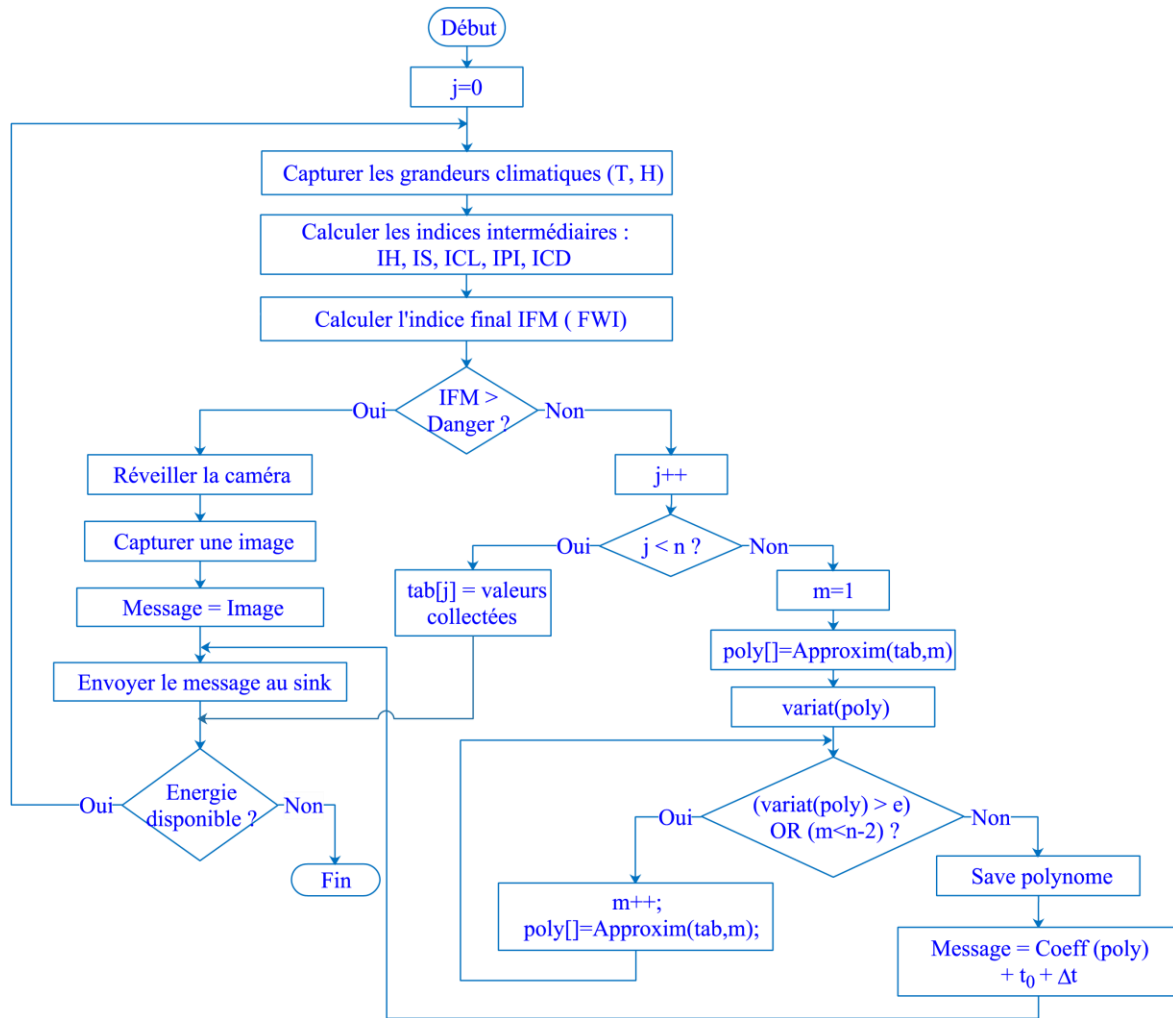


Figure 5.6 : Algorithme de détection et de compression en utilisant APFF d'un nœud

D'autre part, la tâche assurée par le Sink consiste à réceptionner les données qui sont évidemment les coefficients, avec ou sans l'alarme (selon qu'il y a présence du feu ou non), transmis par les nœuds capteurs sources, sans négliger bien sûr la tâche de décompression des données (restitution des données à partir des coefficients des polynômes). La figure 5.7 montre ce comportement du Sink.

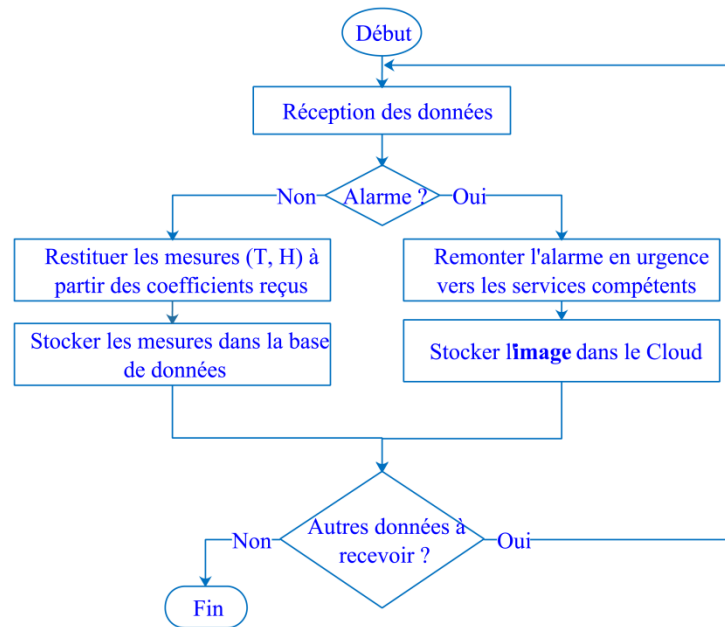


Figure 5.7 : Algorithme de réception de données par le Sink

B. L'algorithme APFV

Dans cet algorithme, le nombre d'échantillons est connu mais la dimension de la fenêtre de temps est variable, elle est identifiée au fur et à mesure jusqu'à obtenir un polynôme optimale (qui correspond à une bonne compression). Au départ, un polynôme (de degré moindre et qui a une variation d'erreur inférieure fixée par l'application) est généré à partir d'un ensemble de données capturées. À chaque nouvelle lecture, l'algorithme ajoute celle-ci à l'ensemble des valeurs constituant la fenêtre de départ, et approxime à nouveau un polynôme. Si le degré du nouveau polynôme augmente (par conséquent le nombre de coefficients augmente aussi), alors la nouvelle valeur ajoutée identifiera une nouvelle fenêtre, et le dernier polynôme sera retenu. Par conséquent, les coefficients de ce dernier polynôme seront communiqués à la station de base via le réseau. Ce processus se répète tant que le nombre de cycles reste inférieur au nombre total de valeurs à approximer. Les instructions qui implémentent cet algorithme sont présentées dans la figure 5.8

Algorithme APFV

```

//Dim : nombre de lectures par séquence -Variable-
//Tab : lectures collectées
//Poly : Coefficients du polynôme
//S : objet capteur
//VarErr : évaluation du polynôme et calcul de la variation de l'erreur
//FenetreMin : taille minimale de la fenêtre de temps
//FenetreMax : taille maximale de la fenêtre de temps
//AncienDegré : degré du la dernière approximation
Début
Dim = FenetreMin; //Taille Initiale de la fenêtre
//Phase de collection de la fenêtre initiale
Pour i = 0 à Dim faire
    Tab [i] ← S.Collecte ();
    
```

```

Si (Tab[i] > seuil de danger) alors
    Réveiller la caméra ;
    Capturer une image ;
    Envoyer l'alarme au Sink ; //alarme=image+ tableau de données scalaires
finsi
Finpour
//Phase d'approximation
AncienDegré=Dim+1; //Initialisation du degré
//Limiter l'algorithme par un nombre max de lectures
Tant que (Dim<=FenetreMax) faire
    //Approximer la nouvelle fenêtre
    n=1;
    VarErr (Poly)=  $\epsilon$ 

    Tant que ((VarErr(Poly) >  $\epsilon$ ) ou (n = Dim)) faire
        Poly []= Approximer (Tab, n); // approximer ce poly avec le degree n
        n = n+1;
    Fintantque
    //Tester si l'ancien Polynôme est extensible
    Si (n <= AncienDegré) Alors
        //Une nouvelle valeur est ajoutée et on garde l'ancien degré
        Dim=Dim+1;
        Tab [Dim] (S.Collecte ());
        AncienDegré=n;
        //La nouvelle lecture n'est pas considérée, la boucle est interrompue
    Sinon
        Sauvegarder Poly [];
        Envoyer le poly[] au Sink ;
        Retour;
    Finsi
Fintantque
Fin

```

Figure 5.8 : Algorithme de compression polynômiale avec fenêtre variable

V.5.2.3. Modélisation de la technique d'approximation

Nous présentons dans cette section la conception de notre solution d'optimisation du système M2FS basée sur l'approximation polynômiale indépendamment des outils d'implémentation. Nous utilisons pour cette fin, deux diagrammes UML : un diagramme de séquence du système M2FS en intégrant la solution de compression des données scalaires au niveau de la source ; et un diagramme de cas d'utilisation qui tient compte de cette optimisation.

Le diagramme de cas d'utilisation (*use case*) permet de présenter les différentes interactions entre les acteurs et le système.

Ce diagramme est illustré dans la figure 5.9 est conçu afin d'examiner les différentes tâches effectuées par les différents acteurs du système (utilisateur, Sink, nœud relais, Cloud et nœud capteur). Une étape de mise à jour des tâches de chaque acteur est souvent établie après la conception de ce diagramme. Les différentes interactions représentées par les acteurs avec

l'environnement d'évaluation et de test montrent les différentes utilisations de notre environnement dans le contexte de la compression des données scalaires.

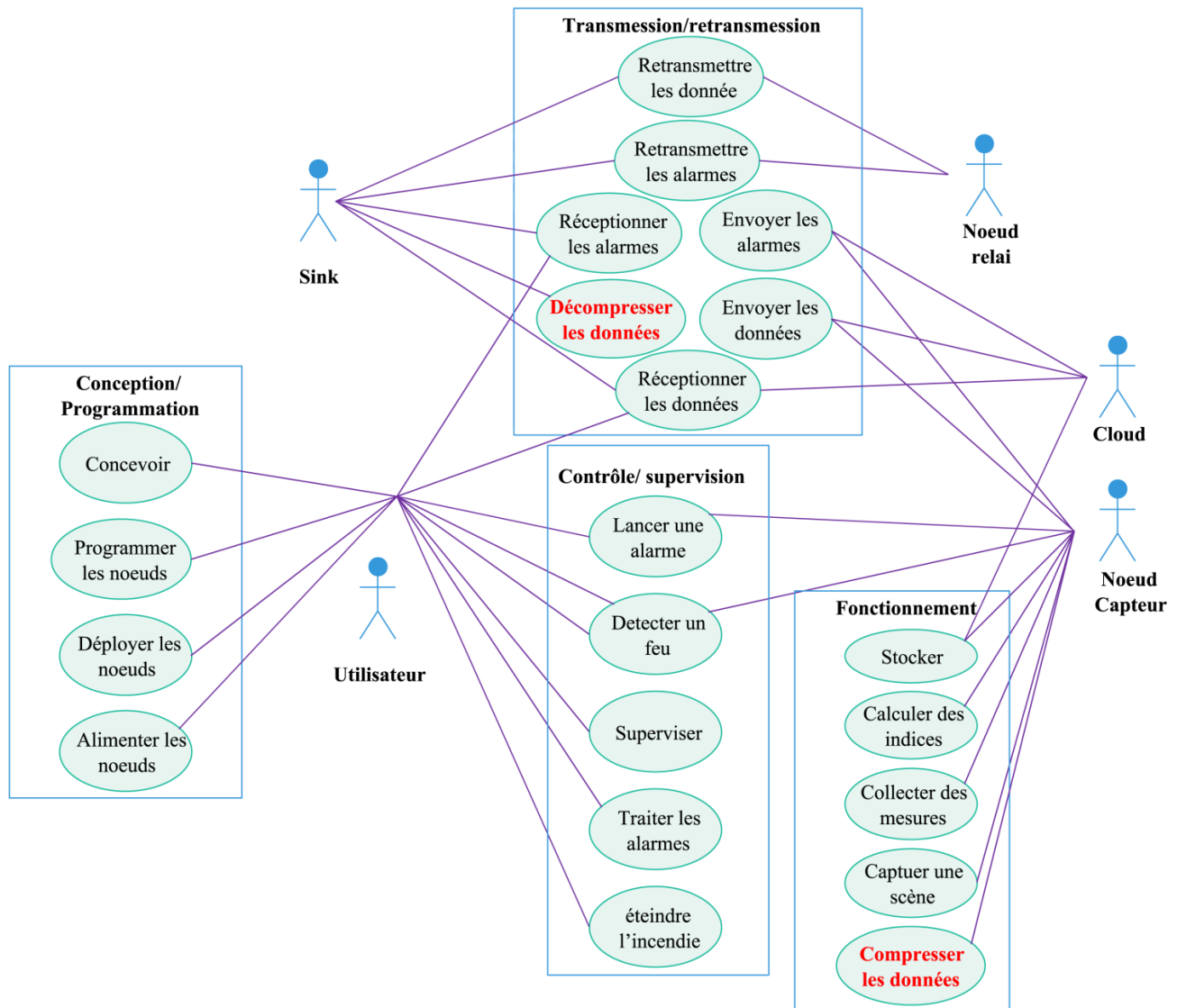


Figure 5.9 : Diagramme de cas d'utilisation de M2FS en tenant compte de la compression

Le diagramme de séquence, illustré par la figure 5.10, montre les échanges de messages entre les différents acteurs du système, suivant une vue chronologique de leur apparition. Les tâches assurées par les acteurs de ce diagramme sont multiples, à savoir :

- L'utilisateur (poste de traitement/décision) : permet (1) d'initialiser la station de base, autrement dit, déployer le réseau et le mettre en service, (2 et 3) programmer les nœuds du réseau avec l'algorithme présenté dans la figure 5.6 (cas de compression avec APFF).
- Le nœud capteur : permet (4 et 4') de collecter les données environnementales et exécute l'algorithme de compression programmé. Il envoie (5 et 5') les coefficients du polynôme généré (et l'alarme si elle est déclenchée en cas d'évènement) vers les nœuds relais qui se trouvent à proximité.

- Le nœud relais : retransmet (6 et 6') les valeurs (coefficients des polynômes avec ou sans l'alarme) reçues du nœud source vers la station de base.
- La station de base : permet (7 et 7') de décompresser les données reçues (restitution des valeurs des grandeurs physiques), et les stocker en vue de les partager avec l'utilisateur final.
- L'utilisateur : permet (8 et 8') de traiter toute donnée stockée et partagée par le réseau.

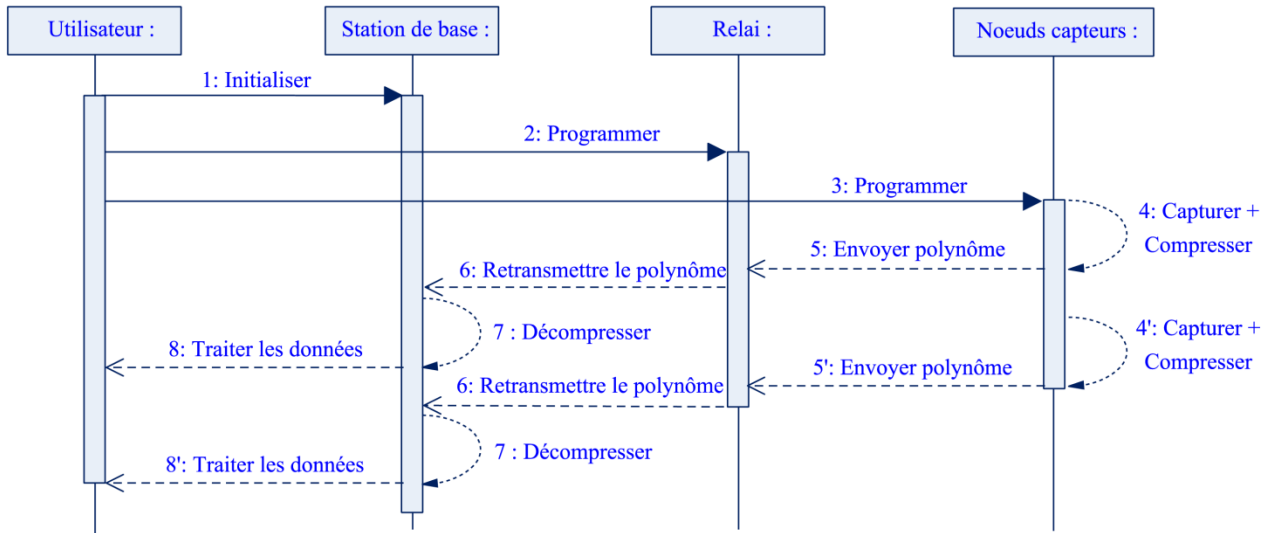


Figure 5.10 : Diagramme de séquence de M2FS en tenant compte de la compression de données

V.6. Résultats et discussion

Dans cette section, nous évaluons les performances des algorithmes de compression intégrés dans le système de détection M2FS conçu, implémenté et testé expérimentalement dans le chapitre 4. Les expériences sont effectuées en utilisant même banc d'essai présenté et illustré dans figure 4.17 (chapitre 4), constitué d'un nœud capteur de type Arduino Due (qui collecte les grandeurs physiques –par le capteur DHT22/DHT11 –, calcule l'indice de danger IFM, détecte la présence d'un incendie, le confirme avec une prise d'image– par la caméra OV2640 attachée avec le nœud Arduino Due via le Shield Arducam – et communique l'alarme à la station de base – par le module Xbee attaché à la carte via le Shield Xbee –), et d'une station de base puissante cette fois-ci pour pouvoir exécuter des tâches complexes (la décompression des données). Nous avons utilisé pour cela des ordinateurs (PC de bureau ou portable) dont les caractéristiques sont donnés dans le tableau 4.6 du chapitre 4.

Les performances à évaluer sont :

- le temps nécessaire pris par le nœud capteur afin de détecter un événement et le communiquer à la station de base,
- le taux de compression des données scalaires,
- et enfin, la quantité d'énergie consommée par le nœud capteur en tenant compte de la compression.

Le schéma de compression employé s'applique uniquement sur les données générées par le nœud capteur, du moment qu'il collecte le plus souvent des données scalaires que les données multimédias (images générées seulement en cas d'évènement) durant le processus de surveillance (pendant une année par exemple). Aussi, une alerte multimédia déclenchée par le nœud capteur suite à un évènement est plus prioritaire et peut ainsi interrompre le mécanisme de compression en cours et oblige par conséquent le système de communiquer l'alerte à la station de base.

Par ailleurs, nous avons également appliqué le mécanisme de compression multimédia pris par défaut – JPEG – et offert par le circuit intégré du Shield Arducam, afin de réduire la taille de l'image capturée et de gagner par conséquent au niveau du temps de transmission et de l'énergie dissipée.

V.6.1. Contrainte temporelle

Le tableau 5.1 présente le temps nécessaire (en milliseconde) pris par le nœud capteur utilisé (Arduino Due) afin d'exécuter des tâches programmées. Ces tâches représentent les actions suivantes : capturer les grandeurs physiques (température et humidité), calculer l'indice IFM, appliquer l'algorithme de compression approprié (TiNA, APFF) et communiquer les valeurs capturées et compressées (selon le mode de compression, à savoir : TiNA, APFF, sans compression) au Sink.

Nombre d'échantillons	Temps de capture	Temps de calcul de l'indice IFM	Temps de compression		Temps de transmission			Temps Total utilisé		
			TiNA	APFF	TiNA	APFF	Sans compres	TiNA	APFF	Sans compres
10	813	10	2	216	3,33	4,81	11,11	828,33	1043,81	834,11
20	1359	20	5	1098	7,78	5,41	22,22	1391,78	2482,41	1401,22
30	2168	30	7	3007	12,22	7,44	33,33	2217,22	5212,44	2231,33
43	2712	43	10	7484	20,00	7,54	47,78	2785,00	10246,54	2802,78
50	3525	50	12	11004	22,22	7,78	55,56	3609,22	14586,78	3630,56
60	4066	60	14	17676	26,11	7,00	66,67	4166,11	21809,00	4192,67
70	4883	70	16	24791	31,67	6,67	77,78	5000,67	29750,67	5030,78

Tableau 5.1 : Temps de capture, de calcul, de compression et de transmission des données scalaires

Nous avons utilisées les données (température et humidité) du dataset de station météo à Florida en USA (Long Key C-MAN Weather Station) [131] pour élaborer les résultats du tableau 5.1. Nous exploitons dans nos calculs que les 360 premières mesures (suivant les ressources de calcul du nœud utilisé – Arduino Due –) parmi l'ensemble des données composé de 19434 mesures climatiques relevés sur deux ans, à une heure d'écart depuis juillet 1998 jusqu'à septembre 2001.

Les deux premières colonnes représentent respectivement le temps nécessaire pris par le nœud afin de capturer les grandeurs physiques (température et humidité), et celui du calcul de l'indice IFM. Ces deux temps sont déjà présentés dans le tableau 4.7 du chapitre précédent (en choisissant un

nœud de type Arduino Due et le capteur DHT22) avec une différence remarquable dans le temps de capture. Nous remarquons que ce dernier augmente avec l'augmentation du nombre d'échantillons, mais ne se multiplie pas (c.à.d, ne suit pas l'allure du nombre d'échantillons), et cela est dû au mode de fonctionnement du capteur DHT22.

DHT22 passe de l'état veille à l'état d'activité dès qu'il reçoit une demande de capture. À la fin de la collecte, il bascule vers l'état veille s'il ne reçoit plus de demande de capture [132]. La formule (5.7) donne le temps pris par le capteur lors d'une demande de collecte :

$$T_{ca} = T_r + T_{co} + T_{rep} + T_v \quad (5.7)$$

Avec : T_{ca} est le temps de capture, T_r est le temps de réveil, T_{co} est le temps de collecte, T_{rep} est le temps de réponse du capteur et enfin T_v est le temps de mise en veille.

En cas de collecte de plusieurs valeurs – consécutives –, le temps nécessaire pour collecter n valeurs est donné par :

$$T_{ca} = T_r + n * (T_{co} + T_{rep}) + T_v \quad (5.8)$$

Dans l'équation 5.8, la non multiplication des temps T_r et T_v par le nombre de mesurer n , explique une évolution lente du temps de capture par rapport à l'évolution du nombre d'échantillons. Puisque le temps de basculement entre les modes T_r et T_v est plus important que celui de la collecte et de réponse (T_{co} , T_{rep}).

La troisième colonne montre le temps de compression pris par les méthodes TiNA et APFF. Nous remarquons que le temps pris par TiNA pour compresser l'ensemble des mesures est beaucoup plus petit que celui pris par APFF. Cela revient à la complexité de l'algorithme de compression utilisé. TiNA permet de vérifier chaque pair de valeurs, et passe à la compression de ces valeurs dès que la Tct est petite.

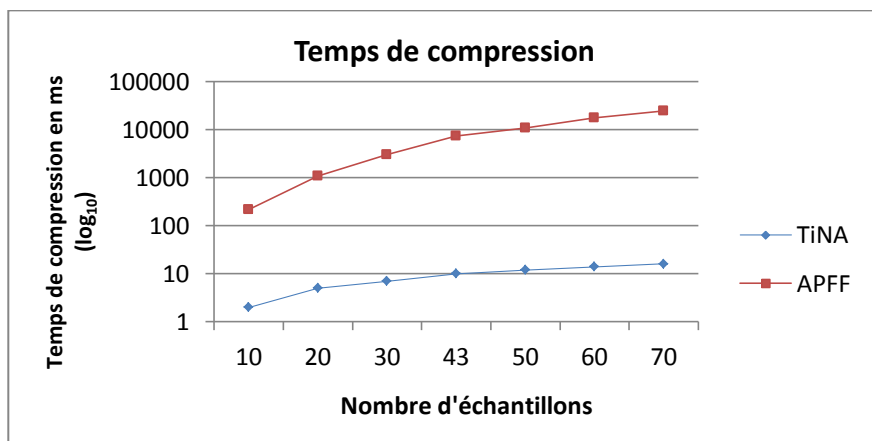


Figure 5.11 : Comparaison entre les délais de compression des méthodes TiNA et APFF

Par contre, APFF permet d'appliquer sur l'ensemble de mesures une méthode d'approximation, et calcule des coefficients obtenus par compression qui nécessitent pour chaque ensemble de mesures, un temps qui est illustré dans la courbe de la figure 5.11.

Afin de présenter efficacement les valeurs du temps de compression des deux méthodes APFF et TiNA, nous avons appliqué sur l'ensemble des données (les temps) le logarithme pour réduire l'échelle de la courbe. Nous constatons que, plus le nombre de données augmente plus la méthode consomme plus du temps (cycles processeur), pour traiter les données (compresser), cela est expliqué par la complexité de la méthode.

La figure 5.12 montre une comparaison entre le temps de transmission des données par les méthodes de compression – APFF et TiNA – et la méthode de collecte sans compression. Ce temps est représenté aussi dans le tableau 5.1 dans la quatrième colonne. Il est calculé en se basant sur la quantité de données envoyés et de la vitesse de communication. Si nous calculons le temps de transmission des coefficients générés par la méthode de compression APFF, appliqué à un ensemble de données collectées (cas de dix valeurs par exemple), on aura :

Premièrement, sur un ensemble de données de 360 valeurs (température et humidité) fragmentées en 30 séquences de 10 valeurs, l'algorithme APFF a généré 260 $(93+107+60)^5$ coefficients⁶ (somme des coefficients des polynômes générés), à partir des données de départ (température, humidité). Ces coefficients de type double précision⁷ (dans le programme Arduino) vont être codés sur 8 octets chacun. Le temps de transmission T_{tr} est calculé par la formule 5.9 :

$$T_{tr} = \frac{D}{V} \quad (5.9)$$

Avec D désigne la charge de données à envoyer (en bits), V désigne la vitesse de communication (115200 b/s pour notre cas).

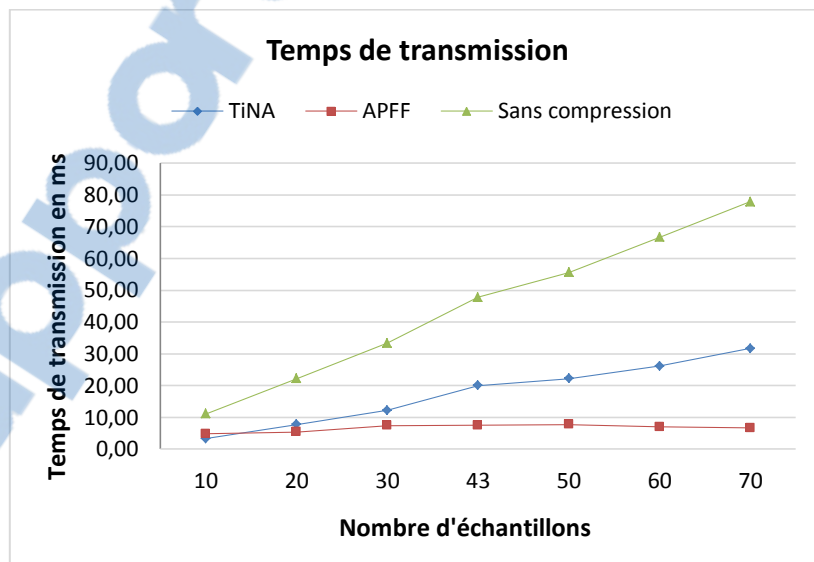


Figure 5.12 : Comparaison entre le temps de transmission des trois méthodes

⁵93 coefficients des polynômes de température, 107 coefficients pour l'humidité et 2×30 pour les instants de départ (t_0) et la fréquence d'échantillonnage Δt

⁶En choisissant une erreur de tolérance = 0.7

⁷Exigé dans les calculs

Nous constatons que l'algorithme APFF présente pratiquement le meilleur temps de transmission pour une compression de données supérieur ou égale à 20. La méthode TiNA présente aussi un temps de transmission meilleur par rapport à la transmission de la totalité des données (sans compression). L'écart entre les délais de transmission entre TiNA et APFF est proportionnel selon la valeur de l'erreur choisie. Une augmentation de cette valeur permet d'augmenter le nombre de valeurs à envoyer dans les deux algorithmes.

La dernière colonne du tableau 5.1 affiche l'ensemble des temps discutés plus haut (celui de collecte, de calcul, de compression et de transmission). En général, le temps pris par la méthode TiNA est meilleur par rapport aux deux autres à cause de la simple sélection de données à envoyer, et qui n'influe pas sur le temps de compression des données sélectionnées.

Remarque : les temps présentés dans le tableau 5.1 des trois méthodes a été évalué en collectant à chaque fois 10, 20,... 70 valeurs. Suivant le nœud utilisé (Arduino Due), et il n'est pas possible d'appliquer notre algorithme (toutes les tâches) sur un ensemble de données qui dépasse 70 mesures. Cela dépend des limites de calcul et de stockage dont dispose ce nœud. Cependant, les valeurs montrées par le tableau 5.1 peuvent être améliorées en modifiant la plateforme qui implémente notre algorithme par une plateforme plus puissantes (en traitement et stockage).

V.6.2. Taux de compression

V.6.2.1. APFF

Une bonne compression revient à maximiser l'écart entre le nombre de messages envoyés sans traitement (compression) et ceux générés suite à un traitement (la compression). En d'autres termes, pour notre cas, la comparaison se fera entre le nombre de mesures collectées et le nombre de coefficients envoyés vers la station de base. Cela est traduit mathématiquement par le calcul du gain de compression donnée par l'équation 5.10:

$$\text{GainCompression} = \frac{\text{Nbre_Mesure} - \text{Nbre_Coefficients_transmissibles}}{\text{Nbre_Mesure}} \quad (5.10)$$

Nous visons dans un premier temps la précision de données générées par la méthode APFF. Pour se faire, nous calculons pour chaque séquence de données le nombre de coefficients du polynôme généré par l'approximation vérifiant l'erreur tolérable, en variant à chaque fois la taille de la fenêtre. La figure 5.13 présente des courbes de différentes tailles de fenêtre (6, 12, 18 et 24) et qui montrent leur impact sur l'efficacité de la compression selon la variation de l'erreur. Il apparaît clairement que les courbes (de la figure 5.13 (a)) sont relativement stables et suivent la même allure. Cette stabilité s'explique par la nature du domaine des échantillons pris par le nœud capteur – la température –. En d'autres termes, l'écart entre les valeurs est proportionnellement homogène (pas de piques entre les valeurs). Il est important de noter que ces expériences sont faites en appliquant

l'algorithme APFF sur un ensemble de 360 valeurs⁸ de la base de données *Long Key C-MAN Weather Station*.

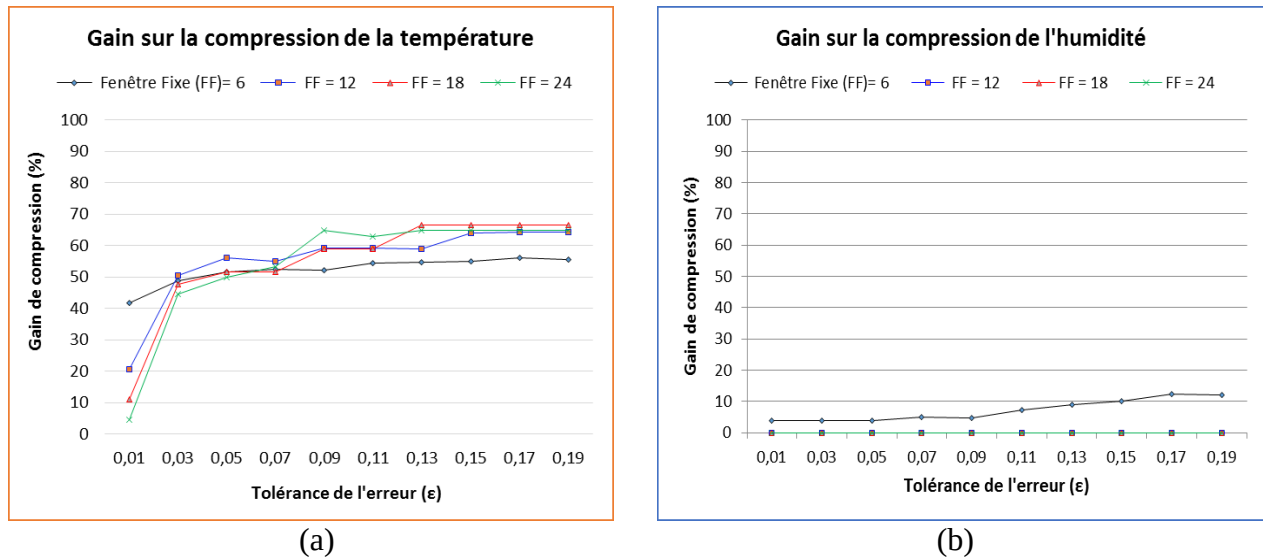


Figure 5.13 : Impact de la taille d'une séquence sur l'efficacité de l'approximation de l'APFF

Par contre, les fluctuations entre les valeurs de l'humidité (écart non homogène) de la même base de données (même taille aussi), montrent une instabilité sur le gain de compression générée par l'algorithme de compression APFF comme le montre la figure 5.13 (b). Autrement dit, la valeur maximale (0 %) affectée pour le gain de compression des fenêtres fixes (FF) 12,18 et 24, montrent que l'algorithme APFF n'a pas trouvé aucun polynôme de degré inférieur à $FF-2^9$ qui a une variation d'erreur inférieure à la valeur fixée par l'utilisateur. Dans ce cas, notre algorithme se comporte comme le cas « sans compression » et envoie la totalité des données collectées vers la station de base.

Nous constatons aussi dans les courbes de compression de la température (figure 5.13 (a)), que le taux de compression (ou 100% - gain de compression) est faible lorsqu'on augmente la précision de la méthode (tolérance de l'erreur ϵ est petit). Cela est logique puisque à chaque fois que l'algorithme trouve une variation de l'erreur qui dépasse la marge d'erreur spécifiée (ϵ), il augmente le degré du polynôme (générer un coefficient de plus) afin d'améliorer la précision et de diminuer par conséquent la variation de l'erreur.

D'autre part, on remarque que plus la taille de la fenêtre augmente (le maximum est 18) plus le taux de compression diminue, par conséquent le gain de la compression augmente. Donc, il est préférable d'étaler (jusqu'au 18) la taille de la fenêtre afin de minimiser le nombre de coefficients à envoyer, et de gagner en temps de transmission et aussi en énergie par la suite.

⁸Suivant les capacités du nœud capteurs utilisé (Arduino Due)

⁹Deux valeurs qui entrent dans la composition du paquet de données (t_0 et Δt)

V.6.2.2. TiNA

En appliquant le même principe de calcul du gain de compression vu précédemment, le nombre de coefficients transmissibles cette fois-ci concerne la valeur de la température et de l'humidité qui vérifient la relation d'ajustement du Tct (donné dans l'équation 4.1). La courbe illustrée par la figure 5.14 montre l'impact de la variation de la valeur du Tct sur le taux de compression.

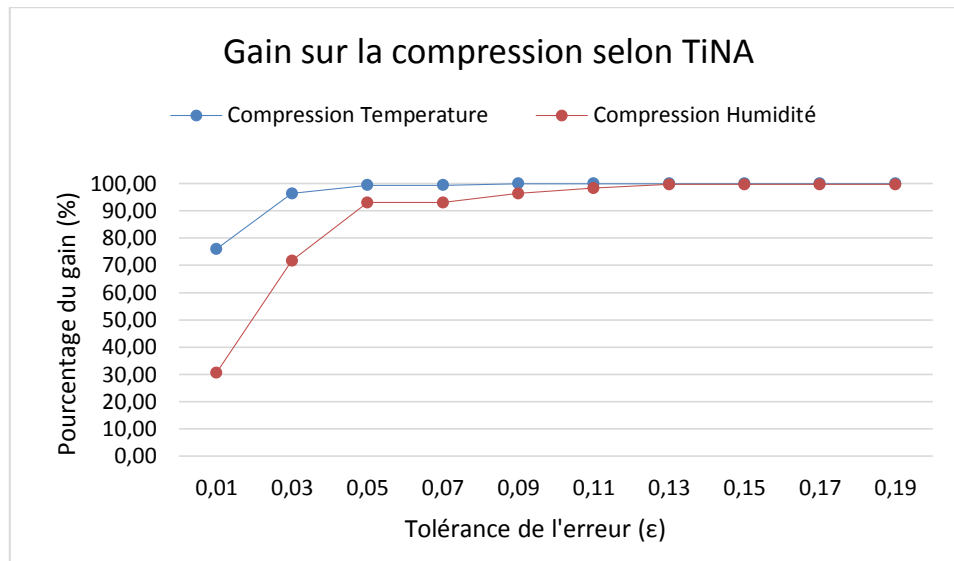


Figure 5.14 : Impacte de la valeur du Tct sur le taux de compression à l'aide de TiNA

Nous constatons que plus la valeur du Tct augmente, plus le gain sur la compression augmente. Autrement dit, si l'écart entre deux mesures consécutives diminue, la nouvelle valeur capturée ne sera pas envoyée, et par conséquent le taux de compression augmente. Cela dépend de la corrélation des valeurs collectées, plus la corrélation est forte entre les paires de valeurs, plus le taux de compression sera important. Cette contrainte est vérifiée par les valeurs de la température visualisées dans la figure 5.14. Par contre, la fluctuation entre les valeurs de l'humidité a influencé le taux de compression, c'est pour cette raison que la courbe de la température montre un gain sur la compression important par rapport à la courbe de l'humidité.

En comparant les courbes des deux méthodes APFF et TiNA présentées respectivement dans les figures 5.12 et 5.13, nous constatons que la méthode TiNA présente en général des gains de compression plus élevés par rapport aux méthodes concurrentes dans le cas de la température pour les erreurs tolérées supérieures à 0,03. De même pour la taille de la fenêtre 6 dans le cas de compression APFF de l'humidité pour les erreurs 0,01 et 0,03. Par contre, TiNA présente un gain remarquable pour les autres erreurs en l'appliquant sur les valeurs de l'humidité.

V.6.2.3. Discussion

C'est vrai que les tâches (complexes) de compression appliquées sur des dispositifs à ressources limitées posent des problèmes importants sur la durée de vie du nœud capteur. D'autre part, elles présentent pour l'ensemble du réseau des avantages encourageants tels que :

- Moins de paquets circulant dans le réseau, et par conséquent un gain sur la consommation d'énergie,
- Consolidation de la vulnérabilité du réseau : les valeurs (coefficients des polynômes) communiquées dans le réseau représentent une sorte de chiffrement,

Seulement, il faut trouver le meilleur compromis entre la complexité de l'algorithme de compression, la consommation d'énergie des nœuds capteurs et la précision des données compressées. Autrement dit, il faut chercher la taille adéquate du polynôme (maximum possible) pour générer un nombre restreint de coefficients, et qui donnent une image cohérente et compatible aux données de départ, et aussi une variation de l'erreur soit minimale possible. Pour satisfaire ces exigences, il fallait lancer une série d'expériences en termes de :

- Changement de taille de fenêtre,
- Variation de l'erreur de calcul (ϵ),
- Consommation d'énergie expérimentalement en utilisant un oscilloscope.

La majorité des expériences illustrées par les courbes montrent que la méthode TiNA est meilleur en compression, cela donne un plus à cette méthode de ce côté. Aussi, dans la contrainte temporelle, TiNA exécute son algorithme de compression dans un temps réduit en se référant à la figure de la comparaison de 5.11 et 5.12.

Pour vérifier l'exactitude entre les valeurs collectées au départ, et les valeurs compressées, nous définissons un paramètre appelé facteur de restitution des données reçu par le Sink.

V.6.2.4. Restitution de données

La restitution des valeurs compressées reçus par le Sink, est une opération qui permet de reconstituer les valeurs de départ (avant la compression) à partir des valeurs compressées reçues. La ressemblance entre la valeur d'origine et celle restituée témoigne la robustesse de la méthode de compression à la source, plus la ressemblance est forte entre chaque paire de valeur (de départ et celle restituée), plus la méthode de compression est précise. Autrement dit, la variation de l'erreur du polynôme respecte celle tolérée par l'utilisateur. La figure 5.15 montre la restitution des valeurs de la température et de l'humidité compressées à la source.

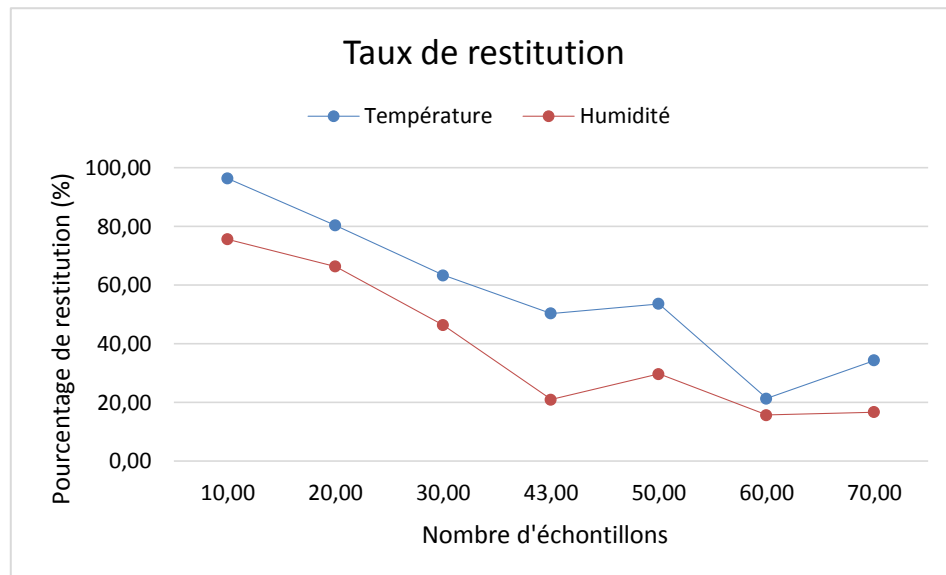


Figure 5.15 : Comparaison entre la restitution de la température et de l'humidité compressées à la source

La forte ressemblance entre les valeurs de la température a donné des polynômes plus précis que ceux de l'humidité. Cela est bien démontré par les courbes de restitution de la température et de l'humidité illustrées dans la figure 5.15. On remarque que plus la taille de la fenêtre augmente, plus le taux de restitution diminue, et ceci est toujours lié à la notion de corrélation entre les valeurs.

V.6.2.5. Compression multimédia

Notre objectif étant de transmettre l'alarme sous forme de flux de données multimédia en cas d'occurrence d'événement (présence du feu). Pour cela il devient nécessaire de compresser ce flux de données pour réduire la consommation d'énergie d'une part, et de gagner du temps pour transférer cette alarme qui est urgente. Pour atteindre ces buts, deux alternatives peuvent être envisagées. La première consiste à compresser le flux de données multimédia à l'aide de modules embarqués dans le banc d'essai, ou bien en mettant en œuvre un algorithme de compression de flux multimédia approprié.

Théoriquement, 921 600 octets (900 Ko) est la taille d'une image ayant une résolution de 640 x 480 pixels, en couleur, non compressée.

Cette taille est obtenue suite au calcul suivant : $921\,600 \text{ octets} = 480 * 640 * 3$, où 480 est le nombre de pixels d'une colonne, 640 est le nombre de pixels d'une ligne, 3 est nombre de couleur d'un pixel qui est codé sur un octet. D'autre part, la taille de l'image capturée par notre banc d'essai (résolution: 640x480 pixels, compression: JPEG) reçue par le Sink occupe 13,08 Ko.

L'énorme écart entre l'image non compressée (900 ko théoriquement calculée) et l'image compressée (13,08 ko), nous encourage à utiliser le circuit dédié (intégré dans le Shield Arducam) afin de compresser l'image capturée avant sa transmission vers le Sink.

Une étude plus approfondie sur les algorithmes de compression multimédia avec des tests expérimentaux permet d'améliorer les paramètres de performance comme le temps de traitement, l'énergie dissipée et par conséquent la durée de vie du réseau, tout en respectant les défis qui entourent le système en tant que qualité d'image et ressources matériels d'un nœud capteur. Ces exigences ne sont pas abordées dans notre thèse, mais elles peuvent faire l'objet de nos futurs travaux.

V.6.3. Contrainte énergétique

Notre objectif à propos du calcul de la consommation énergétique dans cette contribution est triple :

- i) Mesurer expérimentalement la quantité d'énergie consommée par notre système de détection M2FS, à l'aide d'un oscilloscope.
- ii) Calculer le surplus d'énergie consommée lors de l'intégration de la solution de compression,
- iii) Comparer entre les consommations des deux algorithmes de compression adoptés et soutenir la moins gourmande pour notre système M2FS.

Comme dans la première contribution, pour calculer la consommation, nous avons utilisé un oscilloscope numérique qui offre une sortie USB afin d'acquérir les données sous forme d'un fichier csv pour des futurs traitements. *MCP lab electronics DQ8104* (Figure 4.22 chapitre 4) est un oscilloscope numérique à 4 voies avec une fréquence qui atteint les 200 MHz.

La méthode et les équations qui entrent dans le calcul de la consommation sont déjà détaillées dans la partie résultat du chapitre précédent.

La figure 5.16 montre un écran généré par l'oscilloscope lors de la mesure du courant (quantité d'énergie par conséquent) consommée par un nœud capteur (de type Arduino Due) qui exécute le programme de détection d'un incendie de forêt, en tenant compte de la compression de données par la méthode APFF.

L'oscilloscope montre une variation de tension dans l'écran montrée dans la figure 5.16 (et du même pour la figure 5.17), mais en réalité, il s'agit de la variation du courant pour lequel nous avons placé les bornes de la sonde de l'oscilloscope sur une résistance de 1 ohm relié en série avec le fil d'alimentation, ce qui permet d'évaluer le courant au lieu de la tension ($U = I$ car $U(volts) = R(\Omega) * I(ampère)$ et $R = 1\Omega$) (plus de détail est donnée dans la partie consommation d'énergie du chapitre 4). Notons à ce stade que le nœud capteur est alimenté par une source d'énergie à une tension de sortie de 5volts.

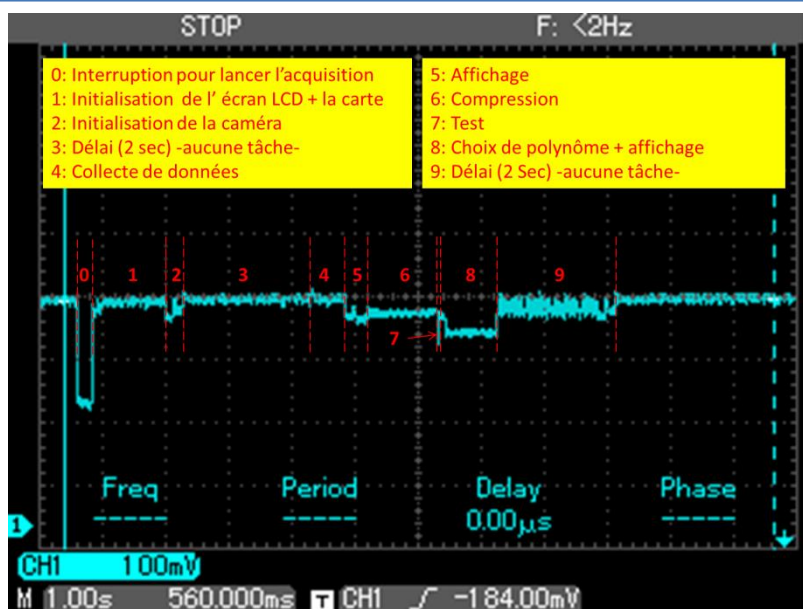


Figure 5.16 : Signal généré par l'oscilloscope qui mesure le courant en tenant compte de la compression APFF

La figure 5.16 montre un signal de 12 secondes (1 carré = 1 sec) collecté par la voie CH1, sur une fréquence inférieure à 2 Hz, partitionné manuellement en 9 intervalles (6 intervalles pour le cas la figure 5.17), dont le détail de chaque intervalle (tâche exécutée) est illustré dans la zone jaune de la figure 5.16.

De la même manière, nous montrons dans la figure 5.17 l'écran généré par l'oscilloscope lors de la mesure de la quantité d'énergie consommée (plutôt le courant) par le même nœud capteur, mais qui exécute cette fois-ci le programme de détection en tenant compte de la compression par la méthode TiNA.

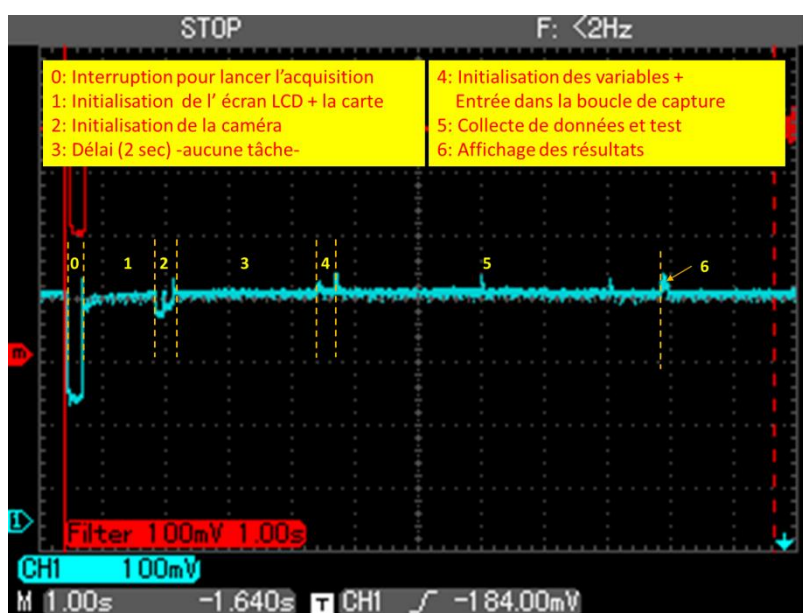


Figure 5.17 : Signal généré par l'oscilloscope pour mesurer le courant en tenant compte de la compression TiNA

Pour définir les intervalles dans le signal, nous avons calculé par le programme le temps utilisé pour exécuter chaque opération. Ensuite, nous avons projeté ces délais dans le signal montré afin de connaître le niveau de chaque opération.

Remarque :

La grande précision fournie par l'oscilloscope nous permet de détecter le temps exact pris par une opération donnée, par conséquent on peut en déduire l'erreur commise par les temporisateurs de la carte Arduino. Par exemple, le délai de 2000 ms (millisecondes) lancé après l'initialisation (comme montré dans la figure 5.18)) par le programme, est exécuté en réalité par la carte Arduino dans un délai de 2058 ms d'après son mesure dans le premier signal par l'oscilloscope!

```

16 void setup() {
17   Serial.begin (115200);
18   Serial.println ("Module ArduCAM, initialisation...");
19   .....
20   Serial.println ("----- Fin d'initialisation -----");
21   a1 = millis ();
22   delay (2000);
23   a1 = millis () - a1;
24 }

```

Figure 5.18 : Méthode de calcul du temps pris par une tâche

Nous illustrons dans la figure 5.19 une comparaison entre les quantités d'énergie prises selon les trois modes : sans compression, compression avec TiNA et compression avec APFF de l'indice IFM. Les barres en Blue représentent la consommation des tâches de collecte et de traitement pour la méthode sans compression, et en plus la tâche de compression de données en cas de compression. Les barres en rouges représentent en plus la consommation de la plateforme (Arduino Due) pour les trois expériences (sans compression, TiNA, APFF).

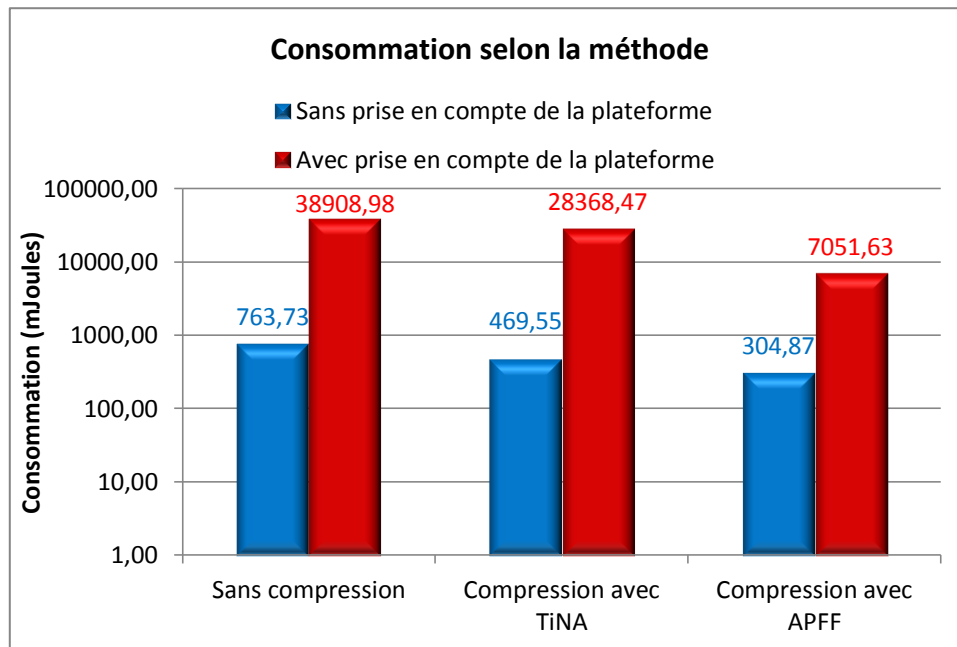


Figure 5.19 : Comparaison entre la consommation d'énergie selon trois modes : sans compression, avec les compressions TiNA et APFF¹⁰

Nous observons dans la figure 5.19 que le système M2FS sans la prise en compte des solutions de compression consomme plus d'énergie, ce qui influe directement sur sa durée de vie.

Par contre, les solutions de compression montrent un gain remarquable en énergie, quoi que ce soit en énergie dissipée par fonction ou par nœud. Encore plus, la solution de compression avec fenêtre fixe est meilleure en économie d'énergie, ce qui la rend plus adéquate pour notre système, malgré qu'elle n'est pas la plus rapide dans l'exécution des tâches de détection et de compression conformément au tableau 5.1 et la courbe de la figure 5.12.

Par ailleurs, nous pouvons toujours adapter la solution de compression APFF à notre système M2SF en l'ajustant de sorte qu'elle soit la plus rapide dans le traitement des données, et aussi la moins consommatrice d'énergie. En choisissant un nombre de données à collecter égal à 20, elle assure les deux contraintes : meilleur temps et meilleure consommation !

Remarque :

Les quantités d'énergie montrées dans la figure 5.19 sont calculées en collectant un échantillon de données égal à 20.

V.7. Conclusion

Dans ce chapitre nous avons présenté notre seconde contribution, en partant par une étude approfondie du domaine de compression, et les domaines annexes (corrélation et agrégation). Ensuite, nous avons présenté les algorithmes de compression dédiés aux RCSF (TiNA et APFF)

¹⁰ Pour des fins d'une meilleure illustration, nous avons changé l'échelle de la figure en appliquant le logarithme

afin de les intégrer en pratique dans les nœuds capteurs sources. Nous avons également présenté l'algorithme APFV que nous proposons en perspective. Nous avons modélisé ensuite notre système M2FS avec prise en compte de ces méthodes de compression de données. Nous avons enfin, présenté les résultats obtenus par notre banc d'essai en termes de temps nécessaire pris pour la compression des données collectées, en taux de compression et en énergie consommée en tenant compte de la solution de compression proposée.

Conclusion générale & perspectives

Dans cette thèse, nous avons abordé un sujet d'actualité et d'intérêt sociétal qui traite la problématique des feux de forêt. Les incendies génèrent des dégâts néfastes et contribuent aux dégagements des gaz à effet de serre classés comme l'un des acteurs indispensables au réchauffement climatique de l'univers.

Nous avons proposé un système pour la détection des feux de forêt appelé M2FS en utilisant une approche multimodale (scalaire et multimédia) dans le but est de minimiser au maximum les fausses alertes.

La méthodologie suivie pour la mise en œuvre de ce système s'articule autour des phases suivantes : phase de conception indépendamment des choix techniques, phase d'implémentation et phase de tests et d'évaluation. Ceci a constitué notre première contribution. Nous notons que le prototype réalisé est limité à un nœud source, un nœud relai et un collecteur (Sink) avec un accès au Cloud, et ce selon le cahier des charges établi ; car l'objectif visé par la thèse n'était pas de développer un banc d'essai complet et opérationnel mais plutôt valider expérimentalement nos contributions.

Dans le but d'optimiser le système M2FS sur le plan énergétique, nous avons constaté que les nœuds capteurs remontent le plus souvent au centre de décision, des grandeurs physiques scalaires (température, humidité) d'une manière périodique, et ce en absence d'événement instantané des feux de forêt. Nous avons aussi proposé une approche de compression de ces données qui est fondée sur l'approximation polynomiale pour remonter un polynôme représentatif de ces valeurs au lieu de les transmettre toutes vers le Sink.

Le délai causé par cette technique de compression n'a en fait aucun impact sur le mécanisme de surveillance vu que les données retardées et compressées par la suite sont des données non urgentes (données qui représentent une situation normale sans incendie) et qui peuvent être stockées pour des utilisations ultérieures.

Le système M2FS, avec compression basée sur une fenêtre fixe, développé dans le cadre de cette thèse pourra faire l'objet d'un prototype opérationnel après l'avoir validé par les experts du domaine.

Cependant, ce système est loin d'être finalisé, et nécessite selon notre expérience plusieurs améliorations que nous pourrions proposer comme travaux de recherche en perspective, et qui peuvent se résumer comme suit :

- Implémentation de la version actuelle du système M2FS en considérant la technique intéressante de compression scalaire avec fenêtre variable, dont la conception a été abordée dans cette thèse.
- Enrichir le prototype développé en ajoutant plusieurs nœuds capteurs relais et plusieurs nœuds sources pour la mise en place d'un banc d'essai complet et près à être déployé réellement pour les services de feux de forêts.
- Organisation du RCMSF en clusters dans le but est d'enrichir le système M2FS par la fonction d'agrégation de polynômes au niveau de cluster chefs générés par les nœuds membres du cluster.
- Stockage des données collectées par le système M2FS au niveau des structures modernes comme le Cloud ou le *fog*, et développement des techniques de *data mining* et de l'intelligence artificielle pour une meilleure exploitation des données.

Annexe A :

Code sources et programmes

1. Introduction

Nous présentons dans ce qui suit les codes sources et les programmes qui entrent dans l'élaboration de la solution de détection d'un feu de forêt développée.

Nous avons développé deux applications, la première concerne le système M2FS qui est détaillée dans la première contribution (chapitre 4). La deuxième fait intervenir la compression de données avant la transmission de deux façons TiNA et APFF comme montré dans la deuxième contribution (chapitre 5).

Dans les deux applications, nous avons fait intervenir le même banc d'essai (présenté dans le chapitre 4). Aussi, le code de collecte et de détection des événements ou de capture d'image décrit reste identique dans les deux contributions. Ce qui différencie la deuxième contribution, est l'intégration des deux modes de compression (APFF, TiNA), et le type des données scalaires remontés au Sink.

Les programmes chargés des opérations citées plus haut sont écrits en langage Arduino.

D'autres programmes sont aussi implémentés au niveau de la station de base en langage Python pour remonter et stocker les données et les images vers un Cloud (DropBox). En utilisant python toujours, nous avons développé une application qui permet de restituer les données compressées, afficher les images stockées (sur DropBox) et consulter le contenu de la base de données. Cette dernière application s'installe au niveau de superviseur ou l'utilisateur chargé de prise de décision !

2. Programme de M2FS

Le code source Arduino affiché dans la figure A.1 permet de collecter les grandeurs physiques température et humidité, calcul l'indice IFM (ici c'est l'appelle à la fonction IFM, son détail est illustrée dans la figure A.2 dans le fichier IFM.h), capturer une image si l'indice IFM dépasse le seuil, et envoi l'image au Sink.

Arduino.ino	
1	// Bibliothèques et initialisation
2	
3	void setup()
4	{
5	...
6	Serial.begin(38400);
7	Serial.println("Module ArduCAMdemarrage...");
8	// initialize SPI:
9	SPI.begin();
10	//Check if the camera module type is OV2640
11	myCAM.rdSensorReg8_8(OV2640_CHIPID_HIGH, &vid);
12	myCAM.rdSensorReg8_8(OV2640_CHIPID_LOW, &pid);
13	if((vid != 0x26) (pid != 0x42))
14	Serial.println("Can'tfind OV2640 module!");
15	else
16	Serial.println("Camera OV2640 connectee correctement ");
17	...
18	Serial.println("----- Fin d'initialisation -----");
19	}
20	voidloop()
21	{
22	...
23	floatresult;
24	float h = dht1.readHumidity();
25	float t = dht1.readTemperature();
26	if (isnan(t) isnan(h)) {
27	Serial.println("Initialisation de DHT ...");
28	}
29	else {
30	result = fwi(t, h, 13, 0, 7); // calculate IFM index
31	Serial.print("\nHumidite: ");
32	Serial.print(h);
33	Serial.print(" %;\t");
34	Serial.print("Temperature: ");
35	Serial.print(t);
36	Serial.print(" °C;\t");
37	Serial.print(" IFM: ");
38	Serial.println(result);}
39	
40	if((myCAM.read_reg(ARDUCHIP_TRIG)) && (result>5))
41	{
42	Serial.println("----- Alerte scalaire ==> Activer la Camera -----");
43	...
44	myCAM.set_format(JPEG);
45	myCAM.InitCAM(); // Initialisation de la caméra
46	myCAM.OV2640_set_JPEG_size(OV2640_160x120); //Resolution = 160*120
47	...
48	myCAM.flush_fifo();//Flush the FIFO
49	myCAM.start_capture(); //Start capture
50	if(myCAM.read_reg(ARDUCHIP_TRIG) && CAP_DONE_MASK)
51	{
52	Serial.println("==> Capture terminee !");
53	i = 0;
54	temp = myCAM.read_fifo();
55	buf[i++] = temp; //Write first image data to buffer
56	//Read JPEG data from FIFO
57	while((temp != 0xD9) (temp_last != 0xFF))

```

58 {
59   temp_last = temp;
60   temp = myCAM.read_fifo();
61   Serial.print(temp_last,HEX);//Write image data to buffer if not full
62   ...
63   Serial.println("==> Image affichee !");
64   Serial.println("-----");
65   ...
66   ...
67 }

```

Figure A.1 : Extrait du code source pour programmer la carte Arduino

La figure A.2 illustre les étapes de calcul de l'indice IFM.

IFM.h	
1	//Initialisation
2	...
3	// calcul du FPMC
4	Void ffmc(int T, int H, int W, int r0, int f0){
5	...
6	// calcul du DMC
7	Void dmc(int T, int H, int W, int r0, fl int oat P0, int i)
8	{
9	...
10	// calcul du DC
11	Void dc(int T, int H, int W, int r0, int D0, int i)
12	...
13	//calcul de FWI
14	Void fwi(int T, int H, int W, int r0, int i)
15	{
16	...
17	if (g==0)
18	f0=85; P0=6; D0=15; g=1;
19	else
20	f0=F; P0=P; D0=D;
21	Ff=ffmc(T, H, W, r0, f0);
22	Dm=dmc(T, H, W, r0, P0, i);
23	Dc=dc(T, H, W, r0, D0, i);
24	s=0.4*Dc;
25	if (Dm <=s)
26	U=0.8*Dm*Dc/(P+s);
27	else
28	U=Dm-((1-((0.8*Dc)/(Dm+s)))*(0.92+pow((0.0114*Dm),1.7)));
29	if (U <=80)
30	t=0.626* pow(U,0.809)+2;
31	else
32	t=1000/(25+(108.64*exp((-0.023)*U)));
33	B = 0.1*t*0.208* 91.9*exp((-0.1386)*m)*(1+(pow(m,5.31)/(4.93 * pow(10,7))))* exp(0.05039*W);
34	if (B > 1)
35	return exp(2.72*pow(0.434*log(B),0.647)); //c'est l'indice IFM (FWI)
36	else
	return B; // B est la valeur de l'indice IFM (FWI)
	}

Figure A.2 : Extrait du code source de calcul de l'indice IFM

3. Programme de compression pour M2FS

Le code source Arduino présenté dans la figure A.3 permet de compléter le code de la figure A.1 afin d'intégrer le code de la compression APFF.

APFF.ino	
1	// Bibliothèques et initialisation
2	...
3	// Code de collecte montré dans la figure 6.1 et de calcul de l'ifim montré dans la figure 6.2
4	...
5	deg = 1; // initialisation du degré à 1
6	Serial.print("\nApproximation ... \n");
7	while (deg < n)
8	{
9	polyfit(x, y, n, deg, sol);
10	polyfit(x, hy, n, deg, hsol);
11	poly_val(sol, n, deg, sol_img, x); // régénération des valeurs de T après l'approximation
12	poly_val(hsol, n, deg, hsol_img, x); // régénération des valeurs de H après l'approximation
13	err_appr(y, sol_img, err_approx, n); // erreur d'approximation T
14	err_appr(hy, hsol_img, herr_approx, n); // erreur d'approximation H
15	var = variat(err_approx, n); // variation de l'erreur T
16	hvar = variat(herr_approx, n); // variation de l'erreur H
17	err_min[deg] = var;
18	herr_min[deg] = hvar;
19	deg++;
20	}
21	...
22	
23	Serial.print(" \n\n\tRésumé des résultats\n-----");
24	Serial.print("\nLe Nombre de coefficients transmissibles (temp)est =");Serial.print(valeur);
25	valeur = valeur / (i * j) * 100;
26	Serial.print("\nLe taux de compression (temp) est : ");Serial.print(valeur); Serial.print(" %");
27	Serial.print("\nLe gain (temp) = ");Serial.print(100 - valeur); Serial.print(" %");
28	Serial.print("\n\nLe Nombre de coefficients transmissibles (hum)est =");Serial.print(hvaleur);
29	haleur = hvaleur / (i * j) * 100;
30	Serial.print("\nLe taux de compression (hum) est : ");Serial.print(hvaleur); Serial.print(" %");
31	Serial.print("\nLe gain (hum) = "); Serial.print(100 - hvaleur); Serial.print(" %");
32	Serial.print("\n\nLe Nombre total de valeurs est : ");Serial.print(j * i);
33	Serial.print("\nLe nombre de mesures par séquence ");Serial.print(n);
34	Serial.print("\nLe Nombre total de séquences est : ");Serial.print(nb_seq);
35	Serial.print("\nL'erreur tolérée fixe est : ");Serial.print(err_tolér);
36	Serial.print("\n\tRésumé terminé\n-----");
37	a2 = millis() - a2;
38	ta2 += a2;
39	Serial.print("\n le temps total d'approximation (ms), a0 ="); Serial.print(ta0);
40	Serial.print("\n le temps de creation des polynomes (ms),a1 ="); Serial.print(ta1);
41	Serial.print("\n le temps total d'affichage et de stockage (ms), a2 =");Serial.print(ta2 + ta3);
42	delay(3000);
43	}
44	...
45	}

Figure A.3 : Extrait du code source pour programmer la carte Arduino en tenant compte de la compression APFF

Le deuxième mode de compression élaboré dans la deuxième contribution est celui du TiNA, la figure A.4 montre un extrait du code source du programme Arduino qui tient compte de la compression TiNA.

TiNA.ino	
1	// Bibliothèques et initialisation
2	...
3	// Code de collecte montré dans la figure 6.1 et de calcul de l'IFM montré dans la figure 6.2
4	...
5	Serial.print("\nCOMPRESSION ... \n");
6	j=0;
7	k=0;
8	for(i=1;i<n;i++)
9	{a0=millis();
10	tval = abs(t[i] - t[i - 1]) / t[i - 1];
11	if (tval>Tct)
12	{
13	tsend[i] = tval;
14	j++;
15	}
16	hval = abs(h[i] - h[i - 1]) / h[i - 1];
17	if (hval>Tct)
18	{
19	hsend[i] = hval;
20	//Serial.print(hval);
21	k++;
22	}
23	a0=millis() - a0;
24	ta0+=a0;
25	}
26	...
27	Serial.print("\n-----Résumé des Résultats-----\n ");
28	Serial.print("\nLe Nombre total de valeurs est : "); Serial.print(n);
29	Serial.print("\nLa tolérance de cohérence temporelle (Tct)= "); Serial.print(Tct);
30	Serial.print("\nLe Nombre de coefficients transmissibles (temp) est ="); Serial.print(j);
31	//Serial.print("\nLe taux de compression (temps) est : "); Serial.print(j / n * 100); Serial.print(" %");
32	//Serial.print("\nLe gain = "); Serial.print(100 - (j / n * 100)); Serial.print(" %");
33	Serial.print("\n\nLe Nombre de coefficients transmissibles (hum) est ="); Serial.print(k);
34	//Serial.print("\nLe taux de compression (hum) est : "); Serial.print(k / n * 100); Serial.print(" %");
35	//Serial.print("\nLe gain = "); Serial.print(100 - (k / n * 100)); Serial.print(" %");
36	Serial.print("\n\nLe temps de compression = "); Serial.print(ta0); Serial.print(" ms");
37	Serial.print("\n-----Résumé terminé\n-----");
38	...

Figure A.4 : Extrait du code source pour programmer la carte Arduino en tenant compte de la compression TiNA

Bibliographie

- [1] A. A. A. Alkhatib, "A Review on Forest Fire Detection Techniques," (in en), *International Journal of Distributed Sensor Networks*, vol. 10, no. 3, p. 597368, March 1, 2014 2014.
- [2] V. Chowdary, M. K. Gupta, and R. Singh, "A Review on Forest Fire Detection Techniques: A Decadal Perspective," (in en-US), *International Journal of Engineering & Technology*, vol. 7, no. 3.12, pp. 1312-1316, 2018/07/20 2018.
- [3] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: a survey," (in en), *Computer Networks*, vol. 38, no. 4, pp. 393-422, 3/2002 2002.
- [4] K. BAKAR, "Techniques de compressiion et d'agrégatiion de à effiicaciité énergétiique dans lles RCSF," Magister en Informatique, Université d'Oran1, Oran, Algérie, 2008.
- [5] B. Kechar and B. Zahia, "Energy Efficient Data Compression technique for Wireless Sensor Networks in Tolerant End-to-End Delay Monitoring Applications," (in en), *Communications of the ACS*, vol. 3, no. 2, 24/02/2011 2011.
- [6] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A survey on sensor networks," *IEEE Communications Magazine*, vol. 40, no. 8, pp. 102-114, August 2002 2002.
- [7] Z. J. Chew and M. Zhu, "Combined power extraction with adaptive power management module for increased piezoelectric energy harvesting to power wireless sensor nodes," in *2016 IEEE SENSORS*, 2016, pp. 1-3.
- [8] D. Zhu and S. Beeby, "Kinetic Energy Harvesting," in *Energy Harvesting Systems: Principles, Modeling and Applications*, T. J. Kaźmierski and S. Beeby, Eds. New York, NY: Springer New York, 2011, pp. 1-77.
- [9] C. Alippi and C. Galperti, "An Adaptive System for Optimal Solar Energy Harvesting in Wireless Sensor Network Nodes," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 55, no. 6, pp. 1742-1750, July 2008 2008.
- [10] D. W. Wang, J. L. Mo, X. F. Wang, H. Ouyang, and Z. R. Zhou, "Experimental and numerical investigations of the piezoelectric energy harvesting via friction-induced vibration," *Energy Conversion and Management*, vol. 171, pp. 1134-1149, September 1, 2018 2018.
- [11] "List of wireless sensor nodes," in *Wikipedia*, [En ligne]. Disponible sur: https://en.wikipedia.org/wiki/List_of_wireless_sensor_nodes [Consulté le: 10-11-2018].
- [12] S. Dixit, « Wireless Sensor Networks: Issues & Challenges », *International Journal of Computer Science and Mobile Computing (IJCSMC)*, Vol. 3, no 6, pp 681-685, june 2014.
- [13] A. Hadjidj, M. Souil, A. Bouabdallah, Y. Challal, and H. Owen, "Wireless sensor networks for rehabilitation applications: Challenges and opportunities," *Journal of Network and Computer Applications*, vol. 36, no. 1, pp. 1-15, January 1, 2013 2013.
- [14] Y. Zatout, "Conception et évaluation de performances d'un réseau de capteurs sans fil hétérogène pour une application domotique," Doctorat, Université Toulouse 2 Le Mirail (UT2 Le Mirail), France, 2011.
- [15] G. Zhou, J. Lu, C.-. Wan, M. D. Yarvis, and J. A. Stankovic, "BodyQoS: Adaptive and Radio-Agnostic QoS for Body Sensor Networks," in *IEEE INFOCOM 2008 - The 27th Conference on Computer Communications*, 2008, pp. 565-573, Phoenix, AZ,USA: IEEE.
- [16] N. Correal et N. Patwari, « Wireless sensor networks: Challenges and opportunities », in *MPRG/Virgina Tech Wireless Symposium*, USA, pp 1-9, 2001.
- [17] J. A. Rice and B. F. S. Jr, "Structural health monitoring sensor development for the Imote2 platform," in *Sensors and Smart Structures Technologies for Civil, Mechanical, and*

- Aerospace Systems 2008*, 2008, vol. 6932, p. 693234: International Society for Optics and Photonics.
- [18] I. F. Akyildiz, T. Melodia, and K. R. Chowdhury, "A survey on wireless multimedia sensor networks," *Computer Networks*, vol. 51, no. 4, pp. 921-960, March 14, 2007 2007.
 - [19] G. C. Holst, *Ccd Arrays, Cameras, and Displays*, 2nd ed. Winter Park, FL : Bellingham, Wash., USA: Society of Photo Optical, 1998, p. 378.
 - [20] E. Culurciello and A. G. Andreou, "CMOS image sensors for sensor networks," (in en), *Analog Integrated Circuits and Signal Processing*, vol. 49, no. 1, pp. 39-51, 2006-10-01 2006.
 - [21] S. Mohammadi and A. Esfandiari, "Magnetostrictive vibration energy harvesting using strain energy method," *Energy*, vol. 81, pp. 519-525, March 1, 2015 2015.
 - [22] C. Pham, "Communication performances of IEEE 802.15.4 wireless sensor motes for data-intensive applications: A comparison of WaspMote, Arduino MEGA, TelosB, MicaZ and iMote2 for image surveillance," *Journal of Network and Computer Applications*, vol. 46, pp. 48-59, November 1, 2014 2014.
 - [23] S. Hengstler, D. Prashanth, S. Fong, and H. Aghajan, "MeshEye: A Hybrid-Resolution Smart Camera Mote for Applications in Distributed Intelligent Surveillance," in *2007 6th International Symposium on Information Processing in Sensor Networks*, 2007, pp. 360-369.
 - [24] M. Rahimi *et al.*, "Cyclops: In Situ Image Sensing and Interpretation in Wireless Sensor Networks," 2005, pp. 192-204, New York, NY, USA: ACM.
 - [25] B. N. Chun *et al.*, "Mirage: A Microeconomic Resource Allocation System for SensorNet Testbeds," (in en_US), *Proceedings of the 2nd IEEE workshop on Embedded Networked Sensors*, 2005 2005.
 - [26] A. Rowe, A. Goode, I. Nourbakhsh, and D. Goel, "CMUcam3: An Open Programmable Embedded Vision Sensor," Carnegie Mellon University, Pittsburgh, Pennsylvania, "Projet CMU-RI-TR-07-13, may 2007 2007.
 - [27] J. Hill, M. Horton, R. Kling, L. Krishnamurthy, and L. Krishnamurthy, "The platforms enabling wireless sensor networks," *Communications of the ACM*, vol. 47, no. 6, pp. 41-46, 2004 2004.
 - [28] C. B. Mudgule, U. Nagaraj, P. D. Ganjewar, and P. Student, "Data Compression in Wireless Sensor Network: A Survey," (in en), vol. 2, no. 11, p. 10, 2007 2007.
 - [29] A. Milenković, C. Otto, and E. Jovanov, "Wireless sensor networks for personal health monitoring: Issues and an implementation," *Computer Communications*, vol. 29, no. 13, pp. 2521-2533, August 21, 2006 2006.
 - [30] G. Anastasi, M. Conti, M. Di Francesco, and A. Passarella, "Energy conservation in wireless sensor networks: A survey," *Ad hoc networks*, vol. 7, no. 3, pp. 537-568, 2009 2009.
 - [31] H. Bambang and H. Song, "Wireless Multimedia Sensor Networks Applications and Security Challenges," in *International Conference on Broadband, Wireless Computing, Communication and Applications*, 2010, pp. 842-847.
 - [32] S. Han, E. Chang, L. Gao, and T. Dillon, "Taxonomy of Attacks on Wireless Sensor Networks," 2006, pp. 97-105: Springer London.
 - [33] N. Bendimerad, "Système de surveillance d'infrastructures publiques à l'aide des réseaux de capteurs vidéo sans fil," Doctorat, ORAN, Algérie, 2015.
 - [34] N. Houache, "Conception et implémentation d'un système de surveillance des feux de forêts basé sur les réseaux de capteurs sans fils," Magister en Informatique, Université d'Oran1, Oran, Algérie, 2012.
 - [35] S. Park, A. Savvides, and M. B. Srivastava, "SensorSim: A simulation framework for sensor networks," 2000, pp. 104-111: ACM.

- [36] D. Pediaditakis, Y. Tselishchev, and A. Boulis, "Performance and scalability evaluation of the Castalia wireless sensor network simulator," 2010, pp. 53-59: ICST (Institute for Computer Sciences, Social-Informatics and
- [37] P. Gaur and M. P. Tahiliani, "Operating systems for IoT devices: A critical survey," 2015, pp. 33–36: IEEE.
- [38] M. Imran, A. M. Said, and H. Hasbullah, "A survey of simulators, emulators and testbeds for wireless sensor networks," in *2010 International Symposium on Information Technology (ITSim 2010)*, 2010, pp. 897-902, Kuala Lumpur, Malaysia: IEEE.
- [39] T. Issariyakul and E. Hossain, "Introduction to Network Simulator 2 (NS2)," in *Introduction to network simulator NS2*: Springer, 2009, pp. 1–18.
- [40] A. Varga and R. Hornig, "An overview of the OMNeT++ simulation environment," 2008, p. 60: ICST (Institute for Computer Sciences, Social-Informatics and
- [41] A. Sobeih *et al.*, "J-Sim: a simulation and emulation environment for wireless sensor networks," *IEEE Wireless Communications*, vol. 13, no. 4, pp. 104–119, 2006 2006.
- [42] X. Zeng, R. Bagrodia, and M. Gerla, "GloMoSim: a library for parallel simulation of large-scale wireless networks," 1998, pp. 154–161: IEEE.
- [43] X. Chang, « Network simulations with OPNET », in WSC'99. 1999 Winter Simulation Conference Proceedings.'Simulation-A Bridge to the Future'(Cat. No. 99CH37038), vol. 1, pp 307–314, 1999.
- [44] P. Levis, N. Lee, M. Welsh, and D. Culler, "TOSSIM: Accurate and scalable simulation of entire TinyOS applications," 2003, pp. 126–137: ACM.
- [45] J. Polley, D. Blazakis, J. McGee, D. Rusk, and J. S. Baras, "ATEMU: a fine-grained sensor network simulator," 2004, pp. 145–152: IEEE.
- [46] B. L. Titzer, D. K. Lee, and J. Palsberg, "Aurora: Scalable sensor network simulation with precise timing," 2005, pp. 477–482: IEEE.
- [47] L. Girod, J. Elson, A. Cerpa, T. Stathopoulos, N. Ramanathan, and D. Estrin, "EmStar: A Software Environment for Developing and Deploying Wireless Sensor Networks.," 2004, pp. 283–296.
- [48] G. Werner-Allen, P. Swieskowski, and M. Welsh, "Motelab: A wireless sensor network testbed," 2005, p. 68: IEEE Press.
- [49] G. Barrenetxea, F. Ingelrest, G. Schaefer, M. Vetterli, O. Couach, and M. Parlange, "Sensorscope: Out-of-the-box environmental monitoring," 2008, pp. 332–343: IEEE Computer Society.
- [50] D. Johnson *et al.*, "Mobile emulab: A robotic wireless and sensor network testbed," 2006, pp. 1–12: IEEE.
- [51] R. Crepaldi *et al.*, "The design, deployment, and analysis of signetlab: A sensor network testbed and interactive management tool," 2007, pp. 1–10: IEEE.
- [52] D. Raychaudhuri *et al.*, "Overview of the ORBIT radio grid testbed for evaluation of next-generation wireless network protocols," 2005, vol. 3, pp. 1664–1669: IEEE.
- [53] L. Sadoudi, "Étude et développement d'une plateforme de communication pour les réseaux de capteurs acoustiques sans fil: application au contrôle-santé des rails par corrélation du bruit ambiant," PhD Thesis, Université de Valenciennes et du Hainaut-Cambresis, 2016.
- [54] « conference sur le climat », www.connaissancedesenergies.org. [En ligne]. Disponible sur: <https://www.google.com/url?sa=i&source=web&cd=&ved=0ahUKEwjdiYmascTjAhWQAWMBHcBOAI0QzPwBCAI&url=https%3A%2F%2Fwww.connaissancedesenergies.org%2Ffiche-pedagogique%2Fcop21-conference-sur-le-climat-de-paris&psig=AOvVaw0UISd8lOCX4ecpL13CAS9o&ust=1563742581794367>. [Consulté le: 16-juin-2019].

- [55] N. Alsharabi, L. R. Fa, F. Zing, and M. Ghurab, "Wireless sensor networks of battlefields hotspot: Challenges and solutions," in *2008 6th International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks and Workshops (WiOPT)*, 2008, pp. 192-196, Berlin, Germany: IEEE.
- [56] J. Yick, B. Mukherjee, and D. Ghosal, "Wireless sensor network survey," *Computer Networks*, vol. 52, no. 12, pp. 2292-2330, August 22, 2008 2008.
- [57] L. Xuemei, J. Liangzhong, and L. Jincheng, "Home healthcare platform based on wireless sensor networks," in *2008 International Conference on Information Technology and Applications in Biomedicine*, 2008, pp. 263-266.
- [58] T. S. Perry, « Sensors Slip into the Brain, Then Dissolve When the Job Is Done », *IEEE Spectrum: Technology, Engineering, and Science News*, Bethesda Maryland USA, 2016.
- [59] S. V. Srikanth, P. P.J, D. K.P, T. S, M. U. Patil, and S. C. B. N, "Design and Implementation of a Prototype Smart PARKing (SPARK) System Using Wireless Sensor Networks," in *2009 International Conference on Advanced Information Networking and Applications Workshops*, 2009, pp. 401-406.
- [60] H. Tsai, O. K. Tonguz, C. Saraydar, T. Talty, M. Ames, and A. Macdonald, "Zigbee-based intra-car wireless sensor networks: a case study," *IEEE Wireless Communications*, vol. 14, no. 6, pp. 67-77, December 2007 2007.
- [61] « Les forêts de nos territoires », Office national des forêts, [Page web], Disponible sur: <https://www.onf.fr/onf/forets-et-espaces-naturels/+20::les-forets-de-nos-territoires.html>, [Consulté le: 07-mai-2019].
- [62] "Bilan Final des incendies de forets -2017-," DIRECTION DE LA PROTECTION DE LA FAUNE ET DE LA FLORE, Oran, Algérie 2017 2017.
- [63] F. Z. HAMDI, "Analyses des bilans des incendies de forêts de la Wilaya De Mascara: période (2010-2016)," Master Forestier, UNIVERSITE de TLEMCEN, Tlemcen, Algérie, 2017.
- [64] B. GHERABI, "CONTRIBUTION A L'ETUDE DE LA REPRISE VEGETATIVE DU CHENE LIEGE APRES INCENDIE CAS DE LA FORET DE ZARIFET (WILAYA DE TLEMCEN)," Ingéniorat forestier, UNIVERSITE de TLEMCEN, Tlemcen, Algérie, 2013.
- [65] « Planetoscope - Statistiques ». [En ligne]. Disponible sur: <https://www.planetoscope.com/forets/903-hectares-de-terres-detruits-dans-le-monde-par-des-incendies.html>. [Consulté le: 01-déc-2018].
- [66] J. Fleming and R. G. Robertson, "Fire Management Tech Tips: The Osborne Fire Finder," Technical Report 0351 1311-SDTDC, USDA Forest Service 2003 2003.
- [67] R. Smith and S. Rivers, "The Osborne Firefinder," ed: U.S. Department of Agriculture Forest Service, 1955.
- [68] A. A. Alkhatib, M. Alia, and A. Hnaif, "Smart System for Forest Fire Using Sensor Network," *International Journal of Security and Its Applications*, vol. 11, no. 7, pp. 1-16, 2017 2017.
- [69] « Detecting forest fires with satellites (MODIS and VIIRS) », UN-SPIDER Knowledge Portal. [En ligne]. Disponible sur: <http://www.un-spider.org/news-and-events/news/detecting-forest-fires-satellites-modis-and-viirs>. [Consulté le: 07-mai-2019].
- [70] « Global Fire Monitoring », 22-oct-1999. [En ligne]. Disponible sur: https://earthobservatory.nasa.gov/features/GlobalFire/fire_5.php. [Consulté le: 07-mai-2019].
- [71] H. Lin, X. Liu, X. Wang, and Y. Liu, "A fuzzy inference and big data analysis algorithm for the prediction of forest fire based on rechargeable wireless sensor networks," *Sustainable Computing: Informatics and Systems*, vol. 18, pp. 101-111, 2018 2018.
- [72] P. Bolourchi and S. Uysal, "Forest fire detection in wireless sensor network using fuzzy logic," 2013, pp. 83-87: IEEE.

- [73] S. Garcia-Jimenez, A. Jurio, M. Pagola, L. De Miguel, E. Barrenechea, and H. Bustince, "Forest fire detection: A fuzzy system approach based on overlap indices," *Applied Soft Computing*, vol. 52, pp. 834–842, 2017 2017.
- [74] A. Kansal, Y. Singh, N. Kumar, and V. Mohindru, "Detection of forest fires using machine learning technique: A perspective," 2015, pp. 241–245: IEEE.
- [75] K. Trivedi and A. K. Srivastava, "An energy efficient framework for detection and monitoring of forest fire using mobile agent in wireless sensor networks," in *2014 IEEE International Conference on Computational Intelligence and Computing Research*, 2014, pp. 1-4.
- [76] J. Fernández-Berni, R. Carmona-Galán, J. F. Martínez-Carmona, and Á. Rodríguez-Vázquez, "Early forest fire detection by vision-enabled wireless sensor networks," (in en), *International Journal of Wildland Fire*, vol. 21, no. 8, pp. 938-949, 2012/12/18 2012.
- [77] A. Díaz-Ramírez, L. A. Tafoya, J. A. Atempa, and P. Mejía-Alvarez, "Wireless Sensor Networks and Fusion Information Methods for Forest Fire Detection," *Procedia Technology*, vol. 3, pp. 69-79, January 1, 2012 2012.
- [78] S. R. Vijayalakshmi and S. Muruganand, "Real Time Monitoring of Wireless Fire Detection Node," *Procedia Technology*, vol. 24, pp. 1113-1119, January 1, 2016 2016.
- [79] M. Anand, U. A. Ganesh, S. Arun, M. Dinesh, P. Gunaseelan, and R. Karthik, "Forest Fire Detection Using Optimized Solar – Powered Zigbee Wireless Sensor Networks," (in en), vol. 4, no. 6, p. 11, 2013 2013.
- [80] H. Cruz, M. Eckert, J. Meneses, and J.-F. Martínez, "Efficient Forest Fire Detection Index for Application in Unmanned Aerial Systems (UASs)," (in eng), *Sensors (Basel, Switzerland)*, vol. 16, no. 6, Jun 16, 2016 2016.
- [81] M. Erdelj, M. Król, and E. Natalizio, "Wireless Sensor Networks and Multi-UAV systems for natural disaster management," *Computer Networks*, vol. 124, pp. 72-86, September 4, 2017 2017.
- [82] M. Dutta, S. Bhowmik, and C. Giri, "Fuzzy Logic Based Implementation for Forest Fire Detection Using Wireless Sensor Network," 2014, pp. 319-327: Springer International Publishing.
- [83] S. Roy, R. Bose, and D. Sarddar, "Self-servicing energy efficient routing strategy for smart forest," (in en), *Brazilian Journal of Science and Technology*, vol. 3, no. 1, p. 13, 2016-04-19 2016.
- [84] A. Alkhatib, "Sub-Network Coverage Method As an Efficient Method of Wireless Sensor Networks for Forest Fire Detection," 2016, pp. 65:1–65:7, New York, NY, USA: ACM.
- [85] K. Bouabdellah, H. Noureddine, and S. Larbi, "Using Wireless Sensor Networks for Reliable Forest Fires Detection," *Procedia Computer Science*, vol. 19, pp. 794-801, January 1, 2013 2013.
- [86] D. Gao, X. Yin, and Y. Liu, "PREDICTION OF FOREST FIRE USING WIRELESS SENSOR NETWORK," *Journal of Tropical Forest Science*, vol. 27, no. 3, pp. 342-350, 2015 2015.
- [87] B. Kosucu, K. Irgan, G. Kucuk, and S. Baydere, "FireSenseTB: A Wireless Sensor Networks Testbed for Forest Fire Detection," 2009, pp. 1173–1177, New York, NY, USA: ACM.
- [88] J. G. Cantuña, D. Bastidas, S. Solórzano, and J. Clairand, "Design and implementation of a Wireless Sensor Network to detect forest fires," in *2017 Fourth International Conference on eDemocracy eGovernment (ICEDEG)*, 2017, pp. 15-21.
- [89] A. S. Bhosle and L. M. Gavhane, "Forest disaster management with wireless sensor network," in *2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT)*, 2016, pp. 287-289.

- [90] Y. E. Aslan, I. Korpeoglu, and Ö. Ulusoy, "A framework for use of wireless sensor networks in forest fire detection and monitoring," *Computers, Environment and Urban Systems*, vol. 36, no. 6, pp. 614-625, November 1, 2012 2012.
- [91] Y. Sabri and N. El Kamoun, "Forest fire detection and localization with wireless sensor networks," 2013, pp. 321–325: Springer.
- [92] M. Saoudi, A. Bounceur, R. Euler, and T. Kechadi, "Data mining techniques applied to wireless sensor networks for early forest fire detection," 2016, p. 71: ACM.
- [93] S. Anand and K. M. RK, "FPGA implementation of artificial neural network for forest fire detection in wireless sensor network," 2017, pp. 265–270: IEEE.
- [94] V. G. Gasull, D. F. Larios, J. Barbancho, C. León, and M. S. Obaidat, "A wildfire prediction based on fuzzy inference system for wireless sensor networks," 2011, pp. 43–59: Springer.
- [95] C. E. Van Wagner and T. L. Pickett, *Equations and FORTRAN program for the Canadian Forest Fire Weather Index System*. 1985.
- [96] J. A. Turner and B. D. Lawson, "Weather in the Canadian forest fire danger rating system. A user guide to national standards and practices," (in English), vol. 177, 1978 1978.
- [97] V. W. C.E., "élaboration et structure de la méthode canadienne de l'indice Forêt-Météo," ed: Micromedia Ltée, 1987.
- [98] R. n. Canada. *Système canadien d'information sur les feux de végétation | Exemples de comportement du feu dans des peuplements de pins gris*.
- [99] A. J. Dowdy, C. f. A. W. a. C. Research, Australia, B. o. Meteorology, and C. S. a. I. R. O. (Australia), *Australian fire weather as represented by the McArthur Forest Fire Danger Index and the Canadian Forest Fire Weather Index*. Melbourne: Centre for Australian Weather and Climate Research, 2009.
- [100] S. Andrade-Morelli, E. Ruiz-Sánchez, E. Granell, and J. Lloret, "Energy Consumption of Wireless Network Access Points," 2013, pp. 81-91: Springer Berlin Heidelberg.
- [101] B. Son and J.-G. Kim, "A Design and Implementation of Forest-Fires Surveillance System based on Wireless Sensor Networks for South Korea Mountains," 2006.
- [102] C. Hartung, R. Han, C. Seielstad, and S. Holbrook, "FireWxNet: A Multi-tiered Portable Wireless System for Monitoring Weather Conditions in Wildland Fire Environments," 2006, pp. 28–41, New York, NY, USA: ACM.
- [103] N. Houache and B. Kechar." *Field Experiment Testbed for Forest Fire Detection using Wireless Multimedia Sensor Network*", *International Journal of Sensors, Wireless Communications and Control*, 2019, vol. 9, no. 4, pp.1-12.
- [104] Spécifications techniques de LORA «Low-Power Long Range LoRa® Technology Transceiver Module », disponible sur : ww1.microchip.com/downloads/en/DeviceDoc/50002346C.pdf, [Consulté le : 30/06/2019].
- [105] Spécifications techniques de xbee« Xbee , Digi », disponible sur <https://www.digi.com/xbee>, [Consulté le : 30/06/2019].
- [106] C. W. Clenshaw and A. R. Curtis, "A method for numerical integration on an automatic computer," *Numerische Mathematik*, vol. 2, no. 1, pp. 197–205, 1960 1960.
- [107] S. M. Magdi and X. Yuanqing, "Cloud Computing," in *Networked Control Systems, Cloud Control and Secure Control*Elsevier ed., pp. 91-125.
- [108] U. J, V. T, and D. P, "A new lossless neighborhood indexing sequence (NIS) algorithm for data compression in wireless sensor networks," *Ad Hoc Networks*, vol. 83, pp. 149-157, February 1, 2019 2019.
- [109] T. Schoellhammer, B. Greenstein, E. Osterweil, M. Wimbrow, and D. Estrin, "Lightweight temporal compression of microclimate datasets [wireless sensor networks]," in *29th Annual IEEE International Conference on Local Computer Networks*, 2004, pp. 516-524.

- [110] E. P. Capo-Chichi, H. Guyennet, and J. Friedt, "K-RLE: A New Data Compression Algorithm for Wireless Sensor Network," in *2009 Third International Conference on Sensor Technologies and Applications*, 2009, pp. 502-507.
- [111] C. M. Sadler and M. Martonosi, "Data Compression Algorithms for Energy-constrained Devices in Delay Tolerant Networks," 2006, pp. 265–278, New York, NY, USA: ACM.
- [112] X. Fatos, L. Fang-Yie et H. Li-Ling, «*Smart Sensors Networks*», Book, Elsevier, 2017, Doi: 10.1016/C2015-0-05917-2..
- [113] F. Marcelloni and M. Vecchio, "An Efficient Lossless Compression Algorithm for Tiny Nodes of Monitoring Wireless Sensor Networks," *The Computer Journal*, vol. 52, no. 8, pp. 969-987, November 2009 2009.
- [114] P. Kasirajan, C. Larsen, and S. Jagannathan, "A New Data Aggregation Scheme via Adaptive Compression for Wireless Sensor Networks," *ACM Trans. Sen. Netw.*, vol. 9, no. 1, pp. 5:1–5:26, November 2012 2012.
- [115] B. Gedik, L. Liu, and P. S. Yu, "ASAP: An Adaptive Sampling Approach to Data Collection in Sensor Networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 18, no. 12, pp. 1766-1783, December 2007 2007.
- [116] L. Sun, W. Cai, and X. Huang, "Data aggregation scheme using neural networks in wireless sensor networks," in *2010 2nd International Conference on Future Computer and Communication*, 2010, vol. 1, pp. V1-725-V1-729.
- [117] W. Bo, H. Han-ying, and F. Wen, "A Pseudo LEACH Algorithm for Wireless Sensor Networks," 2007, pp. 1366-1370.
- [118] C. Liu, K. Wu, and J. Pei, "An Energy-Efficient Data Collection Framework for Wireless Sensor Networks by Exploiting Spatiotemporal Correlation," *IEEE Transactions on Parallel and Distributed Systems*, vol. 18, no. 7, pp. 1010-1023, July 2007 2007.
- [119] L. A. Villas, A. Boukerche, D. L. Guidoni, H. A. B. F. de Oliveira, R. B. de Araujo, and A. A. F. Loureiro, "An energy-aware spatio-temporal correlation mechanism to perform efficient data collection in wireless sensor networks," *Computer Communications*, vol. 36, no. 9, pp. 1054-1066, May 15, 2013 2013.
- [120] G. Li and Y. Wang, "Automatic ARIMA modeling-based data aggregation scheme in wireless sensor networks," *EURASIP Journal on Wireless Communications and Networking*, vol. 2013, no. 1, p. 85, March 25, 2013 2013.
- [121] S. Santini and K. Römer, "An Adaptive Strategy for Quality-Based Data Reduction in Wireless Sensor Networks," 2006, pp. 29–36.
- [122] K. Khedo, R. Doomun, and S. Aucharuz, "READA: Redundancy Elimination for Accurate Data Aggregation in Wireless Sensor Networks," (in en), *Wireless Sensor Network*, vol. 02, p. 300, 2010-04-23 2010.
- [123] S. Ozdemir and Y. Xiao, "Polynomial Regression Based Secure Data Aggregation for Wireless Sensor Networks," in *2011 IEEE Global Telecommunications Conference - GLOBECOM 2011*, 2011, pp. 1-5.
- [124] I. F. Akyildiz, M. C. Vuran, and Ö. B. Akan, "On exploiting spatial and temporal correlation in wireless sensor networks," 2004, pp. 71–80.
- [125] S. Pattem, B. Krishnamachari, and R. Govindan, "The impact of spatial correlation on routing with compression in wireless sensor networks," in *Third International Symposium on Information Processing in Sensor Networks, 2004. IPSN 2004*, 2004, pp. 28-35.
- [126] M. A. Sharaf, J. Beaver, A. Labrinidis, and P. K. Chrysanthis, "TiNA: A Scheme for Temporal Coherency-Aware in-Network Aggregation," (in en), p. 8.
- [127] « Théorème de Stone-Weierstrass », théorème de l'approximation polynomiale, [Page web], Disponible sur: <http://www.les-mathematiques.net/phorum/read.php?17,741365,816339>, [Consulté le: 30-juin-2019].

-
- [128] M. H. Stone, "The Generalized Weierstrass Approximation Theorem," *Mathematics Magazine*, vol. 21, no. 5, pp. 237-254, 1948 1948.
- [129] A. Pinkus, "Weierstrass and Approximation Theory," *Journal of Approximation Theory*, vol. 107, no. 1, pp. 1-66, November 1, 2000 2000.
- [130] K. Molugaram and G. S. Rao, "Chapter 6 - Correlation and Regression," in *Statistical Techniques for Transportation Engineering*, K. Molugaram and G. S. Rao, Eds.: Butterworth-Heinemann, 2017, pp. 293-329.
- [131] Data set of « C-MAN station LONF1- Long Key, FLORIDA, USA », *National Data Buoy Center*, [Page web], Disponible sur: https://www.ndbc.noaa.gov/station_page.php?station=lonf1, [Consulté le: 30-juin-2019].
- [132] Spécifications techniques du capteur DHT22, « Capteurs de température et d'humidité DHT22 », Disponible sur : www.didel.com/OledDHT22.pdf, [Consulté le: 30-juin-2019].

Résumé

Aujourd'hui, la vie quotidienne des personnes, des animaux et des biens de la planète en général dépend fortement des résultats issus par des capteurs à proximité. L'utilisation de ces petits dispositifs dans les domaines à criticité élevée présente une avancée généreuse et efficace envers la société. La surveillance des feux de forêt par des réseaux de capteurs multimédias sans fil (RCMSF) fait l'objet de cette thèse. Nous déployons dans une région forestière un ensemble de nœuds capteurs fonctionnant en deux étapes pour prévenir et détecter le déclenchement d'un incendie. La première étape consiste à calculer le danger potentiel d'inflammation de la forêt par des grandeurs scalaires telles que la température et l'humidité. La deuxième étape, permet de consolider l'alerte scalaire déclenchée par une source riche en information - l'image - afin d'en s'assurer de la fiabilité de la détection, et d'en distinguer parmi les alertes celles qui sont vraies. Nous intégrons dans notre solution des techniques à économie d'énergie afin de maintenir une plus longue durée de surveillance de la région d'intérêt.

Mots-clés:

RCMSF; Détection des incendies de forêt ; Capteurs; Compression; Economie d'énergie; Image; Surveillance; Alerte; Température; Humidité